

# jTDC Documentation

generated from jTDC github wiki: 22.03.2016

# Contents

Preface	3
General features of the jTDC	3
1 Design Details	4
1.1 BRAM recorder	4
1.2 Trigger channel	4
1.3 Double page memory	5
1.4 High resolution sampling & encoding	5
1.5 Serializer	6
2 Read Data from TDC	8
2.1 Extracting times from raw HIT and CLOCK words	8
2.2 Error Flags	9
3 Example Implementations	10
3.1 VFB6 LVDS	11
3.1.1 Config Register A @ 0xbase0020	11
3.1.2 Config Register B @ 0xbase0028	12
3.1.3 Trigger Register @ 0xbase0024	12
3.1.4 Enable Registers	13
3.1.5 Scaler Readonly Registers	13
3.2 VFB6 DISC	14
3.2.1 Config Register A @ 0xbase0020	14
3.2.2 Config Register B @ 0xbase0028	15
3.2.3 Trigger Register @ 0xbase0024	15
3.2.4 Enable Registers	16
3.2.5 Scaler Readonly Registers	16
3.2.6 Thresholds and Hysteresis settings	16
3.2.7 Readback of analog voltages	17
4 Lab Measurements	18
Remarks	19

# Preface

The jTDC is an FPGA based high resolution TDC implemented on Xilinx Spartan 6 and compatible FPGAs. It is used at the BGO-OD experiment [\(↗\)](#) located at the ELSA facility [\(↗\)](#) in Bonn/Germany.

This repository not only contains the sources for the core modules of the jTDC, but also two complete example implementations designed for the VFB6 FPGA board [\(↗\)](#) manufactured by ELB [\(↗\)](#). I will include any other working implementation for other FPGA boards, if provided.

Furthermore, this repository contains *jDaqLite*, which is a reference tool to configure, monitor and read-out the jTDC. It also contains *jAnalyzer*, which reads data files generated by jDaqLite, extracts the actual tdc data and generates some example plots.

## General features of the jTDC

- Up to 100 TDC channels per board with 40ps average bin size (30ps RMS \*\*).
- All inputs can be inverted.
- Scalers (32bit @200MHz) for every input channel, which can also operate in duty cycle mode.
- Scalers support a lifetime-count-mode, if DAQ provides a busy signal.
- Maximum input rate without missing hits: 200MHz \*\* (valid for all inputs at the same time).
- Minimum length of input signals: 3ns \*\*
- Double pulse resolution: 5ns \*\*
- Max hits per event stored in DATA-FIFO: 15.360, limiting the trigger window to 775ns at full rate on all channels (at lower rate limited by BRAM size to 1250ns).
- Read-out during lifetime.
- 2 Trigger outputs available (logical OR of all input signals or subsets).

\*\* These features are limited by the hardware (Xilinx Spartan 6) and cannot be easily improved. All other limits are limits by choice (only 17% of RAM resources used, so plenty of space for more channels/larger FIFOs).

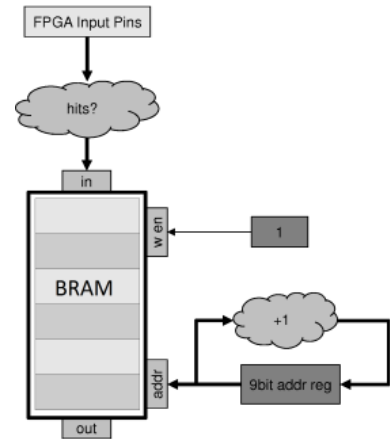
# 1 Design Details

This chapter provides details about the most important elements and concepts of the jTDC.

## 1.1 BRAM recorder

The central element of the jTDC is the BRAM recorder. Since most FPGAs feature lots of on-chip-memory (BRAM), the input status of each data channel (hit or no hit) is directly stored in the BRAM without doing any buffering, filtering or sorting. On Xilinx Spartan6 FPGAs, a single BRAM can be configured as 32x512, which provides storage for 512 32bit words (9 bit address space). Consider the following setup:

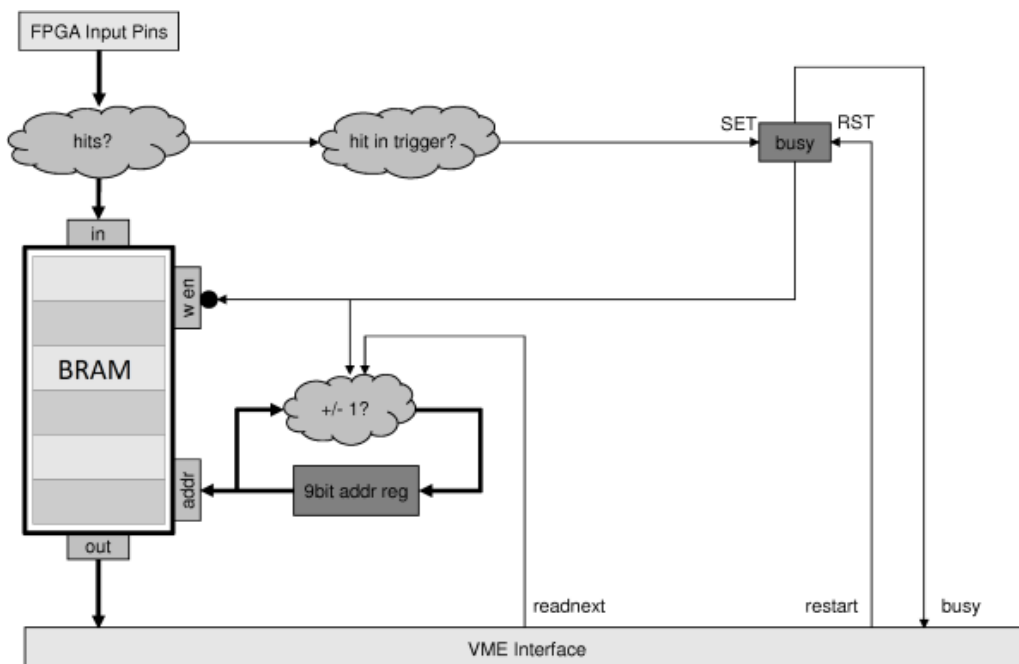
- BRAM is driven by a 200MHz clock and is always in write mode.
- 9bit-BRAM-address is incremented on each clock cycle (looping after 512 cycles).
- Input status of 32 data channels are used as BRAM inputs.



