The TURBOchannel Interface ASIC

Technical Overview

0. Introduction

The TURBOchannel Interface ASIC (TcIA) provides a single-chip TURBOchannel interface for a general TURBOchannel peripheral option. Use of the TcIA in a TURBOchannel option will substantially simplify the design of the option and minimize the cost and board area of the interface. Digital's design of the TcIA is based on input from a number of third party vendors experienced in designing TURBOchannel options, with iterated review cycles during the development of the TcIA functional specifications.

The 160-pin chip supports all TURBOchannel functions over the full range of TURBOchannel clock frequencies, and adds very little latency to most transactions. Prior to the TcIA, option designers have used three to five PALs, plus needed transceivers and connecting logic, to implement the TURBOchannel interface functionality. The TcIA provides all the functionality typically needed in any TURBOchannel interface, plus many additional features that are valuable in advanced applications. Another potential benefit of using such a standard hardware interface is the possibility of making use of previously developed controlling software and firmware.

A major portion of the usefulness of the TcIA is that it provides an intelligent DMA engine, which automatically manages many of the details of the option's functions in block data transfers between the option and the system memory. TcIA manages DMA at the level of DMA operations, the block transfers requested by software, which each generally require several or many TURBOchannel DMA transactions. The role of the TcIA in TURBOchannel DMA operations is described further in Section 3.

Several other notable features of the TcIA are:

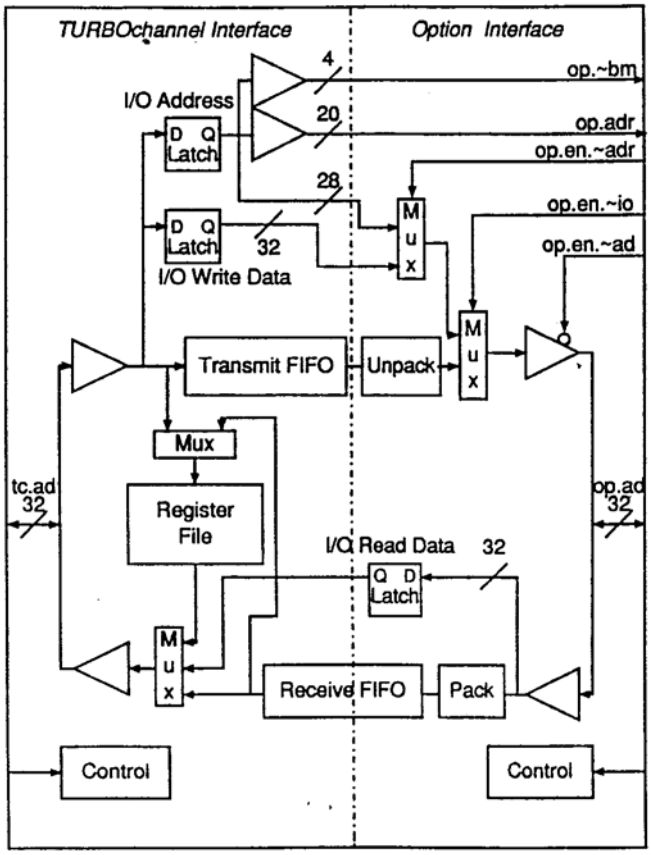
- Synchronization service, which allows the option to run at a different frequency from the TURBOchannel clock;
- Synchronized operation mode, which allows the option to run on the TURBOchannel clock, free of added latency;
- Support for half-word and byte oriented options, with automatic packing and unpacking, selectable byte ordering ("endian-ness"), and further features to support non-word-aligned DMA operations;
- Management of DMA memory buffer pointers, including automatic reloading from memory tables when virtual addresses cross physical page boundaries (gather/ scatter);
- Parity generation and checking

The possibility of non-synchronous operation is very useful for options that do not require the highest data transfer rate, or must run at some fixed frequency independent of the TcIA. They can be designed more economically to run at a fixed lower frequency, while remaining compatible with any TURBOchannel system.

In addition to DMA, the TURBOchannel protocol defines a very simple programmed I/O transaction (PIO), which moves only one data word per transaction. The just mentioned synchronization and parity management features of the TcIA abet the design of the PIO interface as well as the DMA interface.

