1 Astro IOC

The Astro block will be referred to as the Runway to IOC (R2I) block. The majority of this chapter describes the operation of the R2I block. It is assumed that the reader is very familiar with Elroy. Information contained in the Elroy ERS will not be repeated in the Astro ERS.

1.1 Overview

The AIOC consists of two major blocks, the R2I and the IOC. The majority of the complexity of the AIOC resides in the IOC. The AIOC design is centered around a 16-line, fully associative coherent cache which resides in the IOC block. The Runway side of the cache is 144-bits wide and operates in the runway frequency domain. Data is routed directly from the Runway pads to the cache. The cache communicates with I/O devices via up to 8 different I/O ropes with an Elroy at the end. The 8 ropes can be bundled in two different configurations to support I/O devices ranging in bandwidth requirements from 250MB/s (single wide rope) to 500MB/s (dual wide rope). The ropes use an Elroy chip to convert to the PCI bus. The AIOC also drives a small point-to-point collection of signals to the Dillon chip which is the interface to the boot ROMs and any other PDH devices that are required.

All Runway transactions with an address that falls in I/O space will be forwarded to the AIOC from the RBIB. The R2I will decode the address and send the transaction to the appropriate destination. Possible destinations include the R2I_regbus, IOC_regbus, elroy_regbus, pdh, or a PCI bus on one of the eight ropes. If the I/O address is outside of Astro's I/O address space the transaction will not be forwarded.

When the AIOC has a transaction it wants to initiate (or complete in the case of PIO reads) it will make a request to the RBIB. The RBIB will arbitrate for the Runway bus on behalf of the AIOC and informs the AIOC 1 cycle prior to becoming the Runway bus master. The AIOC will then drive the request and/or the completion data directly to Astro's Runway pads.

When dma data is returned from the Runway pads to the AIOC it will be stored directly in the cache. Data from Astro's memory controller is always sent to the Runway bus even if it is destined for the IOC. Dma data will always be full cachelines (4 data cycles). The data cycles will always be on consecutive cycles.

A major feature of the AIOC is that the coherent cache straddles an asynchronous frequency domain boundary in Astro. This allows the cache to be snooped at full speed in the Runway bus frequency domain while returning data at full speed in the PCI frequency domain. The Astro design target for the PCI side of the cache is 133MHz and the Runway side of the cache is 125MHz.

1.2 Rope Support

Astro will support a maximum of 8 ropes but only in very restricted configurations. Table 1 shows the rope bundle configuration that will be tested in the initial release of Astro. Other configurations will likely work but will not be tested nor officially supported in the initial implementation of Astro. The ropes can be configured as single-wide ropes to support PCI 2x or double wide to support PCI 4x. Astro will only support ropes running at 133MHz (266MT/s) with Elroy operating at divide by 2 or 4. This results in 66MHz and 33MHz PCI only. Other Configurations and frequencies may be possible but will not be tested.

Rope Number							
0	1	2	3	4	5	6	7
single	single	double	NA	double	NA	single	single

Table 1: Supported Rope Configurations

1.3 Peer-to-peer Support

PCI peer-to-peer (P2P) transactions will only be supported on PCI devices residing on the same PCI bus. Inter-rope P2P will be supported by the design but it will not be verified (and therefore will likely not work). This chapter discusses P2P related transactions as if they are supported.

In Astro, inter-rope P2P traffic will be visible on the Runway bus. This is done primarily to simplify several aspects of the design.

PCI device writes to CPU registers is supported (transaction based interrupts). This type of transaction is viewed by Elroy as the front end of a peer-to-peer transaction

1.4 Regbus I/F

The regbus block is used to drive the R2I_regbus. The R2I_regbus will be routed to all blocks that contain registers that live in the 125MHz frequency domain in Astro. All Astro registers are 64-bits. The regbus is a 32-bit bus so all accesses have 2 data cycles to access the entire 64-bits.

1.5 PDH I/F

The PDH I/F block will be used to communicate with the Dillon chip via a 4-bit point-to-point bus. The Dillon chip provides a connection to the boot ROMs and any other PDH devices that may be required. Astro does not support a PDH cache but instead supports accesses to PDH via memory space accesses which are cacheable in the CPU. Reading PDH via memory space has the additional performance benefit of allowing 64-byte loads from Dillon which is much more efficient than the 8-byte loads that occur through I/O space.

The PDH will be accessible with a cacheline read transaction on Runway at addresses 0xEF_0000_0000 to 0xEF_FFFF_FFFF. The 16MB Dillon address space will be aliased across this entire space. This space is read-only and will load 64-bytes of data about 3 times faster than doing 8, 8-byte READ_SHORT transactions to PDH in IO space.

