AMT-3 (ATLAS Muon TDC version 3) & AMT-2

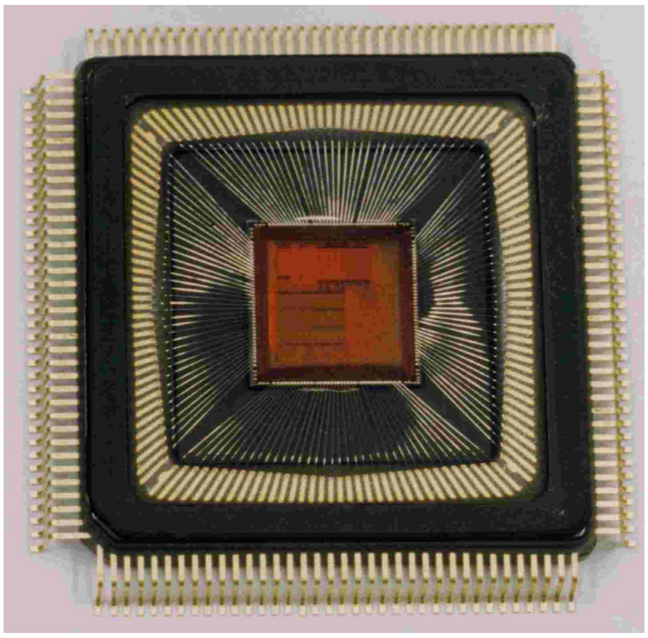
User's Manual

Yasuo Arai

KEK, National High Energy Accelerator Research Organization
1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan
yasuo.arai@kek.jp, http://atlas.kek.jp/tdc/
Tel : +81-29-879-6211, Fax : +81-29-864-3284

AMT-2 ES chip produced on May, 2001
AMT-3 ES chip produced on Oct, 2003

Rev. 0.34 Dec. 10, 2009.



Contents

0. Notice

0.1. Document Change

0.2. Changes from AMT-2

(1) Bugs in trigger matching circuit, which occasionally cause hit miss and false hit, were corrected.

(2) 'strobe_select=2' was modified not to generate any strobe signal.

(3) bug fixed in error marking circuit.

(4) CSR interface through 12 bit parallel bus was modified to perform more stable operation.

(5) JTAG ID was changed to '38b85031'

(6) Modified not to interfere clock and reset signals by boundary scan.

(7) rejected_hit_error is masked in enable_reject=0.

(8) Modified to reset JTAG controller with reset signal.

(9) Add one wait to service_request to do fair arbitration.

0.3. Known Bug's in AMT-3

1. Introduction

The ATLAS Muon TDC (AMT) is a Time to Digital Converter (TDC) designed for the Monitored Drift Tubes (MDT) of the ATLAS muon detector. It is processed in Toshiba 0.3 μm CMOS Sea-of-Gate Technology (TC220G).

Basic requirements on the AMT chip were summarized in ATLAS note (MUON-NO-179, May 1997) by J. Christiansen and Y. Arai. Then AMT-0 [1] was designed in a 0.7 μm full custom CMOS process based on the 32 channel TDC for the quick test of front-end electronics and MDT chambers.

On the other hand, it was decided to use a Toshiba's 0.3 μm CMOS process for a final production. To develop and test many critical elements in the 0.3 μm process, a TEG (Test Element Group) chip (AMT-TEG, [2])was designed, fabricated and tested successfully at KEK. The AMT-TEG was processed in a new 0.3 μm process which will be used in final mass production. The present AMT-1 design is based on the AMT-0, but many modifications are done since the technology is different and many experience was obtained.

After the circuit test of AMT-1, AMT-2 chip was developed and produced on May 2001. AMT-2 chips were used in H8 test beam experiment, and also tested at Univ. of Michigan. In these tests, a serious bug was found in trigger matching circuit. This bug and several other minor bugs are fixed in AMT-3 which was produced on Oct. 2003.

A time bin size (~0.78 ns) is obtained using the basic gate delay as the base for the time measurement. This scheme prevents the use of very high speed clocks in the circuit and results in a low power device (~20 mW/channel). The gate delay of CMOS devices normally have very large variations as function of process, voltage, and temperature. In this TDC, a phase locked loop (PLL) circuit is implemented to stabilize the gate delay. Oscillation frequency of the internal ring oscillator is multiplied by two with the PLL, thus generate 80 MHz clock. This oscillator has 16 taps, and time difference of two taps are exactly 1/16 of the clock period. When a hit enters, state of these 16 taps are latched and generate a fine time. The fine time measurement is extended by a 13 bit coarse counter.

Although the PLL clock is 80 MHz, most of the logic runs at 40MHz which is same as that of the LHC clock.

Each channel can buffer 4 measurements until they can be written into a common 256 words deep level 1 buffer. The individual channel buffers works as small derandomizer buffers before the merging of hit measurements into the common L1 buffer. A trigger matching function can select events related to a trigger. The trigger information consisting of a trigger time tag and an event ID can be stored temporarily in an eight words deep trigger FIFO. Measurements matched to the trigger are passed to a 64 words deep read-out FIFO, or

A time window of programmable size is available for the trigger matching to accommodate the time spread of hits related to the same event. Optionally channels with hits in a time window before the trigger can be flagged. The trigger time tag can optionally be subtracted from the measurements so only time measurements relative to the trigger needs to be read out. Accepted data can be read out in a direct parallel format or be serialized at a programmable frequency.

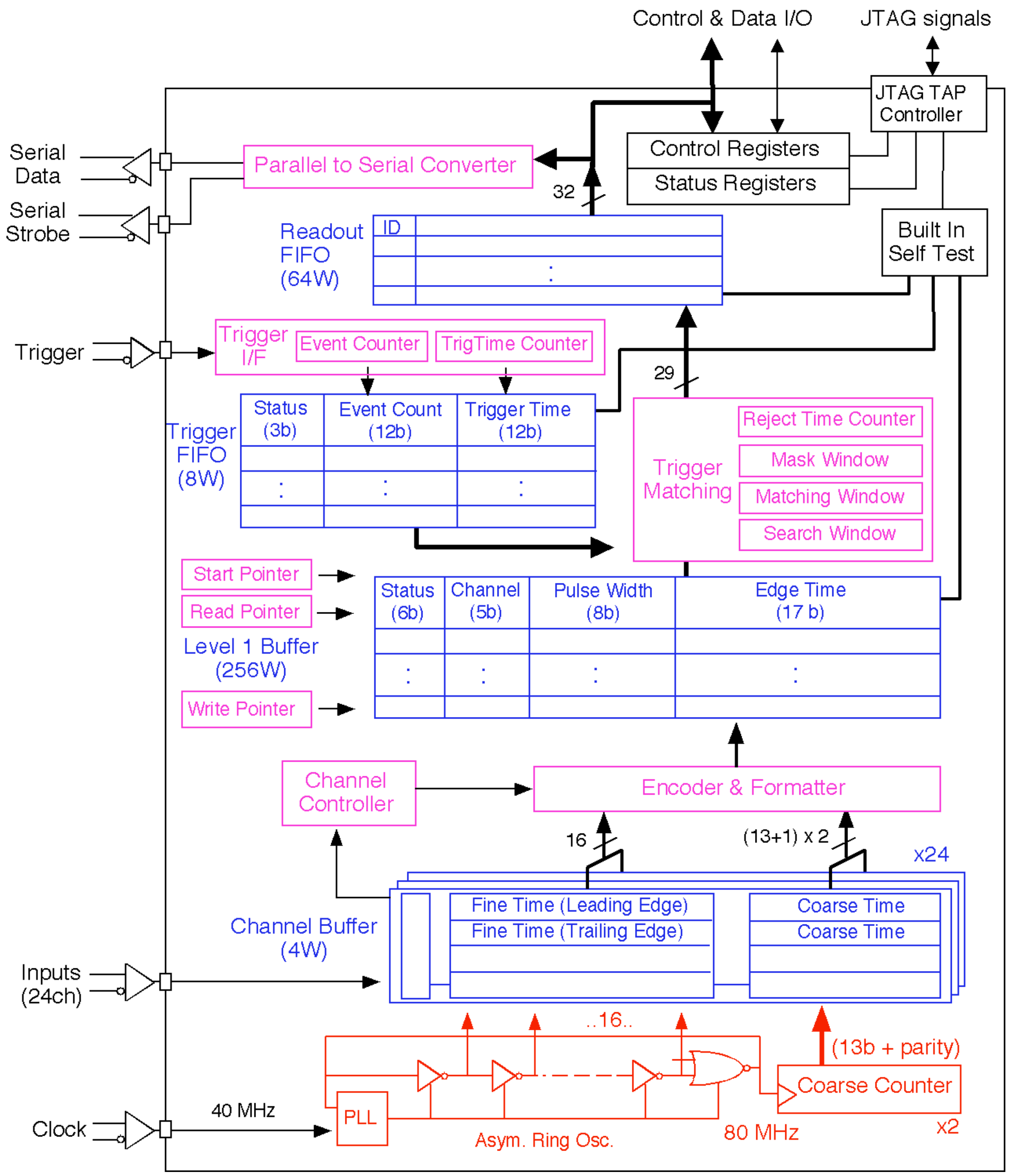


Fig. 1. Block diagram of the AMT-3

2. Circuit Description

Fig. 1 shows a block diagram of the AMT-3 chip. There are 24 channels of inputs. Table. 1 summarizes the main features of the AMT-3 chip.

Table. 1 AMT-3 MAIN FEATURES
(System Clock Frequency is 40 MHz otherwise noted)

• Least Time Count	0.78125 ns/bit (rising edge) 0.78125-100ns/bit (falling edge)

• Time Resolution	RMS = 250 ps (rising edge) ?
	RMS = 250ps ~ 29ns (falling edges)

• Dynamic range	13 + 4 = 17 bit (102.4 μsec)
• Integral Non Linearity	< 80 ps
• Differential Non Linearity	< 80 ps
• Difference between channels	Maximum one time bin
• Stability	< 0.1 LSB (3.0 - 3.6 V. 0 - 70 ˚C)
• Input Clock Frequency	10 - 70 MHz (@ x2 mode)
• PLL mode	x1, x2, x4 or x8
• Internal System Clock	Input Clock x (PLL mode) / 2
• No. of Channels	24 Channels
• Level 1Buffer	256 words
• Read-out Buffer	64 words
• Trigger Buffer	8 words
• Minimum pulse width accepted	5 ns
• Minimum time between pulses accepted	5 ns
• Maximum rate where loss of data is negligible	400 kHz/ch if all 24 channels are used. 20 MHz if only one channel is used
• Maximum trigger rate where loss of triggers is negligible	200 kHz
• Hit Input Level	Low Voltage Differential Signaling (LVDS) Internal 100 Ohm termination.
• Supply Voltage	3.3+-0.3V. <15 mW/channel (<360 mW/chip)
• Temperature range	0 - 85 Deg. Cent
• Process	0.3 μm CMOS Sea-of-Gate (Toshiba TC220G) die size: 6 mm x 6 mm
• Package	0.5 mm lead pitch, 144 pin plastic QFP

2.1. Fine Time Measurement

The original idea of the TDC, which use internal gate delay as a fine time element and stabilize the element with a feedback circuit, was born in 1986 [3]. The chip was called TMC (Time Memory Cell) chip. Initial TMC chip use a DLL (Delay Locked Circuit) technique and then PLL (Phase Locked Loop) technique has been used in recent chips.

In the PLL version, we have been using a new kind of voltage controlled ring oscillator, asymmetric ring oscillator (Fig. 2). To obtain <1 ns timing resolution, 16 taps are extracted from the oscillator. Fig. 2 shows a simplified schematics and its timing diagram of the asymmetric ring oscillator. Fig. 2 only shows 8 stages but the actual chip implements 16 stages. The asymmetric ring oscillator was creates equally spaced even number (16) of timing signals.

The PLL circuit comprises a phase frequency detector (PFD), a charge pump, a loop filter (LPF), and a voltage-controlled oscillator (VCO; asymmetric ring oscillator in this case). An external capacitor (Cvg) is required in the loop filter. The PLL has "divide by 2, 4, and 8" counter , thus the frequency of the VCO can be either the same or the "multiplied by 2, 4, or 8 " of the input frequency.

The propagation delay of the delay elements that determine the oscillation frequency of the VCO is controlled through a control voltage (VGN).

When a hit occurs, the state of the 16 taps (and coarse counter) are latched into a hit register.

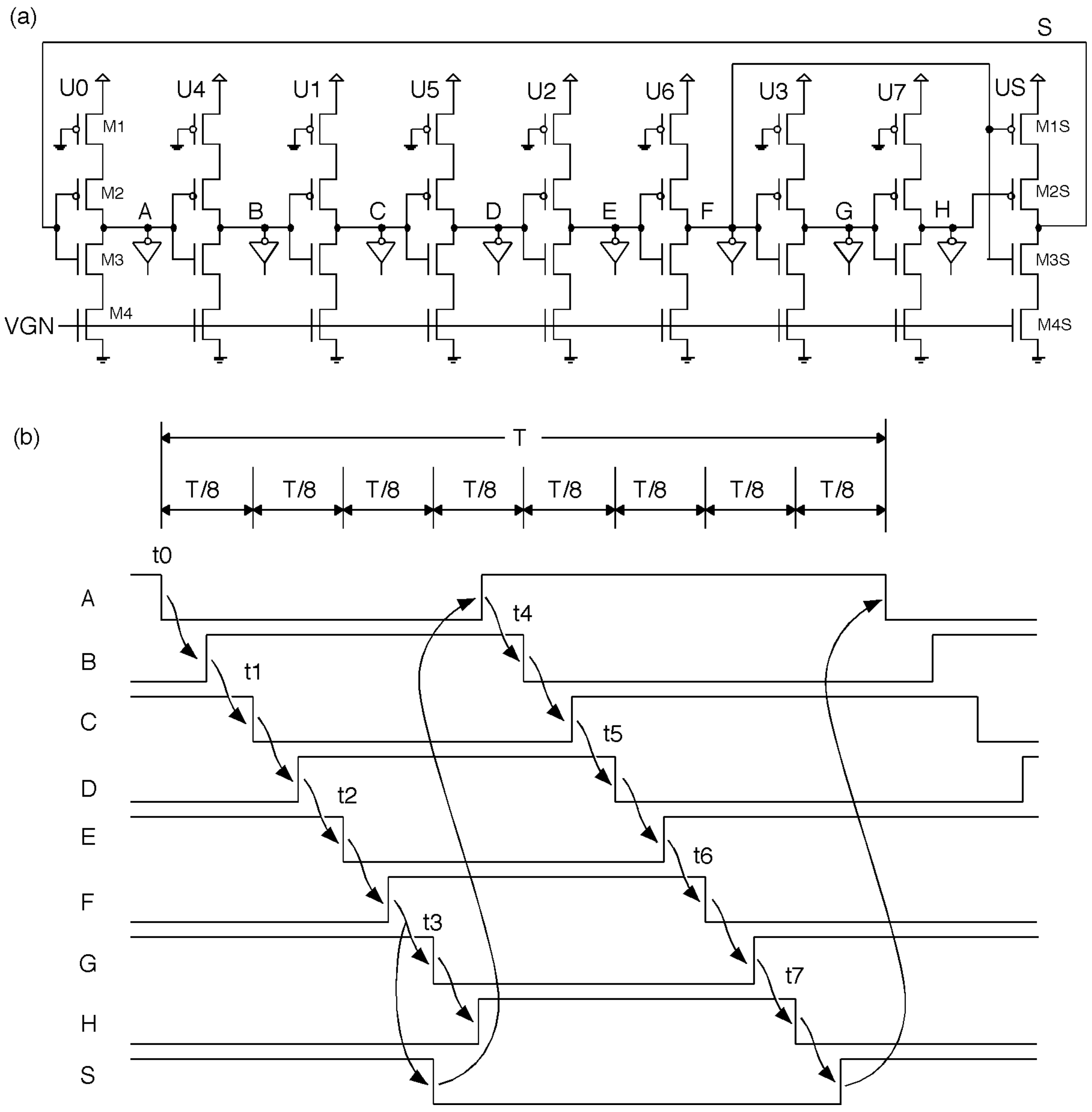


Fig. 2 (a) Asymmetric ring oscillator, (b) extracted timing signal.

2.2. Coarse Counter

The dynamic range of the fine time measurement, extracted from the state of the VCO, is expanded by storing the state of a clock synchronous counter. The hit signal may though arrive asynchronously to the clocking and the coarse counter may be in the middle of changing its value when the hit arrives. To circumvent this problem two count values, 1/2 a clock cycle out of phase, are stored when the hit arrives (Fig. 3). Based on the fine time measurement from the PLL one of the two count values will be selected, such that a correct coarse count value is always obtained.

