LABORATORY FOR COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Arctic User's Manual

Computation Structures Group Memo 353
February 22, 1994

George A. Boughton et al.

This report describes research done at the Laboratory for Computer Science of the Massachusetts Institute of Technology. Funding for the Laboratory is provided in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-92-J-1310.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

DRAFT Arctic User's Manual

Andy Boughton Greg Papadopoulos Bob Greiner Satoshi Asari
Steve Chamberlin Jack Costanza Richard Davis Thomas Deng
Tom Durgavich Doug Faust Eric Heit Tom Klemas Jimmy Kwon
Wing Chi Leung Elizabeth Ogston Gowri Rao Ralph Tiberio

January 14, 1994

NOTE: EVERYTHING DESCRIBED BELOW IS SUBJECT TO CHANGE.

Contents

1 Introduction

Arctic (A Routing Chip That Is Cool) is a four input four output packet switch on a chip. It will be used to construct the interconnection network for *T.

Arctic will support two levels of priority. It will support normal packets and high priority packets.

Most of Arctic will operate off a single 50MHz local clock. However each of the four input sections will operate off a separate 50MHz clock. Each of these input clocks may have a different phase but all must have a frequency which is extremely close to the local clock.

Arctic will handle packets that range from 128 bits to 768 bits. The packet format is described in more detail below.

Links will always be wired in pairs. If input i of a given Arctic chip is connected to output j of another chip then output i of the given chip will be connected to input j of the other chip.

2 Basic Structure

All buffering in Arctic is associated with the inputs as is shown in Figure 1. A 768 bit buffer will be allocated for each packet. Obviously in most cases the buffer will not be fully utilized (because the packet will likely be less than 768 bits and because the head of the packet will likely be forwarded before the tail is received).

The data crossbar in Arctic has an input connected to each buffer of each input section and an output connected to each output section. Thus it can simultaneously transfer packets from any four buffers to the four output sections. (For example, the crossbar can simultaneously transfer 3 packets from input section 0 and 1 packet from input section 1.)

It should be noted that due to the way in which buffering and scheduling are done in Arctic, it is possible for Arctic to reorder a sequence of packets flowing from one of its inputs to one of its outputs (however in most cases this will not happen).

3 Link Specification

3.1 Overview

The data links that connect to Arctic's inputs and outputs have the structure shown in Figure 2. GTL is used for all signals.

The data link uses a 16 bit wide data path. Each bit is transmitted using a single ended GTL driver. Data is sent at 100Mbits/sec on each signal line.

3.2 Data and Frame Timing

A clock (50MHz) is transmitted with the data. Two phases of the clock (180 degrees out of phase) are transmitted. The clock is used by the link receiver to safely latch link data.

Only one edge, the falling edge, of each clock phase is used. However the two link clock phases must meet certain requirements (at the transmitter pad). The falling edge of one phase should

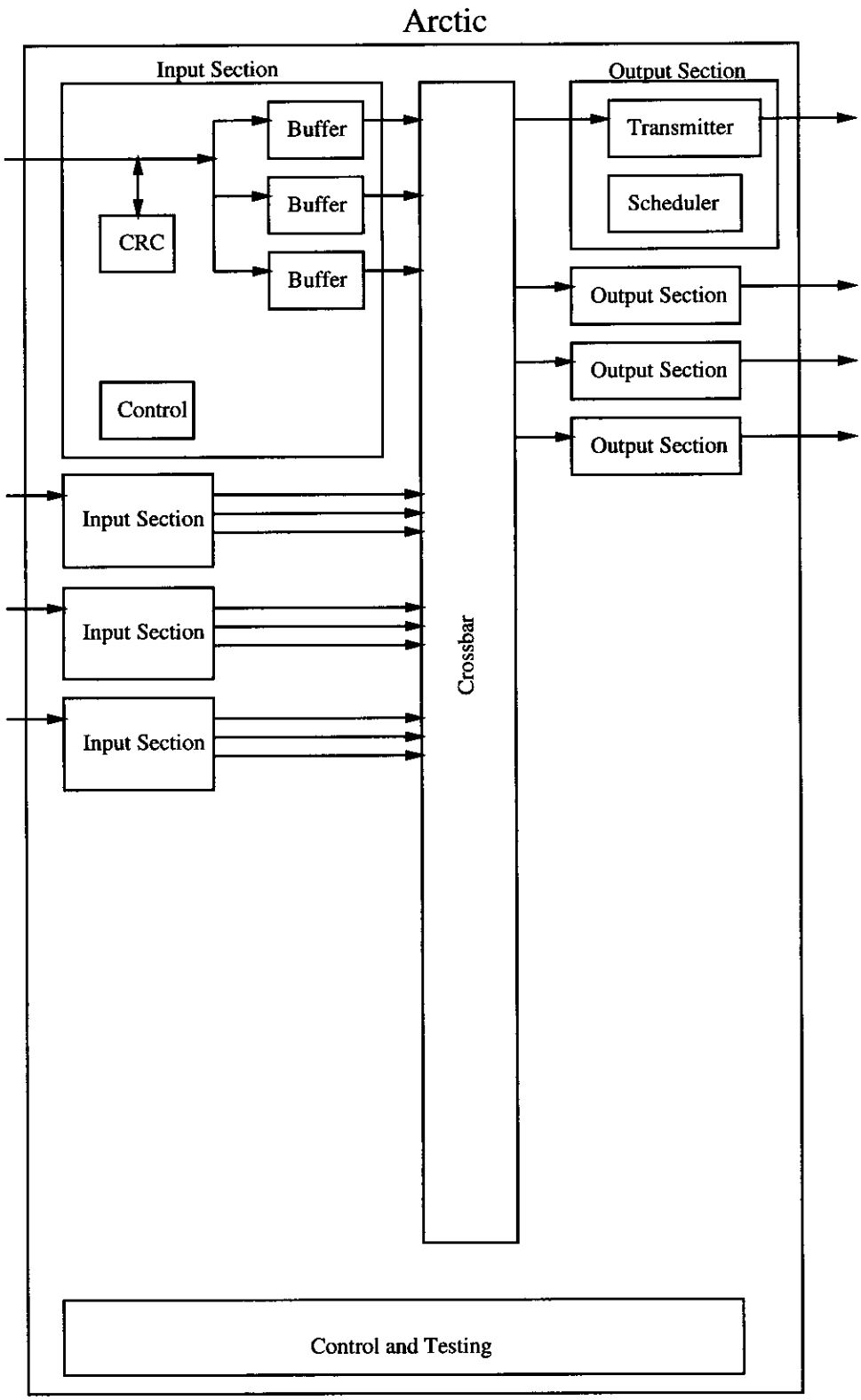


Figure 1: Arctic Block Diagram

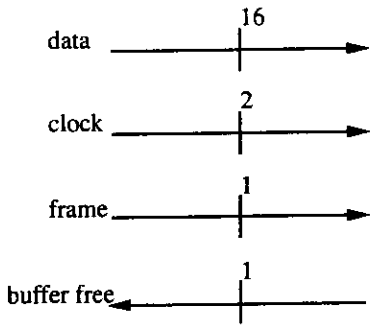


Figure 2: Network Link

occur 10ns +/- 1.5ns after the falling edge of the other phase. The rising edge of each phase should occur 10ns +/- 3ns after the falling edge of that phase.

A (16 bit) half word of data is sent for each falling edge of each of the two clocks. The timing constraints of data relative to the clocks are described below.

A frame signal is sent with the data. The frame signal is *Manchester* encoded. Thus one bit of frame information is sent for each two (16 bit) half words of data. The frame signal is logically asserted for all (16 bit) half words of the packet except for the last two half words. The frame signal is logically unasserted for the last two half words of the packet and all idle patterns.

The frame signal is *Manchester* encoded. A logical assertion is transmitted as roughly 10ns of 1 followed by roughly 10ns of 0. (The 1 is sent following the falling transition of the negative clock phase and the 0 is sent following the falling transition of the positive clock.) A logical unassertion is transmitted as roughly 10ns of 0 followed by roughly 10ns of 1. (The 0 is sent following the falling transition of the negative clock phase and the 1 is sent following the falling transition of the positive clock.) The detailed timing of the frame signal is given below.

Data (and the frame signal) transitions (at the transmitter pad) just after the falling edge of the appropriate clock phase. In particular, data should transition no sooner than 250ps after the falling edge and the data should be stable no later than 1.5ns after the falling edge.

The system interconnect should introduce no more that 500ps skew between any two signal lines (including the clocks).

Finally, it is the responsibility of the system designer to ensure that all signals meet certain quality constraints at the receive pads. In particular the clocks must remain ???mV above $V_{threshold}$ for at least ???ps and remain ???mV below $V_{threshold}$ for at least ???ps. If we say (for the purpose of this paragraph) that a clock transition occurs when a falling edge of either clock phase passes through $V_{threshold}$, then the data lines and the frame line (at the receiver pad) must be valid (???mV above (or below) $V_{threshold}$) no later than ???ns before each clock transition and must remain valid until ???ns before the next clock transition.

3.3 Idle Pattern

Idle patterns will be sent on a link whenever packets are not being sent. The idle pattern is a 32 bit (two 16 bit half word) patterns. The legal idle pattern is (in hex) AAAA followed by 5555.

3.4 Buffer Free

The buffer free signal is a flow control signal that goes in the reverse direction of the data.