Section 1 contains a brief description of the TcIA at the block-diagram level. Section 2 describes the use of the TcIA in programmed I/O operations. Section 3 describes the use of the TcIA in DMA operations. Section 4 discusses some of the features relevant to both transaction types and summarizes the benefits of using the TcIA.

Sections 2 and 3 together incorporate a brief description of the functioning of the TURBOchannel interconnect for the benefit of the reader who is not already familiar with it. For a more detailed description of the TURBOchannel operation, including signal timing diagrams, see the TURBOchannel Technical Overview or the TURBOchannel Specifications Kit. For a more detailed description of the TcIA, see the TURBOchannel Interface ASIC Functional Specification.



1. TURBOchannel Interface ASIC Description

The TcIA is a gate-array device, using 0.8-micron CMOS sea-of-gates technology. It uses roughly 22,000 gates of a 54,000 gate master. It is packaged in a 160-pin short-lead metric plastic quad flat pack.

Figure 1 shows a block diagram of the TcIA. The signals on the left are the TURBOchannel signals and the signals on the right are the option interface signals. The latter are the signals that directly concern the option designer using the TcIA. (The diagram does not show all the signals on either side, and omits some of the internal data paths as well). Tables 1 and 2 respectively list the TURBOchannel signals and the option interface signals. Sections 2 and 3 discuss the functions of some of these signals in the course of describing the TcIA operation.

The TURBOchannel signal *tc.clk* is the TURBOchannel clock, which is driven by the system at a fixed frequency between 12.5 and 25 MHz. The signal *op.clk* is the option-side clock which may be driven by the option at a different rate from the TURBOchannel clock. In asynchronous mode, the option interface signals are synchronized with *op.clk*; in synchronous operation the option interface signals are synchronized with tc.clk. The option selects synchronous operation by asserting *op.synchronous.*

Thus, the TcIA can provide an interface between two time domains--the TURBOchannel domain and the option domain, which are indicated as the left and right sides of the block diagram. The latch blocks provide buffering between the two time domains for PIO transactions. The two 15-word FIFOs provide buffering between the two time domains for DMA transactions. The transmit FIFO, used in DMA read transactions, is filled in step with the TURBOchannel clock and drained in step with the option clock. The receive FIFO, used in DMA write transactions, is filled in step with the option clock and drained in step with the TURBOchannel clock. The DMA engine of the TcIA reduces the option's view of DMA operation to interfacing with two synchronous FIFOs.

The Register File block contains the TcIA internal control and status registers. These registers are mapped to the option's address space and so are accessible to the system by means of PIO transactions. Some of the important fields in the Register File are

- Pointers to locations in DMA read and write buffers in system memory
- DMA operation word counters for DMA reads and DMA writes;
- DMA burst size;
- Pointers to buffer address tables to support the gather/scatter feature;
- Interrupt request bits and interrupt enable mask.

The use of these Register file fields is discussed further below in Sections 2, 3, and 4. The Register File contains numerous other control and status fields, of which some will be mentioned in the ensuing Sections 2, 3, and 4.

The MUX blocks are switches that select among several possible data sources. The two on the option interface side select data under control of the signals *op.enadr, op.enio,* and *op.en-ad.* The two MUXs on the TURBOchannel side of the diagram are controlled by the TcIA internally.

TURBOchannel DMA bursts move one word-aligned 32-bit word on the *tc.ad[31..0]* lines per clock cycle, but for some options the natural data transfer unit (option-word) is 16 or 8 bits. If the option-word is different from 32 bits, a DMA burst moves one option word onto the appropriate low order lines of op.ad per option clock cycle. The option-word size is held in a field in the Register File.

There are 44 TURBOchannel signals and 84 option interface signals. The difference in signal counts for the two interfaces is largely accounted for by the fact that the TcIA services include latching, unpacking, and demultiplexing of TURBOchannel data.

