VLBI Phased Array iBob and BEE2 Programmer's Guide

David MacMahon

February 3, 2012

Abstract

Describes the registers and devices in the various iBob and BEE2 designs used in the VLBI phased array processor for CARMA and SMA.

1 Gateware Versions

The history of the gateware designs has two major epochs: real and complex. The real epoch covers the original designs, which processed the signal as real (i.e. non-complex) data. Due to limitations in dealing with real data, namely lack of phase information and control, the gateware designs were modified to convert the real signal into complex form. These designs dealing with complex data comprise the complex epoch.

Within each epoch, registers and devices have been added, removed, and renamed. Instead of attempting to describe all revisions of the gateware, this document only describes specific versions of the gateware as noted in each section. The complete history for the various designs are in a Git repository. The Git repository is available using one of these URLs:

`git://github.com/sma-wideband/phringes.git`

`http://astro.berkeley.edu/~davidm/phringes.git`

2 iBob Phased Array (IPA)

This section documents the ibob_phased_array_2k_2012_Feb_02_1601 design.

In addition to synchronizing internal timing signals with external timing signals. The ibob_phased_array_2k design has a few critical registers that must be set to non-zero values for proper operation. These registers are briefly listed here; full details are provided below.

gain0 - gain3 [R/W registers]
At least one gain register must be non-zero to get non-zero output through to the DBE.

2.1 Synchronization

Different observatories use different synchronization and phase switching schemes. The ibob_phased_array_2k design accommodates the schemes used at SMA and CARMA. Additionally, the design requires an external 1 PPS pulse with which it synchronizes internally generated sync signals used with downstream systems.

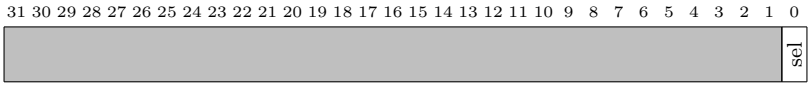
SMA phase switching is based on a Heartbeat (HB) signal with frequency $\frac{52\,\mathrm{MHz}}{2^{19}} \approx 99.182\,\mathrm{Hz}$. This signal is generated on board from the 256 MHz FPGA clock and is synchronized with the external SMA Start of Walsh Frame (SOWF) signal using the `armsowf` command. An internal SOWF signal is also generated on board synchronized with the external SOWF signal. The frequency of SOWF is $\frac{52\,\mathrm{MHz}}{2^{25}} \approx 1.550\,\mathrm{Hz}$.

CARMA phase switching is based on a 1024 PPS signal. This signal is generated on board from the 256 MHz FPGA clock and is synchronized with the external 1 PPS signal using the `arm1pps` command. An internal 1 PPS signal is also generated on board synchronized with the external 1 PPS.

The design also uses the internally generated 1024 PPS and 1 PPS signals for other purposes (e.g. XAUI synchronization and synchronous noise source seeding) independent of the phase switching scheme being used, so the external 1 PPS must always be supplied and the internal 1 PPS must always be synchronized to it via the `arm1pps` command.

ppssel [R/W register]

Selects the ADC sync pulse to which the internally generated 1 PPS signal will be synchronized.



0 Select ADC0 sync signal (typically for CARMA)

1 Select ADC1 sync signal (typically for SMA)

onepps_sync/diff [R register]

This register captures the difference, in FPGA clock cycles, between the external 1 PPS signal and the internally generated 1 PPS signal. Normally this register will read 0, but since the external 1 PPS signal is not synchronous with the ADC sample clock, values of 1 or 256,000,000 are also acceptable. Other values indicate a problem.

onepps_sync/ext_cnt [R register]

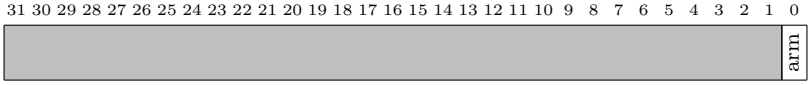
This register contains the number of rising edges seen on the external 1 PPS signal since the last re-arming of the internal 1 PPS signal.

onepps_sync/ext_period [R register]

This register contains the number of FPGA clock cycles between the two most recent rising edges seen on the external 1 PPS signal. It should read 256,000,000 $\pm$ 1.

onepps_sync/arm1pps [R/W register]

A 0 to 1 transition of bit 0 of this register will arm the internal 1 PPS generator to re-synchronize with the next rising edge of the selected sync pulse.



This register is not often used directly since (re-)arming is normally done via the `arm1pps` command.

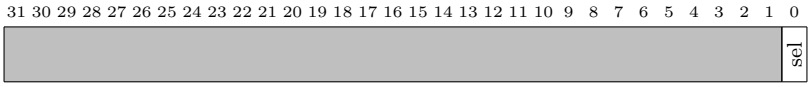
onepps_sync/arm1pps_ack [R register]

A 1 to 0 transition of bit 0 of `onepps_sync/arm1pps` clears (i.e. sets to zero) this register.

Bit 0 of this register is set to 1 on the rising edge of the next sync pulse that re-synchronizes the internal 1 PPS generator. This can be used to verify that the internal 1 PPS generator did indeed re-synchronize after being armed. This register is not often used directly since (re-)arming is normally done via the `arm1pps` command.

sowfsel [R/W register]

Selects the source of the external SOWF signal to which the internally generated SOWF signal will be synchronized.



0 Select GPIO 0 bit index 8 (aka J6 pin 17)

1 Select ADC0 sync signal

hb_sync/diff [R register]

This register captures the difference, in FPGA clock cycles, between the external SOWF signal and the internally generated SOWF signal. Normally this register will read 0, but since the external SOWF period is not commensurate with the ADC clock period, values of 1 or 165,191,049 (plus or minus a few clock cycles) are also acceptable. Other values indicate a problem.

hb_sync/ext_cnt [R register]

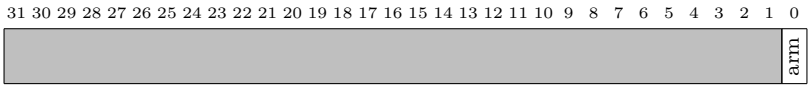
This register contains the number of rising edges seen on the external SOWF signal since the last re-arming of the internal SOWF signal.

hb_sync/ext_period [R register]

This register contains the number of FPGA clock cycles between the two most recent rising edges seen on the external SOWF signal. It should read 165,191,049 ± 2.

hb_sync/arm_sowf [R/W register]

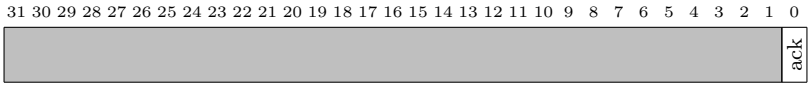
A 0 to 1 transition of bit 0 of this register will arm the internal HB and SOWF generators to re-synchronize with the next rising edge of the external SOWF signal.



This register is not often used directly since (re-)arming is normally done via the `armsowf` command.

hb_sync/armsowf_ack [R register]

A 1 to 0 transition of bit 0 of `hb_sync/arm\_sowf` clears (i.e. sets to zero) this register.



This register is set to 1 on the next rising edge of the external SOWF signal. This can be used to verify that the internal HB and SOWF generators did indeed re-synchronize after being armed. This register is not often used directly since (re-)arming is normally done via the `armsowf` command.

uptime [R register]

The uptime register is a 32 bit counter that counts seconds since the FPGA started running. It cannot be reset (other than reprogramming the FPGA) and is not synchronized with any other 1 PPS source.

2.2 Input Selection

insel [R/W register]

Input selection. Uses two bits per time sample (i.e. demux path) per input.

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
in3 t3	in3 t2	in3 t1	in3 t0	in2 t3	in2 t2	in2 t1	in2 t0	in1 t3	in1 t2	in1 t1	in1 t0	in0 t3	in0 t2	in0 t1	in0 t0

0 Select ADC for one demux path of one input

1 Select noise source for one demux path of one input

2 Select constant 0 for one demux path of one input

Examples:

0x00000000 Select ADC for all demux paths of all inputs

0x00000001 Select noise source for demux path 0 of input0, select ADC for all other demux paths

0x00000055 Seelct noise source for all demux paths of input 0, select ADC for all other inputs

0x55555555 Select noise source for all demux paths of all inputs

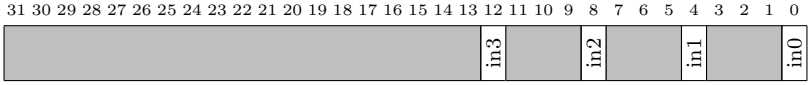
0x55000000 Select noise source for all demux paths of input 3, select ADC for all other inputs

0xA0000000 Select constant 0 for demux paths 2 and 3 of input 3, select ADC for all other inputs

2.3 Noise Generators

noise/arm [R/W register]

A 0 to 1 transition of bit 0 of each nybble arms that input's noise generator.