Each Arctic output section is responsible for keeping a running count of the number of free buffers in the input section of the Arctic to which the output is connected. The output section appropriately decrements the count each time a packet is sent. The buffer free signal is used by the input section to inform the connected output section that a buffer has been freed and that the output section can appropriately increment its counter.

The definition of the buffer free signalling depends on the buffering scheme. The Arctic buffering scheme allocates a 768 bit buffer for each packet. The input section has some number of 768 bit buffers. The output section is given this number (for the input section to which it is connected) at system initialization time as part of its configuration information. The output section initializes its counter with this number. The output section appropriately decrements the count each time a packet is sent. (A 5 bit counter is used.)

A *10* buffer free signal is sent from the receiving input section to the transmitting output section every time a new buffer is made available for use. The output section increments its counter by one for each *10* buffer free signal.

If the output section thinks that only one buffer remains, then it is only able to transmit a high priority packet. Normal packets are blocked. This ensures that normal packets can not deadlock the flow of high priority packets.

A *10* buffer free signal is sent as a *Manchester* encoded one followed by a *Manchester* encoded zero. Thus a *10* buffer free signal is sent as a 1001 pattern. The buffer free signal is timed to the link clocks from the associated reverse direction link. Each transition (or lack of transition) of the 1001 pattern corresponds to a falling edge of one of the reverse link clocks. The pattern begins following a falling edge of the negative clock phase with a transition to 1. The next transition (to 0) occurs following the falling edge of the positive clock phase. The final transition (to 1) follows the next falling edge of the positive clock phase.

The idle buffer free signal is a *Manchester* encoded 0. Thus an idle is sent as a 01 pattern. The pattern begins following a falling edge of the negative clock phase with a transition to 0. The next transition (to 1) occurs following the falling edge of the positive clock phase.

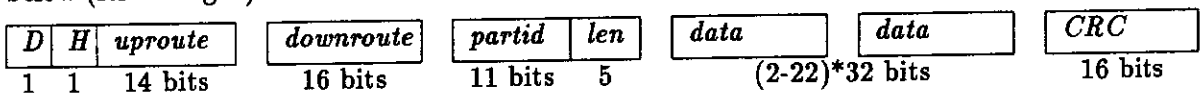
The timing constraints for the buffer free signal relative to the reverse link clocks are the same as those for the reverse link data lines and the reverse link frame line (discussed above under data and frame timing). In particular, the buffer free signal transitions (at the transmitter pad) just after the falling edge of the appropriate clock phase. In particular, the buffer free line should transition no sooner than 250ps after the falling edge and the buffer free signal should be stable no later than 1.5ns after the falling edge.

4 System Clock

Arctic requires a (roughly) 50MHz system clock. Arctic expects the system clock to meet the following criteria (at Arctic's input pads). The period of the clock should be at least 20ns. The time between any two edges of the clock should be ((the period) / 2) +/- 1ns. Terminated transmission lines should be used to deliver the system clock to Arctic. All clocks used in an Arctic network should have the same frequency within ??%.

5 Packet Format

The format for network packets is shown below. The 16 bit half words are sent in the order given below (left to right). Each half word is shown from MSB to LSB (left to right).



D is the double size packet bit. It will be 1 if and only if the packet is greater than 384 bits long.

H is the high priority bit. It will be 1 for high priority packets and 0 for regular packets.

uproute is a field typically used to specify the up route of a packet in a fat tree network.

downroute is a field typically used to specify the down route of a packet in a fat tree network.

partid is the partition id.

len is the length of the packet. This number times 32 is the total number of bits in the packet.

data is the payload of the packet and it's length (in bits) is a multiple of 32 in the range from 64 to 704.

CRC is the 16 bit CRC computed on all the previous bits of the packet using CCITT CRC-16.

6 Routing

Arctic uses a routing scheme that we are calling the *two-comparator scheme* to determine the destination output port for each packet. The scheme basically has the form of

$$if\ B_1\ then\ R_1\ else\ if\ B_2\ then\ R_2\ else\ R_3$$

B_1 and B_2 are expressions, each of which evaluate to a boolean value. R_1, R_2, and R_3 are expressions, each of which evaluate to a 3 bit value.

B_1 and B_2 are two step operations. A 16 bit mask and a 16 bit reference value are specified in the configuration information for each of B_1 and B_2. In the first step the *downroute* field of the packet is bitwise anded with the mask. In the second step the result of the first step is compared to the reference value. The final result is true if and only if identical values are compared. Thus,

$$B_1 = (DOWNROUTE\ and\ RULE1\_MASK)\ equal?\ RULE1\_REF$$

$$B_2 = (DOWNROUTE\ and\ RULE2\_MASK)\ equal?\ RULE2\_REF$$

Where RULE1_MASK, RULE1_REF, RULE2_MASK, and RULE2_REF are configuration registers as discussed later in the section on configuration registers.

R_1, R_2, and R_3 are fairly simple bit selections. Three 5 bit fields are specified in the configuration information for each of R_1, R_2, and R_3. Each of these fields is used to select one bit for the final result. Each bit is selected from 32 possibilities. These are a constant 1, a constant 0, the 14 *uproute* bits, and the 16 *downroute* bits as shown below.

31	30	29 16	15 0
1	*0*	*uproute(13:0)*	*downroute(15:0)*

Thus if for the purpose of this paragraph we call the 32 bit field above ADR_BITS then

$$R_1 = (ADR\_BITS(SEL1\_1) * 2) + ADR\_BITS(SEL1\_0)$$

$$R_2 = (ADR\_BITS(SEL2\_1) * 2) + ADR\_BITS(SEL2\_0)$$

$$R_3 = (ADR\_BITS(SEL3\_1) * 2) + ADR\_BITS(SEL3\_0)$$

Where SEL1_1, SEL1_0, SEL2_1, SEL2_0, SEL3_1, and SEL3_0 are configuration registers as discussed later in the section on configuration registers.

It should be noted that there is separate configuration data for each of Arctic's input ports.

This routing scheme is of course general enough to handle a fat tree structure such as the one shown in Figure 3. We assume source based routing. The *uproute* of each packet is randomly generated. The *downroute* is the physical address of the destination endpoint. In this simple structure, only one comparison is needed to determine the routing of a packet. An Arctic input is configured to have RULE1_MASK extract no bits, bit 2, or bits 2 and 1 of the *downroute* (based on whether the Arctic is at the root, middle, or leaf level). RULE1_REF is set to correspond to the common preface of the addresses of all endpoints in the tree below this router. In other words the comparison determines whether the packet is destined for an endpoint in the tree below this Arctic and thus should be routed down. SEL1_1 and SEL1_0 are configured to handle the case of a packet that should go down and they select bit 2, bit 1, or bit 0 of the *downroute* (based on whether the Arctic is at the root, middle, or leaf level). RULE2_MASK and RULE2_REF are set such that B_2 is always true. SEL2_1 and SEL2_0 are set to handle sending a packet up and they select bit 1 or bit 0 of the *uproute* (based on whether the Arctic is at the middle, or leaf level).

7 Test and Control

7.1 Modes of Operation

Arctic has four modes of operation. These are *normal*, *configuration*, *low-level-test*, and *manufacturing-test*. *Manufacturing-test* mode will only be used when Arctic is in the tester at the fabrication plant. It will be entered only when the a special *manufacturing-test* mode pin is asserted. The other modes will be controlled through the maintenance interface.

Normal mode is used for regular operation. Arctic routes packets in *normal* mode (and no other mode). While Arctic is in *normal* mode only the control registers, statistics counters, error counters, and buffer free counters are accessible through the maintenance interface.

Configuration mode is used to load configuration information into Arctic. There is a setup procedure required to enter *configuration* mode. All of the input sections and output sections should be disabled (using the *port/statistics* control register discussed below) for 480ns + ???? before *configuration* mode is entered.