Table 1 TURBOchannel Signals

Name	Source	Function
tc.ad[P, 31,.., 0]	system/option	address/data bus
tc. –sel	system	PIO read/write select
tc. write	system	PIO read/write specifier
tc.~ack	system	DMA acknowledge
tc. –err	system	DMA error
tc. -reset	system	System reset
tc. Clk	system	TURBOchannel clock
tc.~rdy	option	PIO read/write ready
tc~.conflict	option	PIO read/write conflict
tc.~rReq	option	DMA read request
tc.~wReq	option	DMA Write request
tc.~int	option	I/O interrupt

Table 2 : TcIA Option Interface Signals

Name	Direction	Function
op. ad[31..0]	*I/O*	Multiplexed option interface address and data
op. adr[19..0]	*O*	Latched PIO address bits <21..2>
op.~bm[3:0]	*O*	Byte mask for PIO transactions
op. clk	I	Option clock
op. en. ad	*I*	Option asserts to enable *op. ad* drivers
op. en. -io	*I*	Option asserts to select I/O address/data onto *op. ad,* deasserts to select transmit FIFO data onto *op. ad.*
op.en.~--adr	*I*	Option asserts to select I/O address onto *op. ad* lines; deasserts to select data onto *op. ad* lines.
op. ~gpi[3..0]	*I*	General purpose input. See Section 4.3
op. gpo[7..0]	O	General purpose output. See Section 4.3
op.synchronous	*I*	Option asserts for synchronous operation, deasserts for asynchronous operation.
op. ~conflict	*I*	Option asserts to abort PIO transaction
op.~rcvAdr	*I*	Used in intelligent DMA to signal that DMA Write address is on *op. ad.* See Section 3.4
op.~trAdr	*I*	Used in intelligent DMA to signal that a DMA Read address is on the *op. ad.* See Section 3.4.
op. -rcvDat	*I*	Option asserts to load the data on the *op. ad* lines into the receive FIFO on the next option clock
op.~rcvValid	*O*	TcIA asserts to signal option that there is free space in the receive FIFO. See Section 3.2.
op.~trValid	*I*	TcIA asserts when there is data in the Transmit FIFO to be read by the option. See Section 3.2
op.~rdy	*I*	Option asserts to acknowledge PIO read or write data.
op.~sel	*O*	TcIA asserts to signal option selected for PIO operation,
op. -trDack	*O*	Option asserts to read from transmit FIFO.
op- *write*	*O*	TcIA asserts to indicate that the current PIO transaction is a write.
op.~test	*I/O*	For use in testing. See the TcIA Functional Specification.

2. The TURBOchannel interface ASIC in Programmed I/O

Programmed I/O transactions (PIO) occur in response to a system processor's load/store instructions to the option's slot in the physical address space. PIO transactions transfer one data word per transaction. Virtually all TURBOchannel options make use of PIO for control functions, as when host-resident driver software controls the TURBOchannel device by writing and reading option control and status registers. Some TURBOchannel options use PIO to transfer data as well.

2.1 The TURBOchannel PIO transaction protocol

The TURBOchannel interconnect protocol for PIO transactions is simple, and so it is already easy to implement the option side of the PIO protocol, even without a special ASIC. The system initiates PIO transactions by simultaneously asserting the option's select line *tcsel,* placing a 27bit address offset on *tc.ad[31..5]* and a four-bit byte mask on *tc.ad [4..1],* and asserting or deasserting *tc.~write* according to whether the transaction is a PIO write or a PIO read. The address offset and byte mask are held on *tc.ad [31..1]* for only one TURBOchannel clock cycle, because the lines must be used subsequently for the data word. The *tc.~write* state is held to the end of the transaction.

The address offset is a word count relative to the origin of the option's slot in the physical address space. The byte mask allows selecting any of the four bytes comprised in the 32-bit TURBOchannel data word; a bit-value of 0 in the byte mask enables the corresponding byte of the data word.

