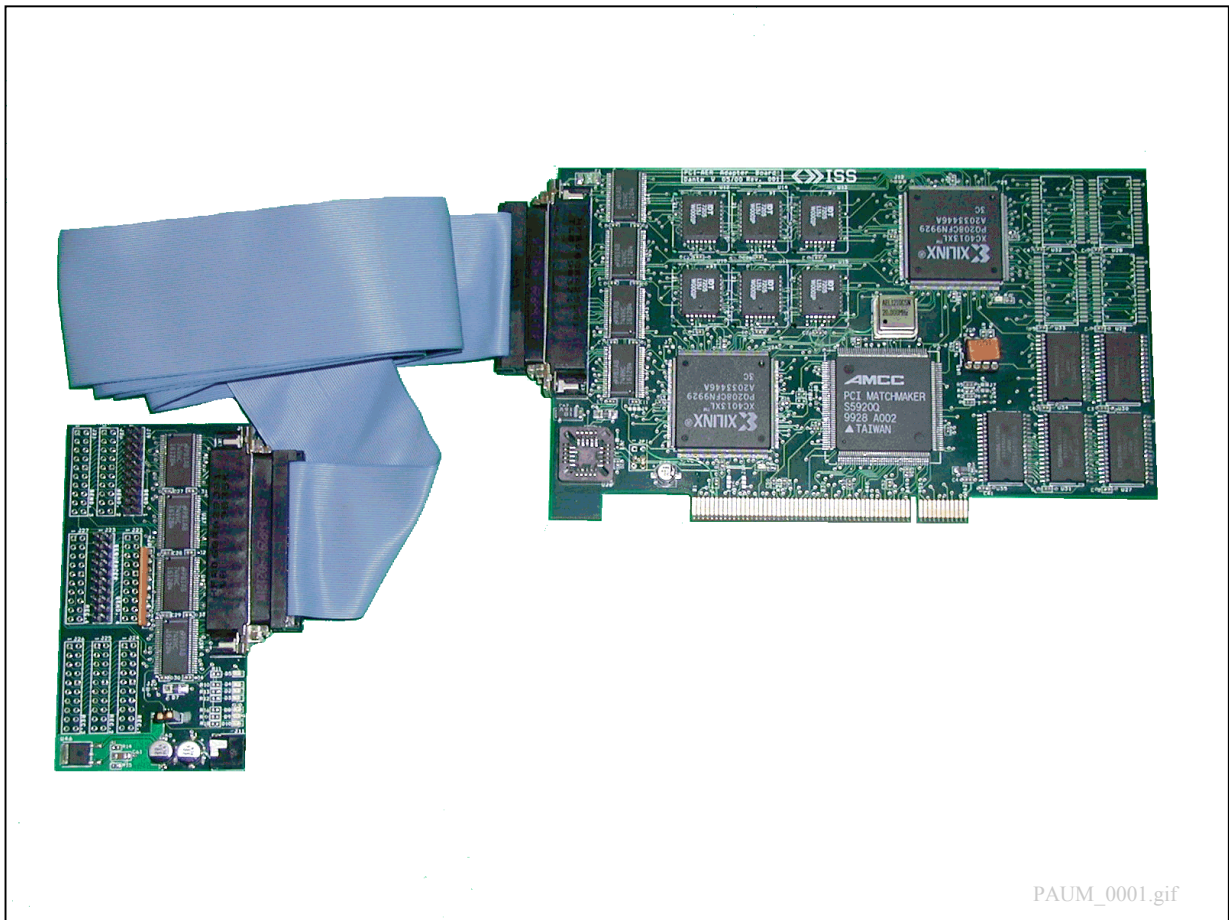


PCI - AER Adapter board User Manual

Dante V.
Istituto Superiore di Sanità
Rome, Italy
2002 – 2003

(Rel. 1.1 - 5 November 2004)
(FPGA1 rel. 4202; FPGA2 rel. 4203)



Release history

- Release 1.0 of this document refers to implementation Beta 1 of both FPGA. Therefore the release register for FPGA1 has value 4201h and release register for FPGA1 has value 4201H.
- Release 1.1 of this document was updated during my visit at INI and refer to the solution of some bugs signalled on the Adrian_Issue.doc document. Some corrections on FPGA2's State Machine has been necessary. The new value for FPGA Release register are 4201 for FPGA1 and 4203 for FPGA2. IN Release.doc document are reported the modification bring to the PCI-AER board and FPGA configuration. For Software change see the Adrian M. Whatley Software and Drivers documentation.

Release history I

Table Index..... IV

Figure Index IV

WARNING1

Overview 2

 Description of the main PCI-AER board blocks..... 3

Block description..... 4

 AER CLOCK..... 4

 TIME Counter 4

 ARBITER 4

 MONITOR..... 5

 SEQUENCER 6

 MAPPER 8

 MAPPER-IN 9

 FIFO..... 9

 MAPPER-OUT 9

 MMU and SRAM architecture..... 9

 AER DEMUX 10

Description of the Cable Adapter board blocks 11

PCI-AER system Board Layout 12

External Board Led description..... 13

PCI CONFIGURATION SPACE FOR PCI-AER ADAPTER BOARD.....14

Base Address Register and Memory Region 14

 Base Address 0. S5920 Register 14

 Base Address 1. Registers of PCI-AER board..... 15

 Base Address 2. Access to the FIFO 15

 Base Address 3. Access to MAPPER SRAM..... 15

REGISTER ARCHITECTURE OF PCI-AER BOARD.....16

FPGA1 register set..... 16

 Status Register A1 16

 Configuration Register A1 17

 Configuration Register B1 19

 Configuration Register C1 20

 Release Register FPGA1 21

FPGA2 register set..... 22

 Status Register A2..... 22

 Configuration Register A2..... 23

Release Register FPGA2 24

APPENDIX A25

AER specification..... 25

 Multi sender implementation of AER. 25

 Point to point implementation of AER..... 27

APPENDIX B28

PCI-AER board standard pin connection..... 28

APPENDIX C29

PCI-AER Accessories..... 29

 AER standard converter..... 29

Table Index

Table 1: Allowed ARBITER configurations.....	4
Table 2: Word identification in the MONITOR FIFO.	5
Table 3: Allowed SEQUENCER commands.....	7
Table 4: Pointer Table commands on “one-to-many” mode	9
Table 5: Board Memory Space.....	14
Table 6: Register list of S5920 (Base Address 0).....	14
Table 7: Register list of the PCI-AER board (Base Address 1).....	15
Table 8: Configurations for the input channel of the Sequencer.....	18
Table 9: Configurations of the CLOCK AER frequency.....	19
Table 10: Configurations for the ARBITER.....	19
Table 11: Mapper Out configuration.....	23
Table 12: Configurations for AER Out.....	24
Table 13: AER Transaction timing.....	26
Table 14: AER Connector pin description.....	28

Figure Index

Figure 1: Illustration of the PCI-AER system architecture to test neuromorphic chip by AER bus.	2
Figure 2: Blocks scheme of the PCI-AER board.....	3
Figure 3: Monitor and Arbiter highlighted	5
Figure 4: Sequencer blocks.....	6
Figure 5: SEQUENCER FIFO word format.....	7
Figure 6: sequence of commands in the Sequencer FIFO	7
Figure 7: Block diagram of MAPPER.....	8
Figure 8: SRAM architecture.....	10
Figure 9: Blocks of Cable Adapter board	11
Figure 10: Layout of PCI – AER board.....	12
Figure 11: Layout of Adapter Cable board (external board).....	12
Figure 12: LED layout on External Board.....	13
Figure 13: Bit description of STATUS REGISTER A1	16
Figure 14: Bit description of CONFIGUATION REGISTER A1	18
Figure 15: Bit description of CONFIGUATION REGISTER B1	19
Figure 16: Bit description of CONFIGUATION REGISTER C1	20
Figure 17: Release register of FPGA1.....	21
Figure 18: Bit description of CONFIGUATION REGISTER A2.....	23
Figure 19: Release register of FPGA2.....	24
Figure 20: Multi sender AER transaction Specification.....	25
Figure 21: Multi sender AER Time Specification.....	25
Figure 22: AER transaction Specification	27
Figure 23: Top view of AER connector used on external board.....	28
Figure 24: Top view of AER standard converter.....	29

Warning

Be careful that PCI BUS of PC have 3.3V power supply line. The PCI-AER use 3.3 Volt for Xilinx FPGA and don't have voltage regulator inside.

Connection cable between PCI-AER and External Board is fragile. Be careful when manipulate it.

The PCI-AER board is an ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 volts, which readily accumulate on the human body and on test equipment, can discharge without detection. Therefore, proper ESD precautions are recommended to avoid any loss of functionality.

Overview

Figure 1 illustrates the main components of the system.

The system includes two boards: the PCI-AER Board is the PCI interface placed in the host computer and the Cable Adapter Board is an external board connected via a flat cable (68 pin) to the PCI-AER board and directly connected to the AER bus.

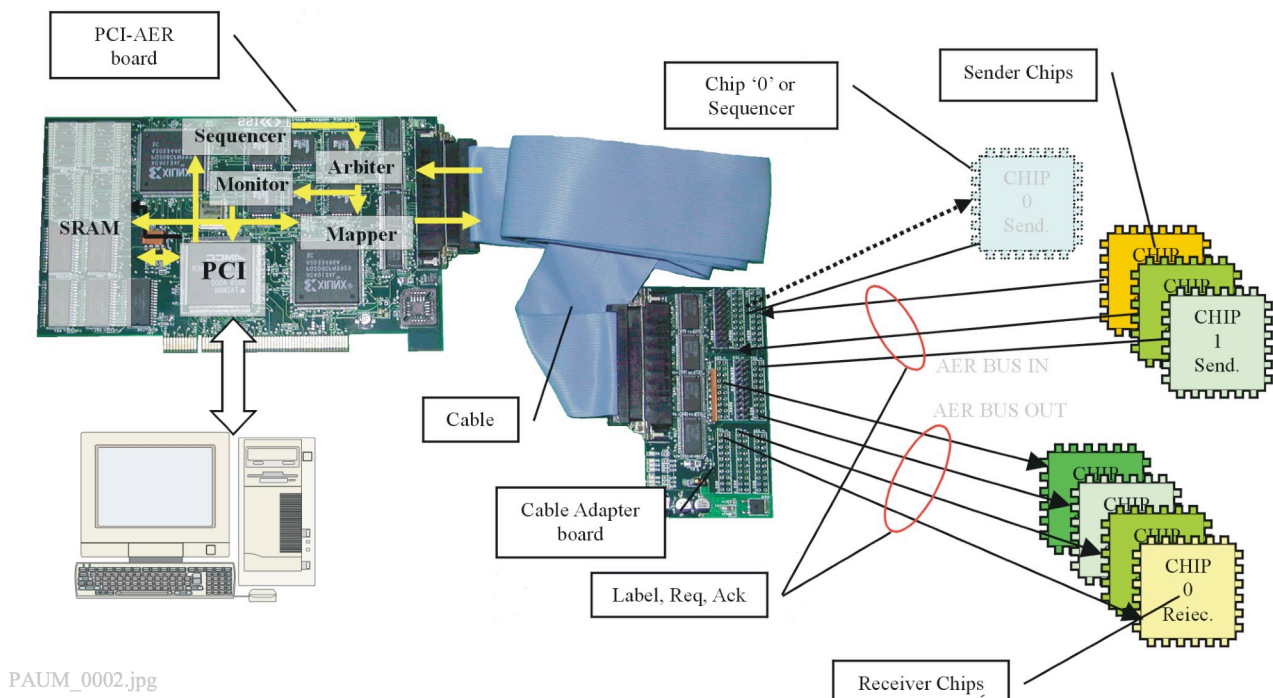
This configuration simplifies the placing and the interfacing of the system to test, allowing a friendly management of the connections.

The board in the host includes all the working components of the system PCI-AER and the buffers driving the internal communication by local buses.

The external board manages the signal transmission from the chips to the cable connecting the Cable Adapter Board to the PCI-AER board. It includes 9 AER connectors for chips connection.