This setup allows to reconstruct the input status of all 32 channels for the last 2500ns (after the BRAM looped once). Since the input status of each channel is refreshed every 5ns, the BRAM recorder has a double pulse resolution of 5ns (if two hits are within 5ns, only one of them will be recorded).

## 1.2 Trigger channel

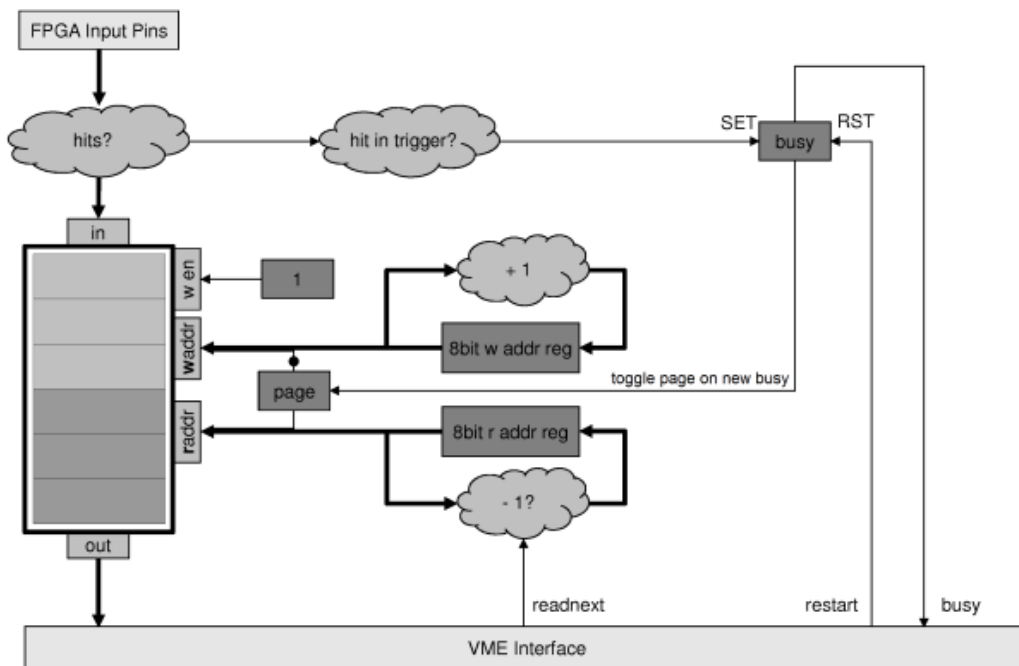
A dedicated trigger channel is used to stop the recording by setting a busy flag. If enabled, the 9bit-BRAM-address is no longer incremented on each clock cycle and the BRAM itself is no longer in write mode, it is waiting for read-out. During read-out, this simple tdc is not recording any data.



The read-out is controlled by sending "readnext" commands via VME, which decrements the 9bit-BRAM-address. The currently addressed memory cell can be read via VME. After all data has been read by rewinding the BRAM, a "restart" command is issued which clears the busy flag and switches the BRAM back to recording mode.

### 1.3 Double page memory

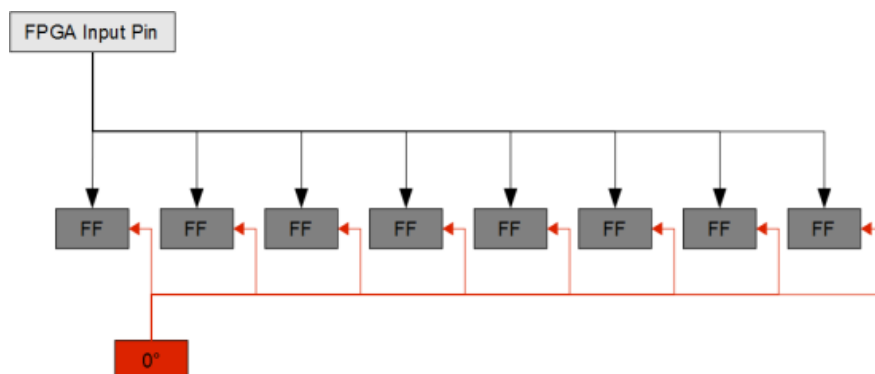
To record data even during read-out, the tdc needs a second memory page. On trigger input, the recording is not stopped but simply continued in that second memory page by flipping a page bit. The former write-to-page becomes the read-from-page by using the page bit as the 9th bit of the BRAM-read-address and the inverted page bit as the 9th bit of the BRAM-write-address.



