Designing a MIL-STD-1553 System Using Core1553 and Core8051

Introduction

MIL-STD-1553 is a command/response, time-multiplexed, serial data bus with a 1 Mbit/sec data rate. The bus contains a bus controller and up to 31 remote terminals. Actel Core1553 cores meet all requirements for dual-redundant bus operation. Actel has developed flexible MIL-STD-1553 bus controller (BC), remote terminal (RT), and monitor terminal (MT) cores for use with its devices in high reliability applications. Table 1 shows variations of Actel MIL-STD-1553 solutions.

Table 1 • **Actel Core1553 Variations**

Core1553 Variation	Description
Core1553BRT	Includes MIL-STD-1553B remote terminal functionality
Core1553BBC	Includes MIL-STD-1553B bus controller functionality
Core1553BRM	Includes MIL-STD-1553A/B bus controller, remote terminal, and monitor terminal functionality

Bus interfaces such as MIL-STD-1553 (Core1553), ARINC 429 (Core429), and Ethernet MAC (Core10/100) are used in systems in which there is a host processor or controller. MIL-STD-1553, with a 1 Mbit/sec data rate, is not particularly a fast bus interface. Therefore, a simple 8-bit microcontroller fulfills the requirements of the host processor. Both bus interfaces and the system host usually require backend memory for their operation. A single backend memory space may belong to both host and bus interface.

When used together in a system, Core8051 and Core1553 may share the backend memory space. This document describes different implementations of such applications. As discussed in other sections of this document, the structure of these implementations may depend on the Core1553 variation (BBC, BRT, or BRM) and other system requirements. Furthermore, Core1553 addresses a different memory space size/ configuration than Core8051, which addresses a 64Kx8 memory space. Hence, some implementations may require additional considerations for proper functionality.

The memory access requirements are very low in a MIL-STD-1553B bus interface. The 1553B cores will typically perform approximately six to eight accesses to start a message, then one access every 20 µs. In a worst case scenario, this is eight accesses every 20 µs, or one access every 2.5 µs, and at 12 MHz this is one access per 30 clock cycles. As a result, it is very convenient to share a single memory between the host processor and Core1553 without affecting the overall system performance.

This document discusses multiple implementations that can be used when Core8051 and Core1553 share the backend memory space. The sections are based on the Core1553 variations (BRT, BRM, and BBC). Each section is also broken down based on different memory sharing implementations.

Core8051 and Core1553BRT

Core1553BRT, Remote Terminal (RT) variations of Core1553, can connect to the backend memory space directly or through a memory request/grant protocol. Furthermore, this core does not include a CPU interface. Therefore, if the core accesses the memory space directly, the CPU should have access to the memory as well. This can be achieved by using dual-port memories.

Core1553 is equipped with memory request/grant I/Os which can be used when Core8051 and Core1553 share the same bus. In this case, a bus arbiter is required to turn over the control of the memory bus to Core1553 and Core8051 accordingly. Figure 1 on page 2 depicts the overall architecture of the above implementations.

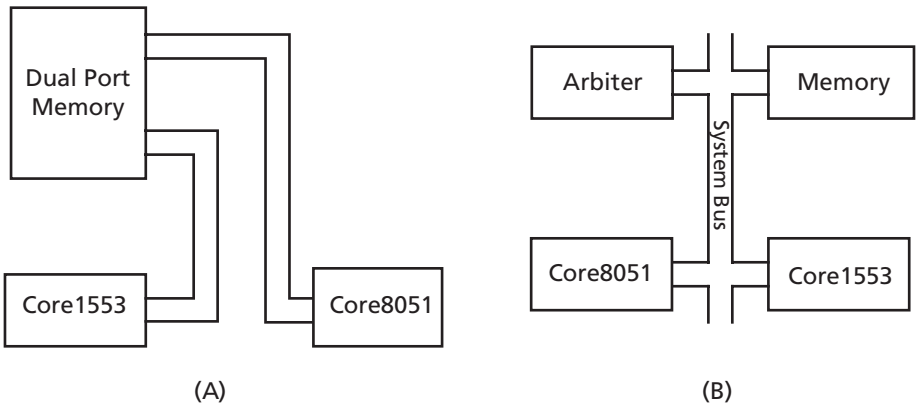

Figure 1 • **System Level Architecture of the Core1553-Core8051 Backend Memory System**

Figure 1 (A) shows a system in which a dual-port memory provides two separate connections for Core8051 and Core1553. In the Figure 1 (B) implementation, a single bus serves the system. The bus control is overseen by the arbiter, which provides access to the bus for Core1553 and Core8051, respectively. In all implementations, if the backend uses the embedded synchronous memories, the ASYNCIF input of Core1553 should be tied low. The architectures, depicted in Figure 1, are discussed in the following sections.