1.6 PIO Write Coalescing

The R2I block will attempt to coalesce PIO writes into long PCI bursts. Both 4-byte and 8-byte transactions will be coalesced. PIO writes will be coalesced only if they are in the LMMIO or GMMIO address ranges. There is a bit in the IOC_CTRL register that must be set to enable coalescing.

The R2I block does two different stages of coalescing. The first is to assemble 4-byte and 8-byte data from Runway into 16-byte chunks. The second stage involves deciding if the 16-byte chunk is coalescable with the previous 16-byte data chunk. All data that is not coalescable requires a new address to be written to the CD fifo along with the data.

Writes will be coalesced only if the are in MMIO space and are 4 or 8 byte writes to ascending addresses. Any non-coalescable transaction will break the coalescing stream even if it may have been possible to maintain it (eg. an R2I register access). Once the stream is broken all data will be forwarded immediately. Write data will be held in the R2I a maximum of 16 cycles (without stop_io asserted) waiting for the next write transaction. After 16 cycles the data will be forwarded to the CD fifo and the coalesce stream will be broken.

1.7 Interrupts

The R2I block will convert interrupt transactions from the format generated by Elroy into a PA style interrupt transaction. The interrupt block will use the information already contained in the interrupt vector to create a PA style interrupt. When an interrupt transaction shows up in the I/O fifo, the destination ID will be used to generate the address for a write short to a CPU's IO_EIR. The "interrupt vector" will be used as the value written to the EIR. Note that CPUs running in 32-bit mode only look at the lower 5-bits of the interrupt vector and CPUs in 64-bit mode look at the lower 6-bits.

The 40-bit address generated for the Runway WRITE_SHORT transaction that is the result of an interrupt is shown in Table 2.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
N/A																								0xFF							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF				Dest EID[7:0]								Dest ID[7:0]								0x000											

Table 2: PA Interrupt Address

Note that an I/O card can generate an interrupt in three different ways. It can pull the PCI interrupt wire provided by Elroy. Each wire has a unique set of registers that contain the Dest ID, Dest EID and VECTOR values which are used to generate the WRITE_SHORT interrupt transaction on Runway. An interrupt can also be generated by the PCI card by mastering a write transaction with the address and data shown in Table 3. Astro will extract the relevant information from the transaction and convert it to a PA interrupt as described above.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Address	0xFEE												Dest ID[7:0]								Dest EID[7:0]								unused			
Data	unused (by Astro)																										VECTOR[5:0]					

Table 3: PCI Interrupt Message

The third way for a PCI card to generate an interrupt is by writing the interrupt vector directly to the CPU's IO_EIR. If Elroy is configured properly it will recognize the address as a write to a remote location in IO space and forward the write to Astro. Astro will F-extend the address and forward the transaction to the Runway bus and hence the destination CPU.

1.8 Transfer of Control (TOC)

The R2I block contains a TOC_L input signal. Astro will issue a directed CMD_RESET write (data=0x00000005) to the monarch processor’s IO_COMMAND register in response to a negative edge on the TOC_L signal. The TOC_L signal is asynchronous and must be asserted for > than 2 periods of the Runway clock (>15nS). The address of the monarch’s IO_COMMAND register is stored in the TOC_MONARCH_CLIENT_ID Register. Firmware must initialize the 4-bit client_id field to point to the monarch processor. The bit definition of the TOC_MONARCH_CLIENT_ID register is described in Register 1. All other fields of the register are read-only and will be unaffected by writes. The upper 32-bits of the register will always return 0s on reads and are unaffected by writes.

1.8.1.1.1 TOC_MONARCH_CLIENT_ID Register

<table>
<tr><td>MSB</td><td colspan="30">TOC_MONARCH_CLIENT_ID Register
(address: 0xFF_FED0_0010)</td><td>LSB</td></tr>
<tr><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td><td>48</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td></tr>
<tr><td colspan="24">unimplemented</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="32">Power On Initialization</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">0xF</td><td colspan="10">flex=1111111110</td><td>1</td><td colspan="4">client_id</td><td>0</td><td colspan="6">regset=000000</td><td colspan="4">reg=1100</td><td colspan="2">00</td></tr>
<tr><td colspan="32">Power On Initialization</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Size: 64-bit only		
Field	**Access**	**Description**
reg	R	register number
regset	R	register set
client_id	R/W	client_id
flex	R	flex

Register 1: TOC_MONARCH_CLIENT_ID Register

There is a bit in the IOC_CTRL register that is used to enable the TOC function. This bit is cleared (disabled) at power-up and must be initialized by firmware. The bit is also cleared anytime a TOC is issued and must be set before additional TOC's will be issued. This has the effect of de-bouncing the TOC signal.

