SLAC – PUB – 3769
September 1985
(I)

A FASTBUS INTERFACE FOR THE 3081/E*

L. BARKER, P. F. KUNZ, A. J. LANKFORD, G. OXOBY
L. PAFFRATH, P. RANKIN, AND Q. TRANG
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

Abstract

The design of a FASTBUS interface to the 3081/E is presented. The interface consists of two boards, one specific to FASTBUS, the other usable by other interfaces to the 3081/E. The FASTBUS board is a dual-ported slave, permitting access from either of two cable segments. The general purpose board supports transfers to and from 3081/E memory and provides control of program execution. It also has several features which facilitate software debugging.

1. Introduction

The 3081/E[1] is a processor which emulates the instruction set of an IBM mainframe computer. Its implementation as part of an on-line data acquisition system increases the processing power available for such tasks as data preprocessing, software triggering/event flagging, and event reconstruction. This paper describes the design of an interface of the 3081/E to FASTBUS, which will be used to incorporate 3081/E's into the data acquisition system of the Mark II detector[2] at the SLAC Linear Collider.

2. Description of the Interface

2.1. OVERVIEW

The FASTBUS to 3081/E interface consists of two boards. The first board (FASTBUS Interface) is a dual-ported FASTBUS slave which allows access from either of two FASTBUS cable segments to an internal bus between this board and the second board (Common Interface). The protocol on this bus is a two-cycle (address and data) handshake. The signals on this bus can load a set of control registers on the Common Interface board which drive the 3081/E busses. The Common Interface is sufficiently general that it can be used as part of other interfaces to the 3081/E.

2.2. FASTBUS INTERFACE – BOARD 1

The FASTBUS Interface acts as a FASTBUS slave and has two ports to FASTBUS cable segments, contention logic, and logic for communication with the Common Interface board. Figure 1 shows a block diagram of the FASTBUS Interface.

The two FASTBUS cable segment ports are logically identical. Thus, the interface is symmetric with respect to both cable segment ports except as defined by the application. The slaves are referred to as near-side and far-side. The near-side slave as viewed by one cable segment port becomes the far-side slave when viewed from the other port. The slaves respond to geographical addressing and to certain types of broadcast addressing (general, pattern select, TP, and TP if SR). Four data transfer modes are supported — random data, block transfer, secondary address, and pipelined transfer. Data transfer speed is limited by the Common Interface to about 25 Mbyte/sec.

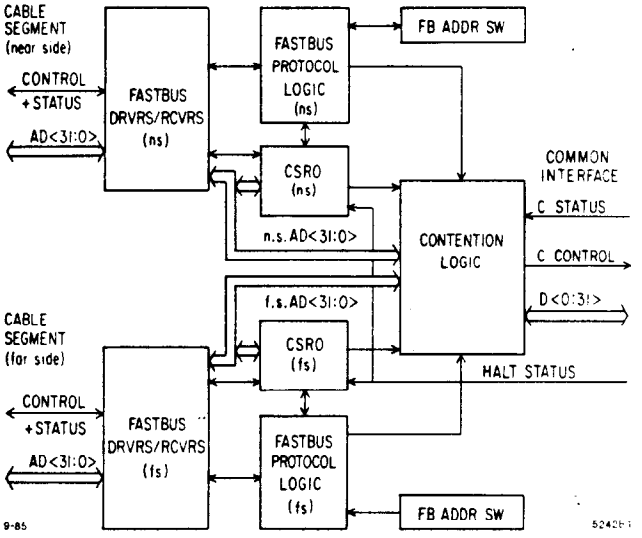

Fig. 1. Block diagram of the FASTBUS interface board.

