A Description of the BGA2 Firmware

William E. Earle

June 22, 2005

1 Introduction

This document provides a description of much of the firmware contained in the BGA2 FPGA, the FPGA that interfaces the WFD to VMEbus.

An external 50 MHz oscillator is the clock source for BGA2, with a DCM being used to ensure that the 50 MHz internal clock signal is free of noise. The 50 MHz clock is also sent to the other FPGA’s, BGA1M and BGA1S so those FPGA’s can synchronously communicate with BGA2.

BGA2 uses a single 20 state, one-hot state-machine to control all of its operations, each using one or more states of this state machine. At startup, the idle, or do-nothing state is entered. The DS0* and DS1* VMEbus inputs are synchronized to the 50 MHz clock and used to cause entry to some other state. Decoding of the AM code and address selects the state to be entered. Not including the idle state, 10 operations recognized by BGA2. These include reading and writing control parameters, and reading and writing FIFO data. Reading FIFO data is a very complex procedure that involves BGA2 communicating with the other two FPGA’s, BGA1M and BGA1S. Most other functions performed by BGA2 are quite simple compared to reading FIFO data.

Decoders examine the AM code and the address signals when DS0* and/or DS1* go low to determine which function BGA2 is to perform. If a function is not recognized or is recognized as illegal, then BGA2 may not respond to the VMEbus request by not providing the expected DTACK response. For example, a write to the CR area will not yield a DTACK because the CR area is read-only.
2 Reading CR data

The CR (control ROM) contains read-only data in accordance with the VME64X specification in locations 0 through 0xFFF plus a small user-defined read-only area starting at location 0x1000. For the WFD, the user-defined data uses CR addresses 0x1000 through 0x102F giving a 4144 byte CR address space, but since only every fourth byte contains data, only a 1036 byte ROM is needed. The ROM is provided by a Xilinx read-only-memory IP core in conjunction with an initialization file called rom1036.coe. At startup, the ROM data in rom1036.coe is loaded into the ROM for access by the VHDL code.

Data is constantly being read from the ROM, but unless the VMEbus is requesting CR data, this data is not returned over VMEbus. Only one state of the state machine is used to read CR data.

Fig.1 shows the timing of the signals involved with the readout of CR data. Signal as_b is a registered copy of AS* and ds1 is a registered copy of DS0*. ds2 is a registered copy of ds1 and ds3 is a registered copy of ds2. The condition of ds2 being high while ds3 is still low enables a latch_state signal that causes the new state (cr_state) to be entered. This state enables the ROM data back to VMEbus and also enables a request for a DTACK* response to VMEbus via the signal req_dtack. The req_dtack signal causes dtack_a to be set, which in turn causes dtack_b to be set. These two signals enable the DTACK* to VMEbus, which responds by returning DS0* to its high logic level. When DS0* goes high, ds1, ds2 and ds3 are set low in serial fashion, with the low state of ds1 causing dtack_a and dtack_b to be returned low. After ds2 goes low, the state reverts to “idle”. After dtack_a has gone low but before dtack_b goes low, a rescind signal is enabled. The rescind signal causes an active pull-up of the DTACK* signal lasting for 20 nanoseconds after DTACK* has gone high. The purpose of the rescind pull-up is to overcome any delay in DTACK* returning to its high condition because of capacitive loading of the DTACK* signal by the several VME cards. DTACK* may be pulled low by any VMEbus card, but it relies on a passive resistor pull-up to return to its high level, so the capacitive loading of many VMEbus cards could delay DTACK* from returning high, which in turn might slow the VMEbus rate.
3 Serial Number

The board serial number, programmed via 10 jumpers, is read as the Board ID of the CR area, locations 0x33 through 0x3F. Although four bytes are available, the 10 bit serial number only occupies locations 0x3B and 0x3F.

Since locations 0x33 through 0x3F are located in the ROM address space, special decoders are used to replace the ROM data with the jumper-programmed data. A single state is used to read the serial number, with two VMEbus read operations needed to obtain all 10 bits.

The timing shown in Fig 1 also applies to reading the serial number.