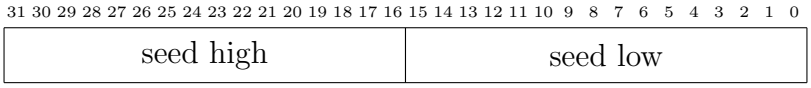


The least significant nybble corresponds to input0. A value of 0x0011 will arm the noise generators for input0 and input1. A value of 0x1111 will arm all noise generators.

Once a noise generator is armed, it will re-seed starting on the next internal 1 PPS rising edge. This can be used to synchronously seed noise generators across multiple devices or within one device.

noise/seed/0
noise/seed/1
noise/seed/2
noise/seed/3 [R/W registers]

Sets 32 bit seed for noise generators for corresponding inputs. Because the inputs are demultiplexed by 4, each noise generator actually contains two noise generators that each output two independent normally distributed values each. The 32 bit seed is actually split into two 16 bits seeds to seed the two contained noise generators. Using a seed whose two halves are identical (i.e. a multiple of 65537) will result in a signal that has a lag-2 correlation coefficient of 1 resulting in a non-flat spectrum.

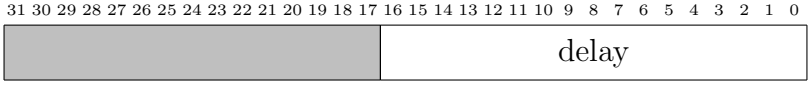


2.4 Delay Core

The Delay Core delays the input signal and converts it from real to complex representation. The delay can range from 0 to $8191\frac{15}{16}$ samples in increments of $\frac{1}{16}$ of a sample. The delay is implemented as a combination of coarse delay (in units of FPGA clock cycles), fine delay (in units of ADC sample clock cycles), and sub-sample delay (in units of $\frac{1}{16}$ an ADC sample clock cycle). The sub-sample delay is implemented using a quadrature fractional delay FIR filter that not only does sub sample delay, but also the conversion to complex form. Converting to complex form involves mixing by a local oscillator at $\pm F_s/4$. The sign of the LO frequency is user selectable and determines whether the Delay Core's complex output will consist of the positive or negative frequencies of the real input. Selecting the negative frequencies will result in a frequency reversal and conjugation relative the positive frequencies. See the `sbsel` register for more details.

delay0
delay1
delay2
delay3 [R/W registers]

The 17 least significant bits of these registers specify the "constant" (i.e. rarely changing) delay to be applied by each input's Delay Core. The overall delay that is applied may optionally include an additional time-varying delay from the delay LUTs.



The value written into these registers can be computed as

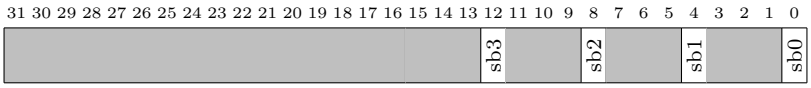
$$\begin{aligned} register\_value &= 16 \cdot adc\_sample\_delay + 64 && (\mathrm{mod}\ 2^{17}) \\ &= \mathrm{round}(16.384 \cdot ns\_delay) + 64 && (\mathrm{mod}\ 2^{17}) \end{aligned}$$

For example, to obtain a delay of 123.456 ns, the value to use would be computed as

$$\begin{aligned} register\_value &= \text{round}(16.384 \cdot 123.456) + 64 \pmod{2^{17}} \\ &= 2087 \end{aligned}$$

Because of the 64 sample offset and modulo 2^{17} behavior, minimum delay is obtained with a value of 64, while maximum delay is obtained with a value of 63.

sbsel [R/W register]
The `sbsel` (sideband select) register specifies whether the complex output will consist of the positive or negative frequencies of the real input.

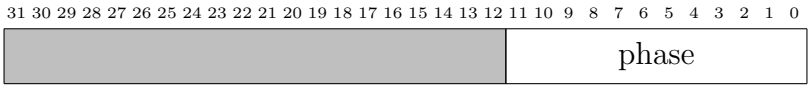


Each of the four least significant nybbles (i.e. hex digits) of `sbsel` corresponds one input. The least significant nybble specifies the sideband selection for input 0. A value of 0 selects the positive frequencies; a value of 1 selects the negative frequencies. Writing 0x0100 will select positive frequencies for inputs 0,1,3 and negative frequencies for input2.

2.5 Phase Rotation

After the Delay Core's conversion to complex, the signal passes through a phase rotator. The amount of phase rotation applied is a combination of the phase offset introduced by fine delay, the phase value specified in the input's `phaseN` register, and the 0/90/180/270 degree phase switching specified by the input's current phase switching state. The phase offset introduced by fine delay is automatically compensated; no user input is required. The `phaseN` registers are described here and the phase switching states are described in a separate subsection.

phase0
phase1
phase2
phase3 [R/W registers]
These registers specify the amount of user-specified "constant" (i.e. rarely changing) phase rotation to be applied to each input's complex signal. The overall phase offset that is applied may optionally include an additional time-varying phase offset from the phase LUTs. These phase offsets are independent phase switching, which is handled separately.



Because only the 12 least significant bits of these registers are used for one complete turn in phase, the resolution is $\frac{2\pi}{2^{12}}$ radians per step or $\frac{360}{2^{12}}$ degrees per step (i.e. approximately 88 millidegrees per step).

The value written into these registers can be computed as

$$register\_value = \text{round}\left(phase\_radians \cdot \frac{2^{12}}{2\pi}\right)$$

or

$$register\_value = \text{round}\left(phase\_degrees \cdot \frac{2^{12}}{360}\right)$$

For example, to phase rotate a given input by 30 degrees, the value to use would be computed as

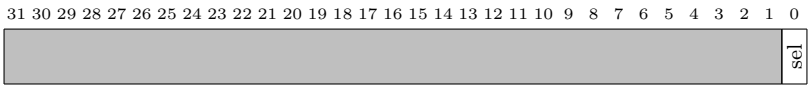
$$\begin{aligned} register\_value &= \text{round}\left(30 \cdot \frac{2^{12}}{360}\right) \\ &= 341 \end{aligned}$$

2.6 Phase Switching

As mentioned elsewhere, the design supports both SMA and CARMA phase switching schemes. Both observatories use nested Walsh functions whereby the 180 degree phase switching cycle completes once per 90 degree phase switching state. The Wash functions themselves are stored in two shared block RAM lookup tables. One table is for the 180 degree phase switch cycle and the other table is for the 90 degree phase switching cycle. Each entry (i.e. row) in a Walsh table is 32 bits wide. Each bit (i.e. column) corresponds to a given Walsh function. Each input uses a user specified column for both the 90 and 180 degree Walsh functions. To provide the simplest and most versatile interface, the Walsh tables are stored in unrolled form.

walsh/syncsel [R/W register]

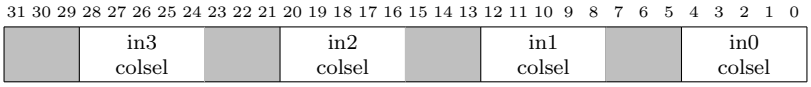
This register is used to select the timing of the Walsh functions.



A value of 0 selects the internal 1024 PPS (for CARMA). A value of 1 selects the internal HB signal (for SMA).

walsh/colsel [R/W register]

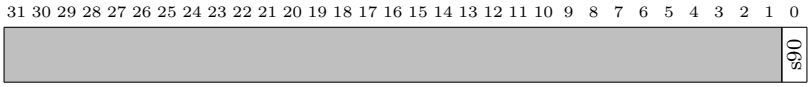
This register is used to specify which columns (i.e. bits) of the Walsh tables to use for each input.



Each input is controlled by one byte of this register. Input 0 uses the least significant byte of this register. Only the five least significant bits of each byte are used to select columns 0 through 31. Column 0 is the most significant bit of the 32 bit Walsh table entries.

walsh/sign90 [R/W register]

The least significant bit of the `walsh/sign90` register specifies the sign of 90 degree phase switch in the upper sideband.



The signal and its 90 degree phase switching may be conjugated multiple times before the 90 degree phase switching gets undone (for the sky upper sideband) in the phase rotator. The conjugations are caused by lower sideband downconversions, including sampling the second (or any even) Nyquist zone and setting `sbsel=1`, both of which are essentially lower sideband down conversions. If the total number of conjugations is odd, then the final signal and its 90 degree phase switching will be conjugated, which will require rotating the phase in the opposite direction as compared to the original, non-conjugated signal.