Dual-Port Memory Implementation

In this implementation, a dual-port memory is used to provide direct access to memory space for both Core1553 and Core8051. Figure 2 depicts the usage of a dual-port memory in a system which utilizes both Core8051 and Core1553BRT.

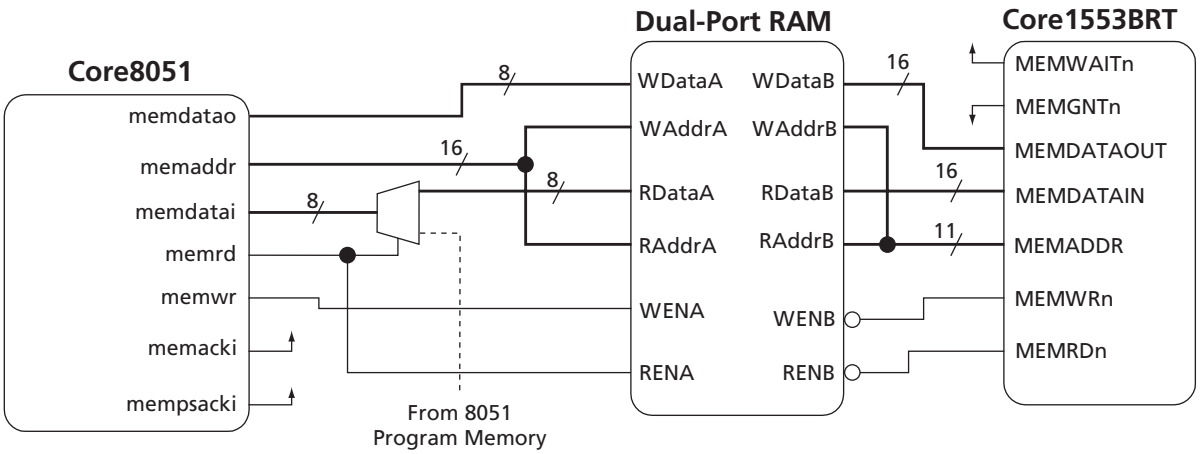

Figure 2 • **Using Dual-Port Memory Space For Core8051 and Core1553BRT**

In the Figure 2 implementation, port A of the memory is configured to address a 64Kx8 memory space, and provides direct access to memory for Core8051. The dotted part of the implementation indicates optional circuitry, which is used to connect the external CPU program memory to the core. Since the CPU has direct access to the memory, memacki, and mempsacki inputs of Core8051 are tied to logic high.

Port B of the dual-port RAM in Figure 2 is configured as a 2Kx16 memory space and solely serves Core1553BRT memory access ports. Since Core1553BRT has unlimited access to the memory space in this configuration, MEMGNTn and MEMWAITn inputs are tied to logic low and high, respectively.

Table 2 lists Actel FPGAs that contain embedded SRAM memories.

Table 2 • **Actel FPGA Families with Embedded SRAM Blocks**

FPGA Family	Dual-Port RAM	Single-Port RAM
ProASIC3E	Yes	Yes
ProASIC3	Yes	Yes
ProASIC^{PLUS}	No	Yes
Axcelerator	No	Yes
RTAX-S	No	Yes
ProASIC	No	Yes

For devices that do not support embedded dual-port RAM blocks, single-port RAM blocks can be used to implement dual-port memory blocks[1].

Core1553BRT and Core8051 Direct Access To Single-Port Memory

Core1553BRT addresses a 2Kx16 memory space while it only writes into the lower half (address 0x000 to 0x3FF) and reads from the upper half of the memory space (0x400 to 0x7FF). This allows for an implementation in which both Core1553BRT and the host processor access the memory space directly if single-port RAM blocks are used. In this implementation, the host processor has write access only to the upper half, and it can only read from the lower half of the memory space. Since Core8051 performs byte-size memory access and Core1553BRT supports 16-bit word-size memory access, the memory space should be configured accordingly to provide direct access to both Core1553BRT and Core8051. A simple example of such an implementation is depicted in Figure 3.

