AMT-0

ATLAS Muon TDC version 0

Version 1.0, July 1998

J. Christiansen
CERN/EP - MIC
Email: jorgen.christiansen@cern.ch

1. Introduction

The ATLAS Muon TDC version 0 (AMT-0) is the first prototype version of a Time to Digital Converter (TDC) for the Monitored Drift Tubes (MDT) of the ATLAS muon detector. It has been implemented to perform system tests of the complete chain of front-end electronics to be integrated into the detector (analog front-end plus TDC). The AMT-0 design is to a large extent based on the 32 channel general purpose TDC also designed in the Micro electronics group at CERN.

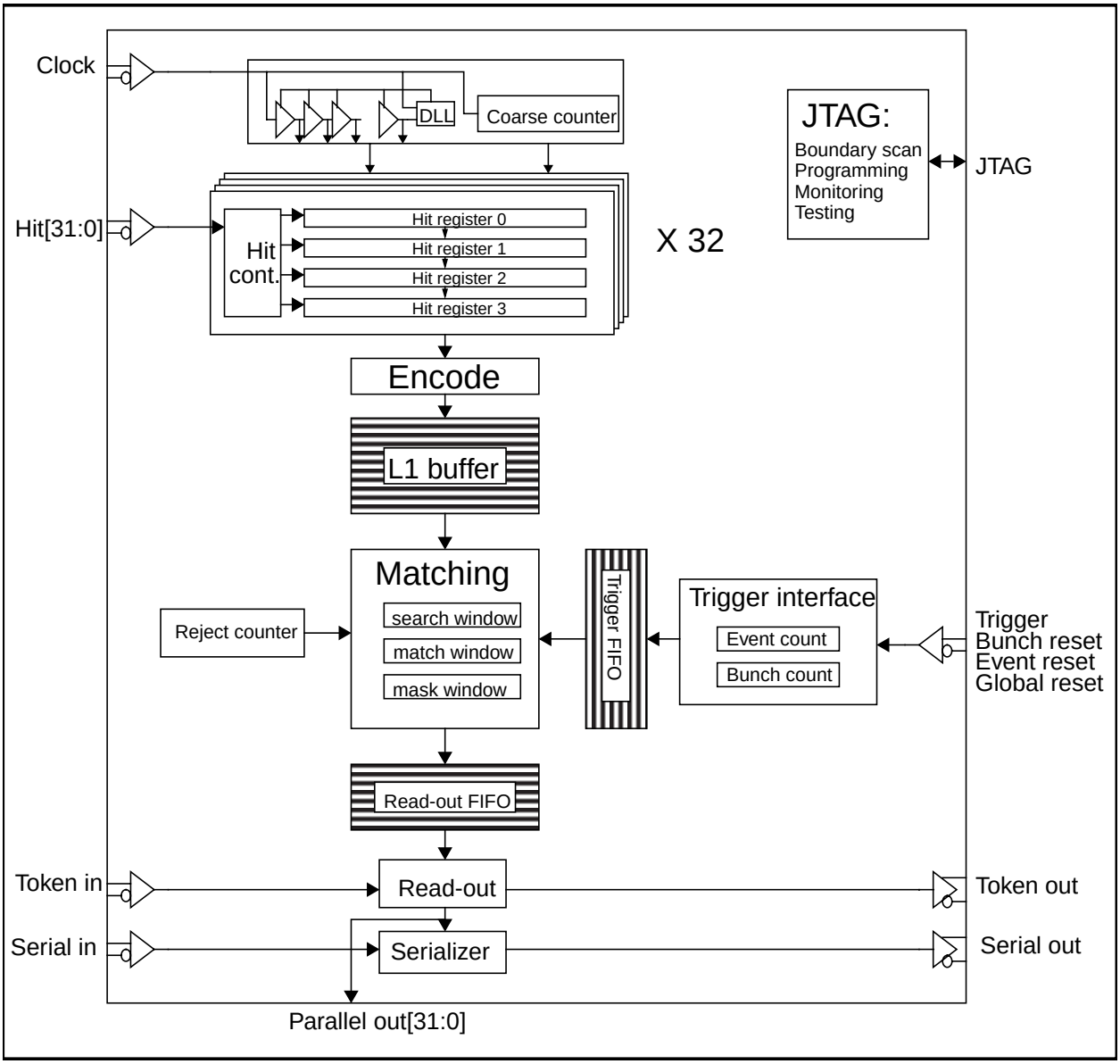


Fig. 1 :Block diagram of AMT0.

A time bin size of 0.78 ns at 40 MHz 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 (~10 mW/channel). The gate delay of CMOS devices normally have very large variations as function of process, voltage, and temperature. In this TDC a self calibrating scheme is implemented by using voltage controlled delay elements as a part of a Delay Locked Loop (DLL). The fine time measurement from the DLL is extended by a 12 bit coarse time counter.

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. Measurements stored in the level 1 buffer can be passed directly to a 32 words deep read-out FIFO, or 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. 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.

2. Delay Locked Loop.

The DLL, which generates the basic timing signals to obtain the required time resolution, consists of three major components: A/ Chain of 32 delay elements which delay can by adjusted by a control voltage. B/ Phase detector measuring the phase error between the clock and the delayed clock from the delay chain. C/ Charge pump and level shifter generating the control voltage to the delay elements based on the measured phase error from the phase detector.

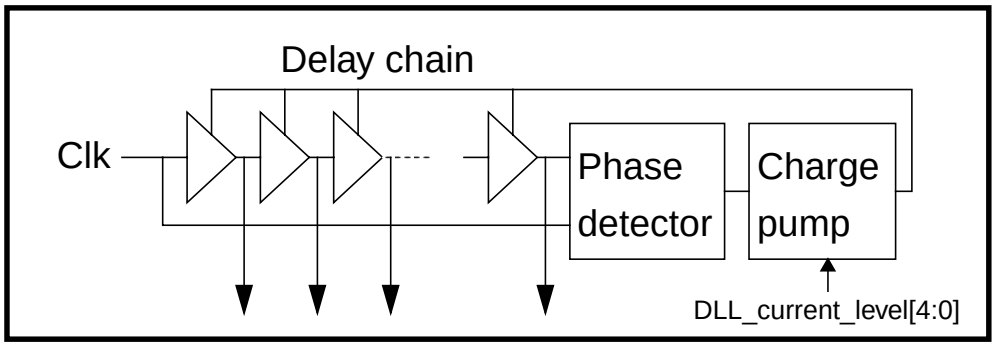


Fig. 2 :DLL with its main components.

After a reset of the DLL a certain time is required for it to reach lock. When correct locking has been obtained the lock status is set and the TDC is ready to perform time measurements. During normal operation the DLL is continuously monitored. If locking is lost (may happen if the clock has been removed for some time or a large clock frequency change has occurred) and a time measurement is performed a vernier error status is set. If this happens the DLL must be reinitialized to guarantee correct function.

The DLL is sensitive to jitter on its input clock. Any jitter on the clock will directly deteriorate the precision of the time measurements. If the jitter is large (> 0.5 ns) it will result in the vernier error status bit to be set.

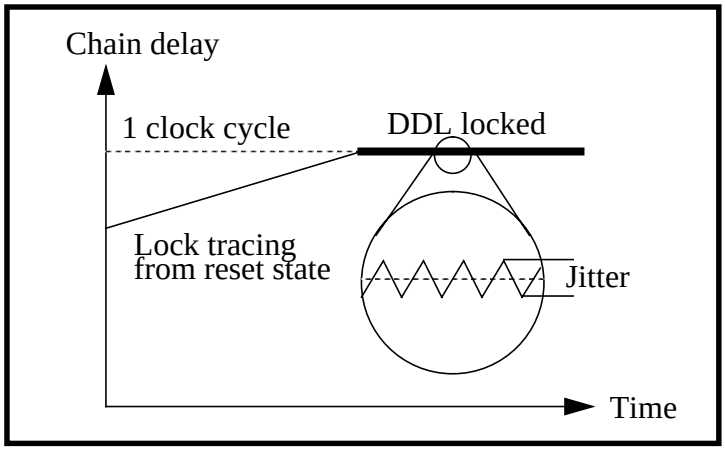


Fig. 3 :Closed loop behaviour of DLL from out of lock condition.

