LOCKHEED MARTIN

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

TO: File
FROM: Greg Buchanan
DATE: July 31, 1996
SUBJECT: Application Notes – Myrinet to SVI External Network Interfac

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

Introduction

The Myrinet to SVI External Network Interface (ENI) is designed to serve as the network interface component for a RASSP MYA Reconfigurable Network Interface (RNI). The ENI acts as a fully compliant full–duplex, 1–meter Myrinet 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 Myrinet 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 Myrinet.

For a detailed description of the Myrinet protocol, see “Myrinet Link Specification” by Myricom, Inc.

Due to the proprietary nature of the Myrinet model, this model is not available to the general public.

Supported Functions

As all data transfers on the Myrinet are WRITEs, any incoming message on the Myrinet In port will be converted to an “External Write Request” on the SVI Out port. Likewise, an “External Write Request” is the only SVI command that will be accepted on the SVI IN port. Should the network on the opposing side of the RNI have capabilities beyond a WRITE, the RNI bridge will have the task of transforming these other “commands” into a sequence of WRITEs. For example, a READ command coming from the opposing ENI, would have to be transformed into a WRITE of a message requesting data from the target node; this node would in turn write the the requested “read” data back to the requestor.

SVI Data Width

Since the Myrinet word length is 32–bits, the SVI data_in and data_out signals are also 32–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

In general, the VHDL code for elements of an RNI — including the ENI — are intended to be written in a fully synthesizable RTL style. Portions of the Myrinet ENI however, are non–synthesizable. This is due primarily to the fact that Myrinet signals have a 1.2 volt transitions, rendering the input circuitry unsynthesizable (currently available target technology is only 5V or 3.3V). In addition, the asynchronous nature of the Non-Return-to-Zero (NRZ) data makes it difficult to produce synthesizable VHDL code for the input logic. Therefore, the code for the Myrinet input logic is written in a behavioral, non–synthesizable style.

Operation

A Myrinet packet consists of a sequence of nine–bit "flits" (flow control units) consisting of: a header of zero or more route flits; four packet type flits; the payload containing zero or more data flits; one data flit containing the CRC–8 byte of the previously transmitted flits; and a tail flit containing a GAP control character. Each flit contains one byte of data along with a ninth "data" bit, which when deasserted identifies the contents as a control character; only the GAP control character is recognized by the Myrinet ENI. The MSB of the first byte (8th bit) of a packet, when asserted, identifies the header route flits. The first byte with an MSB of zero following the route bytes marks the beginning of the four packet type flits, which are followed by the data payload, CRC flit, and GAP symbol.

The Myrinet ENI performs word packing and alignment of incoming Myrinet data flits, on a 4–byte/32–bit word basis. Packing is the process of grouping four contiguous bytes of Myrinet data into a single SVI data word. Aligning is the process of ensuring that the most significant byte of a Myrinet word appears in the most significant byte of the SVI data word. Since the number of header flits may be of an arbitrary quantity, the first packet type flit may not arrive aligned on a 4–byte interval. In order to perform proper word alignment, the Myrinet ENI pads the last header route word with sufficient pad bytes so as to align the first packet type byte on a word boundary. Since route bytes all have their MSB's set, any non–word–aligned bytes following a route byte and having ZERO–valued MSB's are identified as pad bytes. The first word with an MSB (bit 31) of ZERO is identified as the packet type word; all following flits are interpreted as data until a GAP character is received, which terminates the packet. The GAP symbol also indicates that the preceding data flit was the CRC flit. On the SVI side, the byte following the last byte of payload data is the EXOR of the *received* CRC–8 byte with the *calculated* CRC–8 of the received data; any value other than ZERO indicates the occurrence of a transmission error. The last SVI word (occurring coincident with the svi_last_word_out signal) contains the number of pad bytes in the preceding word, not including the EXOR'ed CRC byte.

