

**Documentation of the Correlator Card**  
**ALMA07003SX0004**  
**By Joseph Greenberg**  
**March 5, 2002**

## Update History

### Changes for Next Version

1. Make Mux signals go directly off the card, instead of through the CPLD.
2. Make the Connector to the Mezzanine card a 12 pin, instead of 8 pin connector.
3. On the above connector make pin 8, a 3.3 V signal to go to the analog mux.
4. On the above connector, make pin 10 the Mux Ground.
5. Have the Mux Ground pin go directly off the card as pin 1E15. This currently is a normal ground.
6. Signals AUXTIME[1:0] only needs to be a single signal AUXTIME0.
7. Signal AUXTIME0 is currently terminated on the card, in the upper left of the top level. Instead the termination should be on the backplane, since 4 cards will be bridged in each direction.
8. In the pads where the power comes down from the Mezzanine card, have extra vias to distribute the current to the interior power plane.
9. Make the board thicker so it won't flex so much.
10. Disconnect SP33B[1:0] between Dataout 6 and 7, to eliminate possible bus contentions, since SP33B is used for test points on the other Xilinx. These could go to additional testpoints.
11. On 5 V trace to the Mezzanine card make it wider in layout (currently 6 mils).
12. The clock driver output out of U3 is rated at 3ma high drive, yet is driving 200 Ohms for about 15 ma. Maybe a higher resistor values are needed, or none at all.
13. Add two plated through holes connected to ground for additional Mezzanine Board support.

### To be Considered for Next Version

1. R159, R333, R22, R21, R35, R85, the 110 Ohm Termination resistors were removed from the CS\ and WRITE\ of the Analog Sum Xilinx. This was hoped to improve the download, since initially a LVTTTL input buffer is used. No effect was noted. Should this be a permanent change?

## **Attached Figures**

### ***Correlator Card Schematics***

Toplevel	Eightasics
Almachip	Dataoutxilinx
Asumendxilinx	Asum2ndxilinx
Asummidxilinx	Microinterface
Aslvds	Dataoutlvds

### ***CLPD Logic***

Corrcpld

### ***Analog Sum Xilinx***

Asumend1.	Asum2nd1
Asummid1	Addoffset
Add56and	Add32ant
Add16ant	2bb8ant
add8ant	add4ant
2bitadder	finaladd
clocks-mic	clockymicro

### ***Dataout Xilinx***

Dataoutblockdiagram	
Dataout1	Dataoutpath2
Sequencer	Rams
Loadmux	Intaddress

### ***Power Supply Mezzanine Board***

Powersupplies

### ***Correlator Card-LTA Test Fixture***

Corrltatestfix

# Card Layout

Figure 1 shows the layout of the correlator card. The card has chips on both sides. The board is shown top side up, with the chips on that side shown as solid lines. The 64 asics, and U3 the clock driver are on the top. All the other chips are on the back.

FIGURE 1



## The ALMA1 Correlator Chip

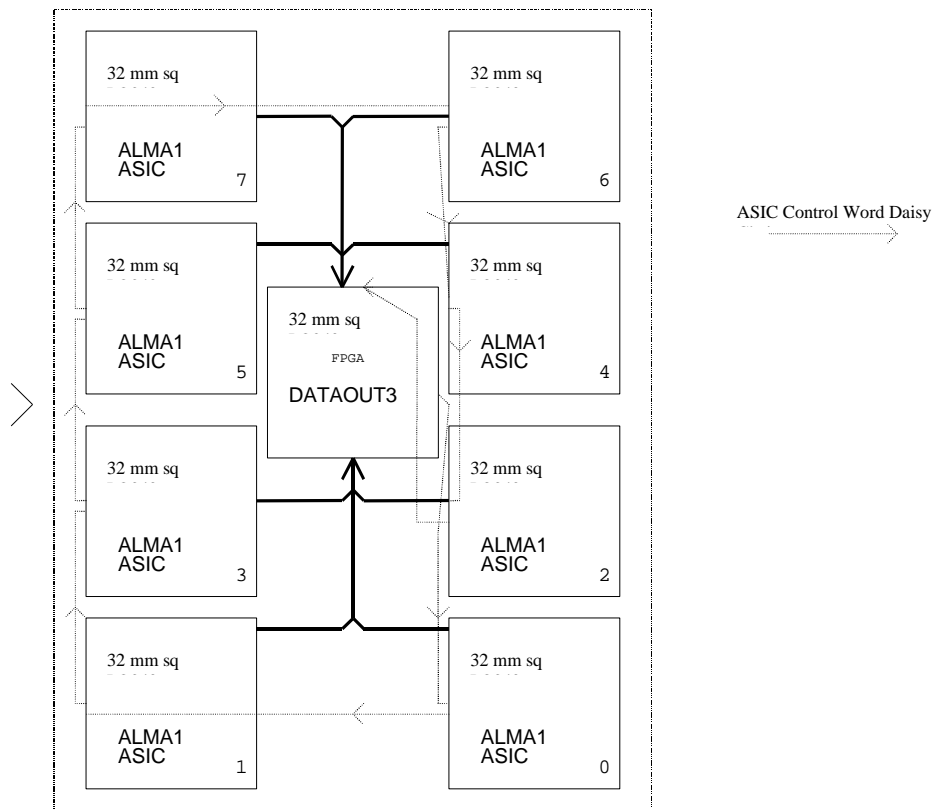
The heart of the Correlator Card is the ALMA1 Custom Correlator Chip. See the card's description on EAGLE:\MMA Correlator\jhg\ALMA CHIP\alma1spec.doc. The associated figures are contained in ALMA1SPEC.DSN. The pinout and control word are described in alma1spec.xls.

### Card Data Flow

The data from 32 antennas on the main bus, and possibly an additional 32 antennas on the AUX bus, comes in the bottom of the card. The data goes through the 5 Analog Sum Xilinx. These buffer the data, and allow for taking the Analog Sum. The analog sum flows towards the center Xilinx and off the card as LVDS. The data through the ASICs goes up the card and also to the left or right. All input data flow is from ASIC to neighboring ASIC. This is described in detail in the ALMA1 Chip Spec.

Each of the eight Dataout Xilinx drains data from its eight surrounding ASICs as shown, in Figure 2. There are two, tri-state busses each serving four ASICs. Also shown is the path the control word is shifted through the eight ASICs.

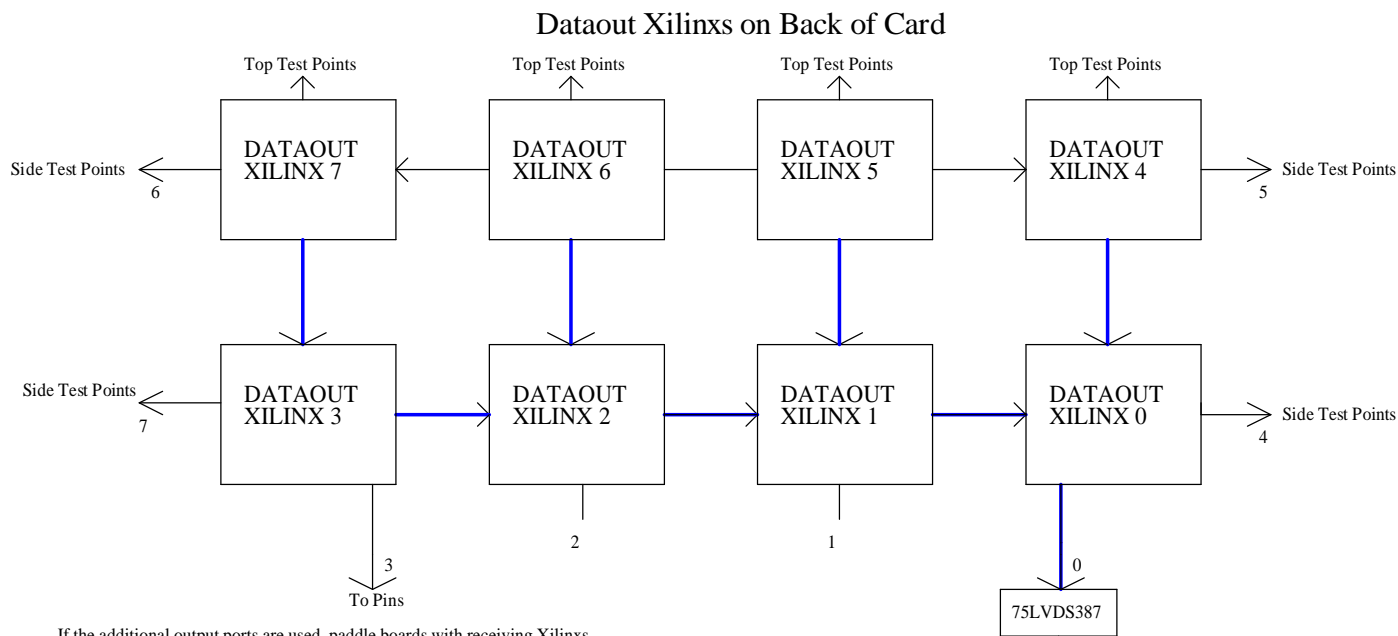
FIGURE 2 Each DATAOUT Xilinx is connected to eight ASICs, on the opposite side of the card, via two tri-state busses, as shown.