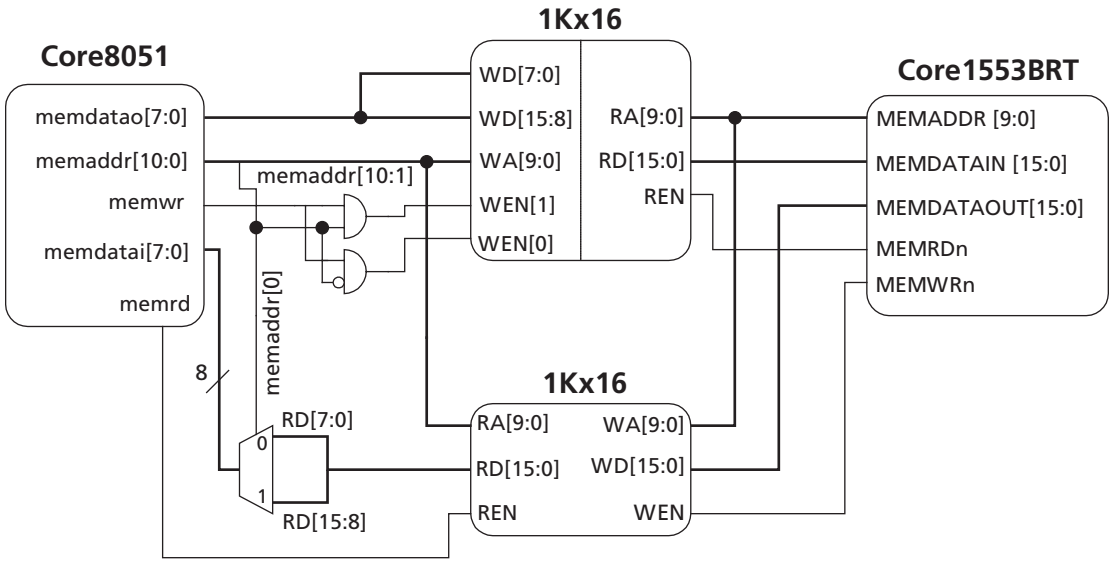

Figure 3 • **Design Example with Direct Access to Memory**

1. *For more information, refer to the following documents:*
   Implementing Multi-Port Memories in ProASIC^{PLUS} Devices
   Implementing Multi-Port Memories in Axcelerator Devices

In the Figure 3 on page 3 implementation, the 2Kx16 memory is configured as two blocks of 1Kx16, since each core needs write and read access to half of the memory space. Core8051 can only perform byte-size writes, so the upper half of the memory (address space 0x400 to 0x7FF) should support byte-size write operations (depicted by the dotted line in Figure 3 on page 3 and two WEN ports for the upper half). In Core8051 read operations, the memaddr[10:1] drives the address input of the lower half of the memory, and memaddr[0] selects the upper or lower byte of the data word to be input to Core8051.

On the Core1553BRT side, since the core supports word-size data transactions, the connections are straightforward, as shown Figure 3 on page 3. The most significant bit of MEMADDR output of the core (MEMADDR[10]) is left unused. This is because the core has read (and write) access to half of the memory space.

This application note contains a design example for the backend memory architecture, shown in Figure 3 on page 3, along with the peripheral glue logic. The design example is simply a 2Kx16 memory space that can connect directly to Core8051 on one side and Core1553BRT on the other side.

The design example utilizes ProASIC^{PLUS} embedded SRAM blocks, generated by ACTgen. Users can easily replace these building blocks with the memory blocks of other FPGAs or any other external SRAM blocks.

In the implementation shown in Figure 3 on page 3 or any other implementation that simply allows direct access to both Core1553BRT and Core8051 to the same memory space, it is possible that Core1553BRT and Core8051 (or any other host processor) read and write in the same location at the same time. This may be especially troublesome when Core1553BRT is in the burst read or write operation. For example, if the host processor updates a memory location in the space that is being read by Core1553BRT in the burst mode, some memory locations read by Core1553BRT will contain old data while other locations contain newer data. Such conflicts should be avoided by the system software. In other words, the system software should prevent the Core1553BRT and host processor from accessing the same memory address range at the same time. There are many different implementations to perform this; however, the best fitted solution completely depends on each specific system application.