The PCI-AER board has four main components:

- The **ARBITER** implements arbitration of events coming from different up to four sender chips.
- The **MONITOR**: allows to tap the transactions on the AER bus, to attach time information to them and to forward the joint information to a PC via PCI. Its available modes of operation allow acquiring every event from the AER bus or selecting events associated with a chosen subpopulation of neurons on the chips.
- The **MAPPER**: essentially implements the connectivity pattern between up to four sender chips, and up to four receiver chips.
- The **SEQUENCER** can be connected directly to the AER bus to emulate a neural chip, i.e. allowing the communication of a pre-determined flux of spikes from a simulation to subpopulations of neurons on the chips, to code the structure of 'external stimuli'.



PAUM_0002.jpg

Figure 1: Illustration of the PCI-AER system architecture to test neuromorphic chip by AER bus.

Description of the main PCI-AER board blocks

Figure 2 illustrates the logic plan of the PCI-AER board for the interface with AER bus. The highlighted parts contain the blocks in each one of the two FPGA XILINX.

Figure 2: Blocks scheme of the PCI-AER board

Block description

AER CLOCK

The AER CLOCK provides timing for all events on the AER bus: it provides a time base for the time label attached by the MONITOR to each spike; the SEQUENCER uses it to establish the time sequence of events put on the AER bus. The CLOCK period can be configured to assume one of the values 1,10,50,100 μ s.

TIME Counter

Time is a 32-bit counter which, once started, provides the absolute time elapsed since a reference start time, for each transaction on the AER bus. The 32-bit label is attached to each spike tapped by the MONITOR, and is stored in the FIFO along with the address. The SEQUENCER uses TIME to implement delays between spikes.

Because AER clock drives the TIME Counter, the maximum duration of a measurement depend on which AER clock is selected (See Table 1).

AER Clock	Approximate duration
1 μ s	1 hour 10 minutes
10 μ s	11 hours 55 minutes
50 μ s	59 hours 39 minutes
100 μ s	119 hours 18 minutes

Table 1: Maximum duration of measurement.

ARBITER

The PCI-AER board can simultaneously manage up to four chips. In the worst case four chips will therefore compete for the bus, and the ARBITER will be in charge of handling the arbitration of the accesses, in compliance with the "Multi sender AER standard" (see appendix A).

The ARBITER can operate in three configurations. In the first only one sender chip is assumed, with the associated 16-bit address. The second configuration can handle two sender chips, with a 15-bit address each, so that each chip can address 32768 neurons. The third configuration allows for four chips, with a 14-bit address each (16384 addressable neurons). In the second and the third case the ARBITER uses the two most significant bits to signal which AER channel (one of the available connectors) carried the spike.

Label width	N. Req	N. Ack	N. Sender Chips
16	1	1	1 Sender Chips
15	2	2	2 Sender Chips
14	4	4	4 Sender Chips

Table 2: Allowed ARBITER configurations.

MONITOR

The MONITOR taps any AER transaction on the ARBITER output. It takes the TIME value when the transaction is detected, and stores the time label and the address in the FIFO. Each event is stored in the FIFO as three successive words. The FIFO word is 18 bits: the two highest bits state the type of the information stored in the other 16 bits. The data structure is illustrated in Table 3: the address is 16 bits and the time label is 32 bits because it is compound by the TIME high and the TIME low word. This structure allows in particular to recover from any loss of synchrony in a FIFO reading operation.

	Bit17,16	Bit 15..0
AER Address	00	Event address label
TIME HI	01	TIME high (TH)
TIME LO	10	TIME low (TL)
Control/Error	11	Error or Control code

Table 3: Word identification in the MONITOR FIFO.

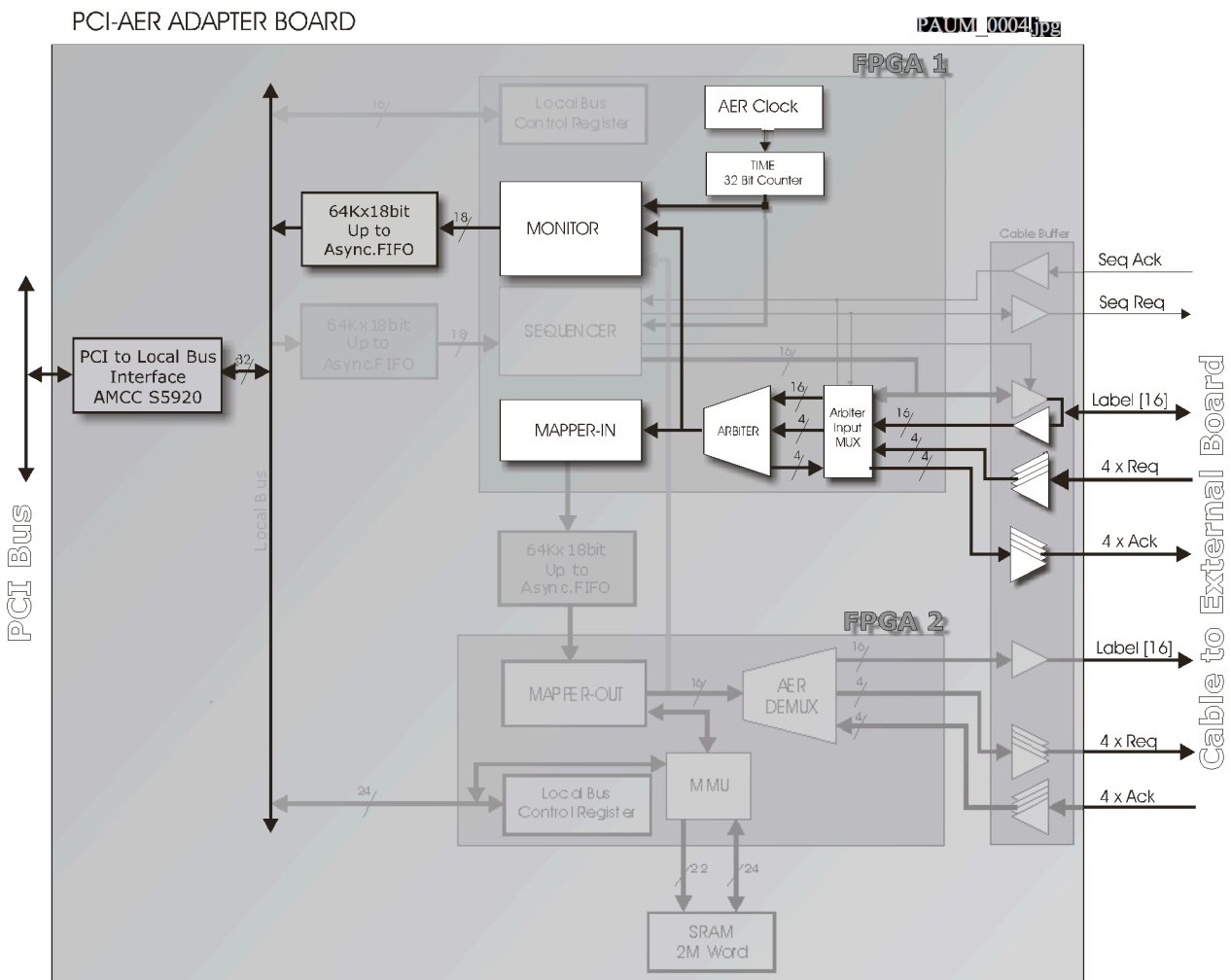


Figure 3: Monitor and Arbiter blocks highlighted.

SEQUENCER

It allows the software generation of AER transactions, thereby allowing, for example, the emulation of a 'virtual' AER-compliant chip, communicating with real chips. The SEQUENCER module (highlighted in figure) includes a FIFO, the SEQUENCER block that reads the FIFO and executes the commands and an interface with the system to test. The FIFO decouples the Sequencer block from the PCI.

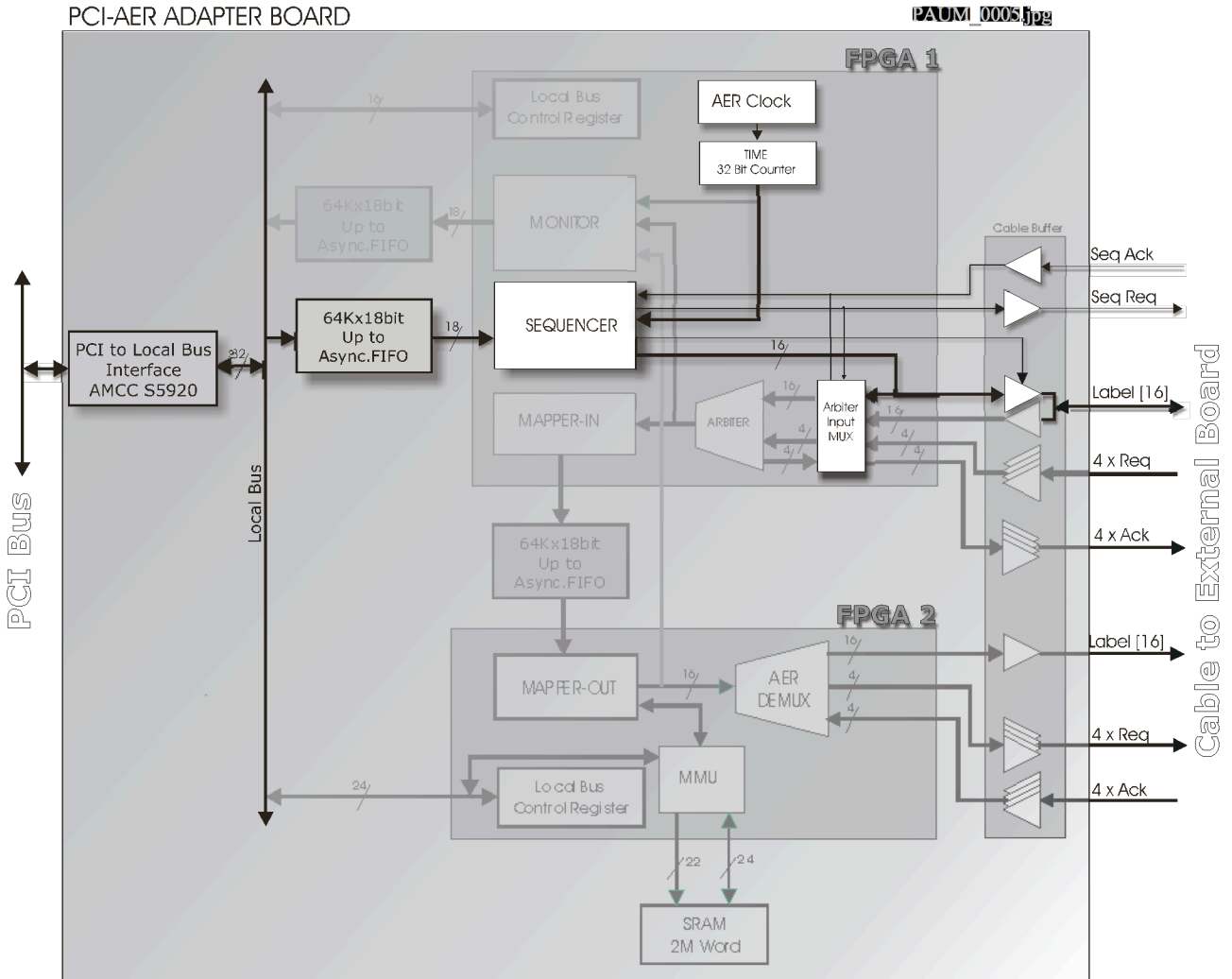


Figure 4: Sequencer blocks highlighted.

The SEQUENCER, on current version of the board, can't be used without ARBITER and MAPPER. In all cases the spike must pass through the ARBITER and MAPPER. The SEQUENCER connector on EXTERNAL board is not used. So the spikes can out from the board only through the RECEIVER connectors 1..4.

The SEQUENCER is handled exactly like a chip. So you must take an input channel to use SEQUENCER. Practically you can use only 3 chips when SEQUENCER is active. If SEQUENCER is inactive all channels are available for chips.

The data structure of the FIFO, and the SEQUENCER commands, are illustrated in Table 4.

Command	Bit17,16	Bit 15..0
End of sequence	00	Reserved
AER cycle	01	Event address label
Delay	10	Delay (AER CLOCK cycle)
Wait TIME	11	TIME for resuming execution (2 words in sequence)

Table 4: Allowed SEQUENCER commands.