An SS=0 (address recognized) response is generated for all primary address cycles. SS responses to other cycles are controlled by an FPLA. For secondary address cycles, an SS=3 (user defined) response is generated if connection cannot be made to the common interface; otherwise an SS=0 (valid action) or SS=7 (data error-accept) response is generated for valid or invalid secondary addresses respectively. For data cycles, an SS=0 response is generated for valid data, an SS=2 (end of block) response is generated when the next transfer address increments out of range, and an SS=6 (data error-reject) response is generated for invalid addresses. FASTBUS parity is neither generated nor checked.

The contention logic examines all secondary address operations and, for addresses other than CSR0 (Table 1), allows connection to the Common Interface if the processor is not executing a program and if another master is not connected to the other port. Via operations to CSR0 it is also possible to disable connection of either cable segment port (bits 24 & 26) or to allow connection to the Common Interface during execution (bit 13). Connection remains intact until the FASTBUS cable segment negates its AS line. While one port is connected attempts to connect by the other port are answered with an SS=3 code. A port can "interrupt" a sequence of FASTBUS operations by the other port by setting bit 11 in CSR0, which causes an SS=3 code on the connected segment.

Each FASTBUS cable segment has a CSR0 which is accessible at all times, regardless of the state of connection to the Common Interface and of the processor. Table 1 shows the contents of CSR0. The first three bits (i.e., 1,2,3) are provided by the Common Interface and reflect the state of the processor.

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.

Poster paper presented at the Nuclear Science Symposium, San Francisco, California, October 23-25, 1985

All other bits control logic on the FASTBUS interface. Several pairs of bits (i.e., 4 & 6, 5 & 7, 8 & 10, 9 & 11) can be set and reset from either cable segment port. This facility allows the two ports to signal each other. Bit 12, fs Slave Selected, allows a port to monitor the connection condition. The service request (SR) of a cable segment port can be reset via CSR0 (bits 21 & 23) and can be set via CSR0 (bits 5 & 7) if enabled on the appropriate segment (bits 4 & 6). When the processor halts SR is set on any enabled cable segment. This service request facility is the usual means for the processor to signal the completion of execution (or an error) to the appropriate port. The manufacturer's ID, 0017 (hex), can be read in the high order bits.

Table 1. CSR#O Contents

Bit	Meaning on READ	Action on WRITE
0		
1	CPU enabled	
2	Clock running	
3	Single Step	
4	ns SR enabled	Enable ns SR
5	ns SR state	Set ns SR
6	fs SR enabled	Enable fs SR
7	fs SR state	Set fs SR
8	ns slave enabled	Enable ns slave
9	ns slave interrupt	Set ns interrupt
10	fs slave enabled	Enable fs slave
11	fs slave interrupt	Set fs interrupt
12	fs slave selected	
13	override lockout	Enable override
14		
15		
16	Mfg. ID# 1	
17	Mfg. ID# 1	
18	Mfg. ID# 1	
19	Mfg. ID# 0	
20	Mfg. ID# 1	Disable ns SR
21	Mfg. ID# 0	Reset ns SR
22	Mfg. ID# 0	Disable fs SR
23	Mfg. ID# 0	Reset fs SR
24	Mfg. ID# 0	Disable ns slave
25	Mfg. ID# 0	Reset ns interrupt
26	Mfg. ID# 0	Disable fs slave
27	Mfg. ID# 0	Reset fs interrupt
28	Mfg. ID# 0	
29	Mfg. ID# 0	Disable override
30	Mfg. ID# 0	
31	Mfg. ID# 0	

Notes:
1. ns => near-side, fs => far-side.
2. Bits 31-16 on read; Mfg. ID# = '0017'X.
3. Bit 0 is least significant bit (i.e. FASTBUS convention).

