jTDC: Documentation

generated from jTDC github wiki: 18.03.2016

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. The first prototype was developed on a Xilinx Virtex 5 FPGA for the COMPASS experiment located at CERN. As a proof of concept, this repository not only contains the sources for the core modules of the jTDC, but also two complete example implementations for the VFB6 FPGA board (↗) manufactured by ELB (↗). I will include any other working implementation for other FPGA boards, if provided.

General features of the jTDC

- Up to 100 TDC channels per board with 40*ps* average bin size (30*ps* RMS **).
- All inputs can be inverted.
- Scalers (32bit @200*MHz*) 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: 200*MHz* ** (valid for all inputs at the same time).
- Minimum length of input signals: 3*ns* **
- Double pulse resolution: 5*ns* **
- Max hits per event stored in DATA-FIFO: 15.360, limiting the trigger window to 775*ns* at full rate on all channels (at lower rate limited by BRAM size to 1250*ns*).
- Readout during lifetime.
- 2 Trigger outputs available ** (logical OR of all input signals or subsets).

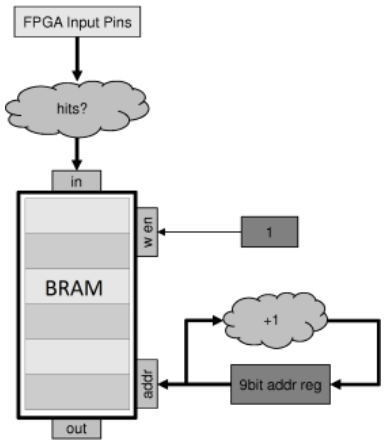
** These features are limited by the hardware (Xilinx Spartan 6) and cannot be easily improved. All other limits are limits by choice (only 17% of RAM resources used, so plenty of space for more channels/larger FIFOs).

1 Design details

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

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:

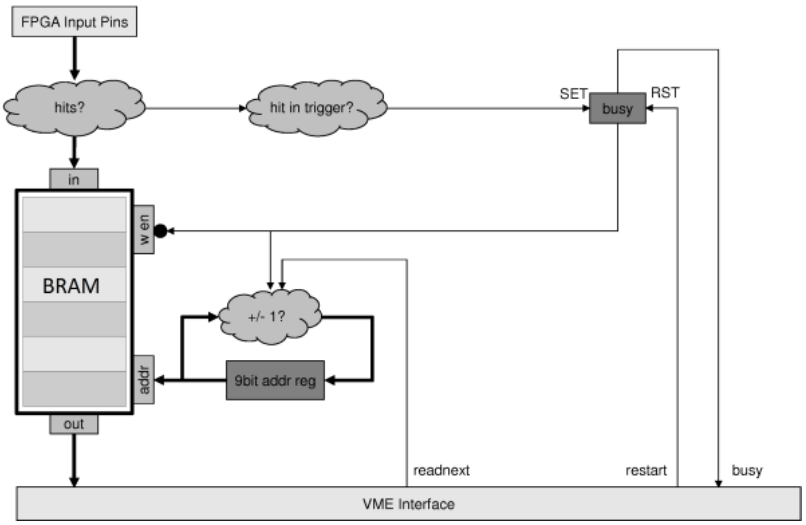


- BRAM is driven by a 200*MHz* clock and is always in write mode.
- 9bit-BRAM-address is incremented on each clock cycle (looping after 512 cycles).
- Input status of 32 data channels are used as BRAM inputs.

