Core1553BRT v4.1

Handbook

Microsemi

Table of Contents

Introduction

Core Overview

Core1553BRT provides a complete, dual-redundant MIL-STD-1553B remote terminal (RT), apart from the transceivers required to interface to the bus. A typical system implementation using Core1553BRT is shown in Figure 1 and Figure 2 on page 5.

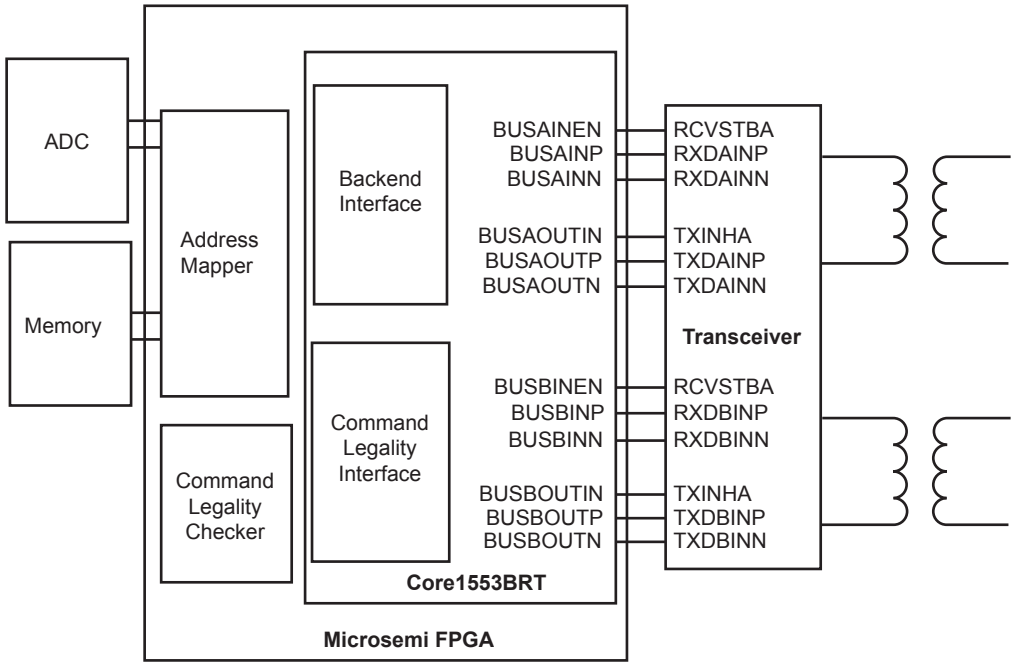


***Figure 1* • Typical Core1553BRT System**

At a high level, Core1553BRT simply provides a set of memory-mapped subaddresses that “receive data written to” or “transmit data read from.” The core can be configured to connect directly to synchronous or asynchronous memory devices. Alternatively, the core can directly connect to backend devices, removing the need for memory buffers. If memory is used, the core requires 2,048 words of memory, which can be shared with the local CPU.

The core supports all 1553B mode codes and allows the user to designate as illegal any mode code or any particular subaddress for both transmit and receive operations. The command legalization can be done within the core or in an external command legality block via the command legalization interface.

The core consists of six main blocks: 1553B encoders, 1553B decoders, the backend interface, a command decoder, RT controller blocks, and a command legalization block (Figure 2).

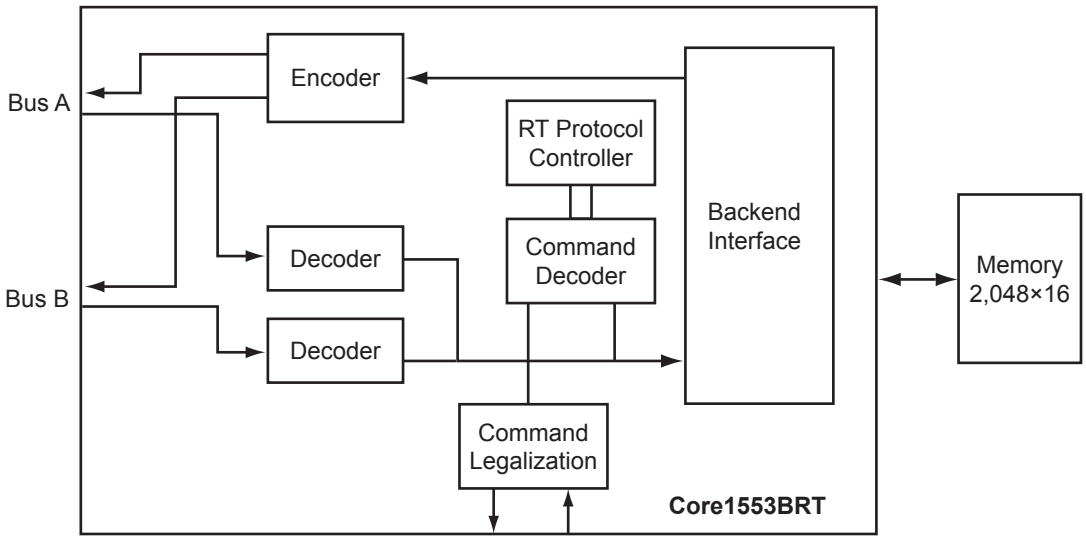


***Figure 2* • Core1553BRT RT Block Diagram**

In Core1553BRT, a single 1553B encoder is used. This takes each word to be transmitted and serializes it, after which the signal is Manchester-encoded. The encoder also includes logic to prevent the RT from transmitting for longer than the allowed period, and loopback fail logic. 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 RT should use to transmit.

The core includes two 1553B decoders. A decoder takes the serial Manchester data received from the bus and extracts the received data words. A decoder requires a 12, 16, 20, or 24 MHz clock to extract the data and the clock from the serial stream.

The decoder contains a digital 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 or data word is received and also performs Manchester encoding and parity error checking.

The backend interface for Core1553BRT allows a simple connection to a memory device or direct connection to other devices, such as analog to digital converters. The access rates to this memory are slow, with one read or write every 20 µs. At 12 MHz operation, this is one read or write every 240 clock cycles.

The backend interface can be configured to connect to either synchronous or asynchronous memory devices. This allows the core to be connected to synchronous logic, memory within the FPGA, or external asynchronous memory.

The core implements a simple subaddress to the memory address mapping function, allowing the core to be directly connected to a memory block. The core also supports an address mapping function that allows the backend memory map to be modified to emulate legacy 1553B remote terminals, therefore minimizing system and software changes when adopting Core1553BRT. Associated with this function is the ability to create a user-specific interrupt vector.

The backend interface supports a standard bus request and grant protocol and provides a WAIT input to allow the core to interface to slow memory devices.

The command decoder and RT controller blocks decode the incoming command words, verifying their legality. Then, the protocol state machine responds to the command, transmitting or receiving data or processing a mode code.

Core1553BRT has an internal command legality block that verifies every 1553B command word. A separate interface is provided that, when enabled, allows the command legality decoder to be implemented outside Core1553BRT. This external interface is intended for use with Obfuscated versions of the core. For the RTL version of the core, this interface can be used, or the source code can be easily modified to implement this function.

The external BIST interface is used to configure the external transmit bit word or internal BIST word. The external BIST is configured when EXTERNAL_BIST parameter/generic is set and external BIST enable is set.

Verification and Compliance

Core1553BRT functionality has been verified in simulation and hardware. Full functional verification against the RT test plan, as defined in MIL-HDBK-1553A, has been carried out using a VHDL simulation environment.

To fully verify compliance, the core has been implemented on an M2S050FG484 part connected to external transceivers and memory. Test Systems Inc. has verified Core1553BRT against the remote terminal test plan in accordance with the RT validation test plan MIL-HDBK-1553A, Appendix A.

Fail-Safe State Machines

The logic design of Core1553BRT implements fail-safe state machines. All state machines include illegal state detection logic. If a state machine should ever enter an illegal state, the core will assert its FSM_ERROR output and the state machine will reset. If this occurs, Microsemi recommends that the external system reset the core and also assert the TFLAG input to inform the bus controller (BC) that a serious error has occurred within the remote terminal.

The FSM_ERROR output can be left unconnected if the system is not required to detect and report state machines entering illegal states.

Core Version

This handbook applies to Core1553BRT v4.1 and later.

Supported Families

- IGLOO®
- IGLOOe
- IGLOOPLUS
- ProASIC®3
- ProASIC3L
- ProASIC3E
- SmartFusion®
- SmartFusion2
- Fusion
- ProASICPLUS®
- Axcelerator®
- RTAX-S
- SX-A
- RTSX-S
- IGLOO®2
- RTG4™

Device Requirements

Core1553BRT can be implemented in several Microsemi FPGA devices. Table 1 gives the utilization and performance figures for the core implemented in these devices.

The core can operate with a clock of up to 24 MHz. This clock rate is easily met in all Microsemi silicon families noted in Table 1.

***Table 1* • Device Utilization and Performance**

Family	Combinatorial	Sequential	Total	Device	Utilization	Performance
ProASIC3	1066	438	1519	A3P600	11.0%	85.543
ProASIC3E	1066	438	1519	A3PE600	11.0%	82.42
IGLOO	1066	438	1519	AGL600V5	11.0%	78.223
IGLOOe	1066	438	1519	AGLE600V5	11.0%	77.304
Fusion	1066	438	1519	AFS600	11.0%	87.367
IGLOO2	756	457	1213	M2GL050T	2.2%	142.349
SmartFusion2	756	457	1213	M2S050T	2.2%	142.369
SmartFusion	1026	404	1430	A2F500M3G	12.4%	92.106
SX-A	750	451	1201	A54SX72A	19.9%	52.089
RTSX-S	742	450	1192	RT54SX72S	19.8%	46.618
Axcelerator	754	440	1194	AX500	14.8%	104.037
RTAX-S	754	440	1194	RTAX1000S	6.6%	72.611
ProASIC^{PLUS}	1404	464	1868	APA450	15.2%	68.357
RTG4	872	426	1298	RT4G150	0.85%	108.9

Utilization data was generated using standard Libero® System-on-Chip (SoC) or Integrated Design Environment (IDE) tool flows with typical core parameter settings. Utilization data will vary slightly with different parameter settings and tool usage.

1 – MIL-STD-1553B Bus Overview

The MIL-STD-1553B bus is a differential serial bus used in military and space equipment. It comprises multiple redundant bus connections and communicates at 1 MB/s.

The bus has a single active BC and up to 31 RTs. The BC manages all data transfers on the bus using the command and status protocol. The bus controller initiates every transfer by sending a command word and data if required. The selected RT will respond with a status word and data if required.

The 1553B command word contains a 5-bit RT address, transmit or receive bit, 5-bit subaddress, and 5-bit word count. This allows for 32 RTs on the bus. However, since RT address 31 is used to indicate a broadcast transfer, only 31 RTs can be connected. Each RT has 30 subaddresses reserved for data transfers. The other two subaddresses (0 and 31) are reserved for mode codes used for bus control functions. Data transfers contain up to thirty-two 16-bit data words. Mode code command words are used for bus control functions such as synchronization.

Message Types

The 1553B bus supports 10 message transfer types, allowing basic point-to-point and broadcast BC-to-RT data transfers, mode code messages, and direct RT-to-RT messages. Figure 1-1 shows the message formats.

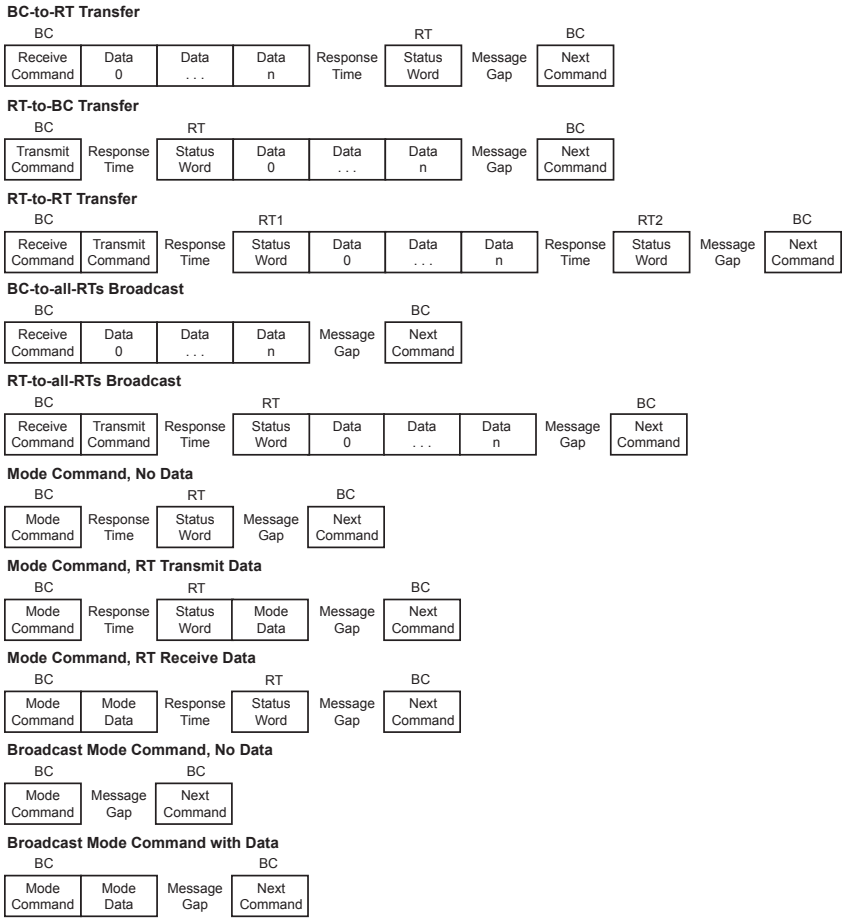


***Figure 1-1* • 1553B Message Formats**

Word Formats

There are only three types of words in a 1553B message: a command word (CW), a data word (DW), and a status word (SW). Each word consists of a 3-bit sync pattern, 16 bits of data, and a parity bit, providing the 20-bit word (Figure 1-2).

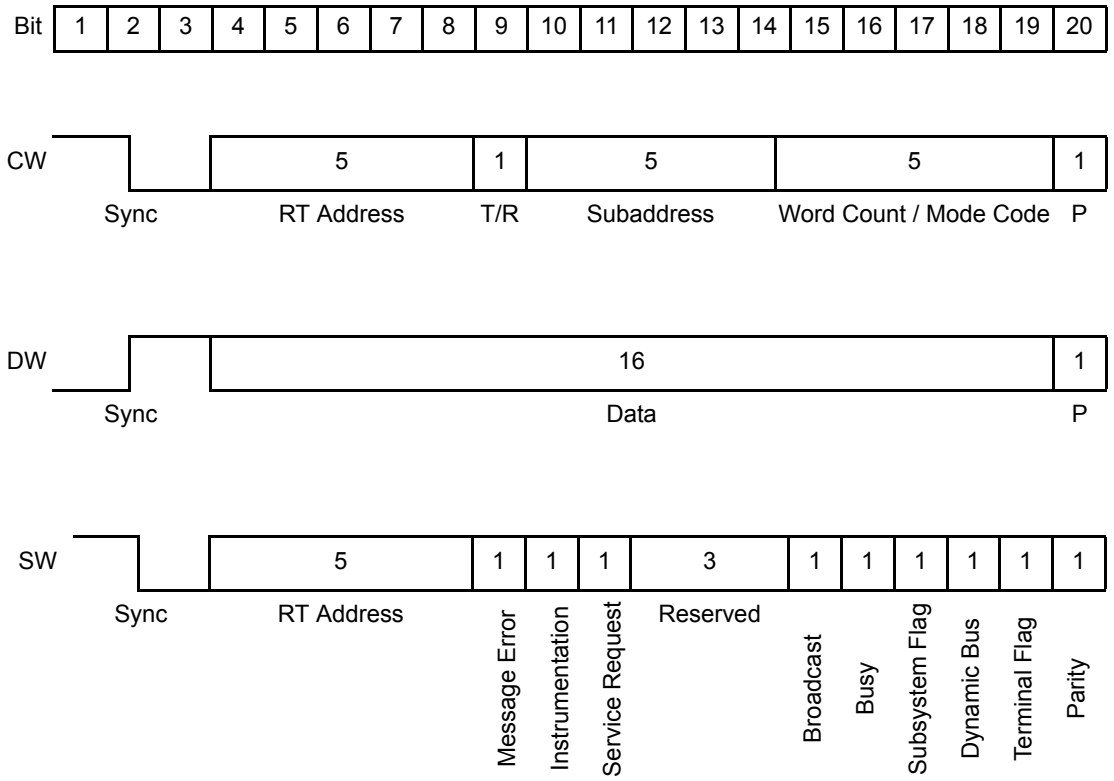


***Figure 1-2* • 1553B Word Formats**

2 – Tool Flows

SmartDesign

Core1553BRT is available for download to the SmartDesign IP Catalog, via the Libero IDE/SoC web repository. For information on using SmartDesign to instantiate, configure, connect, and generate cores, please refer to the Libero IDE/SoC online help.

The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 2-1.