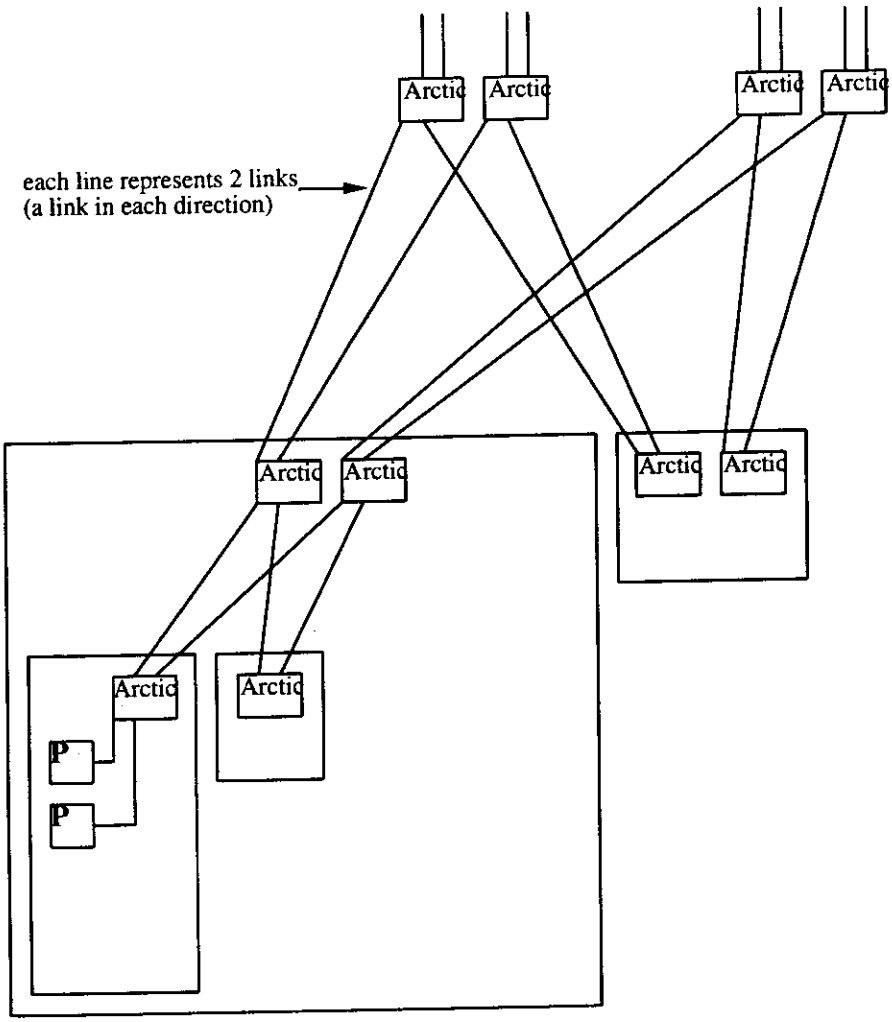


Figure 3: 8 Endpoint Fat Tree Network

Low-level-test mode is used to allow the maintenance interface to access the manufacturing test rings. In *low-level-test* mode, all of Arctic is placed into a single clock domain. *Low-level-test* mode can be used to scan data into the manufacturing test rings, cycle the system clock, and scan the results out of the manufacturing test rings.

Manufacturing-test mode is used by the tester at the fabrication plant. It is entered only when a special *manufacturing-test* mode pin is asserted. When Arctic is placed in *manufacturing-test* mode, certain Arctic pins become control and data pins for the manufacturing test rings. Thus, the maintenance interface is not used in *manufacturing-test* mode. In *manufacturing-test* mode, all of Arctic is placed into a single clock domain. In *manufacturing-test* mode the testor can scan data into the manufacturing test rings, cycle the system clock, and scan the results out of the manufacturing test rings.

7.2 Maintenance Interface

Arctic will support a concise set of access operations through its maintenance interface. This set of operations will make it possible to read and write various maintenance registers in Arctic. The maintenance interface is physically implemented as an IEEE Std. 1149.1 (from the JTAG group) compliant interface. The first description below sketches the general form of the access operations. Next is a description of how the access operations are issued through the physical interface.

7.2.1 Access Operations

Most of Arctic's maintenance registers are accessed using read and write operations that are split phase operations. First the read or write operation will be issued. Some time later Arctic will complete the operation. The *get_status* operation can be used (in most cases) to detect if the read or write has completed. In the case of a completed read, *get_status* will also return the requested data. A new read or write operation should not be initiated if there is a outstanding uncompleted read or write.

Reads and writes on some of Arctic's maintenance registers will complete quickly and each will complete before another access operation can be issued. If a read is issued on such a register then a subsequent *get_status* operation will return the requested data (and no status bits since it is known that the read must have completed). No *get_status* operation should be issued after a write on such a register (since it is known that the write must have completed). As is discussed below these registers are the *port/statistics* control register, the *system-reset* control register, the error counters and the buffer free counters.

Three special access operations will be used in *low-level-test* mode to access the manufacturing test rings.

The seven access operations are listed below.

read_reg(reg_addr) initiates a non-destructive read of the register addressed by *reg_addr.*

write_reg(reg_addr, datum) initiates a write of *datum* into the register addressed by *reg_addr.*

get_status (in most cases) returns the status of the previous read or write. Either *in progress* or *completed.* It also returns the value of the currently addressed (addressed by the last read or write) register. Obviously this value is valid only if the status is *completed.* In the case that one of four special registers (the *port/statistics* control register, the *system-reset* control register, the error

counters or the buffer free counters) are currently addressed, then *get_status* should only be used following a read and it will return only the value of the currently addressed register (and no status bits).

chng_mode(mode) causes the mode of Arctic to be changed to *mode*. (Of course the three possible modes are *normal*, *configuration*, and *low-level-test*. As mentioned above *manufacturing-test* mode is only entered by assertion of a special pin. Also as mentioned above, all of the input sections and output sections should be disabled (using the *port/statistics* control register discussed below in section 7.3.1) for 480ns + ???? before *configuration* mode is entered.) Once a *chng_mode(mode)* command is issued, the mode of Arctic is guaranteed to change within ??ns.

set_up_llm_scan is used in *low-level-test* mode to set up a scan of the manufacturing test rings. Please see the discussion below on the manufacturing test ring register.

llm_scan(datum) is used after a *set_up_llm_scan* operation to scan data into (and out of) the manufacturing test rings. Please see the discussion below on the manufacturing test ring register.

end_llm_scan(datum) is used to finish a scan of the manufacturing test rings. Please see the discussion below on the manufacturing test ring register.

7.2.2 Interface Implementation

The maintenance interface is physically implemented as an IEEE Std. 1149.1 (from the JTAG group) compliant interface. To make the discussion below more concise, the phrase *JTAG interface* will be used to mean an IEEE Std. 1149.1 compliant interface.

Unfortunately, using the JTAG interface to issue the access operations requires using another level of protocol. In other words, it is necessary to use low level JTAG transactions to transfer the access operation commands through the JTAG interface.

A conceptual model of the JTAG interface is shown in Figure 4. Most of this structure is mandated by the standard. The portion labeled *Test Specific Registers* is specialized for the Arctic access operations. These registers include the mode register, the access operation address register, the access operation instruction register, the status and data registers, and the low level mode scan register.

A description of the interface standard (IEEE Std. 1149.1) can be found in *The Test Access Port and Boundary Scan Architecture* by Colin M. Maunder and Rodham E. Tulloss (published in 1990 as an IEEE Tutorial). This reference also describes the format of transactions on a compliant interface.

IT SHOULD BE NOTED that Arctic will be designed to accept a maximum test clock speed of 10MHz.

The state diagram of the Test Access Port (TAP) Controller is shown in Figure 5.

Below is a description of each Arctic access operation in terms of access macros. The access macros are then defined in terms of JTAG transaction macros. The JTAG transaction macros are then defined in terms of TAP states.