0 The analog 90 degree phase switching is assumed to be a +90 degree phase rotation in the USB and a -90 degree phase rotation in the LSB. The complex signal is phase rotated by -90 degrees, which rotates the USB to 0 degrees and the LSB to 180 degrees.

1 The analog 90 degree phase switching is assumed to be a -90 degree phase rotation in the USB and a +90 degree phase rotation in the LSB. The complex signal is phase rotated by +90 degrees, which rotates the USB to 0 degrees and the LSB to 180 degrees.

walsh/table/90
walsh/table/180 [Block RAM]

These shared block RAMs must be programmed with the unrolled Walsh tables. Each table (i.e. BRAM) is 2048 entries long. Each entry is 32 bit wide. Each bit position corresponds to a column of a Walsh table. The most significant bit of each 32-bit entry corresponds to column 0 of the (up to) 32 column Walsh table.

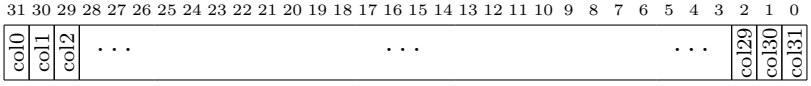


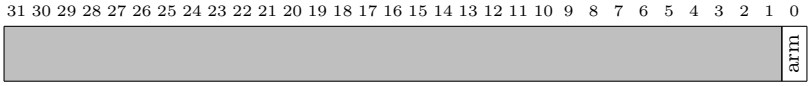
Table `walsh/table/90` is for the 90 degree phase switching states.
Table `walsh/table/180` is for the 180 degree phase switching states.

The Walsh tables are addressed by a counter that increments every phase switching state (i.e. every 1024 PPS for CARMA; every HB for SMA) and is reset at the end of every Walsh function cycle (i.e. every 1 PPS for CARMA and every SOWF for SMA). Only the first N entries need to be populated, where N is the number of phase switching states per complete phase switching cycle (i.e. 1024 for CARMA, 64 for SMA).

2.7 Look-Up Tables

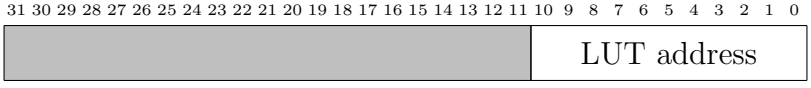
The delay and phase offsets required to properly sum the signals together change over time. Often these changes must be applied more frequently and with greater timing precision (e.g. on 0/180 degree phase switch boundaries) than is possible over the limited interface of the iBobs. To address this problem, the design provides both delay and phase lookup tables (LUTs) which are stepped through at a predetermined rate to provide precisely timed delay and phase tracking. The use of these LUTs is optional. A custom command, `loadlut`, has been added to the iBob interface to facilitate rapid pre-population of the delay and phase lookup tables, thereby avoiding the interface performance limitations of the iBob. The outputs of the LUTs are registered on every address increment so changing the value of the currently addressed LUT location will not change the current overall delay or phase offset.

lut/arm [R/W register]
A rising edge of the least significant bit of this register synchronizes the LUT address counter with the start of the next Walsh cycle.



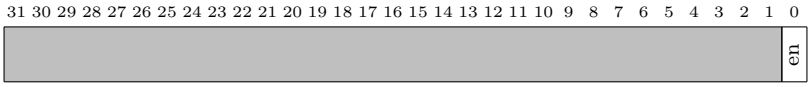
The delay and phase LUTs are addressed by a common counter that increments after every 32 (CARMA) or 64 (SMA) Walsh states, depending on the setting of the `walsh/syncsel` register. The LUT address counter is synchronized with the start of the next complete Walsh cycle after a rising edge of the least significant bit of the `lut/arm` register. Knowing the absolute time of LUT address synchronization and the rate at which the LUT address increments, the controlling software can compute the LUT addresses at which to store upcoming delay and phase tracking values.

lut/addr [R register]
The current value of the 11-bit LUT address counter.



lut/enable [R/W register]

Controls whether the LUT output values are included in the overall delay and phase offsets applied to the input signals.



When the least significant bit of this register is set to 1, the LUT values will be added to the corresponding `delayN` and `phaseN` offsets applied to the input signals, otherwise they will not be added. Note that the LUT address counter always increments regardless of whether the LUT outputs are enabled.

lut/delay/0
lut/delay/1
lut/delay/2
lut/delay/3 [Block RAM]

These shared block RAMs contain the delay LUT values for each of the four inputs. Each LUT is 2048 entries long. The 18 least significant bits of each 32-bit entry is treated as a two's complement value with four fractional bits providing a delay LUT offset ranging from $-8192\frac{0}{16}$ to $+8191\frac{15}{16}$ ADC samples. This allows one to set the "constant" `delayN` offset registers to a midrange value and then use the delay LUTs to go plus or minus the midrange values. When `lut/enable` is 1, the overall delay (in units of $\frac{1}{16}$ of an ADC sample) is given by

$$\texttt{delay}N + \texttt{lut\_delay\_}N[\texttt{lut\_addr}] \pmod{2^{17}}$$

The result will always be treated as a positive value, so care must be taken to avoid unintended overflow/wrapping of the 17-bit unsigned input range of the Delay Core.

lut/phase/0
lut/phase/1
lut/phase/2
lut/phase/3 [Block RAM]

These shared block RAMs contain the phase LUT values for each of the four inputs. Each LUT is 2048 entries long. The 12 least significant bits of each 32-bit entry is treated as an unsigned phase value in units of $\frac{2\pi}{2^{12}}$ radians. When `lut/enable` is 1, the overall phase offset, in units of $\frac{2\pi}{2^{12}}$ radians, is given by

$$\texttt{phase}N + \texttt{lut\_phase\_}N[\texttt{lut\_addr}] \pmod{2^{12}}$$

loadlut [TinySh Command]

Over time intervals of a minute or less, the delay and phase variations that remain in the sampled IF signal can be approximated with negligible error by a linear function. The `loadlut` command can be used to pre-populate a range of consecutive locations within the delay and phase LUTs with such a linear function. The syntax of the `loadlut` command is:

```
loadlut LUT_NAME START_ADDR START_VALUE STEP COUNT SHIFT
```

Where:

`LUT_NAME` is the name of the LUT to load (e.g. `lut/delay/0`).

`START_ADDR` is the address at which the first value will be stored.

`START_VALUE` is the first value to be stored.

`STEP` is the increment between each successive value.

`COUNT` is the total number of values to store.

`SHIFT` is the number of bits to right shift the value before storing.

The LUT addresses are always treated modulo the LUT size. This "wrap around" behavior simplifies pre-populating a series of LUT entries that would otherwise overrun the end of the LUT.

The `SHIFT` parameter allows `START_VALUE` and `STEP` to be given with higher precision than that of the LUT itself.

The values and locations at which they are stored are given by

$$\texttt{LUT\_NAME}[(\texttt{START\_ADDR} + n)\%2048] = (\texttt{START\_VALUE} + n \cdot \texttt{STEP}) \gg \texttt{SHIFT}$$

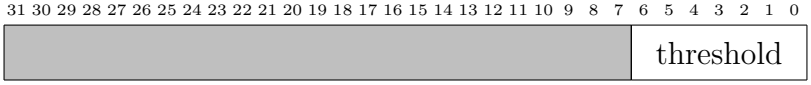
where n ranges from 0 to $\texttt{COUNT} - 1$.

2.8 2-Bit Data Path

After the Delay Core and Phase Rotation, the signals are sent to the calibration correlator and the DBE. The calibration correlator gets 2 bit data (at the ADC sample rate) from all inputs. The quantization to 2 bits is controlled by user specified threshold values. The 2 bit coding represents four distinct values: -3, -1, +1, +3.

quant/thresh0
quant/thresh1
quant/thresh2
quant/thresh3 [R/W registers]

These registers set the thresholding level used for quantizing each input to 2 bits. When using the digital noise sources, 16 is a suitable value. For CARMA IFs, 22 is a suitable value.



The quantized value for a given input value of x and unsigned 7 bit threshold value $thresh$ is determined according to the equations shown here. Note that the output value for a zero valued input alternates between -1 and +1.

$$\begin{aligned} x <= -thresh &\Rightarrow -3 \\ -thresh < x < 0 &\Rightarrow -1 \\ x = 0 &\Rightarrow \pm 1 \\ 0 < x < thresh &\Rightarrow +1 \\ thresh <= x &\Rightarrow +3 \end{aligned}$$

When using a digital noise source as an input, a threshold value of 16 results in a $\left\{\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{1}{6}\right\}$ distribution in the 2 bit quantized values.