A packet which begins with a packet type flit received by a Myrinet switch, is assumed by the switch to be a packed not intended for the switch, and is dropped by the switch. Likewise, a packet that begins with a route flit received by a Myrinet node (such as a LANai), is assumed by the interface to be intended for a switch, and is dropped by the interface. Since the opposing ENI on the RNI may be connected either to a switch or a destination node, a given Myrinet ENI must know whether to drop packets which begin with either packet type or route flits. This is accomplished by passing a VHDL generic to the ENI model; set 'bridge_to_switches' to TRUE if the Myrinet ENI will be bridging to a switch, set 'bridge_to_switches' to FALSE if the Myrinet ENI will be bridging directly to a sourc/destination node.

The Myrinet ENI also supports long–period timeouts in order to detect and respond to source or receiver–blocked packet situations. If the timeout limit is reached by either a Myrinet source or receiver with no sent or received data flits once a message has been initiated, the source/receiver returns to an idle state, resets any internal flags, and in a source–blocked scenario, the Myrinet source transmits a GAP symbol. The user determines the timeout period by equating the generic 'timeout_limit' to: 'sixteenth_sec', 'quarter_sec', 'one_sec', or 'four_sec'.

The Myrinet ENI requires incoming SVI messages to arrive in the same format as that in which it sends outgoing SVI messages, *except* no EXORed CRC byte is received. The last data word contains the number of pad bytes, while the preceding word contains one to four valid data bytes and from zero to three pad bytes.

Message Transactions

To Initiate a Myrinet Write:

– *RNI Bridge element sends an SVI packet to Myrinet ENI containing:*
1. SVI command ”External Write” (cmd_value = 0)
2a. If bridge_to_switches = TRUE:
– Header route word(s): one byte for each switch hop, with MSB = ONE for each route; last route word contains from zero to three pad bytes, with MSB = ZERO for each pad byte.
– OR –
2b. If bridge_to_switches = FALSE:
– NO header route words
3. Packet type word: four bytes, MSB = ZERO for each byte
4. Data word(s); zero or more; word aligned, with last word containing from one to four valid data bytes and from zero to three pad bytes as LSB bytes.
5. Last word: the LSB byte containing the number of pad bytes in the last data word.

– *Myrinet ENI responds by sending a Myrinet packet containing:*
1. Header route flit(s): one flit for each route byte received, if any.
2. Four packet type flits.
3. Data flit(s): one for each valid data byte received.
4. CRC flit: containing the CRC–8 value of *all preceding bytes of current message.*
5. GAP symbol flit.

To Receive a Myrinet Write:

– *Myrinet ENI receives a Myrinet packet containing:*
1. Header route flit(s): one flit for each switch hop on the opposing network side, if any.
2. Four packet type flits.
3. Data flit(s): one for each valid data byte.
4. CRC flit: containing the CRC–8 value of *all preceding bytes of current message.*
5. GAP symbol flit.

– *Myrinet ENI responds by sending an SVI packet to RNI Bridge element:*
1. SVI command ”External Write” (cmd_value = 0)
2. Header route word(s), if any: one byte for each switch hop, with MSB = ONE for each route; last route word contains from zero to three pad bytes, with MSB = ZERO for each pad byte.
3. Packet type word: four bytes, MSB = ZERO for each byte
4. Data word(s); zero or more; word aligned, with last word containing from one to four valid data bytes and from zero to three pad bytes as LSB bytes.
5. Last word: the LSB byte containing the number of pad bytes in the last data word.

Signals

The following signals comprise the Myrinet interface of the ENI:

SIGNAL	DESCRIPTION
send_clk	synchronizes data out
myrinet_out(7:0)	Myrinet output data
myrinet_d_out	Myrinet data ID: 1 = data; 0 = control character
myrinet_block_o	flow control for data out,; 1= block
myrinet_in(7:0)	Myrinet input data
myrinet_d_in	Myrinet data ID: 1 = data; 0 = control character
myrinet_block_i	flow control for data in; 1= block

The following signals comprise the SVI interface of the ENI:

SIGNAL	DESCRIPTION
svi_data_out(31:0)	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(31:0)	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_sreset_in	system reset; resets SVI master and slave

File Descriptions

All referenced files are currently located in /proj/rassp/gbuchana/rni/myrinet_svi.