In the case of a PIO write transaction, the system places the data word on the *tc.ad [31..0]* lines in the second TURBOchannel clock, and holds it until the option asserts *tc.~rdy* to indicate that it has accepted the transferred data, ending the transaction. In the case of a PIO read transaction, the option drives the addressed data onto the *tc.ad [31..0]* lines when it can, and simultaneously asserts tcrdy to signal the presence of data and end the transaction. For both PIO reads and PIO writes, a system-defined time-out period limits the time that a system must wait for a nonresponsive option. An option can refuse a PIO transaction by asserting *tc.~rdy* and *tc.~conflict* together. Such a conflict situation can occur when the option is committed to a DMA transaction when the system selects it for a PIO transaction.

2.2 Using the TcIA for Programmed 1/O

In using the TcIA, all of the TURBOchannel signals connect directly to the TcIA, and the option designer interfaces to the corresponding option interface. One possible benefit of using the TcIA for PIO is isolation from the TURBOchannel time domain, if desirable. Another important service that the TcIA performs for the option is latching the PIO address offset and byte mask, which the system presents packed on the *tc.ad [31..0]* for only one TURBOchannel clock cycle. The TcIA also unpacks these data, presenting the address offset and byte mask on *op. adr[19..0]* and *od.bm[3..0]* respectively.

For the most part, the remainder of the option interface signals involved in PIO are optionsynchronized versions of the corresponding TURBOchannel signals. For example, *op.~sel* is asserted by the TcIA when the system asserts *tc.~sel*, (except when the I/O address is in the TcIA Register File), and *op.~write* is asserted by the TcIA when the system asserts *tc.~write.* The option asserts *op.~rdy* to acknowledge receiving data in the case of a PIO write or to signal valid data on *op.ad [31..0]* in the case of a PIO read. The option can signal a conflict on *tc.~conflict* and *tc.~rdy* by asserting *op. ~conflict* and *op.~rdy.*

Thus, when using the TcIA for PIO, the option designer need supply logic only for decoding addresses and generating corresponding select signals and for managing the handshake signals.

3. The TclA in TURBOchannel DMA operations

Many high-performance options will use DMA to move data more efficiently. DMA operations move blocks of data between buffers in system memory and buffers on the option. Typical DMA operations, as defined by software, are "read a disk block", or "fill a frame buffer". A single DMA operation will generally comprise a number of TURBOchannel DMA transactions. In a TURBOchannel DMA transaction, after a few initial cycles of handshaking and addressing, the data is moved synchronously at one word per clock cycle. Each DMA transaction moves a block of data of a size determined by the option, up to a system-defined limit. Larger DMA blocks increase effective bandwidth. The maximum DMA transaction burst size in existing TURBOchannel implementations is 128 32-bit words.

It is worth reviewing the nomenclature that has been used to specify the direction of DMA transfers. In the language of the TcIA specification, DMA *transmit* operations move data from system memory to the option, and DMA *receive* operations move data from the option to system memory. However, the TURBOchannel specification uses the terms DMA *read* transaction for a transaction that moves data from memory to the option, and DMA *write* transactions for the opposite direction. Therefore, DMA transmit operations are made up of DMA read transactions and DMA receive operations are made up of DMA write transactions.

3.1 The TURBOchannel DMA protocol

In the TURBOchannel DMA protocol, the option initiates a DMA transaction by asserting *tc. ~wReq* for a write transaction or *tc.~rReq* for a read transaction. Then, the option waits for the system to grant the use of the *tc.ad* lines by asserting *tc.~ack,* which could occur in the same clock cycle as the *tc.~wReq* or the *tc.~rReq.* In the cycle following the assertion of *tc.~ack,* the option places an address on the *tc.ad[31..0]* lines for one cycle. The address is the physical address of the first memory buffer location involved in the block transfer. The successive words of a DMA transaction move to or from system memory at successive physical addresses.

In the case of DMA write transaction, the option begins driving data the *tc.ad [31..0]* in the clock cycle following the address cycle. In the case, of a DMA read transaction, the system deasserts *tc.~ack* in the address cycle, and reasserts this signal when the system drives the first data word on *tc.ad [31..0]* perhaps after a delay due to system memory access latency. In either case, the data transfer continues at one word per *tc.clk* cycle up to a DMA burst size that is determined by the option and signaled by the number of clock cycles through which the option continues asserting *tc. ~wReq* or *tc.~rReq.*

While it is the option that initiates and generally controls a DMA transaction at the protocol level (paced by the system-driven clock), the option is usually acting in response to a DMA operation request passed to it, through PIO transactions, by driver software running in the host system. However, there also exist so-called intelligent DMA peripherals, which may well initiate DMA transactions more autonomously, and which may generate their own system memory addresses. (The AMD LANCE Ethernet controller is such a peripheral device). Section 3.4 discusses the TcIA's special provisions to facilitate its use in intelligent DMA controllers.