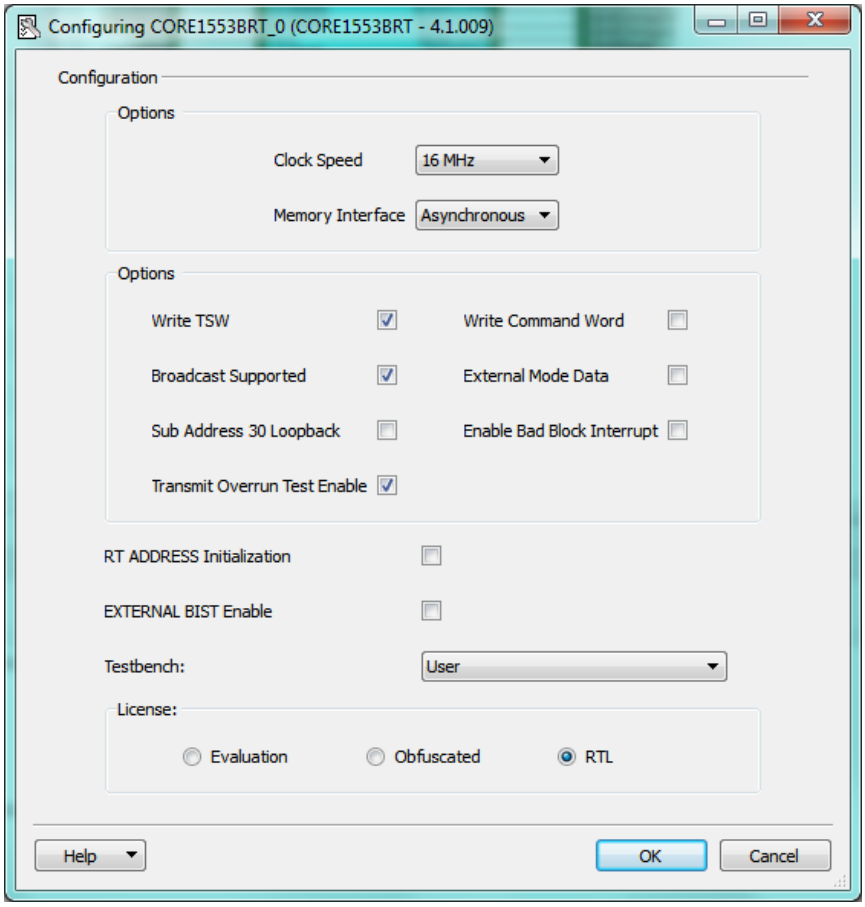


***Figure 2-1* • Core1553BRT Configuration within SmartDesign**

Simulation Flows

To run simulations, the required testbench flow must be selected within SmartDesign. The required testbench is selected through the core configuration GUI. The following simulation environments are supported:

- Full 1553 verification environment (VHDL only)
- Simple testbench (VHDL and Verilog)

When SmartDesign generates the core, it will install the appropriate testbench files. To run the testbenches, simply set the design root to the Core1553BRT instantiation in the Libero IDE/SoC file manager and click the **Simulation** icon in Libero IDE/SoC. This will invoke Model*Sim*® and automatically run the simulation.

3 – Interface Descriptions

Parameters on Core1553BRT

The parameters given in Table 3-1 are used to configure the core.

***Table 3-1* • Core1553BRT Parameters**

Parameter	Range	Description
FAMILY	2 to 24	Must be set to the required FPGA family: 8: SX-A 9: RTSX-S 11: Axcelerator 12: RTAX-S 14: ProASIC^{PLUS} 15: ProASIC3 16: ProASIC3E 17: Fusion 18: SmartFusion 19: SmartFusion2 20: IGLOO 21: IGLOOe 22: ProASIC3L 23: IGLOO PLUS 24: IGLOO2 25: RTG4
CLKSPD	12, 16, 20, or 24	Sets the clock frequency of the core to 12, 16, 20, or 24 MHz
WRTTSW	0 or 1	When 1, the core will write the transfer status word (TSW) to the memory. When 0, the core disables the writing of the transfer status word to memory. This is useful for simple RT applications that do not use memory but have a direct connection to the backend device.
WRTCMD	0 or 1	When 1, the core will write the 1553B command word to the locations used for the TSW values. If WRTTSW is also enabled, the command word is written to memory at the start of a message, and the TSW value will overwrite the command word at the end of the message, unless an external address mapping function is used.
EXTMDATA	0 or 1	When 1, the core reads and writes mode code data words from and to the external memory (except for “transmit last command” and “transmit BIT [Built-In Test] word”). The VWORD input is not used when this input is active.
BCASTEN	0 or 1	This input enables broadcast operation. When 1, broadcast operations are enabled. When 0, broadcast messages (i.e., RT address 31) are treated as normal messages. If the RTADDR input is set to 31, the RT will respond to the message.

***Table 3-1* • Core1553BRT Parameters (continued)**

Parameter	Range	Description
SA30LOOP	0 or 1	This input alters the backend memory mapping so that subaddress 30 provides automatic loopback. When 0, the RT does not loop back subaddress 30. Separate memory buffers are used for transmit and receive data buffers. When 1, the RT maps the transmit memory buffer for subaddress 30 to the receive memory buffer for subaddress 30; i.e., the upper address line is forced to 0.
ASYNCIF	0 or 1	When 1, the backend interface is in asynchronous mode. When 0, the backend interface is in synchronous mode.
INTENBBR	0 or 1	When active (1), the core generates interrupts when both good and bad 1553B messages are received. When inactive (0), the core only generates interrupts when good messages are received.
TESTTXTOUTEN	0 or 1	This enables the TESTTXTOUT input; it is for test use only. This parameter should be set to 0 if it is not required to be able to force transmission overrun for testing the internal transmit timer.
INITLASTSW	0 or 1	This input controls the last status word. When 0, the first received command is a transmit last status word. The core will respond with an undefined status word since no status word has previously been sent (same function as previous core versions.) When 1, the first received command is a transmit last status word. The core will respond with a valid RT address and all other status bits zero, even though no status word was previously sent. It requires PURSTN to be asserted at power-up. The default value of INITLASTSW is 0.
EXTERNAL_BIST	0 or 1	This parameter controls the mode code 19 support. When 0, the internal BIST value as specified in Table 5-6 on page 32 is returned in response to the Transmit BIST mode code. When 1, the input BITIN [15:0] is returned in response to the Transmit BIST mode code. The default value of EXTERNAL_BIST is 0.

I/O Signal Descriptions

Table 3-2 lists the signals for the 1553B bus interface. Table 3-3 on page 14 lists the control and status signals.

Double flip-flop metastability synchronizers are implemented on the following inputs: RTADDR[4:0], RTADDRP, BUSAINP, BUSAINN, BUSBINP, and BUSBINN.

***Table 3-2* • 1553B Bus Interface**

Port Name	Type	Description
RTADDR[4:0]	**In**	**Sets the RT address; RT address can be set to '11111' for normal operation only when BCASTEN is set to 0.**
RTADDRP	**In**	**RT address parity input. This input should be set HIGH or LOW to achieve odd parity on the RTADDR and RTADDRP inputs. If RTADDR is '00000', the RTADDRP input should be 1.**
RTADERR	Out	Indicates that the RTADDR and RTADDRP inputs have incorrect parity, or broadcast is enabled, and the RT address is set to 31. When active (HIGH), the RT is disabled and will ignore all 1553B traffic.
BUSAINEN	Out	Active high output that enables the A receiver
BUSAINP	**In**	**Positive data input from the A receiver**
BUSAINN	**In**	**Negative data input from the A receiver**
BUSBINEN	Out	Active high output that enables the B receiver
BUSBINP	**In**	**Positive data input from the bus to the B receiver**
BUSBINN	**In**	**Negative data input from the bus to the B receiver**
BUSAOUTIN	Out	Active high transmitter inhibit for the A transmitter
BUSAOUTP	Out	Positive data output to the bus A transmitter (held HIGH when no transmission)
BUSAOUTN	Out	Negative data output to the bus A transmitter (held HIGH when no transmission)
BUSBOUTIN	Out	Active high transmitter inhibits the B transmitter
BUSBOUTP	Out	Positive data output to the bus B transmitter (held HIGH when no transmission)
BUSBOUTN	Out	Negative data output to the bus B transmitter (held HIGH when no transmission)

***Table 3-3* • Control and Status Signals**

Port Name	Type	Description
CLK	In	Master clock input (12, 16, 20, or 24 MHz)
RSTn	In	Reset input asynchronous (active low)
SREQUEST	In	Directly controls the Service Request bit in the 1553B status word
RTBUSY	In	Directly controls the Busy bit in the 1553B status word
SSFLAG	In	Directly controls the Subsystem Flag bit in the 1553B status word
TFLAG	In	Controls the Terminal Flag bit in the 1553B status word. This can be masked by the "inhibit terminal flag bit" mode code.
VWORD[15:0]	In	Provides the 16-bit vector value for the "transmit vector word" mode command
BUSY	Out	Indicates that the 1553BRT is either receiving or transmitting data or handling a mode command

Note: All control inputs except RSTn are synchronous and sampled on the rising edge of the clock. All status outputs are synchronous to the rising edge of the clock.

***Table 3-3* • Control and Status Signals (continued)**

Port Name	Type	Description
CMDSYNC	Out	Pulses HIGH for a single clock cycle when the RT detects the start of a 1553B command word (or status word) on the bus. Provides an early signal that the RT may be about to receive or transmit data or mode code.
MSGSTART	Out	Pulses HIGH for a single cycle when the RT is about to start processing a 1553B message whose command has been validated for this RT.
SYNCNOW	Out	Pulses HIGH for a single clock cycle when the RT receives a “synchronize” (with or without data mode) command. The pulse occurs just after the 1553B command word (sync with no data) or data word (sync with data mode code) has been received.
BUSRESET	Out	Pulses HIGH for a single clock cycle whenever the RT receives a “reset mode” command. The core logic will also automatically reset itself on receipt of this command.
INTOUT	Out	Goes HIGH when data has been received or transmitted or a mode command processed. The reason for the interrupt is provided on INTVECT. This output will stay HIGH until INTACK goes HIGH. If INTACK is held HIGH, this output will pulse HIGH for a single clock cycle.
INTVECT[6:0]	Out	This 7-bit value contains the reason for the interrupt. It indicates which subaddress data has been received or transmitted. Bit 6:0: Bad block received1: Good block received Bit 5:0: RX data1: TX data Bits 4:0:Subaddress Further information can be found by checking the appropriate transfer status word for the appropriate subaddress.
INTACK	In	Interrupt acknowledge input. When HIGH, this resets INTOUT back to LOW. If this input is held HIGH, the INTOUT signal will pulse HIGH for one clock cycle every time an interrupt is generated.
MEMFAIL	Out	Goes HIGH if the core fails to read data from or write data to the backend interface within the required time. This can be caused by the backend not asserting MEMGNTn fast enough or asserting MEMWAITn for too long.
CLRERR	In	Used to clear MEMFAIL and other internal error conditions. Must be held HIGH for more than two clock cycles.
TESTTXTOUT	In	This input is for test use only. It should be tied LOW. When HIGH and the TESTTXTOUTEN parameter is set to 1, the RT will transmit more than 32 data words if a “transmit data” command word is received. This will cause the RT to shut down the transmitter and set the TIMEOUT bits in the BIT word.
FSM_ERROR	Out	This output will go HIGH for a single clock cycle if any of the internal state machines enter an illegal state. This output should not go HIGH in normal operation. Should it go HIGH, it is recommended that the core be reset.
PURSTN	In	Asynchronous power-up reset input (active low) that is used to initialize the last status word value. This input is valid only when the parameter INITLASTSW = 1.
BITINEN	In	Transmit bit word enable input (active high). This input is valid when parameter EXTERNAL_BIST = 1 (to support mode code 19).
BITIN[15:0]	In	Transmit bit word input. This input is valid when the parameter EXTERNAL_BIST = 1.

Note: All control inputs except RSTn are synchronous and sampled on the rising edge of the clock. All status outputs are synchronous to the rising edge of the clock.

Command Legalization Interface

The core checks the validity of all 1553B command words. In RTL and Obfuscated versions of the core, the logic may be implemented externally to the core. The command word is provided, and the logic must generate the command-valid input. The command legalization interface also provides two strobes that are used to latch the command value to enable it to be used for address mapping and interrupt vector extension functions (Table 3-4).

Table 3-4 • **Command Legalization Interface**

Port Name	Type	Description
USEEXTOK	In	When 0, the core uses its own internal command-valid logic, enabling all legal, supported mode codes and all subaddresses. When 1, the core disables its internal logic and uses the external CMDOK input for command legality.
CMDVAL[11:0]	Out	ActiveCommand 11:0: Non-broadcast 1: Broadcast 10:0: Receive 1: Transmit 9:5:Subaddress 4:0:Word count / mode code These outputs are valid throughout the complete 1553B message. They can also be used to steer data to particular backend devices. In particular, bit 11 allows non-broadcast and broadcast messages to be differentiated, as required by MIL-STD-1553B, Notice 2.
CMDSTB	Out	Single-clock-cycle pulse that indicates valid command is received on CMDVAL.
CMDOK	In	Command word is okay (active high). The external logic must set this within 3 µs from the CMDVAL output changing.
CMDOKOUT	Out	Command word is okay (output). When USEEXTOK = 0, the core puts out its “internal command word okay” validation signal.
ADDRLAT	Out	CMDVAL address latch enable output (active high). Used to latch CMDVAL when it is being used for an address mapping function. ADDRLAT should be connected to the enable of a rising-edge clock flip-flop.
INTLAT	Out	CMDVAL interrupt vector latch enable output (active high). Used to latch CMDVAL when it is being used for an extended interrupt vector function. INTLAT should be connected to the enable of a rising-edge clock flip-flop.

Backend Interface

The backend interface supports both synchronous operation (to the core clock) and asynchronous operation to backend devices (Table 3-5).

Table 3-5 • **Backend Signals**

Port Name	Type	Description
MEMREQn	Out	Memory Request (active low) output. The backend interface requires memory access completion within 10 µs of MEMREQ going LOW to avoid data loss or overrun on the 1553B interface.*
MEMGNTn	In	Memory Grant (active low) input. This input should be synchronous to CLK and needs to meet the internal register setup time. This input can be held LOW if the core has continuous access to the RAM.
MEMWRn	Out	Memory Write (active low) Synchronous mode: This output indicates that data is to be written on the rising clock edge. Asynchronous mode: This output will be LOW for a minimum of one clock period and can be extended by the MEMWAITn input. The address and data are valid one clock cycle before MEMWRn is active and held for one clock cycle after MEMWRn goes inactive.
MEMRDn	Out	Memory Read (active low) Synchronous mode: This output indicates that data will be read on the next rising clock edge. This signal is intended as the read signal for synchronous RAMs. Asynchronous mode: This output will be LOW for a minimum of one clock period and can be extended by the MEMWAITn input. The address is valid one clock cycle before MEMRDn is active and held for one clock cycle after MEMRDn goes inactive. The data is sampled as MEMRDn goes HIGH.
MEMCSn	Out	Memory Chip Select (active low). This output has the same timing as MEMADDR.
MEMWAITn	In	Memory Wait (active low) Synchronous mode: This input is not used; it should be tied HIGH. Asynchronous mode: Indicates that the backend is not ready, and the core should extend the read or write strobe period. This input should be synchronous to CLK and needs to meet the internal register setup time. It can be permanently held HIGH.
MEMOPER[1:0]	Out	Indicates the type of memory access being performed. 00: Data transfer for both data and mode code transfers 01: TSW 10: Command word 11: Not used
MEMADDR[10:0]	Out	Memory Address output (the subaddress mapping is covered in "Standard Memory Address Map" on page 18)
MEMDOUT[15:0]	Out	Memory Data output
MEMDIN[15:0]	In	Memory Data input

*Note: *The 10 µs refers to the time from MEMREQn being asserted to the core deasserting its MEMREQn signal. The core has an internal overhead of five clock cycles, and any inserted wait cycles will also reduce this time. This time increases to 19.5 µs if the WRTTSW and WRTCMD inputs are LOW.*

***Table 3-5* • Backend Signals (continued)**

Port Name	Type	Description
MEMCEN	Out	Control Signal Enable (active high). This signal is HIGH when the core is requesting the memory bus and has been granted control. It is intended to enable any tristate drivers that may be implemented on the memory control and address lines.
MEMDEN	Out	Data Bus Enable (active high). This signal is HIGH when the core is requesting the memory bus, has been granted control, and is waiting to write data. It is intended to enable any bidirectional drivers that may be implemented on the memory data bus.

*Note: *The 10 µs refers to the time from MEMREQn being asserted to the core deasserting its MEMREQn signal. The core has an internal overhead of five clock cycles, and any inserted wait cycles will also reduce this time. This time increases to 19.5 µs if the WRTTSW and WRTCMD inputs are LOW.*

Standard Memory Address Map

Core1553BRT requires an external 2,048×16 memory device. This memory is split into sixty-four 32-word data buffers. Each of the 30 subaddresses has a receive and a transmit buffer, as shown in Table 3-6.

The memory allocated to the unused receive subaddresses 0 and 31 is used to provide status information back to the rest of the system. At the end of every transfer, a TSW is written to these locations.

