LOCKHEED MARTIN

Advanced Technology Laboratories
1 Federal Street – A&E 2W
Camden, NJ 08102
(609) 338–4250

Application Notes – RACEway to SVI External Network Interface

Date: July 30, 1996

Author: Greg Buchanan

Contact: Janet E. Wedgwood (janet.e.wedgwood@lmco.com)

Introduction

The RACEway to SVI External Network Interface (ENI) is designed to serve as the network fabric interface component for a RASSP MYA Reconfigurable Network Interface (RNI). The ENI acts as a fully compliant half–duplex, RACEway port on one side, and a fully compliant SVI port on the opposing side. The ENI fully translates and converts messages bidirectionally from one port to the other. When connected to an RNI bridge element, incoming messages from the RACEway port will be translated and formatted as required for the opposing or "bridged" ENI; the bridge element will likewise translate messages from the bridged ENI, which are destined to become outgoing messages on the RACEway.

For a detailed description of the RACEway protocol, see "RACEway Interlink Specification (ANSI/VITA 5–1994).

Supported Functions

The RACEway ENI is fully compliant with the VITA RACEway specification, and supports all Raceway network transactions except Broadcast Mode and Priority Kill (these functions exist in the RACEway interface, but are NOT supported at the SVI interface). RACEway transactions that are supported are: Write, Read, Split–read, and Read–Modify–Write. In addition, the ENI automatically terminates and restarts all transactions at every 2k–byte address boundary, supports 25 microsecond timeouts, and supports long and short addressing modes. With this this rich set of operating features, it may be difficult to fully support even a subset the RACEway operations across an RNI boundary, where the opposing ENI may not directly support the same features. Should the network on the opposing side of the RNI not have certain RACEway capabilities, the RNI bridge will have the task of transforming these network services into a sequence of supported features on the opposing side. For example, a Read–Modify–Write command coming from the RACEway could have to be transformed into a separate read and write to the target node.

SVI Data Width

Since the RACEway word length is 64–bits, the SVI data_in and data_out signals are also 64–bits wide. If the SVI data_in port is required to interface with an SVI port of differing data width, a width converter block can be added to the front–end of the SVI slave block (there is currently a 1–to–1 width converter in the SVI slave model as a place–holder).

Code Synthesizability

All portions of the RACEway ENI were written in synthesizable RTL style.

Operation

A RACEway message transaction consists of an exchange of a sequence of 4–byte half–word data on a bidirectional data bus between a RACEway master and a RACEway slave node. All RACEway messages are initiated with the master asserting its Request signal, and sending a route half–word and a address half–word toward the slave. The slave then responds via handshaking signals to indicate that the transaction may proceed. If the master is requesting a WRITE, data follows from the master in two half–word pairs per data word; if the master is requesting a READ, data follows from the slave in two half–word pairs per data word. The RACEway handshaking signals allow data to be throttled in both directions, as well as for the termination of the transaction.

The RACEway ENI performs word packing from the RACEway port to the SVI port, and unpacking of SVI messages to the RACEway. RACEway 32–bit half–words are packed into 64–bit full words on the SVI; 64–bit words are unpacked into half–words for transmission onto the RACEway. There is no need for word alignment. Address and route half–words are packed together in a single SVI word, with the address in the position of the most–significant half of the word. Return route half–word transmissions are packed alone in the least–significant half of the word, with padding bits of arbitrary value in the most–significant half of the word.

As per the RACEway specification, the RACEway ENI introduces a stall condition whenever the data being transmitted (read or write) reaches a 2k–byte boundary. This stall causes the intervening network resources between master and slave to become available for another potential message. Once the other message completes, or if no other message blocks the channel, the message is re–established where it left off. The ENI also supports long–period timeouts in order to fairly distribute network resources. If a master does not complete a transaction in 25 microseconds, it stalls in a manner identical to a 2k address stall, and picks up the transaction after the stall or the completion of an intervening message.