The SEQUENCER checks the FIFO state and, if the latter is not empty, reads its content. Then, depending on the two most significant bits, it can either generate a transaction on the AER bus (a spike), or it can generate a relative time delay, expressed in AER clock cycles, or else an absolute time delay, on the basis of the TIME counter value (word format in Figure 5).

The transactions generated by the SEQUENCER can be presented on one of the channel, thus implementing the mentioned emulation of a virtual AER chip, possibly coexisting with real chips.



Figure 5: Sequencer FIFO word format.

The flux of the FIFO is one-way: the input comes from the PCI BUS, and the SEQUENCER block (on FPGA1) decodes the data on FIFO and producing and AER transaction or a delay, according to the commands in Table 4.

The FIFO is asynchronous (Cypress or IDT) with a maximum storage of 64Kx9Bit.

In Figure 6 there is a typical sequence of commands, with the format inside the FIFO. The HOST manages the data formatting and any error.

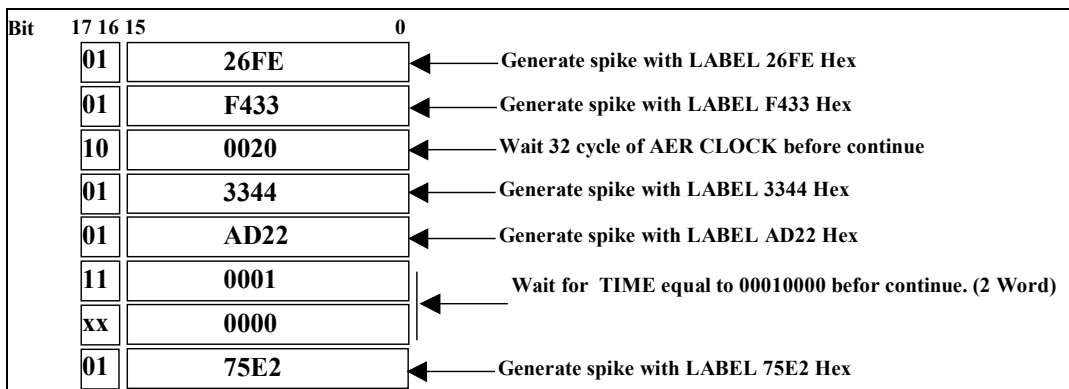


Figure 6: Example of commands in the Sequencer FIFO.

MAPPER

The MAPPER implements the mapping between AER IN and AER OUT addresses, thus allowing, for example, the definition of a connection matrix between neurons on different chips (Senders and Receivers). On the basis of the input LABEL placed on AER bus as input to the Mapper, it possible to send a spike in output for one or many neurons of the receiver chip.

The MAPPER can operate in three modes.

- 1) **“Pass-thru”** mode: the AER IN address is simply replicated on the AER OUT.
- 2) **“One-to-one”** mode: the AER IN address is used as a pointer into a look-up table, and the retrieved content will be the target address on the AER OUT.
- 3) **“One-to-many”** mode: the AER IN address must generate multiple events to be dispatched to different targets; this is achieved by a sequence of look-up table addressing a readings. Since the time each event takes to be put on the bus is shorter than the spike typical time width, usual (not too large) sequences of output events generated by a single AER IN event are to be considered as essentially simultaneous, from the point of view of the recipient chips.

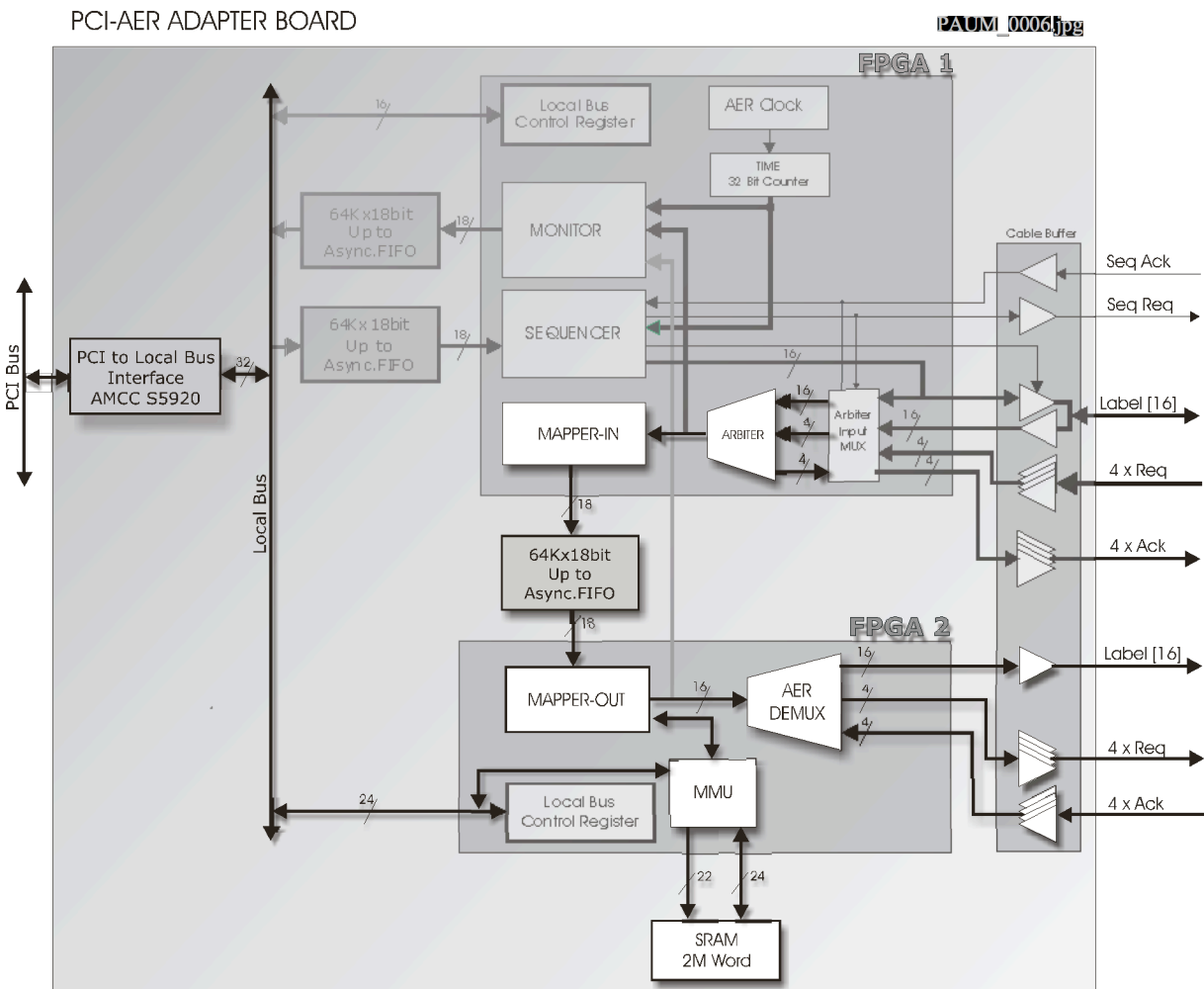


Figure 7: Mapper blocks highlighted.

The MAPPER includes four blocks distributed on both FPGA (Figure 7) :

- Mapper-In
- FIFO
- Mapper-Out
- MMU and SRAM

MAPPER-IN

Handles the flux of spikes at the ARBITER output of the sender chips, and forward them to the MAPPER FIFO. The MAPPER-IN generates an ACK signal to reply to a REQ signal from the ARBITER and complete an AER transaction, thus allowing to emulate the presence of a receiver chip; this is useful if one wants to test a sender chip, in the absence of a receiver chip.

FIFO

Is used in the modes “one-to-one” and “one-to-many” and it is needed for the second one. If a one-to-many mapping between the AER IN and AER OUT addresses, the MAPPER in fact must then sequentially scan the 'axon' look-up table, and generate an AER transaction at each step, at the risk of missing an incoming event, were it not for the FIFO. In this architecture the Sender BUS is completely decoupled from the Receiver BUS. If the event flux from the MAPPER in causes the saturation of the FIFO, an error flag arises for the MAPPER in registers.

MAPPER-OUT

Takes the spike address from the FIFO and performs the sequence of steps needed to generate the appropriate AER OUT events. In “one-to-one” mode it manages the address from the FIFO as a pointer to a look-up table word containing the address of a receiving neuron; in “one-to-many” mode the address from the FIFO is the start pointer for the look-up table scan, stopped when the MAPPER out find an END label. The look-up table data structure is illustrated in and details in “MMU and SRAM architecture” section.

MMU and SRAM architecture.

The memory management unit handles the SRAM accesses. The SRAM can be written and read from the PCI, but it is usually accessed by the MAPPER-OUT since it hosts the look-up table. The MMU manages the access conflicts and configures the data paths for each type of access. (*Is possible access to the SRAM from PCI only if MAPPER-OUT is disable see Configuration Register A2.*) If you try to access the SRAM when MAPPER-OUT is enabled, no errors are generated but, in case of writing access, the data are lost. In reading access the data value is unpredictable.

The SRAM amounts to 2 MWords, organized in four 512 KWords banks. The first one has a 24 bit word, the others have a 16 bit word (see Figure 8). This architecture is needed to use the SRAM in the implementation of two different tables: the *Pointer Table* and the *Spike Table*.

The *Pointer Table* takes up the first 64KW of SRAM and contains a table of 21Bit pointers addressed to a location inside the *Spike Table*.

Pointer Table	Bit 23,22	Bit 21	Bit 20..0
Pnt. Start Table	00	Not Used	Pointer to start word on the Spike Table (21 bit)
Not Used	01	Not Used	Not Used
Not Used	10	Not Used	Not Used
Direct Spike Generation	11	Not Used	Direct Spike generation (only Bit 15..0 is used)

Table 5: *Pointer Table* commands on “one-to-many” mode .

The *Pointer Table* works as a look-up table: the AER Label coming from the working AER entry is a pointer to a first word of an *Spike Table* location, containing an other table of AER Labels

representing the addresses of the receiving neurones. The look-up table scan ends when the END LABEL is reached. Because the END LABEL is represented by the value FFFF Hex, the maximum number of receiver neurones is $2^{16}-1$.

In “one-to-many” mode is also possible generate directly a spike without scan *Spike Table*. Using direct spike generation save time in terms of clock cycle. Is enough set the two higher bit of *Pointer Table* to ‘11’ (see Table 5)

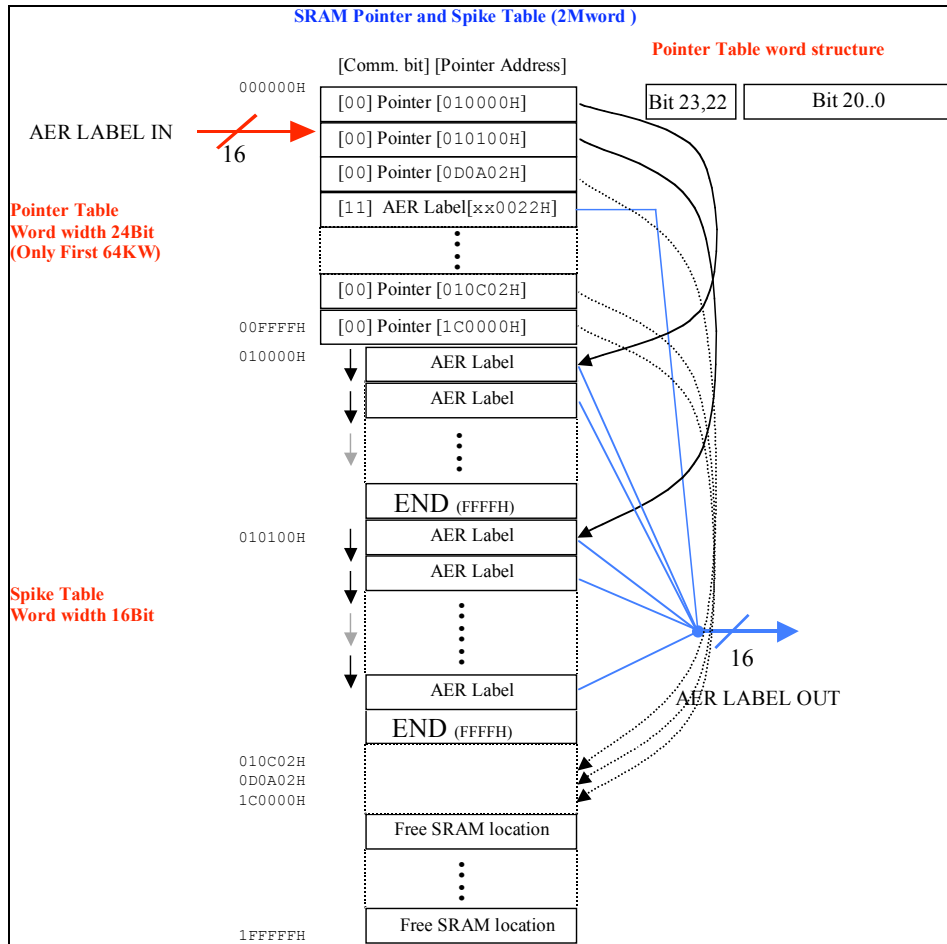


Figure 8: Mapper SRAM architecture.