Arbitration

Core1553BRT has a memory request/grant port which can be used to communicate with a backend bus arbiter. In this case, both Core8051 and Core1553 connect to the bus arbiter and the arbiter turns over the control of memory ports respectively. Figure 4 on page 5 simply depicts the usage of a bus arbiter in a Core8051-Core1553 system. The memory is configured as 32Kx16. However, the arbiter provides byte read/write capability to Core8051.

As can be seen in Figure 4 on page 5, the memory is configured as 16 bits wide. Therefore, byte read/write capability should be provided for Core8051. The byte write capability can be simply implemented by connecting Core8051 memdatao[7:0] to the RAM wdata[15:8] and wdata[7:0]. When even addresses are written, the arbiter pulses the write on memory wdata[D7:0], while odd addresses pulse the write on memory wdata[15:8]. The following VHDL code implements this functionality:

```
RAMWRITELOW   <= memwr and not memaddr[0];
RAMWRITEHIGH <= memwr and memaddr[0];
WAddr[14:0]  <= memaddr(15 downto 1);
```

In the code above, memwr is the memwr output of Core8051, memaddr is the memaddr output of Core8051, and WAddr is the write address input of system memory space.

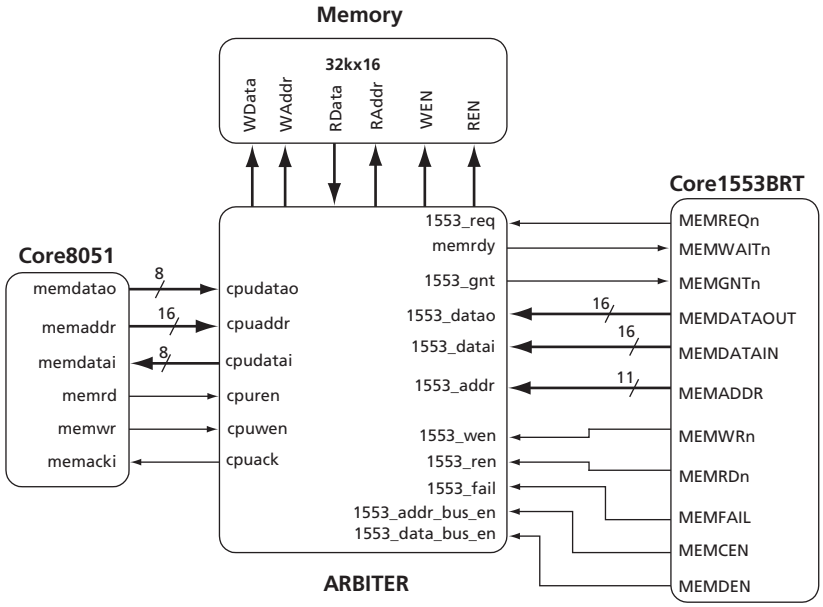

Figure 4 • **Using a Bus Arbiter in the Core8051-Core1553 System**

The byte read capability is implemented by using a MUX architecture. Figure 5 shows a sample byte read implementation.

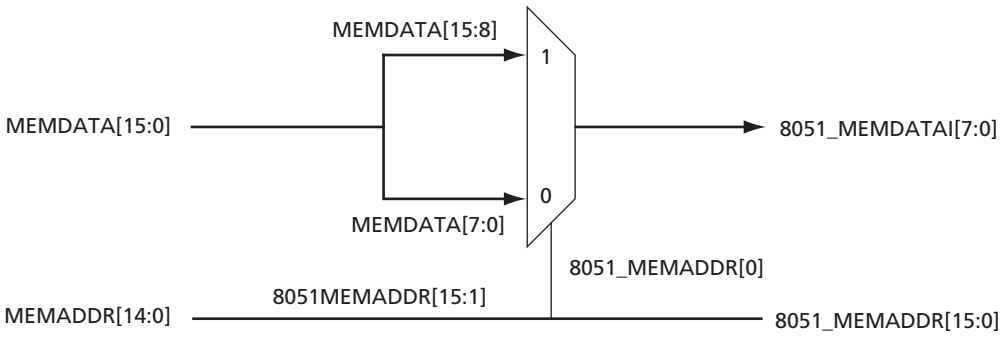

Figure 5 • **Byte Read Implementation for Core8051**

Additional outputs of Core1553 that can be used are MEMFAIL, MEMDEN and MEMCEN.