4 CSR Data

The CSR area defined by VME64X occupies the CR/CSR address space from 0x7FC00 through 0x7FFFF, but only a few locations in this space are of interest to the WFD. These are the BAR (base address register, or slot number), the Bit Set Register and Bit Clear Registers, and the Function 0 ADER and Function 1 ADER registers. Writing or reading any of these locations uses a single, but different, state of the state machine.

Note that four accesses are needed to write the complete ADER registers. It should also be noted that though the least significant bytes of the Function 0 ADER and Function 1 ADER registers can be written with arbitrary values, these least significant bytes are treated as zeros during relocatable addressing. This is because the lower bytes of the ADER registers are intended to hold extended ADM codes, XAM codes, and not address bits, but these are not used in the WFD.

The Function 0 ADER register specifies the address in normal VMEbus address space of the FIFO word counts and the Function 1 ADER register specifies the location of the FIFO data.

Only three bits of the Bit Set and Bit Clear registers are recognized, the reset, fail and enable bits. At startup, these bits are cleared, so the enable bit of the Bit Set Register must be written with a '1' to enable the WFD to collect data.

Fig. 1 applies to writing or reading CSR data.
5 USER CSR Data

As with CSR data, the registers in the User CSR area cannot all be accessed using the same states. Writing to the DAC registers involves setting a bit counter and then serially shifting data into the DAC. Two states in the state machine are used to write to any of the 10 DAC registers, one state to initialize a shift counter and the other to do the actual shifting. DTACK* is delayed until the entire 20 bits of DAC data have been sent to the DAC. Data that has previously been written to a DAC register can be read back to VMEbus, but this data comes from holding registers and not directly from the DAC.

Fig. 2 shows the timing for writing to the DAC. Fig. 1 applies for readback of DAC values.

Accessing the three interrupt registers also uses two states of the state machine.

Most accesses to the other USER CSR registers require communicating with BGA1M and BGA1S, where most of these registers reside, although a few registers are stored in all three FPGA’s. These accesses use two machine states, one of the states used to provide additional time for address decoding in BGA1M and BGA1S.

A seven bit address plus read/write signals, dreq/dwrite, and copies of the 50 MHz clock used in BGA2 are sent from BGA2 to BGA1M and BGA1S.

Data to/from BGA1M and BGA1S uses two 8-bit buses, one for each FPGA. These buses also return interrupt information from BGA1M and BGA1S to BGA2 when a VMEbus cycle is not in progress.

Figs. 3 shows the signal timing for writing or reading to/from BGA1M and BGA1S.

6 The FIFO Count Registers

Reading the FIFO count of a particular channel uses two states of the state machine and requires that BGA2 retrieve the count from BGA1M (for channels 1 or 2) or from BGA1S (for channels 3 or 4). The first state initializes a transfer counter and the second state performs the actual transfers, since the FIFO count is a 17 bit value and three 8-bit transfers from BGA1M or BGA1S are needed to get the complete FIFO count. From the VMEbus standpoint, the three internal transfers are invisible (DTACK is delayed un-
til the three transfers have been completed). Although VMEbus can read the FIFO count using a 1, 2 or 4 byte transfer, unless the WFD is in its stopped condition, only the 4 byte transfer should be used. This is because the FIFO count can change between readouts if 1 or 2 byte transfers are used, so generating the complete count from partial readouts can give an incorrect value.

7 Writing and Reading FIFO Data

The writing and reading of FIFO data can only be done using 64 bit block transfers. A VMEbus write to a FIFO can only be done from the test mode. Five machine states are used for writing and reading FIFO data. When writing test data, an eight-byte VMEbus data value is sent from BGA2 to BGA1M and BGA1S as eight successive bytes, the last byte being accompanied by an end-of-data signal that triggers BGA1M or BGA1S to actually write the data to the correct FIFO. Only after the eight bytes have been output from BGA2 is a DTACK* signal sent to VMEbus, allowing another 8-byte transfer.

Fig.4 shows the timing when writing one 64 bit word from VMEbus to BGA1M or BGA1S.

The readout of FIFO data is a very complicated procedure using the same five machine states used for writing test data, but due to its very lengthy nature, this procedure is discussed in a separate document.
Timing for readout of CR data

Fig. 1
Fig. 2 DAC signal timing
Fig. 3 BGA1M and BGA1S timing for writing or reading parameters
Fig. 4 Signal timing for writing test data