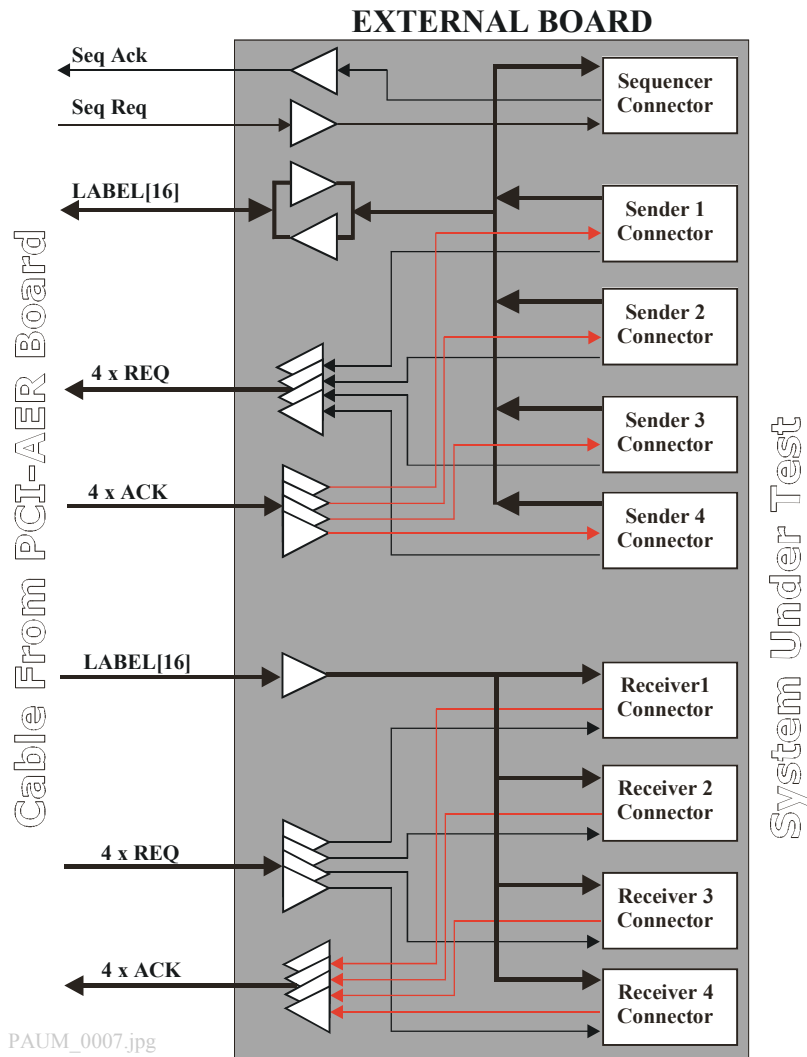
AER DEMUX

It is the ARBITER counterpart on the output side. It can be configured in such a way to handle 1 to 4 receiver chips, too. The two most significant bits of the output address determine the channel hosting the new AER transaction, given the AER DEMUX configuration (see Table 13).

Description of the Cable Adapter board blocks.

The Cable Adapter board decouples electrically the system under test from the PCI-AER board and the cable. This solution assures the integrity of the signals also with long cables.

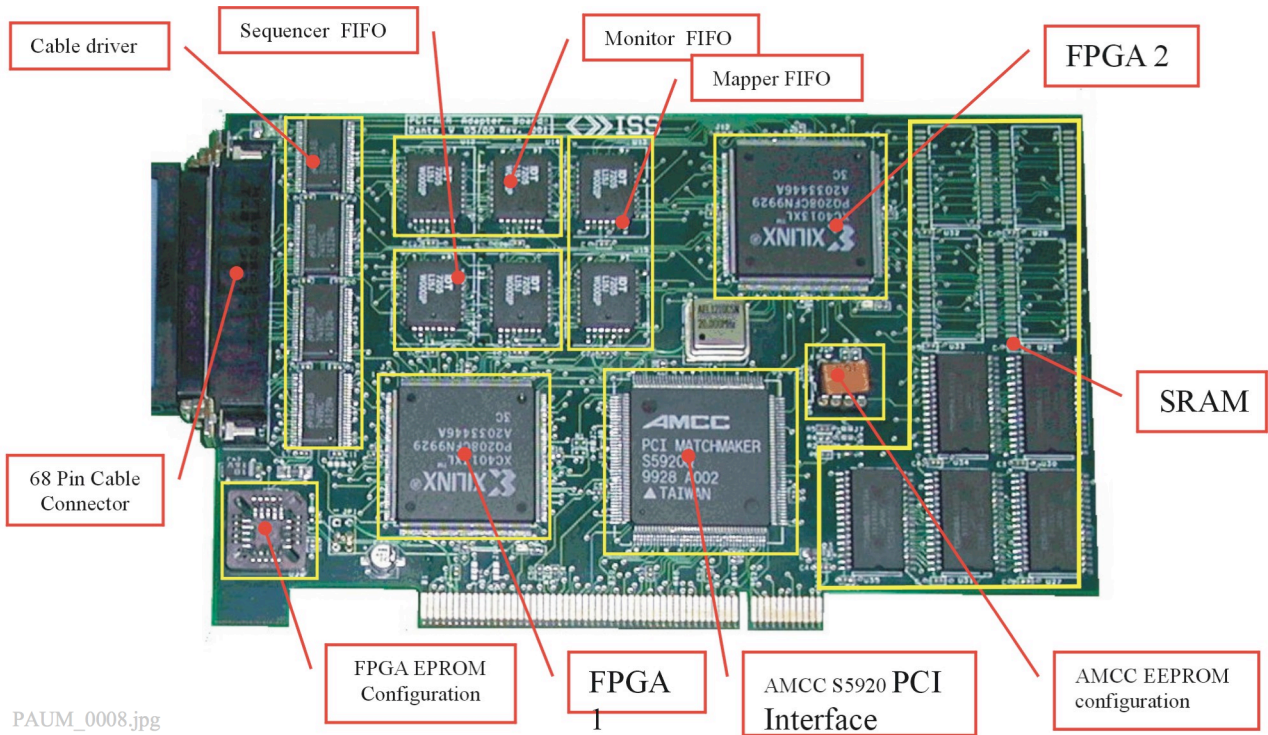
The power supply can be +5V or 3.3V. The External Board can be powered directly from PCI-AER board (5V) or through external power supply. The External Board have a voltage regulator inside. The different power supply source is provided for neuromorphic chips with different power supply.



PAUM_0007.jpg

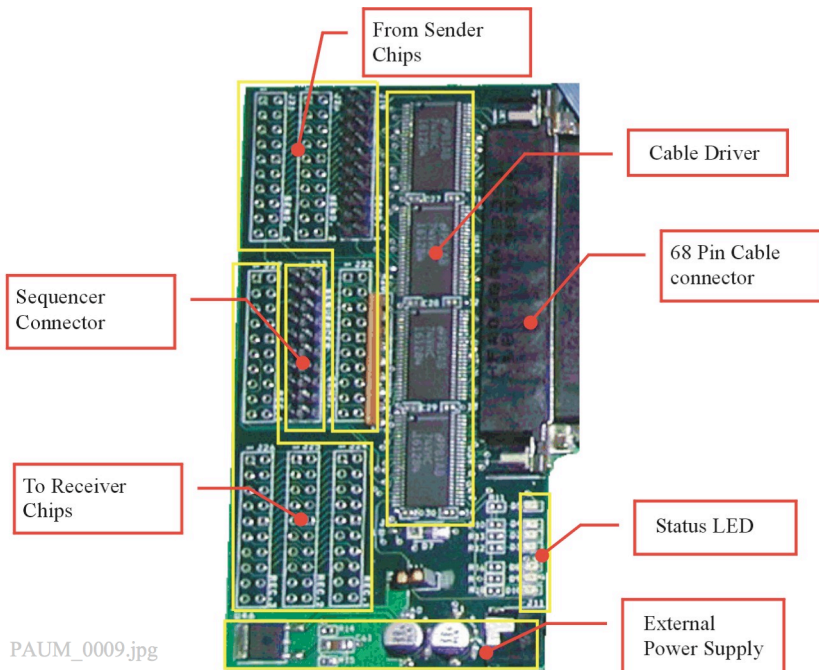
Figure 9: Blocks of Cable Adapter board .

PCI-AER system Board Layout



PAUM_0008.jpg

Figure 10: Layout of PCI – AER board



PAUM_0009.jpg

Figure 11: Layout of Adapter Cable board (External board)

At the moment the Sequencer connector is not used.

External Board Led description

The external board have five LED, three yellow and four red. Near 68 pin connector there is Power ON LED. After there are FIFO Full alert LED. The yellow LED are used to indicate status of primary block of PCI-AER Board (Monitor, Mapper and Sequencer). If LED is lighted, associate block is active.

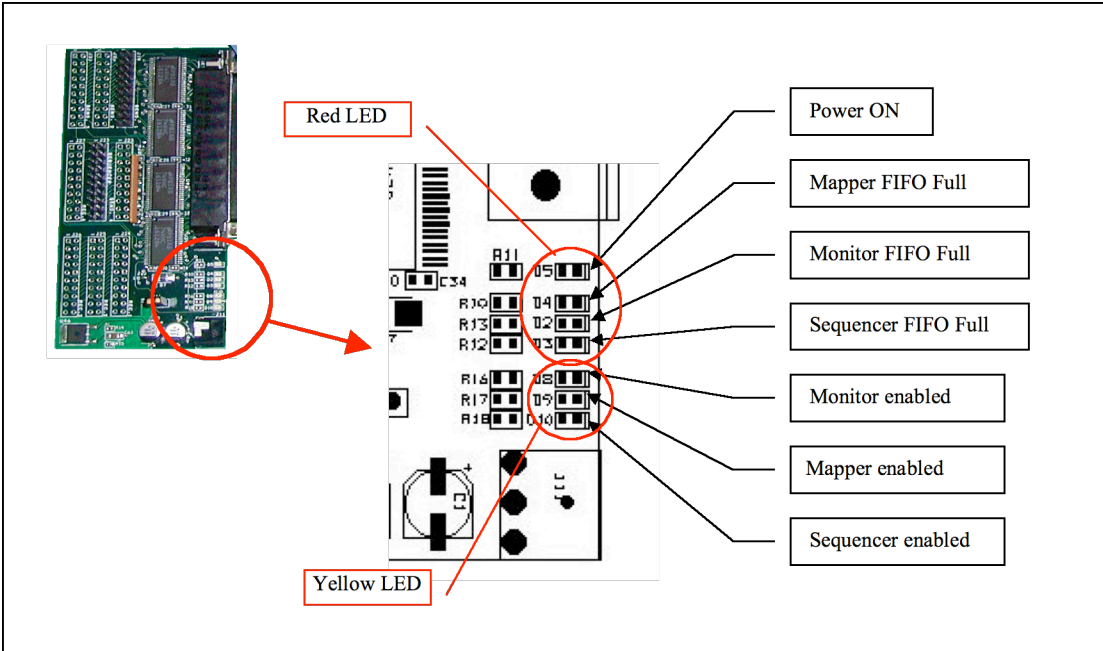


Figure 12: LED layout on External Board

PCI Configuration Space for PCI-AER Adapter Board

At present the PCI SIG (PCI Special Interest Group) has not assigned the IDs necessary for the identification of the PCI-AER board. Therefore for the moment we use the numbers provided by AMCC.