The coarse counter has 13 bits and is loaded with a programmable coarse time offset (the LSB is always 0) at reset. The coarse counter of the TDC will in ATLAS be clocked by the two times higher frequency than the bunch crossing signal thereby the upper 12 bit of the coarse counter becoming a bunch count ID of the measurement. The bunch structure of LHC is not compatible with the natural binary roll over of the 12 bit coarse time counter. The bunch counter can therefore be reset separately by the bunch count reset signal and the counter can be programmed to roll-over to zero at a programmed value. The programmed value of this roll-over is also used in the trigger matching to match triggers and hits across LHC machine cycles.

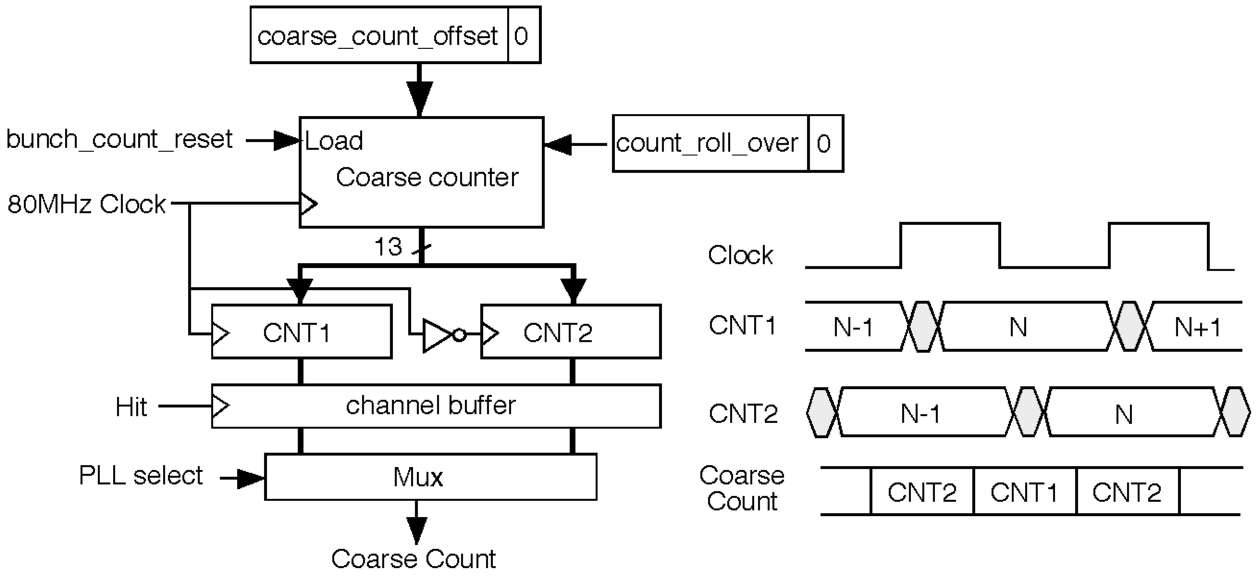


Fig. 3 Phase shifted coarse counters loaded at hit.

2.3. Channel Buffer.

Each channel can store 4 TDC measurements before being written into the common L1 buffer. The channel buffer is implemented as a FIFO controlled by an asynchronous channel controller. The channel controller can be programmed to digitize individual leading and/or trailing edges of the hit signal. Alternatively the channel controller can produce paired measurements consisting of one leading edge and the corresponding trailing edge. In paired measurement, the edge data are always handled as pair, so you can think the channel buffer is 2 words depth in this case (Fig. 4).

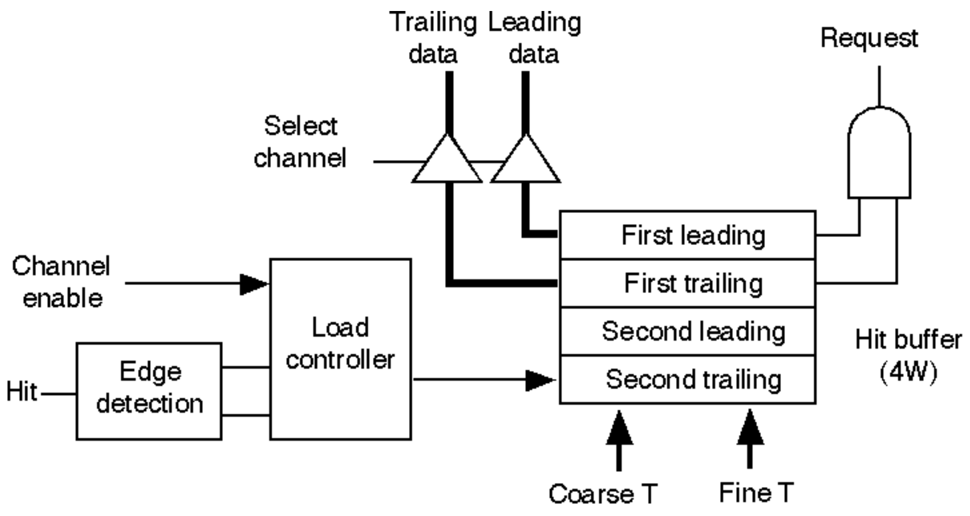
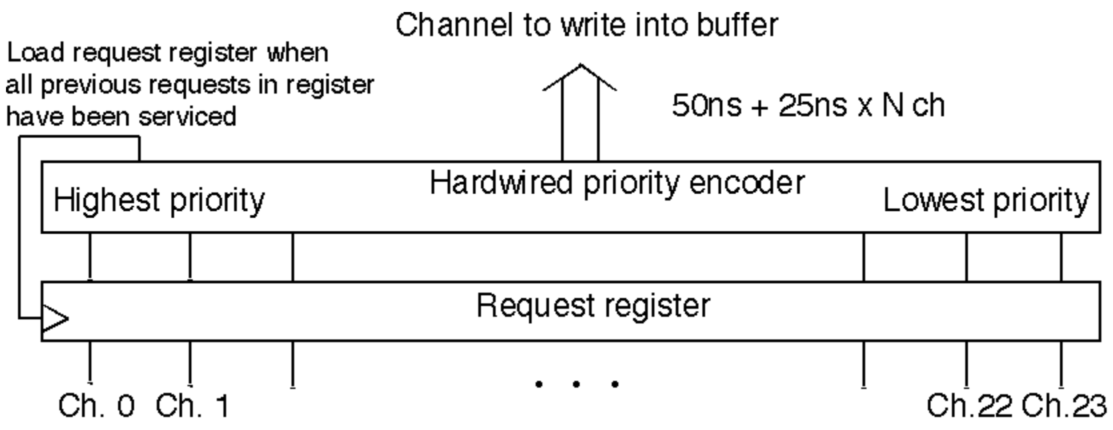


Fig. 4 The channel buffer control scheme in pair mode measurement.

If the channel buffer is full when a new hit arrives, it will be ignored. If the enable_rejected=1 and enable_match=0, the information of the rejected hit is transferred as soon as the channel buffer is available. The data of the rejected has 'hit error flag' and time when the buffer becomes available, so the time is not relevant to hit time.

For the hits stored in the channel buffers to be written into the clock synchronous L1 buffer a synchronization of the status signals from the channel buffers is performed. Double synchronizers are used to prevent any metastable state to propagate to the rest of the chip running synchronously at 40 MHz. When paired measurements of a leading and a trailing edge is performed the two measurements are taken off the channel buffer as one combined measurement.

When several hits are waiting to be written into the L1 buffer an arbitration between pending requests is performed. A registered arbitration scheme illustrated in Fig. 5 is used to give service to all channels equally. Each channel have a hardwired priority but new requests are only allowed into the arbitration queue when all pending requests in the queue have been serviced. The request queue is serviced at a rate equal to the clock frequency with 2 cycle overhead for each arbitration.



Channel to write into buffer
Load request register when all previous requests in register have been serviced
50ns + 25ns x N ch
Highest priority
Hardwired priority encoder
Lowest priority
Request register
Ch. 0 Ch. 1
. . .
Ch.22 Ch.23

Fig. 5 Registered hardwired priority arbitration of channels.

The time measurements are not written into the L1 buffer in strict temporal order. A request to be written into the first level buffer is not made before the corresponding pulse width measurement has been finalized and the channel arbitration does not take into account which hit occurred first. This have important effects on the following trigger matching.

Recording speed of the channel buffer is important to have a good double pulse resolution and edge separation. Minimum edge separation was determined by reducing pulse width and pulse separation until the hit information is lost. Fig. 6 shows an example of short pulses measurable in AMT. Four (2 leading and 2 trailing) edges are measured successfully.

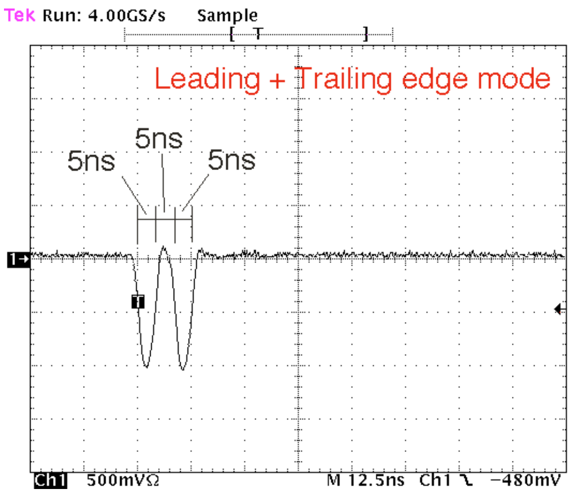


Fig. 6 An example of short pulses measurable in AMT.

2.4. Encoder

When a hit has been detected on a channel the corresponding channel buffer is selected, the time measurement done with the ring oscillator is encoded into binary form (vernier time), the correct coarse count value is selected and the complete time measurement is written into the L1 buffer together with a channel identifier.

Although the ring oscillator and the coarse counter runs at 80 MHz, the base LHC clock is 40 MHz and bunch number is counted at 40 MHz. Most of logics in the AMT are designed to run at 40 MHz. To shift from 80 MHz to 40MHz regime, we would like to define different name to measured time. We call the upper 12 bit of the coarse counter as a ‘coarse time’, and the LSB of the coarse counter plus the vernier time as a ‘fine time’. Thus the coarse time will be equivalent to the bunch count.

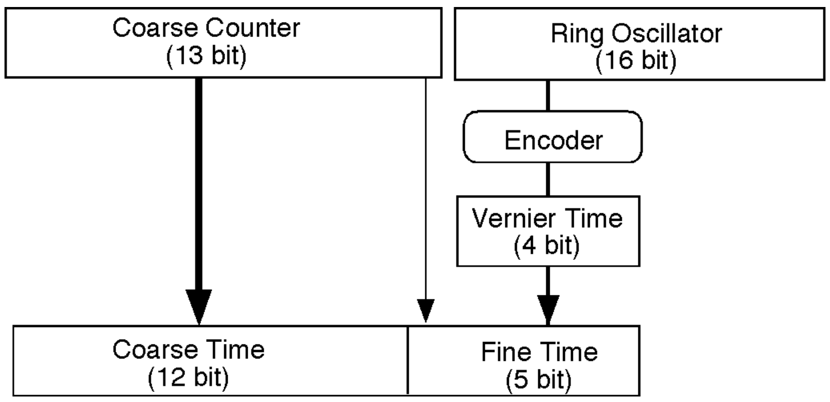


Fig. 7. Definition of coarse time and fine time.

In case a paired measurement of leading and trailing edge has been performed the complete time measurement of the leading edge plus a 8 bit pulse width is written into the L1 buffer. The 8 bit pulse width is extracted from the leading and trailing edge measurement taking into account the programmed roll-over value. The resolution of the width measurement is programmable. In case the pulse width is larger than what can be represented with a 8 bit number the width will be forced to a value of $FF (Hex).

When several hits are waiting in the channel buffers an arbitration between pending requests is performed. New hits are only allowed to enter into the active request queue when all pending requests in the queue have been serviced. Arbitration between channels in the active request queue is done with a simple hardwired priority (channel 0 highest priority, channel 23 lowest priority).

The fact that new requests only are accepted in the active request queue when the queue is empty enables all channels to get fair access to the L1 buffer.

2.5. L1 Buffer.

The L1 buffer is 256 hits deep and is written into like a circular buffer. Reading from the buffer is random access such that the trigger matching can search for data belonging to the received triggers.

When the L1 buffer only have space for one more hit (full -1) and a hit measurement arrives from the channel buffers, the hit is written into the buffer together with a special buffer full flag. After this the buffer is considered full and following hits are discarded. When 4 measurements have been removed from the buffer by the trigger matching (full - 4) and a new hit arrives, the buffer full status is cleared and the hit is written into the buffer with the buffer full flag set. This situation is illustrated in Fig. 8 where the two hits with the buffer full flag set is shown. These two hits define a time window where all hits have been discarded and this is used in the trigger matching to detect if an event potentially have lost some hits.

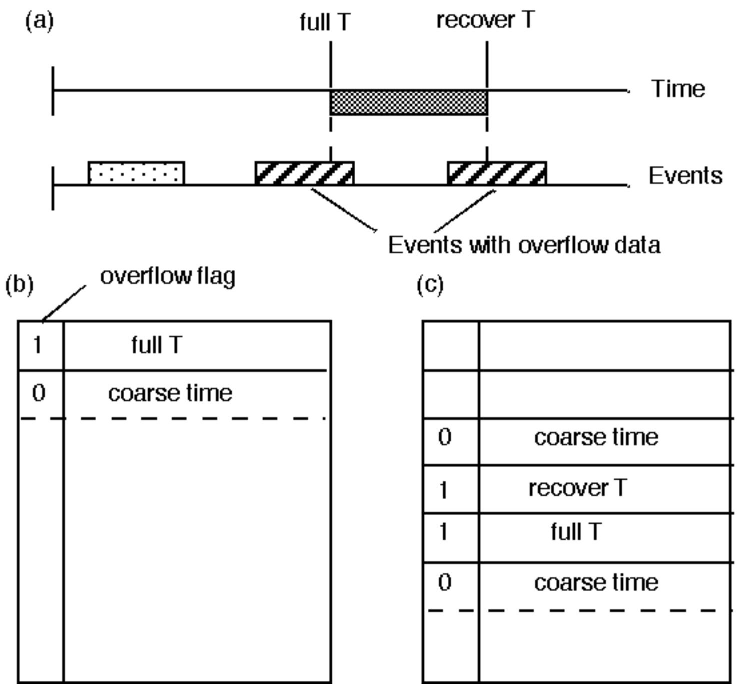


Fig. 8 First level buffer overflow handling. (a) Events of which matching time overlap with the period of overflow are marked. (b) Full time mark in the first level buffer. (c) Recover time mark.

2.6. Trigger FIFO

When a trigger is signaled the value of the bunch counter (trigger time tag) is loaded into the trigger FIFO (Fig. 9). The trigger time tag is generated from a counter with a programmable offset.

In case the trigger FIFO runs full, triggers are rejected and complete events from the TDC are potentially lost. This would have serious consequences for the synchronization of events in the readout of the TDC. A full flag in the trigger FIFO together with the event number of the triggers are used to detect the loss of triggers and to make sure that the event synchronization in the readout is never lost. If a positive trigger is signaled when the trigger FIFO is full the trigger interface stores the fact that one (or several) triggers have been lost.

As soon as the trigger FIFO is not any more full the event number of the latest lost trigger is written to the FIFO together with a trigger lost flag. The trigger matching can then detect that triggers have been lost when it sees the trigger lost flag. The trigger matching can also in this case determine how many triggers have been lost by comparing the event number of the previous trigger and the current event number stored together with the trigger FIFO full flag. This scheme is illustrated in Fig. 9 where it can be seen that triggers for event 11,12,13,14 have been lost. For each lost trigger the trigger matching will generate “empty” events (header + trailer) marked with the fact that it contains no hits because the trigger was lost.