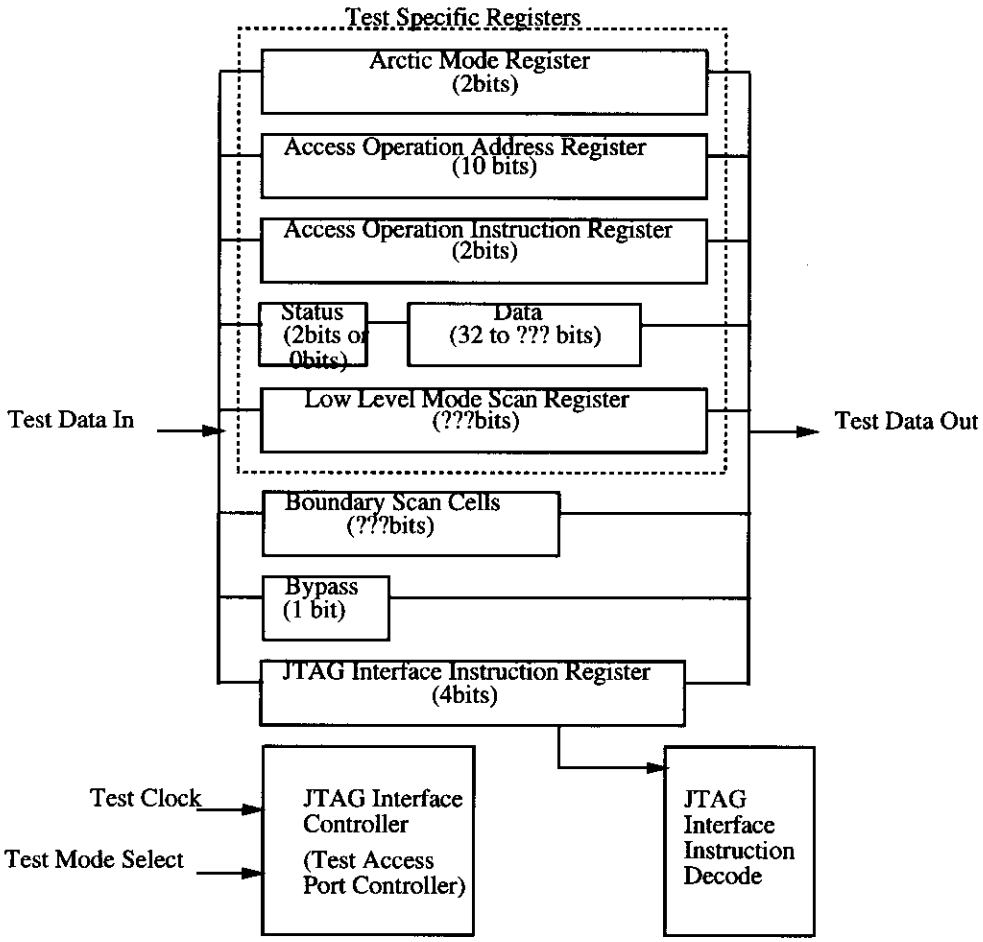


Figure 4: IEEE Std. 1149.1 compliant interface

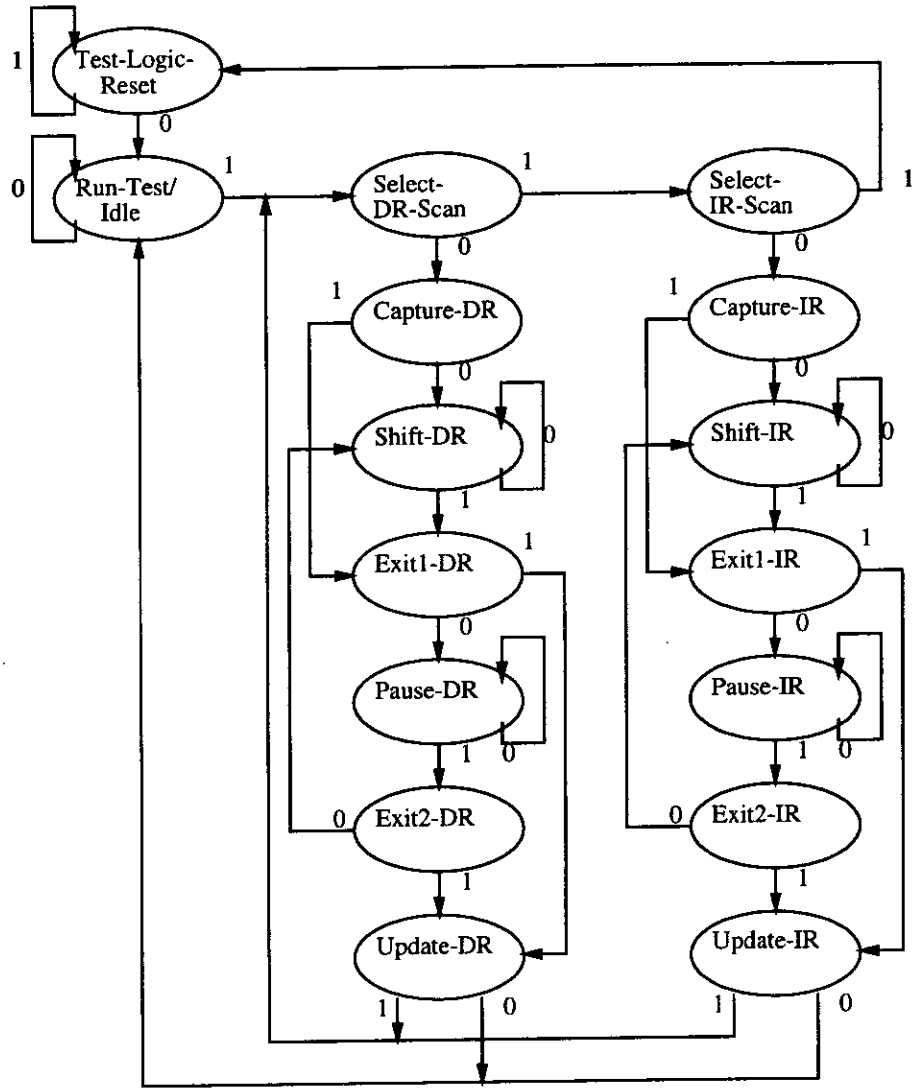


Figure 5: Test Access Port Controller State Diagram

Description of Arctic Access Operations

Access Operation	Definition
read_reg(*reg_addr*)	1) write_AOAreg(*reg_addr*) 2) write_AOIreg(read_reg)
write_reg(*reg_addr*, *datum*)	1) write_AOAreg(*reg_addr*) 2) write_AOS+AODreg(*filler*, *datum*) 3) write_AOIreg(write_reg)
get_status	1) read_AOS+AODreg
chng_mode(*mode*)	1) write_AOMreg(*mode*)
set_up_llm_scan	1) set_up_llm_scan
llm_scan(datum)	1) llm_scan(*datum*)
end_llm_scan(datum)	1) end_llm_scan(*datum*)

Note: *filler* in the table above should be the null string if the AOA has been loaded with the address of the *port/statistics* control register, the *system-reset* control register, the error counters or the buffer free counters. *Filler* should be 00 (binary) if the AOA has been loaded with the address of any other register.

Description of Access Macros

Macro	Definition
write_AOAreg(*reg_addr*)	1) load_JTAGIR(sel_AOAreg) 2) load_JTAGDR(*reg_addr*)
write_AOIreg(*AOInst*)	1) load_JTAGIR(sel_AOIreg) 2) load_JTAGDR(*AOInst*)
write_AOS+AODreg(*filler*, *datum*)	1) load_JTAGIR(sel_AOS+AODreg) 2) load_JTAGDR(*filler*, *datum*)
write_AOMreg(*mode*)	1) load_JTAGIR(sel_AOMreg) 2) load_JTAGDR(*mode*)
read_AOS+AODreg	1) load_JTAGIR(sel_AOS+AODreg) 2) load_JTAGDR
set_up_llm_scan	1) load_JTAGIR(sel_AOAreg) 2) load_JTAGDR(380hex) 3) load_JTAGIR(sel_AOS+AODreg)
llm_scan(*datum*)	1) load_JTAGDR2(*datum*)
end_llm_scan(*datum*)	1) load_JTAGDR_RTI(*datum*)

Description of JTAG Transaction Macros

Macro	Definition
load_JTAGIR(*JTAGInst*)	1) Test-Logic-Reset 2) Run-Test/Idle 3) Select-DR-Scan 4) Select-IR-Scan 5) Capture-IR 6) Shift-IR (*JTAGInst*) 7) Exit1-IR 8) Update-IR 9) Select-DR-Scan
load_JTAGDR(*JTAGdatum*)	1) Select-DR-Scan 2) Capture-DR 3) Shift-DR (*JTAGdatum*) 4) Exit1-DR 5) Update-DR 6) Select-DR-Scan 7) Select-IR-Scan 8) Test-Logic-Reset
load_JTAGDR2(*JTAGdatum*)	1) Select-DR-Scan 2) Capture-DR 3) Shift-DR (*JTAGdatum*) 4) Exit1-DR 5) Update-DR 6) Select-DR-Scan
load_JTAGDR_RTI(*JTAGdatum*)	1) Select-DR-Scan 2) Capture-DR 3) Shift-DR (*JTAGdatum*) 4) Exit1-DR 5) Update-DR 6) Run-Test/Idle 7) Select-DR-Scan 8) Select-IR-Scan 9) Test-Logic-Reset

Description of JTAG Transaction Macros (continued)

Macro	Definition
load_JTAGDR2_RTI(*JTAGdatum*)	1) Select-DR-Scan 2) Capture-DR 3) Shift-DR (*JTAGdatum*) 4) Exit1-DR 5) Update-DR 6) Run-Test/Idle 7) Select-DR-Scan

Access Register Sizes and Op Codes

register	size	code		
Mode	2	0	config	
		1	normal	
		2	low-level-test	
		3	UNDEFINED	
AOA	10	-		
AOI	2	0	write_reg	
		1	read_reg	
		2	UNDEFINED	
		3	UNDEFINED	
AOS	0 or 2	0	completed	Note: The AOS is really accessed as the most significant bits of AOS+AOD. The AOS is 0 bits (or in other words it is not used) if the AOA addresses the *port/statistics* control register, the *system-reset* control register, the error counters or the buffer free counters. Otherwise the AOS is 2 bits
		1	in_progress	Following a read: The read operation is still in progress and the data returned is garbage. Following a write: The write operation is still in progress; another get_status is nessesary.
		2	mode_error	Following a read or write: An attempt was made to access configuration data while in normal mode. The configuration data remains unchanged and the value returned is garbage.
		3	broadcast_read_error	Following a read: An attempt to read from a broadcast address (1E0-1E4) was made. The value returned is garbage.
AOD	32 to 1??	-		Note: The AOD is really accessed as the least significant bits of AOS+AOD.

JTAG Op Codes

JTAG instruction	code (in hex)
bypass	F
extest	0
<reserved>	1
intest	2
runbist	3
sel_Mreg	4
sel_AOAreg	5
sel_AOIreg	6
sel_AOS+AOD_reg	7

7.2.3 Boundary Scan

As part of its JTAG interface, Arctic provides a facility to do boundary scan. Since the JTAG standard provides for boundary scan, Arctic allows JTAG level access to its boundary scan. (In other words, boundary scan is implemented at the JTAG level and not at the higher access operation level.) Unfortunately a completely standard definition of boundary scan through a JTAG interface does not adequately handle Arctic. One set of problems is caused by the fact that most of the data lines of Arctic receive a new bit every half clock cycle. Another set of problems is caused by the fact that Arctic is designed to receive inputs from multiple clock domains. Yet another set of problems is caused by the fact that the control registers can only be loaded through the JTAG interface (and not through non JTAG interface pins).

To address these problems, we have designed Arctic with a augmented definition of boundary scan.