To prevent further page flipping until the read-from-page has been read-out completely, the page bit is locked until a "restart" command is send. So trigger inputs during read-out are ignored.

### 1.4 High resolution sampling & encoding

To obtain a resolution higher than the 5ns from sampling with the system clock, the jTDC implements the tapped-delay-line method using carry chains.



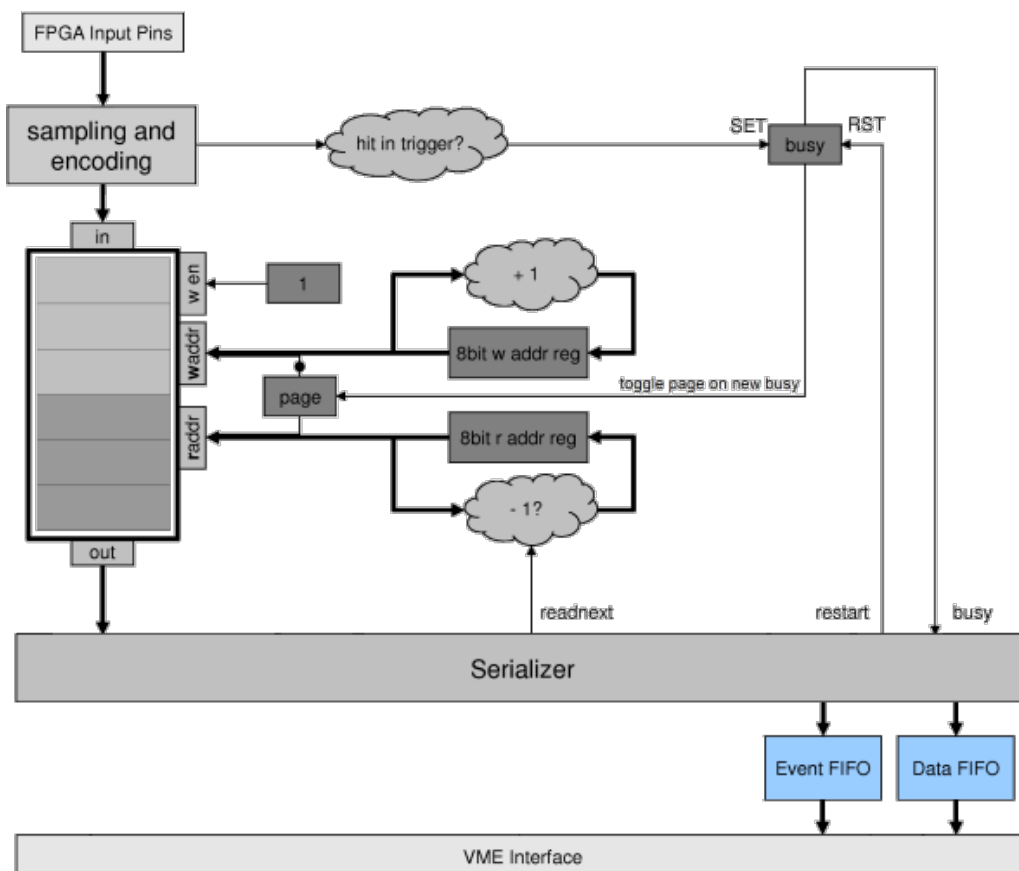
To reduce the length of the carry chain, it is driven by 400MHz instead of 200MHz and its 84 steps cover a bit more than 2.5ns. The 84 bits wide high resolution pattern received from the carry chain sampler is encoded into 7 bits, representing the position of the leading edge of the signal inside the chain, the desired high resolution time information. This implies, that only one leading edge within 2.5ns can be recorded (the first one).

The 7bit information from the encoder is transferred to the 200MHz clock domain and the state of the 400MHz clock is stored in an additional 8th bit. If there are two hits within one 200MHz clock cycle, only the first one survives, so the double pulse resolution is 5ns.

For each input channel, the BRAM has to store the single hit bit (hit or no hit) and the 8bit high resolution time information. So lots of BRAM is needed, but there usually is plenty. The jTDC uses only 17% of the available BRAM on a Xilinx XC6SLX150.

## 1.5 Serializer

The serializer automates the read-out process. As soon as it recognizes the busy state of the tdc (that means a page is waiting for read-out), it rewinds the page and extracts all hits and pushes the data into two FIFOs: An EVENT FIFO which contains an entry for each stored tdc event (containing the event number and the event size), and a DATA FIFO, which actually contains the data. The serializer generates some additional information for each event, stored as header and trailer words in the DATA FIFO alongside the hit data.