1.9 PIO Address Decoding

The RBIB will forward all I/O addresses to the R2I block. The R2I block will then attempt to decode the address into one of several possible destinations. Table 4 shows the possible destinations and their respective address ranges for PIO transactions. Transactions that do not hit in one of these ranges will be discarded.

Destination	Starting Address	Block Size	Description
R2I_regbus	0xFF_FED0_0000	128KB	All Astro registers in the Runway frequency domain. This includes registers in the MC, RBIB, and R2I blocks.

Destination	Starting Address	Block Size	Description
IOC_regbus	0xFF_FED2_0000	64KB	All Astro registers in the I/O frequency domain. These transactions will be forwarded to the cd_fifo with a type indicating an IOC register access.
Elroy_regbus	0xFF_FED3_0000	64KB	These transactions will be forwarded to the cd_fifo with a type indicating an Elroy register access.
PDH	0xF0_F000_0000	16MB	These transactions will be forwarded to the PDH block in Dillon.
Interrupt Acknowledge	0xFF_FEF0_0000	1M	A read_short to any address in this range will result in an interrupt acknowledge transaction. The preferred address to use is FF_FEFE_0000. Interrupt acknowledge has not been tested in Astro and is not expected to be used.
PCI I/O port space distributed range	0xFF_FEE0_0000	2K-512K	Transactions in this address range will get converted to PCI I/O space (IOS) transactions. Each Elroy can use a maximum of 64KB of IOS. This range allows software to get to every address on every Elroy when set to its maximum size. Inter-rope P2P will not work if this range is greater than 64KB. The lower 16-bits of the Runway address are used as the PCI address for these transactions.
PCI I/O port space directed range	programmable	256B-64KB	Transactions in this address range will get converted to PCI I/O space (IOS) transactions. Each Elroy can use a maximum of 64KB of IOS.
PCI LMMIO space distributed range	programmable	8-64MB	Addresses in this space will be divided into 8 equal segments (one per rope). Transactions will be forwarded to the appropriate rope and result in a PCI memory transaction with a 32-bit address. The lower 32-bits of the Runway address will be used for the PCI address.
PCI LMMIO space directed range0	programmable	1-64MB	Transactions in this address space will be forwarded to the appropriate rope and result in a PCI memory transaction. The lower 32-bits of the address will be used for the PCI address. The destination rope is programmable.
PCI LMMIO space directed range1	programmable	1-64MB	Transactions in this address space will be forwarded to the appropriate rope and result in a PCI memory transaction. The lower 32-bits of the address will be used for the PCI address. The

Destination	Starting Address	Block Size	Description
			destination rope is programmable.
PCI LMMIO space directed range2	programmable	1-64MB	Transactions in this address space will be forwarded to the appropriate rope and result in a PCI memory transaction. The lower 32-bits of the address will be used for the PCI address. The destination rope is programmable.
PCI LMMIO space directed range3	programmable	1-64MB	Transactions in this address space will be forwarded to the appropriate rope and result in a PCI memory transaction. The lower 32-bits of the address will be used for the PCI address. The destination rope is programmable.
PCI GMMIO distributed memory space	programmable	4-16GB	Addresses in this space will be divided into 8 equal segments. The first 64MB of each segment will be mapped down to 64KB of IOS for that rope. All other addresses will result in a PCI memory transaction with a dual address cycle (DAC). The entire Runway address will be used to generate the PCI addresses for these transactions.

Table 4: Decoder Ranges

All programmable ranges are required to be naturally aligned and a power of 2 in size. There are two types of ranges, directed and distributed. All transactions with addresses in a directed range are forwarded to a specific rope. The distributed range is divided into 8 equal segment with each segment tied to a specific rope. These ranges are specified by a set of three register, a BASE register, a MASK register and a ROUTE register. The BASE register contains the starting address for the address range. The “Range Enable” bit (RE) in the BASE register is set to enable the range registers for decoding. The mask register specifies the size of the range by specifying which bits in the base register are compared with the incoming address. A “1” in the mask register will cause the same bit position in the base register to be compared with the incoming address. A “0” in the mask register will inhibit the compare for that position. If the incoming address matches the base register in all bit positions that are doing compares the address is considered to fall within that range. The ROUTE register specifies the rope number for directed ranges and the bit positions in the incoming address to use for the rope number in distributed ranges. Bits which are specified as “unimplemented” will always return a 0 when read and are not affected by writes. The unimplemented bits will not be used during compares. Directed ranges will always take precedence over distributed ranges. Directed ranges should not overlap with each other.