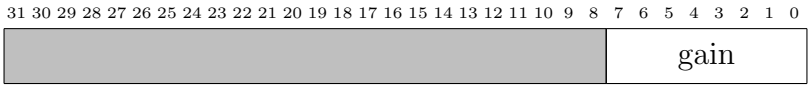
2.9 8-bit Data Path

After the Delay Core and Phase Rotation, the signals are sent to the calibration correlator and the DBE. Before being sent to the DBE, the four 8 bit input signals are summed together then scaled and rounded back to 8 bits. Before the sum each input signal is multiplied by a gain factor. This gain factor is intended to weight the inputs according to system temperature, but thus far they have been used only to scale all inputs equally or to zero out an input to exclude it from the sum.

The sum is scaled by $\frac{1}{2}$ or $\frac{1}{4}$ (see the `adder/shift` register) and then rounded to 8 bits before being sent over the XAUI links to the Digital Back End (DBE). The unbiased rounding mode known as "round-to-even" is used. Scaling by $\frac{1}{2}$ and rounding to even will result in three times as many even values as odd values, assuming the least significant bit is uniformly distributed. Scaling by $\frac{1}{4}$ and rounding to even will result in $\frac{5}{3}$ times as many even values as odd values, assuming the two least significant bits are uniformly distributed.

gain0
gain1
gain2
gain3 [R/W registers]

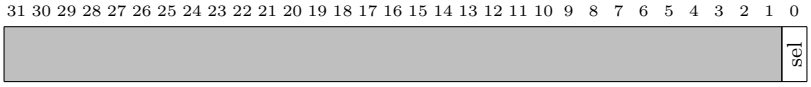
These registers control the gain factor for each input.



Only the 8 least significant bits are used. They are currently interpreted as an unsigned number with 7 fractional bits. Thus, the range of gain values that can be applied to each input is 0 to $1\frac{127}{128}$.

adder/shift [R/W register]

This register select the scaling factor applied to the sum before re-quantizing to 8 bits.



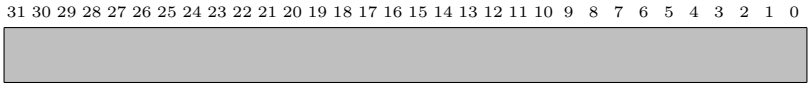
Set this to 0 to scale the sum by $\frac{1}{2}$ (e.g. for independent noise dominated inputs). Set this to 1 to scale the sum by $\frac{1}{4}$ (e.g. for coherent inputs). The current gateware is setup to wrap instead of saturate when scaling by $\frac{1}{4}$. Future gateware versions may change this arguably erroneous behavior.

2.10 XAUI

The sum of the phased inputs is sent to the DBE via the XAUI0 connector. The 2-bit quantized inputs are sent to the calibration correlator via the XAUI1 connector. XAUI data transmission is always enabled whenever the corresponding XAUI link is up.

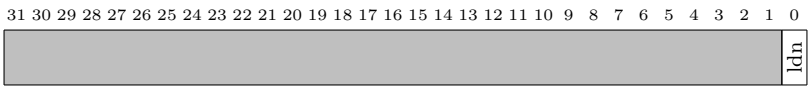
start_xaui [R/W register]

This register is deprecated. It used to control transmission over the XAUI links and generation of the clock that is sent to the BEE2 calibration correlator. These functions are now enabled automatically. They do not require, and function better without, user intervention. This register exists solely to maintain backwards compatibility with existing software.



xaui_0/rx_linkdown
xaui_1/rx_linkdown [R registers]

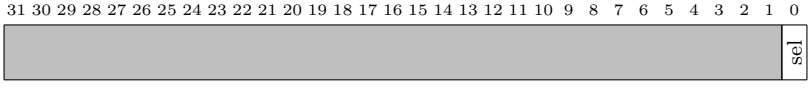
A value of 1 indicates a "link down" status on the (unused) receive side of the indicated XAUI connection, which strongly implies a similar status for its transmit side.



2.11 Front Panel SMA Output

smasel [R/W register]

Selects which signal to output over the front panel `SMA0` (upper) connector.

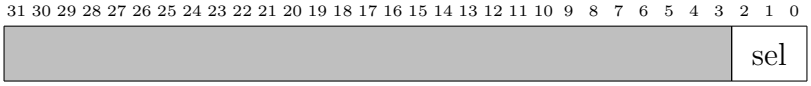


0 Outputs a 128 MHz square wave (50% duty cycle).

1 Outputs a constant 0.

monsel [R/W register]

Selects which signal to output over the front panel `SMA1` (lower) connector.



0 Outputs a constant 0.

1 Outputs the internal 1024 PPS signal.

2 Outputs the internal Heartbeat (HB) signal.

3 Outputs the internal Start of Walsh Frame (SOWF) signal.

4 Outputs the internal 1 PPS signal.

5 Outputs the sampled sync0 signal (from ADC0)

6 Outputs the sampled sync1 signal (from ADC1)

2.12 Front Panel LEDs

LED0 Not used.

LED1 Not used.

LED2 Briefly blinks on each rising edge of the sync0 input.

LED3 Not used.

LED4 Not used.

LED5 Not used.

LED6 Not used.

LED7 Not used.

LED8 Indicates FPGA is configured (non-functional on some iBobs).

3 BEE2 Calibration Correlator (BCC)

This section documents the bee2_complex_corr_floating_2011_Feb_17_2045 design.

The bee2_complex_corr design receives 2+2 bit samples (i.e. 2 real bits and 2 imaginary bits per sample) from a total of eight inputs (two IPAs). It cross correlates one of these signals (user selectable) with itself and the other seven inputs. Each of the 7 cross-correlations and 1 auto-correlation consists of 16 lags (-8 to +7). In addition to the data samples, the design also receives sync and 90 degree phase state information from the IPAs over the XAUI links' out of band channel. This allows the correlator to perform sideband separation, producing both upper and lower sideband lag spectra for each baseline.

3.1 Input numbering

The four signals arriving over XAUI1 are considered to be input0 through input3. The four signals arriving over XAUI0 are considered to be input4 through input7. Note that the higher number XAUI port contains the lower number inputs and vice versa.

3.2 Input values

The four distinct values of the 2 bit samples represent signal values of -3, -1, +1, and +3. The threshold registers in the ibob_phased_array_2k design should be set such that approximately 1/6 of the samples are -3, 1/3 are -1, 1/3 are +1, and 1/6 are +3. With this distribution, the lag 0 auto-correlation will have a mean of 11/3 and a variance of 128/9.

3.3 Integration limit

With an input signal distribution as described above, a one second (i.e. 1024×10^6 samples) integration will have a mean of $\frac{11}{3}\cdot1024\times10^6$ and a variance of $\frac{128}{8}\cdot1024\times10^6$. This integrated mean is more 3.75×10^9, which is well over 2.1475×10^9, the maximum value supported by a signed 32-bit value. To delay overflowing, the BRAM vector accumulators are 36 bits wide. Their output values are rounded to 32 bits wide before writing them into shared BRAMs.

Another factor of 2 savings is obtained by dropping the LSB of the sub-integrations. The input values are all odd so their products will all be odd too. Since an even number of odd products are integrated before the BRAM vector accumulators, the least significant bit will always be 0, so it is dropped without any loss of precision.

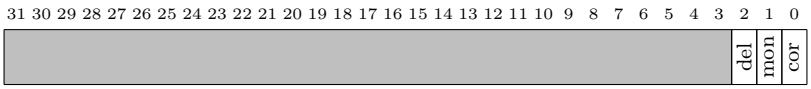
These two techniques result in scaling down the integrations by a factor of 32. This allows integrations as long as 18 seconds without overflowing the 36 bit vector accumulators (or the 32 bit shared BRAMs). This limit is only for highly correlated signals such as the lag 0 auto-correlation or coherent noise sources. Inputs dominated by independent noise (e.g. astronomical cross correlations) have far higher maximum integration times.

3.4 XAUI Alignment and Monitoring

The alignment of the two incoming XAUI streams is critical to achieving proper correlation. The design will automatically align the two XAUI streams and, if ever necessary, re-align them at the next 1024 PPS. This ensures that the two streams are never misaligned for more than 1/1024 of a second. Additionally, the out-of-band sync signals sent over the XAUI link are monitored for any anomalous behavior. The registers involved in the XAUI alignment and monitoring are described here.

xaui_rst [R/W register]

Resets various parts of XAUI related logic.