The dynamics of the control loop in the DLL can be set by controlling the current levels used in the charge pump circuit via the JTAG interface. Low current levels results in slow lock tracing but reduced jitter when locking has been obtained. Large current levels gives fast locking but increased jitter.

It has been found that the DLL is very sensitive to jitter on the clock during lock tracing. A clock jitter larger than 100 ps peak-peak will in some cases prevent the DLL to reach correct lock. In this case the DLL lock status is set as if correct locking has been obtained but the vernier error signal is set when the first hit is introduced. The sensitivity to jitter during locking can be significantly reduced by using the maximum current levels in the charge pump during lock tracing. After lock has been obtained the current levels can be reduced. See also bug list of AMT0.

3. Coarse Time Count.

The dynamic range of the fine time measurement, extracted from the state of the DLL, 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. Based on the fine time measurement from the DLL one of the two count values will be selected, such that a correct coarse time value is always obtained.

At reset the coarse time counter is loaded with a programmable coarse time offset. The coarse time counter of the TDC will in ATLAS be clocked by the bunch crossing signal thereby 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.

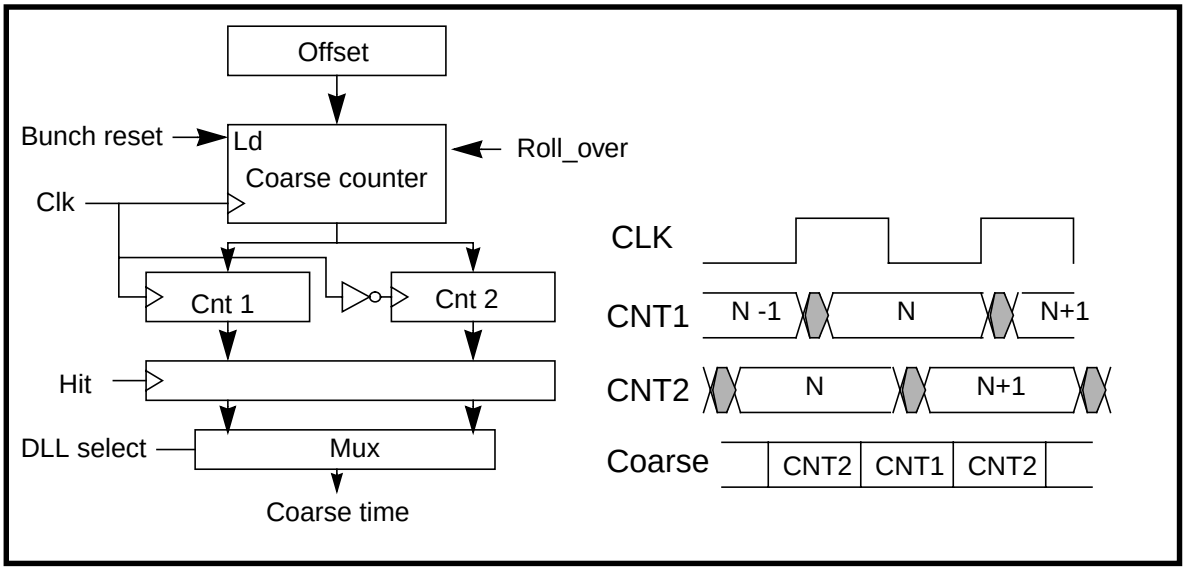


Fig. 4 :Phase shifted coarse time counters loaded at hit.

4. 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. If the channel buffer is full when a new hit arrives it will be ignored. The minimum time between two consecutive time measurements is 15 ns.

For the hits stored in the channel buffers to be written into the clock synchronous L1 buffer a synchronisation of the status signals from the channel buffers is performed. Double synchronisers 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.

5. Writing into event buffer.

When a hit has been detected on a channel the corresponding channel buffer is selected, the fine time measurement is encoded into binary form, the correct coarse time count value is selected and the complete time measurement is written into the L1 buffer together with a channel identifier. 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 31 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.

6. 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. If the L1 buffer runs full the latest written hit will be marked with a special full flag. When the buffer recovers from being full the first arriving hit will be marked with a full recover flag. These flags are used by the following trigger matching to identify events which may have lost hits because of the buffer being full.

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 such that all time measurements read out are referenced to the time (bunch crossing) when the event of interest occurred.

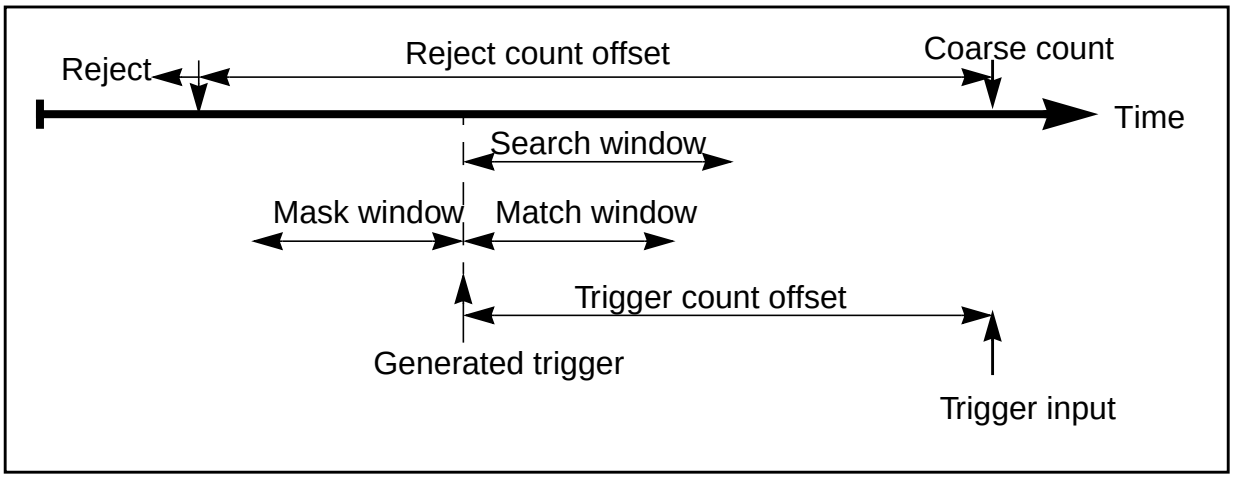


Fig. 5 :Trigger, trigger latency and trigger window related to hits on channels

A match between the trigger and a hit is detected within a programmable time window. 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 trigger matching window will be considered as matching the trigger. The trigger matching being based on the coarse count means that the “resolution” of the trigger matching is one clock cycle 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. 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. 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.

All data belonging to an event is written into the read-out fifo with a header and a trailer. The header contains an event id and a bunch id. The event trailer contains the same event id plus a word count.

The trigger matching function may also be completely disabled 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 + 32 = 288 measurements.

8. Trigger 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 initialises 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.

8.1. Encoded trigger and resets

Four basic signals are encoded using three clock periods. The simple coding scheme is restricted to only distribute one command in each period of three clock periods. A command is signalled with a start bit followed by two bits determining the command.

Trigger:	1 0 0
Bunch count reset:	1 1 0
Event count reset:	1 0 1
Global reset:	1 1 1

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.

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 enabled in the programming (described later).

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 enabled in the programming (described later).