Control Words loaded in bottom ASICs first to allow for a partially populated card, with only

Figure 3 shows the data read out is shifted through the Dataout Xilinx, to Dataout Xilinx 0 where it exits the card as LVDS. There is a Quadrature driver to vary the timing of the output, to allow synchronous capture at the LTA. There is an integer and fractional (in quarters of a cycle) delay, as programmed by the XID word. The fractional portion is floorplanned, to assure precision, on the sheet FRACT-DELAY.

FIGURE 3



If the additional output ports are used, paddle boards with receiving Xilinx will be slipped over the card connectors.

Currently all out from one port.

TI LVDS DRIVERS FROM DATAOUT XILINX

Each 75LVDS387 outputs 16 LVDS differential signals.

All outputs are 16 bits and 3.3 V. Output port 0 goes to the LVDS drivers.

If all 8, 16 bit output busses were used, at 125 MHz, these can drain the card in less than 1 ms. The current LTA will use a single 8 bit differential LVDS bus, running at 125 MHz, to drain the card in 16 ms.

### **125 MHz Clock Distribution**

The clock is distributed by a tree of Xilinx. The internal DLLs are used to minimize clock skew. DLL feedback path lengths are matched to run lengths. The clock runs are shown as dashed lines on the layout diagram. The clock initially enters the card as a sine wave, and is converted to a square wave by a 10H842L. The clock then goes to the center Analog Sum Xilinx. The center Analog Sum Xilinx provides clocks to the 4 other Analog Sum Xilins. Path lengths are equalized. Each of these four Analog Sum Xilins provides clocks to the two Dataout Xilins directly above them. Each Dataout Xilinx provides clocks to its eight asics.

### **Partial Card for the Two Station Correlator**

Initially there will be a two station Correlator for evaluating the first two Alma antennas.

The 4 x 4 array in a chip can process two, 2x2 arrays. The first chip can process Array 0, Ant0 x Ant1, plus Array 1, Ant0 x Ant 1. Note only half the intersections in the chip are doing useful work.

Thus a card with eight asics along the bottom row, can process sixteen Planes. The 4 GHz bandwidth is broken into 32 Planes with 125 MHz bandwidth each. Two cards are required to process all 32 Planes. For convenience of processing in the LTA, the even Planes will be on one card, and the odd Planes on the other card. This is for two basebands. For eight basebands, eight cards with eight asics each are required.

Only the bottom row of chips can be effectively used. The next row would have to have its data supplied by the AUX bus. However, there is no way to get the AUX bus into both axes. Both axes are required for getting the cross products. An alternative could be to put separate antennas in M0 and M1. This would allow the cross products of these antennas to be generated using the single axes. However, we would lose the ability to use M0 and M1 to contain the polarization components of the same antenna.

The four station correlator would require twice the number of Asics and cards, since each chip would handle one plane of 4 stations, instead of 2 planes of 2 stations. A three station correlator would require the same number of asics as a four station correlator, since each chip does 4 stations.

The control word data and clock are serially shifted from the Dataout Xilinx, to the bottom two asics first. Jumpers can be placed from CD2 to CD8, and CC2 to CC8 for the partially populated card. This allows reading the control word back into the Xilinx, bypassing the missing asics.

The full Analog Sum from the Correlator Card will not be possible in the two station Correlator. Since a single card has 16 planes, each two basebands, that makes for 32 sums. Summing two 2 bit numbers together gives 3 bits. 32 sums times 3 bits makes for 96 bits output. The card only has provisions for outputting 16 bits of Analog Sum data.

In the mode, where only one of the planes is used, for 125 MHz bandwidth, then the analog sum of a single plane could be obtained. The mask could be set to just add the two antennas together. Special Xilinx personalities could be written to add two planes together, without being pin limited.

## **Card Schematic Diagram Description**

See EAGLE:\MMA Correlator\jhg\Orcad\CORR\_CARD.DSN

### ***Top Level***

The top level shows the seven sections of the card connector. At the top of the card is a test connector JR1. Around the periphery of the card are numerous test points. These could have pins soldered in for test purposes. On the right side are Power Terminals to allow for a bolt on, 1.8V DC to DC converter board. This would convert 48VDC to 1.8VDC at about 80 Amps. There is also a connector with power supply control functions to the board. Alternately, the 1.8VDC could be provided to the board through the card connector pins.

There are hierarchy elements for each of the five Analog Sum Xilinx. Xilinx personality ASUMEND is used for ALOGSUM0 and 4. ASUM2ND is used for ALOGSUM 1 and 3. ASUMMID is used for ALOGSUM2.

The hierarchy element EIGHTASICS contains a grouping of eight asics and the associated Dataout Xilinx.

The MICRO INTERFACE contains the associated control logic to talk with the microprocessor in the LTA.

### ***EIGHT ASICS***

This contains eight copies of the hierarchy element that contains an asic. Also there is the hierarchy element that contains the Dataout Xilinx which services these eight asics.

### **DATAOUT XILINX**

This hierarchical element contains the Dataout Xilinx. Also there is a 74LVC139 multiplexer. The multiplexer assures that only one output enable to the 8 asics is present at a time. This prevents bus contention, if the Xilinx is not programmed or programmed incorrectly.

### **ALMACHIP**

This hierarchical element contains the ALMA1 Correlator Chip.

### ***MICRO INTERFACE***

The XC95144XL CPLD is the main interface device to the 167 microprocessor on the LTA.

The 8 bit BUS-DATA line comes from the LTA. It exits the FPGA as CD[7:0], and is distributed to the Xilinx as part of the GLOBAL[13:0] bus. This provides data for loading the Xilinx personalities, and data transfer to and from the Xilinx in their programmed function.

### CPLD JTAG Programming

The CPLD is programmed in place via its JTAG port. The JTAG signal can come either from the LTA, or pins allocated as a separate JTAG port on the card.

To program from the LTA, the signals TDI167, TMS167, and TDO167 come from the LTA. The 74LVCH244 routes the signal when the JTAGOE\ signal from the LTA is asserted. JTAGOE\ allows the signals from the LTA to be bussed to the cards, with an independent JTAGOE\ enabling each card. The clock TCK comes from the differential BUSCLK signal from the LTA. It is enabled by JTAGOE from an independent 26LV32.

When JTAGOE\ is not asserted, the JTAG signal is routed via the 244, from the card pins CTDI, CTMS, CTCK, CTDO. This provides a port directly to a PC for the card. The LVCH244 allows the inputs to be floating without pullups.

The differential BUSCLK signal from the LTA serves a second purpose. When JTAGOE is not asserted, the signal provides the CLK-IN signal to the FPGA, for clocking in data from the BUS-DATA[7:0] bus.

### Xilinx FPGA JTAG Programming

To program the 13 Xilins on the card, there is a JTAG port to the card pins going to the CPLD pins XTDI, XTMS, XTCK, & XTDO. These go out of the CPLD on the pins FTDI, FTMS, FTDO, & FTCK[5:0]. The 6 versions of the clock help ensure clock signal integrity. The FTDI signal daisy chains through the 13 Xilins and returns in the FTDO.

### Xilinx FPGA Parallel Programming

The data comes over the 8 bit data bus CD[7:0]. The Xilinx desired is selected via the Chip Select signal DATAOUTCSB[7:0], ALOGSUMCSB[4:0]. Multiple Xilins can be programmed at once, by having multiple Chip Select Signals enabled simultaneously. The Data is clocked by the CCLK[5:0] signal. All 6 CCLKs are identical.

### Xilinx FPGA Data I/O

During operation, the data is transferred to/from the Xilinx by using the same CS signal as above as a clock. The same 8 data lines are used also.

### CPLD Personality

See the corrcpld Xilinx project. This is a variation of the design from Chuck Broadwell. See Chuck for detailed documentation. The corrcpld figure shows the logic contained in the Xilinx. The main difference from Chuck's version is that the inputs SPARE[2:0] go

straight through to the ONOFF[2:0] outputs. This will be bypassed on the next version of the Correlator Card.

## Xilinx Designs

### *Analog Sum Xilinx Designs*

#### Introduction

The Analog Sum Xilinxs serve a triple function.

1. They act as an input buffer for the data entering the card.
2. They perform an analog summation of the data as it goes by. Each 125 MHz clock, calculate a signal representing the summation of up to all 64 antennas.
3. They provide a selectable test data pattern into the Asics.

Refer back to Figure 1. There are five Xilinxs, along the bottom of the card. There are three Xilinx personalities. ASUMID is in the middle Xilinx. ASUMEND is in the far left and far right Xilinxs. ASUM2ND is in the second and fourth Xilinxs.

The Analog sums propagate from the end Xilinxs towards the middle Xilinx. The Analog Sum exits the card from the middle Xilinx as LVDS signals.

#### Bus Structure

The MAIN busses are used for both self and cross cards. The AUX bus is only used on cross cards. Refer to Figure 10 in the Chip Spec, the Self Card. Note on a Self Card, only 32 antennas are input to the card. Refer to Figure 11 in the Chip Spec, the Cross Card. On a Cross Card, all 64 antennas are input to the card. Thus the Cross Card has the capability to do the sum of all 64 antennas. Since there are two Self Cards and two Cross Cards in a plane, they could each calculate the analog sum of different subarrays.