Setting bit 0 (cor) to 1 will reset the XAUI core itself (rarely needed). Setting bit 1 (mon) to 1 will reset the counters associated with XAUI monitoring. Setting bit 2 (del) to 1 will reset the registers that capture the relative delays of the two XAUI links. These bits are level sensitive, not edge sensitive, so they must be cleared to take the associated circuitry out of reset.

xaui0/linkdown_cnt
xaui1/linkdown_cnt [R registers]

Indicates how many times the link has gone down (i.e. transitioned from up to down). Under normal operations this register should read 0, though it may increment some during system startup. Reset by bit 1 of the `xaui_rst` register.

xaui0/period
xaui1/period [R registers]

Indicates the number of FPGA (256 MHz) clock cycles between the most recently received 1024 PPS pulse and the previous one. Under normal operations, this register should read 250000. Reset by bit 1 of the `xaui_rst` register.

xaui0/period_err
xaui1/period_err
xaui0/period_err_cnt
xaui1/period_err_cnt [R registers]

The XAUI monitoring logic constantly monitors the number of FPGA cycles between 1024 PPS pulses. It counts the number of period errors and remembers the most recent erroneous period value. The `xauiN/period_err_cnt` registers indicate the total number of erroneous periods detected. Under normal operations, these registers should read 0. The `xauiN/period_err` registers indicate the most recent erroneous period value. Under normal operations, these registers should read 0 (i.e. no erroneous period measured). These registers are reset by bit 1 of the `xaui_rst` register.

xaui0/sync_cnt
xaui1/sync_cnt [R registers]

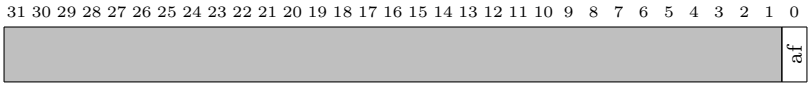
This register indicates the total number of 1024 PPS pulses received. Since it is a 32 bit counter, it will roll over approximately every 48.5 days. Reset by bit 1 of the `xaui_rst` register.

xaui0/rx_empty_cnt
xaui1/rx_empty_cnt [R registers]

Indicates how many times the XAUI FIFO has underflowed. Under normal operations this register should read 0, though it may increment some during system startup. Reset by bit 1 of the `xaui_rst` register.

xaui0/almost_full
xaui1/almost_full [R registers]

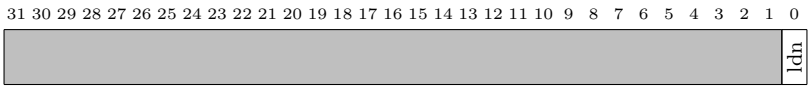
Indicates the current state of the XAUI's `almost_full` status bit.



Under normal operation, this register should read 0.

xaui0/rx_linkdown
xaui1/rx_linkdown [R registers]

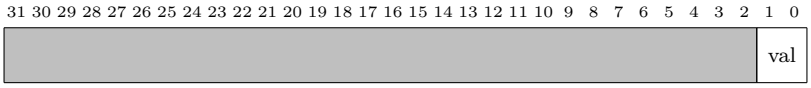
Indicates the current state of the XAUI's `link_down` status bit.



Under normal operation, this register should read 0.

xaui0/valid
xaui1/valid [R registers]

The two least significant bits of this register contain the current and previous value of the XAUI block's `valid` status bit.



Since this status bit should toggle every clock cycle, this register should read 1 or 2 under normal operation.

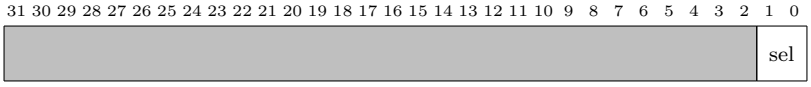
xaui_aligner/delay0
xaui_aligner/delay1 [R registers]

These registers indicate the number of cycles from the stream's incoming 1024 PPS and the reference 1024 PPS. Since the reference 1024 PPS is just a 32 cycle delayed version of XAUI0's 1024 PPS, `xaui_aligner/delay0` should always read 32. Under normal operations, `xaui_aligner/delay1` should be 32 ± 32, but usually it will be 32 ± 4.

3.5 Synchronization and Integration

In order to perform sideband separation, the correlator must integrate for an integral number of Walsh cycles. The registers used to control and monitor synchronization of the integrations are described here.

syncsel [R/W register]
This register selects the sync signal used to drive integration timing.



The sync signal selection determines the units of the `integ_time` register. The four choices are:

0 Use the 1024 PPS signal to define the integration units.

1 Use the HB signal to define the integration units.

2 Use the SOWF signal to define the integration units.

3 Use the 1 PPS signal to define the integration units.

integ_time [R/W register]
This register sets the integration time in units of the sync signal selected via the `syncsel` register. Changing the value of this register resets the correlator's internals as well as the `integ_cnt` register.

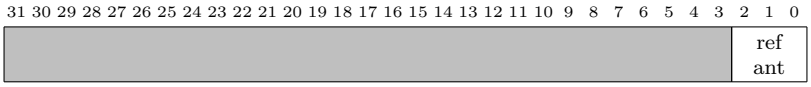
integ_cnt [R register]
This register simply counts the number of integrations that have completed. When this register increments, a new integration is ready to be read. Changing the `integ_time` register resets this register.

sample_cnt [R register]
The correlator always integrates in multiples of 32 samples (since there are 16 lags per baseline and 2 samples per FPGA clock cycle). If the selected integration unit (see `syncsel`) is not commensurate with 32 samples (e.g. the Heartbeat signal) then the number of samples per integration can vary from integration to integration. This register provides the number of 32-sample sub-integrations in the most recently completed integration. It can be used to properly average the data. Since the integrated values are pre-divided by 32 in the FPGA, the output values can be divided by this register value directly to get the average of the integrated products.

subint_cnt [R register]
This register indicates how far along the current integration is. It is in units of the sync signal selected via the `syncsel` register. It counts from 0 to $integ\_time - 1$, then starts over.

refant [R/W register]

This register selects the reference *input* (NB: *not* antenna) that is common to all eight of the computed baselines, including the auto-correlation "baseline".



rx0/usb_real
rx0/usb_imag
rx0/lsb_real
rx0/lsb_imag
⋮
rx7/usb_real
rx7/usb_imag
rx7/lsb_real
rx7/lsb_imag [Block RAM]

These shared BRAMs hold the integrated 16 lag spectra for the upper and lower sidebands of each of the eight computed baselines. The `rx0/usb_real` BRAM contains the integrated real component of the upper sideband (USB) of input0 times the complex conjugate of the reference input. The `rx7/lsb_imag` BRAM contains the integrated imaginary component of the lower sideband (USB) of input7 times the complex conjugate of the reference input. Note that some baselines can have a conjugation convention that differs from MIRIAD's conjugation convention depending on which input is chosen to be the reference input and which antennas are connected to which inputs. The 16 lags are stored from lag -8 to lag +7.

phsw_bal/0
⋮
phsw_bal/7 [R registers]

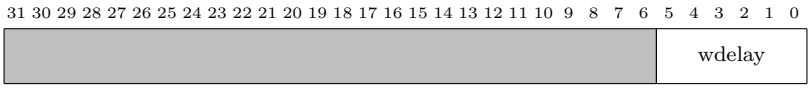
These registers indicate the *phase switching balance* for the most recently completed integration for each of the baselines. During the integration, a counter (accumulator, really) counts +1 for each sample where the inputs were in the same 90 degree phase switch state and -1 for each sample where the inputs were in different 90 degree phase switch states. For optimum sideband separation, the inputs should have an equal number of samples while in the same and different 90 degree phase switching states, so ideally this output should read zero. If it is positive, then there that many more samples with the same 90 degree phase switch state. If it is negative, then there that many more samples with different 90 degree phase switch state. Note that the count is in FPGA cycles, which is half the number of data samples (due to demux-by-2 data).

uptime [R register]

The uptime register is a 32 bit counter that counts seconds since the FPGA started running. It cannot be reset (other than reprogramming the FPGA) and is not synchronized with any other 1 PPS source.

wdelay [R/W register]

This register adjusts the delay applied to the Walsh inputs to align them with the data out from the correlator.