2.3. Protocol Between Interface Boards

The protocol used between the two boards of the interface maximizes block transfer rates without requiring that the design of the Common Interface be specific to this application. It consists of an address cycle, either read or write, followed by any number of read or write data cycles. The signals used are analogous to FASTBUS operations with logical addressing. CAS and CAK,CDS and CDK, and CRD signals are analogous to AS and AK,DS and DK, and RD. CERR is analogous to SS<2:0>, and DataS and INCR serve the function of MS<2:0>, that is they serve to indicate FASTBUS data space operations and to increment the NTA. Primary address cycles on the FASTBUS cable segment do not affect the interboard bus. FASTBUS secondary address cycles produce interboard address cycles, and FASTBUS data cycles produce interboard data cycles. Figure 2 shows a timing diagram for these cycles.

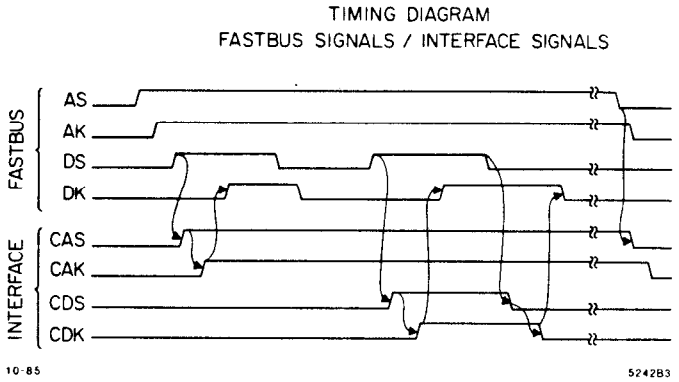

Fig. 2. Timing diagram of FASTBUS and interface signals.

2.4. Common Interface – Board 2

The Common Interface board (Fig. 3) is designed to provide a means of communication with all the 3081/E busses and control of the processor's clock. This allows the transfer of data to and from processor memory, as well as permitting the interface to force execution of any 3081/E instruction or program. It has a total of 25 registers (listed in Table 2).

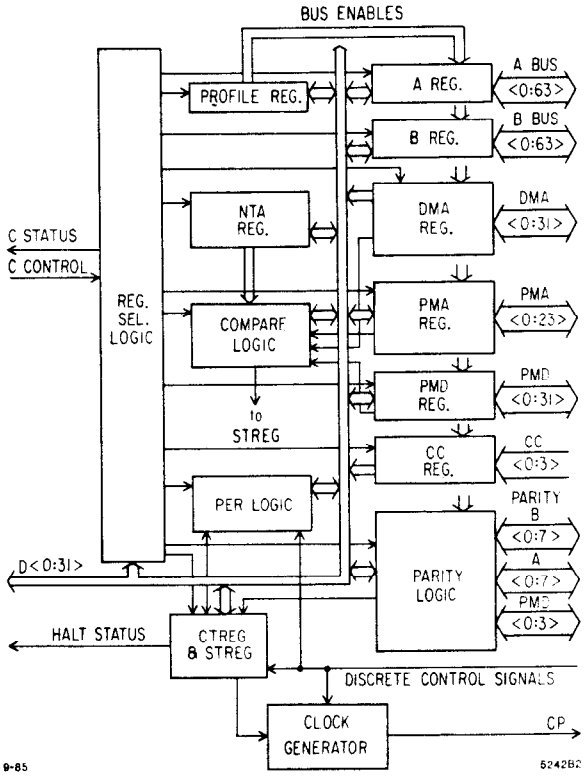

Fig. 3. Block diagram of the common interface board.

Among the most important of these are the profile register,the control register, the status register, and the trap register.

The profile register (PROREG) can assert each of the internal 3081/E buses with the contents of the corresponding register. Read/write access is allowed to the following internal 3081/E busses; the Program Memory Address (PMA) bus, the Program Memory Data (PMD) bus, the Data Memory Address (DMA) bus, and the two operand busses (ABUS and BBUS). Read access only is allowed to the Condition Code (CC) bus.

The control register (CTREG) controls the state of the processor. It can issue resets, enable the processor, or halt it. It also controls the clock which may be stopped, stepped, or allowed to free run. The 3081/E clock is generated by a free running clock generator with a 25 MHz standard frequency. The processor clock operates at a third of this frequency (giving a cycle time of 120 nsec) and with a duty cycle of 2/3 high and 1/3 low.