The RACEway ENI supports long and short addressing modes, where portions of the route field can be used to extend the address field. The route and address field boundaries must be configured into the ENI by specifying the generics ’long_addr_word’ and ’msb_addr_shifts’. To use portions of the route field for addressing, long_addr_word must be equated to ’TRUE’, otherwise it must be equated to ’FALSE’. When long_addr_word is TRUE, msb_addr_shifts must be equated to the number of three–bit route sub_fields that will be used for high–order address bits.

The RACEway ENI also supports RACEway global resets. When the ENI receives a system reset from the SVI port, it asserts the RACEway reset, which in turn is broadcast to all RACEway nodes in the network.

The sequences of SVI transactions that are necessary to negotiate RACEway messages are relatively straightforward SVI decompositions of the RACEway message, and are described in the following section. A RACEway READ slave however, uniquely requires a ’STOP READ REQUEST’ command followed by an ’ABORT DUE TO DESTINATION LOST’. When a RACEway (non–split) READ slave receives the necessary RACEway signals to initiate a read request, the RACEway ENI passes the read request to the RNI Bridge element via the SVI. The Bridge element then eventually feeds a stream of read data words back over the SVI to the RACEway ENI. When the RACEway master decides that enough read data has been received, the READ is terminated with the proper sequence of RACEway handshaking signals. Meanwhile, since a RACEway read has no predetermined length, the Bridge Element (and the storage element behind it) continues to attempt to feed read data to the RACEway ENI. In order to kill this process on the bridge side, when the RACEway ENI receives the READ termination, it sends an SVI message containing a ’STOP READ REQUEST’ to the Bridge element. The Bridge ceases streaming read data to the RACEway ENI, and transmits an ’ABORT DUE TO DESTINATION LOST’ to the SVI slave to force it to cease expecting SVI data (since the data has stopped, the message cannot be terminated with the usual svi_last_word_in signal).

Message Transactions

The following describes the composition of messages on the SVI port of the ENI only. The composition of the message on the RACEway port is described in the RACEway Link Specification.

<u>To Initiate a WRITE as a RACEway Master:</u>

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
1. SVI command ”External Write” (cmd_value = 0)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address
3. An arbitrary length sequence of data words (8 bytes). (last data word sent
   coincident with assertion of svi_last_word_in)

<u>To Respond to a WRITE as a RACEway Slave:</u>

– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:
1. SVI command ”External Write” (cmd_value = 0)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address
3. An arbitrary length sequence of data words (8 bytes). (last data word rcv’d.
   will appear coincident with assertion of svi_last_word_out)

<u>To Initiate a READ as a RACEway Master:</u>

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
1. SVI command ”External Read Request” (cmd_value = 3)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address
3. A data word with 2 LSB bytes containing the count of no. of data words
   requested (coincident with assertion of svi_last_word_in).

– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:

4a. SVI command ”External Read Response” (cmd_value = 5)
5a. A sequence of data words (8 bytes). (The last word arrives coincident with
svi_last_word_out)

– OR, IF SPLIT READ –

4b. SVI command ”External Read – Request Split Read” (cmd_value = 8)
5b. A ”last word” of ”don’t care” value, which is discarded. (rcv’d
coincident with assertion of svi_last_word_out)
and . . .

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*

6b. SVI command ”External Read – Response to Split Read Request” (cmd_value = 9)

7b. Header word (8 bytes); MSB 4 bytes = ”don’t care”, LSB 4 bytes = return Race route (coincident with assertion of svi_last_word_in).

<u>To Respond to a READ as a RACEway Slave:</u>–

– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:
1. SVI command ”External Read” (cmd_value = 3)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address (coincident with assertion of svi_last_word_in).