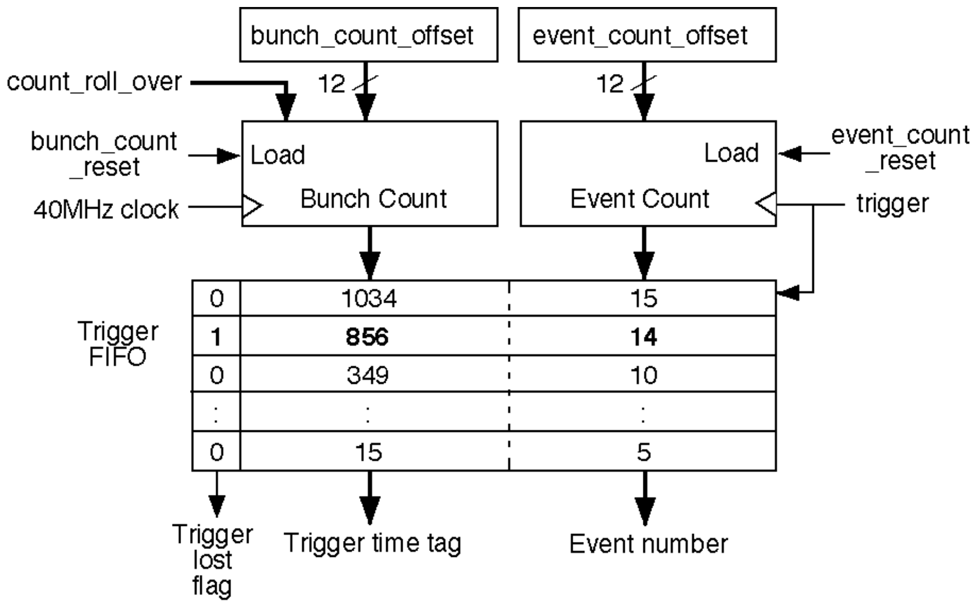


Fig. 9 Trigger FIFO and overflow handling.

2.7. Trigger Matching.

Trigger matching is performed as a time match between a trigger time tag and the time measurements them selves. The trigger time tag is taken from the trigger FIFO and the time measurements are taken from the L1 buffer. Hits matching the trigger are passed to the read-out FIFO. Optionally the trigger time tag can be subtracted from the measurements (enable_relative = 1) such that all time measurements read out are referenced to the time (bunch crossing) when the event of interest occurred.

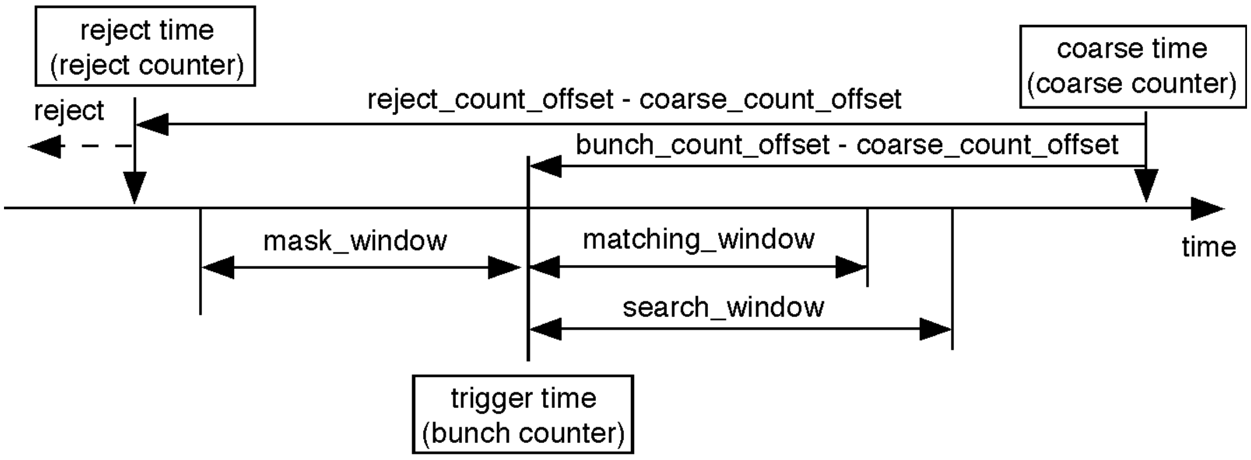

reject time (reject counter)
coarse time (coarse counter)
reject
reject_count_offset - coarse_count_offset
bunch_count_offset - coarse_count_offset
mask_window
matching_window
search_window
time
trigger time (bunch counter)

Fig. 10. Trigger latency and trigger window related to hits on channels. There are 3 time counters ; coarse time, trigger time and reject time counters. xxx_window shows window size and has positive value. 'xxx_offset is loaded into each counter when 'master_reset' or 'bunch_count_reset' is issued (see Fig. 14). The offset values are also positive values but it rolls over with 'count_roll_over' (csr8).

A match between the trigger and a hit is detected within a programmable time window (Fig. 10) if enable_match is set. The trigger is defined as the coarse time count (bunch count ID) when the event of interest occurred. All hits from this trigger time until the trigger time plus the matching window will be considered as matching the trigger. The trigger matching being based on the coarse time means that the "resolution" of the trigger matching is one clock cycle (40MHz) and that the trigger matching window is also specified in steps of clock cycles. The maximum trigger latency which can be accommodated by this scheme equals half the maximum coarse time count = $2^{12}/2$ = 2048 clock cycles = 51 us. The trigger matching function is capable of working across roll-over in all its internal time counters. For a paired measurement the trigger matching is performed on the leading edge of the input pulse.

The search for hits matching a trigger is performed within an extended search window to guarantee that all matching hits are found even when the hits have not been written into the L1 buffer in strict temporal order. For normal applications it is sufficient to make the search window ~8 larger than the match window. The search window should be extended for applications with very high hit rates or in case paired measurements of wide pulses are performed (a paired measurement is not written into the L1 buffer before both leading and trailing edge have been measured).

To prevent buffer overflow and to speed up the search time an automatic reject function can reject hits older than a specified limit when no triggers are waiting in the trigger FIFO and enable_auto_reject bit is set. A separate reject counter runs with a programmable offset to detect hits to reject.

The trigger matching can optionally search a time window before the trigger for hits which may have masked hits in the match window if enable_mask bit is set. A channel having a hit within the specified mask window will set its mask flag. The mask flags for all channels are in the end of the trigger matching process written into the read-out FIFO if one or more mask flags have been set.

In case an error condition (L1 buffer overflow, Trigger FIFO overflow, memory parity error, etc.) has been detected during the trigger matching a special word with error flags is generated if enable_errmark and corresponding enable_error bits are set.

All data belonging to an event is written into the read-out FIFO with a header and a trailer if enable_header and enable_trailer bits are set respectively. The header contains an event id and a bunch id. The event trailer contains the same event id plus a word count.

An example of setting is shown in section 4.1.

2.7.1. L1 buffer pointers

The search for hits in the first level buffer, matching a trigger, can not be performed in a simple sequential manner for several reasons. As previously mentioned the hits are not guaranteed to be written into the first level buffer in strict temporal order. In addition a hit may belong to several closely spaced triggers. A fast and efficient search mechanism which takes these facts into consideration is implemented using a set of memory pointers and a set of programmable time windows;

Write pointer:	Memory address where new hit data is written.
Read pointer:	Memory address being accessed to look for a time match.
Start pointer:	Memory address where search shall start for next trigger.

Mask window:	Time window before trigger time where masking hits are to be found.
Matching window:	Time window after trigger time where matching hits are to be found.
Search window:	Time window specifying how far the search shall look for matching hits. (extends searching range to compensate for the fact that hit data are not perfectly time ordered in the first level buffer).

The pointers are memory addresses being used during the search as illustrated in Fig. 11 and the windows are measures of time differences from the trigger time to the time of the leading edge of the hit signal.

The trigger matching search starts from the location pointed to by the start pointer. When the first masked hit or matched hit is found the start pointer is set to the new location. The search continues until a hit with a coarse time younger than the search limit has been found or there is no data in the buffer. The start pointer is also incremented while data rejection is being done.

The trigger matching can optionally be programmed to reject matched hits when the read-out FIFO is full. This rejection can be made conditional on the fact that the first level buffer is more than 3/4 full. This option will prevent event data to pile up inside the TDC. Event data that have piled up inside the TDC will take very long time before arriving to the second level buffers in the DAQ system. Here they will probably be discarded as they arrive to late. In case the TDC is not programmed to reject matched hits, the trigger matching function will stop when the readout FIFO is full and the l1 buffer and the trigger FIFO will start to fill up.

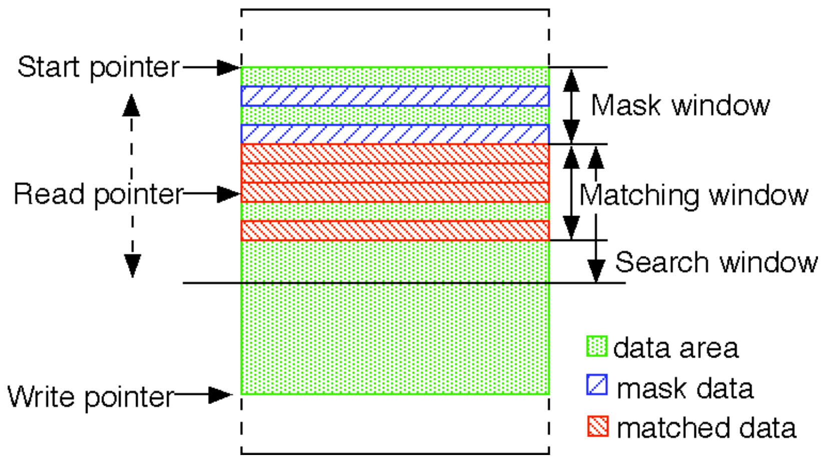


Fig. 11 The L1 buffer pointers and time windows.

2.7.2. Parallel processing

The signal path from the trigger and hit inputs to the readout contains several buffers so multiple events can be processed at the same time as shown in Fig. 12 One trigger in the process of being received, One event in the process of being trigger matched and a third event in the process of being read out.

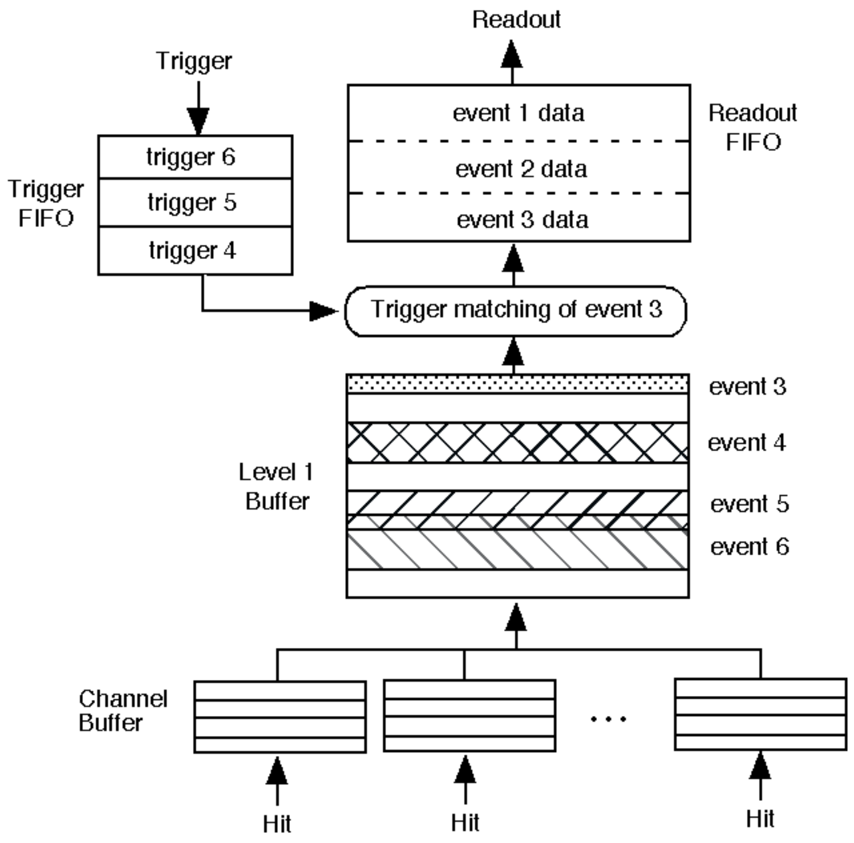


Fig. 12 Processing of several events in parallel.

2.7.3. Shared Hits

In many case the drift time of a chamber may be longer than the minimum interval between two triggers, so some hits may be shared by several triggers. This is illustrated in Fig. 13 where some hits are shared for event N and N+1. Randomly accessed memory is used instead of a FIFO for the L1 buffer to handle shared data.

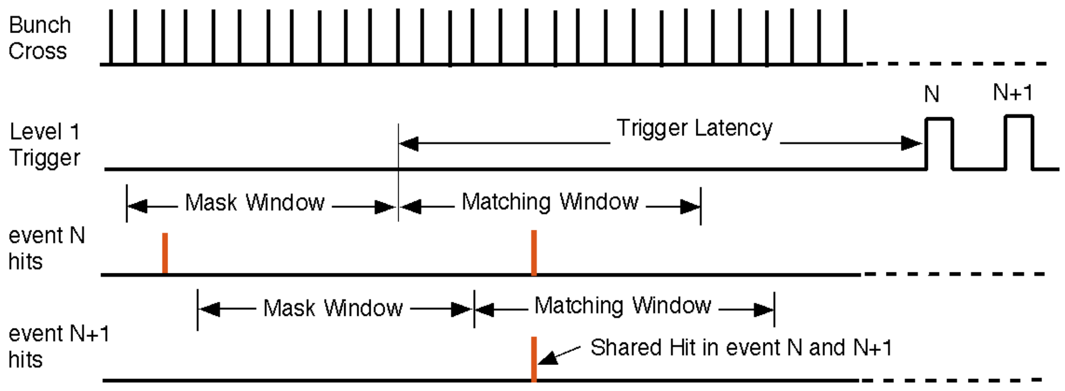


Fig. 13 Mask & Matching window and a shared hit.

2.7.4. No matching mode

The trigger matching function may also be completely disabled (enable_matching = 0) whereby all data from the L1 buffer is passed directly to the read-out FIFO. In this mode the

TDC have an effective FIFO buffering capability of 256 + 64 = 320 measurements. Since the matching circuit which control the L1 buffer is disabled, there is no overflow control to the L1 buffer. Thus old data will be overwritten by new data if the overflow occur.

2.8. Trigger & Reset Interface.

The trigger interface takes care of receiving the trigger signal and generate the required trigger time tag to load into the trigger FIFO. In addition it takes care of generating and distributing all signals required to keep the TDC running correctly during data taking.

The TDC needs to receive a global reset signal that initializes and clears all buffers in the chip before data taking. A bunch count reset and event count reset is required to correctly identify the event ID and the bunch ID of accepted events. These signals can either be generated separately or be coded on a single serial line at 40 MHz.

2.8.1. Encoded trigger and resets

Four basic signals are encoded using three clock periods (Table 1). The simple coding scheme is restricted to only distribute one command in each period of three clock periods. A command is signaled with a start bit followed by two bits determining the command. When using encoded trigger and resets an additional latency of three clock periods is introduced by the decoding compared to the use of the direct individual trigger and resets.

Selection between the encoded signals and the direct signals are schematically shown in Fig. 14.

Table 1 Encoded signal bit pattern.

Meaning	bit 2 1 0
Trigger	1 0 0
Bunch count reset	1 1 0
Global reset	1 0 1
Event count reset	1 1 1

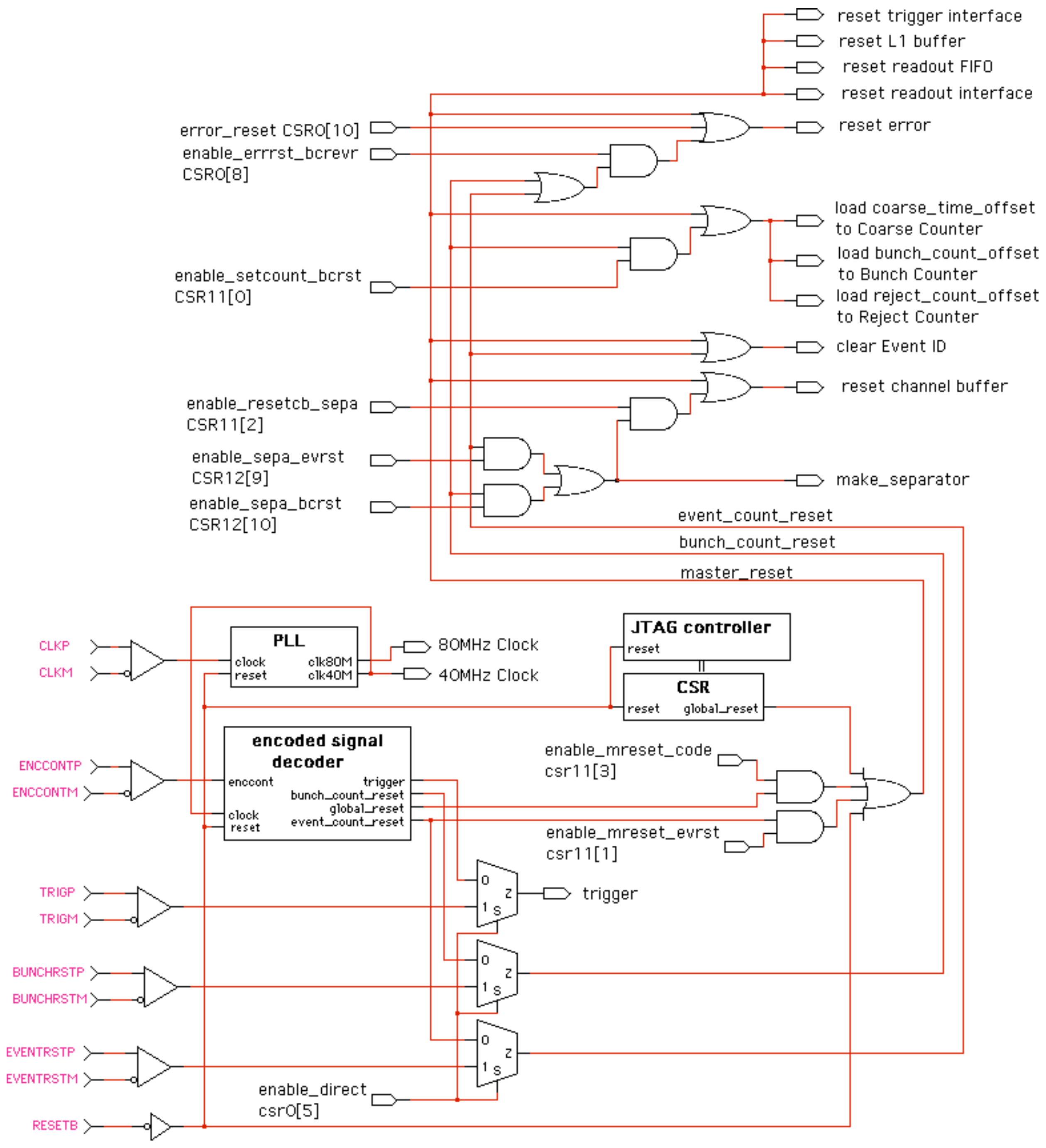


Fig. 14. Simplified diagram of the clock, trigger and reset signals.