The boundary scan ring in Arctic not only contains a scan cell for each (non JTAG interface) signal pin of Arctic, but it also contains many additional scan cells. For example there are 32 scan cells for the data portion of each link (each input link and each output link). These cells correspond to the 32 bits of data that would be passed over the path in a given clock cycle. Thus for the purpose of boundary scan, the data portion of each link appears to be 32 bits wide with a bit transmitted every cycle. (This is of course in contrast to normal operation where only 16 pins are used but where two bits are passed over each pin per clock cycle.) Note that while 32 scan cells are associated with the data portion of each link, obviously only 16 of those cells are directly associated with pins. The cells directly associated with input pins are used by some boundary scan operations to capture data from the pins. The cells directly associated with output pins are used by some boundary scan operations to drive the output pins. The boundary scan operations are discussed below.

The boundary scan ring of Arctic also includes 34??? scan cells corresponding to the 34 bits of the *port/statistics* control register (described below). Values scanned into these scan cells are in effect scanned into the *port/statistics* control register. Thus for the purpose of boundary scan, the *port/statistics* control register is directly loadable and this feature will be used in the *intest* instruction described below.

A complete list of the bits in the boundary scan ring is given later in this manual.

JTAG interfaces normally support three JTAG instructions involving the boundary scan ring. These are *extest*, *sample/preload*, and *intest*. Arctic supports modified versions of *extest* and *intest* only.

(*Sample/preload* appears to be difficult to implement properly given the multiple clock domains of Arctic.)

An *extest* JTAG instruction causes Arctic to first capture data from its input pins. Next it scans in new data to be applied to its output pins and scans out the data captured from its input pins.

Thus, the way in which *extest* is executed over Arctic's JTAG interface is consistent with the JTAG standard, however the semantics of the instruction have been somewhat modified. For example, only those scan cells of the boundary scan ring that correspond to input pins capture useful data. It should also be noted that Arctic does not have boundary update registers and that during *extest* the output pins are driven directly from the boundary scan cells. Thus the values on the output pins will toggle as the boundary scan register is scanned. (Obviously this configuration was used to save gates.)

An *intest* JTAG instruction forces all of Arctic to be placed in a single clock domain (run off the test clock). Arctic first captures the data output by its core circuitry. Next it scans in new data for each input and the *port/statistics* control register, and it scans out the data captured from the core outputs. The input data is scanned directly into the first stage of core registers. (This is in contrast to the conventional definition of *intest* where dedicated boundary scan and update registers are used and those registers apply values to the inputs of the first stage of core registers.) The output data is scanned directly from the last stage of core registers. (This is in contrast to the conventional definition of *intest* where dedicated boundary scan and update registers are used and those registers are feed by the last stage of core registers. Thus Arctic appears to have two less stages of *intest* latency (than would be expected using a conventional definition of *intest*).) Finally the system clock (which is for the purpose of *intest* being sent to all of Arctic) is cycled. Note that the new value scanned into the *port/statistics* control register is in effect for the system clock cycle. As defined by the standard, proper toggling of the JTAG test mode select line can be used to cause multiple system clock cycles to be issued.

Again the way in which *intest* is executed over Arctic's JTAG interface is consistent with the JTAG standard, however the semantics of the instruction have been somewhat modified. As noted above the most important deviation is the fact that input data is scanned directly into the first stage of core registers and that output data is scanned directly out of the last stage of core registers. So in fact the boundary scan cells corresponding to inputs are implemented using the first stage of core registers and the boundary scan cells corresponding to outputs are implemented using the last stage of core registers. (Obviously this was done to save gates.)

It should be noted that special boundary scan cells are used to handle the various clock signals in Arctic. The scan cell for an incoming clock signal is only capable of sampling the incoming clock and not of affecting the value of the clock signal sent to the core circuitry of Arctic. The scan cell for an outgoing clock signal only drives the output pin during an *extest* instruction. Thus the boundary cells for clock signals are primarily used for *extest* and do not provide useful information for *intest*. It should be noted that during the scan portion of an *extest* instruction the output clock pins will toggle as values are scanned through the boundary scan cells.

It should also be noted that the first data capture in an *extest* or *intest* sequence will not capture valid data. This is due to the procedure used to perform the initial switch to a single clock domain. However subsequent data captures will work properly.

Listed below are the definitions of *intest* and *extest* in terms of the JTAG transaction macros.

Description of Boundary Scan Operations

Operation	Definition
intest	1) load_JTAGIR(2) 2) for i = 0 to (n-2), load_DR2_RTI(*testpattern*(i)) 3) load_DR_RTI(*testpattern*(n-1))
extest	1) load_JTAGIR(0) 2) for i = 0 to (n-2), load_DR2(*testpattern*(i)) 3) load_DR(*testpattern*(n-1))

7.3 Accessible Registers

The Arctic maintenance registers (which are read and written using the Arctic access operations) are split into six groups. These are the control registers, the statistic counters, the error counters, the buffer free counters, the configuration registers, and the manufacturing test ring register.

The visibility of a group of registers depends on the Arctic mode. In *normal* mode only the control registers, the statistic counters, the error counters, and the buffer free counters are visible. In *configuration* mode all of those registers plus the configuration registers are visible. In *low-level-test* mode only the manufacturing test ring register is visible. In *manufacturing-test* mode only the manufacturing test rings are visible to the tester and the tester has direct access to those rings using designated pins. The maintenance interface is not be used in *manufacturing-test* mode.

The *reg_addr* argument to the Arctic access operations will have two components. These are a set number and a register number within the set. Each group of registers (such as the control registers, statistics counters, etc.) will be divided into some number of sets.

As is shown in the following table, some of the register groups are read only.

Supported Operations on Each Register Group

Group	Supported Operations
control registers	read and write
statistic counters	read only
error counters	read only
buffer free counters	read only
configuration registers	read and write
manufacturing test ring register	scan

Below we give a description of each group of registers. The register addresses and bit positions for each register are given later in this manual.

7.3.1 Control Registers

There are two control registers; the *port/statistics* control register and the *system-reset* control register.

Note: As is discussed above (in the section on access operations), reads and writes to the two control registers will complete quickly and each will complete before another access operation can be issued. If a read is issued on one of these registers then a subsequent *get_status* operation will return the requested data (and no status since it is known that the read must have completed). No *get_status* operation should be issued after a write on one of these registers (since it is known that the write must have completed).

Port/Statistics Control Register

The *port/statistics* control register contains certain control bits for each input and output section. It also contains certain control bits for the statistics section.

Input Section Control Bits (per input section)

Name	Size	Description
DISABLE_PORT	1	Input disabled

When an input section is disabled it first completes the reception of any packet currently being received. Then it stops monitoring all of its incoming link signal lines. It stops detecting all errors. However it still attempts to process any internally buffered packets. It continues to operate the buffer free line normally.

When an input section is enabled it first clears all of its packet buffers (without sending any buffer free signals). Then it begins receiving packets normally.

IT SHOULD BE NOTED that an input port enable will take effect by four input clock cycles after the completion of the write of the *port/statistics* control register. So obviously new packets should not be presented to an input port until five input clock cycles after the completion of the write of the *port/statistics* control register. (In most cases this should be 100ns and should not present any real constraint since it is likely that enabling another chip to send to Arctic will take much more than 100ns).

Output Section Control Bits (per output section)

Name	Size	Description
DISABLE_PORT	1	Stop sending packets, ignore buffer free line
OUTPUT_BLOCK	1	Like DISABLE_PORT but watch buffer free line
FLUSH	1	Send data without regard to buffer free count

When an output section is disabled it first completes the transmission of any packet currently being sent. Then it stops sending packets and it stops monitoring the incoming buffer free line. It sends out clock, idle on the data lines, and unasserts on the frame line. Any packet that is received by Arctic after the output section has been disabled and is addressed for the disabled output port will be sent to the MISS_PORT (see section below on configuration registers). NOTE: packets that are buffered in Arctic at the time the port is disabled (and are not being transmitted through the output at the time) are NOT sent out the MISS_PORT. These packets are trapped inside Arctic until the output is enabled and at that point they are sent out the (original) output port (and not the MISS_PORT).

When an output section is enabled, it assumes that all input buffers of the input section (attached by the link to this output) are available for packets (it initializes its buffer free count) and it starts normal transmission of packets.

When an output is disabled, the OUTPUT_BLOCK and FLUSH control bits are ignored.

When OUTPUT_BLOCK is asserted, the output section behaves much as if it were disabled but it continues to monitor the buffer free line and appropriately update its buffer free count. Later packets addressed to a blocked output section are buffered (and not sent to MISS_PORT). When OUTPUT_BLOCK is removed, the output section assumes that its buffer free count is correct and returns to normal operation using that buffer free count. Any buffered packets will be sent out.

When OUTPUT_BLOCK is asserted, the FLUSH control bit is ignored.

When FLUSH is asserted, the output section ignores the buffer free line and the buffer free count and sends any available packet. When FLUSH is removed, the output section assumes that all input buffers of the input section (attached by the link to this output) are available for packets and it returns to normal operation.

Statistics Control Bits

Name	Size	Description
NORM	1	Count normal packets for the UP, DOWN, and PACKETS statistics
PRI	1	Count priority packets for the UP, DOWN, and PACKETS statistics

NORM and PRI are used to determine what type of packets are counted for certain statistics. Please see the discussion below on the Statistics Counters.

System-Reset Control Register

A write of the *system-reset* control register causes a system reset of Arctic. A system reset initializes most of the FSM's of Arctic. It clears the error counters. It initializes the *port/statistics* control register to disable all the input and output ports. A system reset does not reset the JTAG controller. It does not reset the Arctic mode register. It does not reset the configuration registers. It does not clear the statistics counters.