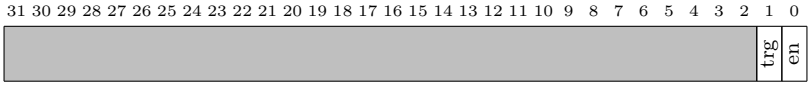
The total delay needed is calculated to be 65, but to accommodate errors in the calculation, a variable delay element is provided. The range of valid values for this register are 0 to 63. Some of the estimated delay is built in to the design already. The estimated value required for this register is 32. Once a good value is determined empirically, it will be hard-coded into the design and this register will be removed.

3.6 Snapshot Data

The design includes a snapshot block that can be used to capture some input samples to help set the upstream threshold registers and other diagnostic purposes.

snap/ctrl [R/W register]

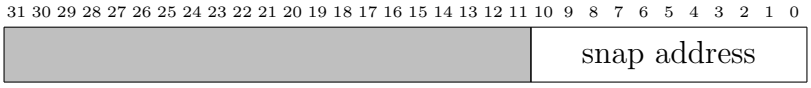
This register controls the snapshot block.



A 0 to 1 transition of bit 0 (en) enables a snapshot sequence. Once the snapshot sequence is enabled, the data capture will begin immediately if bit 1 (trg) is 1 or on the next selected sync pulse (see `syncsel`) if bit 1 (trg) is 0.

snap/addr [R register]

During data capture, this register provides the address within the shared BRAMs to which the data are being written.



When the data capture is complete, the address holds at the final value of 2047. The data capture completes in 8 µs, so this register is useful primarily when the data capture is setup to begin on the next sync pulse (otherwise it is unlikely that any value other than 2047 will be observed in this register).

snap/in0123
snap/in4567 [Block RAM]

These shared BRAMs hold the snapshot data. Each BRAM is 2048 words deep and stores data for the four inputs implied by its name. Within a BRAM, each 32 bit word stores two complex samples for all four inputs. The most significant byte holds two complex samples of the BRAM's lowest numbered input; the least significant byte holds two complex samples of the BRAM's highest numbered input. Within a byte, the two most significant bits represent the real component of the first (in time) of the byte's two complex samples. Thus, each shared BRAM can hold 4096 complex samples for four inputs. The storage format for the first word (BRAM address 0) of `snap/in0123` is shown in the following diagram (`snap/in4567` is similar).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
in0		in0		in0		in0		in1		in1		in1		in1		in2		in2		in2		in2		in3		in3		in3		in3	
re0		im0		re1		im1		re0		im0		re1		im1		re0		im0		re1		im1		re0		im0		re1		im1	

4 XAUI-based Digital Back End (DBE)

This section documents the dbe_xaui_corr_2012_Feb_02_2334 design.

The dbe_xaui_corr design receives two phased sums of four inputs each (aka partial sums) from two different iBob Phased Array Processors via XAUI. It time-aligns these two incoming signals, sums them (producing the final sum), and then processes/formats the summed signal for recording to the VLBI data recorder via the iBob's front panel VSI connector.

The design also provides a number of additional features to facilitate system verification and other engineering purposes. Most notable among the additional features are:

- Single ADC input for a so-called "comparison antenna" with optional fringe tracking
- Digial noise source compatible with the ones in the iBob Phased Array Processor
- Delay Core compatible with the one in the iBob Processor Array Processor
- Single baseline 16-lag correlator to compare various signals
- Snapshot buffer for capturing samples of various signals
- Versatile signal path routing

During VLBI observations, the DBE is controlled by the data recorder via the iBobs RS-232 port. While all the design features are accessible over both Ethernet and serial interfaces, the data recorder typically accesses a fairly minimal subset of registers and Block RAMs.

4.1 Synchronization

The VLBI data recorder requires a 1 PPS signal. In order to provide this, the DBE must be provided an external 1 PPS sync signal (via the ADC0 board) with which it synchronizes internally generated 1 PPS and 1024 PPS signals (via the `arm1pps` command).

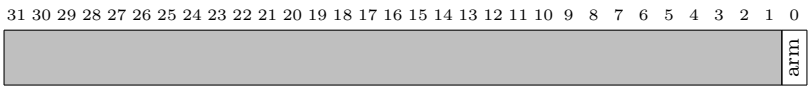
The parital sums are aligned via the 1024 PPS signals that are received over their XAUI out-of-band signals. Since their upper sidebands are already de-Walshed, the partial and final sums have no need for Walsh timing in the DBE. Currently the comparison antenna ADC input is assumed to have no phase switching so it, too, has no need for Walsh timing in the DBE.

onepps_sync/diff [R register]
: This register captures the difference, in FPGA clock cycles, between the external 1 PPS signal and the internally generated 1 PPS signal. Normally this register will read 0, but since the external 1 PPS signal is not synchronous with the ADC sample clock, values of 1 or 256,000,000 are also acceptable. Other values indicate a problem.

onepps_sync/ext_cnt [R register]
This register contains the number of rising edges seen on the external 1 PPS signal since the last re-arming of the internal 1 PPS signal.

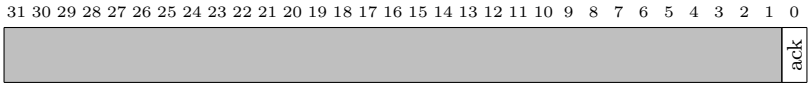
onepps_sync/ext_period [R register]
This register contains the number of FPGA clock cycles between the two most recent rising edges seen on the external 1 PPS signal. It should read 256,000,000 ± 1.

onepps_sync/arm1pps [R/W register]
A 0 to 1 transition of bit 0 of this register will arm the internal 1 PPS generator to re-synchronize with the next rising edge of the selected sync pulse.



This register is not often used directly since (re-)arming is normally done via the `arm1pps` command.

onepps_sync/arm1pps_ack [R register]
A 1 to 0 transition of bit 0 of `onepps_sync/arm1pps` clears (i.e. sets to zero) this register.

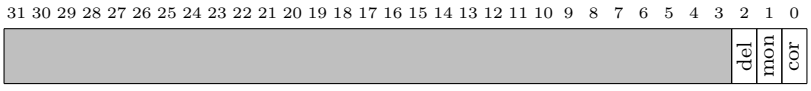


Bit 0 of this register is set to 1 on the rising edge of the next sync pulse that re-synchronizes the internal 1 PPS generator. This can be used to verify that the internal 1 PPS generator did indeed re-synchronize after being armed. This register is not often used directly since (re-)arming is normally done via the `arm1pps` command.

4.2 XAUI Alignment and Monitoring

The alignment of the two incoming XAUI streams is critical to achieving proper summing. The design will automatically align the two XAUI streams and, if ever necessary, re-align them at the next 1024 PPS. This ensures that the two streams are never misaligned for more than 1/1024 of a second. Additionally, the out-of-band sync signals sent over the XAUI link are monitored for any anomalous behavior. The registers involved in the XAUI alignment and monitoring are described here.

xaui_rst [R/W register]
Resets various parts of XAUI related logic.



Setting bit 0 (cor) to 1 will reset the XAUI core itself (rarely needed). Setting bit 1 (mon) to 1 will reset the counters associated with XAUI monitoring. Setting bit 2

(del) to 1 will reset the registers that capture the relative delays of the two XAUI links. These bits are level sensitive, not edge sensitive, so they must be cleared to take the associated circuitry out of reset.

xaui0/linkdown_cnt
xaui1/linkdown_cnt [R registers]
Indicates how many times the link has gone down (i.e. transitioned from up to down). Under normal operations this register should read 0, though it may increment some during system startup. Reset by bit 1 of the `xaui_rst` register.

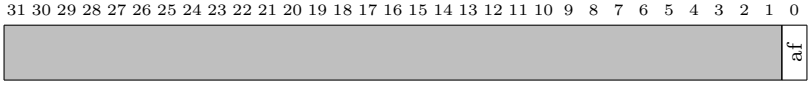
xaui0/period
xaui1/period [R registers]
Indicates the number of FPGA (256 MHz) clock cycles between the most recently received 1024 PPS pulse and the previous one. Under normal operations, this register should read 250000. Reset by bit 1 of the `xaui_rst` register.

xaui0/period_err
xaui1/period_err
xaui0/period_err_cnt
xaui1/period_err_cnt [R registers]
The XAUI monitoring logic constantly monitors the number of FPGA cycles between 1024 PPS pulses. It counts the number of period errors and remembers the most recent erroneous period value. The `xauiN/period_err_cnt` registers indicate the total number of erroneous periods detected. Under normal operations, these registers should read 0. The `xauiN/period_err` registers indicate the most recent erroneous period value. Under normal operations, these registers should read 0 (i.e. no erroneous period measured). These registers are reset by bit 1 of the `xaui_rst` register.

xaui0/sync_cnt
xaui1/sync_cnt [R registers]
This register indicates the total number of 1024 PPS pulses received. Since it is a 32 bit counter, it will roll over approximately every 48.5 days. Reset by bit 1 of the `xaui_rst` register.

xaui0/rx_empty_cnt
xaui1/rx_empty_cnt [R registers]
Indicates how many times the XAUI FIFO has underflowed. Under normal operations this register should read 0, though it may increment some during system startup. Reset by bit 1 of the `xaui_rst` register.