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

Trigger channel

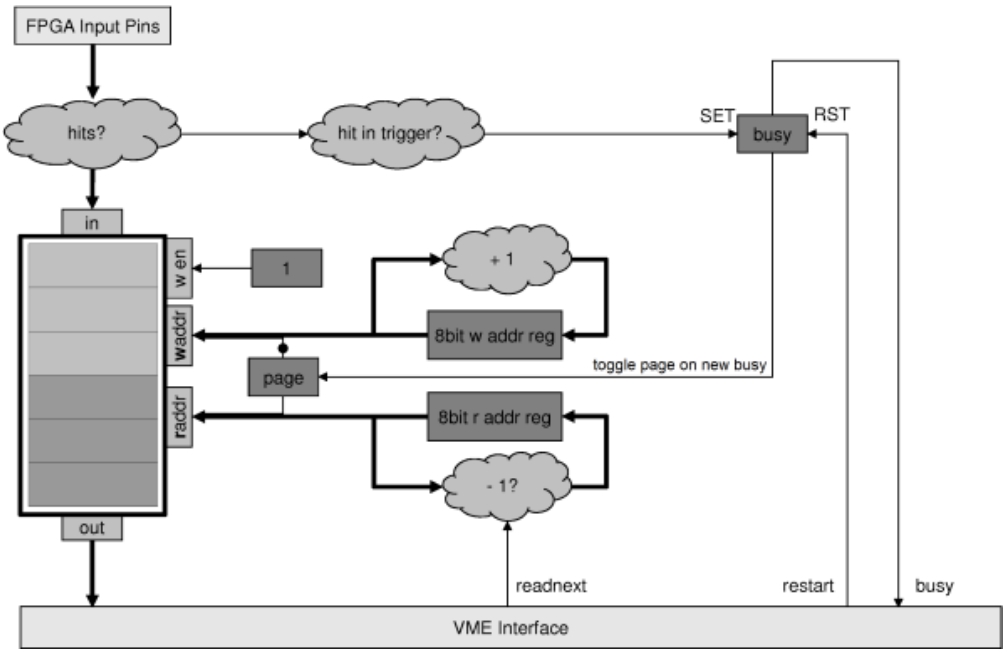
A dedicated trigger channel is used to stop the recording.



A busy flag indicates, whether the tdc is recording or waiting for readout. The readout is controlled by sending "readnext" commands via VME, to decrement 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 and the tdc switches back to recording mode. During readout, this kind of tdc is not recording any data and is ignoring trigger inputs.

Double page memory

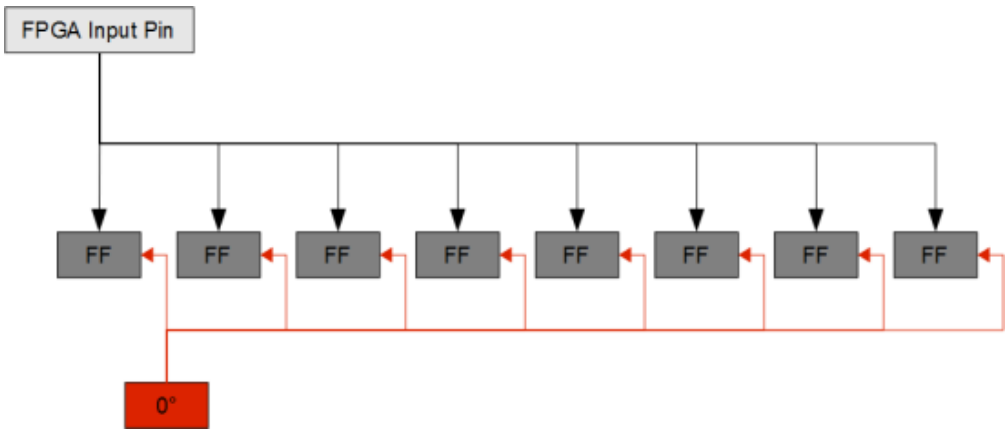
The tdc should continue to record data during readout. This is achieved by using two independent memory areas. On a trigger input, the recording is not stopped but simply continued in a second memory page by flipping the page bit: The page bit is used as the 9th bit of the BRAM-read-address and the inverted page bit is used as the 9th bit of the BRAM-write-address.



By flipping the page bit, the former write-to-page becomes the read-from-page. To prevent further page flipping until the read-from-page has been readout completely, the page bit is locked until a "restart" command is send. So trigger inputs during readout will be ignored.