VENDOR ID

Value: 10E8h
PCI Offset: 00h-01h

DEVICE ID

Value: 5920h
PCI Offset: 02h-03h

Base Address Register and Memory Region¹

The PCI-AER board uses 4 different memory location. All the registers of the board are located in the memory space (no resources are located in I/O space).

Register	Type	Dimension	Description	Wait State
BADDR 0	Memory	128 Byte	S5920 PCI Register	0
BADDR 1	Memory	128 Byte	Board Register	2¹
BADDR 2	Memory	Up to 64 KDW (32Bit)	FIFO Area	1¹
BADDR 3	Memory	2 MDW (32Bit)	SRAM Area	2²

Table 6: Board Memory Space

Base Address 0. S5920 Register²

The registers of S5920 accessible from PCI and necessary for the right configuration of the board (Operation Register) are located in this region of Memory space.

The S5920 works as Pass-Thru in Active Mode for the board control. Therefore we don't need to consider the registers relating to the other modalities of S5920.

Register	Add. Offset	Description	
OMB	0Ch	Outgoing Mailbox Register	Not Used
IMB	1Ch	Incoming Mailbox Register	Not Used
MBEF	34h	Mailbox Empty/Full Status Register	Not Used
INTCSR	38h	Interrupt Control/Status Register	Used
RCR	3Ch	Reset Control Register	Used
PTCR	60h	Pass-Thru Configuration Register	Used

Table 7: Registers list of S5920 (Base Address 0).

¹ During the test stage it is necessary a certain number of wait states for the routing optimization of FPGAs. Then it will be possible to decrease the waiting time. This setting depends from FPGA family .

² Because SRAM is 70nS we need of 2 wait states for the PCI access.

Base Address 1. Registers of PCI-AER board.

The registers from 00H to 2CH are 16 bits and from 30H to 3CH are 24 bits.

In Table 8 there is the list of the PCI-AER board register.

		OffSet	Description	Implemented on
Status Register A1	RO	00H	Status Register A1	FPGA1
Configuration Reg. A1	RW	04H	Configuration Register A1	FPGA1
Configuration Reg. B1	RW	08H	Configuration Register B1	FPGA1
Configuration Reg. C1	RW	0CH	Configuration Register C1	FPGA1
		10H		Not impl.
		14H		Not impl.
		18H		Not impl.
		1CH		Not impl.
Release Register 1	RO	20H	Release Register of FPGA1	FPGA1
		24H		Not impl.
		28H		Not impl.
		2CH		
Status Register A2	RO	30H	Status Register A2	FPGA2
Configuration Reg. A2	RW	34H	Configuration Register A2	FPGA2
		38H		
Release Register 2	RO	3CH	Release Register of FPGA2	FPGA2

Table 8: Registers list of the PCI-AER board (Base Address 1)

Base Address 2. Access to the FIFO

Through this Base Address it is possible to access the Monitor FIFO and the Sequencer FIFO. As the information flux in the FIFOs is one-way, the PCI accesses the Monitor FIFO only in reading mode and the Sequencer FIFO in writing mode. Therefore the contemporary access to both FIFO through the same base address does not generate conflicts.

The maximum storage capacity of the FIFO used is 64K x 18Bit (access by 32 bits words).

Base Address 3. Access to MAPPER SRAM.

Through this Base Address it is possible to access to the MAPPER SRAM. The storage width is 2MW: the first 512KW are 24bit and the other 1,5MW are 16 bit. Only the single access is possible (no burst mode accesses is allowed). **The SRAM access from PCI is denied if the MAPPER OUT is active.** (See *Configuration Register A2* section).

Register architecture of PCI-AER board.

FPGA1 register set

Status Register A1

The bit configuration of the Status Register A1 is in Figure 13. The register bits are read only.

- **MON FE (MonFE bit)** (*Monitor FIFO EMPTY flag*) is an active low bit. It is high when at least a word is in the MONITOR FIFO (FIFO not empty).
- **MON FH (MonFH bit)** (*Monitor FIFO HALF FULL flag*) is an active low bit. It is set to low after half of the FIFO memory is filled and will remain set until the number of data on MONITOR FIFO is more than or equal to one half of the total memory of the device (4096 word in our case).
- **MON FF (MonFF bit)** (*Monitor FIFO FULL flag*) is an active low bit. It is low when the MONITOR FIFO is full. Otherwise is high.
- **SEQ FE (SeqFE bit)** (*Sequencer FIFO Empty flag*) is active low. The bit is high when at least a word is in the SEQUENCER FIFO (FIFO not empty).
- **SEQ FH (SeqFH bit)** (*Sequencer FIFO Half Full flag*) is an active low bit. It is set to low after half of the FIFO memory is filled and will remain set until the number of data on SEQUENCER FIFO is more than or equal to one half of the total memory of the device (4096 word in our case).
- **SEQ FF (SeqFF bit)** (*Sequencer FIFO Full flag*) is an active low bit. It is low when the MONITOR FIFO is full. Otherwise is high.
- **MAP FF (MapFF bit)** (*Mapper FIFO Full flag*) is an active low bit. The bit will set to low when the MAPPER FIFO is full. Otherwise is high.
- **MAP FE (MapFE bit)** (*Mapper FIFO Empty flag*) is an active low bit. The bit will set to low when the MAPPER FIFO is empty. Otherwise is high.

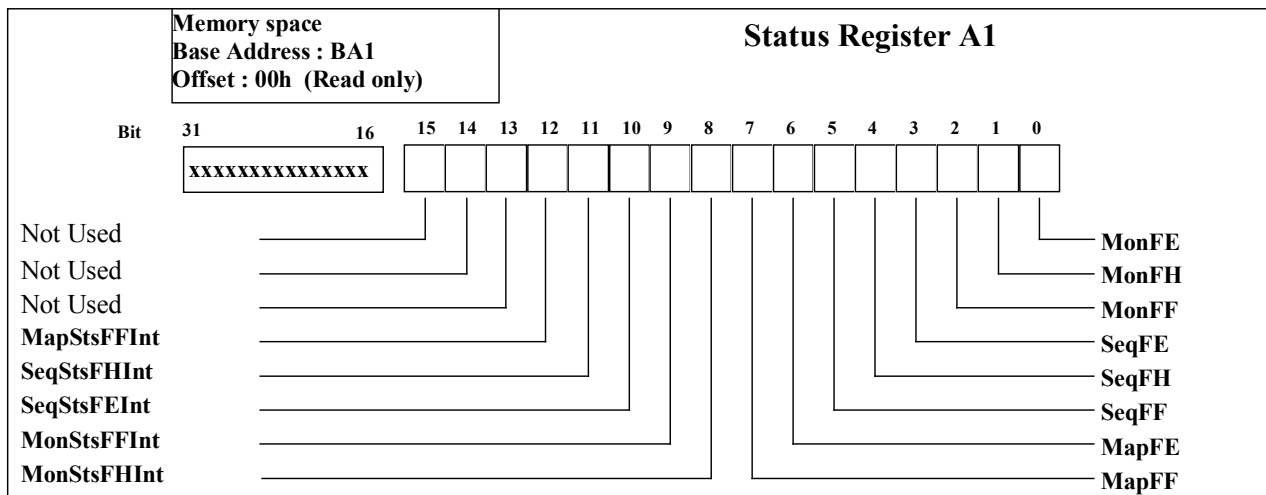


Figure 13: Bit description of STATUS REGISTER A1

On the MAPPER FIFO we don't have FH because the MAPPER FIFO is filled and flushed automatically by MAPPER-IN and MAPPER-OUT state machine. From software is only possible to reset this FIFO.

For more information about behaviour of FIFOs flags signals refer to IDT7205 FIFO data-sheet.

- **MON HF Int (MonStsFHInt bit)** is high when a FIFO HALF FULL interrupt is generated by the Monitor. The FIFO HALF FULL interrupt is generated when MONITOR FIFO becomes half full. This interrupt is useful to handle emptying of FIFO by the data block and not word by word.
- **MON FF Int (MonStsFFInt bit)** is high when an FIFO FULL interrupt is generated by the Monitor. The FIFO FULL interrupt indicates a probably error condition because when MONITOR FIFO is full further incoming spikes are lost.
- **SEQ FE Int (SeqStsFEInt bit)** is high when an FIFO EMPTY interrupt is generated by the Sequencer. The SEQUENCER FIFO EMPTY interrupt is generated when SEQUENCER FIFO becomes empty. This is a probably error condition because the data flow from sequencer can be discontinued.
- **SEQ HF Int (SeqStsFHInt bit)** is high when an FIFO HALF FULL interrupt is generated by the Sequencer. The SEQUENCER FIFO HALF FULL interrupt is generated when the number of the words on FIFO becomes less than one half of the total memory of the device. This is useful to maintain constant data flow from SEQUENCER .
- **MAP FF Int (MapStsFFInt bit)** is high when an FIFO FULL interrupt is generated by the Mapper. The FIFO FULL interrupt is generated when MAPPER FIFO become full. This is probably an error condition because it indicates that the data flow from MAPPER-IN to MAPPER-OUT has stopped with consequent spike loss.

The interrupt flags can be cleared through a write operation at *Status Register A1* address. The value written at this address will be used as a MASK to select which Interrupt Flags to clear. The value is not stored. If a bit (bit 8 to bit 12) in the written value is 1 and the corresponding Interrupt Flag is active, the flag will be cleared, otherwise the flags do not change.

Configuration Register A1

- **EnMON** (Enable Monitor) starts the MONITOR. This read/write bit starts or stops the AER transaction acquisition. If 1 the MONITOR is stopped.
- **MonChSel[3:0]** masks the input channels of the Monitor. For each channel it is possible to enable or disable the AER transactions management. If the bit of a channel is 0, the transactions for that channel are managed by the Monitor and stored in the FIFO, if 1, the Monitor ignores that channel. This feature allows to save space in the FIFO.
- **EnTIMELbl**, if 0, allows to save the spike address with the relative TIME label rather than only the spike address.
- **MonFMR**(Monitor FIFO Master Reset) resets the Monitor FIFO. Usually it is 1 to disable the reset condition, but if we need to reset the FIFO it must be 0 for at least 10ms.
- **MonFRT** (Monitor FIFO ReTrasmit) is the RETRASMITE signal of the Monitor FIFO. This read/write bit is for future expansion of the board, it is not used in the current version and is usually fixed to 1. For a deeper explanation of the functions of this bit, it is necessary to read the FIFO datasheet in the section concern the RT signal.

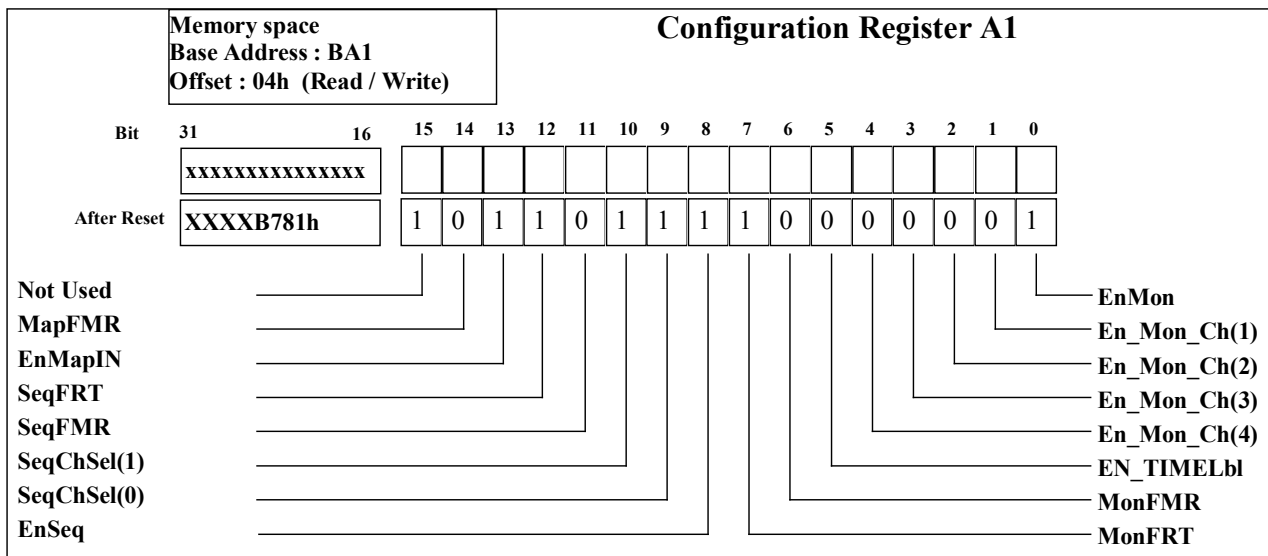


Figure 14: Bit description of CONFIGURATION REGISTER A1

- **EnSEQ** (Enable Sequencer) is a read/write bit that enables the Sequencer function .
- **SeqChSel[1:0]** (Sequencer Channel Select) are read/write bits that select the Arbiter channel where the Sequencer will send the spike information. The selected transmission channel is disconnected from the external AER channels. All the configurations for these bits are in Table 9.

SeqChSel[1:0]	Configuration	Description
00	Channel 0	Sequencer connected to 0 channel of the Arbiter.
01	Channel 1	Sequencer connected to 1 channel of the Arbiter.
10	Channel 2	Sequencer connected to 2 channel of the Arbiter.
11	Channel 3	Sequencer connected to 3 channel of the Arbiter.

Table 9: Configurations for the input channel of the Sequencer.

- **SeqFMR** (Sequencer FIFO Master Reset) is a read/write bit that resets the FIFO of the Sequencer.
- **SeqFRT** (Sequencer FIFO ReTrasmit) is the RETRASMITE signal of the Sequencer FIFO. This read/write bit is for future expansion of the board, it is not used in the current version and is usually fixed to 1. For a deeper explanation of the functions of this bit, it is necessary to read the FIFO datasheet in the section concern the RT signal.
- **EnMapperIN** (Enable Mapper IN) (FPGA1) is a read/write bit to enable the MAPPER in module (if 0), then the transmission of an ACK signal to the REQ signal from the ARBITER to complete an AER transaction.
- **MapFMR** (Mapper FIFO Master Reset), if 0, resets the Mapper FIFO.

Configuration Register B1

- **EnTIME** (Enable TIME counter) is a read/write bit that enables/disables the TIME counter. If bit is 0 TIME Counter is enabled
- **RstTIME** (Reset TIME counter) is a read/write bit that, if 0, the value of the TIME counter is forced to 0.

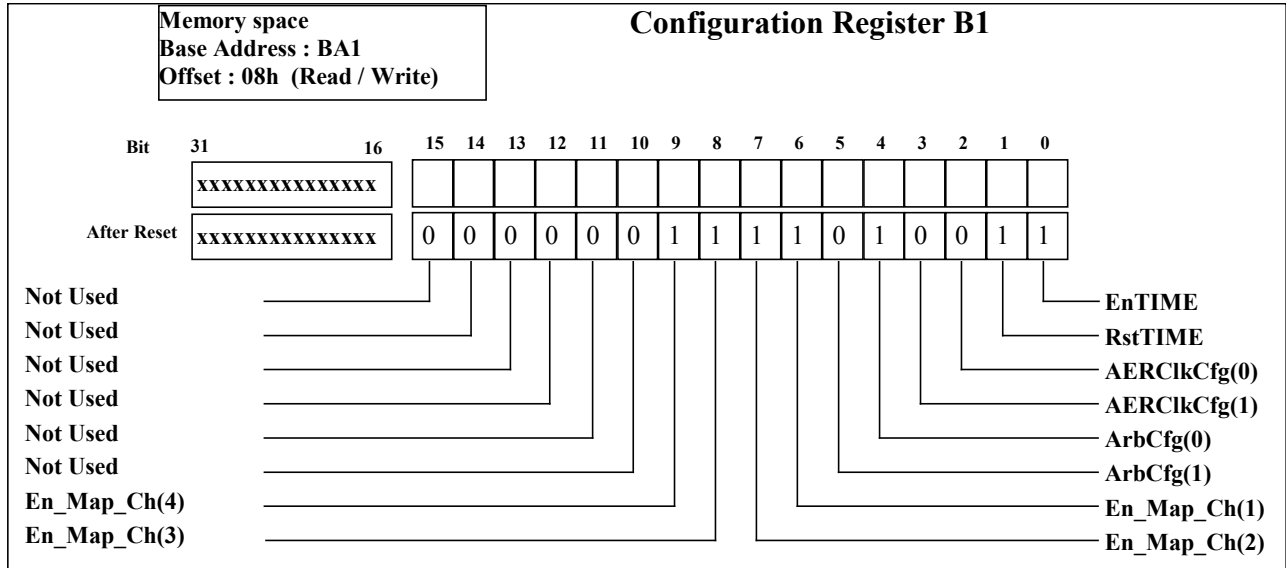


Figure 15: Bit description of CONFIGUARTION REGISTER B1

- **AERClkCfg[1:0]** (AER Clock Configuration) are a read/write bits that select the CLOCK frequency used in the AER management as illustrated in Table 10

AERClkCfg[1:0]	Clock Period
00	1uS
01	10uS
10	50uS
11	100uS

Table 10: AER CLOCK frequency configurations.

- **ARBCFG[1:0]** (Arbiter Configuration) are read/write bits to set the operation mode of the Arbiter. All the configurations are in Table 11.

ARBCFG[1:0]	Label width	N. Req	N. Ack	
00	Disabled			Not Used
01	16	1	1	1 Sender Chips
10	15	2	2	2 Sender Chips
11	14	4	4	4 Sender Chips

Table 11: ARBITER Configurations.

- **En_Map_Ch[4..1]** mask the input channels of the Mapper. If a bit is 0, the corresponding channel is managed by the Mapper, otherwise it is ignored.

Configuration Register C1

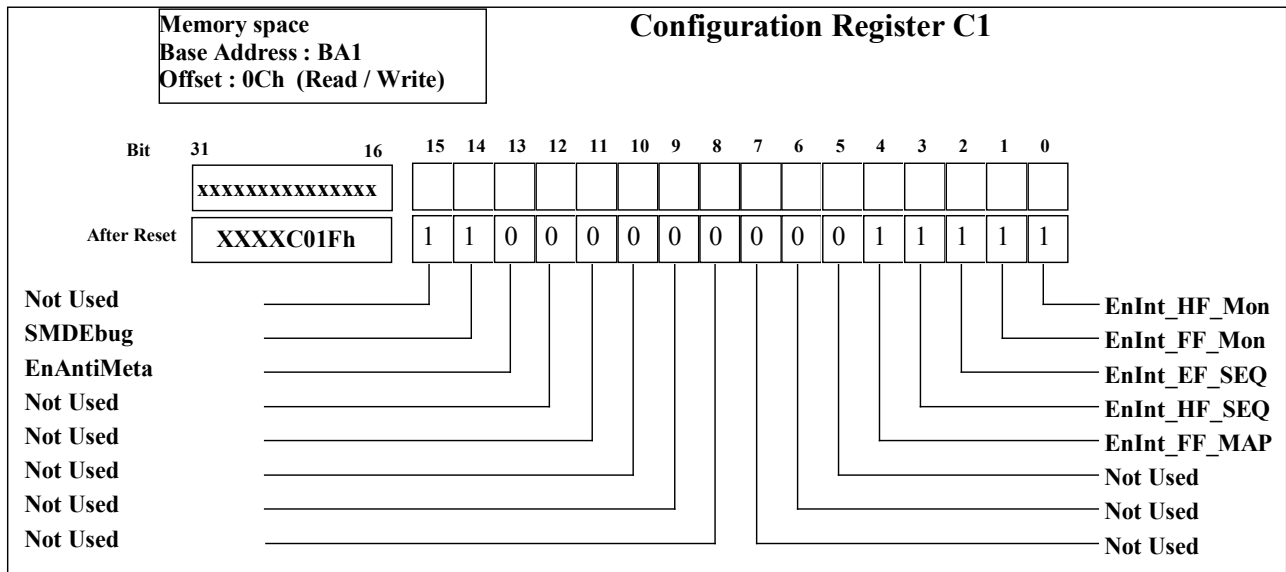


Figure 16: CONFIGUARTION REGISTER C1 bit description

- **MonFHInt**, if 0, enables the generation of the interrupt when the FIFO of the Monitor is half full. This feature is implemented to communicate to the driver that the Monitor FIFO becomes half full.
- **MonFFInt**, if 0, enables the generation of the interrupt when the FIFO of the Monitor is full. This state can generate an error condition because we can loose data.
- **SeqHFInt**, if 0, enables the generation of the interrupt when the FIFO of the Sequencer is half empty. This feature is implemented to communicate to the driver and/or the software that it's possible to transfer new data to the FIFO (at least half FIFO).
- **SeqEFInt**, if 0, enables the generation of the interrupt when the FIFO of the Sequencer is empty. This information can be managed by the driver as an error condition or as the availability of the FIFO for new data.
- **MapFFInt**, if 0, enables the generation of the interrupt when the FIFO of the Mapper is full. This information must be managed by the driver as an error condition because in this case the further incoming spikes are lost.
- **SMDEbug**, If 0 and if MAPPER Out goes through recovery state, start flashing LED. If =1 no special action.
- **EnAntiMeta** if 0 the incoming spike will not synchronized, if 1 the incoming spikes pass thru a double level synchronization block.

Release Register FPGA1

This register stores the release of the FPGA1 firmware and is read only. Its value is fixed at hardware level. The information is stored in the lower 16 bits of registers: the less significant 8 bits store the Revision number and the other bits store the Version number. The higher 16 bit are irrelevant.

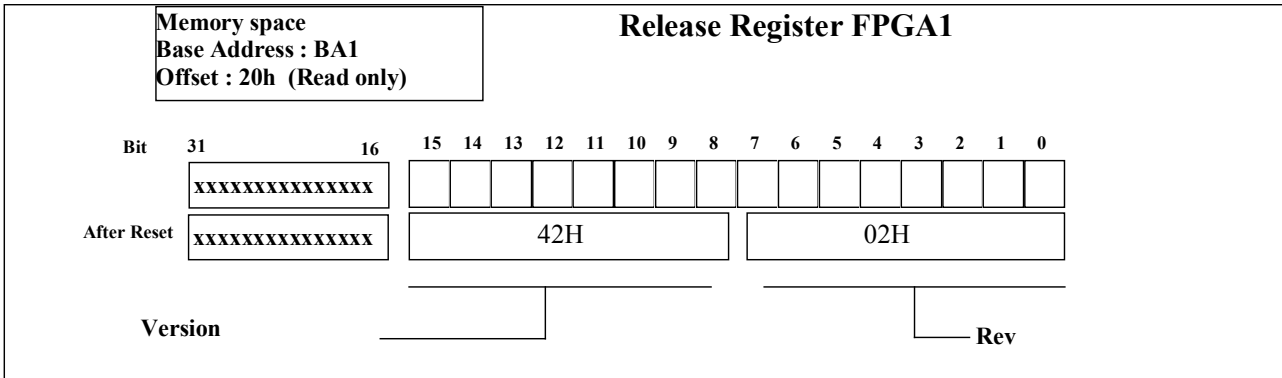


Figure 17: FPGA1 release register.

FPGA2 register set.

Status Register A2

- **MMU_Busy** (Memory Manager Unit Busy) (FPGA 2) It's read only bits. This bit is active low and is 1 only when the MAPPER-OUT is disabled and State machine of MAPPER-OUT is on idle status. This means that MAPPER-OUT have completed last access to the SRAM, after having been disabled. In all others cases is 0. Remember, the SRAM is accessible by PCI only when the MAPPER-OUT is disabled and have finished his last access. Therefore, before to write or to read the SRAM, is needed to check this bit to be sure of the success of operation.

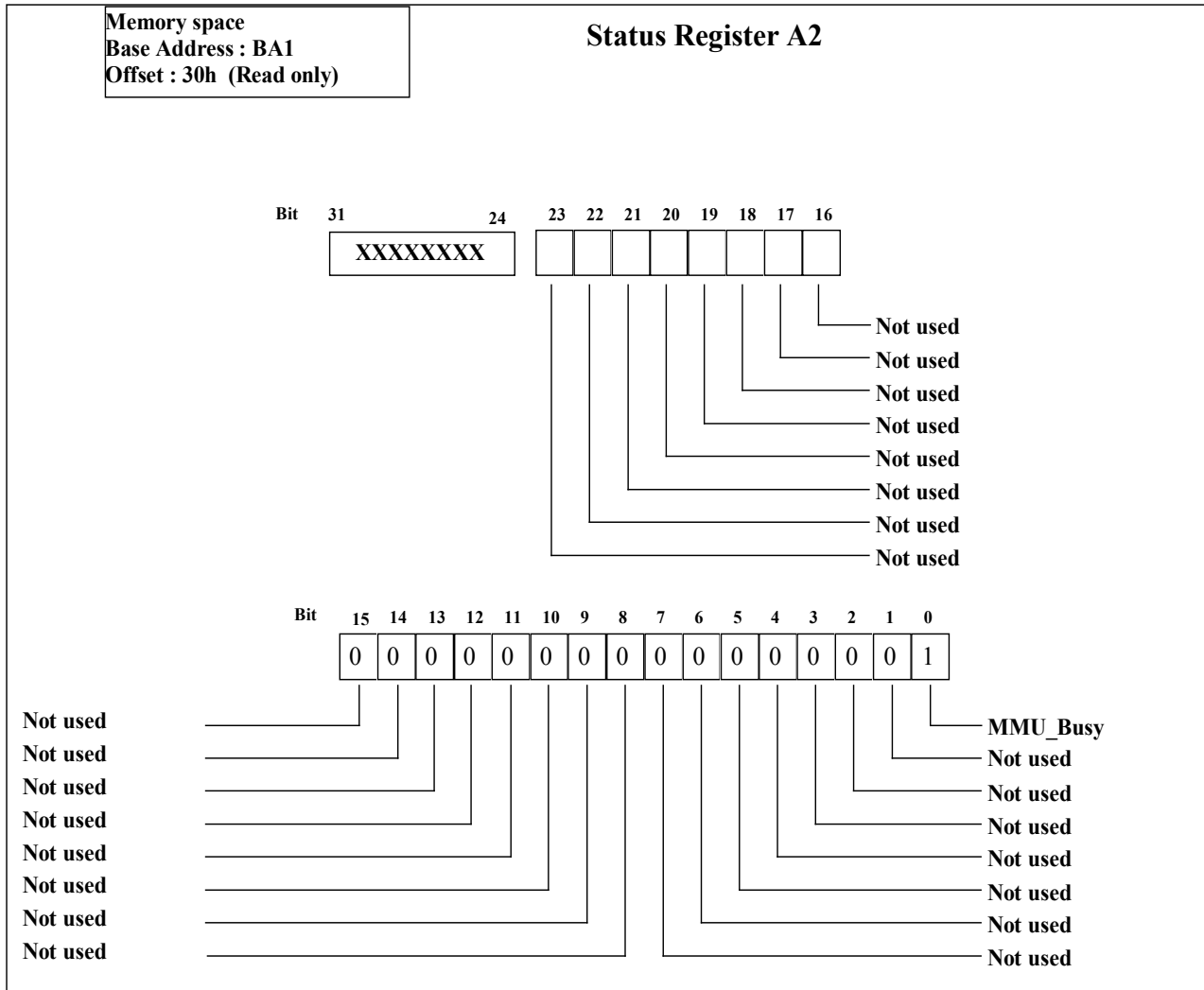


Figure 18: Statutur REGISTER A2 bit description.

Configuration Register A2

- **CfgMapperOut[1:0]** (Configuration Mapper OUT) (FPGA 2) are read/write bits to configure the del MapperOut mode. See Table 12 .

CfgMapperOut[1:0]	Function	Description
00	Not Used	Not Used.
01	Pass-thru	The incoming Spike address is put directly on output without change
10	One to One mode	The incoming spike address is converted and put on the output.
11	One-to-many Mode	Send incoming spike to many different neuron address on the output.

Table 12: MAPPER OUT configurations.

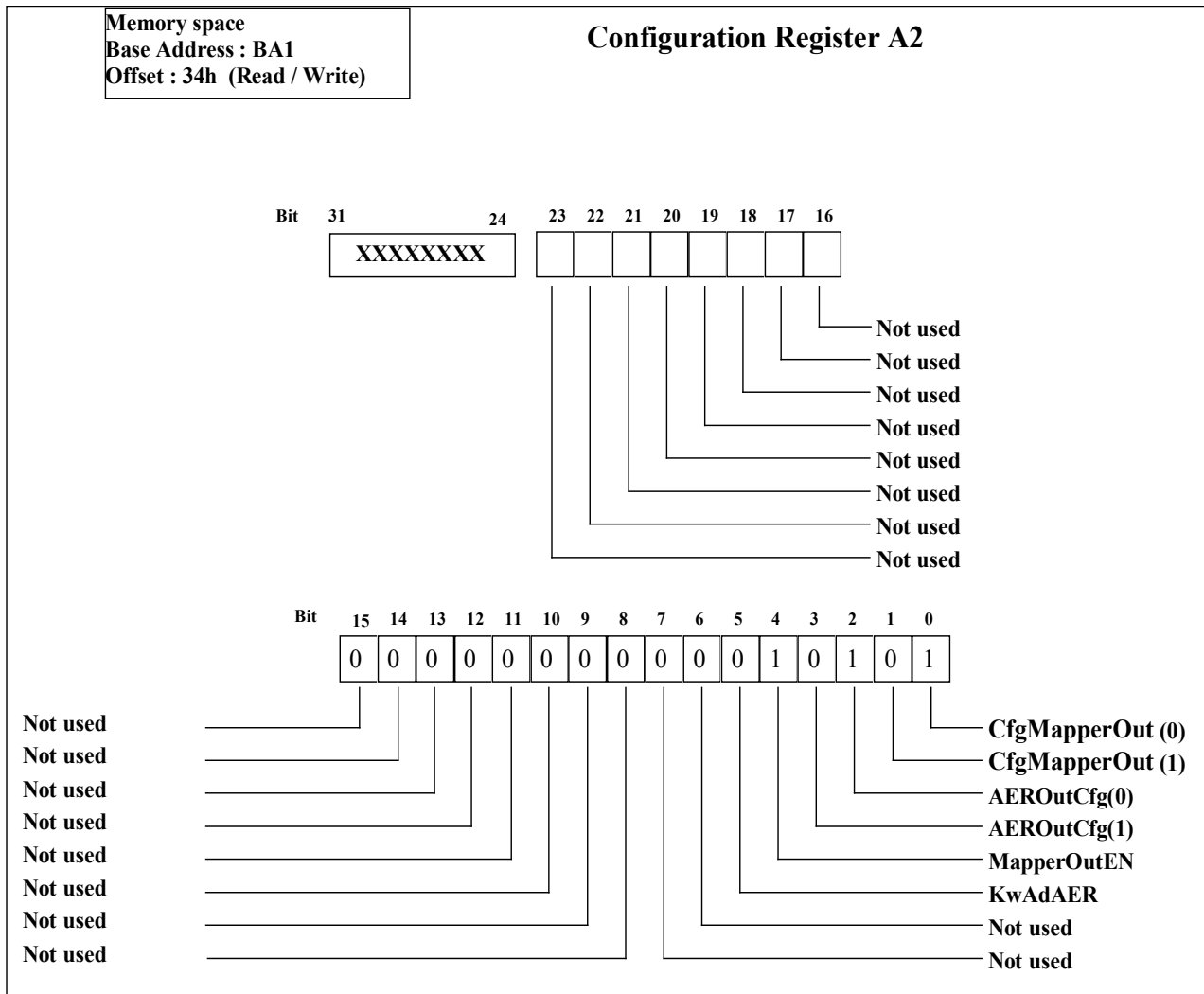


Figure 19: CONFIGUATION REGISTER A2 bit description.

- **AEROutCfg[1:0]** (AER Out Configuration) (FPGA2) are the read/write bits that set the output configuration for the receiver chip. All the configurations are in Table 13 .

AEROutCfg[1:0]	Addr. Bit	Handshake.	Description
00	Disabled		Not Used
01	16	1	1 Receiver Chips
10	15	2	2 Receiver Chips
11	14	4	4 Receiver Chips

Table 13: AER OUT Configurations.

- **MapperEn** if 0 , Mapper-OUT take spike from FIFO Mapper and generate AER transaction un Output stage of PCI-AER board.
- **KwAdAER** selects the output configuration of the PCI-AER board. Is possible to choose if use “Point to Point” implementation of AER or “Multi Sender” AER implementation. 'If bit is 1, output will be set to use "Point to Point" AER otherwise "Multi Sender" AER is used'.

Release Register FPGA2

This register stores the release of the FPGA2 firmware and is read only. Its value is fixed at hardware level. The information is stored in the lower 16 bits of registers: the less significant 8 bits store the Revision number and the other bits store the Version number. The higher 16 bit are irrelevant.

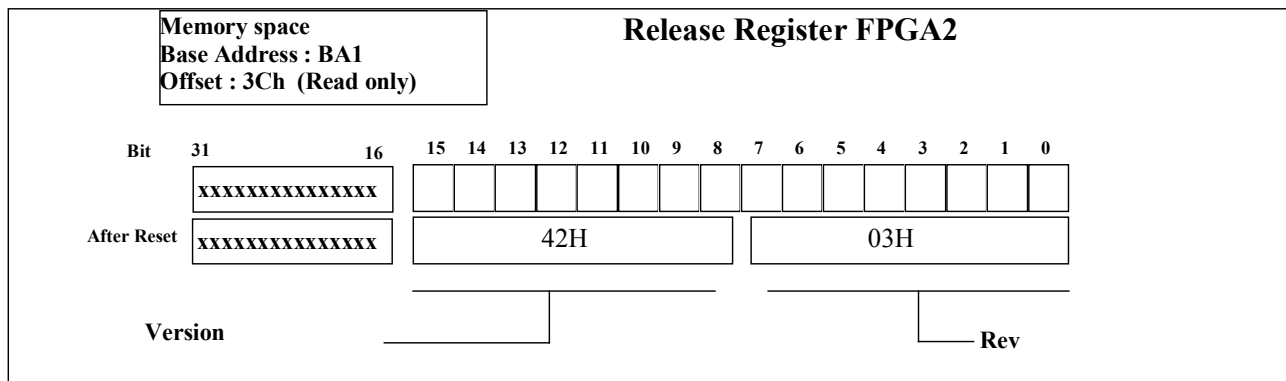
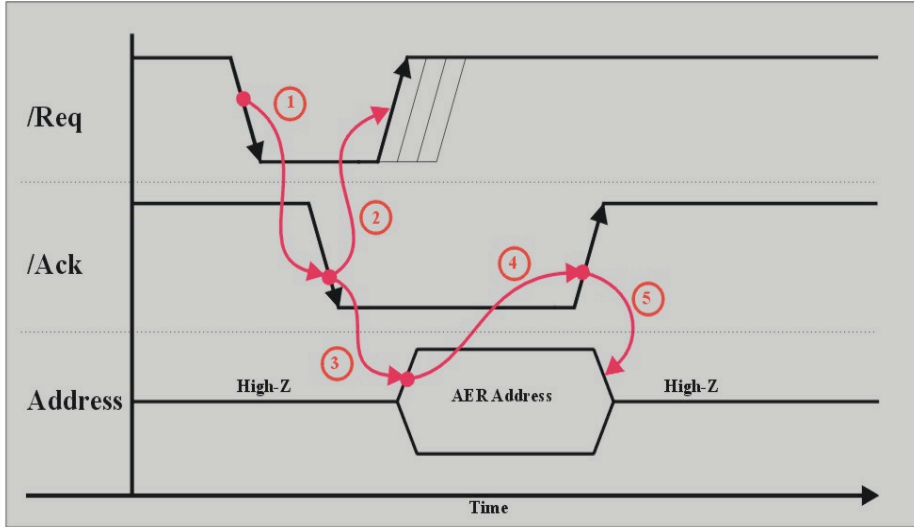


Figure 20: FPGA2 release register.