The following files comprise the VHDL models for the Myrinet ENI and ENI testbench; they are listed in order of descending hierarchy:

File	Entity name	Description
myri_svi_tb.vhd	myri_svi_tb	Testbench; Myrinet ENI
myrinet_svi_ent/arch.vhd	myrinet_svi	Myrinet ENI
myrinet_iface_ent/arch.vhd	myrinet_iface	Myrinet interface
myrinet_master_ent/arch.vhd	myrinet_master	Myrinet interface master
myrinet_slave_ent/arch.vhd	myrinet_slave	Myrinet interface slave
slack_buffer_ent/arch.vhd	slack_buffer	Myrinet slack buffer
unpack_fifo_ent/arch.vhd	unpack_fifo	Myrinet unpack FIFO
svi_myrinet_ent/arch.vhd	svi_myrinet	SVI interface
svi_master_myrinet_ent/arch.vhd	svi_master_myrinet	SVI master
svi_slave_myrinet_ent/arch.vhd	svi_slave_myrinet	SVI slave

svi_types_pkg.vhd	Standard SVI package
svi_myrinet_types_pkg.vhd	SVI interface package
myrinet_iface_types_pkg.vhd	Myrinet interface package
my_hread.vhd	custom hex_read (text_io) pkg

Myrinet ENI VHDL Model – Theory of Operation

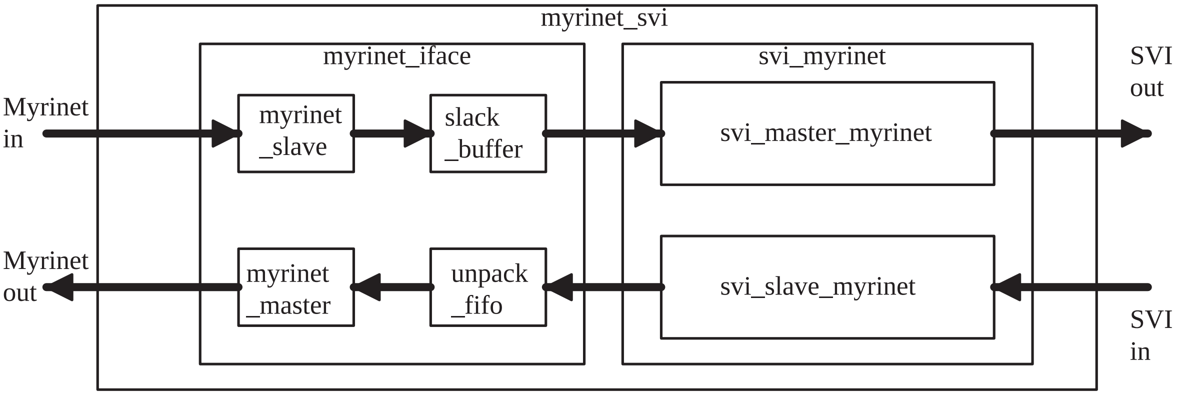

Fig. 1 – Block diagram – VHDL model of Myrinet ENI

For an in–depth understanding of the Myrinet 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.

myrinet_iface:

The myrinet_iface block is responsible for providing a direct interface to the Myrinet; it receives data from the SVI half of the ENI, formats it for transmission on the Myrinet, provides handshaking with the connecting Myrinet port, and receives data on the Myrinet and prepares it for the SVI interface. The myrinet_iface contains a myrinet slack buffer for incoming data from the Myrinet, and an unpacking FIFO which is used to unpack the single data byte for each Myrinet flit from the 32–bit SVI word. The myrinet_iface incorporates four blocks: a Myrinet master, a Myrinet slave, a Myrinet slack buffer, and an unpack FIFO.

myrinet_slave:

The myrinet_slave provides the direct interface to the Myrinet input port. The myrinet data input signals are encoded into an NRZ format, and arrive at the input ports asynchronously due to variations in output circuitry and transmission media. In order to capture the incoming asynchronous data, a sampling “window” is created upon the detection of a data transition on any data input. This window “closes” and data is sampled 2.25 ns after the first detected data transition; 2.25 ns is assumed to be the worst–case data skew in a 1–m Myrinet environment. Data is decoded from NRZ and written to the slack buffer, while the CRC–8 value of the incoming message is calculated. When a GAP symbol is detected on the input, the calculated CRC–8 is EXORed with the received CRC, and written to the slack buffer with the tail bit set.