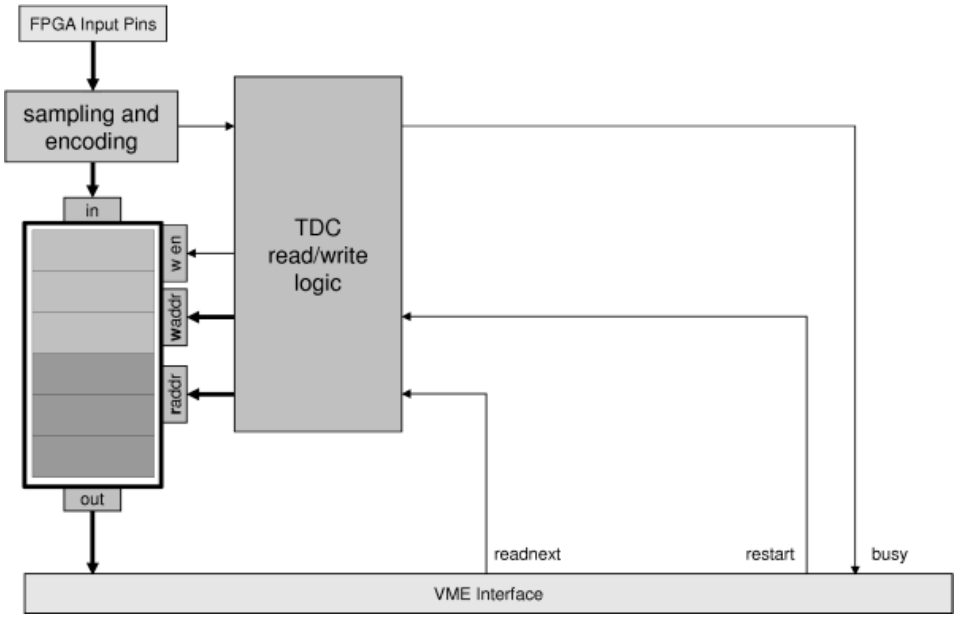
High resolution sampling & encoding

To obtain a resolution higher than the 5*ns* 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 400*MHz* instead of 200*MHz* and its 84 steps cover a bit more than 2.5*ns*. 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

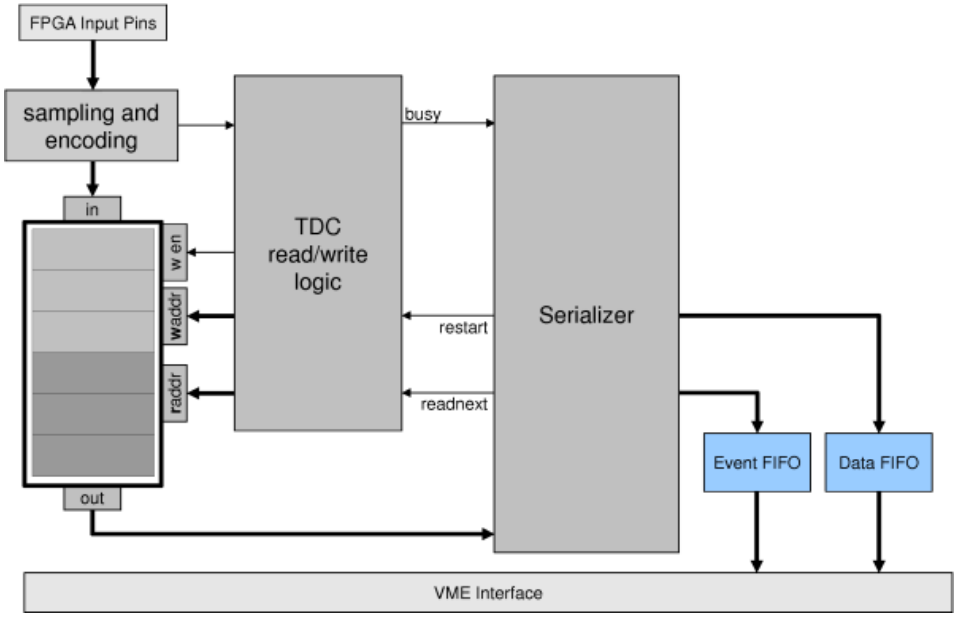
desired high resolution time information. This implies, that only one leading edge within 2.5*ns* can be recorded (the first one). The 7bit information from the encoder is transferred to the 200*MHz* clock domain and the relation to the 400*MHz* clock is stored in an additional 8th bit. If there are two hits within one 200*MHz* clock cycle, only the first one survives, so the double pulse resolution is 5*ns*.