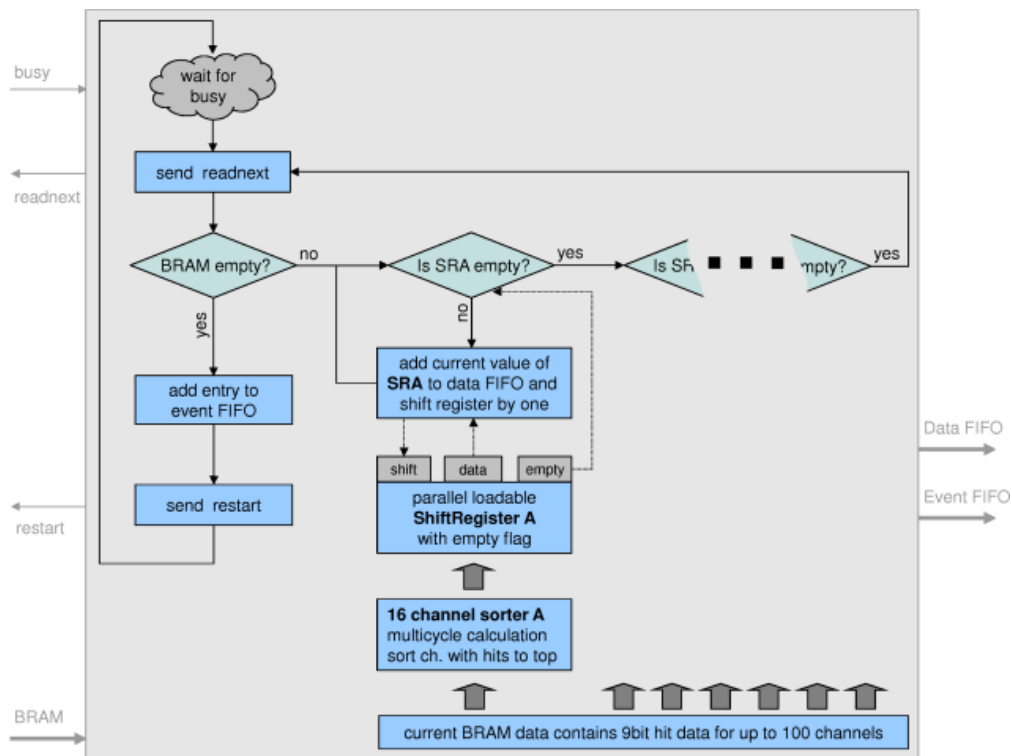


If the serializer is not able to push any more data in the FIFOs, because they are full (not being read out), it simply stops processing the BRAM, thus not issuing any readnext or restart commands. If the event is larger than the the DATA FIFO, the serializer has to truncate the event, because the EVENT

FIFO is only written after the entire event has been pushed into the DATA FIFO - which will never happen, if the event is too large. This will be indicated by the error flag *data fifo overflow* in the trailer word of that event.

After the page has been read-out, the serializer issues the restart command, so the jTDC can process new triggers again. If a trigger was received during read-out, the trailer word of that event will contain the error flag *trigger during read-out*.

The serializer got its name, because it actually has to serialize the parallel BRAM data. The data chunk it gets from the BRAM recorder after issuing a readnext command contains the input condition of ALL input channels at a certain time. Most channels are probably empty and the serializer has to find the ones which contain a hit and has to push them into the data FIFO (one FIFO entry per hit).



In order to get this done fast, the channels are pre-processed in groups of 16 and each group sorts the channels which contain hits to the top. The serializer queries each group and jumps to the next if done. By doing so, the serializer needs only 8 clk cycles to process a BRAM chunk containing just a single hit.

## 2 Read Data from TDC

If there is an event ready for read-out, a read request to the EVENT-FIFO (0xbase8888) will return a 32bit word which contains the event number (highest 16 bits) and the event size (lowest 13 bits). Otherwise, it will return 0.

To obtain the actual event data, read as many 32bit words from the DATA-FIFO (0xbase4444) as indicated by the event size. Each of these 32bit words contains two 16bit words of TDC data. The possible types of 16bit words are as follows:

16bit word	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEADER1	1	1	0	event size												
HEADER2	1	1	1	geoid						bits per channel						
HIT	0	channel number								8bit high resolution sampling						
CLOCK	1	0	0	13bit clock counter since trigger												
FILLER	1	0	1	1												
TRAILER	1	0	1	0	error flags					lowest 8 bit of event number						

An event always starts with a HEADER1 word followed by a HEADER2 word and ends with a TRAILER word. If the TRAILER word is not 32bit aligned, a FILLER word is inserted before the TRAILER.

The header words are followed by the hits of the event. The HIT word contains the channel number and the high resolution sampling information - the offset between the 200MHz sampling clock and the signal - but not the number of clock cycles since trigger. This information is send with the CLOCK word after the HIT word. If hits of different channels are within the same clock cycle, the clock information is send only once after the last hit of this group.