MEMFAIL is an output of the Core1553, which is activated by the core if the memory transaction is unsuccessful. MEMCEN and MEMDEN are also active high control signals that are activated during memory transaction. They can be used in a bus system to activate/deactivate the tristated address buffers and bidirectional data buffers if needed.

Core8051 and Core1553BBC/BRM

The main difference in the backend memory interface between Core1553BBC/BRM[2] and Core1553BRT is the address space. Core1553BBC and Core1553BRM address a 64Kx16 memory space compared to 2Kx16 in Core1553BRT. Furthermore, Core1553BBC and Core1553BRM provide a CPU interface that can be used to connect Core8051 to the backend memory. This section of the application note discusses multiple implementations that can be used if Core8051 and Core1553 use the same memory space in the system. Similar to Core1553BRT, if the backend uses the embedded synchronous memory space (internal to the FPGA), the ASYNCIF input of Core1553 should be tied low.

Dual-Port Memory Implementation

Similar to the RT or MT variation of Core1553BRM, a dual-port RAM can be used to provide direct access to memory space for both Core8051 and Core1553. The only difference, as discussed above, is the address space of Core1553. To fulfill this in the scheme shown in Figure 1 on page 2, the read and write address bus width of port B of the memory should be 16-bit.

Arbitration

Similar to the RT variation of the core, Core1553 is equipped with memory request/grant ports. These ports can be used by a bus arbiter to provide memory access to Core1553 when requested. A similar scheme is described in the "Core8051 and Core1553BRT" section on page 1. The same implementation can be used for Core1553BRM[3]. The only difference would be the larger memory address space in the bus controller of Core1553, compared to the remote terminal of Core1553. Therefore, the system memory should be configured as a 64Kx16 memory.

Core1553BBC/BRM CPU Interface

Core1553BBC and Core1553BRM feature a CPU interface. This interface allows designers to implement a different configuration when both Core1553 and Core8051 access the same memory space in a system. In this configuration, Core1553BRM or Core1553BBC act as bridges to connect Core8051 to memory. In other words, the CPU should go through the bus interface core to access the memory space. Table 7 of the *Core1553BRM MIL-STD-1553 BC, RT, and MT* datasheet describes the CPU interface ports of the core. Figure 6 shows how the CPU interface of Core1553BRM should be connected to Core8051 to provide access to memory space for the host microprocessor controller.

The CPUWRn[1:0] input of the core provides byte-size write capability to the CPU. If CPUWRn[0] is active (low), the data will be written in the lower half of memory width (CPUDIN[7:0]). If CPUWRn[1] is active (low), the data will be written in the upper half of memory width (CPUDIN[15:8]). In the Figure 6 on page 7 implementation, CPUWRn[1:0] is defined as the following:

```
CPUWRn[0] <= memaddr[0] and memwr;

CPUWRn[0] <= ! memaddr[0] and memwr;

CPUADDR[14:0] <= memaddr[15:1]
```

In this case, the CPUADDR[15] input of Core1553BRM can be grounded so that the CPU has access only to the upper half of backend memory space.

As shown in Figure 6 on page 7, Core8051 byte-size read is supported using a MUX architecture. For even memaddr values, the lower half of the data word (CPUDOUT[7:0]) is sent to memdatai input of the CPU. Odd values of memaddr during read operation will cause the upper half of the data word (CPUDOUT[15:8]) to propagate to the memdatai input of Core8051.

2. *This section applies to Core1553BRM if it is not configured as a remote terminal only. In that case, Core1553BRM is used as only a remote terminal, and it should be treated similarly to Core1553BRT.*
3. *MEMREQn and MEMGNTn work in a slightly different way on the Core1553BRM than on the Core1553BRT and Core1553BBC cores. Refer to the core datasheets for details.*