From a bunch reset is given to the TDC until this is seen in the hit measurements them selves a latency of the order of 2 clock cycles is introduced by internal pipelining of the coarse time counter in the TDC. The definition of time 0 in relation to the bunch reset is described in more detail in the chapter: Time alignment between hits, clock and trigger matching.

8.4. Global reset.

A global reset clears all buffers in the TDC and initialises 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.

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 signalled the value of the trigger time tag counter (bunch id) is loaded into the trigger FIFO. The effective trigger latency using this scheme equals the difference between the coarse time count offset and the trigger time tag offset.

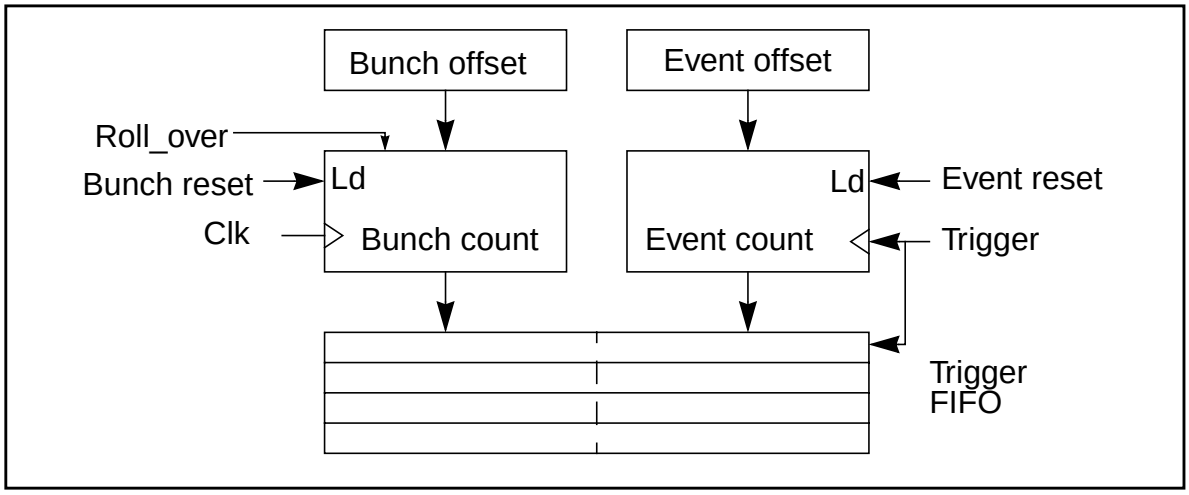


Fig. 6 :Generation of trigger data

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 synchronisation 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 signalling that the whole event has been lost.

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. 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.

9. Read-out FIFO.

The read-out FIFO is 32 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: 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 overflow.

Nearly full reject: In this mode the trigger matching will block until either the L1 buffer or the trigger fifo is nearly full. 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 synchronisation in the DAQ system. Any event which have lost data in this way will be marked with an error flag.

Reject: As soon as the read-out fifo is full, event data (not event headers and trailers) will be rejected. Any loss of data will be signalled with an error flag.

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. Up to 16 chips can be coupled together using a token passing scheme to perform local event building.

The read-out of an event is started by the master chip sending a group event header (if enabled). The master then sends the token to the first slave chip in the chain which then starts to send its data. The event data from each chip typically consists of a single chip 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 single chip event trailer (if enabled). The token is then passed on to the following TDCs in the chain until it finally arrives at the master. The master chip asserts is own event data and finally ends the whole event with a group event trailer (if enabled).

A slightly different read-out protocol is available when trigger matching is not enabled as data is not grouped in events. In this case each TDC having data can be programmed to wait for the token and then send one measurement and immediately give the token to the next chip even if it has more data to send. This ensures that all TDC’s gets equal access to send their data and prevents a TDC with a noisy channel to block all the other TDCs in the read-out chain. The cost of this is that the token is circulated continuously which implies some overhead.

10.1. Parallel read-out

Read out of parallel data from the TDC is performed via a clock synchronous bus. Several TDC’s may share one read-out bus controlled by the circulating token and an external controller. The read out of individual hits are controlled by a Data_ready / Get_data handshake and the event synchronisation is controlled by the circulating token. If the Get_data signal is constantly held active (independent of Data_ready) 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 Data_ready / Get_data handshake protocol to introduce wait cycles. The number of clock periods that the get_data signal is asserted is used my the master TDC to determine the word count for the event available in the global trailer.

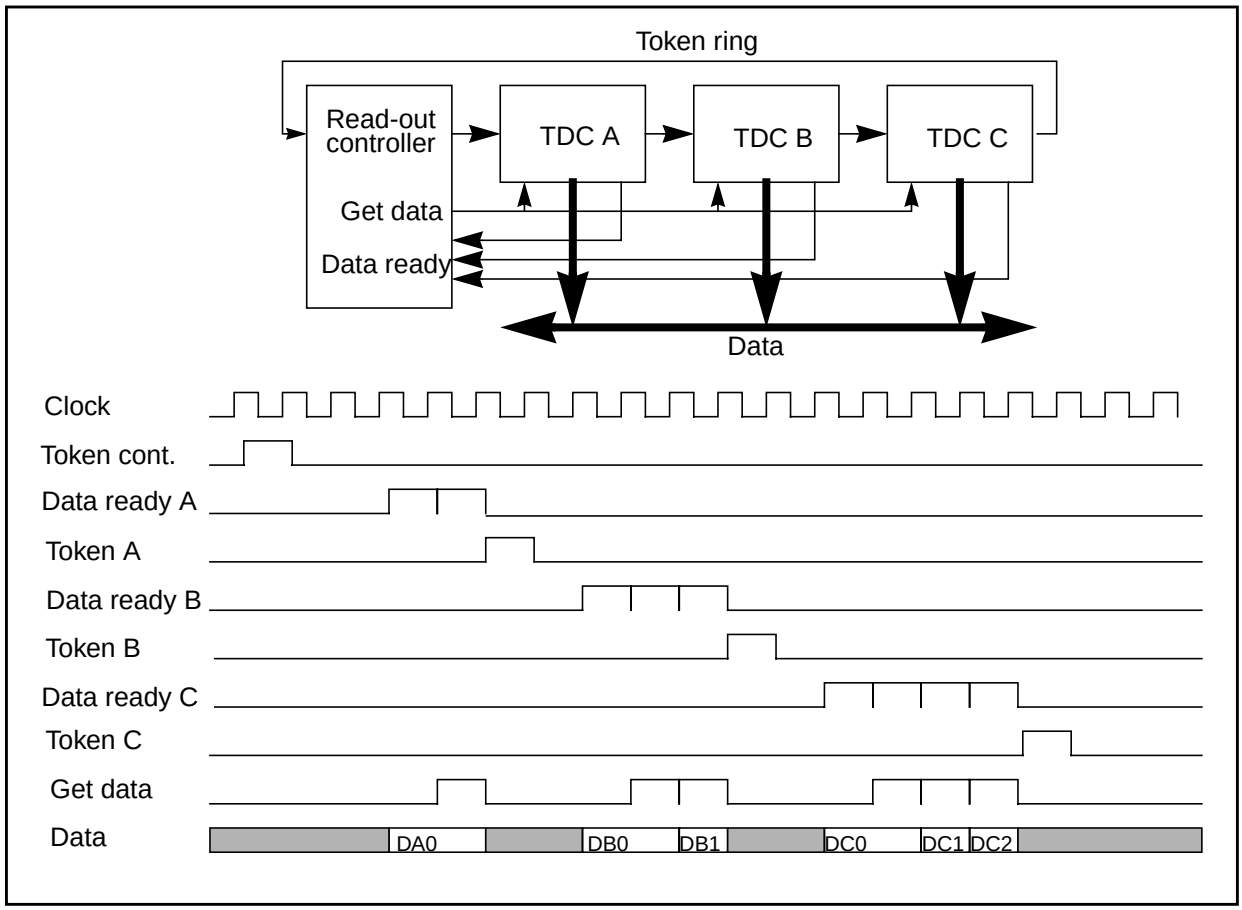


Fig. 7 :Token based parallel read-out with all TDCs configured as slaves controlled by an external controller.