The myrinet_slave block is sequenced through Myrinet slave transaction operations by a state machine with the following states and functions:

State	Functions
idle	Waiting to sink a Myrinet message

rcv_route	Receive Myrinet route flits
rcv_pkt_type	Receive Myrinet packet type flits
rcv_data	Receive Myrinet data flits

slack_buffer:

The slack buffer operates consistently with a general Myrinet slack buffer, as described in The Myrinet Link Specification. The slack buffer is a dual–port FIFO which is written synchronously to the sampling strobe generated by incoming data in myrinet_slave, and read synchronously to the SVI master transmit rate (which is send_clk). The data path into the slack buffer is nine bits wide (one byte plus a tail bit), and the data path out of the buffer is 32-bits – the width of a full Myrinet word. In addition to full and empty flags, the slack buffer outputs: tail_int – a flag indicating a word containing a tail is being read, and orun2_int & orun1_int – flags which indicate how many bytes past the tail byte (the byte containing the EXORed CRC) have been read out. (These flags are consistent with those provided by the LANai chip.) The slack buffer also has a word_align input, which when asserted forces the slack buffer to word align the next incoming byte, thereby word–aligning the first byte of every new message.

The depth of the slack buffer, as well as the location of the high–water and low–water marks is set via the constants ’slack_buffer_depth’, ’stop_slack’, and ’go_slack’ respectively in myrinet_iface_types_pkg.vhd. These constants are currently set to 64, 4, and 4 respectively.

unpack_fifo:

The unpack FIFO is a dual–port FIFO which is written synchronously to the SVI slave clock, and read synchronously to the Myrinet Master send_clk. The data path into the FIFO is 32–bits, and the data path out is 8–bits, thus allowing full Myrinet words to be “unpacked” into one–byte flits for transmission on the Myrinet. The depth of the unpack FIFO is set via the constant ’unpack_fifo_byte_depth’ in myrinet_iface_types_pkg.vhd; this value is currently set to 12 (12 bytes or three myrinet words).

myrinet_master:

The myrinet_master provides the direct interface to the Myrinet output port. The Myrinet master encodes the Myrinet output data signals into an NRZ format after retrieving data from the unpack FIFO and discarding any pad bytes, calculates the CRC–8 value for the message, appends it to the end of the data stream, and terminates the message with a GAP character.

The myrinet_master block is sequenced through Myrinet master transaction operations by a state machine with the following states and functions:

State	Functions
idle	Waiting to source a Myrinet message
xmit_route	Transmit Myrinet route flits
xmit_pkt_type	Transmit Myrinet packet type flits
xmit_data	Transmit Myrinet data flits
xmit_tail	Transmit Myrinet CRC & GAP flits

svi_myrinet:

The svi_myrinet block is responsible for providing a direct interface to the SVI; it receives data from the Myrinet half of the ENI, formats it for transmission on the SVI, provides handshaking with the connecting SVI port, and receives data on the SVI and prepares it for the Myrinet interface. The svi_myrinet incorporates two blocks: an SVI master and an SVI slave.

svi_master_myrinet:

The svi_master_myrinet provides the direct interface to the SVI output port. The SVI master reads data from the slack buffer, packs it into 32–bit SVI words, and pads route and last data words as necessary to provide proper word alignment. Since the slack buffer is filled one byte at a time as Myrinet flits are received, and there may be any number of route flits, data words in the slack buffer are generally not word aligned. The Myrinet SVI master contains a packing buffer which is a two–word deep register from which any contiguous four–bytes can be read. Routing words are read directly out of the packing register (after being obtained from the slack buffer) until a packet type word is encountered, at which time the alignment of packing buffer reads is shifted to coincide with the alignment of the packet type word (and subsequent data words). A separate shadow packing register keeps track of the tail byte (EXORed CRC) position in the packing register; when the SVI master detects that it is transmitting the tail byte in the SVI word, it appends any pad bytes necarry to fill the word, and transmits the pad count as the last SVI word.