It should be noted that internally Arctic switches to a single clock domain (run off the JTAG interface) during reset and then switches back to multiple clock domain operation (if so specified by the Arctic mode register) at the completion of reset. As a result, the link clocks out of Arctic switch frequency (potentially 50MHz to 10MHz and back to 50MHz) during reset.

7.3.2 Statistics Counters

All of the statistics are counted at the output sections. As is described below, six statistics are kept for each output section. As is described below, four of the statistics are obtained by actual count and two of the statistics are obtained by periodic sampling.

For three (PACKETS, UP, and DOWN) of the four counted statistics, Arctic can be configured (using the NORM and PRI bits of the *port/statistics* control register) to count normal packets, high priority packets, or both. Note that there is just one setting for the whole chip, and this single setting determines what type of packet is counted for all of the three counted statistics for all of the four outputs.

NOTE: In the following descriptions we use the terms *above* and *below*. A chip that is *below* the current chip in a fat tree is a chip that is closer to the leaves. A chip that is *above* the current chip in a fat tree is a chip that is farther from the leaves.

Statistics Counters (per output section)

Name	Description
PACKETS	This is a count of the number of designated (by the NORM and PRI control bits) packets transmitted.
PRIORITY	This is a count of the number of high priority packets transmitted.
UP	This is only used if this output section is connected to a chip *above* the current chip. It is a count of the number of designated packets that were received from a chip *below* the current chip.
DOWN	This is only used if this output section is connected to a chip *below* the current chip. It is a count of the number of designated packets that were received from a chip *above* the current chip.
IDLE	This is a sampled statistic that is proportional to the number of cycles that this output section was idle.
WAIT	This is a sampled statistic that is proportional to the number of cycles that this output section was blocked from sending an available packet.

PACKETS, PRIORITY, UP, and DOWN are exact counts of the number of packets in each category.

However, IDLE and WAIT are sampled statistics. The state of the output section is checked every 192 cycles (192 20ns cycles). If the output is in the idle state then the IDLE statistic is incremented. If the output is in the wait state then the WAIT statistic is incremented.

The statistics counters are implemented using a two stage system. There is a 6 bit counter for each statistic (except IDLE and WAIT) in each output section. There is also a centralized RAM with a 36 bit element for each statistic of each output section. A centralized controller reads each of the 6 bit counters in sequence. The controller reads the value of the 6 bit counter, adds it to the value of the corresponding 36 bit element, stores the result back into the centralized RAM, and clears the 6 bit counter. Each 6 bit counter is read once every 192 cycles (192 20ns cycles).

The Arctic access operations actually read the elements in the centralized RAM. Obviously the statistics read in this manner may be 192 cycles old. However given the slow speed of the maintenance interface, this should not be a major concern.

Each statistic will have two access operation register addresses. Use of the first of these addresses will cause a non destructive read of the statistic. Use of the second address will cause a read and a clear of the statistic (clearing its element in the centralized RAM and its 6 bit counter).

There is also a special access operation register which when read will cause a blanket reset of all the statistics. Note that the completion of the blanket reset can be determined using a *get_status* access operation.

7.3.3 Error Counters

Two bits are used to store each error count (except ICLK_ERR). Thus the count is only able to indicate 0, 1, 2, or more than 2 errors.

A single access operation register is used to read all of the error counters at once. This register has two access operation register addresses. Use of the first of these addresses causes a non destructive read of all the error counters. Use of the second address causes a read and a clear of all the error counters.

Note 1: As is discussed above (in the section on access operations), reads of the error counters will complete quickly and each will complete before another access operation can be issued. If a read is issued on the error counters then a subsequent *get_status* operation will return the requested data (and no status since it is known that the read must have completed).

Note 2: The circuitry for counting errors in Arctic assumes that two errors of the same type will be separated by at least 80ns. If two errors of the same type occur within a shorter time interval then Arctic may only increment the appropriate counter once. (However Arctic will increment the appropriate counter *at least* once.)

Note 3: Certain errors on the inputs to Arctic such as frame encoding errors can cause Arctic to generate bogus packets and pass these packets through its outputs. (For example, if a frame encoding error occurs while a packet is being received then Arctic may incorrectly guess the end of the packet. If a frame encoding error occurs during an idle pattern, Arctic may incorrectly assume that a packet has started.)

Note 4: A packet that causes an overflow error may cause a bogus packet to be sent on one of Arctic's outputs.

Input Section Error Counters (per input section)

Name	Description
CRC_ERR	CRC error
OVFL_ERR	Overflow error, packet received but no input buffer available
FR_ERR	Coding error on frame signal
LEN_ERR	Length error, length from packet doesn't match frame signal
IDLE_ERR	Idle pattern not found when no packet data
RT_ERR	Route error. Caused by either a packet from *above* that is addressed to an *above* output or by a packet that is addressed to a disabled output. The packet is sent to the MISS_PORT.
ICLK_ERR	Input clock missing error. This will be set if the input clock is completely missing for at least 16 cycles. However it probably will not be set by a glitch on the input clock. Note that ICLK_ERR is only a one bit flag and not a count.

Output Section Error Counters (per output section)

Name	Description
BF_ERR	Manchester coding error on the buffer free signal

7.3.4 Buffer Free Counters

Five bits are used to give the buffer free count for each output section.

A single access operation register is used to read all of the buffer free counters at once.

Note: As is discussed above (in the section on access operations), reads of the buffer free counters will complete quickly and each will complete before another access operation can be issued. If a read is issued on the buffer free counters then a subsequent *get_status* operation will return the requested data (and no status since it is known that the read must have completed).

Output Section Buffer Free Counters (per output section)

Name	Description
BF_CNT	buffer free count

7.3.5 Configuration Registers

Input Section Configuration Bits (per input section)

Name	Size	Description
CRC_CK_EN	1	CRC error checking enabled, except if CRC insert enabled
CRC_INS_EN	1	CRC insertion enabled, no CRC checking
CRC_CK_FZ	1	CRC error freeze enabled, except if CRC insert enabled
FR_ERR_FZ	1	Frame error freeze enabled
LEN_ERR_FZ	1	Length error freeze enabled
IDLE_ERR_FZ	1	Idle error freeze enabled
RT_ERR_FZ	1	Route error freeze enabled
OVFL_ERR_FZ	1	Overflow error freeze enable
RULE1_MASK	16	Boolean comparison rule 1 mask bits, 1=Valid, 0=Don't Care, see section above on routing
RULE1_REF	16	Boolean comparison rule 1 reference value bits, see section above on routing
RULE2_MASK	16	Boolean comparison rule 2 mask bits, 1=Valid, 0=Don't Care
RULE2_REF	16	Boolean comparison rule 2 reference value bits
SEL1_1	5	Output port mux control for MSB for routing rule 1, see section above on routing
SEL1_0	5	Output port mux control for LSB for routing rule 1
SEL2_1	5	Output port mux control for MSB for routing rule 2
SEL2_0	5	Output port mux control for LSB for routing rule 2
SEL3_1	5	Output port mux control for MSB for routing rule 3
SEL3_0	5	Output port mux control for LSB for routing rule 3
MISS_PORT	3	Output port for misrouted packets. These are packets that cause a route error as described in the section above on error counters. Such a packet is either a packet from *above* that is addressed to an *above* output or a packet that is addressed to a disabled output.
INP_DIR	4	Input direction bits for all 4 input sections. The bit is 1 if the input is connected to an *above* chip and 0 if the input is connected to a *below* chip. This field is used (only) for statistics and to determine if a packet has a route error.
OUT_DIR	4	Output direction bits for all 4 output sections. The bit is 1 if the output is connected to an *above* chip and 0 if the output is connected to a *below* chip. This field is used (only) for statistics and to determine if a packet has a route error.

An input section can be configured to *freeze* on certain errors. Freezing means that after the designated error occurs, the input section automatically disables itself. Once the input section is disabled by a *freeze*, it can be enabled through the control registers. In particular to enable a frozen port, the DISABLE_PORT should be asserted and then unasserted.

Unfortunately a *freeze* may not become effective until 25 system clock cycles after the triggering

error.

An input section is configured to send all misrouted packets to a specific output.

Output Section Configuration Bits (per output section)

Name	Size	Description
BUF_MAX	5	Number of buffers in the input section connected to this output. This is used to initialize the output's buffer free count during reset.

In addition to providing a separate set of configuration registers for each input/output pair, we also provide a set of registers that simultaneously broadcast a single set of configuration data to all of the input/ouput pairs. These registers speed the process of configuring the chip in the case where all inputs and outputs are to be configured identically.

7.3.6 Manufacturing Test Ring Register

In *low-level-test* mode, the maintenance interface will provide access to the manufacturing test rings. In this mode, all of Arctic is placed into a single clock domain. This mode is used to scan new data into and old data out of the manufacturing test rings, and then to cycle the system clock. Only complete scans of all the bits of all the manufacturing test rings are supported.

A scan of the manufacturing test rings is accomplished using special access operations on the manufacturing test ring register. The manufacturing test ring register is 6 bits long. Each bit in the register corresponds to one of the 6 manufacturing test rings in Arctic. Each scan of the manufacturing test ring register causes a scan of one bit of each of the manufacturing test rings. The manufacturing test rings are padded to be ???525??? bits long. So a complete scan of all the manufacturing test rings requires scanning the manufacturing test ring register ???525??? times.