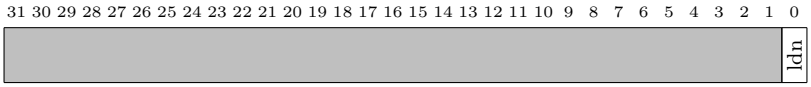
xaui0/almost_full
xaui1/almost_full [R registers]
Indicates the current state of the XAUI's `almost_full` status bit.



Under normal operation, this register should read 0.

xaui0/rx_linkdown
xaui1/rx_linkdown [R registers]

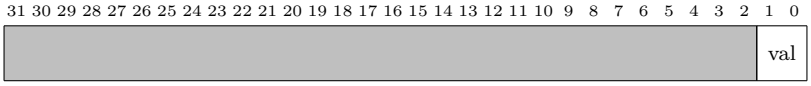
Indicates the current state of the XAUI's `link_down` status bit.



Under normal operation, this register should read 0.

xaui0/valid
xaui1/valid [R registers]

The two least significant bits of this register contain the current and previous value of the XAUI block's `valid` status bit.



Since this status bit should toggle every clock cycle, this register should read 1 or 2 under normal operation.

xaui_summer/delay0
xaui_summer/delay1 [R registers]

These registers indicate the number of cycles from the XAUI stream's incoming 1024 PPS to the internally generated 1024 PPS (delayed by `delay_adj`). A number near 250,000 indicates that the XAUI stream's sync pulse arrived shortly after the delayed internal 1024 PPS signal. If `delay_adj` is set correctly and the XAUI links are funtioning properly, these registers will read 32 ± 24. They should rarely change and even then only by a few cycles. If the vaues seem inconsistent with expectations, the XAUI core may need to be reset (see bit 0 of the `xaui_rst` register).

delay_adj [R/W register]

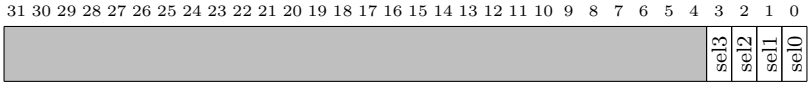
Delays the internal 1024 PPS signal so that it will align with the incoming 1024 PPS signals within the window of the alignment buffer's 64 sample window. This register is initialized to 7 at FPGA startup, but it should be set to 150 to properly align the internal sync with the delayed syncs arriving from the IPAs via the XAUI links.

4.3 Signal Path Routing

The signal path routing options control the selection of the comparison input, the output to the data recorder, and the inputs to the single baseline correlator and snap block. The inputs to the single baseline correlator are independent from the output to the data recorder. This allows, for examle, recording the final sum while correlating the two partial sums to verify that they are being properly aligned by the XAUI alignment logic.

compsel [R/W register]

Selects the comparison input signal. Either the ADC0 input or DBE's digital noise source can be the comparison input signal that passes through the Delay Core. Uses one bit per time sample (i.e. demux path), but most often the same input, either ADC0 or digial noise source, will be selected for all four demux paths. The Delay Core transforms the comparison input signal from real demux-by-4 to complex demux-by-2.

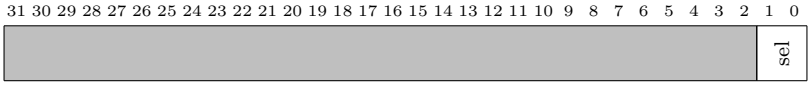


0 Selects the ADC0 input for the given demux path.

1 Selects the digital noise source for the given demux path.

vsisel [R/W register]

Selects the signal to send to the data recorder via the VSI interface.



0 Sends the final sum to the data recorder.

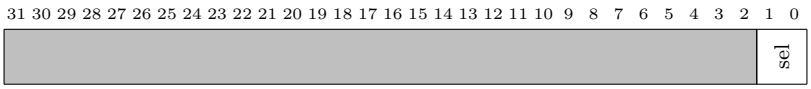
1 Sends the comparison input to the data recorder. See the `compsel` register.

2 Sends the partial sum from XAUI 0 to the data recorder.

3 Sends the partial sum from XAUI 1 to the data recorder.

corrsel0
corrsel1 [R/W registers]

Selects the input signals to the single baseline correlator. `corrsel0` selects the correlator's first input signal; `corrsel1` selects the second. The first input to the correlator is also the input to the snap block.



0 Selects the final sum.

1 Selects the comparison input. See the `compsel` register.

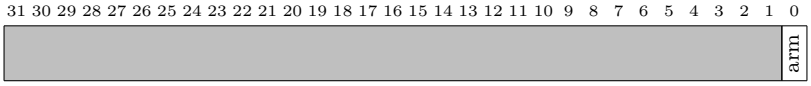
2 Selects the partial sum from XAUI 0.

3 Selects the partial sum from XAUI 1.

4.4 Noise Generator

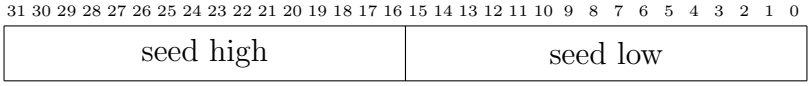
noise/arm [R/W register]

A 0 to 1 transition of bit 0 arms the digital noise generator to re-seed on the next internal 1 PPS pulse. This can be used to synchronously seed noise generators across multiple devices or within one device.



noise/seed [R/W register]

Sets the 32 bit seed for the digital noise generator. Because the inputs are demultiplexed by 4, the digital noise generator actually contains two noise generators that each output two independent normally distributed values each. The 32 bit seed is actually split into two 16 bits seeds to seed the two contained noise generators. Using a seed whose two halves are identical (i.e. a multiple of 65537) will result in a signal that has a lag-2 correlation coefficient of 1 resulting in a non-flat spectrum.

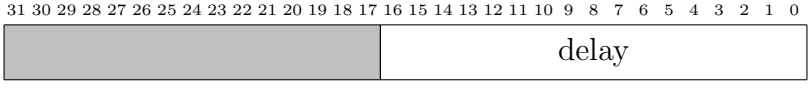


4.5 Delay Core

The Delay Core delays the comparison input signal and converts it from real to complex representation. The delay can range from 0 to $8191\frac{15}{16}$ samples in increments of $\frac{1}{16}$ of a sample. The delay is implemented as a combination of coarse delay (in units of FPGA clock cycles), fine delay (in units of ADC sample clock cycles), and sub-sample delay (in units of $\frac{1}{16}$ an ADC sample clock cycle). The sub-sample delay is implemented using a quadrature fractional delay FIR filter that not only does sub sample delay, but also the conversion to complex form. Converting to complex form involves mixing by a local oscillator at $\pm F_s/4$. The sign of the LO frequency is user selectable and determines whether the Delay Core's complex output will consist of the positive or negative frequencies of the real input. Selecting the negative frequencies will result in a frequency reversal and conjugation relative the positive frequencies. See the `sbsel` register for more details.

delay0 [R/W register]

The 17 least significant bits of this register specify the delay to be applied to the comparison input by the Delay Core.



The value written into these registers can be computed as

$$\begin{aligned} register\_value &= 16 \cdot adc\_sample\_delay + 64 && (\mathrm{mod}\ 2^{17}) \\ &= \mathrm{round}(16.384 \cdot ns\_delay) + 64 && (\mathrm{mod}\ 2^{17}) \end{aligned}$$

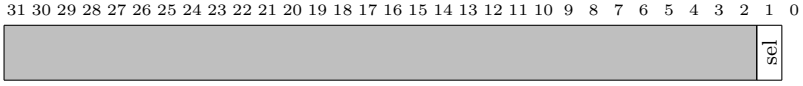
For example, to obtain a delay of 123.456 ns, the value to use would be computed as

$$\begin{aligned} register\_value &= \mathrm{round}(16.384 \cdot 123.456) + 64 && (\mathrm{mod}\ 2^{17}) \\ &= 2087 \end{aligned}$$

Because of the 64 sample offset and modulo 2^{17} behavior, minimum delay is obtained with a value of 64, while maximum delay is obtained with a value of 63.

sbsel [R/W register]

The `sbsel` (sideband select) register specifies whether the complex output will consist of the positive or negative frequencies of the real input.