The bits in each 16 bit bus are assigned as below:

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ANT3 ANT2 ANT1 ANT0 ANT3 ANT2 ANT1 ANT0
-----MEMORY M1 -----|-----MEMORY M0-----
```

The Analog sums for each Memory or baseband must be done separately. In all cases, ANT 0 through 3 for each Memory would be added together.

#### Simple Adder Mathematics

In adding just two antennas together, it must be accounted for that the numbers represent the following weighting:

Bits	Weighting
0 1	+3
0 0	+1
1 1	-1
1 0	-3

This table defines the math for an adder using these weightings. The column to the right of the equal sign represents the sum of the weightings represented by the bits. Since the sum always has an lsb of 0, the lsb can be dropped, as shown in the next column (labeled /2). The columns a, b, and c, show this number in binary. The logic to achieve these values is shown, derived from the Karnaugh Maps.

Analog Sum Truth Table

	w	x	+	y	z	=	Su	m	/2	a	b	c
0	0	0	0	0	0	2	1	0	0	1		
1	0	0	0	1	1	4	2	0	1	0		
2	0	0	1	0	1	-2	-1	1	1	1		
3	0	0	1	1	1	0	0	0	0	0		
4	0	1	0	0	0	4	2	0	1	0		
5	0	1	0	1	1	6	3	0	1	1		
6	0	1	1	0	0	0	0	0	0	0		
7	0	1	1	1	1	2	1	0	0	1		
8	1	0	0	0	0	-2	-1	1	1	1		
9	1	0	0	1	1	0	0	0	0	0		
10	1	0	1	0	0	-6	-3	1	0	1		
11	1	0	1	1	1	-4	-2	1	1	0		
12	1	1	0	0	0	0	0	0	0	0		
13	1	1	0	1	1	2	1	0	0	1		
14	1	1	1	0	0	-4	-2	1	1	0		
15	1	1	1	1	1	-2	-1	1	1	1		

j:/excel/analogsum.xls

$$wy + w \ x \ y \ z \ + \ w \ x \ y \ z \ +$$

a

			y	y
	0	1	3	2
				X
	4	5	7	6
				x
w	12	13	15	14
			X	X
w	8	9	11	10
	X		X	X

$$w \ x \ y \ z \ + \ w \ x \ y \ z \ +$$

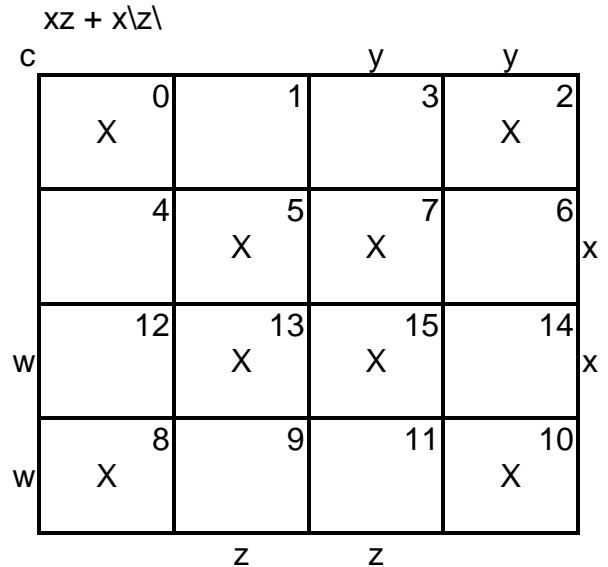
$$w \ x \ y \ z \ + \ w \ y \ z \ +$$

$$w \ x \ y \ z \ + \ w \ y \ z \ +$$

$$w \ x \ y \ z \ + \ w \ y \ z \ +$$

b

			y	y
	0	1	3	2
		X		X
	4	5	7	6
	X	X		x
w	12	13	15	14
			X	X
w	8	9	11	10
	X		X	



This logic is implemented in the Xilinx sheet 2BITADDER. There is a MASK bit, associated with each 2 bit number from an antenna. The mask bit being 1 means include that antenna in the sum. Setting the mask to 0, zeros the antenna input. However, due to the biased math, that means that 1 is added in every time. This can be subtracted out after all the antennas are summed, in the middle Xilinx, based on the number of antennas that have been excluded.

ADD4ANT combines two 2BITADDERs to add up 4 antennas.

ADD8ANT combines the 4 antennas from the 8 bit MAIN bus, and the 4 antennas from an 8 bit AUX bus. Note each time we double the number of antennas added, an extra bit appears in the final sum. The TESTMUX section is in series with the data. This allows substituting test data, for the input data, based on the 3 lsb's of the CONTROL register from the microprocessor, as follows:

- CONTROL = 0,1,2 or 3 All zeros used.
- CONTROL = 4 Normal Input data used.
- CONTROL = 5 COUNT[7:0] from the counter is used.
- CONTROL = 6 Pseudo Random Number generator used.
- CONTROL = 7 All ones used.

The 16MSIN signal resets the above COUNT to zero, and initializes the random number generator to the seed. The Drawing DATAGEN has the standard random number generator. A 32 bit seed is shifted in by four microprocessor writes using WR\_EN12.

2BB8ANT has two ADD8ANT to sum a full, 16 bit main and aux bus together. This is the input to one ASIC.

### Input Stream Data Checking

The sheet PNCHECK allows selecting and the checking of an input data stream. It is assumed the input stream has been generated by our standard Random Number Generator. The Random numbers could have been generated by a Xilinx in the station card, or by a Xilinx in a Paddle Board. The input stream is initially used to seed a Random Number Generator. The generator is then allowed to free run and is compared to the input stream. Error counts are kept, which can be read by the microprocessor. A multiplexer selects which of the up to 64 input streams are analyzed. Since PNCHECK is inserted downstream of TESTMUX, the internal Random number generator can be tested if selected.

### Multiplexed Test Outputs

These are outputs from the chips to the test connector on the top of the card. CONTROL[5:3] of the respective Xilinx selects which of the eight mux positions to use. ASUMEND does not have any spare pins available for this purpose. ASUMMID has 4 pins to the connector. One gives the 16ms pulse. The other 3 give the 3 lsb's of the signal selected by CONTROL. Each of the two ASUM2ND Xilixns have two pins to the connector. These have the two lsb's of the signal selected by CONTROL.

### Internal Logic Analyzers

In each Xilinx, in the CLOCKS-MICRO-INTERFACE section, Rams are used as internal logic analyzers. There is counter, common to all the Rams of the Xilinx, that generates the address LACNT[7:0]. This Ram is cleared by D\_WR\_EN5. The counter counts at a C125 rate for 256 counts, when the count freezes. This gives a 256 count snapshot of the input data streams in the Rams. Each Ram stores a 16 bit field.

### ASUMEND

The ASUMEND Xilinx personality is used on the far left and far right of the five Analog Sum Xilixns.

The top level Xilinx drawing, ASUMEND1 shows the four, 16 bit data busses MAININA, AUXINA, MAININB, AUXINB enter and exit the chip on the way to the Asics. The A and B busses go to separate Asics.

A separate 2BB8ANT module does the sum for the signals for each Asic. The outputs of these are added in two ADD16ANT modules. These output sums for M0 and M1. These sums are routed to the next Xilinx towards the center, where they are added into that Xilinx's sums.

The CLOCKS-MICRO\_INTERFACE diagram provides the system clock and control words from the microprocessor. Control words are the masks for masking antennas, and Control to select test words.

CLOCKMICRO has the standard DLL circuitry. It also has the standard microprocessor interface.

## ASUM2ND

The ASUM2ND Xilinx personality goes in the second and fourth Analog Sum Xilinx.

Looking at the top level, ASUM2ND1, we see a structure similar to ASUMEND. MAININC and AUXINC are handled identically, through a 2BBANT module.

2BB8ANTSP handles only a MAIN bus, instead of a MAIN and AUX bus. That is since ASUM2ND handles only 3 16 bit busses instead of the 4 handled by ASUMEND. That is because it has the added pins required for the sum coming from ASUMEND, and a sum going to ASUMMID. 2BB8ANTSP outputs a sum of 4 antennas.

The ADD16ANT block adds the outputs of 2BB8ANT, and 2BB8ANTSP, to give the 12 antenna sums, M0SUM12CD, and M1SUM12CD. The same module doesn't care if its adding 12 or 16 antennas.

ADD32ANT then sums the signals from this Xilinx, with those from the neighboring ASUMEND, to give 28 antenna sums. Notice the 2X6DELAY element delays the data generated in this Xilinx two clocks, so it will be synchronous with the data from the neighboring ASUMEND.

## ASUMMID

The ASUMMID Xilinx personality is for the middle of the five Xilixs. This adds the results going to one asic, to the rest of the results coming from the neighboring ASUM2ND Xilixs.

Looking at ASUMMID1, there is a single MAIN and AUX bus, going into a 2BB8ANT, to give an eight antenna sum.

ADD56ANT sums the 28 antennas from the Xilixs to the left and right to give a 56 antenna sum.

FINALADD adds the 56 antenna sum, and the 8 antenna sum, to give the final, 64 antenna sum. The 8 antenna signal is delayed 6 counts to match the timing, as described in the 6 COUNT DELAY LINE section.

ADDOFFSET adds the offset required since the mask, in making the results 0, is actually adding 1 each time. To calculate the offset in the microprocessor, take the number of

antennas out of 64 that are masked off. Divide by two, since the lsb was discarded in the initial adder. It's ok to truncate. To subtract the offset, convert the number to an eight bit, two's complement number. For example, masking 5 antennas would give an offset of -2. Note putting the offset in this path allows for a smaller delay register. There are separate offsets for M0 from write select 2, and M1 from write select 3.

The final sums exit the chip. The QUADCAP section delays the outputs to allow synchronous captures at the destination. The OUTCTRL byte is written by D\_WR\_EN14. OUTCTRL is defined as follows:

OUTCTRL[1:0] specifies 0, .25, .50, or .75 bits delay of the M0SUM byte.

OUTCTRL[3:2] specifies 3,2,1, or 0 bits delay of the M0SUM byte.

OUTCTRL[5:4] specifies 0, .25, .50, or .75 bits delay of the M1SUM byte.

OUTCTRL[7:6] specifies 3,2,1, or 0 bits delay of the M1SUM byte.