3.2 Using the TcIA for DMA operations

The TcIA presents the option with a DMA interface equivalent to interaction with a pair of synchronous 15-word FIFOs. The TcIA handles all requesting, addressing, counting, and handshaking with the TURBOchannel. In particular, the TcIA performs the option-side activity of the TURBOchannel DMA transaction protocols described in the preceding section.

In a DMA read transaction, the TcIA fills the Transmit FIFO from the TURBOchannel. The TcIA asserts *op.~trValid* to inform the option when data is present in the Transmit FIFO to be read out by the option. The option removes data from the Transmit FIFO by asserting *op.~trDack.* Whenever the option asserts *op.~trDack* while *op.~trValid* is asserted, then on the following *op.clk* cycle the TcIA presents on *op.ad[31..0]* the next option-word from the Transmit FIFO, through the unpacking functional unit. In case the option-word size is byte or half-word, it is presented on the corresponding low-order lines of *op.ad [31..0]* Of course, when the optionword is a byte or a half-word, then four or, respectively, two option clock cycles are needed to remove each word from the FIFO.

In a DMA write transaction the TcIA sends data from the Receive FIFO to the TURBOchannel. The TcIA asserts *op.~rcvValid* to signal the option when space is available to be filled in the Receive FIFO. Then the option can write data to the FIFO by driving an option-word onto *op.ad [31..0]* (or the appropriate low-order lines thereof) and asserting *op.~rcvDat.* In the partial word case, successive option-words are accumulated to a full word in the packing unit and then the full word is moved to the FIFO.

During DMA operations, the TcIA is controlled by a number of defined fields in registers in the TcIA Register File. Some of these control and status fields were mentioned above in Section 1, and a few others will be specifically mentioned below. A simple DMA operation is initiated when the driver software sets certain fields in the Register File, using PIO transactions. (In the case of intelligent DMA controllers, some of the initial field values are set by the option, as described further in Section 3.4.) Once the DMA operation is initiated, the TcIA automatically manages the DMA transactions needed to complete the operation.

For example, a DMA transmit operation is controlled by a 24-bit Transmit Word Counter (TWC), which is initialized to the number of words to be transferred in the operation and automatically decremented for each transmitted word. Once the DMA transmit operation is initiated, the TcIA continues to initiate DMA read transactions whenever there is sufficient free space in the Transmit FIFO, so long as TWC remains greater than 0. The amount of free space in the Transmit FIFO needed before initiating a DMA read transaction must be at least the burst size of the transaction, unless the option is capable of draining the FIFO at a rate of one word per TURBOchannel clock cycle.

All the DMA read transactions after the first will have a common transmit burst size; the first transaction may be shorter. Both the common burst size and the initial burst size are determined by fields in the Register File that can be set by the controlling software. (Section 3.3 discusses the considerations that affect the choice of the common and initial burst sizes.)

Similarly, a DMA receive transaction is controlled by a Receive Word Counter (RWC), which is initialized to the number of words to be transferred in the operation and automatically decremented for each received word. Once the DMA receive operation is initiated and so long as RWC remains greater than 0, the TcIA continues to initiate DMA write transactions whenever the number of words present in the Receive FIFO reaches a minimum receive burst size. The minimum receive burst size is defined by another field in the Register File, and may be l, 2, 4, or 8 words. Because the option may continue filling the Receive FIFO while a DMA write transaction is in progress, the actual burst size will generally be larger than the minimum. The TcIA terminates a DMA write transaction when the Receive FIFO is empty, when the RWC reaches 0, or (for a reason discussed in Section 3.3) when the destination address in system memory reaches a 64-word boundary.