– IF SLAVE RESPONDING WITH READ DATA –

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
3a. SVI command ”External Read Response” (cmd_value = 5)
4a. A continuous sequence of data words (8 bytes). (The SVI transaction is terminated by an abort when a ’Stop Read Request’ is rcvd by the Bridge element)

– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:
5a. SVI command ”Stop Read Request” (cmd_value = 10)
6a. A ”last word” of ”don’t care” value, which is discarded. (rcv’d coincident with assertion of svi_last_word_out)

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
7a. SVI command “Abort due to Destination Lost” (cmd_value = 4) (since the ”last word” has already been sent, the SVI transaction must be aborted)

– OR, IF REQUESTING SPLIT READ –

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
3b. SVI command ”External Read – Request Split Read” (cmd_value = 8)
4b. A ”last word” of arbitrary value, which is discarded. (sent coincident with assertion of svi_last_word_in)

– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:
5b. SVI command ”External Read – Response to Split Read Request” (cmd_value = 9)
6b. Header word (8 bytes); MSB 4 bytes = ”don’t care”, LSB 4 bytes = return race route (coincident with assertion of svi_last_word_in).

<u>To Initiate a READ–MODIFY–WRITE as a RACEway Master:</u>

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*
1. SVI command ”Read–Modify–Write Request” (cmd_value = 7)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address (after which time SVI master idles (keeping SVI transaction open), while receiving read data

–– *RNI Bridge element receives an SVI packet from RACEway ENI containing*:
3. SVI command ”External Read Response” (cmd_value = 5)
4. A single read data word (8 bytes). (arriving coincident with svi_last_word_out)

– *RNI Bridge element completes the SVI packet to RACEway ENI, with:*
5. A single write data word (8 bytes); (sent after the read word is received and processed by the bridge, and send coincident with assertion of svi_last_word_in).

To respond to a READ–MODIFY–WRITE as a RACEway Slave:

— *RNI Bridge element receives an SVI packet from RACEway ENI containing*:

1. SVI command ”Read–Modify–Write Request” (cmd_value = 7)
2. Header word (8 bytes); MSB 4 bytes = Race route, LSB 4 bytes = Race address
   (after which time SVI slave idles (keeping SVI transaction open), while transmitting read data

– *RNI Bridge element sends an SVI packet to RACEway ENI containing:*

3. SVI command ”External Read Response” (cmd_value = 5)
4. A single read data word (8 bytes). (sent coincident with svi_last_word_in)

— *RNI Bridge element receives completion of the SVI packet from RACEway ENI, with*:

5. A single write data word (8 bytes); (rcv'd after the read word is sent to the bridge, and rcv'd coincident with of svi_last_word_out).

Signals

The following signals comprise the RACEway interface of the ENI:

xbio(31:0)	RACEway bidirectional data bus
xreqi	RACEway request in
xreqo	RACEway request out
xrplyio	RACEway bidirectional reply
xstrobio	RACEway bidirectional data strobe
xrdconio	RACEway read connect
xresetio_n	RACEway reset
xsynci_n	RACEway phase synch.
xclki	RACEway clock in

The following signals comprise the SVI interface of the ENI:

SIGNAL	DESCRIPTION
svi_data_out(63:0)	SVI outgoing data
svi_last_word_out	asserted coincident with last word of outgoing SVI message
svi_clock_out	outgoing SVI synchronization clock
svi_xfer_request_out	asserted during entire outgoing SVI message
svi_ready_in	destination asserts to allow transmission of SVI data
svi_data_valid_out	asserted coincident with each outgoing SVI word
svi_data_in(63:0)	SVI incoming data
svi_last_word_in	asserted coincident with last word of incoming SVI message
svi_clock_in	incoming SVI synchronization clock
svi_xfer_request_in	asserted during entire incoming SVI message
svi_ready_out	asserted to allow transmission of SVI data
svi_data_valid_in	asserted coincident with each incoming SVI word
svi_slave_abort_in	asserted to abort an ongoing read transaction
svi_sreset_in	system reset; resets SVI master and slave