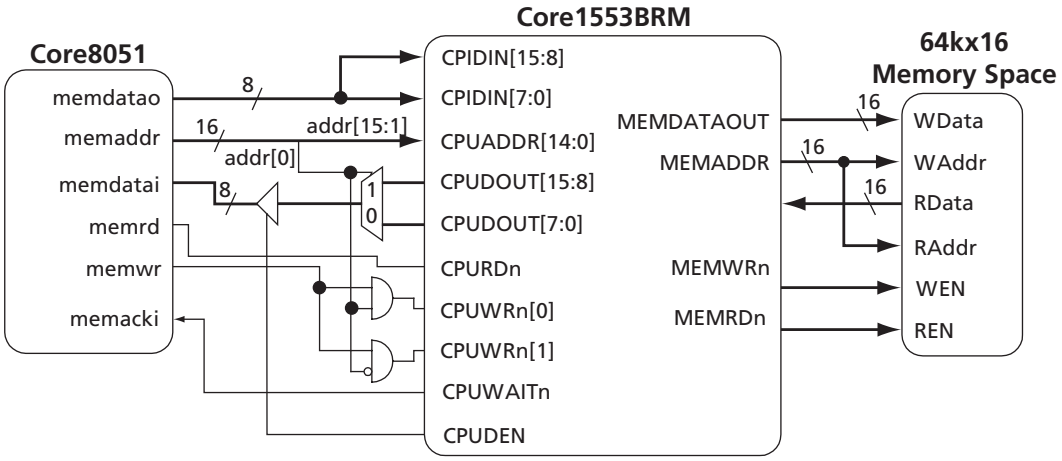

Figure 6 • **Using Core1553BRM CPU Interface with Core8051**

As shown in Figure 6, CPUDEN can be used as a tristate enable signal to memdatai input. This is extremely useful if memdatai is also connected to the CPU program memory.

If the CPU only accesses the memory space, the CPUMEM input of Core1553 should be tied to logic high. Otherwise it can be controlled through a MUX architecture. The select input of the MUX should transfer logic '1' to CPUMEM input if memrd or memwr outputs of the CPU are active.

Using Core8051 SFR Space with the Core1553BBC/BRM CPU Interface

Core1553BRM and Core1553BBC support up to 64 k words (128 kbytes) of memory space. The amount of memory space that is needed is system-dependent. A very simple BC function that is required to send a 32-word message to an RT could be implemented with only 64 words (128 bytes) of memory. On the other hand, a BC function that needs to transmit and receive from 31 RTs each, with multiple sub-addresses, could easily require the complete 128 kbytes of memory supported by the BBC and BRM cores.

When using an 8-bit processor such as Core8051, directly mapping this 128 kbytes of address space into the processor address space is not possible since Core8051 only supports 64 kbytes of data memory space. Even if only a quarter (32 kbytes) of the available memory space is implemented, it still can create a memory allocation problem. The complete 64 k of data memory cannot be allocated to Core1553, as some memory space is required for system memory and other system functions. An alternative to directly mapping the Core1553 address space to the processor address space is implementing a memory interface block that uses 8-byte wide processor locations to access the complete 64 k words of Core1553 space. These 8-byte processor locations can also be mapped to the Core8051 SFR memory space, totally removing the requirement for the Core1553 memory to use the spare processor data memory resources.

This implementation requires eight SFR spaces, as described in Table 3.

Table 3 • **Description of SFR Space Connecting to MIL-STD-1553 Bus Controller**

SFR Space	Description
SFR0	Upper half of memory address (A [15:8])
SFR1	Lower half of memory address (A [7:0])
SFR2	Upper byte of memory write data word (WD [15:8])
SFR3	Lower byte of memory write data word (WD [7:0])
SFR4	Control register
SFR5	Upper byte of memory read data word (RD [15:8])
SFR6	Lower byte of memory read data word (RD [7:0])
SFR7	Status register

The control and status register bits are described in Table 4 and Table 5, respectively.

Table 4 • **Bit Description of Control SFR**

Bits	Signal	Description
0	CPUMEM	CPU sets this signal when requiring memory access
1	WEN	CPU sets this signal when writing into memory
2	REN	CPU sets this signal when reading from memory
7:3	Unused	N/A

Table 5 • **Bit Description of Status SFR**

Bits	Signal	Description
0	Busy	Input to CPU. High when memory is not available
7:1	Unused	N/A

When performing a write into the memory, the CPU should load the address and data into SFR0 to SFR4. Then it should load the control register with a memory write request and check the status register. When the status register is cleared, the CPU can move to its next operation. The following C instruction set depicts a simple memory write access from the CPU:

```
SFR0 <= ADDR_LH;
SFR1 <= ADDR_UH;
SFR2 <= WDATA_LH;
SFR3 <= WDATA_UH;
SFR4 <= 0x03;
While (SFR7 & 0x01) { }
SFR 4 <= 0x00;
```

In the above code, UH and LH indicate the upper half and lower half of the word, respectively.