Thus to do a scan of all the manufacturing test rings, and to cycle the system clock the following access operations are issued in *low-level-test* mode.

1)	set_up_llm_scan	This sets up things up.
2)	llm_scan(*datum*)	*datum* is 6 bits long. This scan corresponds to scanning the manufacturing test ring register once and thus it corresponds to scanning one bit on each of the manufacturing test rings. This step is repeated ???525??? - 1 times so that all but the last bit of each manufacturing test ring are loaded.
3)	end_llm_scan(*datum*)	This scans the last bit in each of the manufacturing test rings and cycles the system clock.

IT SHOULD BE NOTED (as stated above) that ALL of this procedure MUST be executed for each *low-level-test* mode scan of the manufacturing test rings. Partial scans are not supported.

While the manufacturing test rings should provide access to most of the state of Arctic, it will be hard to use these rings since state bits that are logically related may not be adjacent in the rings. The organization of the manufacturing test rings will be designed to minimize the cost of the rings. *Low-level-test* mode will probably be exclusively used to run a set of tests (developed using the Motorola Mustang automatic test pattern generation tool) that verify the electrical operation of most of Arctic's gates and the integrity of most of Arctic's nets.

The bit level specification of the manufacturing test rings will be given in a later version of this manual.

8 Error Pin

Arctic provides a single error pin which is basically an *or* of all the errors. In other words this pin is one if and only if any port has any CRC_ERR's, OVFL_ERR's, FR_ERR's, LEN_ERR's, IDLE_ERR's, RT_ERR's, ICLK_ERR's, or BF_ERR's.

9 Initialization

Configuring an Arctic on power up involves a five step procedure. First, the JTAG controller should be initialized according to the JTAG standard using the JTAG reset line. Second, Arctic should be reset using the *system-reset* control register. (It should be noted that during reset the link clocks out of Arctic may change frequency.) Third, the Arctic mode register should be set to *configuration* mode and the various configuration registers should be loaded with the desired values. Fourth, the Arctic mode register should be set to *normal* mode. Finally, the Arctic statistics should be cleared by reading the special access operation register for clearing the statistics. (Of course *get_status* operations should be used to determine when the statistics have been cleared.)

Assuming that all the Arctics in a network have been configured, the network can be initialized/reinitialized using a five step process. First, the inputs and outputs of each chip should be turned off and all external packet sources and external packet receivers should be turned off. Second, all Arctic chips should be reset (using the *system-reset* control register), and all Arctic statistics should be cleared (by reading the special access operation register on each Arctic for clearing statistics and using *get_status* operations to verify the completion of the clears). (It should be noted that during reset the link clocks out of Arctic may change frequency.) Third, all external packet sources and all external packet receivers should be reset. Fourth, the inputs on each chip should be turned on and the external packet receivers should be turned on. Fifth, the outputs on each Arctic chip should be turned on and the external packet sources should be turned on.

10 Signal Pins

As stated above, Arctic has two input pins for the system clock.

As stated above, Arctic has four network inputs and four network outputs. Each requires 20 pins (not counting power and ground pins). These are discussed in more detail above.

As stated above, Arctic has one maintenance interface which is IEEE Std. 1149.1 compliant. A reference for the standard is given above. This interface requires five pins.

As stated above, Arctic has a *manufacturing-test* mode pin.

As stated above, Arctic has an error pin.

We currently are requesting eight additional pins for various reset functions yet to be defined.

Of course, Arctic has a reference pin used to define $V_{threshold}$.

As required by Motorola ASIC, Arctic has an ENID pin. This pin is only used for electrical testing. In normal operation, this pin should be grounded???

11 Register Address Map

11.1 Control Registers

Hex Addr	Register
200	Port/Statistics Control Register
341	System Reset

11.2 Statistics Counters

Hex Addr	Register
280	Output Port 0 PACKETS Statistic
281	Output Port 0 PRIORITY Statistic
282	Output Port 0 UP Statistic
283	Output Port 0 DOWN Statistic
284	Output Port 0 IDLE Statistic
285	Output Port 0 WAIT Statistic
286	Output Port 1 PACKETS Statistic
287	Output Port 1 PRIORITY Statistic
288	Output Port 1 UP Statistic
289	Output Port 1 DOWN Statistic
28A	Output Port 1 IDLE Statistic
28B	Output Port 1 WAIT Statistic
28C	Output Port 2 PACKETS Statistic
28D	Output Port 2 PRIORITY Statistic
28E	Output Port 2 UP Statistic
28F	Output Port 2 DOWN Statistic
290	Output Port 2 IDLE Statistic
291	Output Port 2 WAIT Statistic
292	Output Port 3 PACKETS Statistic
293	Output Port 3 PRIORITY Statistic
294	Output Port 3 UP Statistic
295	Output Port 3 DOWN Statistic
296	Output Port 3 IDLE Statistic
297	Output Port 3 WAIT Statistic
298	Reserved
299	Reserved
29A	Reserved
29B	Reserved
29C	Reserved
29D	Reserved
29E	Reserved
29F	Reserved
2A0	Reserved

2A1	Reserved
2A2	Reserved
2A3	Reserved
2A4	Reserved
2A5	Reserved
2A6	Reserved
2A7	Reserved
2A8	Reserved
2A9	Reserved
2AA	Reserved
2AB	Reserved
2AC	Reserved
2AD	Reserved
2AE	Reserved
2AF	Reserved

2C0	Output Port 0 PACKETS Statistic (clear)
2C1	Output Port 0 PRIORITY Statistic (clear)
2C2	Output Port 0 UP Statistic (clear)
2C3	Output Port 0 DOWN Statistic (clear)
2C4	Output Port 0 IDLE Statistic (clear)
2C5	Output Port 0 WAIT Statistic (clear)
2C6	Output Port 1 PACKETS Statistic (clear)
2C7	Output Port 1 PRIORITY Statistic (clear)
2C8	Output Port 1 UP Statistic (clear)
2C9	Output Port 1 DOWN Statistic (clear)
2CA	Output Port 1 IDLE Statistic (clear)
2CB	Output Port 1 WAIT Statistic (clear)
2CC	Output Port 2 PACKETS Statistic (clear)
2CD	Output Port 2 PRIORITY Statistic (clear)
2CE	Output Port 2 UP Statistic (clear)
2CF	Output Port 2 DOWN Statistic (clear)
2D0	Output Port 2 IDLE Statistic (clear)
2D1	Output Port 2 WAIT Statistic (clear)
2D2	Output Port 3 PACKETS Statistic (clear)
2D3	Output Port 3 PRIORITY Statistic (clear)
2D4	Output Port 3 UP Statistic (clear)
2D5	Output Port 3 DOWN Statistic (clear)
2D6	Output Port 3 IDLE Statistic (clear)
2D7	Output Port 3 WAIT Statistic (clear)
2D8	Reserved
2D9	Reserved
2DA	Reserved
2DB	Reserved
2DC	Reserved
2DD	Reserved
2DE	Reserved
2DF	Reserved

2E0	Reserved
2E1	Reserved
2E2	Reserved
2E3	Reserved
2E4	Reserved
2E5	Reserved
2E6	Reserved
2E7	Reserved
2E8	Reserved
2E9	Reserved
2EA	Reserved
2EB	Reserved
2EC	Reserved
2ED	Reserved
2EE	Reserved
2EF	Reserved
2FF	Clear All Statistics

11.3 Error Counters

Hex Addr	Register
300	Error Counters
301	Error Counters (clear)

11.4 Buffer Free Counters

Hex Addr	Register
240	Buffer Free Counters

11.5 Configuration Registers

Hex Addr	Register
000	Input Port 0 Configuration; Reg 0
001	Input Port 0 Configuration; Reg 1
002	Input Port 0 Configuration; Reg 2
003	Input Port 0 Configuration; Reg 3
004	Input Port 0 and Output Port 0 Configuration; Reg 4
040	Input Port 1 Configuration; Reg 0
041	Input Port 1 Configuration; Reg 1
042	Input Port 1 Configuration; Reg 2
043	Input Port 1 Configuration; Reg 3

044	Input Port 1 and Output Port 1 Configuration; Reg 4
080	Input Port 2 Configuration; Reg 0
081	Input Port 2 Configuration; Reg 1
082	Input Port 2 Configuration; Reg 2
083	Input Port 2 Configuration; Reg 3
084	Input Port 2 and Output Port 2 Configuration; Reg 4
0C0	Input Port 3 Configuration; Reg 0
0C1	Input Port 3 Configuration; Reg 1
0C2	Input Port 3 Configuration; Reg 2
0C3	Input Port 3 Configuration; Reg 3
0C4	Input Port 3 and Output Port 3 Configuration; Reg 4
100	Reserved
101	Reserved
102	Reserved
103	Reserved
104	Reserved
140	Reserved
141	Reserved
142	Reserved
143	Reserved
144	Reserved
180	Reserved
181	Reserved
182	Reserved
183	Reserved
184	Reserved
1C0	Reserved
1C1	Reserved
1C2	Reserved
1C3	Reserved
1C4	Reserved
1E0	Broadcast to all Input Ports; Reg 0
1E1	Broadcast to all Input Ports; Reg 1
1E2	Broadcast to all Input Ports; Reg 2
1E3	Broadcast to all Input Ports; Reg 3
1E4	Broadcast to all Input Ports and Output Ports; Reg 4

???Reserved for Future Registers???