***Table 3-6* • Standard Memory Address Map**

Address	RAM Contents	Action
000–01F	RX transfer status words	The core only writes to these addresses (except when SA30LOOP is HIGH).
020–03F	Receive subaddress 1	
...	...	
3C0–3DF	Receive subaddress 30	
3E0–3FF	TX transfer status words	
400–41F	Not used	The core only reads from these addresses.
420–43F	TX transfer subaddress 1	
...	...	
7C0–7DF	TX transfer subaddress 30	
7E0–7FF	Not used	

4 – Interface Timing

Specifications

Memory Write Timing – Asynchronous Mode

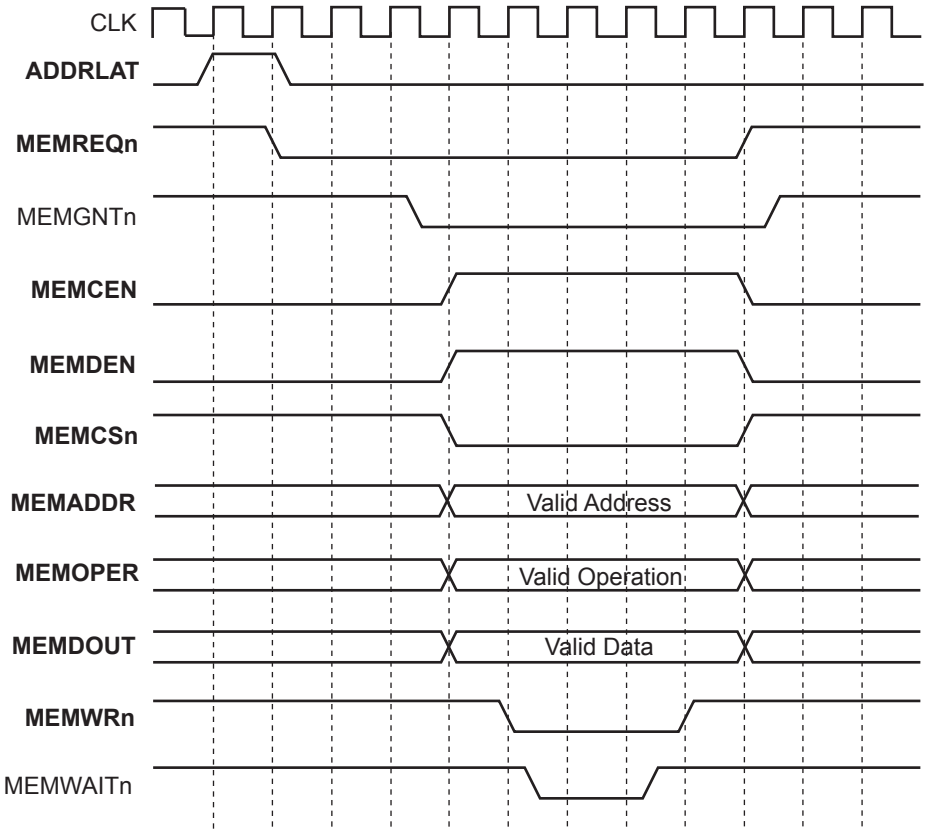


***Figure 4-1* • Memory Write Timing – Asynchronous Mode**

***Table 4-1* • Memory Write Timing**

Parameter	Description	Time
T_{pwWR}	Write pulse width (no wait states)	1 clock cycle
T_{pdGNT}	Maximum delay from MEMREQn to MEMGNTn active	12.0 µs
T_{suDATA}	Data setup time to MEMWRn LOW	1 clock cycle
T_{suADDR}	Address setup time to MEMWRn LOW	1 clock cycle
T_{hdDATA}	Data hold time from MEMWRn HIGH	1 clock cycle
T_{hdADDR}	Address hold time from MEMWRn HIGH	1 clock cycle
T_{suWAIT}	Wait setup to rising clock edge	1 clock cycle

Memory Read Timing – Asynchronous Mode

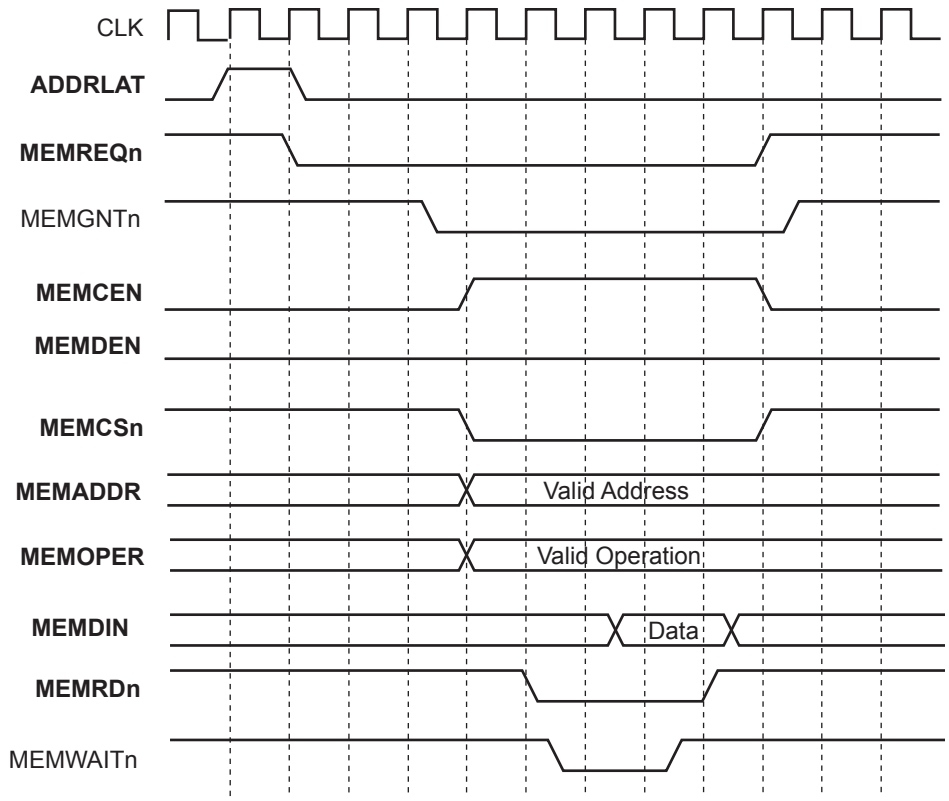


***Figure 4-2* • Memory Read Timing**

***Table 4-2* • Memory Read Timing**

Parameter	Description	Time
T_{pwRD}	Read pulse width (no wait states)	1 clock cycle
T_{pdGNT}	Maximum delay from MEMREQn to MEMGNTn active	12.0 µs
T_{suADDR}	Address setup time to MEMRDn LOW	1 clock cycle
T_{hdADDR}	Address hold time from MEMRDn HIGH	1 clock cycle
T_{suWAIT}	Wait setup to rising clock edge	15.0 ns
T_{suDATA}	Data setup time to MEMRDn HIGH	15.0 ns

Memory Write Timing – Synchronous Mode

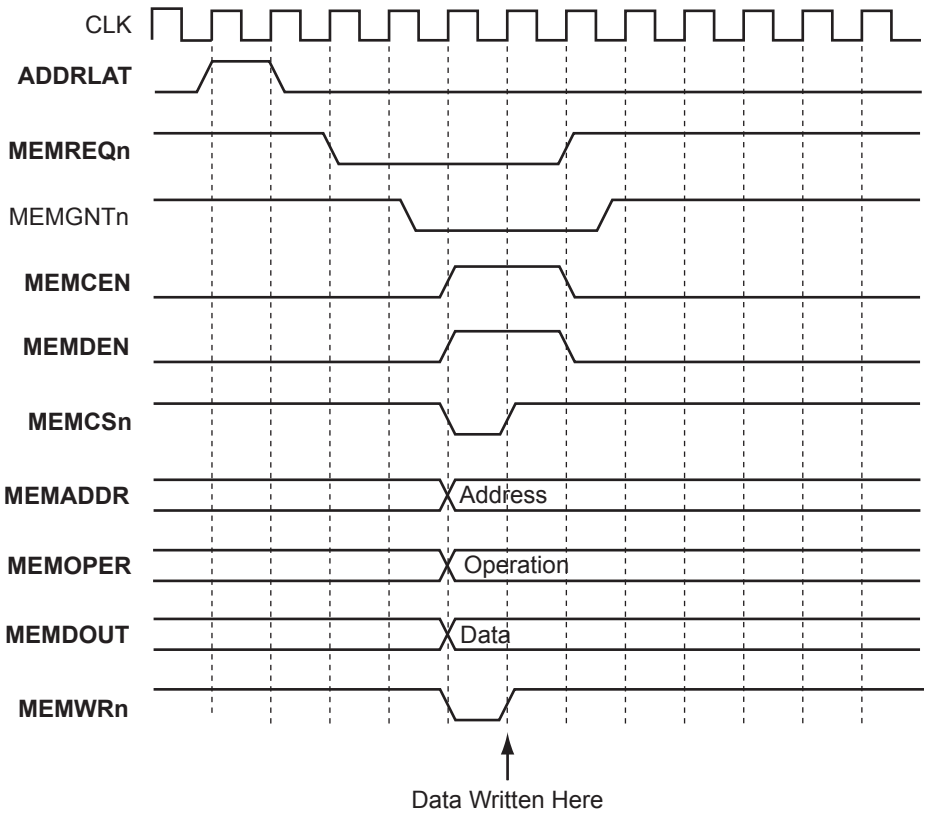


Figure 4-3 **• Memory Write Timing**

Memory Read Timing – Synchronous Mode

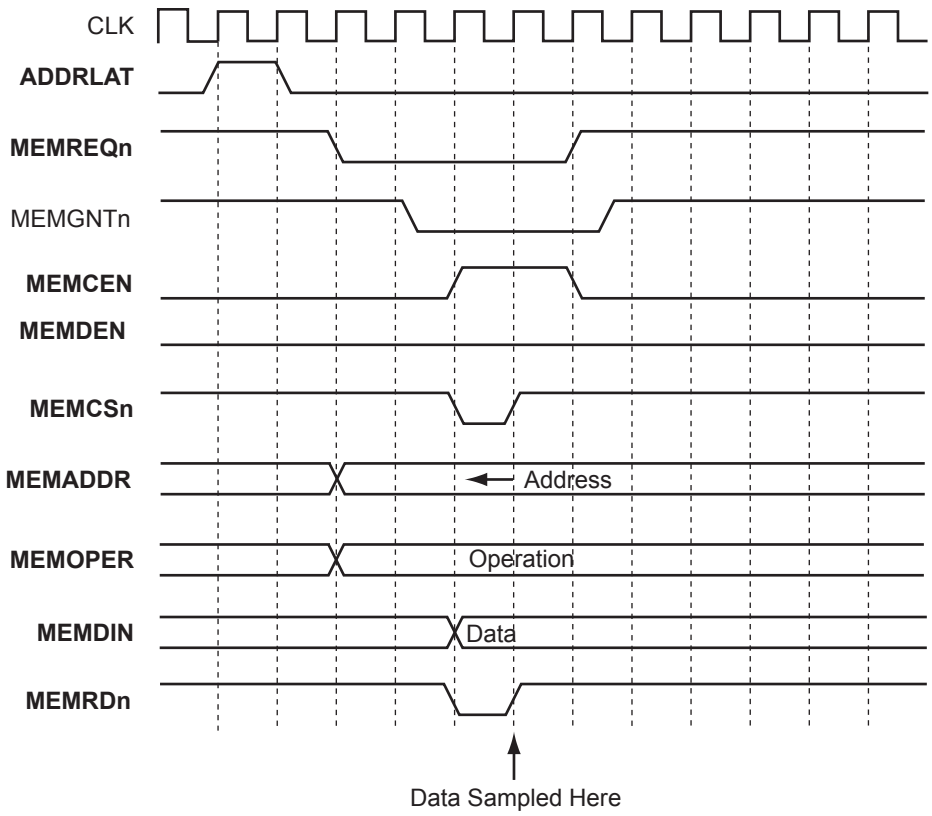


***Figure 4-4* • Memory Read Timing**

Command Word Legality Interface Timing

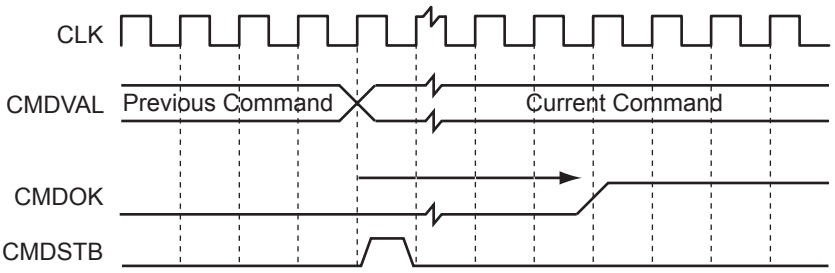
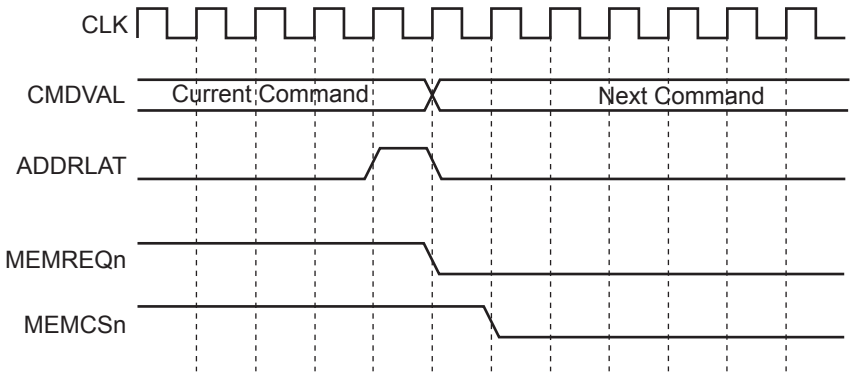


***Figure 4-5* • Command Word Legality Interface Timing**

***Table 4-3* • Command Word Legality Interface Timing**

Name	Description	Time
$T_{pdCMDOK}$	Maximum external command word legality decode delay	3 µs

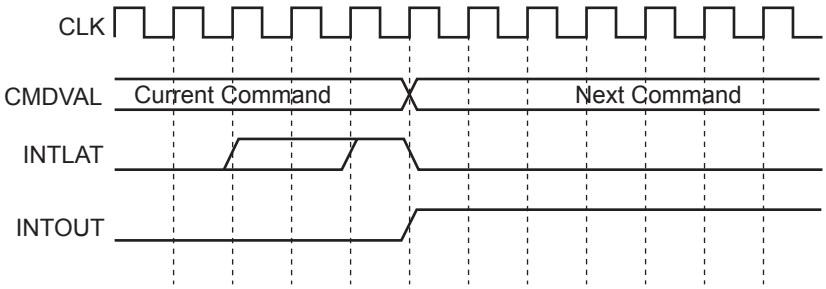
Address Mapper Timing



Note: This figure shows the worst-case timing when a second 1553B command arrives as the core starts a backend transfer and MEMGNTn is held LOW.

***Figure 4-6* • Address Mapper Timing**

Interrupt Vector Extender Timing



Note: This figure shows the worst-case timing when a second 1553B command arrives as the core asserts an interrupt request. Also, INTLAT may be active for several clock cycles prior to INTOUT.

***Figure 4-7* • Interrupt Vector Extender Timing**

RT Response Times

RT response time is from the midpoint of the parity bit in the command word to the midpoint of the status word sync (Table 4-4).

***Table 4-4* • RT Response Times**

Spec	Description	At 12 MHz	At 16 MHz	At 20 MHz	At 24 MHz
T_{rtresp}	RT response time	4.75 to 7.0 µs	4.75 to 7.0 µs	4.75 to 7.0 µs	4.75 to 7.0 µs
T_{rtrtto}	RT-to-RT timeout	57 µs	57 µs	57 µs	57 µs
T_{xxto}	Transmitter timeout	704 µs	668 µs	691 µs	693 µs

The RT-to-RT timeout is from the first command word parity bit to the expected sync of the first data word.

Transceiver Loopback Delays

Core1553BRT verifies that all transmitted data words are correctly transmitted. As data is transmitted by the transceiver on the 1553B bus, the data on the bus is monitored by the transceiver and decoded by Core1553BRT. The core requires that the loopback delay, i.e., the time from BUSAOUTP to BUSAINP, be less than the values given in the Table 4-5.

***Table 4-5* • Transceiver Loopback Requirements**

Clock Speed	Maximum Loopback Delay
12 MHz	2.3 µs
16 MHz	2.3 µs
20 MHz	2.3 µs
24 MHz	2.3 µs

The loopback delay is a function of the internal FPGA delay, PCB routing delays, and internal transceiver delay as well as transmission effects from the 1553B bus. Additional register stages can be inserted on either the 1553B data input or output within the FPGA, providing the loopback delays in Table 4-5 are not violated. This is recommended if additional gating logic is inserted inside the FPGA between the core and transceiver to minimize skew between the differential inputs and outputs.

Clock Requirements

To meet the 1553B transmission bit rate requirements, the Core1553BRT clock input must be 12 MHz, 16 MHz, 20 MHz, or 24 MHz 0.1% (+/- 1000 Hz) long term and 0.01% (+/- 100 Hz) short term.