Table 2. Interface Register List

Address	Reads	Writes	Name
0	Profile Reg	Profile Reg	PROREG
1	Control Reg	Control Reg	CTREG
2	Status Reg		STREG
3	PMABUS	PMABUS Reg	PMAREG
4	DMABUS	DMABUS Reg	DMAREG
5	Condition Code Reg		CCREG
6		Mask Reg	MASKREG
7	Parity Reg	Parity Reg	PTYREG
8		DMAHI Trap Reg (execution)	DMARHI
9		DMALO Trap Reg (execution)	DMARLO
A		PMAHI Trap Reg (execution)	PMARHI
B		PMALO Trap Reg (execution)	PMARLO
C		DMAHI Trap Reg (transfer)	DMAXHI
D		DMALO Trap Reg (transfer)	DMAXLO
E		PMAHI Trap Reg (transfer)	PMAXHI
F		PMALO Trap Reg (transfer)	PMAXLO
10	ABUS<0:31>	ABUS<0:31>Reg	AREGH
11	ABUS<32:63>	ABUS<32:63>Reg	AREGL
12	ABUS Parity		ABPTY
13	BBUS<0:31>	BBUS<0:31>Reg	BREGH
14	BBUS<32:63>	BBUS<32:63>Reg	BREGL
15	BBUS Parity		BBPTY
16	PMDBUS	PMDBUS Reg	PREG
17	PMDBUS Parity		PBPTY
18	Trap Enable Reg	Trap Enable Reg	TRAP

The Common Interface incorporates features which aid in debugging programs running on the processor. It has registers which can halt program execution if certain conditions occur, in a way similar to the Program Event Recording (PER) registers of IBM mainframes. For example, the interface can be enabled to halt execution upon a store within a defined address range, or halt if certain of the processor's general purpose registers are modified. These error traps can be enabled via the trap register (see Table 4).

The status register (STREG), contains information on the state of the processor (see Table 3), such as whether it is running or halted,if errors have occurred in the transfer of data, or if errors occurred during program execution. In addition, the processor may be halted in response to a parity error on certain of its internal buses. In a write to memory operation, the parity is either generated internally prior to the memory write, or supplied by a register (a useful feature for testing the logic of the Common Interface). When the processor is in normal use, the parity logic will be able to monitor the parity on the PMD and operand busses if this feature is enabled via the trap register enable.

The generality of the Common Interface design allows it to be part of an interface of the processor to an IBM PC used as a host/controller. An interface board in the PC is connected via an adapter board to the Common Interface. The approximate transfer rate between the PC and the 3081/E is 0.5 Mbytes/sec.

Table 3. Status Register

Bit	Meaning
31	CPU run
30	CPU halt
29	
28	
27	PMA error
26	PMA transfer error
25	
24	Invalid register
23	
22	PERWR error
21	DMA error
20	DMA transfer error
19	
18	IBM exception
17	CPU exception
16	PBUS parity error
15	ABUS parity error
14	BBUS parity error

Notes: Bit 31 is least significant bit (i.e. IBM convention)

Table 4. Trap Enable Register

Bit	Function
27	PMA trap in/out
26	PMA memory transfer trap in/out
25	PMA trap enable
24	PER branch trap enable
23	PER write memory trap enable
22	PER modify register trap enable
21	DMA trap in/out
20	DMA memory transfer trap in/out
19	DMA trap enable
18	IBM exception trap enable
17	CPU exception trap enable
16	PBUS parity trap enable
15	ABUS parity trap enable
14	BBUS parity trap enable
13	Parity source

Notes: Bit 31 is least significant bit (i.e. IBM convention)

3. Implementation on Mark II

We envisage the use on-line of several 3081/E processors by the Mark II detector at the SLAC Linear Collider. Data will flow from FASTBUS front-end electronics through a short

FASTBUS cable segment to a bank of 3081/E's as shown in Fig. 4. The dual ported nature of the interface allows the input of raw data to one processor at the same time as processed information is read out from another.