1.9.1 PCI GMMIO Distributed Range Registers

The PCI GMMIO distributed range is 4GB-16GMB in size and is distributed evenly between all 8 ropes. The first 64MB of each segment of this range is used for I/O port space. The PCI address for I/O port space accesses will be {16'h0000,adr[25:12],adr[1:0]} where adr[39:0] is the address generated by the CPU. I/O port space accesses are restricted to aligned 4-byte (or less) transactions. If transactions cross a 4-byte address boundary then only the bytes in the lower 4 byte lanes will be transferred. The I/O port space accesses can be disabled by setting a bit in the IOC_CTRL register. All other Runway PIO transactions to this range will result in DAC PCI memory space transactions using the full Runway address and directed to the specified rope.

MSB	GMMIO_DIST_BASE Register (address 0xFF_FED0_0378)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	base_adr			
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	X	X	X	X
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																															RE
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
base_adr	R/W	Base address of the range
RE	R/W	Range Enable

Register 2: GMMIO_DIST_BASE Register

MSB	GMMIO_DIST_MASK Register (address 0xFF_FED0_0380)	LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	adr_mask			
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	X	X	X	X

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

Size: 64-bit only		
Field	**Access**	**Description**
adr_mask	R/W	Selects which bits of the address to use during the comparison

Register 3: GMMIO_DIST_MASK Register

MSB	GMMIO_DIST_ROUTE Register (address 0xFF_FED0_0388)	LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
rid_lsb						unimplemented																									
Power On Initialization																															
X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Size: 64-bit only		
Field	**Access**	**Description**
rid_lsb	R/W	rope ID least significant bit

Register 4: GMMIO_DIST_ROUTE Register

The rid_lsb field of the GMMIO_DIST_ROUTE register is the routing ID and specifies which bit of the incoming address is the least significant bit of the 3-bit rope number. Legal values for rid_lsb are 29-33.

1.9.2 PCI LMMIO Distributed Range Registers

The PCI LMMIO distributed range is 8MB-64MB in size and is distributed evenly between all 8 ropes. Runway PIO transactions to this range will result in SAC PCI memory space transaction to the specified rope.

MSB	**LMMIO_DIST_BASE Register (address 0xFF_FED0_0360)**																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	base_adr					unimplemented																						RE
Power On Initialization																															
1	1	1	1	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
base_adr	R/W	Base address of the range
RE	R/W	Range Enable

Register 5: LMMIO_DIST_BASE Register

MSB	**LMMIO_DIST_MASK Register (address 0xFF_FED0_0368)**																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	adr_mask			unimplemented																						
Power On Initialization																															
1	1	1	1	1	1	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
adr_mask	R/W	Selects which bits of the address to use during the comparison

Register 6: LMMIO_DIST_MASK Register

MSB	LMMIO_DIST_ROUTE Register (address 0xFF_FED0_0370)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
rid_lsb						unimplemented																									
Power On Initialization																															
X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
rid_lsb	R/W	rope ID least significant bit

Register 7: LMMIO_DIST_ROUTE Register

The rid_lsb field of the LMMIO_DIST_ROUTE register is the routing ID and specifies which bit of the incoming address is the least significant bit of the 3-bit rope number. Legal values for rid_lsb are 20-23.

1.9.3 PCI LMMIO Directed Range Registers

Astro supports 4 directed ranges for PCI transactions. The PCI LMMIO directed ranges are 8MB-64MB in size and are directed to a single rope. If peer-to-peer is supported then each rope is restricted to using a maximum of two LMMIO ranges. These two ranges could come from the distributed range and one directed range or from two directed ranges (if the distributed range is never used for that rope). Runway PIO transactions to this range will result in SAC PCI memory space transaction to the specified rope. The PCI address will be the lower 32-bits of the Runway address.

Astro has four copies of each of the LMMIO range registers which are described in Register 8, Register 9 and Register 10. Table 5 gives the address and name for each of the 12 registers.

Register Name	Address
LMMIO_DIRECT0_BASE	0xFF_FED0_0300
LMMIO_DIRECT0_MASK	0xFF_FED0_0308
LMMIO_DIRECT0_ROUTE	0xFF_FED0_0310
LMMIO_DIRECT1_BASE	0xFF_FED0_0318

LMMIO_DIRECT1_MASK	0xFF_FED0_0310
LMMIO_DIRECT1_ROUTE	0xFF_FED0_0328
LMMIO_DIRECT2_BASE	0xFF_FED0_0330
LMMIO_DIRECT2_MASK	0xFF_FED0_0338
LMMIO_DIRECT2_ROUTE	0xFF_FED0_0340
LMMIO_DIRECT3_BASE	0xFF_FED0_0348
LMMIO_DIRECT3_MASK	0xFF_FED0_0350
LMMIO_DIRECT3_ROUTE	0xFF_FED0_0358