RACEway ENI VHDL Model – Theory of Operation

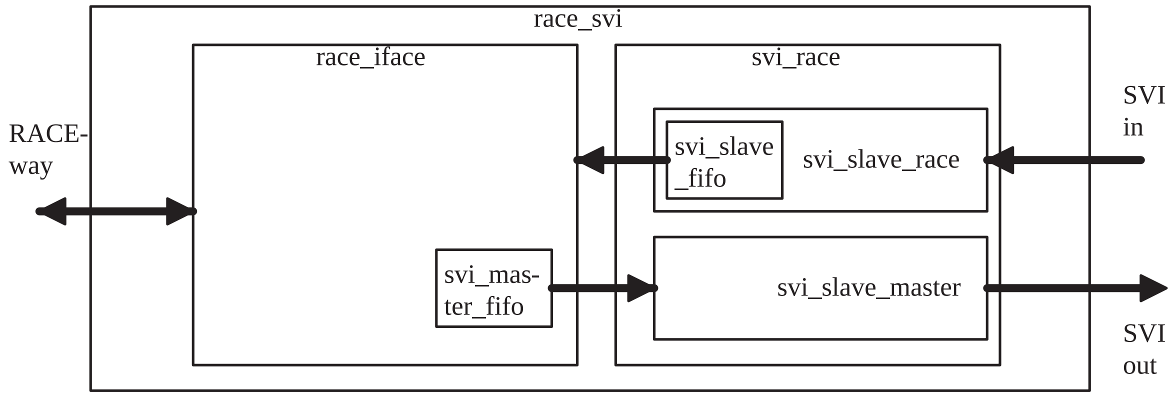

Fig. 1 – Block diagram – VHDL model of RACEway ENI

For an in–depth understanding of the RACEway ENI, refer to the VHDL code. The code is fully commented, and all internal signals, flags, and ports are described. A less detailed overview of the theory of operation of the constituent blocks of the ENI follows.

race_iface:

The race_iface block is responsible for providing a direct interface to the RACEway; it receives data from the SVI half of the ENI, formats it for transmission on the RACEway, provides handshaking with the connecting RACEway port, and receives data on the RACEway and prepares it for the SVI interface. The race_iface incorporates a data buffer, svi_master_fifo, which buffers incoming data on the RACEway and makes it available for reading by the SVI interface. The depth of this FIFO is set by the constant 'svi_master_fifo_depth' in race_iface_types_pkg.vhd. Data, received every other cycle on the RACEway in 32–bit half–words, is packed into a packing register before being written as whole words into the FIFO. Incoming data from the SVI interface is in whole words. As described above, the RACEway ENI supports long and short addressing modes, and is configured by specifying the generics 'long_addr_word' and 'msb_addr_shifts'.

The race_iface block is sequenced through RACEway transaction operations by a state machine with the following states and functions:

State	Functions
idle	Waiting to source or sink RACE message
xmit_route	Transmit RACEway route half–word (R/W master)
xmit_sh_route	Transmit RACEway shifted route half–word (R/W master)
xmit_addr	Transmit RACEway address half–word (R/W master)
xmit_data_wrt	Transmit RACEway write data (R/W master)
xmit_split_read_route	Transmit RACEway route as split–read response (R/W master)
rcv_data_read	Receive RACEway read data (R/W master)
stall	Stall from: 2K address, timeout, or kill
rcv_route	Receive RACEway route half–word (R/W slave)
rcv_sh_route	Receive RACEway shifted route half–word (R/W slave)
rcv_addr	Receive RACEway address (R/W slave)
rcv_data_wrt	Receive RACEway write data (R/W slave)

xmit_data_read	Transmit RACEway read data (R/W slave)
split_rd_req	Transmit RACEway split–read request (R/W slave)
master_reset	Broadcast RACEway reset (R/W master)

svi_race:

The svi_race block is responsible for providing the SVI interface to the RACEway ENI; it receives data from the RACEway half of the ENI, formats it for transmission on the SVI, and receives data on the SVI and prepares it for the RACEway interface. The svi_race incorporates an SVI master and slave (svi_master_race and svi_slave_race), and the SVI slave contains a data buffer, svi_slave_fifo, which buffers incoming data on the SVI and makes it available for reading by the RACEway interface. The depth of this FIFO is set by the constant 'svi_slave_fifo_depth' in race_iface_types_pkg.vhd.