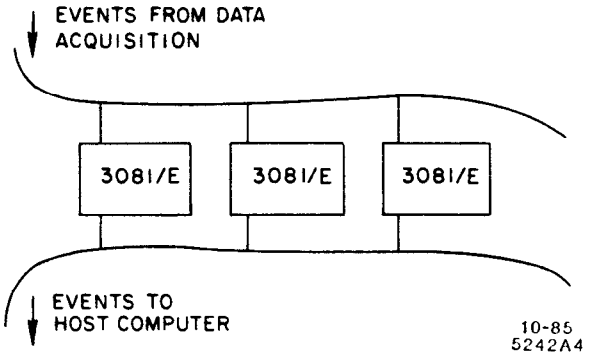

Fig. 4. Block diagram of a set of 3081/E processors sharing two FASTBUS cable segments.

The assertion of a service request on its associated cable segment signals to a FASTBUS master that a 3081/E is available to accept data from it. The interface is then attached to the master by a primary address cycle. As a precaution the interfaces CSR0 may be read, without affecting the processors state or interfering with operations controlled by the far-side master, to check the processors availability. If the master is to go ahead with data transfer to the processor the first stage will be to load the registers of the interface to set them up for the transfer. This stage writes a store instruction into the program memory data register, sets up the profile register so that the BBUS, DMABUS and PMDBUS contents are supplied by the registers on the common interface, and issues a clock to transfer the store instruction onto the memory board. The second stage can then be started using another secondary address cycle, the most significant nibble of the address being set to 4 or 8 to differentiate between whether the following data is to be stored in the processor's program or data memory.

After all data transfer operations are finished the next step is to prepare for program execution. The trap register can be set-up so that certain errors will halt program execution; for example sections of data memory can be protected against overwriting. The user can also decide if parity errors should be flagged but program execution allowed to continue. The master can choose to set the service request enable bits either for subsequent attachment by masters on the same cable segment (near-side) or for attachment by far-side masters if this is appropriate. Finally, the processor clock can then be started to allow program execution.

When the processor halts a service request will again be issued. The status register can be examined to see if the program finished execution successfully, or if execution was halted in response to a trap condition being met (such as an arithmetic exception). To read out the data from a processor a similar sequence of operations to those used to load it is needed, the main difference being the substitution of a load instruction for the store instruction. Each 3081/E may be sent parts of an event for data preprocessing and formatting or entire events for on-line event analysis. Not all 3081/E's need to be executing the same program.

4. Summary and Status

The FASTBUS to 3081/E interface is a dual-ported FASTBUS slave which can accept data from FASTBUS masters, control program execution on the 3081/E, and allow the results to be read out by another FASTBUS master. It is currently being debugged and should be available for use by the end of this year.

Acknowledgements

The 3081/E project is being carried out as a collaboration between SLAC and CERN DD division and the work has been divided equally between them. Discussions on the interface with Brian Martin are gratefully acknowledged , as is Joel De Witts work in helping to bring the interface into operation.

References

1. P. F. Kunz *et al.*, "The 3081/E Processor", SLAC PUB-3332, April 1984, CERN DD/84/4, April 1984. P. Rankin *et al.*, "On-line use of 3081/E Microprocessors", IEEE Trans. Nucl. Sci. *NS-32*, 1321 (1985).
2. Mark II Collaboration "A Proposal for the Mark II at the SLC", SLAC PUB 3561; CALT 68-1015. A.J. Lankford and T. Glanzman, "Data acquisition and FASTBUS for the Mark II detector", IEEE Trans. Nucl. Sci. *NS-31*, 255 (1985).