

## UTMC APPLICATION NOTE

---

### S $\mu$ MMIT<sup>TM</sup> & S $\mu$ MMIT<sup>TM</sup> LX to FPGA Interface

#### Basic Operation

For this application the S $\mu$ MMIT or S $\mu$ MMIT LX (hereinafter referred to as S $\mu$ MMIT) interfaces to a FPGA. The system does not allocate any memory for 1553 message storage. All data associated with 1553 message processing is retrieved from or stored into the FPGA. The FPGA architecture allocates a 34 x 16-bit register file for message processing: 32 registers are read/write which support the maximum message length of 1553, two additional words are allocated for the message time-tag word and information word. The FPGA also contains two 16-bit registers for the storage and retrieval of housekeeping words that the S $\mu$ MMIT accesses during message processing (Control Word and Transmit/Receive Data Pointer). The total memory/register requirements for the S $\mu$ MMIT is 36 x 16 bits.

#### FPGA Register Access

The S $\mu$ MMIT requires four “housekeeping” words for each message transaction. The S $\mu$ MMIT “burst” DMA reads these four words after reception of a valid command word. With ping-pong and broadcast operation disabled, only two words (i.e., Control Word and Data Pointer A) are required. The Control Word and Data Pointer A are read first and second respectively in the read DMA burst. The last two words are “don’t care”. At the end of message processing, the S $\mu$ MMIT writes/updates the Control Word and Data Pointer A. The FPGA register containing the Control Word and Data Pointer A can be either read-only or read/write. If read only, the FPGA discards any write information. In this architecture, the S $\mu$ MMIT maps all 1553 message into the same 34 word register file. To support message processing the S $\mu$ MMIT reads or writes data words approximately every 20 $\mu$ S. The S $\mu$ MMIT Reference Manual and S $\mu$ MMIT Family Product Handbook contain additional timing diagrams and information on memory access.

#### FPGA Register Address Map

The S $\mu$ MMIT will access up to 36 register locations, within the FPGA, during message processing. The S $\mu$ MMIT treats these 36 register locations as the Descriptor Space look-up table and subaddress/mode code data buffer. The S $\mu$ MMIT accesses the 36 x 16 register file with A(5:0), D(15:0),  $\overline{RCS}$ ,  $\overline{RRD}$ , and  $\overline{RWR}$ . Do not use address outputs A(15:6) to access the FPGA register access.  $\overline{RCS}$ ,  $\overline{RRD}$ , and  $\overline{RWR}$  control the selection and read/write process.

During a Descriptor Block read address zero and one (i.e., 0<sub>16</sub> and 1<sub>16</sub>) correspond to subaddress/mode code Control Word and Data Pointer A respectively. The S $\mu$ MMIT will attempt to read address locations 2<sub>16</sub> and 3<sub>16</sub>; the FPGA can return a value “XXXX<sub>16</sub>” for these read cycles. Register file location 1<sub>16</sub> must contain a value of 4<sub>16</sub>. Register file locations 2<sub>16</sub> and 3<sub>16</sub> correspond to Data Pointer B and the Broadcast Data Pointer. Reserve register file locations 4<sub>16</sub> to 25<sub>16</sub> contain either transmit or receive data words. For transmit commands, the S $\mu$ MMIT reads these locations at a 20 $\mu$ S interval. For receive commands, the S $\mu$ MMIT writes data into these locations at a 20 $\mu$ S interval. 32 locations are reserved to support the maximum message length of the MIL-STD-1553 protocol. After the reception or transmission of an entire message, the S $\mu$ MMIT updates the Control Word and Data Pointer A by writing to register locations 0<sub>16</sub> and 1<sub>16</sub> respectively.

## DMA Operation

Prohibit the S $\mu$ MMIT from accessing the FPGA register file when updating or off-loading the register file contents. To prohibit S $\mu$ MMIT access, prevent  $\overline{\text{DMAG}}$  from going low after the S $\mu$ MMIT asserts  $\overline{\text{DMAR}}$ . A simple logical “block” signal, in the FPGA, prevents  $\overline{\text{DMAG}}$  from equaling  $\overline{\text{DMAR}}$  when the register file is being updated.

The “block” signal asserts when  $\overline{\text{TERACT}}$  is a logical one. Likewise the FPGA “holds-off” accessing the register file while the remote terminal is active (i.e.,  $\overline{\text{TERACT}}$  logical zero). Using  $\overline{\text{TERACT}}$  to block DMA access to the FPGA eliminates the need of synchronizing  $\overline{\text{DMAR}}$  into the FPGA. Figure 1 is an example circuit.