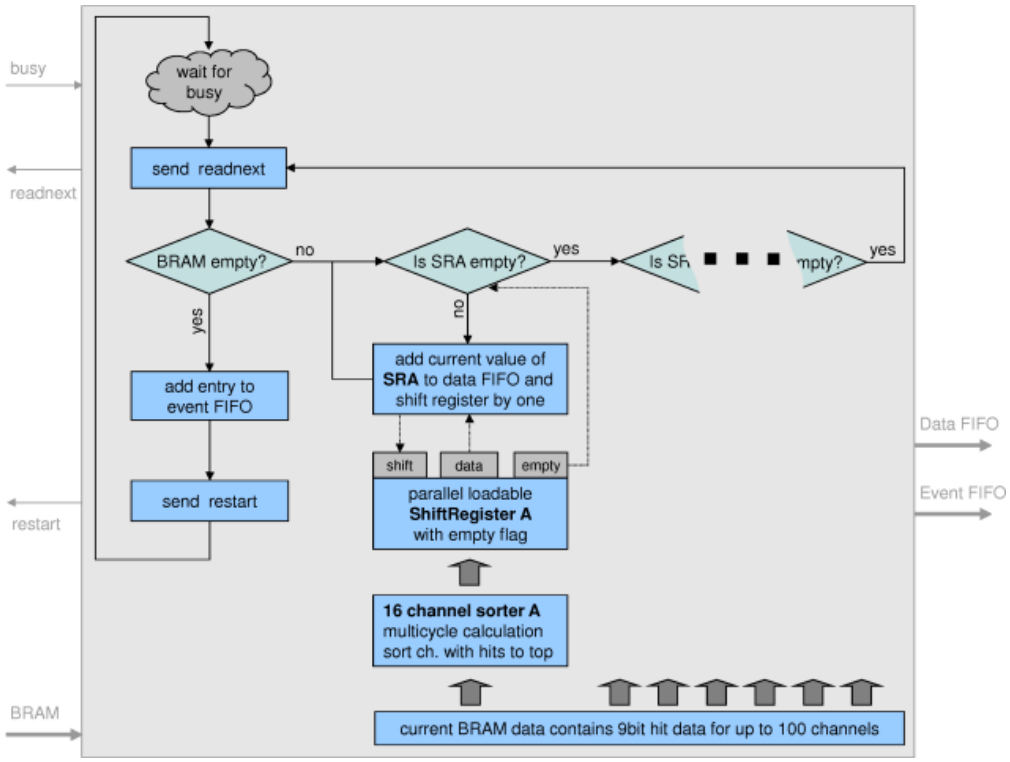
So besides the single hit bit (hit or no hit) for each input channel, the jTDC also stores the high resolution time information (8 bit) for each channel into the BRAM. So the jTDC needs lots of BRAM, but there usually is plenty (17% used BRAM on Xilinx XC6SLX150).

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 a FIFO. At the same time, the serializer generates some additional information for each event, stored in header and trailer words written to the FIFO alongside the hit data.



After the page has been read-out, the serializer issues the restart command, so the jTDC can process new triggers again. The serializier fills two different FIFOs: An event FIFO which contains an entry for each stored tdc event (containing an event number and the event size), and a data FIFO, which actually contains the data. If the serializer is not able to push any more data in the FIFOs, because they are full (not read out), it simply stops processing (thus not issuing any readnext or restart commands).