Hex Addr	Register
380	Reserved
3C0	Reserved

12 Register Definitions

12.1 Control Registers

Port/Statistics Control Register

Bit Position	Meaning	
0	DISABLE_INPUT	(port 0)
1	DISABLE_OUTPUT	(port 0)
2	OUTPUT_BLOCK	(port 0)
3	FLUSH	(port 0)
4	DISABLE_INPUT	(port 1)
5	DISABLE_OUTPUT	(port 1)
6	OUTPUT_BLOCK	(port 1)
7	FLUSH	(port 1)
8	DISABLE_INPUT	(port 2)
9	DISABLE_OUTPUT	(port 2)
10	OUTPUT_BLOCK	(port 2)
11	FLUSH	(port 2)
12	DISABLE_INPUT	(port 3)
13	DISABLE_OUTPUT	(port 3)
14	OUTPUT_BLOCK	(port 3)
15	FLUSH	(port 3)
16	NORM	(stats module)
17	PRI	(stats module)

12.2 Error Counters

Error Word

Bit Position	Meaning	
0	CRC_ERR(0)	(port 0)
1	CRC_ERR(1)	(port 0)
2	OVFL_ERR(0)	(port 0)
3	OVFL_ERR(1)	(port 0)
4	FRAME_ERR(0)	(port 0)
5	FRAME_ERR(1)	(port 0)
6	LENGTH_ERR(0)	(port 0)
7	LENGTH_ERR(1)	(port 0)
8	IDLE_ERR(0)	(port 0)
9	IDLE_ERR(1)	(port 0)
10	ROUTE_ERR(0)	(port 0)
11	ROUTE_ERR(1)	(port 0)
12	ICLK_ERR	(port 0)
13	BF_ERR(0)	(port 0)
14	BF_ERR(1)	(port 0)
15	CRC_ERR(0)	(port 1)

16	CRC_ERR(1)	(port 1)
17	OVFL_ERR(0)	(port 1)
18	OVFL_ERR(1)	(port 1)
19	FRAME_ERR(0)	(port 1)
20	FRAME_ERR(1)	(port 1)
21	LENGTH_ERR(0)	(port 1)
22	LENGTH_ERR(1)	(port 1)
23	IDLE_ERR(0)	(port 1)
24	IDLE_ERR(1)	(port 1)
25	ROUTE_ERR(0)	(port 1)
26	ROUTE_ERR(1)	(port 1)
27	ICLK_ERR	(port 1)
28	BF_ERR(0)	(port 1)
29	BF_ERR(1)	(port 1)
30	CRC_ERR(0)	(port 2)
31	CRC_ERR(1)	(port 2)
32	OVFL_ERR(0)	(port 2)
33	OVFL_ERR(1)	(port 2)
34	FRAME_ERR(0)	(port 2)
35	FRAME_ERR(1)	(port 2)
36	LENGTH_ERR(0)	(port 2)
37	LENGTH_ERR(1)	(port 2)
38	IDLE_ERR(0)	(port 2)
39	IDLE_ERR(1)	(port 2)
40	ROUTE_ERR(0)	(port 2)
41	ROUTE_ERR(1)	(port 2)
42	ICLK_ERR	(port 2)
43	BF_ERR(0)	(port 2)
44	BF_ERR(1)	(port 2)
45	CRC_ERR(0)	(port 3)
46	CRC_ERR(1)	(port 3)
47	OVFL_ERR(0)	(port 3)
48	OVFL_ERR(1)	(port 3)
49	FRAME_ERR(0)	(port 3)
50	FRAME_ERR(1)	(port 3)
51	LENGTH_ERR(0)	(port 3)
52	LENGTH_ERR(1)	(port 3)
53	IDLE_ERR(0)	(port 3)
54	IDLE_ERR(1)	(port 3)
55	ROUTE_ERR(0)	(port 3)
56	ROUTE_ERR(1)	(port 3)
57	ICLK_ERR	(port 3)
58	BF_ERR(0)	(port 3)
59	BF_ERR(1)	(port 3)

12.3 Buffer Free Counters

Buffer Free Counter Word

Bit Position	Meaning	
0	BF_CNT(0)	(port 0)
1	BF_CNT(1)	(port 0)
2	BF_CNT(2)	(port 0)
3	BF_CNT(3)	(port 0)
4	BF_CNT(0)	(port 1)
5	BF_CNT(1)	(port 1)
6	BF_CNT(2)	(port 1)
7	BF_CNT(3)	(port 1)
8	BF_CNT(0)	(port 2)
9	BF_CNT(1)	(port 2)
10	BF_CNT(2)	(port 2)
11	BF_CNT(3)	(port 2)
12	BF_CNT(0)	(port 3)
13	BF_CNT(1)	(port 3)
14	BF_CNT(2)	(port 3)
15	BF_CNT(3)	(port 3)

12.4 Configuration Registers

Register 0

Bit Position	Meaning
0	MIS$_PORT[0]
1	MIS$_PORT[1]
2	Unused
3	SEL3_0[0]
4	SEL3_0[1]
5	SEL3_0[2]
6	SEL3_0[3]
7	SEL3_0[4]
8	SEL3_1[0]
9	SEL3_1[1]
10	SEL3_1[2]
11	SEL3_1[3]
12	SEL3_1[4]
13	Unused
14	Unused
15	Unused
16	Unused
17	Unused
18	SEL2_0[0]
19	SEL2_0[1]
20	SEL2_0[2]

21	SEL2_0[3]
22	SEL2_0[4]
23	SEL2_1[0]
24	SEL2_1[1]
25	SEL2_1[2]
26	SEL2_1[3]
27	SEL2_1[4]
28	Reserved
29	Reserved
30	Reserved
31	Reserved

Register 1

Bit Position	Meaning
0	Unused
1	Unused
2	Unused
3	Unused
4	Unused
5	SEL1_0[0]
6	SEL1_0[1]
7	SEL1_0[2]
8	SEL1_0[3]
9	SEL1_0[4]
10	SEL1_1[0]
11	SEL1_1[1]
12	SEL1_1[2]
13	SEL1_1[3]
14	SEL1_1[4]
15	Unused
16	Unused
17	Unused
18	Unused
19	Unused
20	OVFL
21	RT_ERR_FZ
22	IDLE_ERR_FZ
23	LEN_ERR_FZ
24	FR_ERR_FZ
25	CRC_ERR_FZ
26	CRC_INS_EN
27	CRC_CK_EN
28	Reserved
29	Reserved
30	Reserved
31	Reserved

Register 2

Bit Position	Meaning
0	RULE2_REF[0]
1	RULE2_REF[1]
2	RULE2_REF[2]
3	RULE2_REF[3]
4	RULE2_REF[4]
5	RULE2_REF[5]
6	RULE2_REF[6]
7	RULE2_REF[7]
8	RULE2_REF[8]
9	RULE2_REF[9]
10	RULE2_REF[10]
11	RULE2_REF[11]
12	RULE2_REF[12]
13	RULE2_REF[13]
14	RULE2_REF[14]
15	RULE2_REF[15]
16	RULE2_MASK[0]
17	RULE2_MASK[1]
18	RULE2_MASK[2]
19	RULE2_MASK[3]
20	RULE2_MASK[4]
21	RULE2_MASK[5]
22	RULE2_MASK[6]
23	RULE2_MASK[7]
24	RULE2_MASK[8]
25	RULE2_MASK[9]
26	RULE2_MASK[10]
27	RULE2_MASK[11]
28	RULE2_MASK[12]
29	RULE2_MASK[13]
30	RULE2_MASK[14]
31	RULE2_MASK[15]

Register 3

Bit Position	Meaning
0	RULE1_REF[0]
1	RULE1_REF[1]
2	RULE1_REF[2]
3	RULE1_REF[3]
4	RULE1_REF[4]
5	RULE1_REF[5]
6	RULE1_REF[6]
7	RULE1_REF[7]
8	RULE1_REF[8]

9	RULE1_REF[9]
10	RULE1_REF[10]
11	RULE1_REF[11]
12	RULE1_REF[12]
13	RULE1_REF[13]
14	RULE1_REF[14]
15	RULE1_REF[15]
16	RULE1_MASK[0]
17	RULE1_MASK[1]
18	RULE1_MASK[2]
19	RULE1_MASK[3]
20	RULE1_MASK[4]
21	RULE1_MASK[5]
22	RULE1_MASK[6]
23	RULE1_MASK[7]
24	RULE1_MASK[8]
25	RULE1_MASK[9]
26	RULE1_MASK[10]
27	RULE1_MASK[11]
28	RULE1_MASK[12]
29	RULE1_MASK[13]
30	RULE1_MASK[14]
31	RULE1_MASK[15]

Register 4

Bit Position	Meaning
0	INP_DIR[0]
1	INP_DIR[1]
2	INP_DIR[2]
3	INP_DIR[3]
4	Unused
5	Unused
6	Unused
7	Unused
8	OUT_DIR[0]
9	OUT_DIR[1]
10	OUT_DIR[2]
11	OUT_DIR[3]
12	Unused
13	Unused
14	Unused
15	Unused
16	BUF_MAX[0]
17	BUF_MAX[1]
18	BUF_MAX[2]
19	BUF_MAX[3]
20	BUF_MAX[4]

21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved