The new Little Associative Memory Board (LAMB) (DRAFT October 14, 2004)

1 Introduction

The purpose of the Little Associative Memory Board (LAMB) is to house up to 32 AMchip03, 16 per face. The structure of the LAMB is shown in figure 1 The AM chips come in PQ208 packages, and contain the stored patterns and read-out logic.

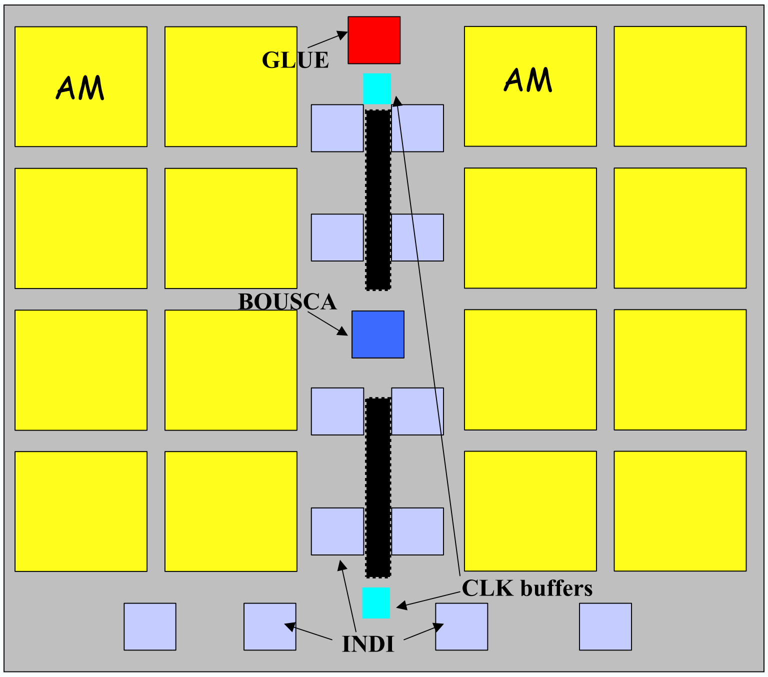

Figure 1: LAMB layout showing the placement of different kind of chips on board.

The LAMB card for every event receives hits and propagates them to the 32 AMchips through 12 INDI (INput DIstributors) chips. During and after the hits loading phase the LAMB GLUE collects roads from the AMchips and feeds them to the TOP GLUE on the AM board. The LAMB GLUE propagates incoming OPCODEs to all AMchips to change the configuration of matching logic and to impose the readout order of different kind of roads (e.g. first XFT + SVX 5/5 roads, then XFT + SVX 4/5, ...). The default matching criterion, that is also the most stringent, must be active until few cycles after the last hit is received to make sure that all roads relative to that criterion are readout before going to a looser criterion. In the same way, each subsequent change of the matching criterion must be activated with enough delay to readout all relative roads. We implement a mechanisms to reduce as much as possible the time from last hit-in to when the last road of the event is out.

2 Lamb description

Six hit busses are fed in parallel to the four LAMBs and distributed with fan-out 4 to the 32 AM chips on the LAMB, through 12 fan-out chips called Input Distributors (INDI). For the road address bus, the AM chips on each LAMB are connected into 4 pipelines (chains in the following), each corresponding to a column in figure 1, consisting of 8 chips, 4 on the front and 4 on the back face. Each AM chip receives an input bus where road addresses found in previous chips are propagated, multiplexes it with the road addresses internally found and provides an output bus to be sent to the next chip. Signals propagate from one chip to another on the top and bottom PCB planes through very short pin-to-pin lines, without any "via".

The outputs of the 4 chains are then multiplexed into a single bus by a purposely-designed GLUE chip available on each LAMB.

The patterns and internal configuration of AMchips are written using the JTAG protocol. A dedicated chip called BOUSCA allows to access JTAG pins through VME operations. The internal AMchip configuration specifies a default matching criterion that is activated at the beginning of each event. The matching criterion is controlled through 2 main parameters: THR and required_layers (see the AMchip03 specifications). THR is the number of layers that are required to be matched in a road in order to get a road match. required_layers is a 2 bits option. Each bit activate the option for one out of two predefined layers. The road match can be generated only if all active required_layers have a match. The matching criterion can be changed during the event processing through the OPCODEs.

3 LAMB usage

The following is the list of operations the LAMB will perform for each event.

1. init: the LAMB is reset just after the end of previous event. The default matching criterion is loaded (THR and required_layers).

2. hits loading: hits are fed to INDI chips from the AM board and propagated to the AMchips

3. roads readout(1): roads are readout during "hits loading" phase.[1] The matching criterion must not be changed (i.e. OPCODEs must not be sent to AMchips), until the last incoming hit has arrived and all generated roads have been readout.

4. send OPCODE: one opcode is distributed to the AM chips in order to change the matching logic configuration.

5. roads readout(2): roads generated with the new matching logic configuration are readout.

The "send OPCODE" and "roads readout(2)" operations (points 4 and 5) are iterated for every OPCODE.

The first (init) operation is very simple. The init signal[2] from the LAMB connector is received by INDI chips that propagate it to every AM chip and to the GLUE chip. The reset happens for all chips on the LAMB (INDIs, AMchips, GLUE) on the same clk rising edge.

[1] This step is optional and depends on the internal AMchip03 configuration.
[2] The "init" signal is also called "init event" signal.

During the “hits loading” operation layer matches are recorded in the internal logic of the AM chips.

The “hits loading” operation does not use the busses necessary for roads readout. Hence, in the meanwhile the readout of roads can start. With this option the last road would leave the LAMB few clk cycle after the last hit gets in. However it is useful, for the Road Warrior, to separately readout roads of different kinds at different times (e.g. first XFT+5/5 roads, then XFT+4/5, ...). For this reason during “roads readout(1)” only roads corresponding to the most stringent matching criterion (those corresponding to XFT+5/5) will be allowed to reach the P3 connector. The roads corresponding to other matching criterion will be readout on demand after the appropriates OPCODEs are received.

4 GLUE logic

A simplified scheme of the GLUE chip is shown in figure 2. The first purpose of the GLUE is to multiplex into a single stream the 4 AM chain outputs (2 word packets: Road-ID, bitmap). The upper part of figure 2 shows that after the multiplexing there is a register, then a 2 word FIFO and finally the output register. This means that each words needs 3 or 4 cycles to go from input pins to output pins depending on the state of the FIFO.[3] The controlling FSM makes sure that 2 words (road ID, bitmap), that constitute a packets, are kept together, one after the other.

Figure 3 shows the interconnections between the 4 AM chains and the GLUE. Road-IDs and bitmaps are transfered through the pipeline toward the GLUE. The protocol is the following: a 2 word packet is transfered when both “Output Ready” (OR)[4] and “Select” (SEL)[5] signals are asserted. The OR signals the availability of a couple of words. The SEL signals the availability of enough space to receive the 2 words. The first word (road ID) is latched in the input register on the clk rising edge on which both OR and SEL are asserted. The second word (bitmap) is latched on the next clk cycles regardless of the state of control signals.

The second purpose of the GLUE is to send OPCODEs to the AMchip03 chains. Figure 3 shows OPCODE and wiredOR (wired Output Ready) interconnections. The OPCODE is sent in parallel to all AMchips in a chain. The eight wiredOR[6] signals, that are the output of eight AMchips in a chain, are wired together into the GLUEwiredOR signal. The GLUEwiredOR is the logic OR of all wiredOR in a chain. The GLUEwiredOR provides a fast information to the GLUE on the availability of roads inside the chain made of eight AMchips. Each wiredOR signal is active if the corresponding AM chip has at least one road available for readout. The GLUE logic must take care of the fact that the wiredOR signal becames active with some clk cycles of delay with respect to the incoming hit or OPCODE that actually generates the road match.[7] The GLUE guarantees that roads extracted in correspondence with different OPCODEs are not mixed. The GLUE logic informs the TOP GLUE as soon as all found roads for a given OPCODE have been extracted.

The bottom part of figure 2 shows the OPCODE logic. The incoming OPCODEs go through a register and then reach the Opcode FIFO. This FIFO stores all OPCODE words (up to three) that have not been

[3] We are assuming that the TOP glue is always asking data to this GLUE.

[4] The OR signal is called Data Available (DA or DAout_Low) in the AMChip03 document

[5] The SEL signal is called Space Available (SA or SAin_Low) in the AMChip03 document

[6] The wiredOR signal is called wired Data Available (wiredOR_Low) in the AMChip03 document

[7] When a road-ID, bitmap packet is moving through the AM chain from AMchip to AMchip the GLUEwiredOR signal is continuously kept asserted.

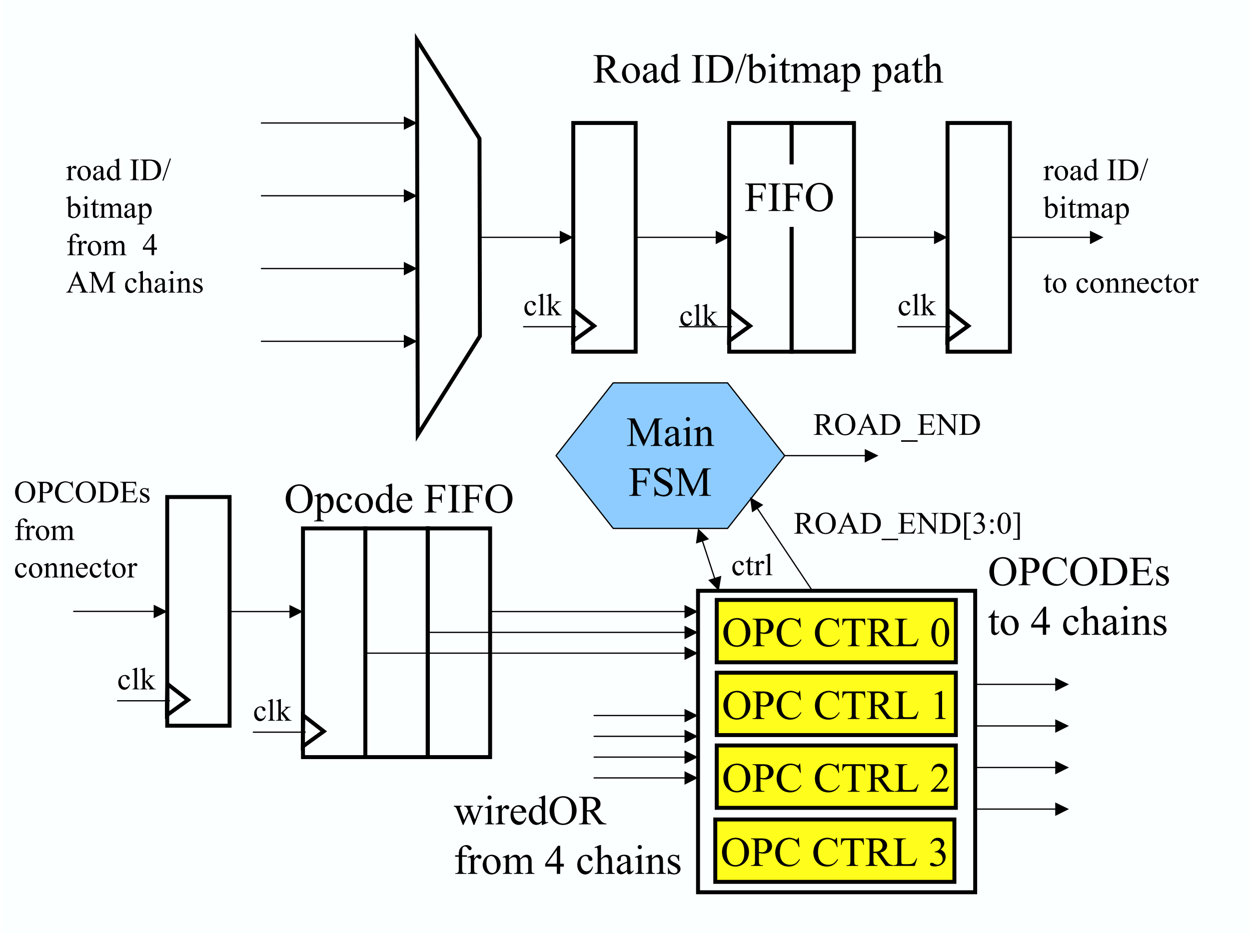

Figure 2: Diagram of the GLUE logic.

sent to all chains of AMchip03 yet. There are 4 "OPC CTRL" blocks working in parallel, one for each AM chain. All 3 registers of the Opcode FIFO are accessed in parallel by the "OPC CTRL" blocks.

Figure 4 shows the block diagram for one of the "OPC CTRL" blocks. Each block sends OPCODEs to one chain of AMchip03. The 4 blocks are coordinated by the "Main FSM". If an OPCODE is made of 2 words (OPCODE word + DATA word), the "OPC CTRL" block identifies those and keeps them together.

Every OPCODE is sent (written into the output register) to the chain only when the previous OPCODE has been completely processed. An OPCODE is considered processed when all roads generated by the OPCODE itself have been extracted from the AMchip03 chain. This condition is verified checking the AND of these two conditions:

- At least N clk cycles elapsed since the OPCODE has been sent, where N is the number of cycles necessary for the OPCODE to activate the GLUEwiredOR signal (in case of found roads).[8]

[8]If GLUEwiredOR is not active after N clk cycles no roads are found in the corresponding AM chain.

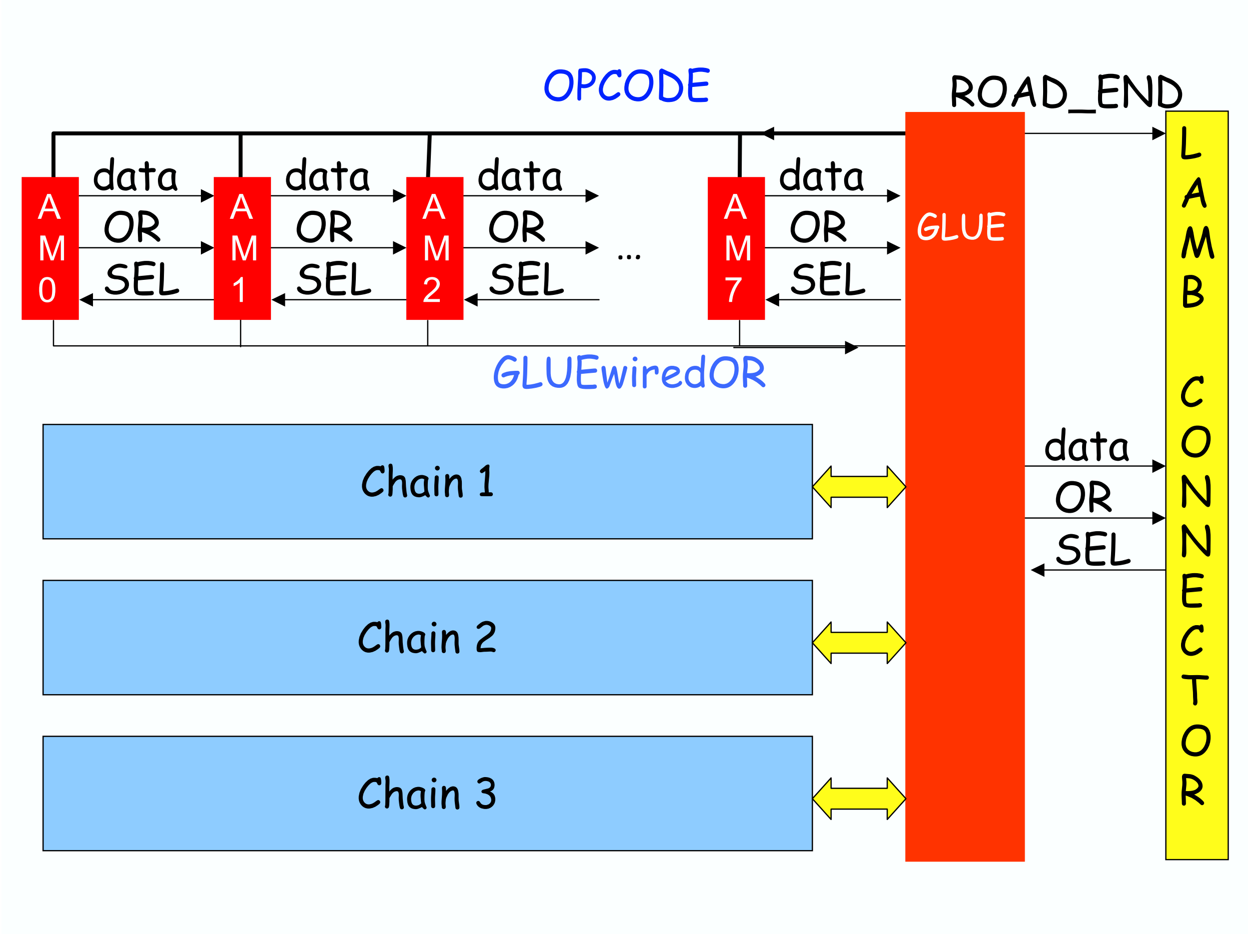

Figure 3: Diagram of interconnections of the LAMB GLUE with the 4 AM chains and with the connector. The interconnections within the first data chain (Chain 0) of 8 AMchips are shown on top.

- No more roads are left in the chain. This is verified checking that the GLUEwiredOR signal is inactive. This signal, described before, tells us if there are roads left inside any chip of the chain.

When the OPCODE is completely processed a ROAD_END signal is sent from the "OPC CTRL" to the "Main FSM" and the next OPCODE, if any, is sent to the chain. The purpose of the ROAD_END signal is to tell the "Main FSM" that roads for that OPCODE are finished.

With regards to OPCODEs the "Main FSM" has 3 jobs to do:

1. make sure that roads extracted with different OPCODEs are not mixed together.
2. propagate a ROAD_END signal, when all the four chains have exhausted their roads, to the TOP GLUE on the AM board for the same purpose.
3. remove OPCODEs already processed by all 4 chains from the FIFO.

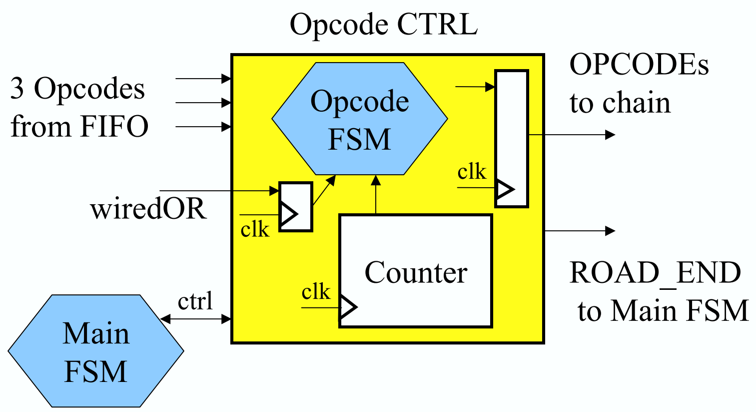

Figure 4: Diagram of the "OPC CTRL" logic.

In order to accomplish the 1st task the"Main FSM" holds (it does not produce the SEL signal) all data streams BUT those coming from chains that are processing the earlier OPCODE. For the 2nd task it will send the ROAD_END signal only after the last road corresponding to an OPCODE has left the GLUE.

4.1 Internal Pipelines

In this section we summarize the structure of the most important LAMB internal pipeline latencies. Understanding them is necessary for correct operations of the LAMB. Table 1 summarize the pipeline latencies.

- Pipeline # 1 tells us that the init signal operates a synchronous reset, with a delay of 3 cycles with respect to the cycle it appears at the LAMB connector. Hence the first valid hit must be sent with a delay of at least 3 cycles with respect to the init signal.

- Pipeline # 2 tells us that 24 clk cycles is the maximum time that can occur from the presence of the hit, that generates a road, on the LAMB connector, to the presence of the road-ID out on the LAMB connector. In this case the fired road is in the AMchip # 0 in the chain, the farthest from the GLUE.

- Pipeline # 3 is the same as 2 but from OPCODE on the LAMB connector to road-ID out. These 2 pipelines correspond to the time we should wait if we did NOT use the wiredOR signal and the OPCODE logic.

#	input	output	# cycles	List of steps in the pipeline
1	init	init action	3	INDI reg, input reg (any chip), reset
2	hit	patt_data (glue out, worst)	24	INDI reg, AMchip internal regs, AMchips pipeline, GLUE inreg, GLUE FIFO, GLUE outreg
3	OPCODE	patt_data (glue out, worst)	26	GLUE inreg, GLUE FIFO, GLUE outreg, AMchip internal regs, AMchips pipeline, GLUE inreg, GLUE FIFO, GLUE outreg
4	hit	wiredOR	7	INDI reg, AMchip internal regs, GLUE inreg
5	OPCODE	GLUE OPCODE FIFO	2	GLUE inreg, GLUE FIFO
6	OPCODE	wiredOR	7	GLUE outreg, AMchip internal regs, GLUE inreg

Table 1: Summary of pipelines internal to the LAMB board.

The cycles described above are necessary to get a road generated far from the GLUE. If no roads are found it is possible to proceed without waiting so many cycles thanks to the wiredOR information.

- Pipeline # 4 is from the time the hit is present on the connector to when wiredOR is available to “OPC CTRL”.
- Pipeline # 5 is from the OPCODE on the connector to the OPCODE available to “OPC CTRL”.
- Pipeline # 6 is from the OPCODE available to “OPC CTRL” to wiredOR available to “OPC CTRL”.

Pipeline # 6 tells us that we should wait 7 clk cycles after an OPCODE is sent to make sure that if a road was generated it would have activated the wiredOR signal. Pipeline # 4 tells us that we should wait 7 clk cycles after the last sent hit to make sure that if a road was generated it would have activated the wiredOR signal.

In order to extract all roads corresponding to the default matching criterion, that is activated by INIT at the beginning of the event, the first OPCODE must be distributed with an appropriate delay with respect to the time on which the last hit is received. Because the LAMB GLUE does not know which is the last hit in the event, the only way to control the timing of the first OPCODE is that the LAMB GLUE itself receives it with the appropriate delay. This delay is 5 clk cycles corresponding to the difference between pipeline # 4 and # 5. Since the whole AM++ does not know about the hit event structure, the sequencer, that does know, has to wait enough clock cycles after it receives EE on the hit stream before sending the first OPCODE to the AM++.

5 JTAG/VME interface

The configuration of AMchips is performed using the JTAG interface. The JTAG interface is in turn accessed from the board VME interface. The BOUSCA chip on the LAMB converts VME operations into

JTAG operations. With regard to JTAG interface the 32 AMchips are organized in 8 chains of 4 chips.[9] Each JTAG chain is made of 4 chips all on the same face connected in the vertical direction. There is an additional chain that connects the GLUE and all INDI chips to allow firmware update through VME or through a Xilinx download cable.

Accessing different VME registers it's possible to independently set TDI, TMS and to read TDO. For every VME access to a register used for JTAG operations all 4 LAMBs are accessed in parallel. Each LAMB is connected to 8 VME data bits that are used to program in parallel all 8 chains on the board. For example with a single VME operation the 32 TDI signals are updated (8 chains times 4 LAMBs).

6 Connectors & Pinout

Two types of connectors provide signals and power supply to the Lamb++. While a Molex mini-fit connector feeds main power supply (1.8 Volts and Ground), signals and low-current power supplies (3.3V) go across two very dense SMD connectors. The signal connectors are two fine pitch surface-mount connectors (for board-to-board stacking), featuring 100 SMD pins each, arranged in 2-row, 0.8 mm pitch (AMP-FH179031-4). Together with the mating connector, mounted on the AM++ (AMP-FH177983-4) the AM++/LAMB++ assembly realizes a total stacking height of 8 mm, which is fine for mounting both TQFP chips on AM++ top side and HQFP chips on LAMB++ bottom side, facing each other. The total air gap separating the two boards is between 2.7mm and 8mm, being mostly 3.9mm. Table 3 resumes the pin assignment of the two signal connectors, J1 and J2.

- Hit Data to AM chips. 6 separate Busses, each 18-bits wide.
  Hit0in(0:17) ,.., Hit5in(0:17)
- Address Data Bus, 20-bits wide.
  ADDOUT(0:19)

  The numbering of roads within LAMB is shown in table 2. The two spare bits are discarded by the TOP glue on the AMB++ board.

9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
chain #		spare		chip #			pattern # within AMchip												

Table 2: Numbering of roads

- Address Bus handshake control signals
  DR, SEL, ROAD_END
- Operation codes to AMchips
  OPCODE(0:3)
- A Group of VME data and control Signals, to access to AM chips programming (Pattern writing as well as threshold setup)

[9] Please note that there are 4 data chains of 8 chips.

VMEDATA(0:8) data to VME register
IADD(2:0) internal VME register selection
RDPam, WRPam Read/Write control
BITMAP_STATUS, vme option from AM++ glue

- A group of Boundary Scan signals to get full access to programmable logic (CPLD programming and in-circuit tests performing) from the Xilinx download cable that is located on the motherboard
  TMS_ID
  TCK_ID
  TDO_ID
  TDI_ID
- Two separate Clocks lines, for the two low skew buffer on the LAMB.
- Three separate Init Event signals to be distributed to 12 INDIs

In addition to the above signals, some pins are dedicated to Ground and 3.3V supply.

The power connector is a Molex automotive connector, rated for a maximum 11A current on each of the four pins. We estimate the power consumption of each AMchip03 Chip to be at most 0.5 watts while working at full speed (50 MHz). The current draw for the whole LAMB++ mounting 32 Scam chips is about 9A, assuming that it is mainly supplied by 1.8V.

An issue about power on the LAMB++ is that the 1.8V power plane is vertically split in two halves, so that we can bring two separate power lines to each half, each feeding one half of the chips, see figure 5. Each of the two halves is connected to the AM++ supply with one of the power connector pins. The remaining two pins of the power connector are assigned to Ground

The LAMB power capacity (22A @ 1.8V) is larger than needed for possible use of multiple busses to load hits in parallel that would increase the power consumption.

J1 pin	Signal Name	J1 pin	Signal Name	J2 pin	Signal Name	J2 pin	Signal Name
1	HIT4_IN1	51	GND	1	HIT1_IN1	51	ROAD_END
2	HIT4_IN4	52	RDPAM	2	OPCODE_IN3	52	ADDOUT17
3	HIT4_IN3	53	VMEDATA5	3	HIT1_IN3	53	VCCGLUE
4	HIT4_IN11	54	VMEDATA6	4	HIT1_IN4	54	VCCGLUE
5	HIT4_IN5	55	VMEDATA7	5	HIT1_IN5	55	ADDOUT10
6	HIT4_IN15	56	IADD2	6	HIT1_IN7	56	ADDOUT8
7	HIT4_IN8	57	WRPAM	7	HIT1_IN8	57	ADDOUT6
8	HIT4_IN12	58	VMEDATA4	8	HIT1_IN12	58	ADDOUT5
9	HIT4_IN16	59	GND	9	HIT1_IN16	59	ADDOUT2
10	INIT_EV_IN2	60	HIT5_IN7	10	VCC1V8	60	ADDOUT4
11	GND	61	HIT5_IN17	11	BITMAP_STATUS	61	ADDOUT3
12	HIT2_IN1	62	HIT5_IN14	12	VCC	62	ADDOUT1
13	HIT2_IN4	63	HIT5_IN13	13	VCC	63	ADDOUT0
14	HIT2_IN3	64	HIT5_IN9	14	VCC	64	HIT3_IN17
15	HIT2_IN11	65	HIT5_IN10	15	CLKA	65	HIT3_IN7
16	HIT2_IN5	66	HIT5_IN6	16	GND	66	VCC
17	HIT2_IN15	67	HIT5_IN0	17	GND	67	GND
18	VCC	68	HIT5_IN2	18	OPCODE_IN1	68	VCC
19	HIT2_IN8	69	HIT0_IN17	19	OPCODE_IN2	69	HIT3_IN13
20	HIT2_IN12	70	VCC	20	VCC	70	HIT3_IN14
21	HIT2_IN16	71	HIT0_IN13	21	VCC	71	HIT3_IN9
22	GND	72	HIT0_IN14	22	GND	72	HIT3_IN10
23	HIT0_IN7	73	HIT0_IN9	23	GND	73	HIT3_IN6
24	HIT0_IN1	74	HIT0_IN10	24	HIT3_IN1	74	GND
25	HIT0_IN4	75	HIT0_IN6	25	HIT3_IN4	75	HIT3_IN0
26	HIT0_IN3	76	HIT0_IN0	26	HIT3_IN3	76	INIT_EV_IN0
27	HIT0_IN11	77	HIT0_IN2	27	HIT3_IN11	77	HIT3_IN2
28	HIT0_IN5	78	VCC	28	HIT3_IN5	78	GND
29	HIT0_IN15	79	GND	29	HIT3_IN15	79	GND
30	HIT0_IN8	80	HIT2_IN17	30	HIT3_IN8	80	VCC
31	HIT0_IN12	81	HIT2_IN13	31	GND	81	VCC
32	HIT0_IN16	82	HIT2_IN7	32	HIT3_IN12	82	OPCODE_IN0
33	HIT5_IN1	83	VCC	33	HIT3_IN16	83	TDI_ID
34	HIT5_IN4	84	HIT2_IN14	34	GND	84	TCK_CON
35	HIT5_IN3	85	HIT2_IN9	35	VCC	85	TMS_CON
36	HIT5_IN11	86	HIT2_IN10	36	VCC	86	TDO_ID
37	HIT5_IN5	87	HIT2_IN6	37	ADDOUT9	87	GND
38	HIT5_IN15	88	HIT2_IN0	38	ADDOUT11	88	GND
39	HIT5_IN8	89	HIT2_IN2	39	ADDOUT14	89	GND
40	HIT5_IN12	90	INIT_EV_IN1	40	GND	90	VCC1V8
41	HIT5_IN16	91	HIT4_IN7	41	GND	91	HIT1_IN15
42	VCC	92	HIT4_IN17	42	ADDOUT18	92	HIT1_IN17
43	TCKIN	93	CLKB	43	ADDOUT12	93	HIT1_IN11
44	IADD0	94	HIT4_IN13	44	ADDOUT15	94	HIT1_IN13
45	IADD1	95	HIT4_IN14	45	ADDOUT19	95	HIT1_IN14
46	VMEDATA3	96	HIT4_IN9	46	ADDOUT16	96	HIT1_IN9
47	VMEDATA2	97	HIT4_IN10	47	DR_PAM	97	HIT1_IN10
48	VMEDATA1	98	HIT4_IN6	48	ADDOUT7	98	HIT1_IN6
49	VMEDATA0	99	HIT4_IN0	49	SEL_PAM	99	HIT1_IN0
50	GND	100	HIT4_IN2	50	ADDOUT13	100	0 HIT1_IN2

Table 3: LAMB Signal connectors pins assignment.

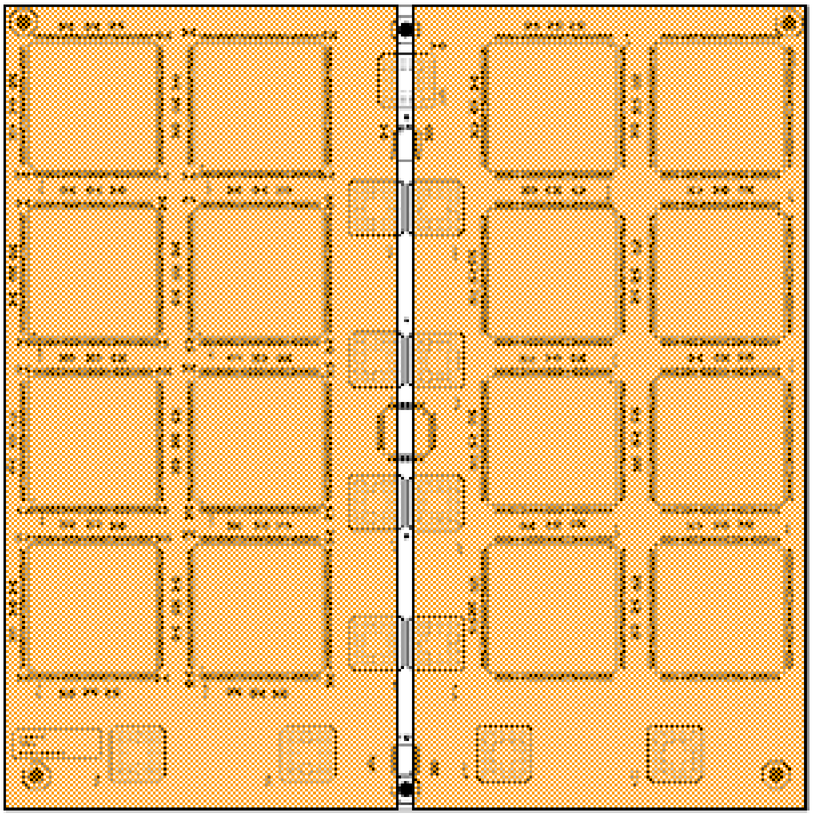

Figure 5: A view of the LAMB++ Power Plane Splitting.