5 – Operation

Standard Memory Address Map

Core1553BRT requires an external 2,048×16 memory device. This memory is split into sixty-four 32-word data buffers. Each of the 30 subaddresses has a receive and a transmit buffer, as shown in Table 5-1.

The memory allocated to the unused receive subaddresses 0 and 31 is used to provide status information back to the rest of the system. At the end of every transfer, a transfer status word is written to these locations.

***Table 5-1* • Standard Memory Address Map**

Address	RAM Contents	Action
000–01F	RX transfer status words	The core only writes to these addresses (except when SA30LOOP is HIGH).
020–03F	Receive subaddress 1	
...	...	
3C0–3DF	Receive subaddress 30	
3E0–3FF	TX transfer status words	
400–41F	Not used	The core only reads from these addresses.
420–43F	TX transfer subaddress 1	
...	...	
7C0–7DF	TX transfer subaddress 30	
7E0–7FF	Not used	

If the SA30LOOP input is set HIGH, the RT maps transmit subaddress 30 to receive subaddress 30; i.e., the upper address bit is forced to 0. This provides a loopback subaddress, as per MIL-STD-1553B, Notice 2. The TSW is still written to address 03FE. It should be noted that this is not strictly compliant with the specification, since the transmit buffer will contain invalid data if the received command fails, e.g., with a parity error. The transmit buffer should only be updated if the receive command had no errors. To implement this function in full compliance, the SA30LOOP input should be tied LOW, and the RT backend should copy the receive memory buffer to the transmit memory buffer only after the RT signals that the message was received with no errors.

When the memory buffer is implemented within the FPGA using dual-port RAMs, separate receive and transmit RAM blocks can be used (each as 1 k words), as shown in Figure 5-1. In these cases, the RX memory is selected when A10 = 0 and the TX memory when A10 = 1. In this case, the SA30LOOP input must be tied LOW.

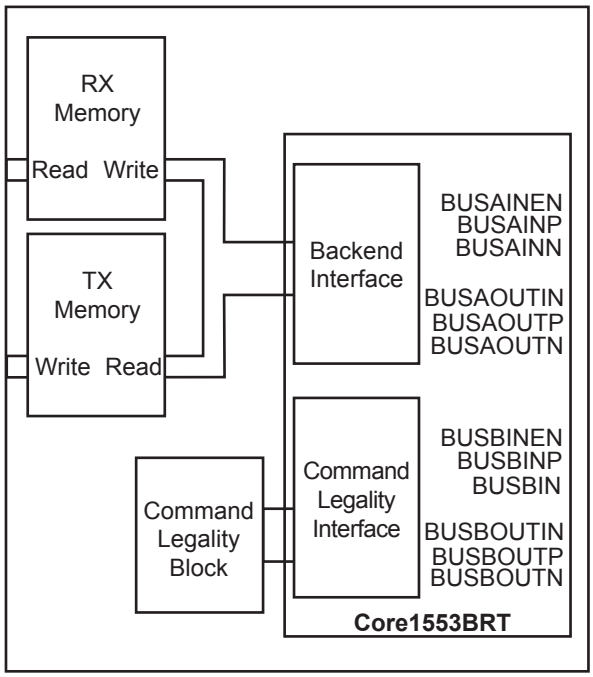


Figure 5-1 • **Using Internal FPGA Memory Blocks**

Memory Address Mapping

The core supports an external memory address mapper that allows the RT memory allocation to be easily customized. To use this function, the CMDVAL output must be latched by the ADDRLAT signal as shown in Figure 5-2. Then, the address mapper function can map the 1553B command words, data words including mode code data, and transfer status words to any memory address.

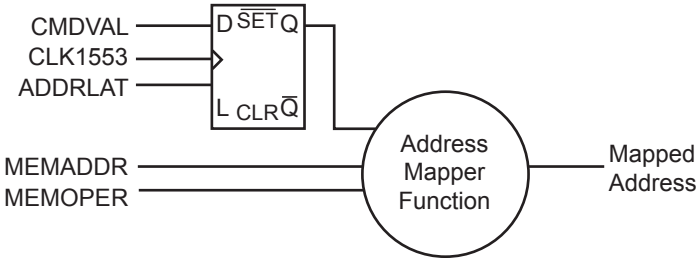


Figure 5-2 • **Memory Address Mapping**

Interrupt Vector Extension

The core generates a 7-bit interrupt vector that contains the subaddress and whether it was a transmit or receive message. Some systems may need to include whether the message was a broadcast, a mode code, or the actual word count in the interrupt vector. The core supports an interrupt vector extension function, similar to the address mapper function using the INTLAT signal, as shown in Figure 5-3.

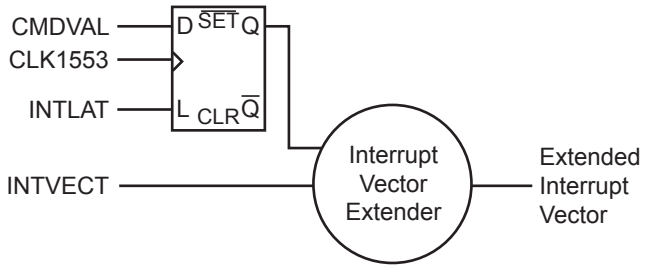


Figure 5-3 • **Interrupt Vector Extension**

Status Word Settings

Core1553BRT sets bits in the 1553B status word in compliance with MIL-STD-1553B. This is summarized in Table 5-2.

Table 5-2 • **Status Word Bit Settings**

Bit(s)	Function	Setting
15:11	RT Address	Equals the RTADDR input.
10	Message Error	Set whenever the RT detects a message error.
9	Instrumentation	Always 0
8	Service Request	Controlled by the SREQUEST input.
7:5	Reserved	Always 000.
4	Broadcast Received	Set whenever a broadcast message is received.
3	Busy	Controlled by the RTBUSY input.
2	Subsystem Flag	Controlled by the SSFLAG input.
1	Dynamic Bus Acceptance	Always 0. Core1553BRT does not operate as a bus controller.
0	Terminal Flag	Controlled by the TFLAG input. If an “inhibit terminal flag” mode code is in effect, will be 0.

Command Word Storage

At the start of every 1553B bus transfer, the 1553B command word is written to RAM locations 000–01F for receive operations and 3E0–3FF for transmit operations. The address used is as follows:

- CMD location, RX commands: '000000' and SA
- CMD location, TX commands: '011111' and SA

If the RT is implemented without a memory-based backend, the writing of the command word can be disabled (WRTCMD input). This simplifies the design of the backend logic that directly controls the backend function.

Transfer Status Words

At the end of every 1553B bus transfer, a transfer status word is written to the RAM in locations 000–01F for receive operations and 3E0–3FF for transmit operations. The address used is as follows:

- TSW location, RX commands: '000000' and SA
- TSW location, TX commands: '011111' and SA

As an example, the TSW address for a transmit command with subaddress 20 would be '01111110100' (3F4h). The TSW contains the information in Table 5-3.

If the RT is implemented without a memory-based backend, the writing of the TSW can be disabled. This simplifies the design of the backend logic that directly controls backend functions.

***Table 5-3* • Transfer Status Word**

Bit(s)	Name	Description	
15	USED	Set to 1 at the end of the transmit or receive command.	
14	OKAY	Indicates that no errors are detected; i.e., bits 11 to 5 are all 0.	
13	BUSN	Indicates on which bus the command was received: 0: BUSA 1: BUSB	
12	BROADCAST	Indicates a broadcast command.	
11	LPBKERRB	Indicates that the loopback logic detected an error in the transmitted data for bus B.	
10	LPBKERRA	Indicates that the loopback logic detected an error in the transmitted data for bus A.	
9	ILLEGAL CMD	The command was illegal. A request to transmit from either an illegal subaddress or an illegal mode code was received.	
8	MEMIFERR	Indicates that the DMA memory access failed to complete quickly enough.	
7	MANERR	Indicates that a Manchester encoding error was detected in the incoming data.	
6	PARERR	Indicates that a parity error was detected in the incoming data.	
5	WCNTERR	Indicates that an incorrect number of words was received.	
4:0	COUNT	SA1 to SA30	Indicates the number of words received or transmitted for that subaddress. If WCNTERR is 0, '00000' indicates 32 words. Otherwise, '00000' indicates zero words transferred.
		SA0 or SA31	Indicates which mode code was received or transmitted per the 1553B specification.

Backend Access Times

During normal operation, the backend must allow a memory access to complete within 19.5 µs. When either the command word or the TSW is written to memory, the backend must be capable of completing memory accesses in 10 µs.

While the status word is being transmitted, the core must write the command word to memory and fetch the first data word. Two memory accesses are performed in the 20 µs that the status word takes to transmit.

At the end of a “broadcast receive” command, Core1553BRT writes the last data word and the TSW value before the RT decodes the next command. Two memory accesses occur in the 20 µs that the command word is being decoded.

The core includes a timer that is set to terminate backend memory access at 19.5 µs or 10.0 µs when either WRTCMD or WRTTSW are active.

Data Transfers – Receive

When a “receive data transfer” command is detected, the core will decode each incoming word. At the end of each word, the core will assert MEMREQn. When MEMGNTn goes LOW, the core will write the data word to memory and release MEMREQn. This process is repeated until the correct number of words has been transferred. The core will then transmit its 1553B status word. Finally, the TSW is also written to memory.

Data Transfers – Transmit

When a “transmit data transfer” command is detected, the core will transmit its status word and assert MEMREQn. When MEMGNTn goes LOW, the core will read a data word from memory and release MEMREQn. Once the word is available, the core will transmit the data word. The core will continue to request data from the memory interface until the required number of words has been transferred. Finally, the TSW is written to memory.

RT-to-RT Transfer Support

The core supports RT-to-RT transfers. If a transmitting core does not start transferring data within the required time, the core will detect this and set the WCNTERR bit in the transfer status word.

Mode Codes

When the core receives a mode code, it first checks its command validity. If the command is valid, it is processed in accordance with the specification. Otherwise, the message error bit will be set in the 1553B status word. Table 5-4 lists the supported mode codes.

Two mode codes, (1) “transmit a vector word” and (2) “synchronize with data,” require external data. When EXTMDATA is inactive, the vector word value is set by the VWORD input, and the “synchronize with data” word is discarded. When EXTMDATA is active, these values are read from and written to memory. The MEMADDR output will be similar to a single-word data transfer message; bit 10 will reflect the command word TX bit, and bits 9:5 will be 00h or 1Fh, depending on whether the mode code subaddress is set to 0 or 31. Bits 4:0 will be zero. This implies the vector word will be read from location 400h or 7E0h, and the “synchronize with data” word is written to location 000h or 3E0h, depending on whether subaddress 0 or 31 is used.

When both WRTCMD and WRTTSW are active for each message, the command word and TSW value will be written to the same location; these writes can be distinguished by the MEMOPER output. This may cause some system problems, but this can be avoided by implementing an external address mapper function to map these accesses to different addresses.

Table 5-4 • **Supported Mode Codes**

T/R Bit	Mode Code	Function and Effect	Data Word	Core Supports	Broadcast Allowed
1	00000 0	Dynamic Bus Control The core does not support bus controller functions, so it will set the Message Error and Dynamic Bus Control bits in the status word.	No	No	No
1	00001 1	Synchronize The core will assert its SYNCNOW output after the command word has been received.	No	Yes	Yes
1	00010 2	Transmit Status Word The core retransmits the last status word.	No	Yes	No

***Table 5-4* • Supported Mode Codes (continued)**

T/R Bit	Mode Code	Function and Effect	Data Word	Core Supports	Broadcast Allowed
1	00011 3	Initiate Self-Test The core does not support self-test. Since the core supports the “transmit BIT word” mode code, this command is treated as legal and will not set Message Error.	No	Yes	Yes
1	00100 4	Transmitter Shutdown The core will disable the encoder on the other bus.	No	Yes	Yes
1	00101 5	Override Shutdown The core will re-enable the encoder on the other bus.	No	Yes	Yes
1	00110 6	Inhibit Terminal Flag The core will mask the TFLAG input, and the Terminal Flag bit in the status word will be forced to zero.	No	Yes	Yes
1	00111 7	Override Inhibit Terminal Flag The core will re-enable the TFLAG input.	No	Yes	Yes
1	01000 8	Reset Remote Terminal The core will assert its BUSRESET output after the command word has been received. It will also reset itself.	No	Yes	Yes
1	10000 16	Transmit Vector Word The core will transmit a single data word that contains the value on the VWORD input.	Yes	Yes	No
1	10010 18	Transmit Last Command Word The core will transmit a single data word that contains the last command word received.	Yes	Yes	No
1	10011 19	Transmit BIT Word The core will transmit a single data word that contains the extended core status information. The value of this word is defined in Table 5-6 on page 32.	Yes	Yes	No
0	10001 17	Synchronize with Data The core will assert its SYNCNOW output after the data word has been received.	Yes	Yes	Yes
0	10100 20	Selected Transmitter Shutdown The core only supports two busses. Hence, this command is illegal. The Message Error bit in the status word will be set.	Yes	No	Yes
0	10101 21	Override Selected Transmitter Shutdown The core only supports two busses. Hence, this command is illegal. The Message Error bit in the status word will be set.	Yes	No	Yes

Loopback Tests

Core1553BRT performs loopback testing on all of its transmissions. The transmit data is fed back into the receiver, and each transmitted word is compared. If an error is detected, the loopback fail bit is set in the TSW and also in the BIT word.

Error Detection

Table 5-5 gives action for error conditions detected.

***Table 5-5* • Error Detection**

Error Condition	Action
Command Word 1. Parity or Manchester encoding errors 2. Incorrect SYNC waveform	Command is ignored. No interrupt generated.
Mode Codes 1. Illegal mode code or invalid subaddress (from internal or external legality block)	MSGERR in SW is set, and the SW is transmitted. Message Failure interrupt generated.
Broadcast Data Commands 1. TX bit set in command word	Data transfer is aborted. MSGERR in SW is set, and the SW is not transmitted. Message Failure interrupt generated.
Data Word 1. Parity or Manchester encoding errors 2. Incorrect number of words received 3. Data words are continuous. 4. Incorrect SYNC waveform	Data transfer is aborted. MSGERR in SW is set, and the SW is not transmitted. Message Failure interrupt generated.
RT-to-RT 1. First command word must be RX. 2. Second command word must be TX and non-broadcast. 3. RX RT checks the TX SW and verifies the SYNC pattern, RT address, MSGERR, and BUSY fields. 4. The first data word sync must be received within 57 µs of the command word parity bit.	Data transfer is aborted. MSGERR in SW is set, and the SW is not transmitted. Message Failure interrupt generated.
Transmit Data Error 1. The RT monitors its transmissions on the bus through its decoder and verifies that the correct data is transmitted with no Manchester or parity errors.	Data transfer is aborted. MSGERR in SW is set, and the SW is not transmitted. Message Failure interrupt generated.
Backend Failure 1. The RT makes sure that the backend responds to read and write cycles within the required time.	Data transfer is aborted. MSGERR in SW is set, and the SW is not transmitted. Message Failure interrupt generated.
BUSY 1. Backend RTBUSY input is active at any point during the message.	Data transfer is aborted. BUSY in SW is set, and the SW is transmitted. Message Failure interrupt generated.
Transmitter Overrun 1. Transmits for greater than 668 µs. The internal state machines prevent this from happening, but the core includes the required timer and functionality. This is implemented separately to the encoder to provide complete protection.	Core shuts down transmissions on bus.

Built-In Test Support

Core1553BRT provides a BIT word. This is used to communicate fail information back to the bus controller. The BIT word contains the information in Table 5-6.

***Table 5-6* • BIT Word**

Bit(s)	Function	Description
15	BUSINUSE	Indicates on which bus the transmit BIT word command was received: 0: Bus A 1: Bus B
14	LPBKERRB	Indicates that the loopback logic detected an error on the transmitted data for bus B. This bit is cleared by the CLRERR input.
13	LPBKERRA	Indicates that the loopback logic detected an error on the transmitted data for bus A. This bit is cleared by the CLRERR input.
12	SHUTDOWNB	Indicates that bus B is shut down. This occurs after a “transmitter shutdown” mode code is received or the hardware timer detected that the core transmitted for longer than 668 µs on bus B.
11	SHUTDOWNA	Indicates that bus A is shut down. This occurs after a “transmitter shutdown” mode code is received or the hardware timer detected that the core transmitted for longer than 668 µs on bus A.
10	TFLAGINH	Terminal flag inhibit setting
9	WCNTERR	A word count error has occurred. This bit is cleared by the CLRERR input.
8	MANERR	A Manchester encoding error has occurred. This bit is cleared by the CLRERR input.
7	PARERR	A parity error has occurred. This bit is cleared by the CLRERR input.
6	RTRTTO	The transmitting RT did not provide data on RT-to-RT transfer. This bit is cleared by the CLRERR input.
5	MEMFAIL	The backend memory interface failed to complete an access within the required time. This bit is cleared in the CLRERR input.
4:0	VERSION	Indicates the core version: '00001': version 2.0 '00010': version 2.1 '00011': version 2.12 '00100': version 2.2 '00101': version 2.21 '01000': version 3.0 '01001': version 3.1 '01010': version 3.2 '01011': version 3.3 '01100': version 3.4 '01101': version 4.0 '01110': version 4.1

Command Legalization Interface

1553B commands can be legalized in two ways with Core1553BRT. For RTL versions, one of the modules in the source code can be edited to legalize or make illegal command words based on the subaddress, mode code, word count, or broadcast fields of the command word. For Obfuscated and RTL versions, external logic can be used to decode the legal/illegal command words (Figure 5-4).