Floorplanning is used for the fractional portions.

Setting Bit 7 of the CONTROL byte routes the random stream RAND directly out the LVDS port.

The signals are converted to LVDS by a TI chip, and exit the card.

An additional card (or cards) will be needed to convert the 32, 125 MHz outputs from the 32 planes, into a meaningful sum.

### ***Dataout Xilinx Design***

Refer to figures.opj/Dataout Xilinx Block Diagram.

The description in the upper right describes the Timeslot 0/Timeslot1 breakdown. Each ms, the 32 intersections in Timeslot 0, plus the 64 intersections of the appropriate Timeslot 1 are processed, for a total of 96 intersections. There are 16 separate Timeslot 1 selections, corresponding to the 16 ms in the cycle. Each intersection will cause a check of the Timeslot 0 & 1 Ram, to see if it is a transfer or a skip. If it is a skip, the microcode will immediately jump to the next intersection. If it is a transfer, the microcode will transfer the data from that intersection's ASIC.

### **Control Word Shifting**

On the RAMS Xilinx sheet, the Control Words for the ASICS are stored in the four rams on the right side of the page. Each Ram holds 4 banks, for a total of 16 banks. Each bank holds a complete set of control words for the 8 ASICS served by this Xilinx. Each of the eight Xilins serve their 8 ASICs

A shift out of the selected bank is triggered by a D\_WR\_STB17 pulse from the microprocessor. This is synchronized to become a SHIFTCW pulse. This starts a CWCNT address counter, by resetting it. CWCNT10 going low starts the generation of the CWCLK. This is gated with 62CE to give a 62.5 MHz clock. CWCLK goes to the

first asic in the chain, along with CWDATA from the selected ram. The CWBANK byte is written from the microprocessor to select the control word banks written into and read out. There are 16 banks of control words. CWCLK and CWDATA are daisy chained through the 8 Asics.

CWCLK and CWDATA come from the 8<sup>th</sup> asic back into the Xilinx. They are shifted into the ASIC Control Word Read RAM, on the bottom center of the page. The same SHIFTCW pulse as above resets the INCNT counter, which provides the address. CWCNT10 can be read by D\_RD\_EN10 to verify the control word is done shifting in. Note the ram reads the data shifted out of the asics, so it is reading one cycle earlier than what was shifted in. The microprocessor can read the ram.

The Control Words are strobed into the Asics by ASICCWSTB. See the CWSTBGEN hierarchical element on the SEQUENCER diagram. D\_WR\_EN11 sets a register. Two CE registers sync this pulse up to the 125 MHz clock.

D\_WR\_EN11 also strobes a count LOADCNT, from the microprocessor into a register. LOADCNT gets loaded into the CWCNT counter when (BLANKING and DUMPENBL) is not high. When (BLANKING and DUMPENBL) becomes high, that count clock enables the counter, anded with 62CE, to make the counter count at 62 MHz. When the counter is all 1s, TC is 1. This is pipelined and output as ASICCWSTB. This also clears the counter and registers, bringing TC back to 0. ASICCWSTB should be 16 ns wide.

The purpose of the counter delay is to allow for the various delays associated with the Blanking pulse on the Correlator Card. It is desirable for the new control word to be strobed during the blanking period for every chip on the card. The blanking pulse is delayed as it shifts up the card, 16 clock cycles after going through 8 chips. There is also the Blanking Delay line in the asic, to compensate for lateral shifts of data. This could add up to 31 more clocks. A 48 count delay covers the worst case sum of 31 plus 16. The Blanking pulse is a minimum of 64 clock wide, so it can cover a 48 clock delay.

Note the counter counts at 62.5 MHz, so 24 counts give the 48 clock delay. The count being all 1's is 63, so 63 minus 24 plus 1 is 40 decimal, or 28 hex, as the number to be loaded into the counter.

ASICCWSTB transfers the data from the Asic shift in register, to the register that controls the asic functions. The Control Word Strobe is daisy chained through the Dataout's eight asics, along the same path as the Control Word Clock and Data. The Strobe does not return to the Xilinx, since there is no double buffering of the returned word in the Xilinx. The delay of the daisy chain could result in the asics towards the end of the chain being strobed a clock pulse later than the asics at the beginning of the chain. This should not be a problem. Typically, the control word will change in the middle of a blanking cycle.

### Blanking RAM Description

See the RAMS drawing. The Blanking signal tells the ASICs to stop accumulating. Typically the signal pulses 256 clocks high every ms. The minimum length is 64



The 8 Dataout Xilinx will have different values programmed in their blanking rams, to ensure that the blanking pulses are synchronous, even though the 16MSIN pulses are offset from each other. The staggering of the 16MSIN is shown on the DATAOUTPATH2 drawing.

Delaying the Program Word Strobe pulse 48, 16 ns counts, or 768 ns assures the pulse will occur in a 128, 8 ns clock, or 1024 ns wide blanking pulse. Recall the blanking pulse could be delayed up to 48 clocks in its distribution on the board. That includes 16 clocks going up the board, and 31 clocks in an internal delay line in the asic. Thus the delay has a factor of two overkill in it, and could be lessened if short blanking pulses are desired.

### Dump Enable RAM Description

The DUMP ENABLE pulse will overlap a blanking pulse, when there is to be an accumulator reset and dump to storage in the ASICS. The timing is less critical than for the Blanking pulse, since it just has to cover the Blanking pulse.

Two banks are provided for having an active and inactive bank.

The address uses the whole 16 ms range. COUNT [15:10] form the lsbs. COUNT10 gives a resolution of  $2^{10} * 16\text{ns} = 16.384 \text{ usec}$ .

In the default values in the ucf file for bank 0, the first memory location is 1, all others are zero. That gives a Dump Enable for the first 16.384 usec of every 16 ms period. Recall the default Blanking Pulse is 2 usec wide, 4 usec in, every ms. So Dump Enable will cover the pulse every 16 ms.

### Microcode Program Description

The Program Ram contains Microcode that is executed by the sequencer. See the SEQUENCER Xilinx diagram. Each program word is 16 bits. PROGWRD[5:0] is the A or Address field. PROGWRD[15:6] selects the instruction. The instructions are as follows:

Bit 6 HOLD - Wait for MSSTB to reset the Program Counter

Bit 7 JUMPUP – Jump to the Microprocessor Address

Bit 8 JUMPSEQ – Jump to the address A (2 msbs from the microprocessor).

Bit 9(A) LDINTCTR – Load Intersection counter with value in A. 2 msbs 10.

Bit 10(A) LOOPINT- Loop back on Intersection counter, jumping to A, unless TC.  
Intersection Counter incremented.

Bit 11(A) PAUSE – Load Result Counter with value in A. 2 msbs 11.

Loop back on Result counter, freezing the Program Counter, unless TC  
Result Counter incremented. Follow by a NOOP.

Bit 12(A) LDSELCTR – Load Select counter with value in A. 2 msbs 11.

Bit 13(A) LOOPSEL – Increment Select Counter, Jump to A, unless TC.

Bit 14 RDCLKENBL – During a PAUSE issue RDCLKENBL.

Bit 15(A) JUMPXFER – If Transfer, jump to A. 2 msbs determined by microprocessor.

All of the above bits being zero gives a NOOP operation.

This table shows the program sequence.

**SEQUENCER MICROCODE**

Stored in J:\OrCAD\hwdwgs\Documentation\dopgm.xls Sheet 1  
 MSSTB pulse starts the program at step 0.

Adr	Label	Instruction	msbyte	lsbyte	To Label	Comment
0		LDINTCTR	02	20		See Note 1.
1	ILOOP	PAUSE	08	26		See Note 2. 26 instruction delay.
2		NOOP	00	00		
3		JUMPXFER	80	0A	SELLOOP	Jump if there is a transfer.
4		LDSELCTR	10	30		See Note 3.
5		LOOPINT	04	01	ILOOP	Otherwise skip point.
6		NOOP	00	00		
7		HOLD	00	40		Wait for next MSSTB pulse.
8		HOLD	00	40		Need two Holds due to pipelining.
9			00	00		Spare Location
A	SELLOOP	PAUSE	08	31		See Note 4. 15 instruction delay.
B		RDCLKENBL	40	00		
C		PAUSE	08	3D		See Note 5. 3 instruction delay.
D		NOOP	00	00		48 clock cycles per 16 results.
E		LOOPSEL	20	0A	SELLOOP	
F		NOOP	00	00		
10		LOOPINT	04	01	ILOOP	Timing ok to go back in loop.
11		NOOP	00	00		
12		HOLD	00	40		Wait for next MSSTB pulse.
13		HOLD	00	40		

Load into Memory, by insertion into the dataout Xilinx ucf file:

```
# \-F-\E-\D-\C-\B-\A-\9-\8-\7-\6-\5-\4-\3-\2-\1-\0-\
INIT_00 = 0000200A0000083D40000831000000400040000004011030800A000008260220 ;
INIT_01 = 0000000000000000000000000000000000000000000000000000000040004000000401 ;
```

See Sheet 2 for some timing related charts.

Need noop after pause, jumpxfer, loopint, loopsel

Note 1. For both time slots want 96 intersections.

A0 needs to be loaded. Since the 2 msbs are locked 10

The value = 20. The 2 msbs of the A field are part of the program code.

Note 2 A total time of 64 8ns clocks, or 32 62.5 MHz cycles is desired  
 if there is no transfer per loop. The loop overhead is 6 instructions.  
 32-6=26 =0x1A delay needed. 0x100-0x1A=0xE6.

Since the 2 msbs are locked high, the E6 value translates to 26.

Note 3 Loop 16 times for a total of 256 points xferred.

0x100 - F0 = 0x10 or 16 iterations. The 2 msbs are tied high so F0 -> 30.

LDSELCTR is executed every time JUMPXFER is, due to the pipeline delay.

Note 4 We want 16 RDCLKENBLs. Pausing for 15, gives 15 RDCLKENBLs,  
 plus one executed after leaving the PAUSE loop.

0x100-F=F1 -> 31 since the 2 msbs are high.

If not this Xilinx, RDCNTENBL switched off in INTADDRESSMUX

Note 5 We want SELLOOP to output 16 RDCLKENBLs every 48 clock cycles or 24  
 instructions. There is the 15 instruction delay in the second pause,  
 + 6 instructions overhead = 21 instructions.

For 3 more instructions delay: 0x100 - 3 = FD -> 3D.

Note this 16 ns pause the Select lines to settle.

There is a register at the output of the Program Ram, clock enabled by 62CE, to make the sequencer run at 62.5 MHz. This pipeline register also effects the sequence of program execution, by delaying when an instruction is executed by a 62.5 MHz instruction cycle. The following table shows this effect.

Stored in J:\Orcad\hwdwgs\Documentation\dopgm.xls Sheet 2.

adr	Lable	Inst.	Executing	Comment
0		ldintctr		
1	ILOOP	pause	ldintctr	
2		noop	pause	Loads value. TC low to freeze PC.
2		noop	noop	When TC high, PC advance.
3		jumpxfer	noop	Now TC is high, so PC advanced.
4		ldselctr	jumpxfer to SELLOOP	
5		loopint	ldselctr	
6		noop	loopint to ILOOP	
7		hold	noop	
8		hold	hold	
A	SELLOOP	pause	ldselctr	Enable rdclkenbl while pausing.
B		rdclkenbl	pause	rdclkenbl high for the pause duration.
C		pause	rdclkenbl	Plus one more rdclkenbl at the end.
D		noop	pause	Wait
E		loopsel	noop	
F		noop	loopsel to SELLOOP	
A	SELLOOP	pause	noop	
		....		
10		loopint	noop	If loopsel was done
11		noop	loopint to ILOOP	
1	ILOOP	pause	noop	Without NOOP, executing hold here would freeze pause.
2		...		
12		hold	noop	If loopint was done
13		hold	hold	
13		hold	hold	

Need noop, rdclkenbl, or ldselctr, after pause, jumpxfer, loopint, or loopsel  
 Need two hold instructions in a row, to counteract pipelining.

Notice that instructions that effect the program counter (PAUSE, JUMPXFER, LOOPINT, and LOOPSEL) require special consideration. Due to the instruction pipelining, the instruction following the will be executed before any branching can take place. Thus it is desired that the instruction following (such as NOOP, RDCLKENBL, LDSELCTR) not effect the program counter.

See the LOADMUX Xilinx diagram. Initially, the MSSTB pulse (which occurs each millisecond), causes the program counter to have its 5 lsb's reset, and the 3msbs

determined by the UPADDRESS register. This allows for 8 separate programs to be in the program memory, each up to 32 instructions. This starts the program at step zero.

The first instruction on dopgm.xls is the LDINTCTR instruction. This loads the intersection counter with the value in A (A0), via pulsing the Load signal. The intersection counter will be the index for the intersection loop. The loop will be closed by the LOOPINT instruction. It will keep looping until the intersection counter reaches FF. Then the TC output will disallow the JINTLOOP instruction.  $FF-A0 + 1 = 60$  or 96 decimal, for the 96 intersections in a ms's Timeslot 0 plus Timeslot 1.

The next section of the program is labeled ILOOP. If an intersection is being skipped, it is desired for the program to take 64, 8ns clock cycles to process the intersection. Thus some stalling is required. To do the stalling, we PAUSE 26 instructions. The 26 instructions are in addition to the PAUSE and the following NOOP. Here the result counter is used strictly as a timer. The PAUSE instruction freezes the program counter until the result counter TC goes high. 100 hex – E6 = 1A or 26 decimal. Since all CE signals are enabled by 62CE, to make things run at 62.5 MHz, the 26 translates to a delay of 52 clocks. The remainder of the loop is six cycles, or 12 125 MHz clocks, if there is no transfer. Note the NOOP following LOOPINT is executed as part of the loop.

The JUMPXFER(A) command jumps to A if a transfer is indicated by the value in the Timeslot 0&1, transfer ram. JUMPXFER is anded with TRANSFER from the Timeslot 0&1 Ram. The transfer signal is based on the Intersection number and the ms count. LDINTCTR initialized the intersection number.

LDSELCTR 30 to loop 16 times for a total of 256 points transferred. The LTA input data rate requirement is 16 results every 48 clock cycles. This is implemented by having 16 results in 32 clock cycles, surrounded by 16 clock cycles padding. The SEL counter counts out the 16 blocks of 16 results. 30 is obtained by  $0x100 - 0x10 = F0$ . The two msbs are tied high, so 30 is used. Reloading the Select counter on ILOOPS without a transfer does not hurt anything. Note that due to the address pipelining, the LDSELCTR instruction will be executed every time there is a JUMPXFER command, regardless of whether there is a transfer.

If a transfer is not required, the LOOPINT ILOOP instruction increments the intersection counter, and loops back to ILOOP. This is accomplished by CE ing the intersection counter, and generating a JINTLOOP, to load the program counter with the value in A. The JINTLOOP signal is gated with TC from the intersection counter, so that if the count is complete, the jump does not take place.

When the intersection looping is complete, HOLD disables the program counter, and it will stay frozen until the next MSSTB pulse, at the beginning of the next ms. Two HOLD instructions are always needed in a row for the hold to work, due to the instruction pipelining.

At SELLOOP, doing a PAUSE, followed by a RDCLKENBL has the result counter actually loading results. Pausing for 15 causes 15 RDCLKENBL pulses to be generated during the pause time, then one more to be generated when the instruction following the PAUSE is executed. 16 results are loaded from the selected ASIC into the Xilinx on the way to the LTA.

PAUSE and NOOP, are used to stall similar to above. The time spent not sending RDCLKENBL should be 16 clock cycles. Three instructions loop overhead take up 6 clock cycles.  $0x100-3 = FD$ . Since the 2 msbs are hardwired high, 3D is loaded.

Note this slack time is used to advantage. The test mode using TESTADR (See INTADDRESSMUX) has the CE to the TESTADR counter delayed 8 bits. This allows the timing for TESTADR to be the same delays as with the Asics in place. The SEL counter is not delayed 8 bits, since it also feeds the Asics. The above delay takes up the slack, so the SEL timing is good in both cases.

LOOPSEL then increments the Select counter and loops back, unless the TC indicates the counter is complete. This is similar to the LOOPINT command. SELECT3 and SELECT2 specify which 64 lag block to address, to the Asics.

LOOPINT then loops back to ILOOP, as it did above to run out the Intersection Counter.

HOLD then waits for the next MSSTB.

### Intersection Counting

See figures/Dataout Xilinx Block Diagram

Each ms, the intersection counter must count through intersections 0 to 31 for Timeslot 0, and then through 0 to 63 for Timeslot 1. The intersection accessed in Timeslot 1 rotates each ms, through the 16 ms cycle.

The Intersection Counter is on the SEQUENCER Xilinx Schematic.

In the program, the intersection counter is initially loaded with A0. Refer to the INTADDRESSMUX Xilinx diagram. INT[7:0] is the intersection count from the Intersection Counter. INT6 is used for TS1SEL, Timeslot 1 Select, to state we are using Timeslot 1.

A0 = 1010 0000

INT= 7654 3210

So note TS1SEL is initially 0, for Timeslot 0.

We now count the 32 counts of Timeslot 0.

$A0 + 20$  (decimal 32) = C0 = 1100 0000.

So now INT6 = 1 for Timeslot 1.

We now count 64 counts of Timeslot 1.

$C0 + 40$  (decimal 64) = 100 which is the end of the count cycle.

## Initial Intersection RAM Setting

For the default value for Bank 0, we want the 32 Timeslot 0 values to skip, or be zero. We want the 64 Timeslot 1 values to be 1. Using the above scheme, we want the last 64 locations of the 128 to be set to 1. An INIT block of 256 locations would have the values for two milliseconds. For example:

```
INIT_00=FFFFFFFFFFFFFFFF0000000000000000FFFFFFFFFFFFFFFF0000000000000000;
```

For the 16 milliseconds in Bank 0, the above would be repeated eight times.

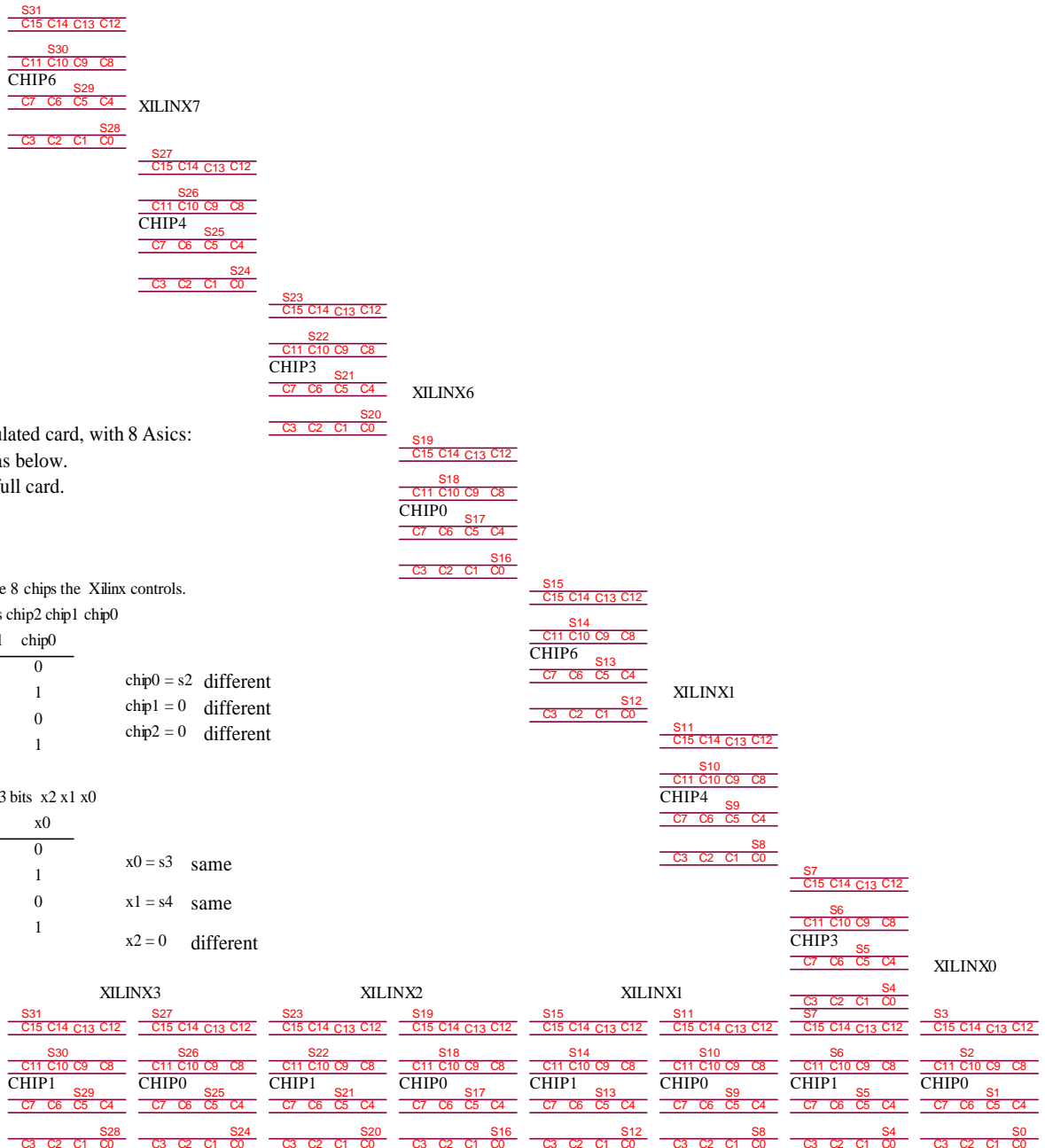
For the default value for Bank 1, we want the 32 Timeslot 0 values to transfer, or be one. We want the 64 Timeslot 1 values to be zero. Using the above scheme, we want the first 64 locations to be one. We want the last 64 locations of the 128 to be set to 0. Note the first 32 locations are actually don't care. An INIT block of 256 locations would have the values for two milliseconds. For example:

```
INIT_08=0000000000000000FFFFFFFFFFFFFFFF0000000000000000FFFFFFFFFFFFFFFF;
```

For the 16 milliseconds in Bank 1, the above would be repeated eight times.

The below figure shows the sequence and coordinate conversion for Timeslot 0. The intersection counter sequences S0 through S31 along the diagonal. The output is the Xilinx number (referring to which of the 8 Dataout Xilinx), the chip Number (referring to which of the 8 Asics controlled by that Dataout Xilinx), and the intersection number (which of 16 intersections in the Asic). Also shown is the sequence for a partially populated card, with 8 asics along the bottom row. The bit from a microprocessor register XID4 is set to 1 for a partially populated board. The control logic is implemented on INTADDRESSMUX.

TIMESLOT 0 INTERSECTION MAP



For the partially populated card, with 8 Asics:  
The TS0 sequence is as below.  
Cx is identical to the full card.

CHIPx equals which of the 8 chips the Xilinx controls.

Each CHIPx is 3 bits chip2 chip1 chip0

s3	s2	chip2	chip1	chip0	
0	0	0	0	0	
0	1	0	0	1	chip0 = s2 different
1	0	0	0	0	chip1 = 0 different
1	1	0	0	1	chip2 = 0 different

XILINXx = 3 bits x2 x1 x0

s4	s3	x2	x1	x0	
0	0	0	0	0	
0	1	0	0	1	x0 = s3 same
1	0	0	1	0	x1 = s4 same
1	1	0	1	1	x2 = 0 different

For the Fully populated card, go up the diagonal as defined below.

In Timeslot 0 the Intersection counter counts from 0 to 31

Cx indicates the intersection number in the ASIC.

Each Cx is 4 bits c3 c2 c1 c0

Sx = timeslot0 intersection number

Sx = 5 bits s4 s3 s2 s1 s0

This gives the following addresses:

s1	s0	c3	c2	c1	c0
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	1	1

c0 = c2 = s0  
c1 = c3 = s1

CHIPx equals which of the 8 chips the Xilinx controls.

Each CHIPx is 3 bits chip2 chip1 chip0

s3	s2	chip2	chip1	chip0
0	0	0	0	0
0	1	0	1	1
1	0	1	0	0
1	1	1	1	0

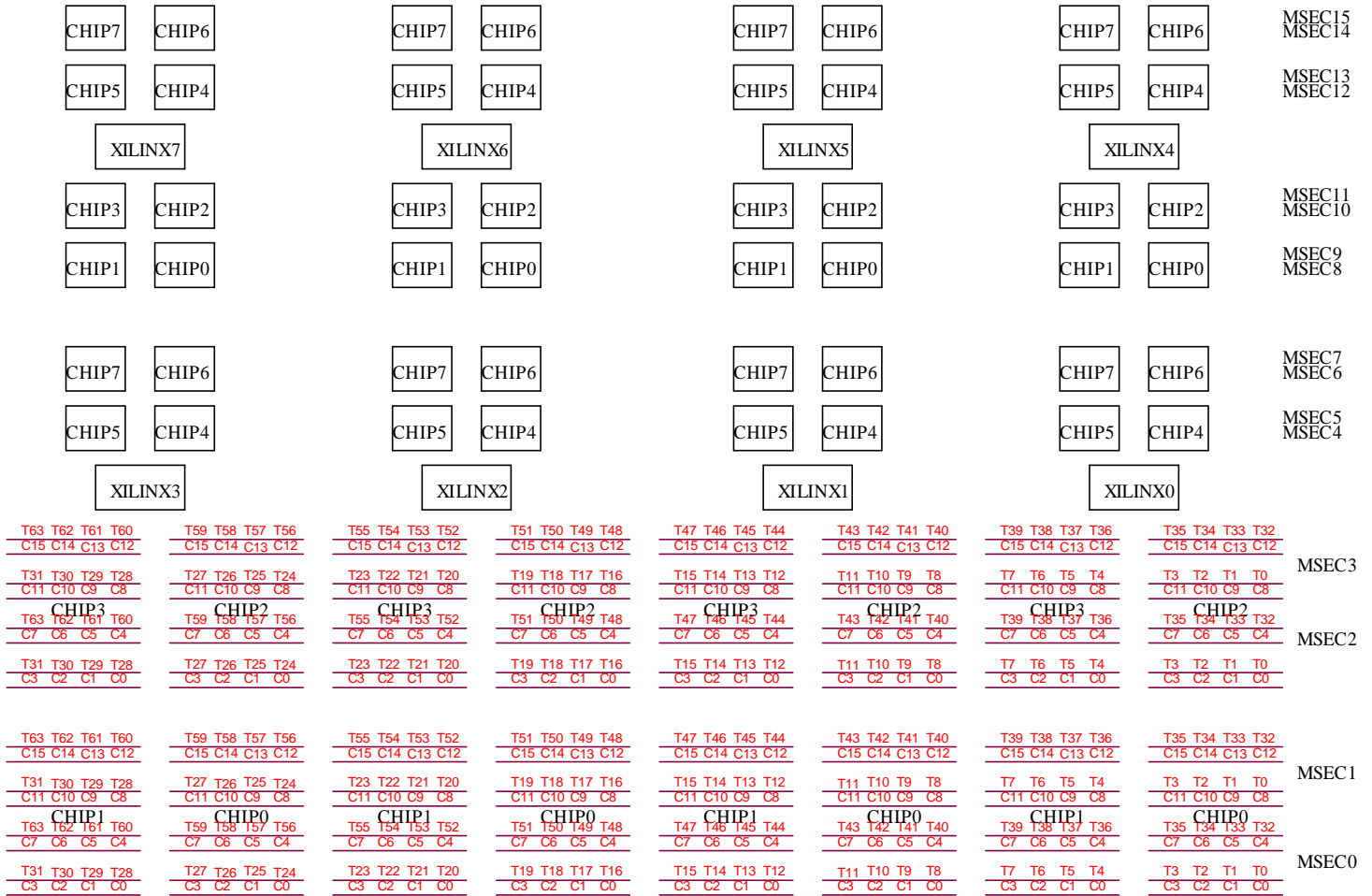
chip0 = s2 \* s3  
chip1 = s2  
chip2 = s3

XILINXx = 3 bits x2 x1 x0

s4	s3	x2	x1	x0
0	0	0	0	0
0	1	0	0	1
1	0	1	1	0
1	1	1	1	1

x0 = s3  
x1 = x2 = s4

The chart on the next page shows the Intersection Map for Timeslot 1. This shows the sequence and coordinate conversion for Timeslot 1. The intersection counter is the T0 through T64 for each ms. Note for a partially populated card, only MSEC0 and MSEC1 will be associated with any Asics. The rest of the ms's can access empty logic if desired. Alternately, only the intersections with data could be accessed by appropriate use of the Transfer/Skip Ram for Timeslot 1. (That is of the array of 16 intersections in a ASIC, Transfer the four in the upper left and the four in the lower right.) There is no change to the logic required for the partially populated board. Note the same intersections that were accessed in Timeslot 0 are also accessed in Timeslot 1. The Transfer/skip programming from the LTA will have to be sure to not try to read from the same location twice. The control logic is implemented on INTADDRESSMUX.



**INTERSECTION ADDRESSING SEQUENCE**

**OUTER LOOP**

Count MSEC0 to MSEC15

**INNER LOOP**

Count T0 to T63

CHIPx equals which of the 8 chips the Xilinx controls.

Each CHIPx is 3 bits chip2 chip1 chip0

Cx indicates the block number in the ASIC.

Each Cx is 4 bits c3 c2 c1 c0

Tx indicates the intersection out order for timeslot 1.

Each Tx is 6 bits t5 t4 t3 t2 t1 t0

MSECx = 4 bits ms3 ms2 ms1 ms0

XILINXx = 3 bits x2 x1 x0

c0 = t0

c1 = t1

c2 = t5

c3 = ms0

chip0 = t2

chip1 = ms1

chip2 = ms2

x0 = t3

x1 = t4

x2 = ms3

**TIMESLOT 1 INTERSECTION MAP**

**Data Output Paths**

Refer to the Xilinx sheet DATAOUTPATH2.

Each Dataout Xilinx services 8 asics. There are 4 asics in parallel in the BOTASICS bus and the other 4 asics in parallel in the TOPASICS bus. The sequencer sends an independent RDCLKEN to the appropriate asic, to generate a 16 bit output stream at 62.5 MHz.

The block OUT16TO8 converts the 16 bit, 62.5 MHz data stream to an 8 bit, 125 MHz data stream, if the microprocessor bit XID3 equals 0. The savings is that only 8 data lines to the LTA at 125 MHz are needed, instead of 16 data lines at 62.5 MHz. If XID3 equals 1, the 8 LSBs are output at the 62.5 MHz, and the 8 MSBs are lost. Only 8 bits are routed through the Xilinx regardless.

Having the multiplexed data duplicated in the MSByte and LSByte is useful in the test fixture for testing the LTA. It provides a data source for LTA inputs that would be vacant, since the test fixture would not have a full complement of eight Correlator Cards. In the full system, zeroing the MSByte saves power.

Back on DATAOUTPATH2 an 8 bit, Pseudo Random Number can be switched in at this point by setting XID6 equal to one. This comes from the standard block DATAGEN, as used in the Analog Sum Xilinks. Note the 125 MHz random data will have different random numbers in the two bytes.

Each of the 8 Dataout Xilinks has its sequencer going through the same program. The RDCLKENBL signal (which causes data to actually be read from Asics) is only generated if the microprocessor supplied XID, matches the Xilinx number the sequencer has progressed to. A demultiplexer generates the 8 RDCLKENBL signals to the 8 asics, based on the Chip ID. This logic is on the INTADDRESSMUX page.

When the RDCLKEN signal reads an asic, it is routed through the Xilinx along to the LTA. The figure on DATAOUTPATH2 shows the routing of the data. The top row of Xilinks all output to the Xilinx directly below it. The Xilinks on the bottom send the data stream to the right. Each Xilinx adds its data to the data from the previous stream of Xilinks and outputs to the next Xilinx. In Xilinx 0, the combined data stream from all eight Xilinks is routed off the card to the LTA, via an external LVDS driver. When in random number mode, the random streams should merge seamlessly.

In order for the data streams from the 8 Xilinks to add seamlessly together, the sequencers in the Xilinks have to be running offset from each other, to allow for the pipelining in the adder tree. This staggering is accomplished by staggering the 16MSIN pulse, which initializes the sequencers. The figure on DATAOUTPATH2 shows how this is accomplished. The 16MSIN signal comes from the LTA and goes into Xilinx 3. The number next to the signal shows the number of clock cycles delay the signal has in the various Xilinks. These match up with the pipeline delays in adding the Xilinx outputs together to get everything synchronized.

## Synchronization

The time delays from the 16 ms pulse to data output are calculated as follows:

Refer to the DATAOUTPATH2 diagram. Clocks refer to 125 MHz clocks.

2 clocks- 16MSCARD through ASUMMID

1 clock - delay to get the 16MSIN pulse into Datout3.

16 clock - delay by the time it becomes 16MSSYNC in Dataout0.

1 ms from 16MSSYNC to first MSSTB. Note the 16<sup>th</sup> MSSTB from the previous cycle acts as the first cycle of the next 16 ms period.

2 clocks – MSSTB loads address for starting program.

2 clocks – Program pipelining out of Ram

64 clock – Wait at the beginning of the intersection.

16 clock – Wait during the 48 clock cycle.

INTADDRESSMUX shows the timing of the RDCLKENBLs to the Asics. Since the data from the Asics is clocked into the Xilinx at 62.5 MHz. The RDCLKENBL needs to be timed to get the output on the correct edge of 62CE. The extra pipeline stage to give XRCE1 provides that timing. The delays to generate the clock enable to the TESTADR counter, allow the same timing to be used for the Asics as the TESTADR.

Delays from XRCE0 are:

2 clock – Delay for RDCLKENBL to leave Xilinx.

3 clocks – Delay in Asic – 62.5 MHz Data out (See Figure 5 of the Asic Spec for timing)

2 clock- BOTASICS or TOPASICS input register inside Xilinx. (CE at 62.5 MHz)

Here the data joins with the TESTADR simulated data.

## High Bandwidth Mode

The bandwidth can be doubled by having all 16 bits outputting data at 125 MHz, instead of the current 8 bits.

There is a 16 bit bus going to pins from Dataout Xilinx 3 that can be also used to output data. This bus does not have an LVDS driver as the bus from Dataout Xilinx 0. Thus it would need an external driver board.

These changes would require different Xilinx personalities.

Thus there is the potential for quadrupling the current data output rate.

## Register Programming

### BANK[7:0] Programming

The BANK[7:0] bits are written into each Dataout Xilinx from the microprocessor via D\_WR\_EN13. The bits are assigned as follows:

BANK0 – Selects between the two banks of the TIMESLOT0 and 1 Ram.

Note this bit only is double buffered and strobed by ASICCWSTB to make it synchronous with the Control Word changes.

BANK1 – Selects between the two banks of the ASIC Control Word Read Ram.

BANK4 – Selects between the two banks of the BLANKING Ram.

BANK5 – Selects between the two banks of the DUMP ENABLE Ram.

## ASIC Control Word Programming

The CWBANK[7:0] bits are written into each Dataout Xilinx from the microprocessor via D\_WR\_EN9. CWBANK selects which of the 16 banks of ASIC control words are written into or read out. The bits are assigned as follows:

CWBANK[3:0] selects which of the 16 banks are byte wise written in, or read out of the rams, by the microprocessor. The read out is to check the contents. D\_WR\_EN7 writes the control words to the selected bank using the incrementing UP\_ADR counter. D\_RD\_EN7 allows reading the selected bank back into the microprocessor.