2.8.2. Event count reset

An event count reset loads the programmed event_count_offset into the event ID counter. In addition a separator can be injected into the L1 buffer and the trigger FIFO if enable_sepa_evrst is set.

2.8.3. Bunch count reset.

The bunch count reset loads the programmed offsets into the coarse time counter, the trigger time tag (bunch id) counter and the reject counter. In addition a separator can be injected into the L1 buffer and the trigger FIFO if enable_sepa_bcrst is set.

From a bunch count reset is given to the TDC until this is seen in the hit measurements themselves, a latency of the order of 2 clock cycles is introduced by internal pipelining of the coarse time counter. The definition of time 0 in relation to the bunch count reset is described in more detail in section 4.

2.8.4. Global reset.

The global reset generate master_reset signal if enable_mreset_code is set (see Fig. 14). The master_reset clears all buffers in the TDC and initializes all internal state machines to their initial state. Before data taking an event count reset and a bunch count reset must also have been issued. As shown in Fig. 14, decoder circuit, CSR, JTAG contrioller and PLL circuit are not reset by the master_reset.

2.8.5. Trigger

The basis for the trigger matching is a trigger time tag locating in time where hits belong to an event of interest. The first level trigger decision must be given as a constant latency yes/no trigger signal. The trigger time tag is generated from a counter with a programmable offset. When a trigger is signaled the value of the bunch counter (trigger time tag) is loaded into the trigger FIFO. The effective trigger latency using this scheme equals the difference between the coarse_time_offset and the bunch_count_offset.

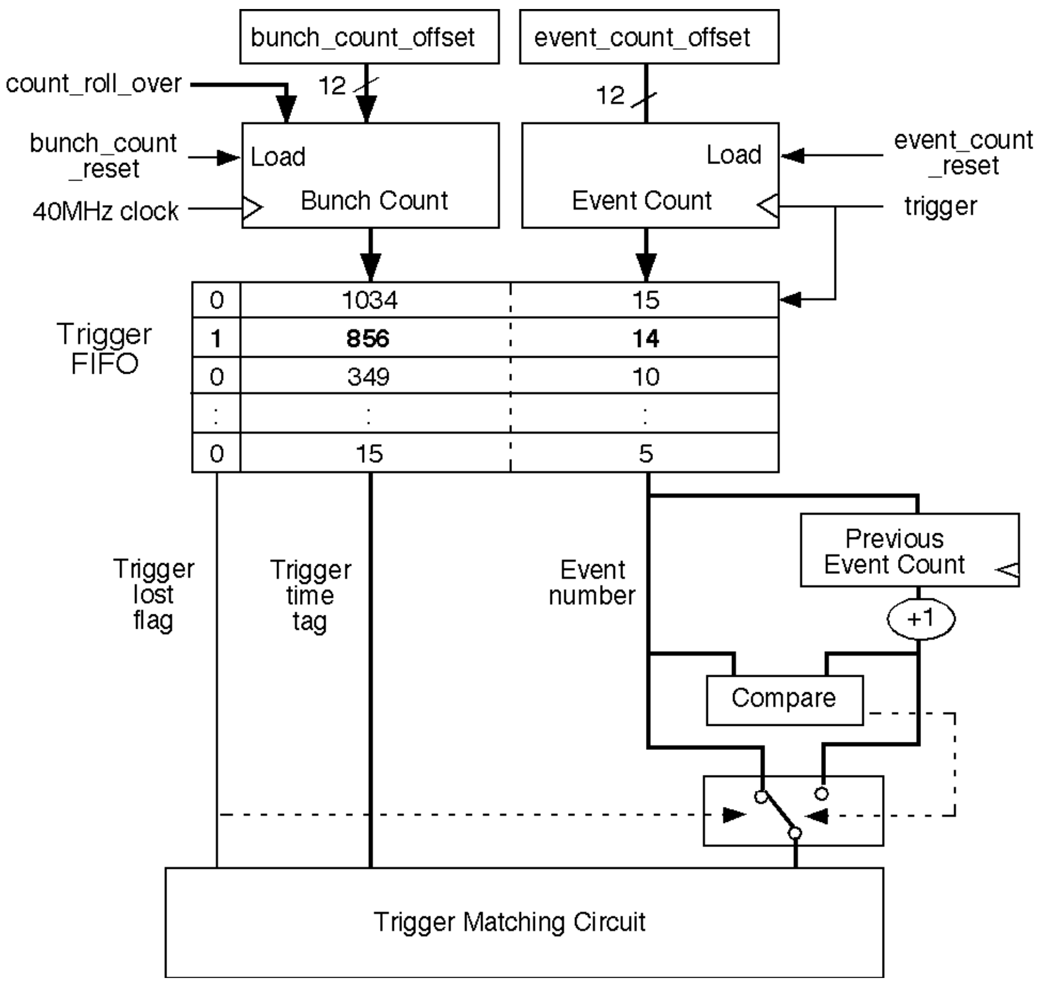


Fig. 15. Generation of trigger data.

If the trigger FIFO runs full the trigger time tags of following events will be lost. The trigger interface keeps track of how many triggers have been lost so the event synchronization in the trigger matching and the DAQ system is never lost. For each event with a lost trigger time tag the trigger matching will generate an event with correct event id and a special error flag signaling that the whole event has been lost.

2.8.6. Separators

The TDC is capable of running continuously even when bunch count resets and event count resets are issued. Matching of triggers and hits across bunch count resets (different machine cycles) are handled automatically if the correct roll-over value have been programmed.

Alternatively it is possible to insert special separators in the trigger FIFO and the L1 buffer when a bunch count reset or an event count reset have been issued (enable_sepa_bcrst or enable_sepa_evrst). These will make sure that hits and triggers from different event count or bunch count periods (machine cycles) never are mixed. In this mode it is not possible to match hits across bunch count periods.

This separator can be readout if enable_sepa_readout bit is set. This mechanism is conceptually shown in Fig. 16. A caution must be paid that a hit will be lost when the separator is inserted into the L1 buffer. Furthermore, if enable_resetcb_sepa is set, all data within the channel buffer is cleared when the separator is inserted.

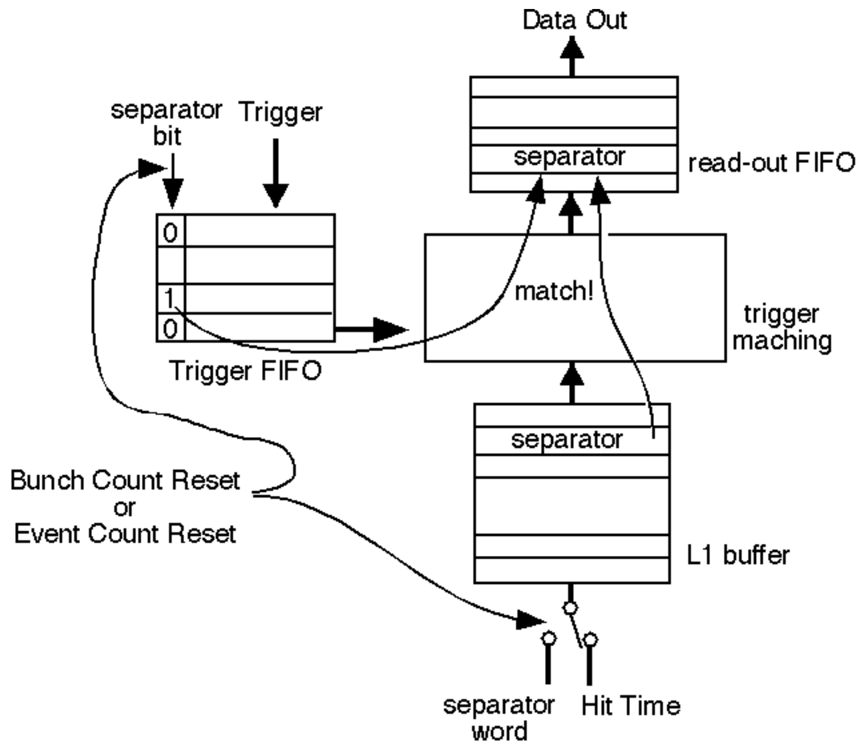


Fig. 16 Conceptual view of the separator insertion and matching.

2.9. Read-out FIFO and Buffer Full Control.

The read-out FIFO is 64 words deep and its main function is to enable one event to be read out while another is being processed in the trigger matching. If the read-out FIFO runs full there are several options of how this will be handled.

Back propagate:

(enable_rofull_reject =0, enable_l1full_reject = x, enable_trfull_reject = x)
The trigger matching process will be blocked until new space is available in the read-out FIFO. When this occurs, the L1 buffer and the trigger FIFO will be forced to buffer more data. If this situation is maintained for extended periods the L1 buffer or the trigger FIFO will finally become full and the measurement is stopped.

Nearly full reject:

(enable_rofull_reject =1. enable_l1full_reject = 1 and/or enable_trfull_reject = 1)
In this mode the trigger matching will be blocked if the read-out FIFO becomes full and either the L1 buffer is nearly full (256 - 64 = 191 words occupied) or the trigger FIFO is nearly full (contains 4 triggers). If this occurs event data will be rejected to prevent the L1 buffer and the trigger FIFO to overflow. The event header and event trailer data will never be rejected as this would mean the loss of event synchronization in the DAQ system. Any event which have lost data in this way will be marked with an error flag.

Reject data:

(enable_rofull_reject =1, enable_l1full_reject = 0, enable_trfull_reject = 0):
As soon as the read-out FIFO full, event data (not event headers and trailers) will be rejected. Any loss of data will be signaled with an error flag.

To show the difference of these mode, an simulation example is shown in Fig. 17. At low input rate there is little difference in these mode, but at high rate L1 buffer occupancy and event delay will be very different [4].

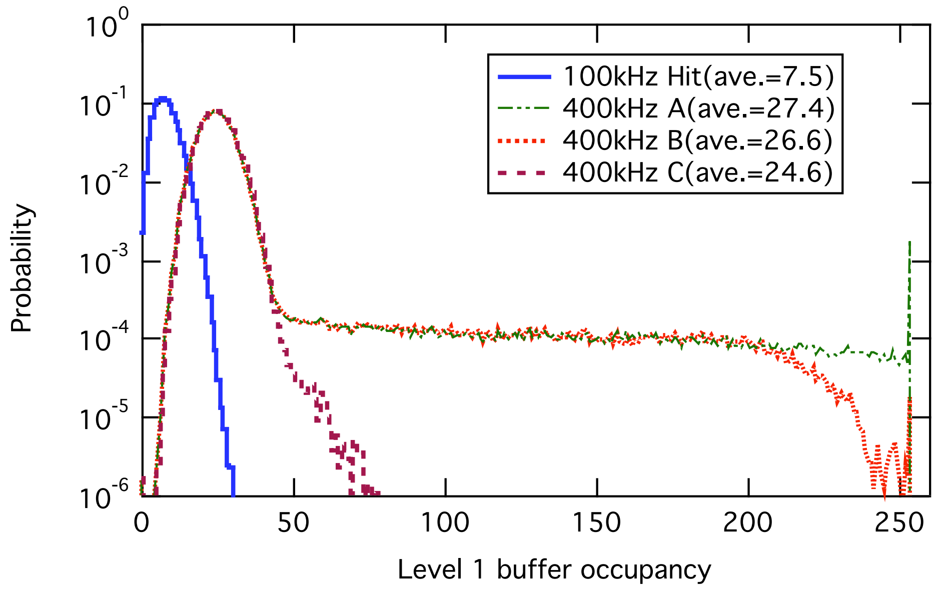


Fig. 17 An example of simulation for the level 1 buffer occupancy at 100kHz and 400kHz hit input rate in all 24 channels. Data readout is 40MHz serial. Case A is 'Back Propagate', case B is 'nearly full reject' and case C is 'Reject data'. Case C has less occupancy since the data is soon rejected from L1 buffer when readout FIFO becomes full.

2.10. Read-out Interface.

All accepted data from the TDC can be read out via a parallel or serial read-out interface in words of 32 bits. The event data from a chip typically consists of a event header (if enabled), accepted time measurements, mask flags (if enabled), error flags (if any error detected for event being read out) and finally a event trailer (if enabled).

2.10.1. Parallel read-out

Read out of parallel data from the TDC is enabled by setting enable_serial=0 and performed via a clock synchronous bus. Several TDC's may share one read-out bus, and each TDC will be selected with CS (chip select) and DSPACE (data space) signals.

The read out of individual hits are controlled by a DREADY (data ready) / GETDATA (get data) handshake. If the GETDATA signal is constantly held active (independent of DREADY) it is interpreted as the read-out can be performed at the full speed of the TDC. The effective read-out speed can be slowed down by using the DREADY / GETDATA handshake protocol to introduce wait cycles. The number of clock periods that the GETDATA signal is asserted is used to determine the word count for the event available in the global trailer.

2.10.2. Serial read-out

The accepted TDC data can be transmitted serially over twisted pairs using LVDS signals by setting enable_serial = 1. Data is transmitted in words of 32 bits with a start bit set to one and followed by a parity bit and 2 stop bit. The parity is odd parity (exclusive OR of the 32 data bits). If number of '1' bits in the data is odd, the parity bit will be '1'. The serialization speed is programmable from 80 to 10 Mbits/s.

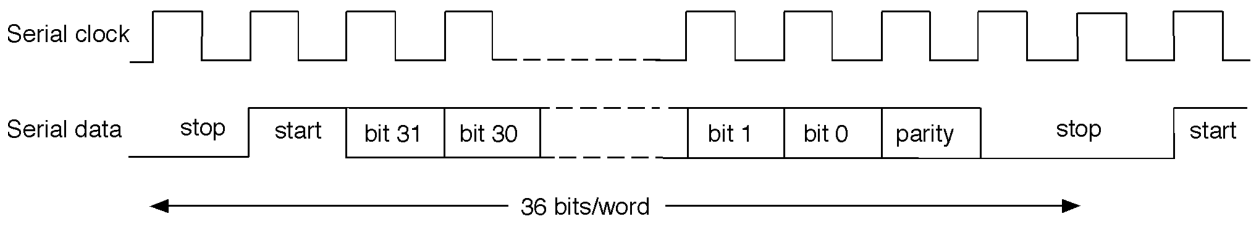


Fig. 9 :Serial frame format with start bit and parity bit

In addition to the serialized data an LVDS pair can carry strobe information in a programmable format as shown in Fig. 18.

Leading Strobe:	Direct serializing clock to strobe data on rising edge.
DS Strobe :	DS strobe format as specified for transputer serial links. DS strobe only changes value when no change of serial data is observed.

For each format, there are two modes , one is continuous strobe regardless of data existence, another mode generate strobe signal only when data is available.