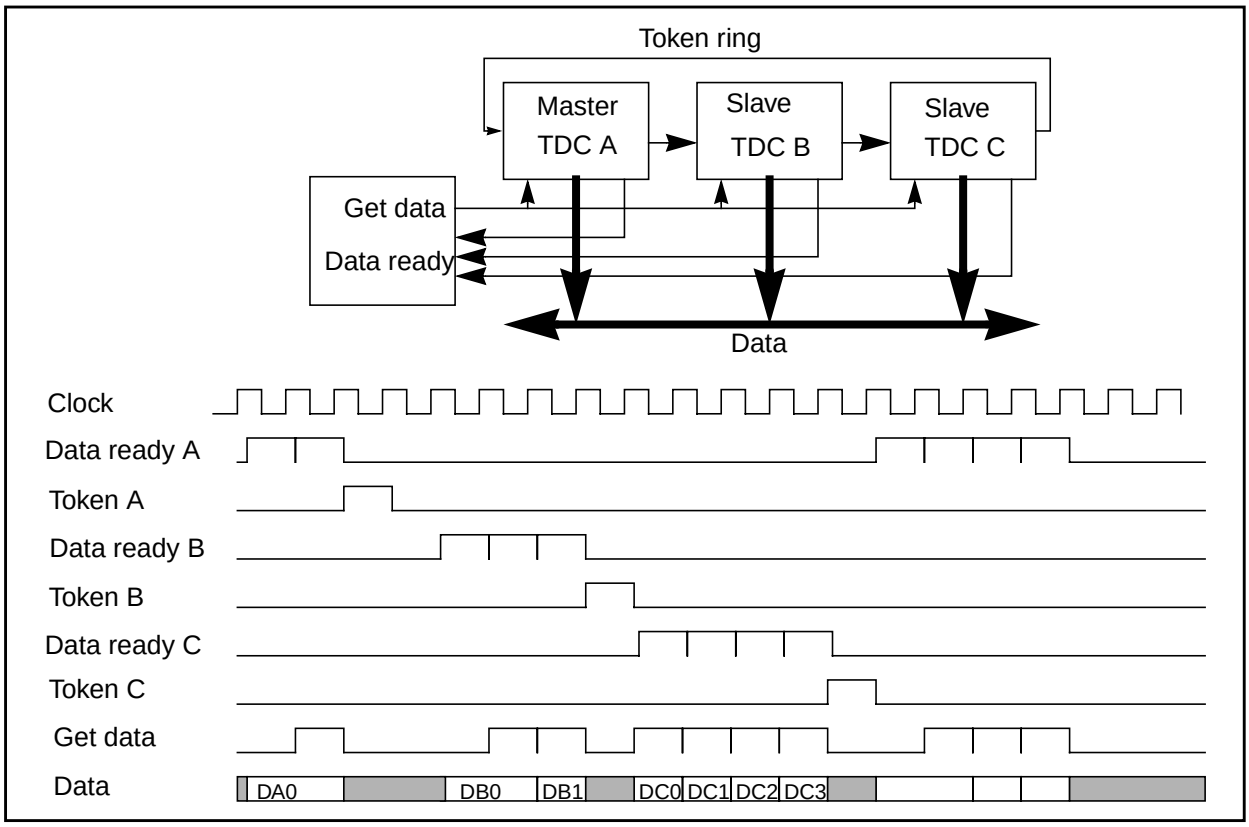


Fig. 8 :Token based parallel read-out with one master TDC and a simple read out controller.

10.2. Serial read-out

The accepted TDC data can be transmitted serially over twisted pairs using LVDS signals. Data is transmitted in words of 32 bits with a start bit set to one and followed by a parity bit. When no data is transmitted the serial line is kept at zero. The serialization speed is programmable from 80 (160) to 0.3215 Mbits/s.

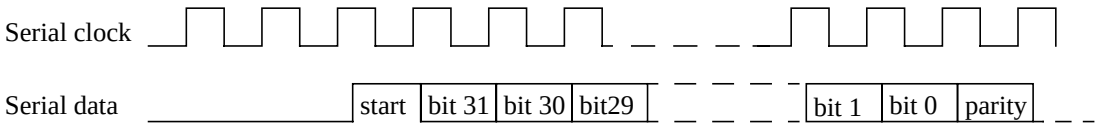


Fig. 9 :Serial frame format with start bit and parity bit

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

Clock: Direct serializing clock to strobe data on rising edge.

Edge: Edge strobe to strobe data on both rising and falling edge.

DS: DS strobe format as specified for transputer serial links. DS strobe only changes value when no change of serial data is observed.

None: No strobe signal.

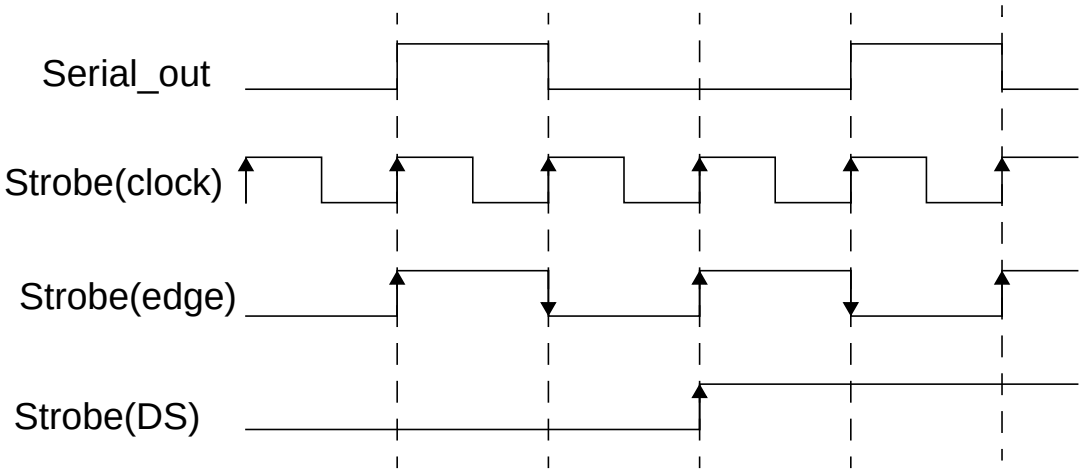


Fig. 10 :Different strobe types.

The serial data is connected directly from one slave TDC into the following TDC. When a TDC does not have the token the serial data is just re-timed with the serializing clock and directed to the output. The internal clocking of serial data prevents excessive delays to build up when passing through several slave TDC's. Only the master chip drives the serial data on the twisted pair cable going to the DAQ system.

10.3. By pass

The use of a token ring to perform local event building is simple and effective but is sensitive to any failure in a single component in the chain. A set of additional inputs for the token and the serial data are available to enable a slave TDC to be bypassed in case of failure.