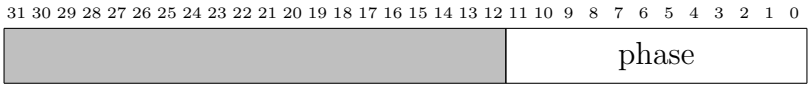
A value of 0 selects the positive frequencies; a value of 1 selects the negative frequencies.

4.6 Phase Offset and Fringe Tracking

The comparison input can have a constant phase offset and variable pahse offset (i.e. fringe tracking) applied.

phase0 [R/W register]

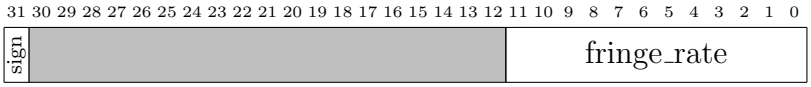
This register specifies the amount of user-specified "constant" (i.e. rarely changing) phase rotation to be applied to the comparison input's complex signal. The overall phase offset that is applied may optionally include an additional time-varying phase offset from the fringe tracking logic.



Because only the 12 least significant bits of these registers are used for one complete turn in phase, the resolution is $\frac{2\pi}{2^{12}}$ radians per step or $\frac{360}{2^{12}}$ degrees per step (i.e. approximately 88 millidegrees per step).

fringe_rate [R/W register]

This register specifies the fringe rate that is to be applied to the comparison input (in addition to the phase offset). The most significant bit is the sign of the fringe rate. The 12 least significant bits specify the fringe rate.

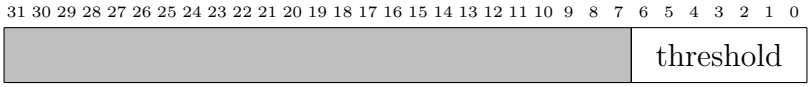


[TODO: What is the unit of fringe_rate?]

4.7 Single Baseline Correlator

quant/thresh0
quant/thresh1 [R/W registers]

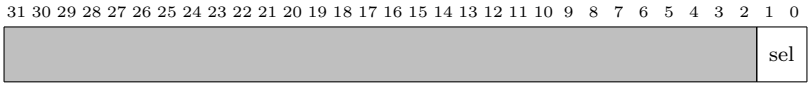
These registers set the thresholding level used for quantizing the inputs to the single baseline correlator. The first input's threshold is given by `quant/thresh0`; the second input's by `quant/thresh1`.



For more details. see the `quant/thresh`*N* registers in the iBob Phased Array section.

corr/syncsel [R/W register]

This register selects the sync signal used to drive integration timing.



The sync signal selection determines the units of the `integ_time` register. The four choices are:

0 Use the 1024 PPS signal to define the integration units.

1 Use the HB signal to define the integration units.

2 Use the SOWF signal to define the integration units.

3 Use the 1 PPS signal to define the integration units.

corr/integ_time [R/W register]

This register sets the integration time in units of the sync signal selected via the `syncsel` register. Changing the value of this register resets the correlator's internals as well as the `integ_cnt` register.

corr/integ_cnt [R register]

This register simply counts the number of integrations that have completed. When this register increments, a new integration is ready to be read. Changing the `integ_time` register resets this register.

corr/sample_cnt [R register]

The correlator always integrates in multiples of 32 samples (since there are 16 lags per baseline and 2 samples per FPGA clock cycle). If the selected integration unit (see `syncsel`) is not commensurate with 32 samples (e.g. the Heartbeat signal) then the number of samples per integration can vary from integration to integration. This register provides the number of 32-sample sub-integrations in the most recently completed integration. It can be used to properly average the data. Since the integrated values are pre-divided by 32 in the FPGA, the output values can be divided by this register value directly to get the average of the integrated products.

corr/subint_cnt [R register]

This register indicates how far along the current integration is. It is in units of the sync signal selected via the `syncsel` register. It counts from 0 to $integ\_time - 1$, then starts over.

corr/real
corr/imag [Block RAM]

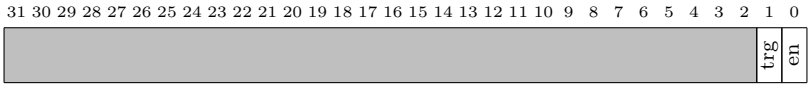
These shared BRAMs hold the integrated 16 lag spectrum for the single baseline correlator. The `corr/real` BRAM contains the integrated real component. The `corr/imag_imag` BRAM contains the integrated imaginary component. The 16 lags are stored from lag -8 to lag +7.

4.8 Snapshot Data

The DBE design includes a snapshot block that can be used to capture some input samples to help set the threshold registers and other diagnostic purposes. The input to the snap block is the same as the input to the single baseline correlator (see the `corrsel0` register).

snap/ctrl [R/W register]

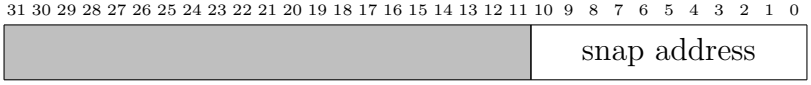
This register controls the snapshot block.



A 0 to 1 transition of bit 0 (en) enables a snapshot sequence. Once the snapshot sequence is enabled, the data capture will begin immediately if bit 1 (trg) is 1 or on the next selected sync pulse (see `syncsel`) if bit 1 (trg) is 0.

snap/addr [R register]

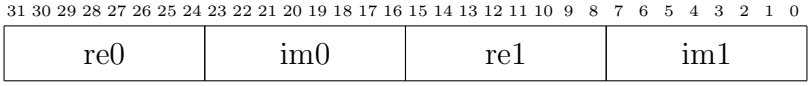
During data capture, this register provides the address within the shared BRAMs to which the data are being written.



When the data capture is complete, the address holds at the final value of 2047. The data capture completes in 8 µs, so this register is useful primarily when the data capture is setup to begin on the next sync pulse (otherwise it is unlikely that any value other than 2047 will be observed in this register).

snap/bram [Block RAM]

This shared BRAMs hold the snapshot data. Each BRAM is 2048 words deep. Within the BRAM, each 32 bit word stores two complex samples for the selected input. The most significant byte holds the real component of the first complex sample; the least significant byte holds the imaginary component of the second complex sample. Thus, the shared BRAM can hold 4096 complex samples for the selected input. The storage format for the first word (BRAM address 0) of `snap/bram` is shown in the following diagram.



4.9 VSI Formatting

outsel [R/W register]

This register has something to do with the formatting of the data for the data recorder. It is initialized to 3 at FPGA startup. **Do not change this register!**

pol0/gainctrl0
pol0/gainctrl1 [Block RAM]

These Block RAMs control the 2-bit quantization of the data being sent to the data recorder. The first 8 locations are initialized to 200 at FPGA startup. The data recorder have software that analyzes the histogram of the 2-bit data to compute new values to optimize the quantization.

pol0/gainreset [R/W register]

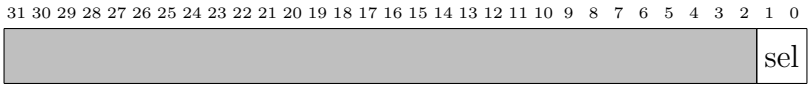
This register has something to do with the `pol0/gainctrl`*N* Block RAMs, but I'm not sure of the exact purpose.

shiftctrl_reg [R/W register]

This register controls the shifting within the PFB that channelizes the data before sending it to the data recorder. It is initialized to `0xffffffff` (i.e. shift every stage) at FGPA startup.

4.10 Front Panel SMA Output

sma1pps_sel [R/W register]
Selects which signal to output over the front panel `SMA1` (lower) connector.

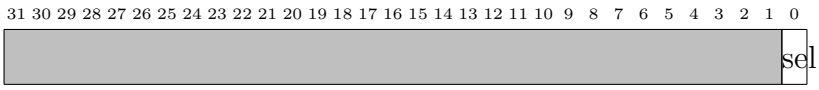


0 Outputs the sampled 1 PPS signal (from ADC0)

1 Outputs the internal 1 PPS signal.

2 Outputs the internal 1024 PPS signal.

smavsisel [R/W register]
Selects which signal to output over the front panel `SMA0` (upper) connector.



0 Outputs the 64 MHz clock being sent over VSI to the data recorder.

1 Outputs the 1 PPS signal being sent over VSI to the data recorder.

4.11 Front Panel LEDs

The DBE design does not use any of the front panel LEDs.

`LED8` indicates that the FPGA is configured (non-functional on some iBobs).