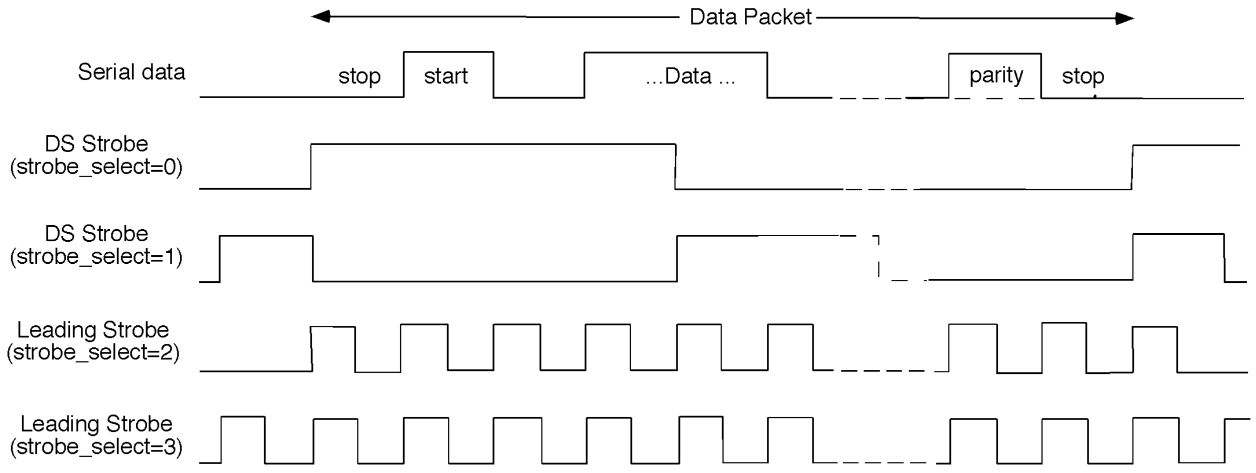

Data Packet
Serial data
stop
start
...Data ...
parity
stop
DS Strobe (strobe_select=0)
DS Strobe (strobe_select=1)
Leading Strobe (strobe_select=2)
Leading Strobe (strobe_select=3)

Fig. 18. Different strobe types.

2.10.3. Data format

Data read out of the TDC is contained in 32 bits data packets. For the parallel read-out mode one complete packet (word) can be read out in each clock cycle. In the serial read-out mode a packet is sent out bit by bit. The first four bits of a packet are used to define the type of data packet.

The following 4 bits are used to identify the ID of the TDC chip (programmable) generating the data. Only 7 out of the possible 16 packet types are defined for TDC data. The remaining 9 packet types are available for data packets added by higher levels of the DAQ system.

TDC header: Event header from TDC

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
1	0	1	0	TDC ID				Event ID												Bunch ID											

TDC: Programmed ID of TDC.
Event ID: Event ID from event counter.
Bunch ID: Bunch ID of trigger (trigger time tag).

TDC trailer: Event trailer from TDC

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
1	1	0	0	TDC ID				Event ID												Word Count											

Word count: Number of words from TDC (incl. headers and trailers).

Mask flags: Channel flags for channels having hits with in mask window

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
0	0	1	0	TDC ID				Mask flags																							

Mask flags: Channels flagged as having hits with in mask window.

Single measurement data: Single edge time measurement

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
0	0	1	1	TDC ID				Channel					T	E	Coarse Time												Fine Time				

Channel: TDC channel number.
Coarse Time: Coarse time measurement (in bins of 25ns).
Fine Time: Fine time measurement from PLL (in bins of 25ns/32).
T: Edge type: 1 = leading edge, 0 = trailing edge.
E: Error. An error has been detected in the hit measurement (a coarse counter error, a channel select error, or a rejected hit error)

Combined measurement data: Combined measurement of leading and trailing edge

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
0	1	0	0	TDC ID				Channel					Width								Coarse Time						Fine Time				

Width: Width of pulse in programmed time resolution (CSR9 width_select). If the pulse width excess the width range (overflow), the width will be $FF. If trailing edge time becomes less than leading edge time due to count rollover (underflow), the width will be $00.
Coarse Time: Coarse time measurement of leading edge relative to trigger (in bins of 25ns).
Fine Time: Fine time measurement of leading edge (in bins of 25ns/32).

Errors: Error flags sent if an error condition have been detected

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
0	1	1	0	TDC ID														Error flags													

Error flags: see Table. 2 and Table. 3.

Debugging data: Additional information for system debugging

<u>Separator</u>

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
0	1	1	1	TDC ID				0	0	0	0	-								Bunch ID											

Bunch ID: Trigger time tag counter when separator was generated.

<u>Buffer Occupancy</u>

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
0	1	1	1	TDC ID				0	0	0	1	-											R	L1 Occupancy							

L1 occupancy: L1 buffer occupancy.
R: Read-out FIFO full.

2.10.4. Event format

Event format is shown in Fig. 19. Each word is controlled in corresponding control bits, and only exists when enabled.

	corresponding control bits
:	
Header	enable_header
Hard Error	enable_error, enable_errmrk
Single/Combined data	enable_pair, enable_leading, enable_trailing
:	:
Mask Flag	enable_mask
Temporal Error	enable_errmrk_ovr, enable_errmrk_rejected
Trailer	enable_trailer
Debugging data	enable_sepa_readout, enable_sepa_bcrst, enable_sepa_evrst, enable_l1occup_readout
Header	
:	

Fig. 19 Event Format

2.11. Error monitoring.

There are two kinds of error; hard errors and temporal errors. Hard errors are enabled with enable_error bits (CSR12[8:0]) and read through error_flags (CSR16[8:0]). Temporal errors are such as buffer overflow, and resumed automatically if data rate decreases. These errors are embedded within data stream and available only through error word.

Since the hard error and temporal error are generated separately, the error word only contains one kind of error flags.

2.11.1. Hard Errors

All functional blocks in the TDC are continuously monitored for error conditions. Memories are continuously checked with parity on all data. All internal state machines have been implemented with a “one hot” encoding scheme and is checked continuously for any illegal state.

The JTAG instruction register have a parity check to detect if any of the bits have been corrupted during down load. The CSR control registers also have a parity check to detect if any of the bits have been corrupted by a Single Event Upset (SEU). The error status of the individual parts can be accessed via the CSR status registers (Table. 2).

Any detected error condition in the TDC sets its corresponding error status bit. The error bits are reset by a global reset, or when error_reset (CSR0[10]) bit is set, or when error (error word) is marked. The error bits are also reset by bunch count reset or event count reset if enable_errrst_bcrevr is set. All the available error flags are OR-ed together with individual programmable mask bits to generate an ERROR signal. When the ERROR signal becomes active the TDC can respond following ways:

Ignore:	No special action will be performed (enable_errmark = 0)
Mark events:	All events being generated after the error has been detected will be marked with a special error flag (enable_errmark = 1).

Table. 2. Hard Errors (comes after header word)

Error flag	bit# in enable_error, error_flags, and error word	Description
Coarse count error	0	A parity error in the coarse count has been detected in a channel buffer.
Channel select error	1	A synchronization error has been detected in the priority logic used to select the channel being written into the L1 buffer. (more than 1 channel are selected)
L1 buffer error	2	Parity error detected in L1 buffer.
Trigger FIFO error	3	Parity error detected on trigger FIFO.
Matching state error	4	Illegal state detected in trigger matching logic.
Read-out FIFO error	5	Parity error detected in read-out FIFO.
Read-out state error	6	Illegal state detected in read-out logic.
Control parity error	7	Parity error detected in control registers.
JTAG error	8	Parity error in JTAG instruction.

* Matching State Error

The matching state error (bit 4) is set when state sequencer of trigger matching found an illegal state. The state register has 11 bits wide and only 1 bit is set for normal state. If more than a bit are set or no bit is set, the error occurs.

below relevant Verilog code is shown.

```
----------------------------
output[10:0]        state;
parameter
     state_waiting =                          11'b00000000001,
     state_write_event_header =               11'b00000000010,
     state_write_occupancy =                  11'b00000000100,
     state_active =                           11'b00000001000,
     state_write_mask_flags =                 11'b00000010000,
     state_write_error =                      11'b00000100000,
     state_write_event_trailer =              11'b00001000000,
     state_generating_lost_event_header =     11'b00010000000,
     state_generating_lost_event_trailer =    11'b00100000000,
     state_waiting_for_separator =            11'b01000000000,
     state_write_separator =                  11'b10000000000;

//state checking
always @(posedge clk)
     begin
     if (reset)
          state_error <= #1 0;
     else
          state_error <= #1 state_error |
                ~( (state == state_waiting ) |
                (state == state_write_event_header ) |
                (state == state_write_occupancy ) |
                (state == state_active ) |
                (state == state_write_mask_flags ) |
                (state == state_write_error ) |
                (state == state_write_event_trailer ) |
                (state == state_generating_lost_event_header ) |
                (state == state_generating_lost_event_trailer ) |
                (state == state_waiting_for_separator ) |
                (state == state_write_separator ) );
     end
-----------------------
```

Therefore If this error happens, this means something wrong is happened in the state register. It may be due to Single Event upset or the register bit failure etc. Once this error happened, the error will continue until L1 buffer reset (global_reset via encoded signal, CSR0[11], or hardware reset) is asserted.

2.11.2. Temporal Errors

Temporal errors are embedded in data and available only through an error word. Effective error bits are different depend on the setting of 'enable_match'.

Table. 3. Temporary Error (comes after data)

enable_ match CSR10[9]	Error flag	bit# in error word	Description	Comments
0	L1 buffer overflow	9	L1 buffer becomes full and hit data have been lost	Enabled by 'enable_errmark_ovr'
1	L1 buffer overflow	9	L1 buffer becomes full and hit data have been lost	
1	Trigger FIFO overflow	10	Trigger fifo becomes full and events have been lost	= trigger_lost
1	Readout FIFO overflow	11	Readout fifo becomes full and hit data have been lost	= enable_rofull_reject & readout_fifo_full & (enable_l1full_reject & nearly_full | enable_trfull_reject & trigger_fifo_nearly_full | ~enable_l1full_reject & ~enable_trfull_reject)
1	Hit error	12	A hit measurement have been corrupted (channel select error or coarse count error)	This error is set when the data with 'hit_error' comes in trigger matching circuit..
0 (*)	Channel buffer overflow	13	Channel buffer becomes full	This error is set when the data with 'rejected' comes and enable_rejected=1.

(*) Since the channel buffer overflow can be reported after the channel buffer become available, the reported time is overflow end time not start time. Therefore, in matching mode, it is difficult to report the channel buffer overflow combined with related hit data. Thus it is not reported in matching mode.

3. CSR Registers & JTAG access

There are two kinds of 12 bits registers, "CONTROL" and "STATUS" registers. The "CONTROL" registers are readable and writable registers which control the chip functionality. The "STATUS" registers are read only registers which shows chip statuses.

There are 15 Control registers (CSR0-14) and 6 Status registers (CSR16-21). These registers are accessible from 12 bit bus (DIO[11:0]) or through JTAG interface.

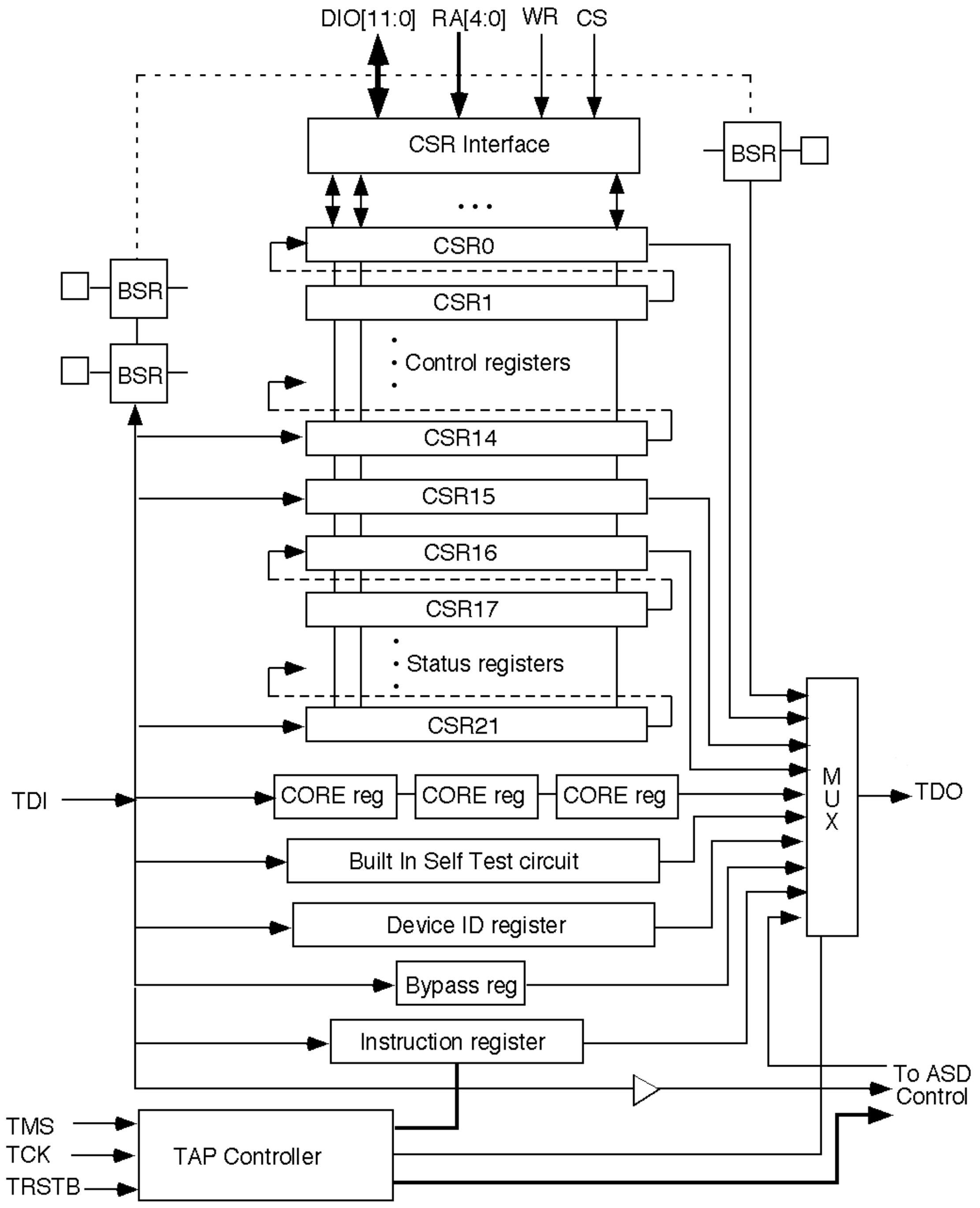


Fig. 20. Structure of JTAG - CSR registers.

3.1. Control registers

Table. 4. Bit assignment of the control registers.

	BIT											
	11	10	9	8	7	6	5	4	3	2	1	0
CSR0	global_reset (0) {11}	error_reset (0) {10}	disable_encode (0) {9}	enable_errrst_bcrevr (0) {8}	test_mode (0) {7}	test_invert (0) {6}	enable_direct (0) {5}	disable_ringosc (0) {4}	clkout_mode (0) {3:2}		pll_multi (0) {1:0}	
CSR1	mask_window (0) {23:12}											
CSR2	search_window (0) {35:24}											
CSR3	match_window (0) {47:36}											
CSR4	reject_count_offset (0) {59:48}											
CSR5	event_count_offset (0) {71:60}											
CSR6	bunch_count_offset (0) {83:72}											
CSR7	coarse_time_offset (0) {95:84}											
CSR8	count_roll_over (FFF) {107:96}											
CSR9	strobe_select (1/0) {119:118}		readout_speed (0) {117:116}		width_select (0) {115:113}			error_test (0) {112}	tdc_id (0) {111:108}			
CSR10	enable_auto_reject (1) {131}	enable_l1occup_readout (0) {130}	enable_match (1) {129}	enable_mask (0) {128}	enable_relative (0) {127}	enable_serial (0) {126}	enable_header (0) {125}	enable_trailer (0) {124}	enable_rejected (0) {123}	enable_pair (0) {122}	enable_trailing (0) {121}	enable_leading (1) {120}
CSR11	enable_rofull_reject (0) {143}	enable_l1full_reject (0) {142}	enable_trfull_reject (0) {141}	enable_errmark (0) {140}	inclk_boost (0) {139}	enable_ermark_rejected (0) {138}	enable_errmark_ovr (0) {137}	enable_l1ovr_detect (0) {136}	enable_mreset_code (0) {135}	enable_resetcb_sepa (0) {134}	enable_mreset_evrst (0) {133}	enable_setcount_bcrst (0) {132}
CSR12	enable_sepa_readout (0) {155}	enable_sepa_bcrst (0) {154}	enable_sepa_evrst (0) {153}	enable_error [8:0] (1FF) {152:144}								
CSR13	enable_channel[11:0] (FFF) {167:156}											
CSR14	enable_channel[23:12] (FFF) {179:168}											
CSR15	general_out[11:0] (0)											

() -- initial value at reset. {} – JTAG bit No.