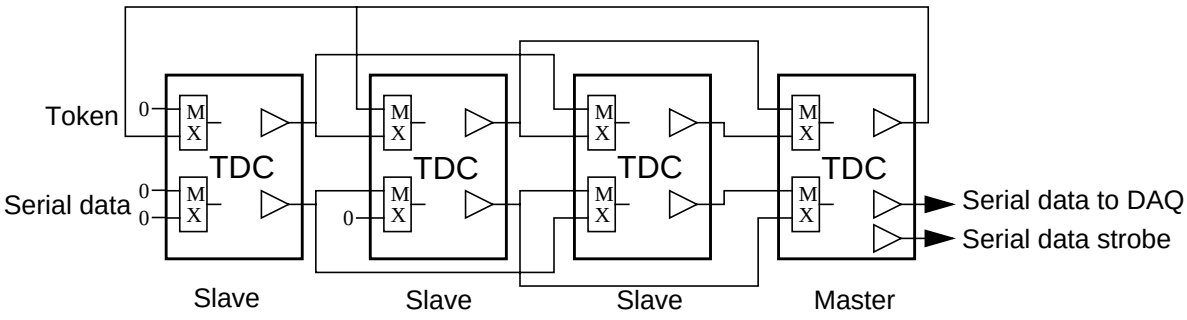


Fig. 11 :Serial connection of TDC's with option of bypassing failing slave TDC

10.4. Packet 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 9 out of the possible 16 packet types are defined for TDC data. The remaining 7 packet types are available for data packets added by higher levels of the DAQ system.

Group header: Event header from master 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	0	1	Global_type				Event ID												Bunch ID											

Global_type: Programmed global type (setup[38:35]).
Event ID: Event ID from event counter.
Bunch ID: Bunch ID of trigger (trigger time tag).

Group trailer: Event trailer from master 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	1	Global_type				Event ID												Word count											

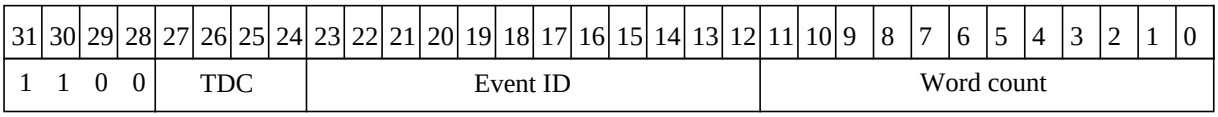
Global_type: Programmed global type (setup[38:35]).
Event ID: Event ID from event counter.
Word count: Total number of words in event (incl. headers and trailers).

TDC header: Event header from TDC (master and slaves)

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				Event ID												Bunch ID											

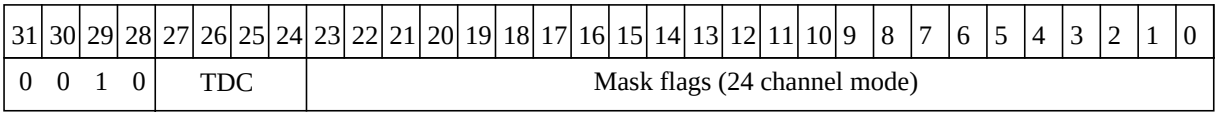
TDC: Programmed ID of TDC.
Event ID: Event ID from event counter.
Bunch ID: Bunch ID of trigger (trigger time tag).

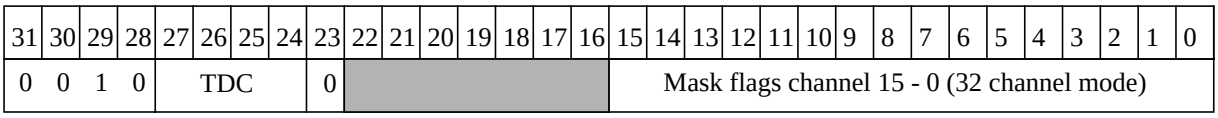
TDC trailer: Event trailer from TDC (master and slaves)

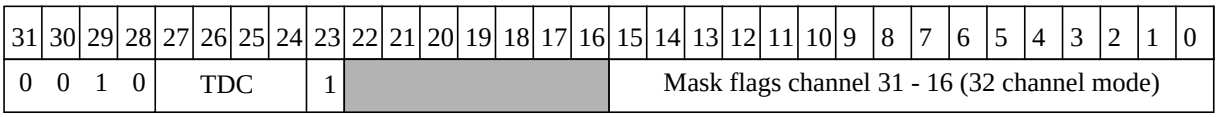


TDC: Programmed ID of TDC.
Event ID: Event ID from event counter.
Word count: Number of words from TDC (incl. headers and trailers).

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

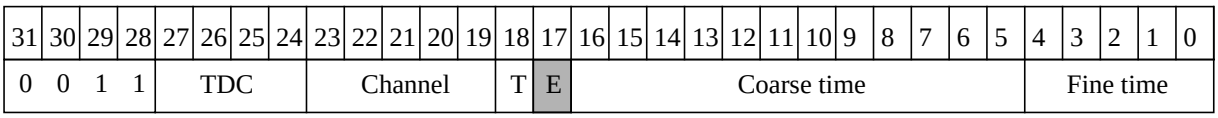






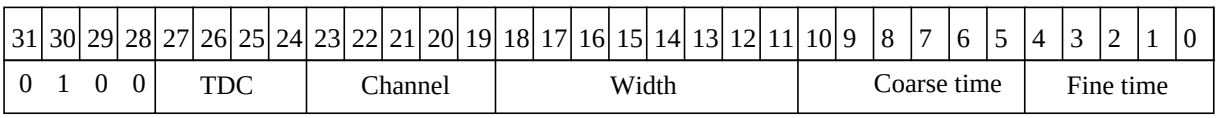
TDC: Programmed ID of TDC.
Mask flags: Channels flagged as having hits with in mask window.
NOTE: Configuration of packet depends on TDC working in 24 or 32 channel mode.

Single measurement: Single edge time measurement



TDC: Programmed ID of TDC.
Channel TDC channel number.
Coarse: Coarse time measurement (in clock cycles).
Fine Fine time measurement from DLL (in bins of 25ns/32).
T: Edge type: 1 = leading edge, 0 = trailing edge.
E: Hit error. An error has been detected in the hit measurement (DLL lost lock)

Combined measurement: Combined measurement of leading and trailing edge



TDC: Programmed ID of TDC.
Channel: TDC channel number.
Width: Width of pulse in programmed time resolution.
Coarse: Coarse time measurement of leading edge relative to trigger (in clock cycles).
Fine: 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																					Error flags						

TDC: Programmed ID of TDC.
Error flags: [0]: Hit data have been lost (L1 buffer overflow).
[1]: Event have been lost (trigger fifo overflow).
[2]: Hit data have been lost (read-out fifo overflow).
[3]: A hit measurement have been corrupted (DLL may be out of lock).
[4]: An internal error have been detected in TDC.
[5]: Buffer overflow (matching not enabled).
[6]: Hit have been rejected in channel buffer (NOT IMPLEMENTED).

Debugging data: Additional information for system debugging

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				Sub type																							

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				0	0	0	0									Bunch ID											

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				0	0	0	1												R	L1 occupancy							

TDC: Programmed ID of TDC.
Sub type: Debugging data sub type:
0000: separator
0001: Buffer occupancy
Bunch ID: Trigger time tag counter when separator was generated.
L1 occupancy: L1 buffer occupancy.
R: Read-out fifo full.

11. Error monitoring.

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. DLL signals captured when a hit is detected is checked to verify that the DLL is in a correct locking state. The JTAG programming data and the JTAG instruction register also have a parity check to detect if any of the bits have been corrupted during down load or by a Single Event Upset (SEU). The error status of the individual parts can be accessed via the JTAG status scan path.

Vernier error: DLL signals in a hit measurement indicates that the DLL is not in correct lock.

Coarse error: A parity error in the coarse count has been detected in a channel buffer.

Select error: A synchronisation error has been detected in the priority logic used

	to select the channel being written into the L1 buffer.
L1 buffer error:	Parity error detected in L1 buffer.
Trigger FIFO error:	Parity error detected on trigger FIFO.
Matching state error:	Illegal state detected in trigger matching logic.
Read-out FIFO error:	Parity error detected in read-out FIFO.
Read-out state error:	Illegal state detected in read-out logic.
Setup error:	Parity error detected in setup data.
Control error:	Parity error detected in control data.
JTAG error:	Parity error in JTAG instruction.