### 2.1 Extracting times from raw HIT and CLOCK words

For each hit the jTDC provides a 13bit clock counter (coarse sampling at 200MHz) and a 8bit high resolution sampling information. These two information can be combined to get the full 21bit tdc time information:

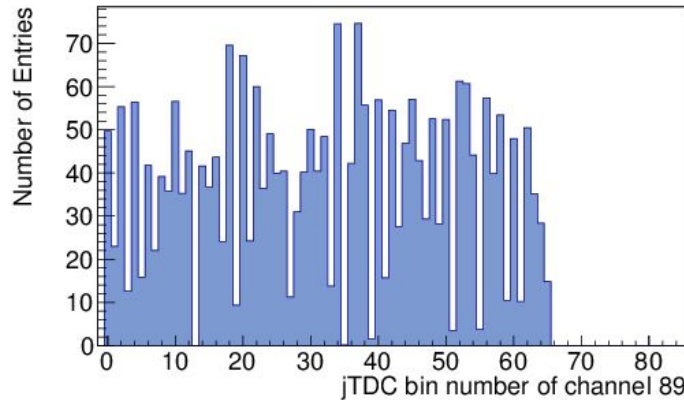
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
13bit 200MHz clock sampling													8bit high resolution							

As described in the previous chapter, the coarse sampling was actually done at 400MHz, so bit #7 of the high resolution sampling information is simply the clock state of the 400MHz sampling clock and the true high resolution sampling information is only 7bit wide:

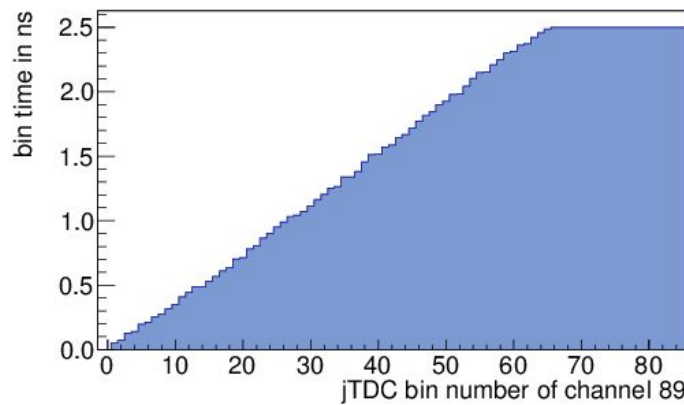
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
14bit 400MHz clock sampling														7bit high resolution						



Filling these reduced 7bit high resolution bin numbers of a larger set of tdc events into a histogram, produces a differential nonlinariaty plot:



If we assume that the input data is white noise and that we have enough statistics, the above bin distribution can be used to calculate the individual width of each bin by normalizing the integral of the histogram to  $2.5ns$ :



From this data one can obtain the time for each tdc bin. This offline calibration needs to be done for each tdc channel. For a detailed howto look at the source of *jAnalyzer*, which is doing just that.

## 2.2 Error Flags

During an event, the following errors could be encoded into the error flags of the TRAILER word:

bit	error	description
0	trigger during read-out	There was at least one trigger while the TDC was still busy pushing the data from the BRAM into the DATA-FIFO. Such triggers are ignored.
1	data fifo overflow	The event was too large, some of the older hits of this event could not be pushed into the DATA-FIFO. These hits are lost.
2	too old tdc data ignored	While pushing the data from the BRAM into the DATA-FIFO, the TDC encountered hits older than specified by the maximum trigger window limit. All hits older than this limit were ignored. (The first "to old" hit still makes it into the data.)

# 3 Example Implementations

All example implementations can be configured, monitored and read-out by *jDaqLite*.

At current, there are only two example implementations which are designed for the VFB6 FPGA board ([↗](#)) manufactured by ELB ([↗](#)) with two different input card configurations. I will include any other working implementation for other FPGA boards, if provided.

The VFB6 board has the following I/O:

- 3 mezzanine input/output card connectors
- 4 NIM connectors, each configurable as input or output

The first VFB6 implementation (a.k.a. [VFB6-LVDS Version](#)) is designed for the LVDS INPUT cards (32 LVDS inputs per mezzanine, thus 96 channel in total), the second VFB6 implementation (a.k.a. [VFB6-DISC Version](#)) is designed for the DISCRIMINATOR INPUT cards (16 analog inputs per mezzanine, thus 48 channels in total). The discriminators operate independent from the FPGA, it only receives the discriminated time-over-treshold (ToT) signal. The DISC version is recording both edges of the discriminated signal, to be able to extract the ToT information from the data.

Besides the 96 LVDS / 48 analog input channels, two of the NIM inputs are recorded by the TDC, one of them can be used as trigger input.

## 3.1 VFB6 LVDS

This example implementation of the jTDC is designed for the VFB6 board from ELB using 3 LVDS input mezzanine cards. It's inputs and outputs are mapped as follows:

- MEZA: LVDS input channel 01-32
- MEZB: LVDS input channel 33-64
- MEZC: LVDS input channel 65-96
- NIM[0]: Channel 0 input (trigger and busy) input (frontpanel, left NIM socket)
- NIM[1]: Trigger A output (frontpanel, right NIM socket)
- NIM[2]: Channel 97 input (backpanel, left NIM socket)
- NIM[3]: Trigger B output (backpanel, right NIM socket)

All registers are 32bit registers and you must read/write all 32bits, even if you want to change/set/get only a specific bit. Some registers are so called toggle registers, they toggle back to 0x00000000 after one clock cycle and are used to trigger events (for example resets). The read value from such a toggle register has mostly a completely different source (for example status information) and has nothing to do with the toggle state of the register.