Table 5: LMMIO_DIRECT Register Addresses

MSB **LMMIO_DIRECT_BASE Register** LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	base_adr					unimplemented																						RE
Power On Initialization																															
1	1	1	1	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
base_adr	R/W	Base address of the range
RE	R/W	Range Enable

Register 8: LMMIO_DIRECT_BASE Register

PCI_LMMIO_DIRECT_MASK Register

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	adr_mask			unimplemented																						
Power On Initialization																															
1	1	1	1	1	1	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
adr_mask	R/W	Selects which bits of the address to use during the comparison

Register 9: LMMIO_DIRECT_MASK Register

LMMIO_DIRECT_ROUTE Register

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																													rope_num		
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X

Size: 64-bit only		
Field	**Access**	**Description**
rope_num	R/W	Destination rope number for this address range

Register 10: LMMIO_DIRECT_ROUTE Register

The rope_num field specifies the rope number that will be the target of transactions with addresses in the directed range.

1.9.4 PCI I/O Port Space Distributed Range Registers

The PCI IOP distributed range is 2KB-512KB in size and is distributed evenly between all 8 ropes. Runway PIO transactions to this range will result in a PCI I/O space transaction to the specified rope. The smallest range size (2KB) allows 256 bytes per rope. The largest size

(512KB) allows the entire 64KB I/O ports space for each rope to be fully accessible by software. Note that this space resides at a fixed location and therefore only the RE bit in the base register can be modified. It should also be noted that inter-rope P2P is not possible if this range is made larger than 64KB.

MSB — IOS_DIST_BASE Register (address 0xFF_FED0_0390) — LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	1	1	1	0	0	unimplemented																		RE
Power On Initialization																															
1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
RE	R/W	Range Enable

Register 11: IOS_DIST_BASE Register

MSB — IOS_DIST_MASK Register (address 0xFF_FED0_0398) — LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	adr_mask								unimplemented										
Power On Initialization																															
1	1	1	1	1	1	1	1	1	1	1	1	1	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
adr_mask	R/W	Selects which bits of the address to use during the comparison

Register 12: IOS_DIST_MASK Register

MSB	IOS_DIST_ROUTE Register (address 0xFF_FED0_03A0)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
rid_lsb						unimplemented																									
Power On Initialization																															
X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
rid_lsb	R/W	Routing ID least significant bit

Register 13: IOS_DIST_ROUTE Register

The rid_lsb field of the PCI_IOP_DIST_ROUTE register is the routing ID and specifies which bit of the incoming address is the least significant bit of the 3-bit rope number. Legal values for rid_lsb are 8-16.

1.9.5 PCI I/O Port Space Directed Range Registers

The PCI IOP directed range is 2KB-64KB. Runway PIO transactions to this range will result in a PCI I/O space transaction to the rope specified by the PCI_IOP_DIRECT_ROUTE register. The smallest range size (2KB) allows 256 bytes per rope. The largest size (64KB) allows the entire 64KB I/O port to be directed to any single rope. Note that this range can be used to emulate the Dino I/O registers by setting the PCI_IOP_DIRECT_ROUTE register for every access to IOP.

MSB	IOS_DIRECT_BASE Register (address 0xFF_FED0_03C0)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	base_adr								unimplemented							RE
Power On Initialization																															
1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
base_adr	R/W	Base address of the range
RE	R/W	Range Enable

Register 14: IOS_DIRECT_BASE Register

MSB	IOS_DIRECT_MASK Register (address 0xFF_FED0_03C8)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																								1	1	1	1	1	1	1	1
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	adr_mask								unimplemented							
Power On Initialization																															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
adr_mask	R/W	Selects which bits of the address to use during the comparison

Register 15: IOS_DIRECT_MASK Register

MSB	IOS_DIRECT_ROUTE Register (address 0xFF_FED0_03D0)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																													rope		
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X

Size: 64-bit only		
Field	**Access**	**Description**
rope_num	R/W	Destination rope number for this address range

Register 16: IOS_DIRECT_ROUTE Register

The rope field of the IOS_DIRECT_ROUTE register the 3-bit rope number for this range.

1.10 Transaction ID Encodings

The R2I block encodes information in the Runway transaction ID's it uses when mastering a Runway transaction. Table 6 defines the TID encodings.

TID	Description
00XXXX	DMA read data – The XXXX specifies the cacheline entry where the returning data should be stored.
010XXX	P2P read - The result of a PCI initiated read. XXX specifies the destination rope number. This is not a tested operations.
10XXXX	TLB miss data from the IOPDIR - XXXX specifies the location in the TC to store the data.
110000	Used for PCI initiated writes. This is supported only for writes to CPU registers (Interrupts).