Any detected error condition in the TDC sets its corresponding error status bit which remains set until a global reset of the TDC is performed. All the available error flags are or-ed together with individual programmable mask bits to generate a global error signal. When the global error signal becomes active the TDC can respond in a number of different ways:

Ignore:	No special action will be performed
Mark events:	All events being generated after the error has been detected will be marked with a special error flag.
Bypass:	From the following event the TDC will suppress all its own data and directly pass the token and serial data.

12. JTAG Test and Programming Port.

A JTAG (Joint Test Action Group, IEEE 1149.1 standard) port is used to program the programmable features in the TDC, and also to get access to test facilities built into the TDC. Full 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 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.

Programming of the device is separated into two scan path groups. The setup group consists of setups which can not be changed while the system is actively running and a control group which can be changed during a run (enable and disable of noisy channels). To save silicon area for the large shift register for the setup data it does not have an update register as normally seen on JTAG scan paths.

12.1. JTAG 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

Instruction:	Name	Description
0000:	EXTEST.	Boundary scan for test of inter-chip connections on module.
0001:	IDCODE.	Scan out of chip identification code.
0010:	SAMPLE.	Sample of all chip pins via boundary scan registers.

0011:	INTEST.	Using boundary scan registers to test chip itself.
0100 - 0111:	NOT IMPLEMENTED.	
1000:	Setup.	Load of setup data.
1001:	Control.	Load of control information.
1010:	Status.	Read of status information.
1011:	Coretest	Access to internal test scan registers.
1100 - 1110:	NOT IMPLEMENTED.	
1111	BYPASS.	

12.2. Boundary scan register.

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).

BSR #	Pin name	Description
[0]	token_out	
[1]	strobe_out	
[2]	serial_out	
[3]	error	
[4]	data_ready	
[5]	parallel_enable	enable of parallel_data_out (not direct pin)
[37:6]	parallel_data_out[31:0]	
[38]	encoded_control	
[39]	trigger	
[40]	event_reset	
[41]	bunch_reset	
[42]	get_data	
[43]	serial_bypass_in	
[44]	serial_in	
[45]	token_bypass_in	
[46]	token_in	
[47]	reset	
[48]	clk	
[80:49]	hit	

12.3. ID code

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

[0]		Start bit = 1.
[11:1]	ES2 code	ES2 manufacturer code = 00001000111.
[27:12]	TDC part code	decimal 3500 = binary 0000110110101100
[31:28]	Version code	1000

12.4. Setup registers

The JTAG setup scan path is used to load programming data that can not be changed while the TDC is actively running. No separate update register is implemented on this scan path!.

[0]	enable_error_mark	mark all events with error word if internal error

[1]	enable_error_bypass	bypass TDC chip if internal error detected
[12:2]	enable_error[10:0]	enables of internal error types bit # 0: vernier error 1: coarse error 2: channel select error 3: l1 buffer parity error 4: trigger fifo parity error 5:trigger matching error (state error) 6: read-out fifo parity error 7: read-out state error 8: setup parity error 9: control parity error 10: jtag instruction error
[15:13]	readout_single_cycle_speed	serial transmission speed when single cycle mode 000: 40 Mbits/s 001: 20 Mbits/s 010: 10 Mbits/s 011: 5 Mbits/s 100: 2,5 Mbits/s 101: 1.25 Mbits/s 110: 0.625 Mbits/s 111: 0.3125 Mbits/s
[19:16]	serial_delay[3:0]	programmable delay of serial input (unit ~ 1ns)
[21:20]	strobe_select[1:0]	selection of serial strobe type: 00: no strobe 01: DS strobe 10: leading and trailing edge (edge) 11: leading edge (clock)
[23:22]	readout_speed_select[1:0]	selection of serial read-out speed 00: single cycle 01: 80 Mbits/s (setup[164] must also be set) 10: 160 Mbits/s (setup[163] must also be set) 11: not valid
[27:24]	token_delay[3:0]	programmable delay of token input (unit ~ 1ns)
[28]	enable_local_trailer[0]	enable of local trailers in read-out
[29]	enable_local_header[0]	enable of local headers in read-out
[30]	enable_global_trailer[0]	enable of global trailers in read-out (only valid for master TDC)
[31]	enable_global_header[0]	enable of global headers in read-out (only valid for master TDC)
[32]	keep_token[0]	keep token until event separator (used when matching)
[33]	master[0]	master chip in token ring
[34]	enable_serial[0]	enable of serial read-out (otherwise parallel read-out)
[38:35]	global_type[3:0]	Global_type (bit [27:24]) of global header/trailer
[42:39]	tdc_id[3:0]	TDC identifier
[43]	select_bypass_inputs[0]	select serial in and token in from bypass inputs

[44]	enable_relative[0]	enable read-out of relative time to trigger time tag
[45]	enable_error_mark_overflow[0]	enable of error marking events with overflow
[46]	enable_error_mark_rejected[0]	error mark event if rejected seen during matching (Not implemented in this version)
[47]	enable_gate_trigger_full_reject[0]	read-out fifo full reject only if trigger fifo nearly full
[48]	enable_gate_l1_full_reject[0]	read-out fifo full reject only if l1 buffer nearly full
[49]	enable_full_reject[0]	enable reject hits when read-out fifo full
[50]	enable_l1_occupancy_readout[0]	enable read-out of l1 occupancy for each event (debugging)
[51]	enable_separator_readout[0]	enable read-out of internal separators (debugging)
[52]	channel24[0]	enable of 24 channel mode (determines mapping of mask flags)
[53]	enable_mark_rejected[0]	enable special marker word for rejected hits (Not implemented in this version)
[54]	enable_mask[0]	enable search and read-out of mask flags
[55]	enable_matching[0]	enable of trigger matching
[67:56]	mask_window[11:0]	mask window in number of clock cycles
[79:68]	search_window[11:0]	search window in number of clock cycles
[91:80]	match_window[11:0]	matching window in number of clock cycles
[92]	enable_automatic_reject[0]	enable of automatic rejection
[104:93]	reject_count_offset[11:0]	rejection counter offset
[105]	enable_buffer_overflow_detect[0]	enable of l1 buffer overflow detection
[117:106]	event_count_offset[11:0]	event number offset
[129:118]	trigger_count_offset[11:0]	trigger time tag counter offset
[130]	enable_set_counters_on_bunch_reset[0]	enable all time counters to be reset on bunch count reset
[131]	enable_master_reset_code[0]	enable master reset code on encoded_control
[132]	enable_master_reset_on_event_reset[0]	enable master reset of whole TDC on event reset
[133]	enable_reset_channel_buffer_when_separator[0]	enable reset channel buffers when separator
[134]	enable_separator_on_event_reset[0]	enable generate separator on event reset
[135]	enable_separator_on_bunch_reset[0]	enable generate separator on bunch count reset
[136]	enable_direct_event_reset[0]	enable of direct event reset input pin
[137]	enable_direct_bunch_reset[0]	enable of direct bunch reset input pin
[138]	enable_direct_trigger[0]	enable of direct trigger input pin
[141:139]	width_select[2:0]	pulse width resolution (0 = 0.78 ns, 1 = 1.56 ns, 2 = 3.12 ns, , ,)
[142]	test_invert[0]	automatic inversion of test pattern when in test mode
[143]	test_mode[0]	enable test mode