3.1.1. CSR0

pll_multi[1:0] : The frequency ration between the external clock and the internal ring oscillator frequency is determined by these bits as shown below.

pll_multi[1]	pll_multi[0]	Input frequency : Synthesized frequency
0	0	1 : 2
0	1	1 : 4
1	0	1 : 8
1	1	1 : 1

clkout_mode[1:0] : CLKOUT pin mode.
0 -- Start_Sync : Synchronized output of the START signal.
1 -- 40MHz clk : PLL clock / 2 output. This is the system clock used in most of logics.
2 -- 80 MHz clk : PLL clock outpuut.
3 -- Coarse Counter Carry. : Carry outout of the Coarse Counter. This can be used to extend the time range.

disable_ringosc : Stop the oscillation of the ring oscillator and disable charge pump circuit of the PLL.

enable_direct : enable direct input pins.
0 --- trigger/reset signals are came from encoded input (encontp/enccontm),
1 --- trigger/reset signals are came from direct input pins (trigp/trigm, bunchrstp/bunchrstm, eventrstp/eventrstm).

test_invert : automatic inversion of test pattern in test mode.

test_mode : enable test mode.

enable_errrst_bcrevr : enable error reset when bunch count reset or event count reset is came.

disable_encode : disable fine time encoder and output 111.

error_reset : clear error bits in CSR16[8:0].

global_reset : Global reset of the TDC. Write '1' to this bit generate 'master_reset' signal (see Fig. 14). Write '0' to resume operation.

3.1.2. CSR1

mask_window[11:0] Mask window size. Actual width of the window is mask_window * clock cycles.

3.1.3. CSR2

search_window[11:0] Search window size. Actual width of the window is (search_window +1) * clock cycles

3.1.4. CSR3

match_window[11:0] Matching window size . Actual width of the window is (match_window +1) * clock cycles

3.1.5. CSR4

reject_count_offset[11:0] rejection counter offset

3.1.6. CSR5

event_count_offset[11:0]	event number offset

3.1.7. CSR6

bunch_count_offset[11:0]	trigger time tag counter offset

3.1.8. CSR7

coarse_time_offset[11:0]	coarse time counter offset

3.1.9. CSR8

count_roll_over[11:0]	counter roll over value

3.1.10. CSR9

tdc_id[3:0]	TDC identifier. This ID is attached to the output data.
error_test	set all error flags to test error circuit. Don't set this bit in normal operation.

width_select[2:0] : Set pulse width resolution in pair measurement.

width select	width output	resolution(*)	max. width(*)
0	full_width[7:0]	0.78125 ns	200 ns
1	full_width[8:1]	1.5625 ns	400 ns
2	full_width[9:2]	3.125 ns	800 ns
3	full_width[10:3]	6.25 ns	1.6 µs
4	full_width[11:4]	12.5 ns	3.2 µs
5	full_width[12:5]	25 ns	6.4 µs
6	full_width[13:6]	50 ns	12.8 µs
7	full_width[14:7]	100 ns	25.6 µs

(*) at 40 MHz clock.

readout_speed[1:0] :

readout_speed	speed
0	40Mbps
1	20 Mbps
2	10 Mbps
3	80 Mbps

strobe_select[1:0] :

enable_serial	strobe_select	strobe mode
1	0	gated DS strobe
	1	continuous DS strobe
	2	**no strobe signal** (AMT-2: gated leading edge clock)
	3	continuous leading edge clock
0	0	Continuous parallel output
	1	Handshaked parallel output

3.1.11. CSR10

enable_leading	enable leading edge measurement.
enable_trailing	enable trailing edge measurement
enable_pair	enable pair (leading and trailing edge) measurement. If this bit is set, enable_leading and enable_trailing bits are masked.
enable_rejected	enable force generation of rejected hit. If the channel buffer is full when a new hit arrives it will be ignored. If this bit is set, the information of the rejected hit is transferred as soon as the channel buffer is available even there is no hit. (see section 2.3)
enable_trailer	enable trailer in read-out data.
enable_header	enable header in read-out data.
enable_serial	enable serial read-out (otherwise parallel read-out)
enable_relative	enable read-out of relative time to trigger time tag
enable_mask	enable search and read-out of mask flags
enable_match	enable trigger matching
enable_l1occup_readout	enable read-out of l1 occupancy for each event (use for debugging)
enable_auto_reject	enable of automatic rejection of hits from the L1 buffer (see section 2.7)

3.1.12. CSR11

enable_setcount_bcrst	enable coarse, bunch and event counters to be reset on bunch count reset
enable_mreset_evrst	enable master reset on event reset
enable_resetcb_sepa	enable channel buffer reset when a separator is inserted
enable_mreset_code	enable master reset from global reset of the encoded_control
enable_l1ovr_detect	enable L1 buffer overflow detection. If this bit is not set, write pointer of the L1 buffer will pass read_pointer if the L1 buffer become full. Normally this bit should be set.
enable_errmark_ovr	enable error marking events when L1 buffer overflows (effective only in enable_matching = 0).
enable_errmark_rejected	enable error mark event if rejected hit seen (effective only in enable_matching =0)
inclk_boost	double the internal clock frequency. (Only for test purpose).
enable_errmark	enable error mark word if hard error exists.

enable_trfull_reject	enable event data rejection if trigger FIFO nearly full (see section 2.9)
enable_l1full_reject	enable event data rejection if L1 buffer nearly full (see section 2.9)
enable_rofull_reject	enable event data rejection when read-out FIFO full (see section 2.9)

3.1.13. CSR12

enable_error[8:0]	ERROR signal is asserted if corresponding enable_error bit is set.
enable_sepa_evrst	enable generate separator on event reset
enable_sepa_bcrst	enable generate separator on bunch count reset
enable_sepa_readout	enable read-out of internal separators (debugging)

3.1.14. CSR13, 14

enable_channel[23:0]	enable individual channel inputs

3.1.15. CSR15

general_out[11:0]	12 bit general purpose outputs. The value written to this register is available from the DIO23-DIO12 pins if ASDMOD pin is set to high level.

(JTAG access to this register is independent from CSR0-14 JTAG path.)

3.2. Status registers

Table. 5. Bit assignment of the status registers (all these registers are read only).

	11	10	9	8	7	6	5	4	3	2	1	0
CSR16	rfifo_ empty (1) {11}	rfifo_ full (0) {10}	control_ parity (1) {9}	error_ flags (0) {8:0}								
CSR17	l1_ empty (1) {23}	l1_ nearly_ full (0) {22}	l1_ over_ recover (1) {21}	l1_ over flow (0) {20}	l1_ write_address (0) {19:12}							
CSR18	tfifo_ empty (1) {35}	tfifo_ nearly_ full (0) {34}	tfifo_ full (0) {33}	running (0) {32}	l1_ read_address (0) {31:24}							
CSR19	coarse_ counter [0] (0) {47}	tfifo_ occupancy (0) {46:44}			l1_ start_address (0) {43:36}							
CSR20	coarse_counter[12:1] (0) {59:48}											
CSR21	general_ in[3:0] (DIO27~24) {71:68}				0 (0) {67}	0 (0) {66}	rfifo_ occupancy[5:0} (0) {65:60}					

() -- initial value at reset. {} --- JTAG bit No.

3.2.1. CSR16

error_flags[8:0] : status of error monitoring. (see section 2.11)

bit #
0: coarse error (parity error in the coarse counter)
1: channel select error (more than 1 channel are selected)
2: l1 buffer parity error
3: trigger FIFO parity error
4: trigger matching error (state error)
5: readout FIFO parity error
6: readout state error
7: control parity error
8: JTAG instruction parity error

control_parity	parity of control data (see section 2.11)
rfifo_full[0]	read-out FIFO full
rfifo_empty[0]	read-out FIFO empty

3.2.2. CSR17

l1_write_address[7:0] L1 buffer write address.

l1_overflow	L1 buffer overflow bit
l1_over_recover	L1 buffer overflow recover bit
l1_nearly_full	L1 buffer nearly full bit
l1_empty	L1 buffer empty bit

3.2.3. CSR18

l1_read_address[7:0]	L1 buffer read address
running	status of START signal
tfifo_full	trigger FIFO full bit
tfifo_nearly_full	trigger FIFO nearly full bit
trigger_fifo_empty	trigger FIFO empty bit

3.2.4. CSR19

l1_start_address	L1 buffer start address
tfiffo_occupancy	trigger FIFO occupancy (number of word in trigger FIFO)
coarse_counter	Coarse counter bit 0

3.2.5. CSR20

coarse_counter[12:1]	Coarse counter bit [12:1]

3.2.6. CSR21

rfifo_occupancy[5:0]	Read-out FIFO occupancy (number of word in read-out FIFO)
general_in[3:0]	4 bit general purpose inputs. Input level of the DIO27-DIO24 can be read through this register if ASDMOD pin is set to high level. general_in[3] is shared with ASDIN signal.

3.3. JTAG Port.

JTAG (Joint Test Action Group, IEEE 1149.1 standard) boundary scan is supported to be capable of performing extensive testing of TDC modules while located in the system. Testing the functionality of the chip itself is also supported by the JTAG INTEST and BIST capability. In addition special JTAG registers have been included in the data path of the chip to be capable of performing effective testing of registers and embedded memory structures. Furthermore, it is also possible to access the CSR registers from the JTAG port.

Parity error in JTAG instruction will set error_flag[8] in CSR16. This error can be recognized by reading the register, enable error word, or detect through ERRORB signal. The instruction is stored in instruction register and executed regardless of the presence of the instruction parity error.

JTAG TAP (Test Access Port) state diagram is shown in Fig. 21.

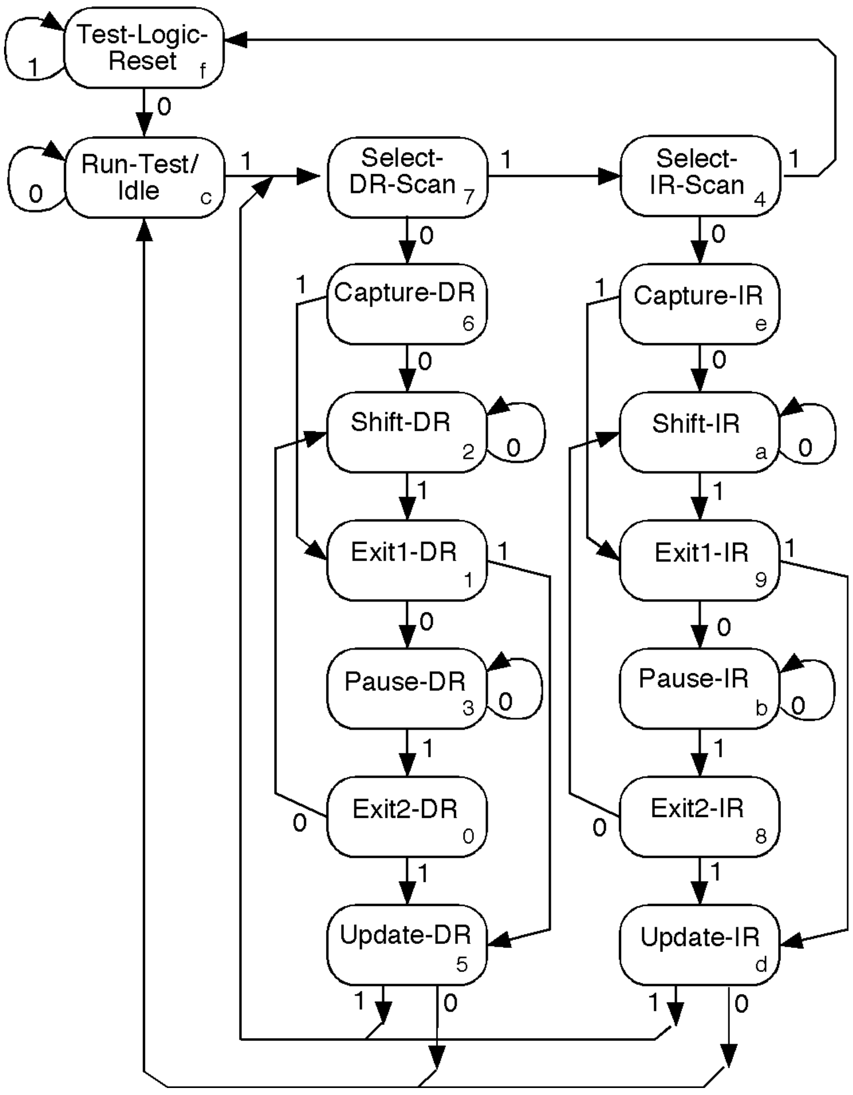


Fig. 21 JTAG TAP controller state diagram. Numbers in each states shown are state variable of the TAP controller.

3.3.1. JTAG controller and instructions

The JTAG instruction register is 4 bits long plus a parity bit:

[3:0]	ins[3:0] JTAG instruction
[4]	parity[0] parity of JTAG instruction

Table. 6. JTAG Instructions.

Parity	Instruction:	Name Description
0	0000:	EXTEST. Boundary scan for test of inter-chip connections on module.
1	0001:	IDCODE. Scan out of chip identification code.
1	0010:	SAMPLE. Sample of all chip pins via boundary scan registers.
0	0011:	INTEST. Using boundary scan registers to test chip itself.
-	0100 - 0111:	(not used)
1	1000:	CONTROL. Read/Write of control data.
0	1001:	ASD control.
0	1010:	STATUS. Read out of status register information.
1	1011:	CORETEST. Read/Write internal registers for debugging.
0	1100:	BIST. Built In Self Test for Memories.
1	1101:	General purpose output port
0	1110:	(not used)
0	1111	BYPASS. Select BYPASS register.

3.3.2. Boundary scan registers.

All signal pins of the TDC are passed through JTAG boundary scan registers. All JTAG test modes related to the boundary scan registers are supported (EXTEST, INTEST, SAMPLE).

Table 2 AMT-3 Boundary Scan Registers.

BSR #	Pin name	Input/Output	Comment
0	ASDMOD	I	
1	RESETB	I	inverted
2	ENCCONTP, ENCCONTM	I	LVDS input
26:3	HITP[23:0], HITM[23:0]	I	LVDS input
27	BUNCHRSTB	I	inverted
28	CLKOEN	I	
29	EVENTRSTB	I	inverted
30	CLKO	O	
31	TRIGGER	I	
32	CLKP, CLKM	I	LVDS input
33	DIOEN	O	
34	do[0]	O	see Fig. 22
35	di[0]	I	see Fig. 22
:	:	:	
56	do[11]	O	see Fig. 22
57	di[11]	I	see Fig. 22
69:58	do12_23 [11:0]	O	
70	dio24_27[0]	O	
71	dio24_27[0]	I	
:	:	:	
76	dio24_27[3]	O	
77	dio24_27[3]	I	
81:78	do28_31[3:0]	O	
82	GETDATA	I	
83	START	I	
84	DSPACE	I	
85	WR	I	
86	CS	I	
91:87	RA[0:4]	I	
92	CLKOUT	O	
93	DREADY	O	
94	ERROR	O	
95	SERIOUTP, SERIOUTM	O	LVDS output
96	STROBEP, STROBEM	O	LVDS output

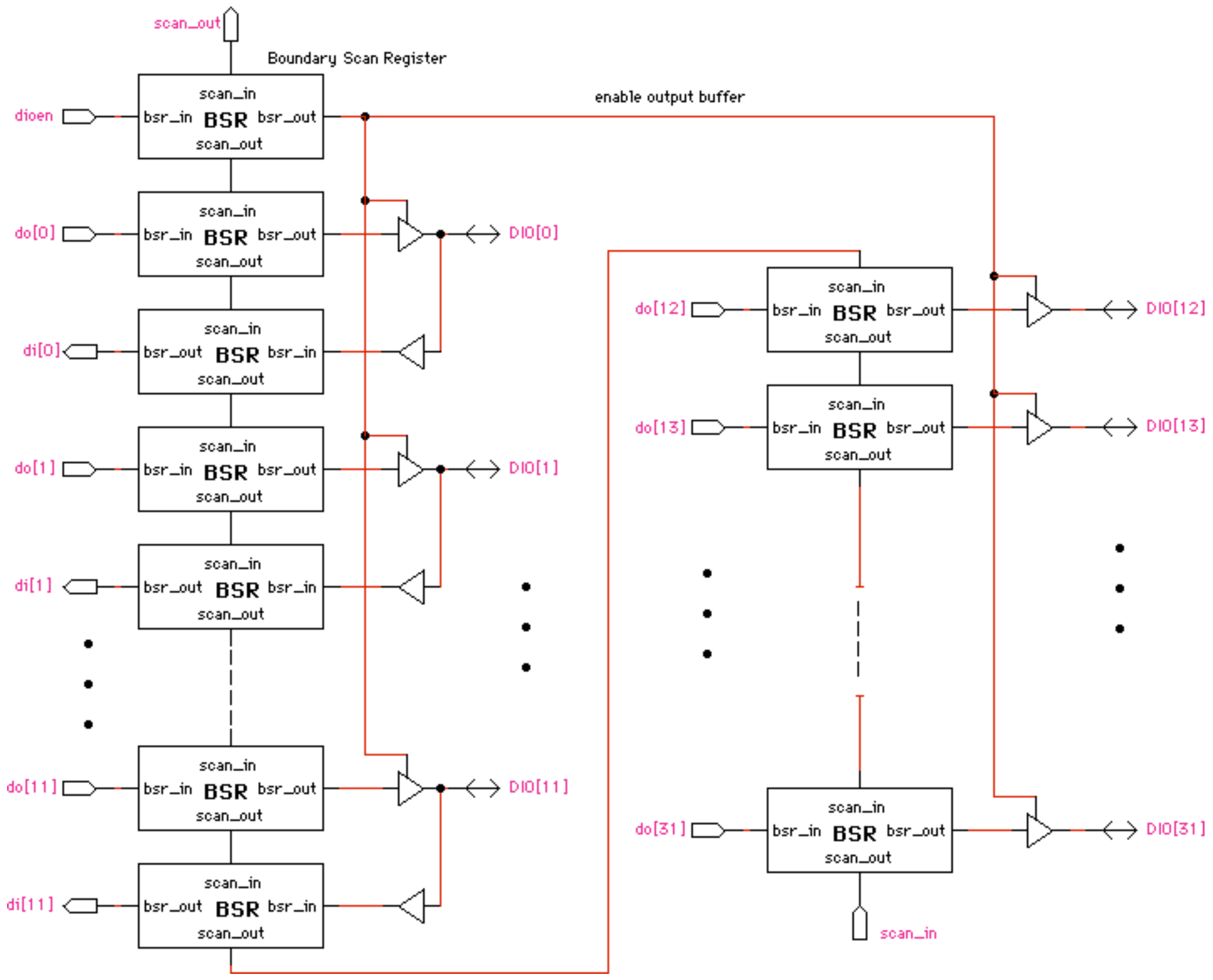


Fig. 22. JTAG boundary scan circuit for the DIO port.