Similarly, the following C instruction set resembles a simple CPU memory read operation:

```
SFR0 <= ADDR_LH;
SFR1 <= ADDR_UH;
SFR4 <= 0x05;
While (SFR7 & 0x01) { }
SFR 4 <= 0x00;
RDATA_LH <= SFR5;
RDATA_UH <= SFR6;
```

In the above read and write operation example, the CPU is stalled in the "while" loop for the time that the busy signal is high. This can be implemented differently according to each application. For example, the CPU can check the status frequently and if the status is busy, it can perform other functions and return to check the status after some period of time.

An external glue logic is necessary to connect the SFR control and status registers to the Core1553 CPU interface. Figure 7 on page 9 shows the interconnect of this implementation.

The glue logic block in Figure 7 on page 9 is simply a state machine. This state machine reads the read and write command from the control register (SFR4) and transfers the memory access request of the CPU to the Core1553 CPURDn and CPUWRn inputs. It also updates the status register (SFR7), based on the operation stage and state of the CPUWAITn output of Core1553.

Glue logic is simply a state machine. Figure 8 on page 9 shows a simple flow chart of the glue logic state machine operation.

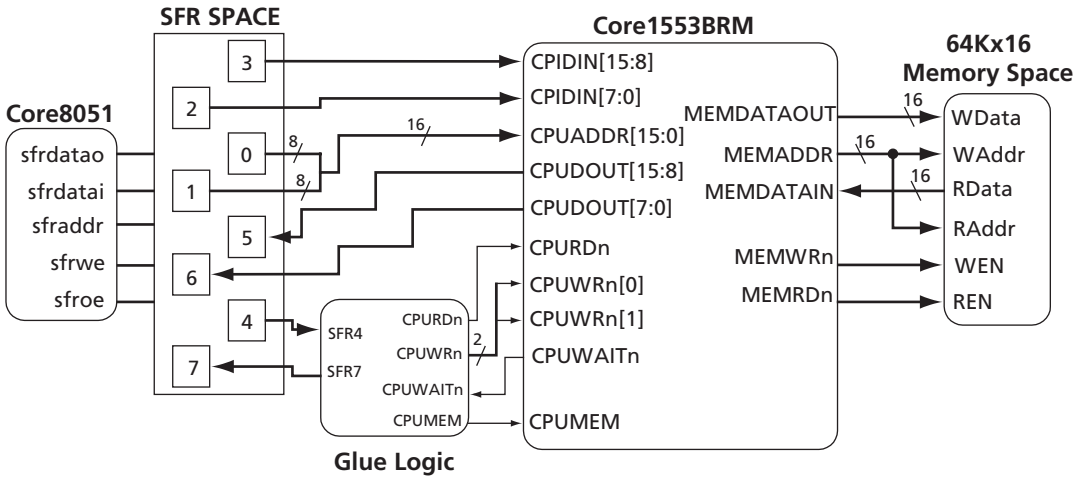

Figure 7 • **System-Level Interconnect of SFR Space Usage**

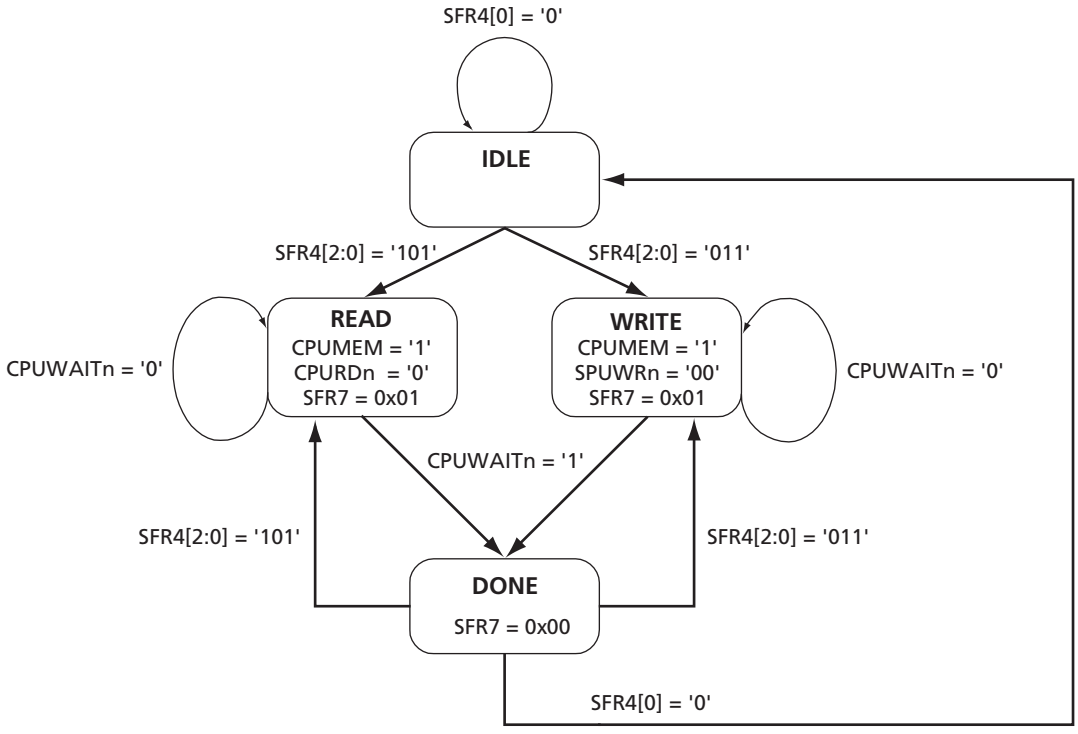

Figure 8 • **Flow Chart of The SFR Glue Logic State Machine**

During the memory read operation, the glue logic can latch the read data into the SFR read data space (SFR5 and SFR6 in this case) to ensure that the CPU receives the valid read data. The rising edge of CPURDn, which flags the end of read operation, can be used to latch the data. This has not been illustrated in Figure 7 or Figure 8 on page 9 for simplicity, but has been included in the glue logic design example located on the Actel website at http://www.actel.com/techdocs/appnotes/products.aspx.

Legacy Support

In order to support legacy MIL-STD-1553B devices, Actel Core1553BRM provides the flexibility to be configured compatibly with these devices. In this case, a wrapper is placed around the core and the CPU interface ports are not accessible anymore. Therefore, if Core1553BRM is selected to support legacy MIL-STD-1553B devices and it shares the backend memory space with the host controller, the arbitration or dual-port memory implementation should be used, since CPU interface ports of the Core1553BRM are disabled.