CWBANK[7:4] selects which of the 16 banks provide the serial control words to the asics. D\_WR\_STB17 starts the counter CWCNT that sends the selected control word to the ASICs.

D\_WR\_EN11 provides the strobe pulse to strobe the control word which has been shifted into the asics, into the Asic's active control word register.

When control words are shifted out, the last ASIC in the chain shifts the Control Word that was in the ASIC, into a Dataout Ram. This Ram can be read by D\_RD\_EN8.

## UPADDRESS[7:0] Programming

The UPADDRESS 8 bit word is written into each Dataout Xilinx from the microprocessor via D\_WR\_EN2. This 8 bit address has two applications. It can be used in the JUMPUP microcode instruction to go to a specific address in the program register.

When a HOLD instruction is issued. The program waits for the next MSSTB to be internally generated. Then the program branches to the address specified by UPADDRESS[7:5], with the 5 lsb's being zero. This allows for eight different programs to be present. The program loaded from the ucf file starts at location zero.

## SEED-IN[7:0] Programming

The SEED-IN 32 bit word is written into each Dataout Xilinx from the microprocessor via D\_WR\_EN12. This serves as the 32 bit seed for the random number generator. This is shifted in as four bytes, LSByte first.

## XID[15:0] Programming

The XID word is written into each Dataout Xilinx from the microprocessor via D\_WR\_EN3, as two bytes, LSByte first. The bits are assigned as follows:

XID[2:0] – Xilinx ID. This tells which of the eight Dataout Xilinxs this one is. Note this value will be different for all eight Dataout Xilinxs. Looking at the back of the card, the Dataout Xilinxs are arranged:

4 5 6 7  
0 1 2 3

XID3 – If 1, jumper control word read ram clock and data directly from signals which drive asics. 0 has the data coming from the asics.

XID4 – Partially Populated. 1 if a partially populated board, with only eight Asics, placed on the bottom row. 0 if the board is fully populated with Asics.

XID5 – spare

XID6 – Random Numbers. 1 for data from 16 bit random number generator, replacing the data from the Asics. 0 for normal operation.

XID7 – Sequential Numbers. 1 for sequential numbers replacing the data from the Asics. 0 for normal operation.

XID[9:8] – Output Quadrature. Select degrees of quadrature output delay of the 8 bits to the LTA. 00 gives 180 degrees. 01 gives 270 degrees. 10 gives 360 degrees. 11 gives 450 degrees.

XID[11:10] – Output bits delay. Select bits delay of the output to the LTA, where 11 gives three bits delay, 10 two bits delay, 01 one bit delay, 00 zero bits delay.

XID12 – This forces the XADD Xilinx address value to 0. This is useful for forcing constant Random numbers out of Dataout 0, to the LTA. Setting XID6 turns on the random numbers. XID[2:0] =0 specifies Dataout 0. Thus XID = 0001 0000 0100 0000 = 0x1040.

XID[15:13] spares

Note that while XID[2:0] are unique for each Dataout Xilinx. XID[7:3] will probably be programmed the same for the eight Dataout Xilinxs. XID[11:8] only have significance for Dataout Xilinx 0, since only that one outputs to the LTA. XID[11:8] is don't care for the other Dataout Xilinxs.

## Random Test Address Mode

As can be seen in the DATAOUTPATH2 Xilinx schematic, setting XID6 to 1 switches to random numbers output, instead of whatever is read from the ASICs. These will be 125

MHz, 16 bit wide numbers. Notice the data streams from the individual Dataout Xilinx are merged together as before.

### Sequential Test Address Mode

Setting XID7 to 1 on DATAOUTPATH2 can be seen to switch to the TESTADR field instead of the data from the ASICs. This field is generated in the SEQUENCER schematic. The lower 8 bits are the 64 result addresses in an intersection. The upper 8 bits represent the intersection count. The intersections first count through the 32 diagonal intersections of Timeslot 0. Then they count through the 64 intersections of Timeslot 1. Recall different sets of 64 intersections are scanned based on the millisecond number.

Having XID3 set to 0 multiplexes the 16 bit, 62.5 MHz TESTADR field onto the eight lsb's which are sent to the LTA at 125 MHz. For Dataout Xilinx 1 through 7, this data is output via the REGOUT bus. The various Dataout Xilinsxs add in their contributions when their Asics are being read.

In Dataout Xilinx 7, the final summation is output via the FINALOUT bus to the LTA. In Dataout Xilinx 7, XID7 being set to 1 also switches the millisecond number, MSEC[3:0] onto the lower four of the REGOUT bus. Note the millisecond number is a low speed signal, changing only after the 96 intersections have been processed, therefore the quadrature output is not necessary. In the test fixture, these eight lsb's will be cabled to an LTA input that would normally come from another correlator card. The LTA has inputs for eight Correlator Cards, but there are only two Correlator Cards in the test fixture.

The sequential addresses from the eight Dataout Xilinsxs will be merged to the common output stream. If working correctly, there should be sequential numbers at the transition points.

### Internal Logic Analyzer

A Block Ram is used as a Logic Analyzer, to record the sequence of program instructions executed by the sequencer. On the sheet SEQUENCER the Logic Analyzer Counter counts the address, based on the PROGCE that enables the Program Counter. The difference is that this counter will not loop back at all. D\_WR\_STB17 is used to reset the counter. The counter will stop counting when all ones. The Ram is in the drawing RAMS. The eight bit field stored in the Ram contains: PROGCTR[3:0], PROGWRD[9:8], LDPAC, TRANSFER. PROGCTR[3:0] identifies which instruction of the 16 instruction program is executed. PROGWRD[9:8] gives two bits of the Program instruction associated with that address. LDPAC give the Load Pulse for the Program Address Counter, used to skip to a new address. TRANSFER from the Transfer Ram, tells whether to transfer or skip that intersection. The Ram is read by the microprocessor.

### Test Point Output Multiplexer

D\_WR\_EN14 writes the TPCTRL[7:0] word. This goes to TESTMUXS to output various signals to testpoints. The upper four Dataout Xilinsxs have available the DUMPENBL, BLK0 and BLK1 outputs, since they are only used on the bottom Xilinsxs.

These signals go to the top test connector. The Xilinx ID, XID, controls the function of these signals.

The currently unused pins 86 and 126 also go to the top test connector on all but Dataout Xilinx 6 and 7. These pins are SP33B[1:0] on the Correlator Card Schematic.

## **1.8 Volt Power Supply Mezzanine Board**

### ***Introduction***

See d:/orcad/corr\_card/powersupplies.dsn for the schematic of the mezzanine board that does the 48V DC to 1.8V DC conversion. Since 80 Amps is required per board, this provides a practical way to distribute the current. This board will be bolted to the main board. The bolts and copper standoffs provide the path for the output current. There is a ribbon cable connector for control signals.

### ***Power Sequencing***

Of concern is the Power Sequencing conditions in the Virtex-E chips on the Correlator Board. 3.3V (Vcco) should not be applied prior to 1.8V (Vint) for more than one second. The ONOFF lines to the 1.8 Volt DC to DC converters will be jumpered so they are always on. The PWRCTRL signal is generated to tell the Bin Power Supply Board that 1.8 Volt Sense is active. Note 1.8 Volt Sense is the voltage sensed from the main board. PWRCTRL is open drain. All the Correlator Cards in the bin have their PWRCTRL signals wire anded together. If any Correlator Card is not generating 1.8 Volt, the Bin Power Supply Card can turn off the 3.3 Volt Supply.

### ***Power Monitoring***

An eight position, analog mux allows monitoring the following analog signals:  
Current from each of the three DC to DC converters,  
V1.8 Volt Sense, V5.0, V48-MON, and V3.3,  
An analog temperature from a temperature chip.

The Mux's 3 bit position comes from the off card. The current is derived from the amplified voltage across shunt resistors. V48-MON is V48 scaled through a voltage divider. MUXOUT leaves the card.

### ***Current Sharing***

The three, half brick 1.8 Volt Power Supplies current share. The ISHARE and START SYNC lines facilitate current sharing. Each supply can supply 60 Amps, for a total of 180 Amps. The Correlator Board is anticipated to require 80 Amps. Therefore if one supply dies, the remaining two may keep things going. The Current measurements for each supply will indicate the bad supply.

# Correlator Card/LTA Test Fixture

## Introduction

The Correlator Card/LTA Test Fixture interfaces an LTA, two Correlator Cards, and a Bin Power Supply Card.

See corrltatestfix.dsn, for the test fixture schematic.

See the below drawing for the layout.

The switches on the left provide the source of the 1.8 Volts for each Correlator Card. The choices are the Mezzanine card, an external bus, or the Bin Power Supply Card.

## Power Distribution

The Test Fixture will have a slot for a Bin Power Supply Card. This is the same card that provides 5 V, 1.8 V, and 3.3 V in each Correlator Card/LTA bin. The card will provide 1.8 V at 60 A, 3.3 V at 50 A, and 5 V at 2 A.