### 3.1.1 Config Register A @ 0xbase0020

bits	config value
4-0	Set GeoID of the module. This ID is written into the header data of each event.
5	Set to 1 to enable dutycycle count mode (check input state each clk cycle and count if state is high)
6	Set to 1 to invert all inputs
7	Set to 1 to use LVDS_A_IN[0] instead of NIM[0] as trigger input
15-8	Set maximum trigger window size to (N*5)ns (max 1250ns)

This is a configuration register, you can readback the current values at any given time.

### 3.1.2 Config Register B @ 0xbase0028

bits	config value
8-0	busyshift** (0-2500ns in steps of 5ns)
9	enable lifetime gate for counters (stop counting on busy)**
15-11	busyextend (0-155ns in steps of 5ns, must not be larger than length of busy)**
19-16	lifetime (length) setting for trigger output: $(5*N + 10)ns$
23-20	deadtime setting for trigger output: $(5*N + 10)ns$
26-24	3bit mask for trigger output NIM[1] to select OR of LVDS_C, LVDS_B and/or LVDS_A (0x7 will put the logical OR of all LVDS inputs on NIM[1], 0x2 just those of LVDS_B)
29-27	3bit mask for trigger output NIM[3] to select OR of LVDS_C, LVDS_B and/or LVDS_A (0x7 will put the logical OR of all LVDS inputs on NIM[3], 0x2 just those of LVDS_B)
30	disable external latch (scalars will not be latched by the trigger input)

\*\* The trigger-input is used twofold: Its leading edge is used as trigger and its length is used as DAQ deadtime/busy indicator. If the DAQ keeps the trigger signal active until it has finished processing it (read-out of all TDCs), the jTDC can use that information as a life-time-gate for its input scalars. To cope with extra delay due to cables and other elements, the busy signal can be shifted and extended.

This is a configuration register, you can readback the current values at any given time.

### 3.1.3 Trigger Register @ 0xbase0024

bit	toggle actions
0	TDC reset
1	counter reset
2	counter latch
3	output reset (must be called after changing the high- or dead time settings, and after initial fw load)

This is a toggle register, the readback value of this register contains the following information:

bit	value
7-0	firmware version
15-8	module id
23-16	number of encoded bits per channel
31-24	number of tdc channels

### 3.1.4 Enable Registers

address	32bit bitmask to enable/disable channels
0xbase2000	LVDS input channels 32 - 01
0xbase2004	LVDS input channels 64 - 33
0xbase2008	LVDS input channels 96 - 65

This is a configuration register, you can readback the current values at any given time.

### 3.1.5 Scaler Readonly Registers

**CAUTION!** The scalers cannot be addressed directly! First read the clock counter and then read register 0xbase4000 98 times to get the 98 scaler values. For convenience, the addressbus is masked for register 0xbase4000, so one could also consecutively read address 0xbase4000 to 0xbase409B.

address	value in read-only register
0xbase0044	clock counter (a read request sets the current read-out scaler to channel 0)
0xbase4000	scaler value of the current read-out scaler (auto incrementation of the current read-out scaler)

## 3.2 VFB6 DISC

This example implementation of the jTDC is designed for the VFB6 board from ELB using 3 DISCRIMINATER mezzanine cards. It's inputs and outputs are mapped as follows:

- MEZA: analog input channel 01-16
- MEZB: analog input channel 17-32
- MEZC: analog input channel 33-48
- NIM[0]: Channel 0 input (trigger and busy) input (frontpanel, left NIM socket)
- NIM[1]: Trigger A output (frontpanel, right NIM socket)
- NIM[2]: Channel 49 input (backpanel, left NIM socket)
- NIM[3]: Trigger B output (backpanel, right NIM socket)

The FPGA only sees the discriminated time-over-threshold signals and samples *BOTH* edges. The trailing edges of input channels 01-48 are stored as tdc channels 51-98.

All registers are 32bit registers and you must read/write all 32bits, even if you want to change/set/get only a specific bit. Some registers are so called toggle registers, they toggle back to 0x00000000 after one clock cycle and are used to trigger events (for example resets). The read value from such a toggle register has mostly a completely different source (for example status information) and has nothing to do with the toggle state of the register.

### 3.2.1 Config Register A @ 0xbase0020

bits	config value
4-0	Set GeoID of the module. This ID is written into the header data of each event.
5	Set to 1 to enable dutycycle count mode (check input state each clk cycle and count if state is high)
6	Set to 1 to invert all inputs
7	Set to 1 to use DISC_A_IN[0] instead of NIM[0] as trigger input
15-8	Set maximum trigger window size to $(N*5)ns$ (max 1250ns)

This is a configuration register, you can readback the current values at any given time.

### 3.2.2 Config Register B @ 0xbase0028

bits	config value
8-0	busyshift** (0-2500ns in steps of 5ns)
9	enable lifetime gate for counters (stop counting on busy)**
15-11	busyextend (0-155ns in steps of 5ns, must not be larger than length of busy)**
19-16	high time (length) setting for trigger output: $(5*N + 10)ns$
23-20	dead time setting for trigger output: $(5*N + 10)ns$
26-24	3bit mask for trigger output NIM[1] to select OR of DISC_C, DISC_B and/or DISC_A (0x7 will put the logical OR of all DISC inputs on NIM[1], 0x2 just those of DISC_B)
29-27	3bit mask for trigger output NIM[3] to select OR of DISC_C, DISC_B and/or DISC_A (0x7 will put the logical OR of all DISC inputs on NIM[3], 0x2 just those of DISC_B)
30	disable external latch (scalars will not be latched by the trigger input)

\*\* The trigger-input is used twofold: Its leading edge is used as trigger and its length is used as DAQ deadtime/busy indicator. If the DAQ keeps the trigger signal active until it has finished processing it (read-out of all TDCs), the jTDC can use that information as a life-time-gate for its input scalars. To cope with extra delay due to cables and other elements, the busy signal can be shifted and extended.