The svi_master_race block is sequenced through SVI transaction operations by a state machine with the following states and functions:

State	Functions
idle	Waiting to source or sink SVI message
transfer_cmd	Transmit SVI command
transfer_header	Transmit RACEway route and address on SVI
transfer_data	Transmit RACEway data on SVI
wait_for_end_of_read	“Idle” while waiting for read to end

Because of the wide variety of RACEway commands supported by the ENI, the operation of the SVI master is further qualified by a mode_type state:

SVI Master Mode	Mode
rcv_data_wrt	Receive data from a RACEway write (R/W slave)
read_mod_wrt_master	Initiating a read–modify–write (R/W master)
rcv_data_read	Receive data from a RACEway read (R/W master)
read_mod_wrt_slave	Initiating a read–modify–write (R/W slave)
split_read_rtn_rte_req	Receiving (split read) return route request (R/W master)
split_read_rtn_rte_resp	Receiving (split read) return route response (R/W slave)
xmit_data_read_req	Transmit request to read data (R/W master)
xmit_stop_read_req	Transmit request to stop read (R/W slave)

RACEway ENI Testbench

The RACEway ENI testbench, Figure 2, is composed of two RACEway ENI's connected via their RACEway ports. Tests are run by stimulating the SVI_in ports and observing the SVI_out ports on both ENI's. If the ENI is modeled properly, write data presented to the SVI_1 input will appear on the SVI_2 output. Likewise, a read request applied to the SVI_1 input will propagate out the SVI_2 output, and read data applied to SVI_2 input will propagate to the SVI_1 output, along with the requisite SVI commands and SVI data–flow signals. Each of the testbench tests contains a compare routine which compares the data on each of the SVI_out ports to a file containing the expected results. As soon as all the expected results are received from each SVI output port, a message to the

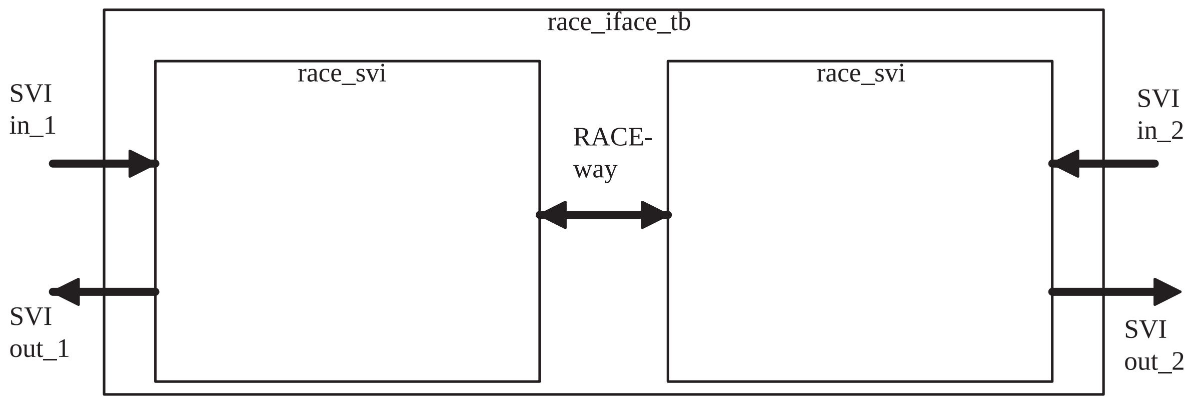

FIG. 2 – Block diagram – VHDL Testbench for RACEway ENI

console indicates that the end of the compare file has been reached with no errors. Alternatively, any miscompares are also identified with error messages to the console.