Table 6:IOC TID Encodings

1.11 Coherency Control

The R2I block is responsible for the cache coherency checks required to keep the I/O cache coherent. When a coherent transaction is detected on Runway the R2I block will cause the CTAG to compare the address on the bus to any valid lines that may be stored at the time. The results of the compare will be written to the CC_FIFO. The CC_FIFO is 9 bits wide (hit, hit_cea[3:0], coh_adr_type[2:0], ioc_trans) and 16 entries deep. The Runway protocol will prevent the CC_FIFO from ever overflowing.

Another state machine in the R2I block (CC_CTRL) will unload the CC_FIFO and drive the results of the coherency check to the RBIB using the IOC_COH [1:0] signals. Table 7 outlines the possible conditions and the resulting action that must be taken when unloading the CC_FIFO. Ownership of the line is assumed (the own bit is set) once a response has been given on the IOC_COH signals to the initial transaction that requested the data. Astro 2.0 and later made a change that prevents DMA data from being used by the IOC until Astro has ownership of the line. This is not required by the Runway spec but avoids many nasty coherency related corner cases.

1.11.1 Sync Transactions

If there is a cache_sync or dma_sync transaction on Runway, the IOC will mark all the lines in the cache that are queued to be flushed. Once those lines have all been flushed the R2I block is free to sign-off on the original sync transaction. Note that the sync transaction does not flush the entire cache. As a result the sync transaction will only have the desired results when doing inbound DMA with full cachelines or with the aggressive flush hint set.

If there are no sync transactions in progress the sync operation will be started immediately following the transaction. If there is already a pending sync, the sync will not be initiated until the sync transaction reaches the head of the CC_FIFO.

hit	IOC trans	type	line status	Action Required
NA	1	NA	NA	Indicate OK
NA	NA	init_sync	NA	Wait for the sync_complete signal to be asserted by the cache and then indicate OK
NA	NA	sync	NA	Assert the dma_sync signal and then wait for the sync_completed signal to be asserted. Then indicate OK.
1	0	shared	shared	Indicate shared
1	0	shared or flush	priv0	Assert the invalidate signal. Wait for invalid response. If still priv0 then indicate OK immediately, else queue the line for a writeback, indicate OK when the writeback is complete.

hit	IOC trans	type	line status	Action Required
1	0	shared or flush	priv	Queue the line for a writeback, indicate OK when the writeback is complete.
1	0	shared or flush	unowned	Indicate OK

Table 7: Actions required in response to coherency checks.

1.12 Non-Coherent Mode

Due to the complexity of cache coherency, Astro added a non-coherent mode to work around potential bugs with the initial silicon. When in this mode all IOC initiated transactions will be issued as READ transactions (instead of READ_PRIV or READ_SHAR_OR_PRIV). The IOC_TRANS bit will be set for all entries in the CC_FIFO which results in a quick OK response from the IOC for all transactions without actually looking in the cache. For full cacheline writes there would be no reads required prior to writing the data to memory.

It is important in non-coherent mode that cachelines not be allowed to remain in the cache. Unused cachelines will become stale and may be used by the next I/O device to do DMA to that address. The problem is most easily avoided turning off all prefetching and turning on aggressive flushing. A carefully written driver can still prefetch as long as all prefetched cachelines are eventually consumed. This is accomplished by always stopping the DMA burst at a “block boundary”. Block boundaries can be set to naturally aligned 512, 1K, 2K, or 4K address boundaries by selecting the appropriate DMA hint.

Non-coherent mode was used for the initial turn-on of Asto 1.0 only until the software was capable of supporting coherent mode. This mode has not been tested since.

1.13 Mastering a Runway Transaction

The AIOC has several sources of transactions which need to be mastered on Runway. All the requests the AIOC is capable of generating are listed in Table 8. Note that all requests are passed directly to the Runway bus. The AIOC has separate request/grant lines for all types of requests. The RBIB determines the relative priorities by the way it issues the grant signals.

Type	Description
IO read return	This is the result of a PIO read from a CPU or a peer-to-peer read between ropes (could be MMIO or IOP). The MID and TID from the original transaction will be used. The maximum data size is 8 bytes. The byte enables will be set to reflect the side of the bus that contains the data.