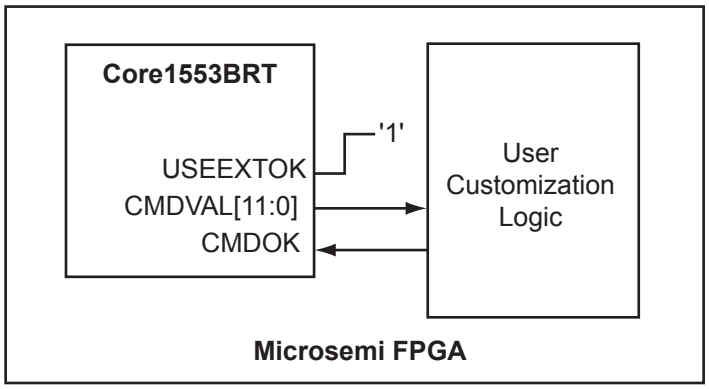


Figure 5-4 • **Command Legalization Logic**

The user customization logic block takes in CMDVAL and simply sets CMDOK for all legal command words. The CMDVAL encoding is given in Table 5-7. The external logic must implement this function within 3 µs.

Table 5-7 • **CMDVAL Encoding**

Bit(s)	Function	Description
11	Broadcast	'1' indicates broadcast; i.e., the RT address was set to 31 in the 1553B command word.
10	Transmit or Receive	TX/RX field from the 1553B command word. '0' indicates receive and '1' transmit.
9:5	Subaddress	Subaddress field from the 1553B command word
4:0	Word Count Mode Code	Word count field from the 1553B command word. When the subaddress is 0 or 31, this contains the 1553B mode code.

6 – Testbench Operation and Modification

Verification Testbench

Microsemi has developed a 1553B verification testbench (Figure 6-1) that you can use to verify core performance per the 1553B specification. The testbench is coded in VHDL and contains four Core1553B remote terminals connected to a bus control function and backend interfaces. A procedural testbench controls the various blocks and implements the tests. The source code is not made available with Obfuscated licenses of the core.

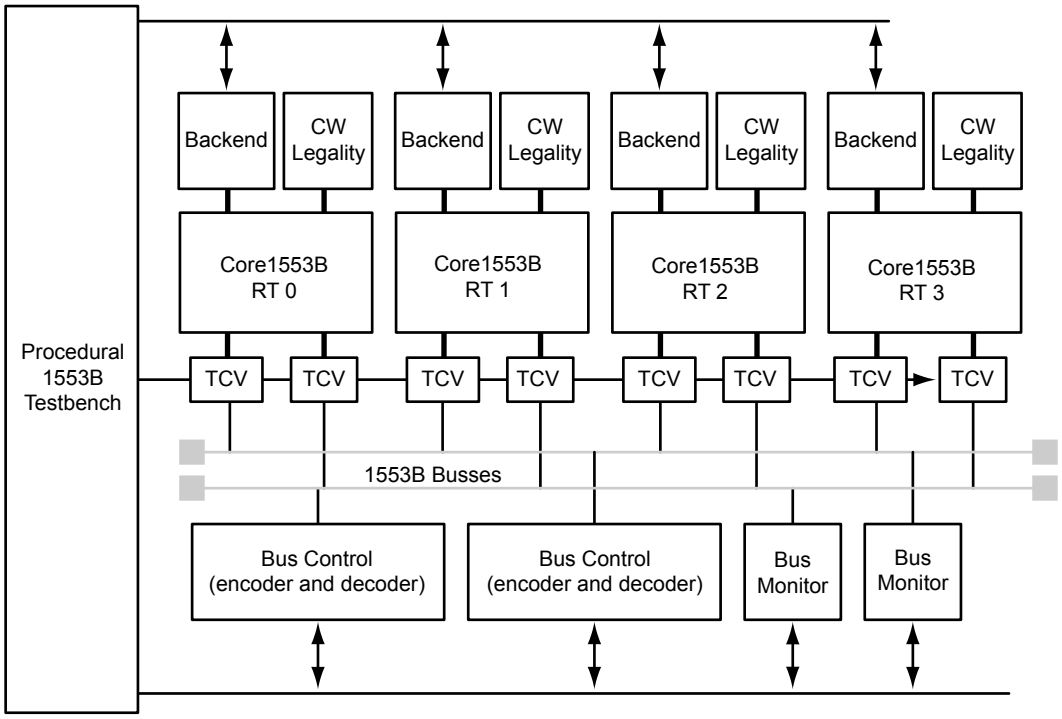


***Figure 6-1* • Verification Testbench**

The testbench includes four RTs to test core variants, each with different setups of the core configuration inputs (Table 6-1).

***Table 6-1* • Verification Testbench RT Configuration**

RT	Clock Speed	Memory Interface	Command Word Legality*
0	Variable	Asynchronous	Internal
1	16 MHz	Synchronous	Internal
2	16 MHz	Asynchronous	Internal
3	16 MHz	Synchronous	External
*Note: *All command word legality interfaces are external for Obfuscated simulation.*			

The testbench uses a command word legality module that disables subaddresses 26 and 27 for transmit commands. For receive commands, subaddress 25 is disabled, and subaddress 27 is only enabled for word counts 1 to 9. "External Command Word Legality Example" on page 50 shows the source code for a command legality module implementing this behavior.

The procedural testbench has direct control of the bus control modules, the transceivers, and the backend interfaces (including all the backend core inputs). This direct control allows the testbench to initialize the backend RAM contents and to verify the contents after each transfer. The bus controller function has the capability of injecting errors and varying the data rate to verify the 1553B decoder behavior.

During operation, the testbench verifies all command, data, and status words. All memory accesses are verified, and the resultant transfer status word for every message is checked.

During invocation, the top-level generics can be set to alter the simulation. These generics are specified in Table 6-2.

Table 6-2 • **Verification Testbench RT Configuration**

Generic	Type	Default Value	Function
ENBUSMON	Boolean	FALSE	Enables the bus monitor function. The testbench displays every 1553B word that is transmitted on the busses.
ENRAMMON	Boolean	FALSE	Enables the RAM monitor function. The testbench displays all the memory reads and writes performed by each of the cores.
RUNTEST	Integer	0	Allows the testbench to run without user input. Forces the testbench to run the test number as defined by the menus (see below). If RUNTEST is greater than 10, the testbench runs test number (n – 10) and quits; for example, if n = 12, the testbench runs test number (12 – 10 = 2), which is “Microsemi Tests – Standard Mode,” and then quits the simulation.

When you run the testbench, it asks which tests you want to run. The options are as follows (the options are summarized in Table 6-3 on page 36):

```
# 1553B Test Harness - Microsemi IP Solutions Group
# Production Release 4.1 - February 2015
#
#  Test Options
#  1 : Quick Run
#  2 : Microsemi Tests - Standard Mode
#  3 : Microsemi Tests - Address Mapper Enabled
#  4 : RT Test Plan
#  5 : Microsemi Tests - Short mode (i.e. fewer tests)
#  9 : Do Everything
#  A : Do Everything, Monitors Off and Quit
#  B : Turn On Bus Monitors
#  R : Turn On RAM Monitors
#  P : Pause Simulation - allows waves to update
#  X : Run for a single 1553B Word; i.e. 20µs
#  M : Send a message    M BUS RT TX SA WC [SEED INC]
#  D : Display RT Memory D RT TX SA
#  S : Set RT Memory     S RT TX SA SEED INC
#  Q : Quit
#
#  Enter Option, for  demo use 1 ?
```

***Table 6-3* • Verification Testbench Menu Summary**

Menu Command	Action
Quick Run	Runs a few simple 1553B command sequences to demonstrate that the core is functioning.
Microsemi Tests – Standard Mode	Runs the complete set of Microsemi verification tests. For the RTL code (*verif_rtl* directory), this takes several hours, depending on the computer used. For the Obfuscated version, the simulation time is significantly longer.
RT Test Plan	This implements a subset of the protocol tests specified in the MIL-HDBK-1553A handbook. This takes a very long time to run, as thousands of 1553B command words need to be transmitted and verified.
Do Everything	This runs the three options listed above, i.e., Quick, Microsemi, and Test Plan.
Microsemi Tests – Address Mapper Enabled	This runs the Standard Mode tests but uses an address mapping function (as on 53).
Microsemi Tests – Short Mode	This runs a subset of the Standard Mode tests; the runtime is much shorter than for the standard tests.
Do Everything and Quit	This runs the three options in Do Everything above and then quits the simulation. Setting the RUNTEST generic to 9 can automatically run this test option.
RAM Monitors	The testbench can display a message every time the backend RAMs are written to or read from. This option enables or disables the RAM monitors. When enabled, the testbench runs more slowly due to the printing overhead in VHDL.
Bus Monitors	The testbench includes 1553B bus monitors that display every word transmitted on the 1553B busses. This option enables or disables the bus monitors. When enabled, the testbench will run slower due to the printing overhead in VHDL.
Pause Simulation	Exits the simulation and returns to the vsim environment. This may be required to cause the waves window to update. Simulation can be simply restarted using the **run -all** command.
Run 1553B Word	Simply allows the simulation to run for 20 µs.
Send a Message	Allows interactive 1553B message creation.
Set RT Memory	Allows the values in one of the RT memories to be set.
Display RT Memory	Displays the contents of one of the RT memories.

Interactive Operation

The verification testbench allows you to create 1553B messages and transmit from the simulator command line. Three commands are provided that support this feature (M, S, and D). Their parameters are given below. All parameters are decimal integers separated by spaces or commas. If a number begins with “#,” “A–F,” or “a–f,” it is interpreted as a hexadecimal value. Basic error checking is performed on the entered values.

Message Parameter M

Consider the following example:

```
M BUS RT TX SA WC [SEED [INC]]
```

- M – Transmit a message
- BUS – Specifies the bus to use, 0 or 1
- RT – Specifies the RT number, 0–31
- TX – RT transmit or receive: 1 = Transmit, 0 = Receive
- SA – Subaddress to use, 0–31
- WC – Number of words to transmit or receive, 0–32
- SEED – Sets the first data used for the message to this value. If the RT is to receive data, the bus controller transmits the data. If the RT is to transmit, the testbench initializes the RT backend RAM for the appropriate subaddress. If no data is provided, the RT receive data is '0000', and the transmit data is whatever the RT memory contains.
- INC – Increments each data word in the message by this value. If not specified, defaults to zero so all data words contain the same value.

Message Parameter S

Consider the following example:

```
S RT TX SA SEED [INC]
```

- S – Set RT memory
- RT – Specifies the RT number, 0–31
- TX – RT transmit or receive memory: 1 = Transmit, 0 = Receive
- SA – Which subaddress, 0–31
- SEED – Sets the first data value
- INC – Increments each data word in the message by this value. If not specified, defaults to zero so all data words contain the same value.

Message Parameter D

Consider the following example:

```
D RT TX SA
```

- D – Display RT memory
- RT – Specifies the RT number, 0–31
- TX - RT transmit or receive memory: 1 = Transmit, 0 = Receive
- SA – Which subaddress, 0–31

Verification Tests

The verification testbench includes test procedures to check the following:

Simple Messages BC–RT and RT–BC

- Subaddress word count combinations

RT-to-RT Messages

Mode Codes

- Verifies all codes

Broadcast

- Status word settings
- All mode codes verified

Illegal Commands (legality interface)

- Mode code
- Disabled subaddresses
- Normal, RT-to-RT, and broadcast
- Internal and external legality logic

Special Features

- Subaddress 30 loopback
- TSW enabled and disabled
- Command word memory write enabled and disabled
- Address mapping functions
- Interrupt vector extension functions

Error Conditions

- Variable bit rates ± 10,000 Hz (1%)
- Variable backend GNT and WAIT delays
- Parity and Manchester encoding errors
- Synchronization pattern corruption
- RT-to-RT illegal command words
- Word count errors
- Word gaps
- Transmitter timeout
- Status word flag bits
- Superseding command acceptance
- RT address parity logic
- Receive on disabled bus
- Transmitter overrun
- BIT word values
- Loopback logic
- Extra command and data words
- Noise on the data bus
- Superseding commands on second bus

VHDL Testbench

Microsemi provides an example testbench that you can use as the starting point for design verification of the core in your design. A block diagram of the testbench is shown in Figure 6-2.

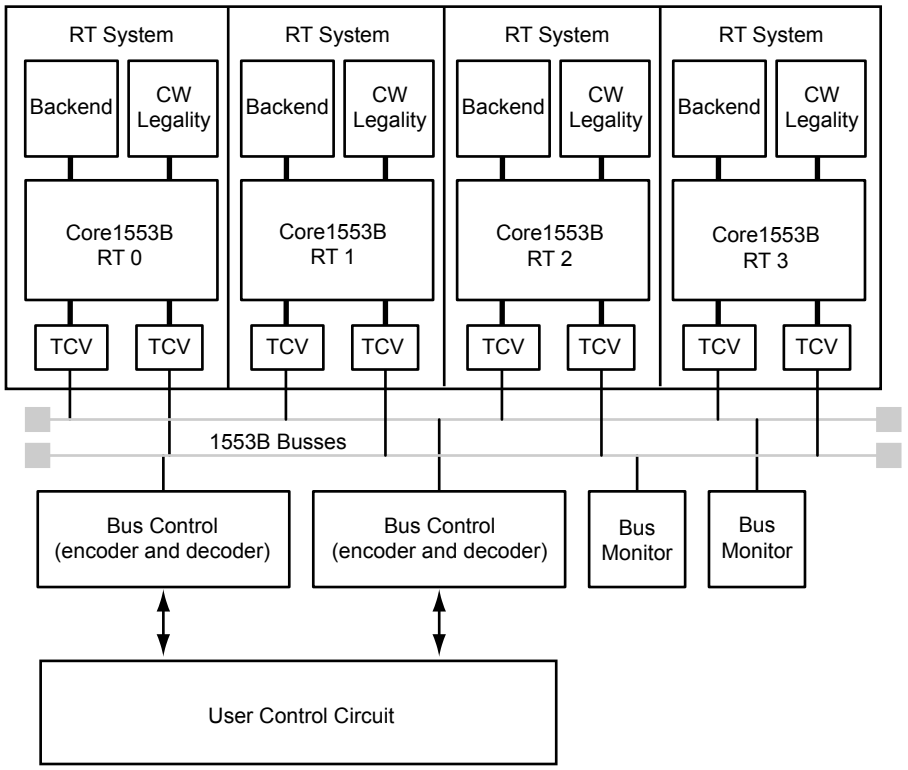


Figure 6-2 • **VHDL Testbench**

The testbench creates an RT System (QRTSystem) by adding the transceivers, backend interface, and command legality interface to the core. The top level (Qtbench) includes four of these cores. All the cores in this case are identical. The source code modules used are listed in Table 6-4 on page 40

.

***Table 6-4* • VHDL Testbench Modules**

Entity	Source Provided	Description
Qtbench	Yes	The testbench top level. This contains the bus control function and several remote terminals.
QTBEncDec	No	Test block that emulates a bus controller. Transmits 1553B command and data words and then decodes the status and data words generated by an RT in response.
QBusmon	No	1553B bus monitor. Monitors the bus and reports on the bus traffic.
QRTsystem	Yes	Hierarchical block with the core plus transceiver, backend, and command word legality blocks to the core.
QBusTransceiver	No	Simply takes the six unidirectional signals from the core and models a transceiver connected to a bus.
QTBBackend	No	Connects to core backend interface and provides the following functions: Implements an asynchronous 2k×16 or 8k×16 memory block, which provides the 32 receive and transmit subaddresses. Has a built-in address mapping function. Loops back the receive subaddress locations to the transmit memory. On a good message received, interrupts the correctly received words, which are copied from the RX subaddress to the TX subaddress. If the address mapping function is enabled, the “synchronize with data” word is copied to the transmit vector word memory location. Generates the interrupt acknowledge.
CWLegality	Yes	Implements external command validity checking

The testbench uses a command word legality module that disables subaddresses 26 and 27 for transmit commands. For receive commands, subaddress 25 is disabled, and subaddress 27 is only enabled for word counts 1 to 9. "External Command Word Legality Example" on page 50 shows the source code for a command legality module implementing this behavior.

The Core1553B Model*Sim*® library (in the *mti/verif_rtl* directory) contains compiled models for the complete environment. Design source code of the top-level blocks is provided in the *source* directory to enable you to create your own simulation environment using this testbench as a starting point. Examine the source files for Qtbench and QRTSystem to obtain a full understanding of the core operation.

Qtbench has a top-level generic, CMODE, that you can set to 0 or 1. When 0, the core is configured with WRTTSW, WRTCMD, and EXTMDATA set to '100'. When 1, these values are set to '011', and the backend module implements an address mapper function as described in "Implementation Hints" on page 49.