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 Readout

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

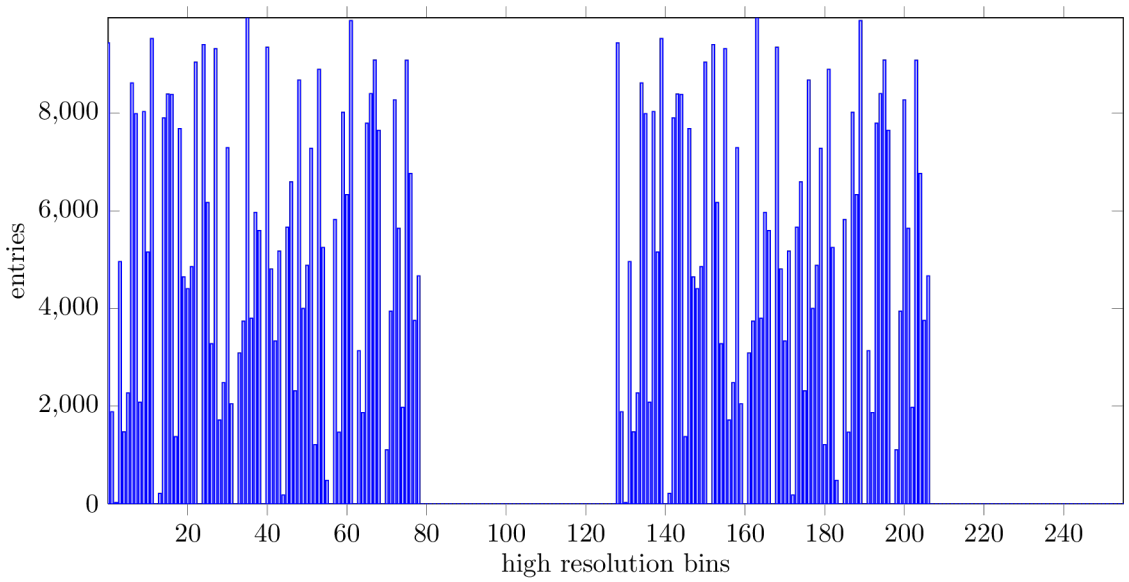
After the two header words the hits of the event are send. The HIT word contains the channel number and the high resolution sampling information - the offset between the 200*MHz* sampling clock and the signal - but not the number of clock cycles since trigger. This information is send with the CLOCK 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.

Extracting the high resolution timing information

For each hit the jTDC provides a 13bit clock counter (coarse sampling information with 5*ns* resolution) and a 8bit high resolution 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 200*MHz* clock sampling			8bit high resolution	

Filling just the high resolution bin numbers of a larger set of tdc events into a histogram, produces a differential nonliniarity plot:



Why are there two groups of filled bins? The coarse sampling is actually done at 400*MHz* and the result is transferred to the 200*MHz* system clock. The highest bit of the high resolution sampling information is simply the clock information (even/odd) of the 400*MHz* sampling clock with respect to the 200*MHz* system clock. Since both bin groups (0-78 and 128-206) are produced by the exact same tapped delay line inside the FPGA, they should have the same distribution.