The TcIA also initiates a DMA write transaction to flush the Receive FIFO when it has held data for a sufficiently long time, as determined by a receive flush counter contained in another Register File field.

3.3 DMA addressing and the gather/scatter feature

As mentioned in Section 3.1, each TURBOchannel DMA transaction includes one address cycle, preceding the data burst, in which the option drives onto *tc.ad[31..0]* the physical address in system memory of the first word of the burst. In a TcIA-based option, the TcIA manages the generation of this address for each DMA transaction of the DMA operation. The Register File contains a Transmit Data Pointer (TDP) and a Receive Data Pointer (RDP), which are initialized by the controlling software for a DMA operation and subsequently automatically incremented for each word transferred. The current values of these pointers provide the addresses to be driven on *tc.ad[31..0]* for each DMA transaction. In case a DMA transaction is interrupted because of an error, the current value in the TDP or RDP serves to locate the error exactly in the data stream.

The use of physical, rather than virtual, addressing on the TURBOchannel lines contributes substantially to the simplicity and efficiency of the TURBOchannel DMA protocol. However, operating systems generally provide DMA buffers that are virtually contiguous, but not necessarily physically contiguous. Hence, the use of physical addressing adds some complexity to the required behavior of the TURBOchannel option in DMA operations. First, because a DMA transaction contains just one address cycle and uses physically consecutive addresses for the words in the data burst, it is essential that a DMA transaction not cross physical page boundaries. Second, there must be provisions for reloading the TDP and RDP when DMA operations cross page boundaries. The TclA includes features that meet both of these requirements.

The means that the TclA uses to prevent DMA transactions from crossing page boundaries depend on the assumption that memory pages have a common size (in bytes) that is a power of two and are aligned on byte addresses that are multiples of that power of two. In the case of DMA receive operations, the TclA prevents transactions from crossing page boundaries simply by terminating a DMA write transaction whenever the TDP reaches a 64-word boundary. This will satisfy the requirement so long as the physical page size is at least 256 bytes. In the case of DMA transmit operations, the TcIA uses a common burst size for all transactions except possibly the first. The common burst size is a power of two (less than the page size) and the burst size of the first transaction is chosen so that the transaction ends at the address that is the next multiple of the common burst size above the initial address in TDP.

To support efficient reloading of the address pointers when DMA operations cross page boundaries, the TcIA Register File includes two 32-bit pointers, called the Transmit Scatter/Gather Table Pointer (TSGP) and the Receive Scatter/Gather Table Pointer (RSGP). When the controlling software initiates a DMA operation, it must set up in system memory a table of the physical addresses of the successive pages of the DMA buffer, and it must load the address of this table into the corresponding Scatter/Gather Table Pointer in the TcIA. Then, in the course of the DMA operation, whenever the TDP or RDP reaches a page boundary, the TcIA automatically reloads the TDP or RDP

from the memory address contained in the TSGP or RSGP (by means of a DMA read transaction), and increments the TSGP or RSGP.

3.4 Intelligent DMA peripherals

The TcIA includes some features to accommodate so-called intelligent DMA peripherals, which can initiate DMA operations autonomously and which can provide their own DMA buffer addresses, or partial addresses. DMA operations generally are governed by numerous data items that the TcIA holds in the Register File. In the case of simple (i.e. non-intelligent) DMA transactions, the controlling data is set in the Register File by the driver software when it initiates the DMA operation. For intelligent DMA operations, the option needs provisions to load some of the controlling data, in particular the DMA addresses, word counts, and burst sizes.

When the option asserts *op.~rcvAdr* (but not *op.~trAdr*), then the address on *op.ad[31..0] is* loaded into the Receive FIFO on the next option clock cycle. When this word reaches the head of the FIFO, it is transferred to the Receive Data Pointer, so the next DMA write transaction will write to this address.

Similarly, when the option asserts *op.~trAdr* (but not *op.~rcvAdr*), then the address on *op.ad [3..0] is* loaded into the Receive FIFO on the next option clock. When this word reaches the head of the FIFO, and the Transmit Word Count is zero, it is transferred to the Transmit Data Pointer, so the next DMA Read transaction will read from this address.