Using the VHDL QTBEncDec Module

You can instantiate the QTBEncDec module in your design and use it to initiate 1553B messages. The top level of the module is shown in the code below, along with a description of these ports (Table 6-5).

```
entity QTBENCDEC is
  port ( CLK16    : in    std_logic;
         RSTn     : in    std_logic;
         STOPCLK  : in    BOOLEAN;
         START    : in    BOOLEAN;
         QMSG     : in    TQMSGREQ;
         BUSY     : out   BOOLEAN;
         QOUT     : out   TQMSGOUT;
         BUSPOS   : inout std_logic;
         BUSNEG   : inout std_logic
       );
end QTBENCDEC;
```

***Table 6-5* • TBEncDec Port Descriptions**

Port	Dir.	Type	Function
CLK16	In	std_logic	Clock source for the encoder and decoder; must be 16 MHz
RSTn	In	std_logic	Active low asynchronous reset; must be pulsed LOW at the start of simulation
STOPCLK	In	BOOLEAN	The encoder has an internal clock generator; when this input is TRUE, the clock generator is halted. This allows the simulator to exit gracefully. This input can be tied permanently FALSE to disable the STOPCLK feature.
START	In	BOOLEAN	This input is pulsed TRUE to start a 1553B message. If it is not synchronized to a clock and only needs to be pulsed for a simulation delta cycle, use the following code: START <= TRUE; wait for 0 ns; START <= FALSE;
QMSG	In	TQMSGREQ	Input record structure that defines the message that will be transmitted; see below
BUSY	Out	BOOLEAN	Indicates that the encoder/decoder is busy
QOUT	Out	TQMSGOUT	Output record structure containing message data transmitted on the bus
BUSPOS	Inout	std_logic	Connects to the positive side of the 1553B bus
BUSNEG	Inout	std_logic	Connects to the negative side of the 1553B bus

To initiate a message, a simple record structure is set up and the START input is strobed. The module asserts BUSY until the 1553B message is completed and then deasserts BUSY. The QOUT record structure contains the data sent and received on the bus. The two record structures used here are shown in Table 6-6 on page 42 and Table 6-7 on page 42.

***Table 6-6* • TMSGREQ Record Structure**

Element	Type	Default	Function
RT	INTEGER, 0 to 31	0	Command word RT value. Broadcast is supported when the RT number is 31.
TX	INTEGER, 0 to 1	0	Command word TX value
SA	INTEGER, 0 to 31	0	Command word SA value
MCWC	INTEGER, 0 to 32	0	Word count or mode code value. For data transfers, values 1 to 32 should be used, and for mode codes, values 0 to 31.
RTRT	BOOLEAN	FALSE	If TRUE, an RT-to-RT command pair is transmitted. The transmit command word uses the RT, TX, and SA elements.
RT2	INTEGER, 0 to 31	0	RT command word value for the receive RT in RT-to-RT messages
SA2	INTEGER, 0 to 31	0	SA command word value for the receive RT in RT-to-RT messages
DATA	PACKET	Unknown, i.e., 'U'	Data to be transmitted for an RT receive message. This is an array(0 to 31) of WORD; WORD is std_logic_vector(15 down to 0). Index 0 is used for mode code data. For example: DATA(0) <= "0000111100001111"; DATA(1) <= to_word(16#1234#); DATA <= initdata(16#1000#); DATA <= initdata(16#5555#,0); The first example sets the first data word using standard VHDL assignments to std_logic_vector. The second example uses a function provided in one of the underlying Core1553B packages to set the word to the hexadecimal value 1234. The third example uses another function call to set the complete data pattern to an incrementing value starting at 1000 hex. The final example sets the complete data pattern to 5555 hex. The second argument is the increment between consecutive data words.

***Table 6-7* • TQMSGOUT Record Structure**

Structure Element	Type	Function
OKAY	BOOLEAN	Indicates that the message was correctly processed with no errors.
COUNT	INTEGER	Number of words received
CW1	WORD	First command word
CW2	WORD	Second command word. If no second command word, will contain '–' in all 16 bits.
SW1	WORD	First status word. If no status word, will contain '–' in all 16 bits.
SW2	WORD	Second status word. If no status word, will contain '–' in all 16 bits (only for RTRT messages).
DATA	PACKET	Data packet for the message. When the RT is receiving, it will contain a copy of the transmitted data when transmitting the data the RT transmitted. On RT-to-RT messages, the data transferred between the RTs will be provided. For mode codes, the data word will be in the first location, i.e., DATA(0).

The VHDL code below shows a simple code fragment that generates a 10-word BC–to–RT 1 subaddress 5 message, with data incrementing from 1200 hex, which then displays the message. “print_msgout” is a procedure provided in the underlying Core1553B package that displays the message details. It needs to pass the two record structures described above.

```
signal BUSASTART : BOOLEAN;
signal BUSAMSG   : TQMSGREQ;
signal BUSABUSY  : BOOLEAN;
signal BUSAOUT   : TQMSGOUT;

begin

process
 begin
  BUSAMSG.RT   <= 1;
  BUSAMSG.TX   <= 0;
  BUSAMSG.SA   <= 5;
  BUSAMSG.MCWC <= 10;
  BUSAMSG.RTRT <= FALSE;
  BUSAMSG.DATA <= initdata(16#1200#);

  -- Do the message
  BUSASTART <= TRUE;
  wait for 0 ns;
  BUSASTART <= FALSE;
  wait for 0 ns;
  while BUSABUSY loop
    wait for 1 us;
  end loop;

  -- Display the data transmitted
  print_msgout(BUSAMSG,BUSAOUT);
  wait;
end process;
```

Verilog Testbench

Microsemi provides an example Verilog testbench that you can use as the starting point for design verification of the core within your design. A block diagram of the testbench is shown in Figure 6-3.

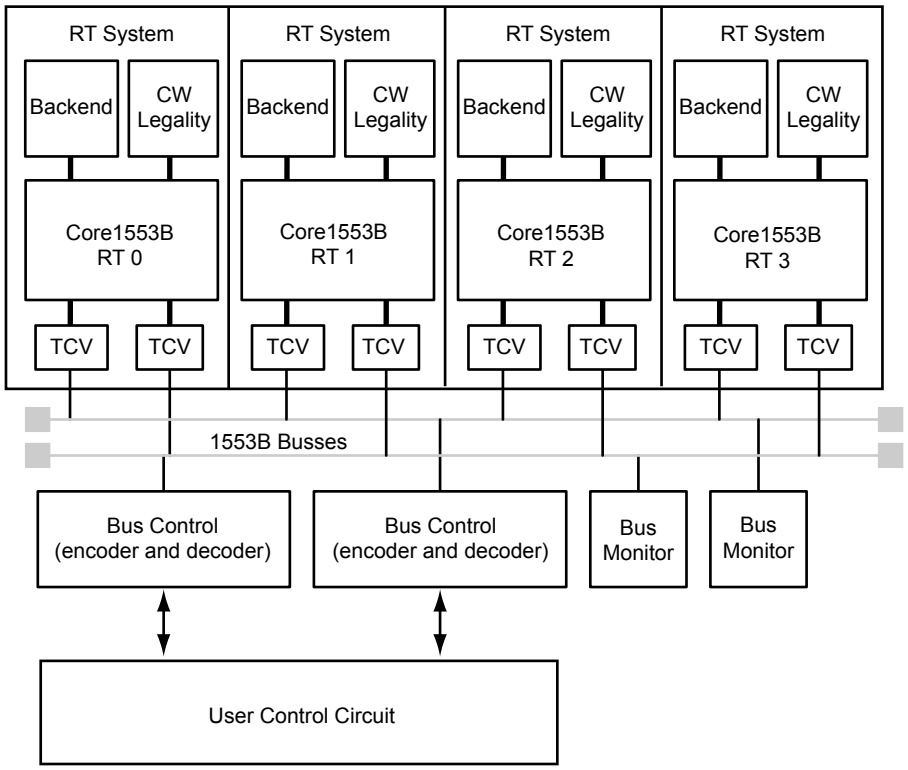


***Figure 6-3* • Verilog Testbench**

The testbench creates an RT System (QRTSYSTEM) by adding the backend interface and the command legality interface to the core. The top level (QTBENCH) includes four of these cores. All the cores in this case are identical. Table 6-8 on page 45 lists the source code modules.

Table 6-8 • **Verilog Testbench Modules**

Module	Source Provided	Description
QTBENCH	Yes	The testbench top level. This contains the bus control function and several remote terminals.
QTBENCDEC	No	Test block that emulates a bus controller. This transmits 1553B command and data words and then decodes the status and data words generated by an RT in response.
QRTSYSTEM	Yes	Hierarchical block with the core plus a transceiver, backend, and command word legality block to the core
QTBBACKEND	Yes	This connects to the core backend interface and provides the following functions: 1. Implements an asynchronous 2k×16 or 8k×16 memory block providing the 32 receive and transmit subaddresses. 2. Has a built-in address mapping function. 3. Loops back the receive subaddress locations to the transmit memory. On a good message received, interrupts the correctly received words copied from the RX subaddress to the TX subaddress. If the address mapping function is enabled, the “synchronize with data” word is copied to the transmit vector word memory location. 4. Generates the interrupt acknowledge.
CWLEGALITY	Yes	5. Implements external command validity checking

The testbench uses a command word legality module that disables subaddresses 26 and 27 for transmit commands. For receive commands, subaddress 25 is disabled, and subaddress 27 is only enabled for word counts 1 to 9. "External Command Word Legality Example" on page 50 gives the source code for a command legality module implementing this behavior.

The Core1553B Model*Sim* library (in the *mti/user_vlog* directory) contains compiled models for the complete environment. Design source code of the top-level blocks is provided (in the *source* directory) to enable you to create your own simulation environment using this testbench as a starting point. Examine the source files for QTBENCH and QRTSYSTEM to obtain a full understanding of the core operation.

QTBENCH has a top-level parameter, CMODE, that can be set to 0 or 1. When 0, the core is configured with WRTTSW, WRTCMD, and EXTMDATA set to '100'. When 1, these values are set to '011', and the backend module implements an address mapper function as described in "Implementation Hints" on page 49.

Using the Verilog QTBENCDEC Module

You can instantiate the QTBENCDEC module in your design and use it to initiate 1553B messages. The top level of the module is shown below, along with a description of these ports (Table 6-9).

```
module QTBENCDEC (CLK, RSTN,
                  BUSPOS, BUSNEG,
                  TXSTROBE, TXCW, TXDATA,
                  RXSTROBE, RXSTAT, RXDATA,
                  BUSY
                 );
input  CLK;
input  RSTN;
inout  BUSPOS;
inout  BUSNEG;
input  TXSTROBE;
input  TXCW;
input  [15:0] TXDATA;
output RXSTROBE;
output [ 3:0] RXSTAT;
output [15:0] RXDATA;
output BUSY;
```

***Table 6-9* • Verilog TBENCDEC Port Descriptions**

Port	Direction	Function
CLK	In	16 MHz clock source for the encoder and decoder. All inputs are sampled on the rising clock edge, and outputs are registered on the rising clock edge.
RSTn	In	Active low asynchronous reset; must be pulsed LOW at the start of simulation.
BUSPOS	Inout	Connects to the positive side of the 1553B bus.
BUSNEG	Inout	Connects to the negative side of the 1553B bus.
TXSTROBE	In	Pulsed HIGH for a clock cycle to load a 1553B word into the encoder.
TXCW	In	0: The encoder transmits a data word. 1: The encoder transmits a command word.
TXDATA	In[15:0]	Transmit word
RXSTROBE	Out	Indicates that the RXSTAT and RXDATA outputs contain valid data.
RXSTAT	Out[3:0]	Provides status information on the RXDATA output; see Table 6-10 on page 46 for a summary.
RXDATA	Out[15:0]	Received word
BUSY	Out	Indicates that either the encoder or the decoder is busy; active when transmit data is queued in the transmit FIFO, or when it is being transmitted.

***Table 6-10* • RXSTAT Value Definitions**

Bit	Function	Description
0	Type	0: Data word 1: Command/status word
1	Burst	Indicates that the data word was contiguous with the previous one.
2	Error	Indicates that an encoding error was detected in the word.
3	From us	Indicates that QTBENCDEC transmitted the word.

QTBENCDEC contains a 1553B encoder and decoder. A transmit FIFO is provided, enabling 64 words of transmit data to be loaded into the module. The maximum 1553B message length that the encoder is required to transmit is 33 words. Receive data is strobed out of the module. All 1553B data words are put out from this module.

The code below shows a simple code fragment that generates a 30-word BC–to–RT 1 subaddress 10 message with data incrementing from 4,096 decimal.

```
/*************************************************************************/
// The Bus Controller Modules
//

QTBENCDEC BCA
  (.CLK(CLK),
   .RSTN(RSTN),
   .BUSPOS(BUSAPOS),
   .BUSNEG(BUSANEG),
   .TXSTROBE(BCA_TXSTB),
   .TXCW(BCA_TXCW),
   .TXDATA(BCA_TXDATA),
   .RXSTROBE(BCA_RXSTB),
   .RXSTAT(BCA_RXSTAT),
   .RXDATA(BCA_RXDATA),
   .BUSY(BCA_BUSY)
  );

/*************************************************************************/
// Store the data put out by the decoder
//
// STAT[3:0] = FROMUS BURST ERROR CW

always @(posedge CLK)
  begin
    if (BCA_INIT==1'b1)  BCA_COUNT = 0;
    if (BCA_RXSTB == 1'b1)
      begin
        BCA_STORE[BCA_COUNT] = {BCA_RXSTAT, BCA_RXDATA };
        BCA_COUNT = BCA_COUNT+1;
      end
  end

/*************************************************************************/
// Main Test Procedure
//

reg [15:0] CW;
reg [15:0] DW;
reg [ 4:0] RT5BIT;
integer  i;
integer  RT;

initial
  begin
    RSTN = 1'b0;
    BCA_TXSTB = 1'b0;    // Used to load a word into the transmitter
    BCB_TXSTB = 1'b0;
    BCA_INIT  = 1'b1;    // Used to reset the store pointer on the Decoder
    BCB_INIT  = 1'b1;
    #312700;
    @(negedge CLK);
    RSTN = 1'b1;

    $display("Core1553BRT Verilog Test Harness Production 19Jan09");

    $display("Receive BC-RT 1 SA 10 WC 8 Message on BUS A");
    CW = { 5'b00001, 1'b0, 5'b01010, 5'b01000 };
    transmit_CW(0,CW);
    for ( i=1; i<=8; i=i+1) transmit_DW(0,4095+i);
    wait_to_complete(0);
    display_bus(0);
    #9000000;
```

```
    $display("Transmit RT-BC 1 SA 10 WC 6 Message on BUS B");
    CW = { 5'b00001, 1'b1, 5'b01010, 5'b00110 };
    transmit_CW(1,CW);
    wait_to_complete(1);
    display_bus(1);
    #9000000;

    $display("Get the Vector Word from each RT");
    for (RT=0; RT<=3;RT=RT+1)
    begin
      RT5BIT = RT;
      CW = { RT5BIT, 1'b1, 5'b00000, 5'b10000 };
      transmit_CW(0,CW);
      wait_to_complete(0);
      $display("RT %0d Got SW %h Got VW %h",
        RT,BCA_STORE[1][15:0],BCA_STORE[2][15:0]);
    #9000000;
    end

    #4000000;
    $display(" ");
    $display("Simulation complete");
    $display(" ");
    $stop;
  end

endmodule
```

7 – Implementation Hints

You can configure Core1553BRT to provide backend interfaces for a variety of hardware and software requirements.

The backend interface has been designed to simplify backend hardware design; the core supports both synchronous and asynchronous backends with bus arbitration and variable read and write strobe pulse widths.

To accommodate software requirements, you can modify the backend address map and interrupt vectors with simple address mapping functions implemented in hardware (explained in "Modifying the Backend Address Map" on page 53).

Table 7-1 shows some typical applications and the core configurations required.

***Table 7-1* • Typical Core Implementations**

Type	Description	Parameters
Standard-CW	A 2k×16 memory buffer is used with 32 words of memory allocated for each TX and RX subaddress. The 1553B command word is written to memory locations unused by the mode code subaddresses. The system can read the command word value to determine the number of words that were received.	WRTTSW = 0 WRTCMD = 1 EXTMDATA = 0 Address Mapper = No CW Legality = No
Standard-TSW	A 2k×16 memory buffer is used with 32 words of memory allocated for each TX and RX subaddress. The Core1553B TSW is written to memory locations unused by the mode code subaddresses. The system can read the TSW value to determine the number of words that were received. This implementation provides extra status information, such as the bus on which the message was received.	WRTTSW = 1 WRTCMD = 0 EXTMDATA = 0 Address Mapper = No CW Legality = No
Direct Device	No memory is used; the Core1553BRT backend directly connects to the device. In this case, all unused 1553B subaddresses should be invalidated. If the device only accepts a fixed number of data words, only that word count should be legal for the subaddress in use. Should an error be detected, this will be indicated by the interrupt vector, and the data should be discarded. No command words or TSW values are written to memory.	WRTTSW = 0 WRTCMD = 0 EXTMDATA = 0 Address Mapper = No CW Legality = Yes
Compatibility Mode	The memory allocation emulates another 1553B remote terminal, allowing software drivers to be reused. A backend address mapping function is implemented that reads and writes command and data words from and to the memory addresses used by the remote terminal that Core1553BRT is replacing.	WRTTSW = 0 WRTCMD = 1 EXTMDATA = 1 Address Mapper = Yes CW Legality = Maybe

External Command Word Legality Example

The core provides three ports (USEEXTOK, CMDVAL, and CMDOK) that allow the legal command word set to be modified. When USEEXTOK is LOW, the core internally decides which command words are legal (the legal command word set is defined in the *Core1553BRT MIL-STD-1553B Remote Terminal* datasheet). When USEEXTOK is HIGH, an external block decodes the CMDVAL output and generates a CMDOK input to indicate legal command words.