3.3.3. ID code register

A 32 bit chip identification code can be shifted out when selecting the ID shift chain.

[0]	Start bit = 1,
[11:1]	manufacturer code =0000 0011 000 (Toshiba Co.),
[27:12]	TDC part code = 1000 1011 1000 0101
[31:28]	Version code = 0011

Thus the total ID code in hex formats is '38B85031'. (AMT-1 ID = 18B83131, AMT-2 ID = 18B85031)

3.3.4. Control registers

The JTAG control scan path is used to set CSR0-14 registers that should not be changed while the TDC is actively running. See section 3.1 for register details.

3.3.5. Status registers

The JTAG status scan path is used to get access to the status of the TDC while it is running (or after a run). See section 3.1.15 for register details.

3.3.6. Core registers

The JTAG core register scan path is used to perform extended testing of the TDC chip. This scan path gives direct access to the interface between the channel buffers and first level buffer logic. It is used in connection with the test_mode bits (CSR0[7]) and is only intended for verification and production tests of the TDC chip.

If the test_invert (CSR0[6]) =1 and hit_load = 1 in the test mode, contents of the hit_data, hit_channel and hit_select_error are inverted in every cycle.

Table. 7. Bit assignment of the core registers.

JTAG bit #	name	R/W	comments
[10:0]	matching_state[10:0]	R	monitoring of trigger matching state
[37:11]	trigger_data[26:0]	R	monitoring of active trigger data (*)
[38]	trigger_ready	R	monitoring of active trigger
[74:39]	l1_data[35:0]	R	monitoring of hit data to trigger matching (*)
[75]	l1_empty	R	monitoring of l1_empty flag
[76]	l1_data_ready	R	monitoring of l1_data_redy flag
[168:77]	hit_data[91:0]	R/W	monitoring/generation of hit_data (*)
[173:169]	hit_channel[4:0]	R/W	monitoring/generation of hit_channel
[174]	hit_select_error	R/W	monitoring/generation of hit_select_errort
[175]	hit_load	R/W	monitoring/generation of hit_load

(*) see Appendix A for internal format.

matching_state[10:0] :

state_waiting	00000000001
state_write_event_header	00000000010
state_write_occupancy	00000000100
state_active	00000001000
state_write_mask_flags	00000010000
state_write_error	00000100000
state_write_event_trailer	00001000000
state_generating_lost_event_header	00010000000
state_generating_lost_event_trailer	00100000000
state_waiting_for_separator	01000000000
state_write_separator	10000000000

3.3.7. ASD Control

To control ASD chip (developed by Harvard Univ.), ASD control signals are implemented in the AMT. Simulated timing diagram is shown in Fig. 24, and its simulation model is shown in Fig. 23. Unfortunately both chip has TDO register, so 2TDO registers are connected serially at the end of the scan chain. Thus the scan chain looks 1 additional scan register.

If the TDO register in the ASD is removed, the timing diagram will be Fig. 25.

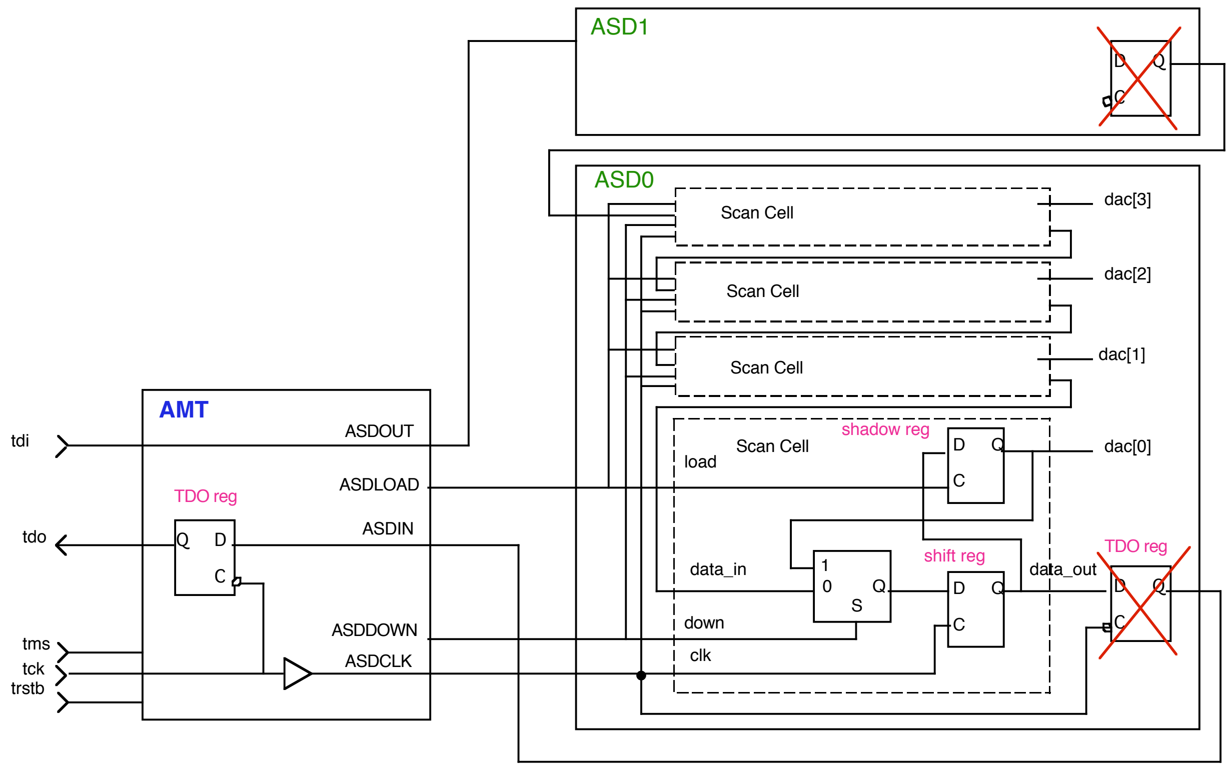


Fig. 23. AMT - ASD control simulation model. There are 2 TDO registers at the end of scan chain. Therefore there looks 1 additional scan register exist in the chain.

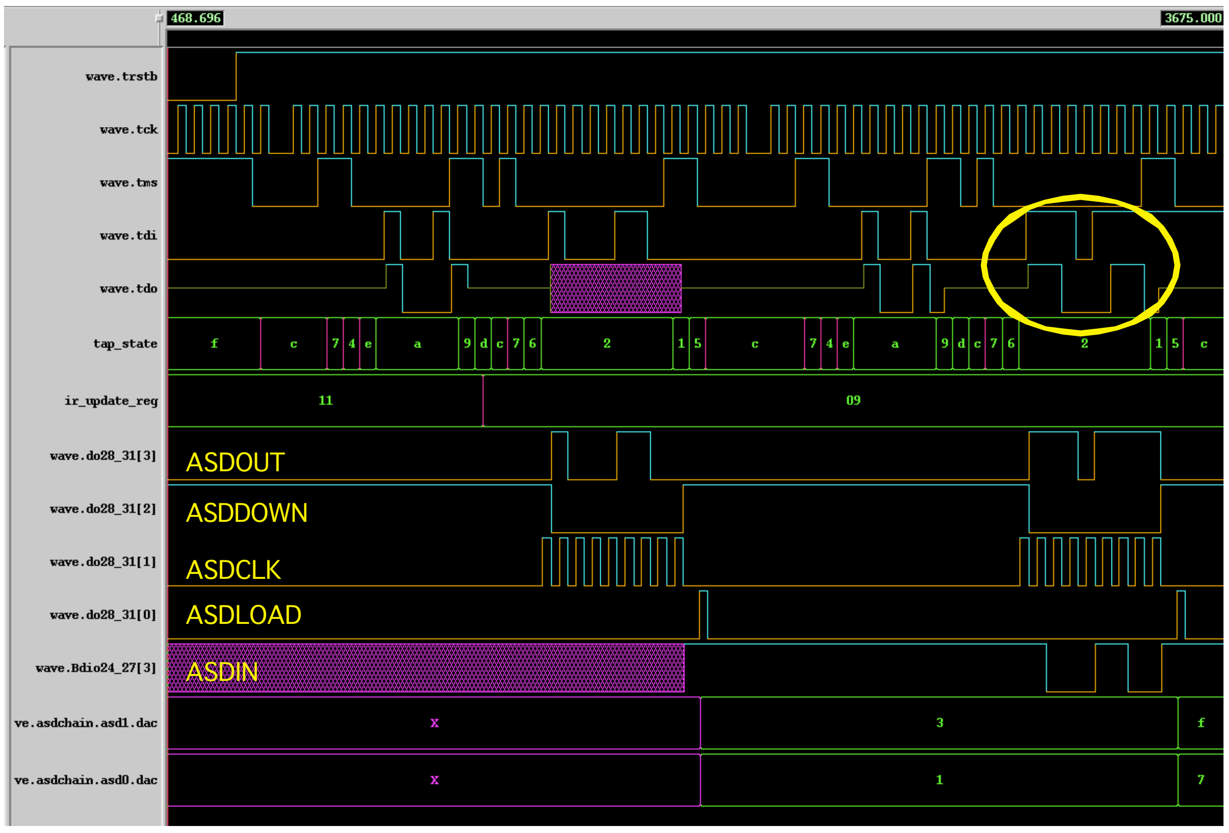


Fig. 24. Simulated timing diagram of the ASD control signals.

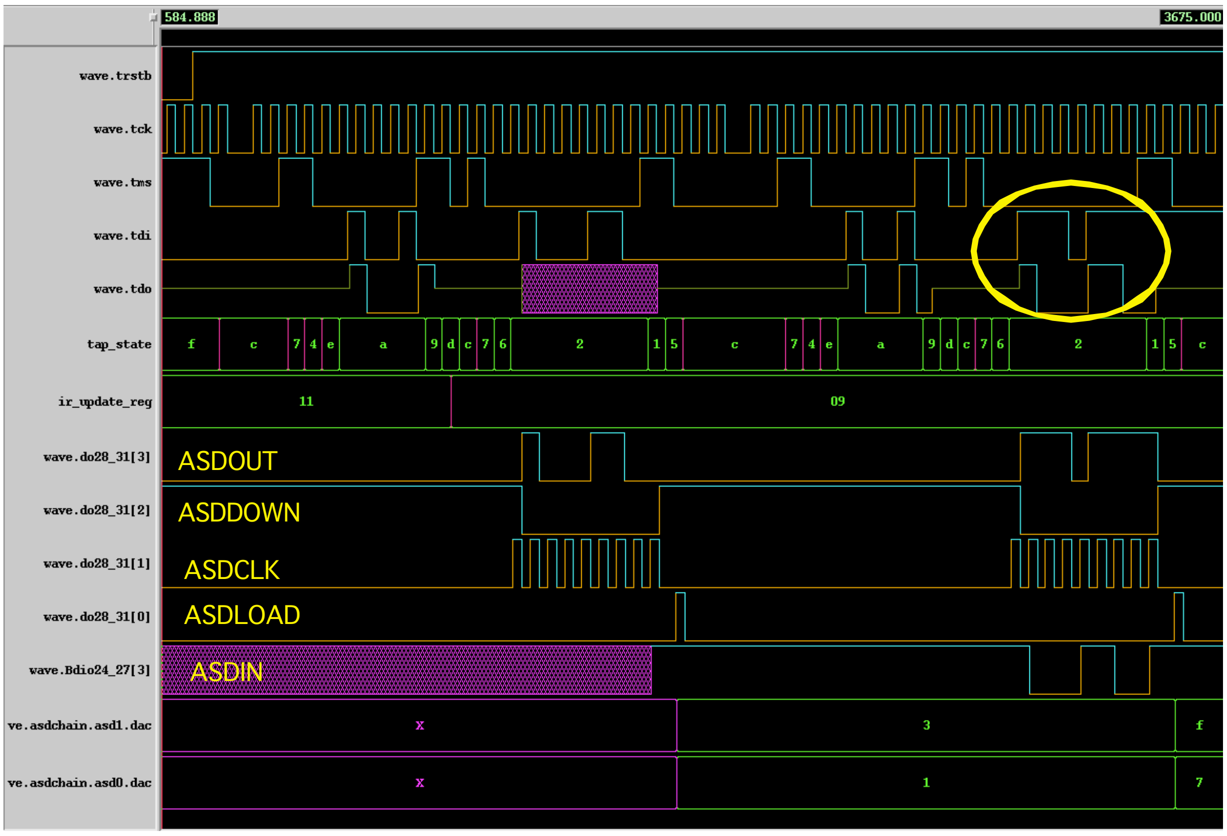


Fig. 25. Simulated timing diagram of the ASD control signal when the TDO register in the ASD is removed.

4. Time alignment between hits, clock and trigger matching

The TDC contains many programmable setups which have effects on the performed time measurements and their trigger matching. The main time reference of the TDC is the clock and the bunch reset which defines the T0 time. The time alignment can basically be divided into three different areas as shown in the figure below.

A: Time relationship between the hits, clock and the bunch count reset generating the basic timing measurements of the TDC.

B: Time relationship between the performed time measurements and the trigger time tag.

C: Time relationship between the time measurements and the automatic reject.