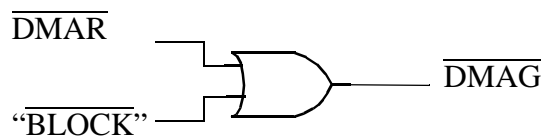


Figure 1. DMA Block Circuit

## Initialization and Mode of Operation

The FPGA configures the S $\mu$ MMIT for operation by writing to the S $\mu$ MMIT’s internal registers after master reset. For simple operation, the FPGA writes only to the Control Register to start operation (i.e., sets  $\text{STEX} = 1, 9800_{16}$ ). The FPGA uses inputs A(4:0), D(15:0),  $\overline{\text{CS}}$ , and R/ $\overline{\text{WR}}$  to access the S $\mu$ MMIT’s internal registers.

Lock the S $\mu$ MMIT’s mode of operation and remote terminal address on the rising edge of master reset by grounding the  $\overline{\text{LOCK}}$  input. The S $\mu$ MMIT latches RTA(4:0), RTPTY and MSEL(1:0) on the rising edge of master reset. Select remote terminal operation by grounding MSEL(1) and tying MSEL(0) to five volts. The FPGA can begin accessing the S $\mu$ MMIT 4.5 $\mu$ S after the rising edge of master reset. Ground input A/ $\overline{\text{B}}$  for MIL-STD-1553B operation.

## Interrupts

The S $\mu$ MMIT interrupt architecture supports either real-time or latent interrupt service. For systems that have interrupt latency, the S $\mu$ MMIT builds an interrupt log list in external memory. The interrupt service routine reads the interrupt log list to process interrupts. For a system that implements real-time interrupt processing, the S $\mu$ MMIT records interrupts into the Pending Interrupt Register. The interrupt processor reads the Pending Interrupt Register and Current Command Register to determine the proper interrupt service.

The FPGA does not reserve any memory for an interrupt log list; therefore, implement a real-time interrupt processing scheme (if desired). Write bit INTEN of the Control Register to a logical zero. The S $\mu$ MMIT generates an interrupt pulse each time a message is processed. The FPGA

reads the Pending Interrupt Register and Current Command Register to determine the proper interrupt service. The FPGA un-masks specific interrupt(s) by writing to the Interrupt Mask register prior to starting operation (e.g., Interrupt When Accesses,  $0400_{16}$ ). Control Word bit 6 is set to a logical one to enable the interrupt.

If employing the S $\mu$ MMIT's interrupt architecture, it is wise to also un-mask the  $\overline{\text{YF\_INT}}$  interrupt. The  $\overline{\text{YF\_INT}}$  is useful for system de-bug. Assertion of the interrupt indicates a hardware failure including: terminal address parity error, wrap-around self-test failure, built-in test failure, or DMA time-out failure. The  $\overline{\text{YF\_INT}}$  does not require enabling in addition to un-masking.

### **Status Word Bits**

To control the S $\mu$ MMIT's Status Word bits, the FPGA writes to the Status Word Bits Register. Accessing this register is optional. The S $\mu$ MMIT automatically controls the Message Error, Terminal Flag, and Broadcast bits; use of all other bits in the remote Status Word is optional (i.e., Service Request, Instrumentation, Busy, Subsystem Flag, and Dynamic Bus Control Acceptance). An external pin (i.e.,  $\overline{\text{SSYSF}}$ ) controls the Subsystem Flag bit. The subsystem asserts the  $\overline{\text{SSYSF}}$  to indicate a subsystem failure to the bus controller.

Depending on the bus controller service algorithm (periodic versus aperiodic), assertion of the Service Request bit may be beneficial. For aperiodic bus controller access, the remote terminal signals the bus controller that pre-determined service is necessary via a polling message. In periodic bus controller applications, messages are transacted at a fixed rate and data is made available to support the system rate of transfer (e.g., 10Hz). In polling systems, assertion of the Service Request bit indicates to the bus controller that service is necessary (e.g., the FPGA register file contains a valid A to D sample and the bus controller must issue a transmit command). In this case, the capability of writing to the Status Word Bits Register is beneficial. The Immediate Clear function eliminates the Service Request clear register write. To set the Service Request bit, the FPGA write  $8100_{16}$  to the Status Word Bits Register.

### **Diagnostics**

The S $\mu$ MMIT supports a comprehensive built-in test; either the bus controller or the FPGA initiates built-in test. The bus controller initiates the built-in test via a mode code; the entire test sequence is automatic. The FPGA initiates a built-in test by using the following sequence: master Reset, wait  $4.5\mu\text{S}$  Control Register write  $4000_{16}$ , wait 1mS for self-test to complete, read BIT Word register. A Control Register write (i.e.,  $2000_{16}$ ) invoking a software reset can replace the master reset.

### **Subaddress and Mode Code Illegalization**

The S $\mu$ MMIT automatically initializes the illegalization register files to support MIL-STD-1553B operation. The FPGA does not need to access registers  $10_{16}$  to  $1F_{16}$ .

Figure2. System Block Circuit