The intelligent DMA option sets the Transmit DMA word count and burst size by packing these into a single word on *op.ad [31..0]*. When the option asserts *op.~trAdr* and *op.~rcvAdr* simultaneously, then on the next option clock cycle the word in *op.ad [31..0] is* loaded into the Receive FIFO. When this word reaches the head of the FIFO, the high-order eight bits are loaded into the burst size field and the low-order 24 bits to the Transmit Word Count field.

Intelligent DMA peripherals may supply addresses that are shorter than 32 bits, being low-order bits of the full address. The TcIA Register File includes a DMA pointer mask that can be used to specify the valid address bits supplied by the option. The remaining high-order bits of the DMA data pointers are not changed when these pointers are loaded with the option-supplied address. Usually, these high order bits will have been set by driver software in the normal way.

4. Other TcIA features

4.1 Interrupts

The TURBOchannel provides just one interrupt line *tc.~int* by which the option may request to interrupt the system processor. Completion of a DMA operation is the most frequent condition for which options request interrupts. Errors and exception conditions also usually warrant interrupts.

The TcIA Register File includes two registers to support the use of interrupts. The Option Interrupt Register (OIR) contains 26 bits which each indicates a defined transition condition, exceptional event, or other status information. Four of these bits are wired directly to the *op.gpi[3..0]* input lines to give the option the ability to define further interrupt conditions external to the TcIA.

The Option Interrupt Mask Register (OIMR) is a corresponding mask to allow enabling the various interrupt conditions defined for the OIR under software control. In the TcIA-based option, at all times the *tc.~int* signal indicates the logical OR of all the bits of the word obtained by ANDing the OIR with the OIMR. When this line is asserted, the interrupt handler can attempt to determine the cause of the interrupt by reading OIR and OIMR.

4.2 Parity

In the TURBOchannel protocol, parity error protection is optional for both system and option. When both participants support it, normal parity is odd word parity for all data and addresses on ad[P, 31..0J. That is, the sum of the 33 bits should be odd.

The TcIA has built-in parity detection and generation circuitry. The TcIA Register File includes a parity enable bit. Normally this bit would be set in the boot-time configuration process, when the system and the option mutually determine whether they both support parity. When parity is enabled then parity violations detected by the TcIA automatically generate the appropriate interrupt.

4.3 General purpose I/O signals

The TcIA option interface contains four input signals and eight output signals by which the system can communicate directly with the option through the Register File. As mentioned in Section 4.1, the *op.gpi[3..0]* lines connect to bits in the Option Interrupt Register. Therefore, they can be used to generate interrupt requests under control of the Option Interrupt Mask Register. Or they can simply be read from this register by the system through an I/O read transaction.

Similarly, the *op.gpo[7..0]* are driven by the TcIA to reflect bits in another option control register that the system can write to.

Effectively, these general purpose I/O signals provide the designer with an external control and status register without having to provide external decoders and latches.

4.4 Concurrency

The TcIA's resources for Transmit DMAs, Receive DMAs, and PIOs are largely independent. Of course, all of these functions use the ad lines on both sides, and so no more than one transaction can be in progress at a time. However, a transmit DMA operation and a receive DMA operation can be in progress concurrently, with their transactions interleaved, and PIO transactions can be interleaved with the transactions of a DMA operation as well. Moreover, a transmit transaction can partially overlap a receive transaction, as when the TcIA is filling the Transmit FIFO while the option is filling the Receive FIFO.

5. Development support

To help designers interested in using the TcIA, Digital is making available a test board and associated documentation. The single-slot test module is designed to exercise the TcIA in almost all of its possible modes. The test board package includes the TcIA Functional Specification, schematics, and the defining PAL equations, sample diagnostic firmware, and sample driver code. The sample firmware and driver code will be invaluable as starting points for developing the firmware and driver code for other TcIA-based options.

In general, Digital provides technical support for TURBOchannel developers through its TRI/ADD program, which can be contacted at 415-617-3452. Further detailed information on the TcIA can be obtained from Bill Samaras at the TURBOchannel program office of Digital Equipment at 508-493-7322. -