[155:144]	coarse_count_offset[11:0]	coarse time counter offset
[159:156]	charge_pump_current_b[3:0]	setting of charge pump current in DLL 0000: during lock acquisition of DLL 1110: during normal operation.
[160]	enable_force_rejected[0]	force generation of rejected hit (not implemented)
[161]	enable_trailing[0]	enable of trailing edges
[162]	enable_leading[0]	enable of leading edges
[163]	enable160[0]	enable generation of 160MHz clock (for 160 Mbits/s serial read-out)
[164]	enable80[0]	enable generation of 80MHz clock (for 80 Mbits/s serial read-out)
[176:165]	count_roll_over[11:0]	counter roll over value
[177]	enable_pair[0]	enable pairing of leading and trailing edges
[178]	enable_ttl_serial[0]	enable TTL input on: serial_in serial_bypass_in token_in token_bypass_in Disables LVDS drivers on: Serial_out strobe_out token_out
[179]	enable_ttl_control[0]	enable TTL input on: trigger bunch_reset event_reset encoded_control
[180]	enable_ttl_reset[0]	enable TTL input on: reset
[181]	enable_ttl_clock[0]	enable TTL input on: clock
[182]	enable_ttl_hit[0]	enable TTL input on: hit[31:0]
[183]	parity[0]	parity check of setup data

12.5. Control registers

The JTAG control scan path is used to enable/disable channels which can be done while the TDC is actively running. The control scan path is also use to initialize the DLL after power up. A global reset (equivalent to asserting the reset pin) can be issued to initialize the global state of the TDC after power up.

[0]	Global_reset[0]	Global reset of TDC via JTAG.
[32:1]	enable_channel[31:0]	enable/disable of individual channels
[33]	dll_reset[0]	reset of delay locked loop
[34]	control_parity[0]	parity of control data

12.5.1.Performing a reset of the DLL.

The Delay Locked Loop is not initialized when a global reset of the TDC chip is performed. The lock tracing of the DLL is a rather slow process (few ms) and is only required to be performed once after power has been applied.

A reset of the DLL must be performed via the JTAG control scan path after power up. First a load of the setup registers must be performed with the charge pump current level set to their maximum level (0000). The DLL_reset bit in the control scan path must now be set to one to force the DLL into a defined state. To enable the DLL to start lock tracing from its reset state the DLL_reset signal must be released again in the same manner. Now the DLL will start lock tracing and when lock has been obtained the not_locked status bit will be cleared. When locking has been obtained the charge pump current level can be lowered to reduce jitter in the DLL (1110).

12.6. 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).

[10:0]	error[10:0]	status of error monitoring bit # 0:vernier error 1:coarse error 2:channel select error 3:l1 buffer parity error 4:trigger fifo parity error 5:trigger matching error (state error) 6:readout fifo parity error 7:readout state error 8:setup parity error 9:control parity error 10:jtag instruction error
[11]	have_token[0]	TDC have read-out token
[12]	readout_fifo_empty[0]	read-out fifo empty
[13]	readout_fifo_full[0]	read-out fifo full
[21:14]	l1_occupancy[7:0]	l1 buffer occupancy
[22]	l1_overflow[0]	l1 buffer overflow
[23]	l1_overflow_recover[0]	l1 buffer overflow recover
[24]	l1_nearly_full[0]	l1 buffer nearly full
[25]	l1_empty[0]	l1 buffer empty
[26]	trigger_fifo_full[0]	trigger fifo full
[27]	trigger_fifo_nearly_full[0]	trigger fifo nearly full
[28]	trigger_fifo_empty[0]	trigger fifo empty
[29]	dll_lock[0]	DLL in lock.

12.7. Internal test registers

The JTAG internal test scan path is used to perform extended testing of the TDC chip. The internal 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 select bits in the setup scan path and is only intended for verification and production tests of the TDC chip.

[11:0]	matching_state[11:0]	monitoring of trigger matching state	
		state_waiting	000000000001
		state_write_event_header	000000000010
		state_write_occupancy	000000000100
		state_active	000000001000

		state_write_mask_flags1	000000010000
		state_write_mask_flags2	000000100000
		state_write_error	000001000000
		state_write_event_trailer	000010000000
		state_generating_lost_event_header	000100000000,
		state_generating_lost_event_trailer	001000000000
		state_waiting_for_separator	010000000000
		state_write_separator	100000000000
[38:12]	trigger_data[26:0]	monitoring of active trigger data	
		[11:0]	bunch id
		[23:12]	event id
		[24]	separator
		[25]	trigger lost
		[26]	parity of trigger data
[39]	trigger_ready[0]	monitoring of active trigger	
[75:40]	l1_data[35:0]	monitoring of hit data to trigger matching	
		COMBINED MEASUREMENT:	
		[4:0]	leading edge fine time
		[16:5]	leading edge coarse time
		[24:17]	width
		[29:25]	channel
		[30]	rejected hit
		[31]	hit error
		[32]	overflow start
		[33]	overflow stop
		[34]	separator
		[35]	parity
		SINGLE MEASUREMENT:	
		[4:0]	edge fine time
		[16:5]	edge coarse time
		[17]	edge type
		[24:18]	undefined (0)
		[29:25]	channel
		[30]	rejected hit
		[31]	hit error
		[32]	overflow start
		[33]	overflow stop
		[34]	separator
		[35]	parity
[76]	l1_empty[0]	monitoring of hit data to trigger matching	
[77]	l1_data_ready[0]	monitoring of hit data to trigger matching	
[197:78]	hit_data[119:0]	monitoring/generation of hit data from TDC macro	
		[31:0]	first vernier
		[43:32]	first coarse 1
		[44]	first coarse 1 parity
		[56:45]	first coarse 2
		[57]	first coarse 2 parity
		[58]	first edge type
		[59]	first rejected

		[91:60] second vernier
		[103:92] second coarse 1
		[104] second coarse 1 parity
		[116:105] second coarse 2
		[117] second coarse 2 parity
		[118] second edge type
		[119] second rejected
[202:198]	hit_channel[4:0]	monitoring of hit data from TDC macro
[203]	hit_select_error[0]	monitoring of hit data from TDC macro
[204]	hit_load[0]	monitoring of hit data from TDC macro

13. Time alignment between hits, clock and trigger matching

The TDC contains many programmable setups which have effects on the performed time measurements and their optional 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.