Appendix A

AER specification

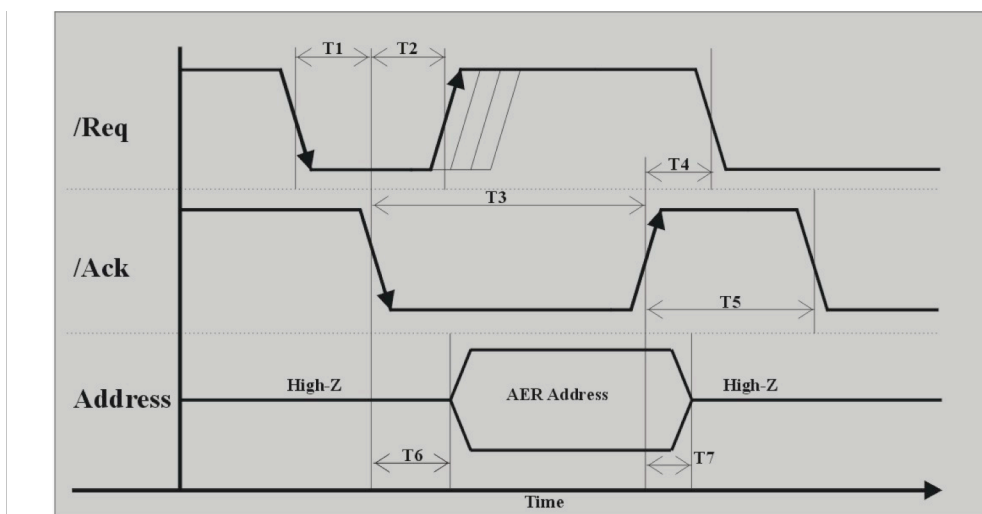
Multi sender implementation of AER.



PAUM_0010.jpg

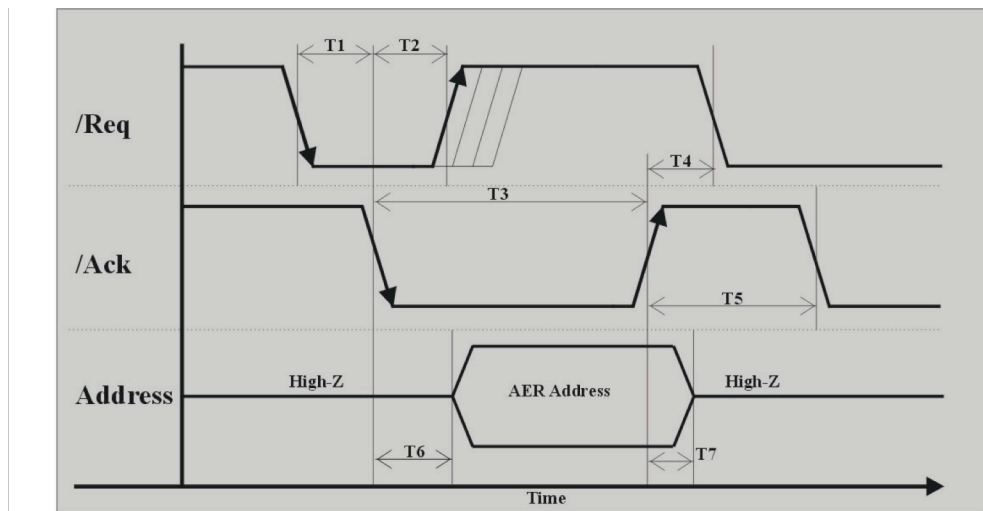
Figure 21: Multi sender AER transaction Specification

1. The Sender Chip begin transaction and assert /Req (Active low) signal.
2. The Receiver chip assert /Ack signal. This mean that Sender Chip can put address, of spiking neuron, on the bus.
3. The Sender Chip put address on the bus.
4. The Sender Chip drive address bus while /Ack signal is active (low). When Receiver Chip is ready set /Ack signal and free Sender Chip.
5. The Sender Chip free Address bus and new transaction can begin



PAUM_0011.jpg

Figure 22: Multi sender AER Time Specification



PA0M_00111.jpg

On Table 14 and

Figure 22 we can see typical time for AER transaction.

Label	Parameter	Min	Max	Unit	Note
T1	Latency time of Receiver			nS	
T2	/Ack fall to /Req rise delay	0	175	nS	
T3	/Ack Width	225		nS	
T4	/Ack Rise to /Req fall delay	0		nS	
T5	/Ack Rise to /Ack fall delay	100		nS	
T6	Label valid from /Ack fall	0	75	nS	
T7	Bus HI-Z from /Ack rise	0	100	nS	

Table 14: AER Transaction timing

Point to point implementation of AER.

This implementation of AER use only one sender and one receiver chip at the time.

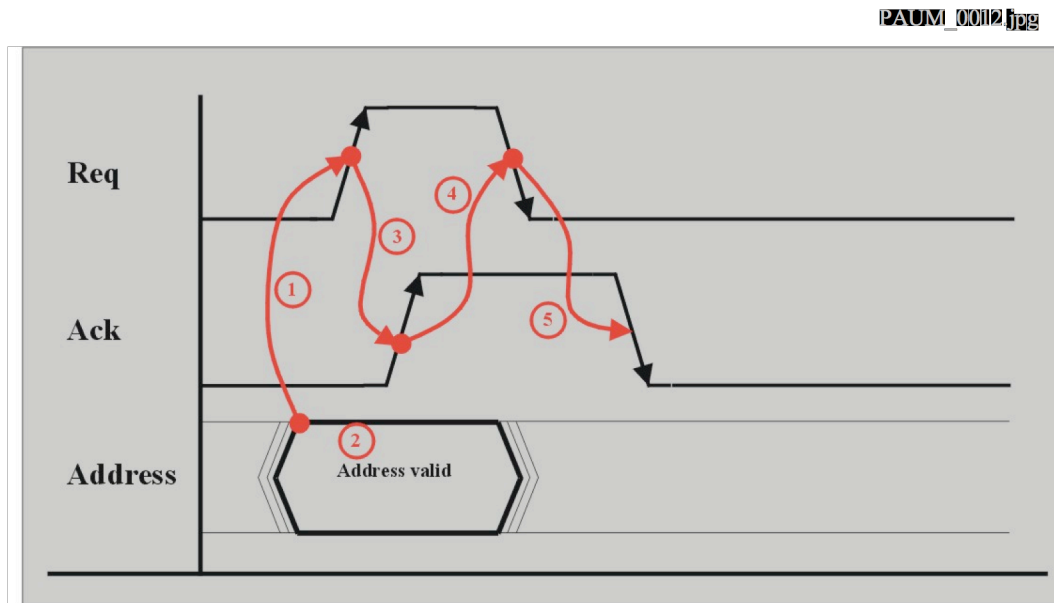


Figure 23: AER transaction Specification

1. Sender chip put valid address of spiking neuron on the bus.
2. Sender chip set REQ signal.
3. Receiver recognize REQ signal form sender and set ACK signal.
4. Sender chip free the BUS and reset REQ signal.
5. Receiver terminate AER transaction and reset ACK signal.

Appendix B

PCI-AER board standard pin connection

On the Table 15 and Figure 24 we can see pin number and short description of AER connector used on external board.

Pin N.	Signal	Description
1	AE0	AER address BIT 0 signal
2	AE1	AER address BIT 1 signal
3	AE2	AER address BIT 2 signal
4	AE3	AER address BIT 3 signal
5	AE4	AER address BIT 4 signal
6	AE5	AER address BIT 5 signal
7	AE6	AER address BIT 6 signal
8	AE7	AER address BIT 7 signal
9	AE8	AER address BIT 8 signal
10	AE9	AER address BIT 9 signal
11	AE10	AER address BIT 10 signal
12	AE11	AER address BIT 11 signal
13	AE12	AER address BIT 12 signal
14	AE13	AER address BIT 13 signal
15	AE14	AER address BIT 14 signal
16	AE15	AER address BIT 15 signal
17	GND	Ground
18	GND	Ground
19	/Req	AER Request signal
20	/Ack	AER Acknowledge signal

Table 15: AER Connector pin description.

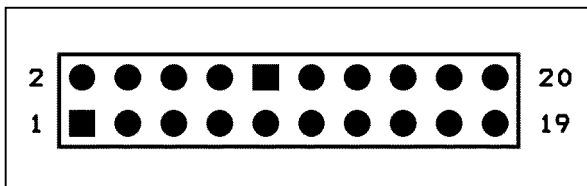


Figure 24: Top view of AER connector used on external board.

Appendix C

PCI-AER Accessories

AER standard converter

The PCI-AER Board accept in input “Multi Sender” AER implementation only. Therefore this small PCB is needed to connect “Point to Point” AER chip to the PCI-AER board. The AER Adapter can be configured both “Multi Sender” to “Point to Point” and viceversa through few PCB jumper. At the moment we use AER Adapter only to transform “Point to Point” devices on “Multi Sender” devices. The Adapter need 5Volt power supply.

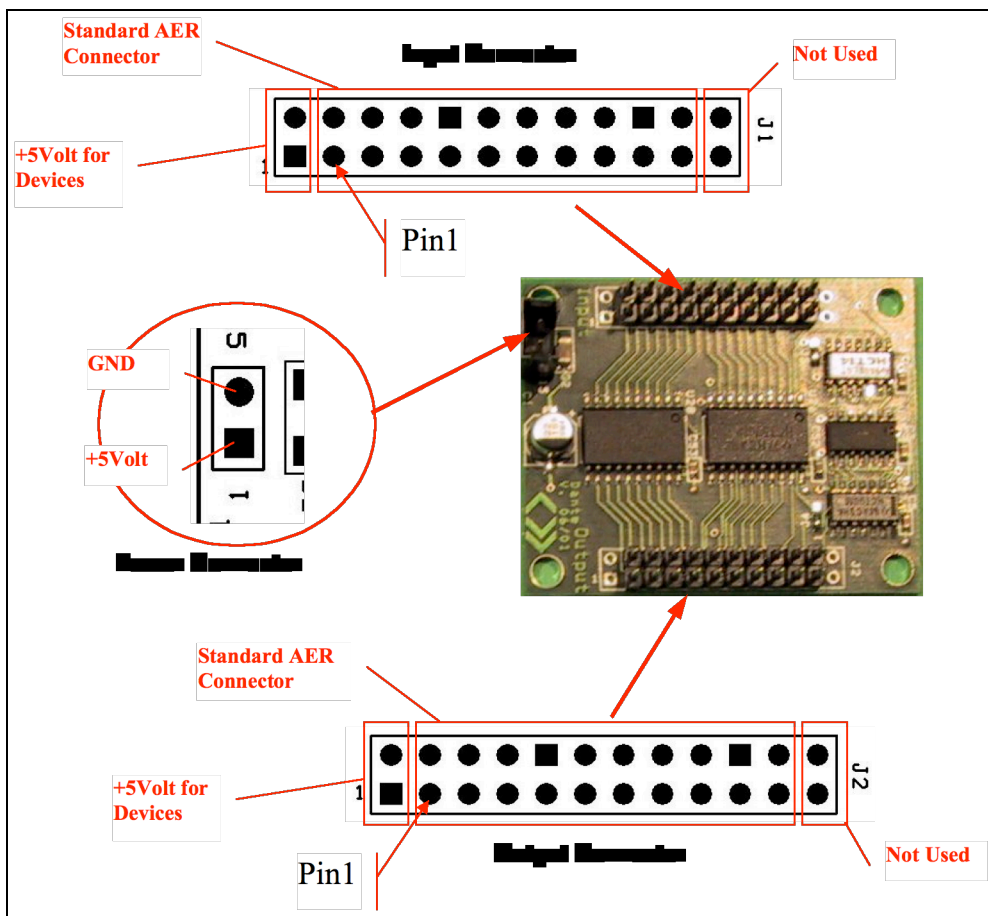


Figure 25: Top view of AER standard converter.

Input and output connectors are conform to the PCI-AER standard connector (See Appendix B).

References

¹ MindShare Inc., Tom Shaley, Don Anderson, "PCI System Architecture", Addison-Wesley Publishing Company, 1995
ISBN: 0-201-40993-3

² PCI PRODUCTS DATA BOOK.
Applied Micro Circuits Corporation
6290 Sequence Drive
San Diego, CA 92121-4358
<http://www.amcc.com>

The Linux driver for the Rome PCI-AER board
[*Adrian M. Whatley, Institute of Neuroinformatics, University & ETH Zurich*](#)