The VHDL and Verilog code blocks below implement an external legality checker that does the following:

- Legalizes mode codes as per the *Core1553BRT MIL-STD-1553B Remote Terminal* datasheet
- Disables transmits from subaddresses 26 and 27
- Disables receives to subaddress 25
- Only enables word counts 1 to 9 and receives to subaddress 27

The source files for these modules are provided in the *source* directory.

The core allows 3 µs for the legality block to decode CWVAL and generate the CMDOK value; this can be implemented within the FPGA, as shown below.

VHDL Example

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity CWLEGALITY is
  port ( CWVAL      : in  std_logic_vector(11 downto 0);
           CMDOKAY : out std_logic
         );
end CWLEGALITY;

architecture RTL of CWLEGALITY is
signal BROADCAST    : std_logic;
signal ISMCODE      : std_logic;
signal TX           : std_logic;
signal SA           : std_logic_vector(4 downto 0);
signal WCMC         : std_logic_vector(4 downto 0);

begin

-- Decode incoming value
BROADCAST   <= CWVAL(11);
TX          <= CWVAL(10);
SA          <= CWVAL(9 downto 5);
WCMC        <= CWVAL(4 downto 0);
ISMCODE     <= '1' when ( SA="00000" or SA="11111") else '0';

-- This process decodes the command word and sets CMDOKAY for legal command words.
PLEGAL:
process(BROADCAST,TX,SA,WCMC,ISMCODE)
 variable OK     : std_logic;
 variable MUXSEL : std_logic_vector(5 downto 0);
 begin

    if (ISMCODE='0') then
        -- Data transfers
        MUXSEL := TX & SA;
        OK := '0';           -- Default is disabled
        ------------------------------------------------------------------------
        -- This case statement legalizes data transfers to certain subaddresses
        case MUXSEL is
          when "111010" => OK := '0';                  -- SA 26 Disabled for TX
          when "111011" => OK := '0';                  -- SA 27 Disabled for TX
          when "011001" => OK := '0';                  -- SA 25 Disabled for RX
```

```
          when "011011" => if WCMC>0 and WCMC <10 then   -- SA 27 Disabled for RX if WC>9
                                              OK := '1';
                                           end if;
            when others   => OK := '1';  -- Legalize all other subaddresses
          end case;
          ------------------------------------------------------------------------
          -- Broadcast transmits are not allowed; overrides above case statement
          if BROADCAST='1' and TX='1' then
            OK := '0';  -- Broadcast transmit is not allowed
          end if;
       else
          ------------------------------------------------------------------------
          -- This case statement legalizes mode codes
          MUXSEL := TX & WCMC;
          OK := '1';       -- Default is OKAY
          case MUXSEL is
            when "100000" => -- Dynamic Bus Control
                             OK := '0';        -- Since we can't do it, we Message Error
            when "100001" => -- Synchronise
            when "100010" => -- Transmit Status Word
                             OK := not BROADCAST;
           when "100011" => -- Initiate Self-Test; we set this because we provide BIT word
                             OK := '1';
            when "100100" => -- Transmitter Shutdown
            when "100101" => -- Override Transmitter Shutdown
            when "100110" => -- Inhibit Terminal Flag
            when "100111" => -- Override Inhibit Terminal Flag
            when "101000" => -- Reset Remote Terminal
            when "110000" => -- Transmit Vector Word
                             OK := not BROADCAST;
            when "010001" => -- Synchronise with Data
            when "110010" => -- Transmit Last Command
                             OK := not BROADCAST;
            when "110011" => -- Transmit BIT Word
                             OK := not BROADCAST;
            when "010100" => -- Selected Transmitter Shutdown
                             OK := '0';
            when "010101" => -- Override Selected Transmitter Shutdown
                             OK := '0';
            when others   => -- All other commands illegal
                             OK := '0';
          end case;
    end if;
    CMDOKAY <= OK;
end process;

end RTL;
```

Verilog Example

```
module CWLEGALITY (CWVAL, CMDOKAY);
   input[11:0] CWVAL;
   output CMDOKAY;
   reg CMDOKAY;
   wire BROADCAST, ISMCODE, TX;
   wire[4:0] SA, WCMC;

   assign BROADCAST = CWVAL[11] ;   // Decode incoming Value
   assign TX = CWVAL[10] ;
   assign SA = CWVAL[9:5] ;
   assign WCMC = CWVAL[4:0] ;
   assign ISMCODE = (SA == 5'b00000 | SA == 5'b11111) ? 1'b1 : 1'b0 ;

   always @(BROADCAST or TX or SA or WCMC or ISMCODE)
```

```
begin : PLEGAL
   reg OK;
   reg[5:0] MUXSEL;
   if (ISMCODE == 1'b0)
   begin
      MUXSEL = {TX, SA};    // Data transfers
      OK = 1'b0;
      // This case statement legalizes data transfers to certain subaddresses
      case (MUXSEL)
         6'b111010 : OK = 1'b0;                  // SA 26 Disabled for TX
         6'b111011 : OK = 1'b0;                  // SA 27 Disabled for TX
         6'b011001 : OK = 1'b0;                  // SA 25 Disabled for RX
         6'b011011 : OK = (WCMC > 0 & WCMC < 10); // SA 27 Disabled for RX if WC>9
         default   : OK = 1'b1;                  // Legalize all other subaddresses
      endcase
     // Broadcast transmits are not allowed; overrides above case statement
      if (BROADCAST == 1'b1 & TX == 1'b1)
         OK = 1'b0; // Broadcast transmit is not allowed
   end
   else
   begin
      //-----------------------------------------------------------------
      // This case statement legalizes mode codes
      MUXSEL = {TX, WCMC};
      OK = 1'b1;
      case (MUXSEL)
         6'b100000 : // Dynamic Bus Control
                     OK = 1'b0; // Since we can't do it, we Message Error
         6'b100001 : // Synchronize
         6'b100010 : // Transmit Status Word
                     OK = ~BROADCAST;
         6'b100011 : // Initiate Self-Test; we set this because we provide BIT word
                     OK = 1'b1;
         6'b100100 : // Transmitter Shutdown
         6'b100101 : // Override Transmitter Shutdown
         6'b100110 : // Inhibit Terminal Flag
         6'b100111 : // Override Inhibit Terminal Flag
         6'b101000 : // Reset Remote Terminal
         6'b110000 : // Transmit Vector Word
                     OK = ~BROADCAST;
         6'b010001 : // Synchronise with Data
         6'b110010 : // Transmit Last Command
                     OK = ~BROADCAST;
         6'b110011 : // Transmit BIT Word
                     OK = ~BROADCAST;
         6'b010100 : // Selected Transmitter Shutdown
                     OK = 1'b0;
         6'b010101 : // Override Selected Transmitter Shutdown
                     OK = 1'b0;
         default   : // All other commands illegal
                     OK = 1'b0;
      endcase
   end
   CMDOKAY <= OK ;
end
endmodule
```

Modifying the Backend Address Map

The default setting of Core1553BRT creates 32 receive and 32 transmit subaddresses, each with 32 words of memory, which in turn requires 2,048 words of memory. Receive subaddresses 0 and 31 are used for storing either the 1553B command word or the TSW value.

You can use an external address mapping function to modify the backend address map to match the software models of legacy 1553B remote terminals, allowing software drivers to be reused (Figure 7-1). You can use the CMDVAL output to generate the mapped address function; however, it must be externally latched using the ADDRLAT signal (ADDRLAT is an active high enable for an external D-type flip-flop).

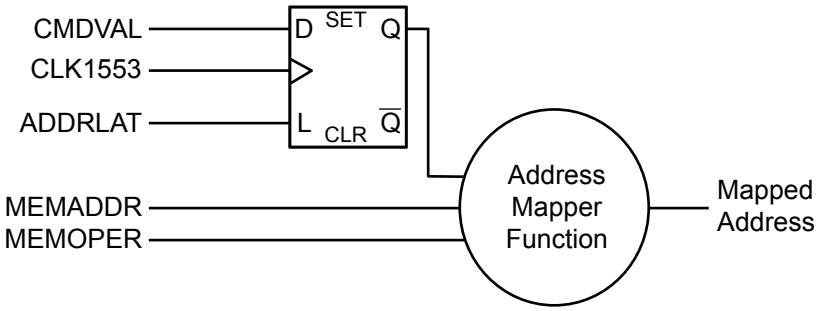


***Figure 7-1* • External Address Mapper Circuit**

A typical address mapping function, given below, remaps the backend memory map to an 8 k word memory. Each subaddress is allocated 64 words, and separate buffers are provided for both broadcast and non-broadcast receive/transmit.

Each non-mode-code subaddress consists of 64 words; the command word or TSW value is written to location 0, and the associated data words are stored at locations 32 to 63.

For the mode code subaddresses, the command word or TSW value is written to location 0 plus the mode code value, and the data word is stored at location 32 plus the mode code value. For the “transmit vector word” command, the command word is stored in location 16, and the vector word is read from location 48. For the “synchronize with data” command, the command word is stored at location 17 and the data written to location 49. Note that separate address spaces exist for transmit and receive mode codes and broadcast transmit and receive mode codes.

This mapping function is very small; it requires only 23 logic modules in the Microsemi SX-A and RTSX-S families. Consider the code below.

VHDL Address Mapping Function

```
entity ADDRESSMAPPER is
 port ( CLK        : in      std_logic;
        ADDRLAT    : in      std_logic;
        CMDVAL     : in     std_logic_vector(11 downto 0);
        MEMADDR    : in     std_logic_vector(10 downto 0);
        MEMOPER    : in     std_logic_vector( 1 downto 0);
        MAPADDR    : out   std_logic_vector(12 downto 0)
      );
end ADDRESSMAPPER;

architecture RTL of ADDRESSMAPPER is
signal CMDADDR : std_logic_vector(11 downto 0);
begin

process(CLK)
 begin
  if CLK'event and CLK='1' then
    if ADDRLAT='1' then
```

```
      CMDADDR <= CMDVAL;
    end if;
  end if;
end process;

process(CMDADDR,MEMADDR,MEMOPER)
variable SA          : std_logic_vector(4 downto 0);
variable WCCW        : std_logic_vector(4 downto 0);
variable WCAD        : std_logic_vector(4 downto 0);
variable MC          : std_logic;
variable BCAST       : std_logic;
variable TX          : std_logic;
variable MSEL        : std_logic_vector( 2 downto 0);
begin
 SA    := CMDADDR(9 downto 5);
 WCCW  := CMDADDR(4 downto 0);
 WCAD  := MEMADDR(4 downto 0);
 BCAST := CMDADDR(11);
 TX    := CMDADDR(10);
 if (SA="00000" or SA="11111") then
   MC := '1';
 else
   MC := '0';
 end if;
 MSEL  := MEMOPER & MC;
 case MSEL is
   when "100"   => MAPADDR <= BCAST & TX & SA & '0' & "00000";  -- CW Data Transfer
   when "101"   => MAPADDR <= BCAST & TX & SA & '0' & WCCW;     -- CW Mode Code
   when "000"   => MAPADDR <= BCAST & TX & SA & '1' & WCAD;     -- DW Transfer
   when "001"   => MAPADDR <= BCAST & TX & SA & '1' & WCCW;     -- DW Mode Code
   when others  => MAPADDR <= ( others => '-');
 end case;
end process;
end RTL;
```

Verilog Address Mapping Function

```
module ADDRESSMAPPER (CLK, ADDRLAT, CMDVAL, MEMADDR, MEMOPER, MAPADDR);
   input CLK;
   input ADDRLAT;
   input[11:0] CMDVAL;
   input[10:0] MEMADDR;
   input[1:0] MEMOPER;
   output[12:0] MAPADDR;

   reg[12:0] MAPADDR;
   reg[11:0] CMDADDR;

   always @(posedge CLK)
   begin
     if (ADDRLAT == 1'b1) CMDADDR <= CMDVAL ;
   end

   always @(CMDADDR or MEMADDR or MEMOPER)
   begin
      reg[4:0] SA;
      reg[4:0] WCCW;
      reg[4:0] WCAD;
      reg MC;
      reg BCAST;
      reg TX;
      reg[2:0] MSEL;
      SA    = CMDADDR[9:5];
      WCCW  = CMDADDR[4:0];
```

```
        WCAD  = MEMADDR[4:0];
        BCAST = CMDADDR[11];
        TX    = CMDADDR[10];
        if (SA == 5'b00000 | SA == 5'b11111)
           MC = 1'b1;
        else
           MC = 1'b0;
        MSEL = {MEMOPER, MC};
        case (MSEL)
          3'b100  : MAPADDR <= {BCAST, TX, SA, 1'b0, 5'b00000} ; // Command Data Transfer
           3'b101  : MAPADDR <= {BCAST, TX, SA, 1'b0, WCCW} ; // Command Mode Code
           3'b000  : MAPADDR <= {BCAST, TX, SA, 1'b1, WCAD} ; // Data Data Transfer
           3'b001  : MAPADDR <= {BCAST, TX, SA, 1'b1, WCCW} ; // Data Mode Code Transfer
           default : MAPADDR <= {13{1'bx}} ;
        endcase
    end
endmodule
```

Modifying the Backend Interrupt Vector

The default setting of Core1553BRT creates a 7-bit interrupt vector, which indicates whether or not a good message was received. It is also used for a transmitter subaddress. If INTENBBR (Interrupt Enable Bad Block Received) is HIGH, interrupts are generated for good and bad 1553B messages. When INTENBBR is LOW, interrupts are only generated for messages that are received or transmitted correctly.

An external interrupt mapping function can be used to add extra information to the interrupt vector (Figure 7-2), such as the received word count. It will also tell you whether the command was broadcast. The CMDVAL output can be used to generate this extra information; however, it must be externally latched using the INTLAT signal (INTLAT is an active high enable for an external D-type flip-flop).

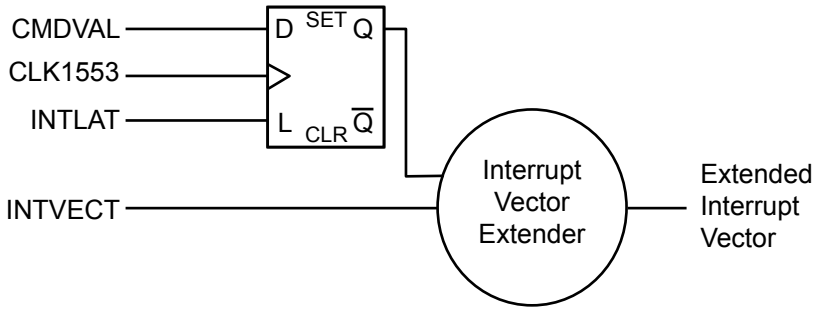


Figure 7-2 • **External Interrupt Vector Extender Circuit**

The interrupt mapping function below creates vector information, indicating broadcast and the received word count value. If an extended interrupt vector is then generated, the need for the system to read the TSW or command word from memory is removed. The interrupt vector provides subaddress, word count data, and an indication that the message was good. In this case, the WRTTSW and WRTCMD inputs can be tied LOW.

Consider the Verilog extended interrupt vector generation example below:

```
module INTVECTEXTENDER (CLK, INTLAT, CMDVAL, INTVECT, MAPVECT);
   input CLK;
   input INTLAT;
   input[11:0] CMDVAL;
   input[6:0] INTVECT;
   output[12:0] MAPVECT;
   reg[12:0] MAPVECT;
   reg[11:0] CMDINT;

   always @(posedge CLK)
   begin
     if (INTLAT == 1'b1)
         CMDINT <= CMDVAL ;
   end

   always @(CMDINT or INTVECT)
   begin
      reg[4:0] SA;
      reg[4:0] WC;
      reg BCAST;
      reg TX;
      reg GBR;
      WC    = CMDINT[4:0];
      BCAST = CMDINT[11];
      GBR   = INTVECT[6];
      TX    = INTVECT[5];
      SA    = INTVECT[4:0];
      MAPVECT <= {GBR, BCAST, TX, SA, WC} ;
   end
endmodule
```

Or, if you are using VHDL, consider the VHDL extended interrupt generation function example code:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity INTVECTEXTENDER is
 port ( CLK         : in   std_logic;
        INTLAT      : in   std_logic;
        CMDVAL      : in   std_logic_vector(11 downto 0);
        INTVECT     : in   std_logic_vector( 6 downto 0);
        MAPVECT     : out  std_logic_vector(12 downto 0)
      );
end INTVECTEXTENDER;

architecture RTL of INTVECTEXTENDER is
signal CMDINT : std_logic_vector(11 downto 0);
begin

process(CLK)
 begin
  if CLK'event and CLK='1' then
    if INTLAT='1' then
      CMDINT <= CMDVAL;
    end if;
  end if;
end process;

process(CMDINT,INTVECT)
variable SA         : std_logic_vector(4 downto 0);
variable WC         : std_logic_vector(4 downto 0);
variable BCAST      : std_logic;
```

```
variable TX          : std_logic;
variable GBR         : std_logic;
begin
 WC    := CMDINT(4 downto 0);
 BCAST := CMDINT(11);
 GBR   := INTVECT(6);
 TX    := INTVECT(5);
 SA    := INTVECT(4 downto 0);
 MAPVECT <= GBR & BCAST & TX & SA & WC;
end process;
end RTL;
```

Connecting the Backend to Internal FPGA Memory

When implementing the core in flash-based FPGA or Axcelerator devices, you can directly connect the core to the flash-based FPGA or Axcelerator memory blocks. Use two memory blocks (1k×16 each—one for transmit and the other for receive memory), as shown in Figure 7-3.