IO read request	An IO read request is usually the result of a peer-to-peer transaction although it could be a PCI device reading a register in a CPU or an Astro register. All read requests will be "F" extended prior to issuing them on Runway. TID[5] will always be set to reflect the fact that this is an I/O generated request (probably peer-to-peer). TID[2:0] will be set to the requesting rope number. A maximum of 8 bytes will be requested. This transaction type is not supported.
IO write request	Usually the result of a P2P transaction or a PCI interrupt. Causes a write_short on Runway. The request will be "F" extended prior to issuing it on Runway.
Memory read request	Requests can be shared, private or private0. They are always a full cacheline in size. Requests can be generated by either the cache or the TLB. TID[4] is set to indicate a TLB generated request. TID[3:0] are used to identify the cache or TLB entry to store the data when it is returned. If the request is marked as private0, no data is required to successfully complete the transaction. The AIOC will always attempt to issue back-to-back requests to maximize Runway bandwidth.
Memory write	Memory writes are always a full cacheline. They are issued as the result of flushing algorithms in the cache or because of a dirty-hit during a coherency check.

Table 8:AIOC Generated Requests

1.14 Other R2I Registers

All registers in the R2I block are 64-bits registers. Bits that are defined as unimplemented will return 0's when read and should be written with 0's during write transactions.

1.14.1 Revision ID Register

The Revision ID register (RID) is a read-only register used by software to identify the Revision number of Astro. The Astro RID is defined in Register 17.

MSB	RID Register (address 0xFF_FED0_0000)																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																											rev_id				
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size: 64-bit only		
Field	**Access**	**Description**
rev_id[2:0]	R	(rev_id[2:0] + 1) is the major revision number of this Astro
rev_id[4:3]	R	rev_id[4:3] is the minor revision number of this Astro

Register 17: Revision ID Register

For example:

rev_id[4:0] = 5’b00000 => Astro 1.0

rev_id[4:0] = 5’b01001 => Astro 2.1

rev_id[4:0] = 5’b00010 => Astro 3.0

1.14.2 I/O Control Register

The I/O Control register (IOC_CTRL) is defined in Register 18. The definition of each bit position is given in Table 9. The IOC_CTRL powers up with values believed to be the normal mode of operation. Most bits are provided as a means to deal with unexpected features (bugs) discovered after tape-release. One exception is the TE bit which needs to be set to re-enable TOC’s following a TOC.

IOC_CTRL Register (address 0xFF_FED0_0008)

MSB … LSB

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented								reserved									DC	DD	CC	D4	ID	NC	RM	L0	RC	IS	OS	IE	DE	CE	TE
Power On Initialization																															
0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	1	1	0	0	1	?	1	0

Size: 64-bit only		
Field	**Access**	**Description**
TE	R/W	TOC Enable
CE	R/W	Coalesce Enable
DE	R/W	Dillon Enable
IE	R/W	IOS Enable
OS	R/W	Outbound Synchronous
IS	R/W	Inbound Synchronous
RC	R/W	Read Current Enable
L0	R/W	0-length read Enable
RM	R/W	Real Mode
NC	R/W	Non-coherent Mode
ID	R/W	Interrupt Disable
D4	R/W	Disable 4-byte coalescing
CC	R/W	Increase Coalescing counter value to 32 (from 17)
DD	R/W	Disable distributed range coalescing (LMMIO only)
DC	R/W	Disable the coalescing counter

Register 18: IOC_CTRL

Symbol	Bit Position	Description
TE	0	TOC Enable - When TE is set, a positive edge on the TOC signal will cause Astro to generate a write_short which will result in a TOC. The TE bit will be cleared by hardware when the TOC is issued. The TOC signal will be ignored when the TE bit is cleared.
CE	1	Coalesce Enable - When CE is set, contiguous write_short transactions to the same rope will be coalesced into a single PCI transaction (provided the data comes in fast enough to keep up with PCI). When CE is cleared no coalescing will happen.
DE	2	Dillon Enable - This bit is set by hardware at reset when a Dillon chip is present. When the bit is cleared the R2I will forward PDH transactions to