This is a configuration register, you can readback the current values at any given time.

### 3.2.3 Trigger Register @ 0xbase0024

bit	toggle actions
0	TDC reset
1	counter reset
2	counter latch
3	output reset (must be called after changing the high- or dead time settings, and after initial fw load)

This is a toggle register, the readback value of this register contains the following information:

bit	value
7-0	firmware version
15-8	module id
23-16	number of encoded bits per channel
31-24	number of tdc channels

### 3.2.4 Enable Registers

address	16bit bitmask to enable/disable channels (highest 16bit of 32bit registers are not used)
0xbase2000	DISC input channels 16 - 01
0xbase2004	DISC input channels 32 - 17
0xbase2008	DISC input channels 48 - 33

### 3.2.5 Scaler Readonly Registers

**CAUTION!** The scalers cannot be addressed directly! First read the clock counter and then read register 0xbase4000 98 times to get the 98 scaler values. For convenience, the addressbus is masked for register 0xbase4000, so one could also consecutively read address 0xbase4000 to 0xbase409B.

address	value in read-only register
0xbase0044	clock counter (a read request sets the current read-out scaler to channel 0)
0xbase4000	scaler value of the current read-out scaler (auto incrementation of the current read-out scaler)

### 3.2.6 Thresholds and Hysteresis settings

The following settings are written into registers of DAQs on the discriminator input card to set different voltages. For the threshold, an analog comparator is used to detect when the input signal is above threshold, thus discriminating the analog input signals. The hysteresis setting is added to the threshold for the start-of-signal-detection and subtracted from the threshold for the end-of-signal-detection.

Furthermore, the DAQs can be fine-tuned by additional offset settings. For more information please contact ELB.

The *index* in the following table is used as follows: 0=all ch, 1-16=ch, 17=offset daq 1a, 18=offset daq 1b, 19=offset daq 2a, 20=offset daq 2b.

address	pattern of dataword to set
0xbaseA004	write any value in this reg to initialize MEZA
0xbaseA044	write any value in this reg to initialize MEZB
0xbaseA084	write any value in this reg to initialize MEZC
0xbaseA010	set thresholds for <i>index</i> of MEZA via {5bit index, 16bit value}
0xbaseA050	set thresholds for <i>index</i> of MEZB via {5bit index, 16bit value}
0xbaseA090	set thresholds for <i>index</i> of MEZC via {5bit index, 16bit value}
0xbaseA020	set hysteresis for <i>index</i> of MEZA via {5bit index, 16bit value}
0xbaseA060	set hysteresis for <i>index</i> of MEZB via {5bit index, 16bit value}
0xbaseA0A0	set hysteresis for <i>index</i> of MEZC via {5bit index, 16bit value}



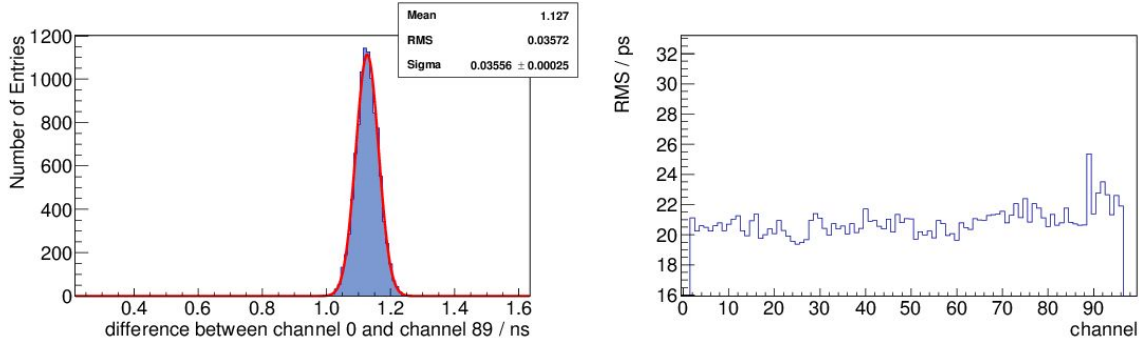
### 3.2.7 Readback of analog voltages

The discriminator input card also contains an ADC, which can be used to readback the actual voltages of the current thresholds, hystereses and others.