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

State	Functions
idle	Waiting to source an SVI message
transfer_cmd	Transmit SVI command word
transfer_route	Transmit Myrinet route word on SVI
transfer_pkt	Transmit Myrinet data on SVI

svi_slave_myrinet:

The svi_slave_myrinet provides the direct interface to the SVI input port. The SVI slave receives data from the SVI input port, and writes it into the unpack FIFO in the Myrinet master block. The operation of the Myrinet SVI slave is quite simple, as it has no packing or alignment functions to contend with.

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

State	Functions
idle	Waiting to sink an SVI message
rcv_cmd	Receive SVI command word
rcv_pkt	Receive SVI data word

Myrinet ENI Testbench

All referenced files are currently located in /proj/rassp/gbuchana/rni/myrinet_svi.

The following files comprise the Myrinet ENI testbench files:

Run script	Stimulus/Compare Files	Description
run_myri_tb_noswitch	myri_svi_*_noswitch.txt	write operation to a destination node

run_myri_tb_switch	myri_svi_*_switch.txt	write operation to a switch
run_all	all above	all tests, run on command line

* -- stim/comp

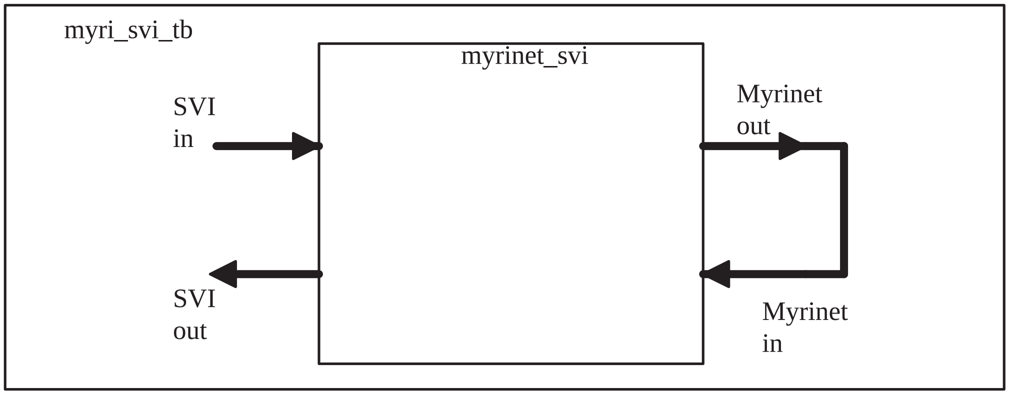

FIG. 2 – Block diagram – VHDL Testbench for Myrinet ENI

The Myrinet ENI testbench, Figure 2, is composed of a single Myrinet ENI with its Myrinet IN and OUT ports connected. Tests are run by stimulating the SVI IN port and observing the SVI OUT port. If the ENI is modeled properly, write data presented to the SVI input will appear on the SVI output. There are two testbench tests: run_myri_tb_noswitch tests the ENI under the condition in which the ENI Myrinet port is connected directly to a destination node (no intervening crossbar switch) and run_myri_tb_switch tests the ENI under the condition in which the ENI Myrinet port is connected to a switch. Both tests include multiple individual message transactions under a variety of SVI input and output conditions, and each test includes a message with (. . .no_switch) or without (. . .switch) a header route byte, in order to test the ENI's ability to drop a message for which it is not intended. As soon as all the expected results are received from the SVI output port, a message to the 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.