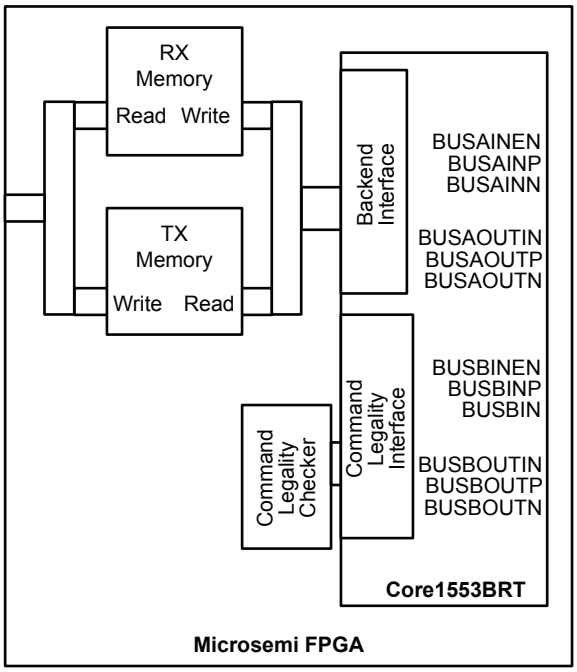


Figure 7-3 • **Using Internal FPGA RAM Modules**

The core must be in synchronous mode (ASYNCIF inactive—LOW). The MEMGNTn input should be tied active (LOW) and the MEMWAITn input inactive (HIGH). This allows the backend to have immediate access to the memory blocks.

Buffer Management

The core implements basic buffer management techniques for the 1553B message protocol. Receive and transmit buffers are provided. This approach requires the backend system to empty and fill the buffers promptly. For a “broadcast receive” command, the core generates an interrupt, indicating that a receive buffer is available as the last data word is written to memory. At the same time, another receive command to the same subaddress could be on the bus, which causes the first word in the memory to be overwritten within 20 µs, depending on the backend request grant times. This time could be reduced to less than 5 µs if the backend delays the core’s access to the memory for the last data word. It also allows immediate access for the first data word in the following message.

The following implementation (Figure 7-4 on page 58) implements an extra buffer level in the backend. The core writes the received data to the small 32×16 memory block. At the end of the transfer, the backend RX state machine captures the TSW value, then bursts the complete receive packet into main memory. This system enables the complete message to be copied to main memory once completely validated. It will stay intact in main memory until the complete following message of at least 40 µs is received.

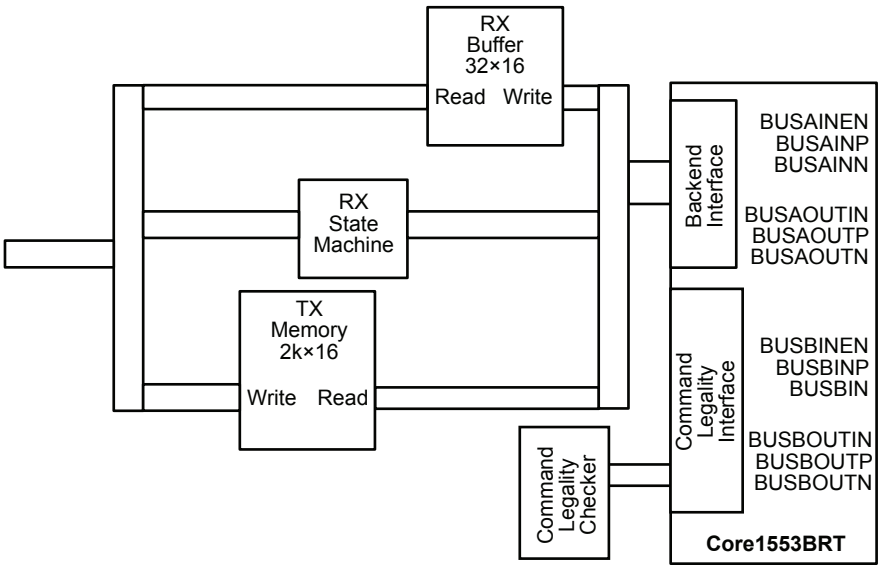


Figure 7-4 • **Buffer Management Circuit**

Buffer management techniques are system-dependent. They depend on the bus traffic scheduling by the bus controllers. The Core1553BRT implementation is intended to provide a flexible interface that can be adapted to the system requirements.

Bus Transceivers

Core1553BRT does not include the transceiver that is required to drive the 1553B bus. Core1553BRT is designed to interface directly to MIL-STD-1553 transceivers. There are several suppliers of MIL-STD-1553 transceivers, such as DDC (BU-63147) and Aeroflex (ACT4402).

When using Fusion, IGLOO/e, ProASIC3/E, ProASIC^{PLUS}, or Axcelerator FPGAs, level translators are required to connect the 5 V outputs of the 1553B transceivers to the 3.3 V inputs of the FPGA.

In addition to the transceiver, a pulse transformer is required for interfacing to the 1553B bus. Figure 7-5 on page 59 and Figure 7-6 on page 60 show the connections required from Core1553BRT to the transceivers and then to the bus via pulse transformers.

Typical RT Systems

Core1553BRT can be used in systems with and without backend memory. Figure 7-5 shows a typical implementation for a system with backend memory and a CPU to process the messages. Figure 7-6 on page 60 shows a system with direct connection between Core1553BRT and external analog to digital converters, etc. In this case, any glue logic required between the core and the device being interfaced to can simply be implemented within the FPGA containing the core.

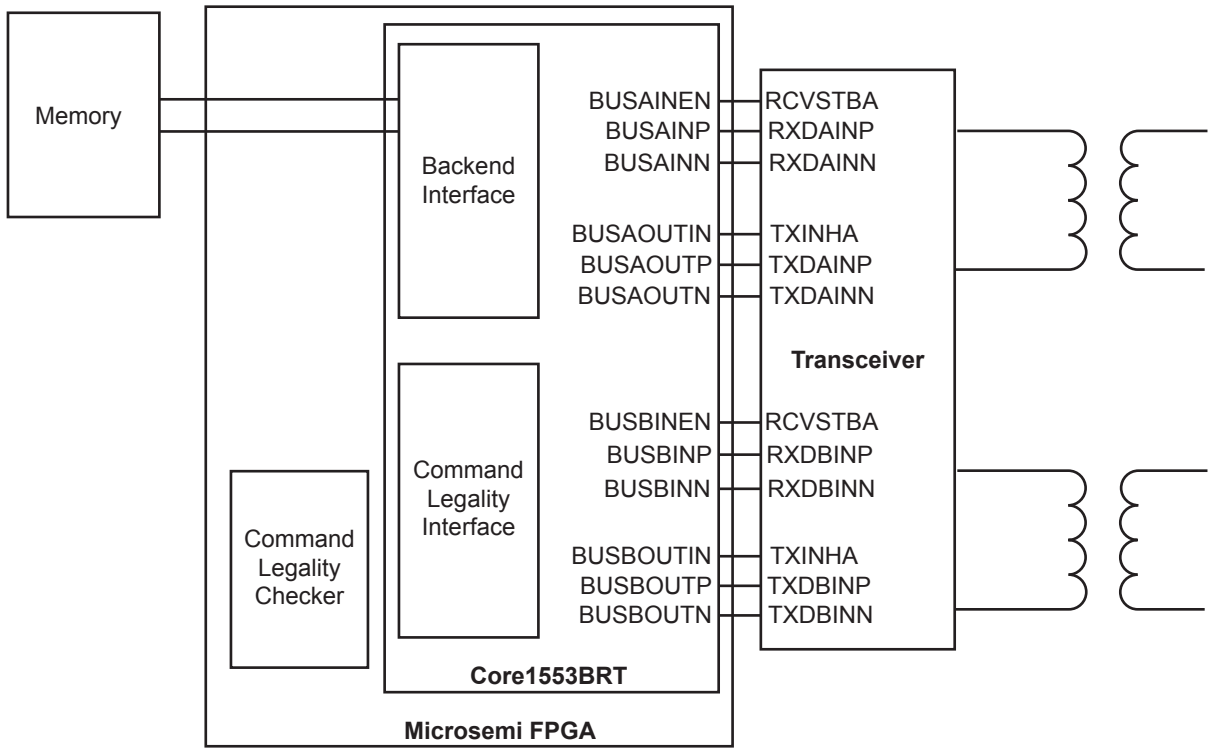


Figure 7-5 • **Typical CPU- and Memory-Based RT System**

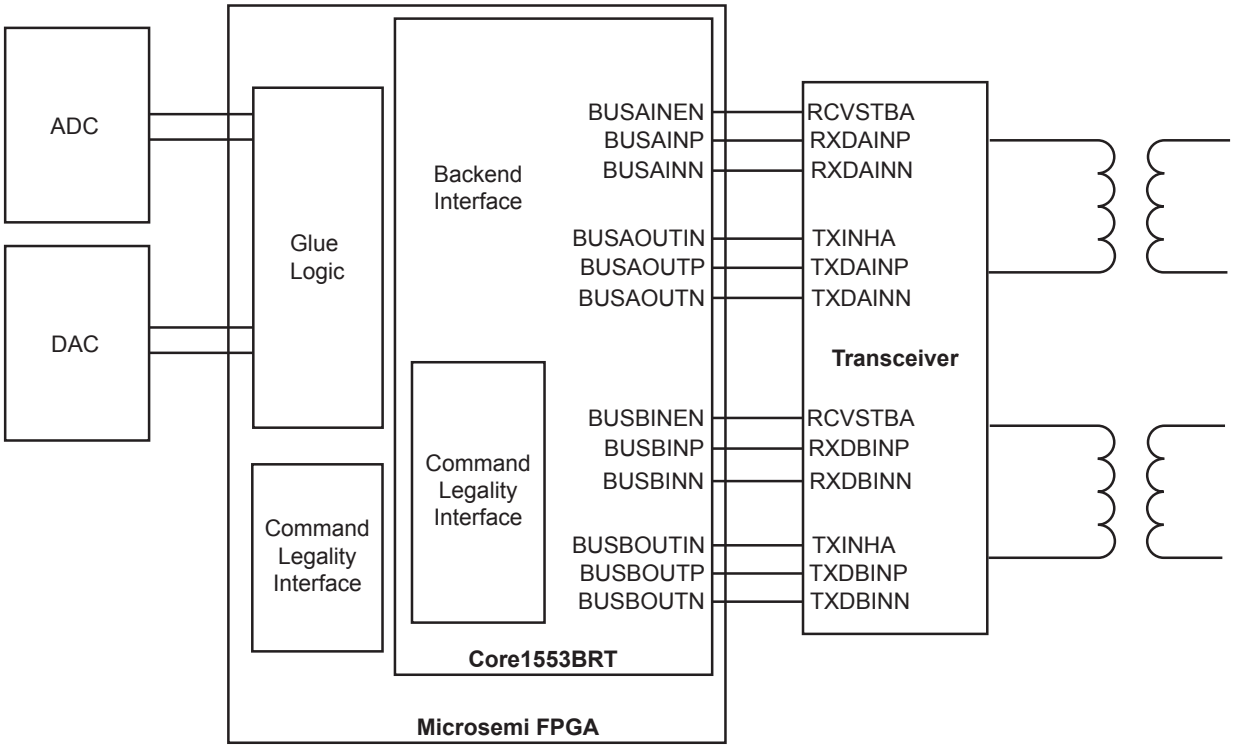


***Figure 7-6* • Typical Non-Memory-Based RT System**

8 – VHDL Testbench Procedure and Function Calls

Both the verification and VHDL user testbenches provided with the core include some low-level support routines that you can use to verify your 1553BRT. These are available in the three packages that can be accessed by adding the following four lines to your VHDL source code:

```
library Core1553B;
use Core1553B.misc.all;
use Core1553B.textio.all;
use Core1553B.test1553b.all;
```

These routines make use of the following types, which are also declared in these packages:

```
subtype INT1BIT    is integer range 0 to 1;
subtype INT5BIT    is integer range 0 to 31;
subtype NIBBLE     is std_logic_vector ( 3 downto 0);
subtype BYTE       is std_logic_vector ( 7 downto 0);
subtype WORD       is std_logic_vector (15 downto 0);
type BYTE_ARRAY    is array ( INTEGER range <>) of BYTE;
type WORD_ARRAY    is array ( INTEGER range <>) of WORD;
type PACKET        is array ( INTEGER range 0 to 31) of WORD;
```

The two record structures used in the encoder decoder module are as follows:

```
type TQMSGREQ is
 record
    RT      : INT5BIT;
    TX      : INT1BIT;
    SA      : INT5BIT;
    MCWC    : INTEGER range 0 to 32;
    RTRT    : BOOLEAN;
    RT2     : INT5BIT;
    SA2     : INT5BIT;
    DATA    : PACKET;
end record;

type TQMSGOUT is
 record
    OKAY    : BOOLEAN;
    COUNT   : INTEGER;
    CW1     : WORD;
    CW2     : WORD;
    SW1     : WORD;
    SW2     : WORD;
    DATA    : PACKET;
end record;
```

The following type conversion routines are provided to convert between integer, std_logic, and Boolean types:

```
function  to_slv1bit( x: INT1BIT ) return std_logic_vector;
function  to_sl1bit( x: INT1BIT ) return std_logic;
function  to_slv5bit( x: INT5BIT ) return std_logic_vector;
function  to_int1bit( x: boolean ) return INT1BIT;
function  to_int1bit( x: std_logic ) return INT1BIT;
function  to_int1bit( x: std_logic_vector  ) return INT1BIT;
function  to_int5bit( x: std_logic_vector ) return INT5BIT;
function  to_byte  ( x : INTEGER ) return BYTE;
function  to_word  ( x : INTEGER ) return WORD;
```

The following function call is provided to create data patterns. This returns a data packet whose first value is set to the seed value and whose subsequent values are incremented by the delta value. If the delta value is zero, all values are the same. For example:

```
function  initdata( SEED : INTEGER; DELTA : INTEGER) return PACKET;
```

A procedure is provided that compares data packets and displays the error when the compare fails. For example:

```
procedure ComparePacket( STR   : STRING;
    EXP     : Packet;
    GOT     : Packet;
    LEN     : Integer;
    ERRCNT  : inout INTEGER );
```

The first parameter is a text string that is printed when a failure occurs, the second parameter is the expected data, and the third parameter is the actual data. The fourth parameter indicates how many of the words are to be compared (the packet contains 32 words). Finally, the fifth parameter is a global error counter that is incremented if a compare fails.

A procedure is provided that displays the received message record structure. This procedure requires both the message request and output record structures so that it can format the printed data correctly. For example:

```
procedure print_msgout( QMSG : TQMSGREQ; Q : TQMSGOUT);
```

To support general printing, a printf routine is provided that supports printing std_logic, Boolean, integer, and std_logic_vector types. The syntax is slightly different from the C-language printf function. For example:

```
printf("Hello World Decimal %d Hex %x Hex4 %04x Bits %04b A string %s",
    fmt(256)&fmt(WORD)&fmt(WORD)&fmt(NIBBLE)&fmt(STRING));
```

prints

```
Hello World Decimal 256 Hex 100 Hex4 0100 Bits 1010 A string thestring
```

List of Changes

The following table lists critical changes that were made in each revision of the document.

Date	Changes	Page
Revision 4 (February 2015)	Updated the document for v4.1	N/A
Revision 3 (January 2014)	Updated the document for v4.0.	N/A
Revision 2 (August 2010)	The core version changed from v3.2 to v3.3.	6
	Information relating to the external BIST interface was added at the end of the "General Description" section.	4
	Figure 2-1 • Core1553BRT Configuration within SmartDesign was replaced.	10
	The INITLASTSW and EXTERNAL_BIST parameters were added to Table 3-1 • Core1553BRT Parameters.	12
	The description for RTADDR[4:0] was revised in Table 3-2 • 1553B Bus Interface.	14
	The following port names were added to Table 3-3 • Control and Status Signals: • PURSTN • BITINEN • BITIN[15:0]	14
	The description for CMDSTB was revised in Table 3-4 • Command Legalization Interface.	16
	In Table 5-6 • BIT Word, the value of bits [4:0] for Core1553BRT v3.3 was added.	32
	The "Connecting the Backend to Internal FPGA Memory" section was revised to change "ProASIC3 or Axcelerator devices" to "flash-based FPGA or Axcelerator devices."	57

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

50200093-4/02.15