Conclusion

Actel MIL-STD-1553A/B bus interface IP core (Core1553) can share the backend memory space with the host processor of the system. Core8051 is an Actel 8-bit IP core which can be used as the system host processor. If used together in a system, Core8051 and Core1553 can share the backend memory space without affecting the MIL-STD-1553B overall system performance. The Actel Core1553 provides more flexibility to perform this than legacy MIL-STD-1553 devices. The implementation of this application, discussed in this document, varies based on Core1553 variations (BRT, BBC, and BRM). Some implementations require external logic such as glue logic or a bus arbiter. These external blocks may be fitted inside the FPGA alongside Core1553 or Core8051, which reduces the cost and space of the board.

Related Documents

Application Notes

Implementing Multi-Port Memories in ProASIC^{PLUS} Devices
http://www.actel.com/documents/APA_MultiPort_AN.pdf
Implementing Multi-Port Memories in Axcelerator Devices
http://www.actel.com/documents/AX_Multi_Port_AN.pdf

Datasheets

Core1553BRM MIL-STD-1553 BC, RT, and MT
http://www.actel.com/ipdocs/Core1553BRM_DS.pdf
MIL-STD-1553B Bus Controller Core1553BBC
http://www.actel.com/ipdocs/Core1553BBC_DS.pdf
MIL-STD-1553B Remote Terminal Core1553BRT
http://www.actel.com/ipdocs/Core1553BRT_DS.pdf
Core8051
http://www.actel.com/ipdocs/Core8051_DS.pdf

List of Changes

Previous Version	Changes in Current Version (51900085-1/1.05*)	Page
51900085-0/1.05*	The "Core1553BBC/BRM CPU Interface" section was updated. Design examples are posted below this app note: VHDL: • SFR Design Files • Backend MEM Design Files Verilog: • SFR Design Files • Backend MEM Design Files	6

Note: **This is the part number located on the last page of the document.*

www.actel.com

Actel Corporation

2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.

Dunlop House, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Phone +44 (0) 1276 401 450
Fax +44 (0) 1276 401 490

Actel Japan
www.jp.actel.com

EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668

Actel Hong Kong
www.actel.com.cn

Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488