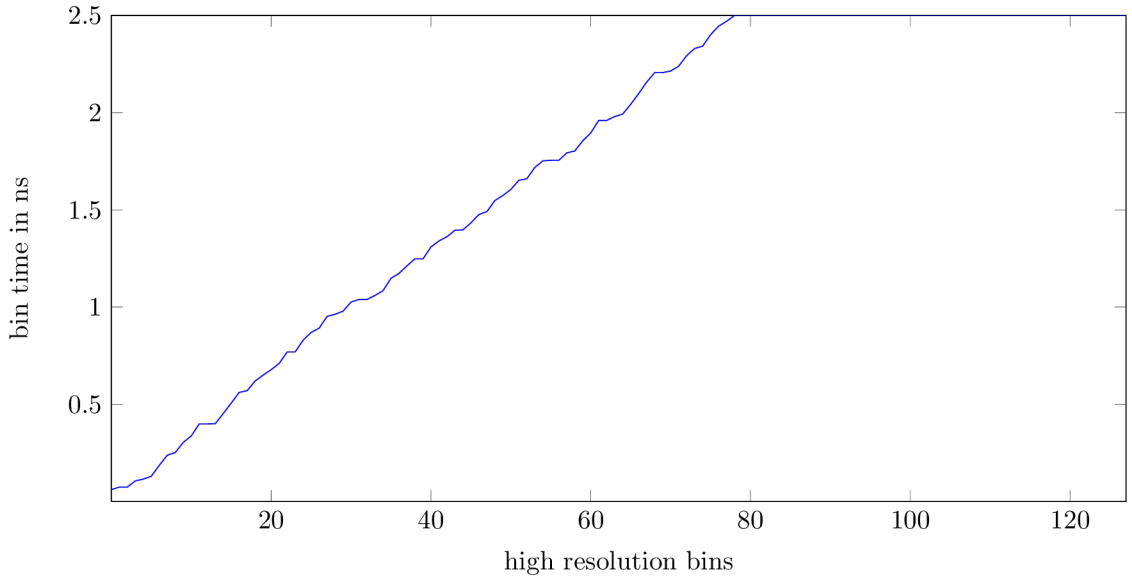
This allows to re-interpret the combined hit information with a course clock sampling of 400*MHz*:

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

This "merges" the two bin groups seen before and doubles the available statistics.

If we assume that the input data is white noise and that we have enough statistics, the merged high resolution bin distribution can be used to calculate the individual width of each bin: Normalize the integral of the merged histogram to 2.5*ns*. This is the main working principle of the jTDC.

If the given example data is merged and normalized to 2.5*ns*, the following integrated nonliniarity plot can be created:



From this data one can obtain the time for each tdc bin. This offline calibration needs to be done for each tdc channel.

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 busy	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

At current, there are only two example implementations for the VFB6 FPGA board (↗) manufactured by ELB (↗) with two different input cards configuration. 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 using 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)

VME config and readout register

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.

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 1250*ns*)

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

Config Register B @ 0xbase0028

bits	config value
8-0	busyshift (0-2500*ns* in steps of 5*ns*)
9	enable lifetime gate for counters (stop counting on busy)
15-11	busyextend (0-155*ns* in steps of 5*ns*, 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 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 (scalers will not be latched by the trigger input)

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

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

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.

Scaler Readout 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 readout scaler to channel 0)
0xbase4000	scaler value of the current readout scaler (auto incrementation of the current readout 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.

VME config and readout register

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.

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 1250*ns*)

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

Config Register B @ 0xbase0028

bits	config value
8-0	busyshift (0-2500*ns* in steps of 5*ns*)
9	enable lifetime gate for counters (stop counting on busy)
15-11	busyextend (0-155*ns* in steps of 5*ns*, 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 (scalers will not be latched by the trigger input)

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

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

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

Scaler Readout 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 readout scaler to channel 0)
0xbase4000	scaler value of the current readout scaler (auto incrementation of the current readout scaler)

Thresholds and Hysterese 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 hysterese settings 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 *index* of MEZA via {5bit index,16bit value}
0xbaseA050	set thresholds for *index* of MEZB via {5bit index,16bit value}
0xbaseA090	set thresholds for *index* of MEZC via {5bit index,16bit value}
0xbaseA020	set hysterese for *index* of MEZA via {5bit index,16bit value}
0xbaseA060	set hysterese for *index* of MEZB via {5bit index,16bit value}
0xbaseA0A0	set hysterese for *index* of MEZC via {5bit index,16bit value}

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*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

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 incomming trigger wil reset the clockcounter even if it is going to be ignored. Thus, when pushing the page int 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.