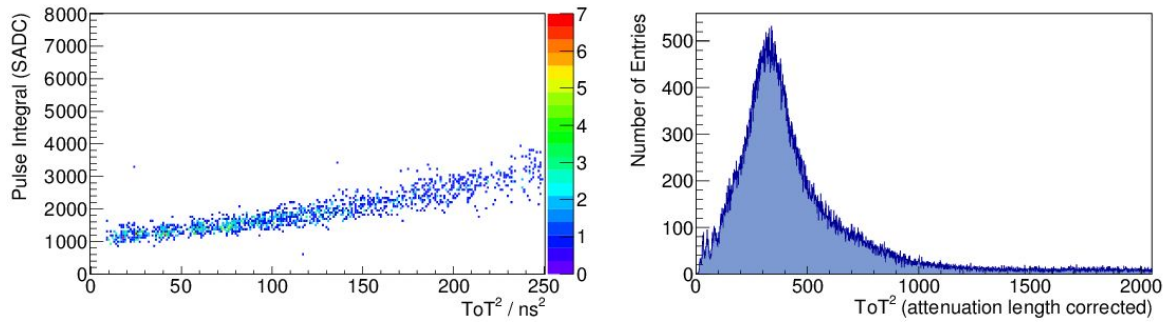
address	analog readback value (one or two 16'bit words)
$0xbaseA100 + 4 \cdot i$	{HYSTERESE & THRESHOLD} of channel $i$
0xbaseA200	{DAC1_OFFSETA & DAC1_OFFSETB} of MEZA
0xbaseA204	{DAC2_OFFSETA & DAC2_OFFSETB} of MEZA
0xbaseA208	{DAC1_REFA & DAC1_REFB} of MEZA
0xbaseA20C	{DAC2_REFA & DAC2_REFB} of MEZA
0xbaseA210	{DAC_GND} of MEZA
0xbaseA240	{DAC1_OFFSETA & DAC1_OFFSETB} of MEZB
0xbaseA244	{DAC2_OFFSETA & DAC2_OFFSETB} of MEZB
0xbaseA248	{DAC1_REFA & DAC1_REFB} of MEZB
0xbaseA24C	{DAC2_REFA & DAC2_REFB} of MEZB
0xbaseA250	{DAC_GND} of MEZB
0xbaseA280	{DAC1_OFFSETA & DAC1_OFFSETB} of MEZC
0xbaseA284	{DAC2_OFFSETA & DAC2_OFFSETB} of MEZC
0xbaseA288	{DAC1_REFA & DAC1_REFB} of MEZC
0xbaseA28C	{DAC2_REFA & DAC2_REFB} of MEZC
0xbaseA290	{DAC_GND} of MEZC

## 4 Lab Measurements

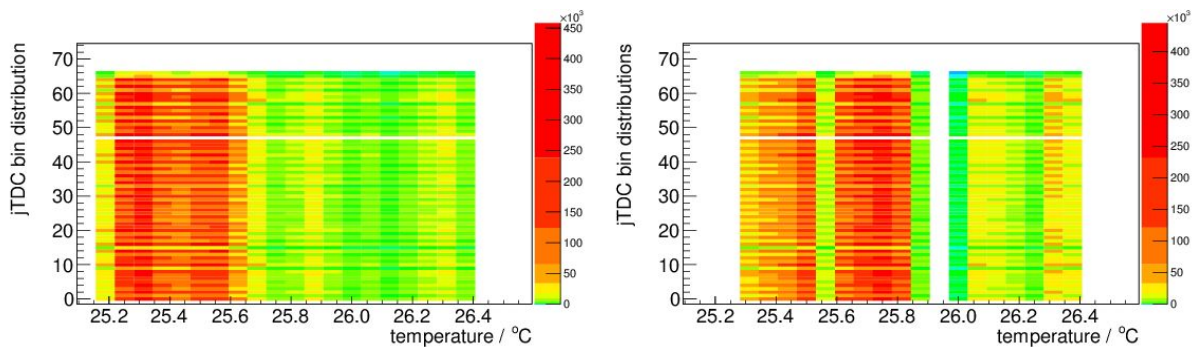
The jTDC resolution is measured by connecting two channels to the same source and calculating the difference. The worst channel during this test was channel 89, its gaussian distribution had an RMS of  $36\text{ps}$  (left plot). For the true single channel resolution, the RMS result of the difference calculation must be divided by  $\sqrt{2}$ . For all channels a resolution of at least  $30\text{ps}$  has been achieved (right plot).



If the analog input card is used, time over threshold values from jTDC can be correlated to the pulse integral of an SADC (left plot). A standard ADC spectrum (without pedestal) can be extracted from the jTDC ToT data. Energy calibration is still needed (right plot).



The jTDC is based on the white noise assumption, so it is important to verify, that the bin distribution is not temperature or rate dependent: The propagation time of the input signal through the TDL of the input sampler must stay constant during data taking. The following plots are generated using 1 week of data of a single channel (left plot low rate, right plot high rate runs). The color code can be ignored, it only shows how many runs have been recorded at a given temperature. The visible structures are much more interesting: They are hardware/FPGA dependent and would move to higher/lower bins, if the propagation time through the TDL would increase/decrease due to temperature and/or rate dependencies. Even though temperature and rate changed during that week, the structures in the bin distributions did not.



# Remarks

- enable serializer look-ahead logic
- measure impact of look-ahead logic
- send coarse counter as slice header, not slice trailer

It also has to be checked, if the current implementation of "trigger-on-busy" is as desired. Every incoming trigger will reset the clockcounter even if it is going to be ignored. Thus, when pushing the page into BRAM the next time it is possible, only the data since the last ignored trigger - not since the last valid trigger - is pushed into the FIFO.