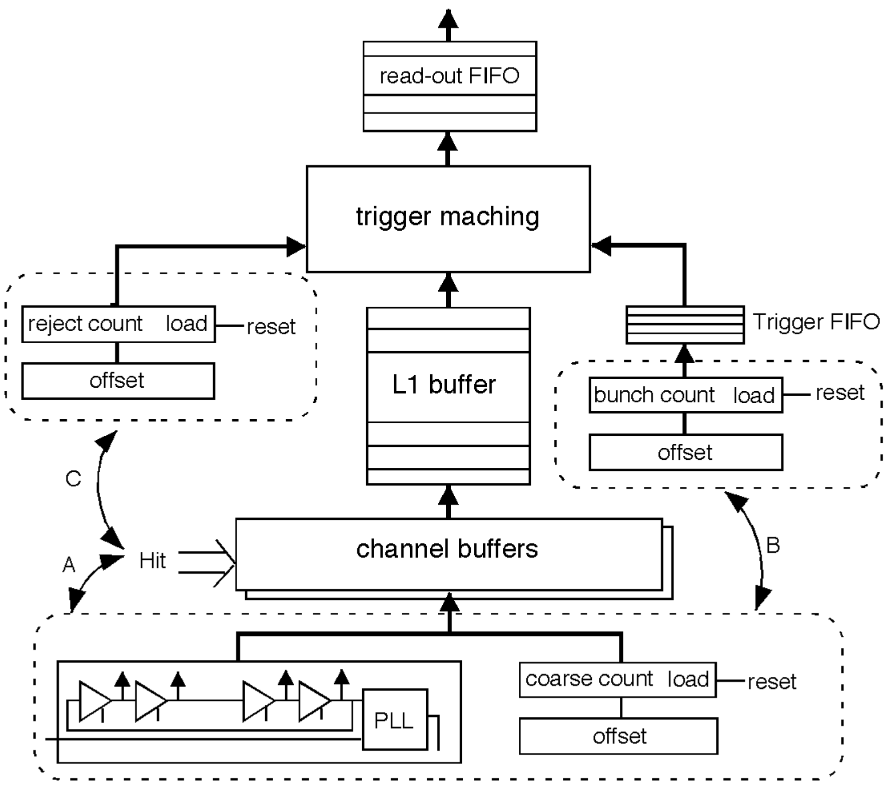


Fig. 12 :Time alignment between hits, trigger and automatic reject

A. Basic time measurement.

As previously stated the basic time reference of the TDC measurements is the rising edges of the clock. The bunch count reset defines the T0 reference time where the coarse time count is loaded with its offset value. The TDC also contains internal delay paths of the clock and its channel inputs which influences the actual time measurement obtained. These effects can be considered as a time shift in relation to the ideal measurement. The time shifts of individual channels may be slightly different but care has been taken to insure that the channel differences are below the bin size (~1 ns) of the TDC. The time shift of the measurements also have some variation from chip to chip and variations with supply voltage and temperature. These variations have also been kept below the bin size of the TDC by balancing the delay paths of the clock and the channels.

In Fig. 26, a hit signal is defined such that the time measurement equals zero (coarse_time_offset = 0). The delay from the rising edge of the clock where the bunch reset signal was asserted to the rising edge of the hit signal is for a typical chip **55ns** (two clock periods plus **5ns**).

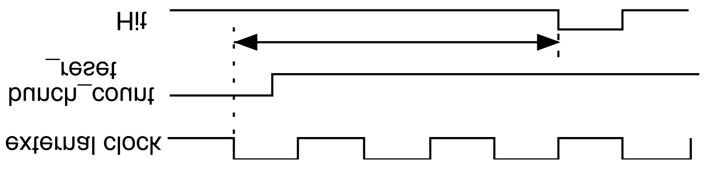


Fig. 26 Definition of reference time measurement.

B. Alignment between coarse time count and trigger time tag.

To perform an exact trigger matching the basic time measurement must be aligned with the positive trigger signal taking into account the actual latency of the trigger decision. The effective trigger latency in number of clock cycles equals the difference between the coarse time offset and the bunch count offset. The exact relation ship is: latency = (coarse_time_offset - bunch_count_offset) modulus (2^{12}).

A simple example is given to illustrate this: A coarse_time_offset of 100 Hex (decimal 256) and a bunch_count_offset of 000 Hex gives an effective trigger latency of 100 Hex (decimal 256). Normally it is preferable to have a coarse time offset of zero and in this case the trigger count offset must be chosen to (000 Hex - 100 Hex) modulus (2^{12}) = F00 Hex.

C. Alignment between trigger time tag and reject count offset.

The workings of the reject counter is very similar to the trigger time tag counter. The difference between the course time counter offset and the reject counter offset is used to detect when an event has become older than a specified reject limit. The rejection limit expression is (coarse_time_offset - reject_count_offset) modulus (2^{12}). The rejection limit should be set equal to the trigger latency plus a small safety margin. In case the extraction of masking hit flags are required the reject limit should be set larger than the trigger latency + masking window + safety margin.

For a trigger latency of 100 Hex (same as in example in the section above) a reject limit of 108 Hex can be considered a good choice (no mask detection). This transforms into a reject count offset of EF8 Hex when a coarse time offset of zero is used.

4.1. Example of Offset Setting

In the LHC experiments there are 3564 clock cycles (88.924 usec/24.9501 ns) in a beam revolution. Thus the count_roll_over (csr8) should be set 3563 ('deb' in hex).

For the trigger latency of 3 usec (120 clock), bunch_count_offset should be set to 3564 - 120 = 3444. This is summarized in Table. 8.

Table. 8 An example of register value setting.

Assumptions:
number of clock cycles per beam revolution (CC) = 3564 cycles
trigger latency (TL) = 100 cycles (2.5 μsec)
maximum drift time (DT) = 32 cycles (800 ns)

register	name	contents	typical value
csr0			$000
csr1	mask_window	>= DT	32 ($020)
csr2	search_window	>= match_window + 8	39 ($027)
csr3	match_window	>= DT -1	31 ($01f)
csr4	reject_count_offset	<= CC -TL -8 -mask_window(*1)	3424 ($D60)
csr5	event_count_offset		0($000)
csr6	bunch_count_offset	= CC - TL	3464 ($D75)
csr7	coarse_time_offset		0($000)
csr8	count_roll_over (*2)	= CC -1	3563 ($DEB)
csr9		strobe=2	$C00 (*3)
csr10		auto_reject, match, serial, header, trailer, leading	$A71 (*3)
csr11		enable...	$e11 (*3)
csr12		enable_sepa..., enable_error	$1FF
csr13	enable_channel[11:0]		$FFF
csr14	enable_channel[23:12]		$FFF

(*1) if enable_mask = 1.

(*2) At present design, count_roll_over must be greater than ($800+search_window). If the bunch count reset signal is applied in each cycle, you can assume larger value for CC, such as CC = 4096.

(*3) These values are largely depend on your measurement conditions.

Actual window size of 'match_windows' and 'search_windows' are "setting value +1". For example setting value of '10' means actual window size of '11'.
As for the 'mask_window', actual window size and the setting value are exactly same.

'xxx_offset' indicates relative time between each counters. These offset is loaded into respective counters when bunch count reset is asserted.
All counters are roll over at value 'count_roll_over'. Thus larger value than 'count_roll_over' has no meaning for the window size and the offsets.

5. Appendix A : Internal Data Format

Channel Buffer

[first data]

45	44	43	42 41 40 ... 32 31 30	29	28 27 26 ... 18 17 16	15 14 13 ... 3 2 1 0
rejected	edge	coarse2 parity	coarse2	coarse1 parity	coarse1	vernier

[second data]

91	90	89	88 87 86 ... 78 77 76	75	74 73 72 ... 60463 62	61 60 59 ... 44 45 46
rejected	edge	coarse2 parity	coarse2	coarse1 parity	coarse1	vernier

if enable_pair = 0 : edge = 0 (trailing edge), edge = 1 (leading edge),
if enable_pair = 1 : edge = 0 (trailing edge found), edge = 1 (trailing edge not found)
rejected : data is rejected since the channel buffer is full.

Level 1 Buffer

35	34	33	32	31 . . . 0
parity	make_ separator_ stored	buffer_ overflow	overflow	see below

overflow : level 1 buffer overflow start (buffer_size >= 253).
buffer_overflow : set to '1' when overflow occured and kept '1' until overflow recover |(buffer_size < 251).
make_separator_stored : separator flag (data[31:0] = 0)
parity : parity of data[34:0]

[enable_pair = 0]

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
hit_error	reject_out	channel	0	edge	edge time

[enable_pair =1, normal data]

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
hit_error	reject_out	channel	width	leading edge time

[enable_pair =1, over_flow]

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
hit_error	reject_out	channel	1 1 1 1 1 1 1 1	leading edge time

[enable_pair =1, under_flow]

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
hit_error	reject_out	channel	0 0 0 0 0 0 0 0	leading edge time

reject_out : enable_pair ? (reject_first | reject_second) : reject_first
hit_error : select_error | coarse_error

Readout FIFO

Event Header

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
parity	1 0 1 0	trigger_data[23:0]	
		event id	bunch id

Lost Event Header

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
parity	1 0 1 0	lost_trigger[11:0] (lost event id)	trigger_data[11:0]

Event Trailer

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
parity	1 1 0 0	trigger_data[23:0]	
		event id	bunch id

Lost Event Trailer

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
parity	1 1 0 0	lost_trigger[11:0] (lost event id)	0

Error

28	27 26 25 24	23 . . . 14	13 . . . 9	8 . . . 0
parity	0 1 1 0	0	error_data	0

error_data[0] : enable_matching & l1 buffer overflow
error_data[1] : enable_matching & trigger_lost
error_data[2] : enable_matching & full_rejected
(= enable_rofull_reject & readout_fifo_full &
(enable_l1full_reject & nearly_full |
enable_trfull_reject & trigger_fifo_nearly_full |
~enable_l1full_reject & ~enable_trfull_reject)
error_data[3] : enable_matching & hit_error
error_data[4] : ~enable_matching & reject_out

Mask Flags

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
parity	0 0 1 0	mask_flags[23:0]

Single Measurement (enable_relative =0)

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
parity	0 0 1 1	0	T	E	l1_data[16:0]

E : Hit Error = l1_data[31] | l1_data[30]

T : Edge Type = l1_data[17]

Single Measurement (enable_relative =1)

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
parity	0 0 1 1	l1_data[29:25]	T	E	coarse_relative[11:0]	l1_data[4:0]

Combined Measurement (enable_relative=0)

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
parity	0 1 0 0	l1_data[29:17]	l1_data[10:0]

Combined Measurement (enable_relative=1)

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
parity	0 1 0 0	l1_data[29:17]	corase_ relative[5:0]	l1_data[4:0]

Separator

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
parity	0 1 1 1	0	trigger_data[11:0]

Buffer Occupancy

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
parity	0 1 1 1	0 0 0 1	0	R	l1_occupancy[7:0]

R : Readout FIFO full

Trigger FIFO

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
parity	trigger lost	separator	event id	bunch id

6. Appendix B : Buffer Overflow Controls (May 7, 2002)

Buffer control mechanism of the AMT is somewhat complicated. Although the detailed explanations are available in reference [5] and this manual, brief summary is presented here for your understanding. All description here assumes 'enable_match' bit is set.

Fig. 27 shows buffer structure of the AMT chip. Overflow control is done for each buffer, and described below. In real situations, all these controls has strong relations.

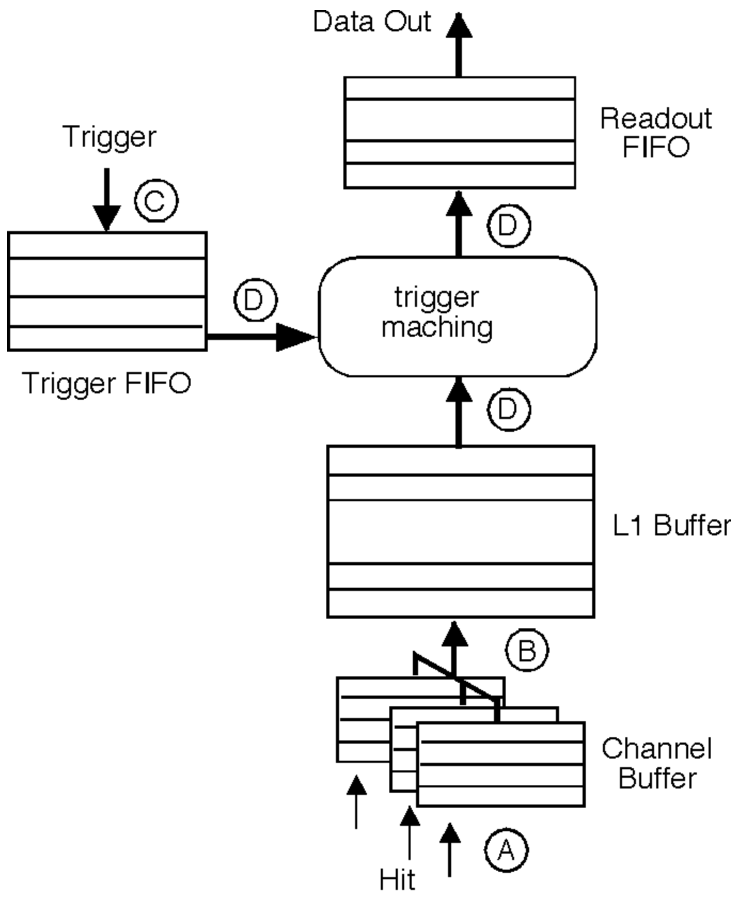


Fig. 27 Buffer controls.

A. Channel buffer overflow

The depth of the channel buffer is 4. If the channel buffer is full when a new hit arrives, the new hit will be rejected(discarded). The occurrence of the rejection is reported by 'E' flag of a data word if 'enable_rejected' bit is set. Otherwise user can not detect the occurence of the rejection.

The information of the rejected hit is transferred as soon as the channel buffer is available. Therfore the time of the data with 'E' flag is the time when the buffer is available and not actual data. Therefore the data should be discarded after error reporting.

In combined measurement (enable_pair=1), there is no 'E' flag in the data word, but the rejected hit can appear in the data if 'enable_rejected' bit is set. This may cause some confusion, so in this mode 'enable_rejected' bit should not be set.

B. L1 buffer overflow

There are 256 words depth in the L1 buffer. If 'enable_l1ovr_detect' bit is not set, there is no pointer control, so the L1 write pointer may pass the L1 read pointer. This bit should be set in normal use.

If the L1 buffer becomes full (≥ 253words), L1 buffer overflow bit is set for next data and further data will be discarded. When the data size becomes less than 251 words, the situation will recover from overflow to normal state.

The overflow event will have an error word with 'L1 buffer overflow' (bit 9) flag if the overflow occur in corresponding matching/mask windows.

C. Trigger FIFO overflow

Trigger FIFO is 8 words depth. If the trigger FIFO runs full the trigger time tags of following events will be lost. The trigger interface keeps track of how many triggers have been lost so the event synchronization in the trigger matching and the DAQ system is never lost.

A full flag in the trigger FIFO together with the event number of the triggers are used to detect the loss of triggers. If a positive trigger is signaled when the trigger FIFO is full the trigger interface stores the fact that one (or several) triggers have been lost. As soon as the trigger FIFO is not any more full, the event number of the latest lost trigger is written to the FIFO together with a trigger lost flag and bunch id of that time.

For each event with a lost trigger time tag the trigger matching will generate an event with correct event id and an error word with 'Trigger FIFO overflow' (bit 10) flag. However, the bunch id of the header indicates the time when the overflow was recovered.

D. Trigger Matching and Readout FIFO overflow

The trigger matching circuit continuously compares data with reject time counter and if the data is older than the reject time the data will be removed from the L1 buffer.

The readout FIFO is 64 words deep. In addition to data and error words, the readout FIFO always stores header and trailer irrespective of 'enable_header' and 'enable_trailer' bits settings. The event header and trailer are removed from the readout FIFO at readout time if above bits are not set.

When the readout FIFO becomes full, there are two modes to handle data.

[back propagate mode (enable_rofull_reject =0)]

In this mode, trigger matching will be blocked when the readout FIFO becomes full. This will resume when a new space is available in the readout FIFO. When this occurs, the L1 buffer and the trigger FIFO will be forced to store more data and finally the measurement is stopped. If this situation lasts longer than 'count_roll_over / 2 ' period, the data integrity can not be guaranteed.

[data reject mode (enable_rofull_reject =1)]

In this mode, data in the L1 buffer will be rejected when the readout FIFO becomes full, so this prevent to stop the measurement. Instead, only header, error word with 'Readout FIFO overflow' (bit 11) flag, and trailer are written to the readout FIFO. This rejection occurs irrelevant to the data volume in the L1 buffer and Trigger FIFO if enable_l1full_reject=0 and enable_trfull_reject=0.

If enable_l1full_reject=1, the above rejection starts when the L1 buffer becomes nearly full at 191 words (= 255 - 64).

If enable_trfull_reject=1, the above rejection starts when trigger FIFO becomes nearly full at 4 triggers,

References

[1] AMT-0 manual. http://micdigital.web.cern.ch/micdigital/amt.htm

[2] AMT chips web page. http://atlas.kek.jp/tdc/

[3] Y. Arai and T. Ohsugi; "An Idea of Deadtimeless Readout System by Using Time Memory Cell", Proceedings of the Summer Study on the Physics of the Superconductiong Supercollider, Snowmass, 1986, p.455-457.

[4] Y. Arai; "Production Readiness Review, ATLAS MDT TDC Chip (AMT)", available from http://atlas.kek.jp/tdc/ PRR page, AMT PRR Document.

[5] Y. Arai and J. Christiansen, "Requirements and Specifications of the TDC for the ATLAS Precision Muon Tracker", ATLAS Internal note MUON-NO-179, 14 May 1997. Also available from http://atlas.kek.jp/tdc/Documents/TDCspec.pdf