Symbol	Bit Position	Description
		rope0. The DE bit can be written by software.
IE	3	IOS Enable - This bit is used to enable the I/O port space that is available through the first 64MB of each segment of the GMMIO distributed address space. When this bit is cleared transactions to this entire space will result in PCI memory transactions.
OS	4	Outbound Synchronous - This bit is set by software if the synchronizers can be bypassed in the outbound direction.
IS	5	Inbound Synchronous - This bit is set by software if the synchronizers can be bypassed in the inbound direction.
RC	6	Read Current Enable - This bit enables Astro to issue a read current transaction. When cleared it will issue a READ_SHARED_OR_PRIVATE instead. Note that read currents will only be issued if they are enabled in Elroy's hint registers.
L0	7	0-length read enable - This bit allows the ioc_0_length hint to be set when a DMA write updates an entire line.
RM	8	Real Mode - This bit forces the virtual index to be the same as address bits 19:12. This allows DMA without the use of an IOPDIR when the CPU is running in real mode. The TLB range enable bit should be cleared when running in this mode.
NC	9	Non-Coherent mode – This bit inhibits Astro from maintaining the coherency of the IOC cache. See the section on Non-coherent mode for more details
ID	10	Interrupt Disable - This bit was intended to force Astro to generate interrupts using 8-byte writes.
D4	11	Disable 4-byte coalescing – When set, coalescing will only be attempted if the Runway transaction is an 8-byte transaction. Note: This feature doesn't work in all cases in Astro 2.1. In some cases it can send Elroy a 12-byte write which can result in a 0-byte enable write on PCI (w/ Elroy 3.0 or earlier). This is not a violation of the PCI spec but several known PCI devices have a problem with it.
CC	12	Coalescing counter increase – When set, the coalescing time-out counter will count to 32 instead of 17 before breaking the coalescing stream.
DD	13	Disable distributed range coalescing – When set, coalescing will not be attempted if the address is decoded by the lmmio distributed range.

Symbol	Bit Position	Description
DC	14	Disable coalesce counter – When set, the coalescing counter will not run which means that the only way to break the coalesce stream in Astro is with a non-coalescable PIO transaction. Note that when this bit is set a single write will remain in Astro until flushed out by a non-coalescable transaction.

Table 9: IOC_CTRL Register Bit Definition

1.14.3 Scratch Registers

Astro initially added 4 registers which were intended to be used for graphics flow control. These registers are still in the design but there is no planned use for them. They have not been tested! These registers implement the lower 8-bits and each one lives on its own 4K page. They are general purpose read/write registers.

MSB	SCRATCH_REG<n>																														LSB
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
unimplemented																															
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
unimplemented																								data[7:0]							
Power On Initialization																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

Size: 64-bit only		
Field	**Access**	**Description**
data	R/W	General purpose read/write bits

Register 19: Scratch Registers

The addresses of the four Scratch registers are given in Table 10.

Register Name	Address
SCRATCH_REG0	0xFF_FED0_4200
SCRATCH_REG1	0xFF_FED0_5200
SCRATCH_REG2	0xFF_FED0_6200
SCRATCH_REG3	0xFF_FED0_7200

Table 10: Scratch Register Addresses

1.15 CD_FIFO Interface

All PIO and peer-to-peer transactions are forwarded to the IOC via the CD_FIFO. The CD_FIFO is split between address and data. The address side is only written when starting a new transaction. The address side is 8 entries deep and the data side is 32 entries deep. When there are more than 4(?) entries in the address side or more than 16 entries in the data side the STOP_IO signal will be asserted. The address side has a 4-bit type field that identifies the type of transaction and a 2-bit length field that identifies amount of data transferred with write transaction. Table 11 defines the type field and Table 12 defines the length field. All types not listed in the table are undefined.

TYPE[3:0]	Description
0	IIOC register transaction
1	PIO MMIO transaction
2	PIO non-posted IOP transaction
3	Interrupt acknowledge
4	Peer-to-peer read return data
5	Peer-to-peer MMIO transaction
6	Peer-to-peer non-posted IOP transaction
7	Elroy register transaction
8	Posted IOP transaction

Table 11: CD_FIFO Type Field Definition

LEN[3:0]	Description
0	8 bytes
1	16 bytes
2	32 bytes (not supported)
3	64 bytes (not supported)

Table 12: CD_FIFO Length Field Description

When a read transaction is written to the address side of the CD_FIFO a corresponding "dummy" data entry must be written to the data fifo if the previous transaction was a write. If the previous transaction was a read then the dummy data is not required.

Write coalescing is done by writing 16 byte values into the data fifo without writing a new entry into the address fifo. There is no limit to the number of bytes that can be coalesced in this way or the amount of time between writes

1.16 IO_FIFO Interface

The IO_FIFO is used to pass transactions and data returns from the IOC to the R2I block. The IO_FIFO has a 5-bit type field which identifies the type of data in each fifo entry. Table 13 defines the IO_FIFO type field.

TYPE[4:0]	Description
0001	PIO MMIO read data, PIO register read data, or PIO IOP read data
0010	Interrupt address
0011	Interrupt data
0100	P2P MMIO write address
0101	P2P MMIO write data
0110	P2P IOP write address
0111	P2P IOP write data
1000	P2P MMIO read address

TYPE[4:0]	Description
1001	P2P MMIO read data or IOP read data or IOP write completion response
1010	P2P IOP read address
1100	Hard fail (PIO read data)
1101	Hard fail (P2P read return/write completion)

Table 13: IO_FIFO Type Field Definition