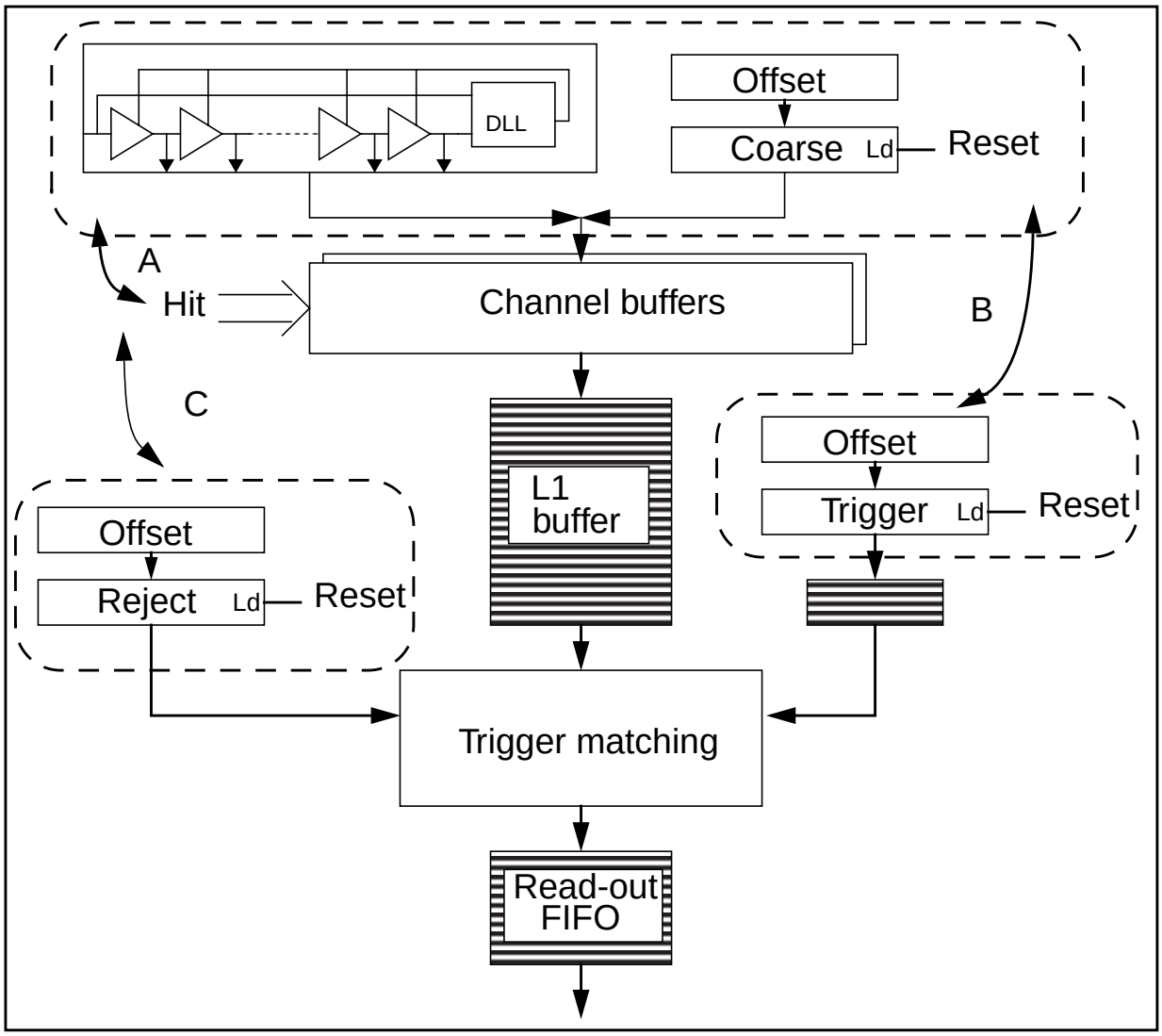


Fig. 12 :Time alignment between hits, trigger and automatic reject

13.1. 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 the figure below a hit signal is defined such that the time measurement equals zero (coarse_count_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).

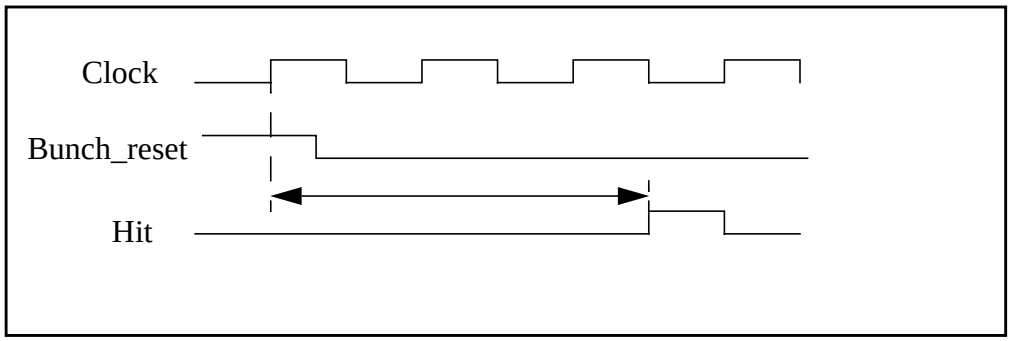


Fig. 13 :Definition of reference time measurement.

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

13.3. 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_count_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 count offset of zero is used.

14. Signals.

Clk:	I	Clock. LVDS mode: clock TTL mode: clock
Clkb:	I	Clock inverted. LVDS mode: clock inverted TTL mode: not used.
reset:	I	master reset of state machines and buffers LVDS mode: reset TTL mode: reset
resetb:	I	master reset inverted LVDS mode: reset inverted TTL mode: not used
bunch_reset:	I	reset of coarse count, trigger time tag count, reject count. LVDS mode: bunch reset TTL mode: bunch reset
bunch_resetb:	I	bunch reset inverted. LVDS mode: bunch reset inverted TTL mode: not used
event_reset:	I	reset of event counter LVDS mode: event reset TTL mode: event reset
event_resetb:	I	event reset inverted. LVDS mode: event reset inverted TTL mode: not used
trigger:	I	load of trigger into trigger fifo LVDS mode: trigger TTL mode: trigger
triggerb:	I	trigger inverted. LVDS mode: trigger inverted TTL mode: not used
encoded_control:	I	encoded control (trigger, bunch reset, event reset, master reset) LVDS mode: encoded control TTL mode: encoded control
encoded_controlb:	I	encoded control inverted. LVDS mode: encoded control inverted TTL mode: not used
hit[31:0]	I	hit inputs LVDS mode: hit inputs TTL mode: hit inputs
hitb[31:0]	I	hit inputs inverted LVDS mode: hit inputs inverted TTL mode: not used

token_in	I	token input LVDS mode: token input TTL mode: token input
token_inb	I	token input inverted LVDS mode: token input inverted TTL mode: not used
token_bypass_in	I	token bypass input LVDS mode: token bypass input TTL mode: token bypass input
token_bypass_inb	I	token bypass input inverted LVDS mode: token bypass input inverted TTL mode: not used
serial_in	I	serial input LVDS mode: serial input TTL mode: serial input
serial_inb	I	serial input inverted LVDS mode: serial input inverted TTL mode: not used
serial_bypass_in	I	serial bypass input LVDS mode: serial bypass input TTL mode: serial bypass input
serial_bypass_inb	I	serial bypass input inverted LVDS mode: serial bypass input inverted TTL mode: not used
token_out	O	token output in LVDS
token_outb	O	token inverted output in LVDS
serial_out	O	serial output in LVDS
serial_outb	O	serial inverted output in LVDS
strobe_out	O	strobe output in LVDS
strobe_outb	O	strobe inverted output in LVDS
token_out_ttl	O	token output in TTL
serial_out_ttl	O	serial output in TTL
strobe_out_ttl	O	strobe output in TTL
error	O	common error output (TTL).
data_ready	O	data ready for read-out (parallel read-out mode) (TTL).
get_data	I	get parallel data (CMOS)
parallel_data_out	O/Z	parallel data out [31:0] (TTL)
tck	I	JTAG test clock (TTL).
trst	I	JTAG reset (active low) (TTL).

tms	I	JTAG test mode select.(TTL)
tdi	I	JTAG test data in.(TTL)
tdo	O/Z	JTAG test data out (TTL).
vdd_core	I	VDD for internal core. (good decoupling recommended)
gnd_core	I	Ground for internal core. (good decoupling recommended)
vdd_out	I	VDD for TTL output drivers.
gnd_out	I	Ground for TTL output drivers.
vdd_lvds_in	I	VDD for LVDS inputs.
gnd_lvds_in	I	Ground for LVDS inputs.
vdd_lvds_out	I	VDD for LVDS outputs.
gnd_lvds_out	I	Ground for LVDS outputs.

14.1. Signal timing.

#	NAME	Description	Reference edge	Min.	Max	units
0	Tpclk	Clock period		25	50	ns
		Jitter	CLK+		0.1	ns
1	Tstok	Token_in setup	CLK+	5		ns
2	Thtok	Token_in hold	CLK+	2		ns
3	Tdtok	Token_out delay	CLK+	4	16	ns
4	Tdrdy	Data_ready delay	CLK+	4	16	ns
5	Tsget	Get_data setup	CLK+	5		ns
6	Thget	Get_data hold	CLK+	2		ns
7	Tdpdat	parallel_data delay	CLK+	4	16	ns
8	Tzpdat	parallel_data tristate delay	CLK+	4	16	ns

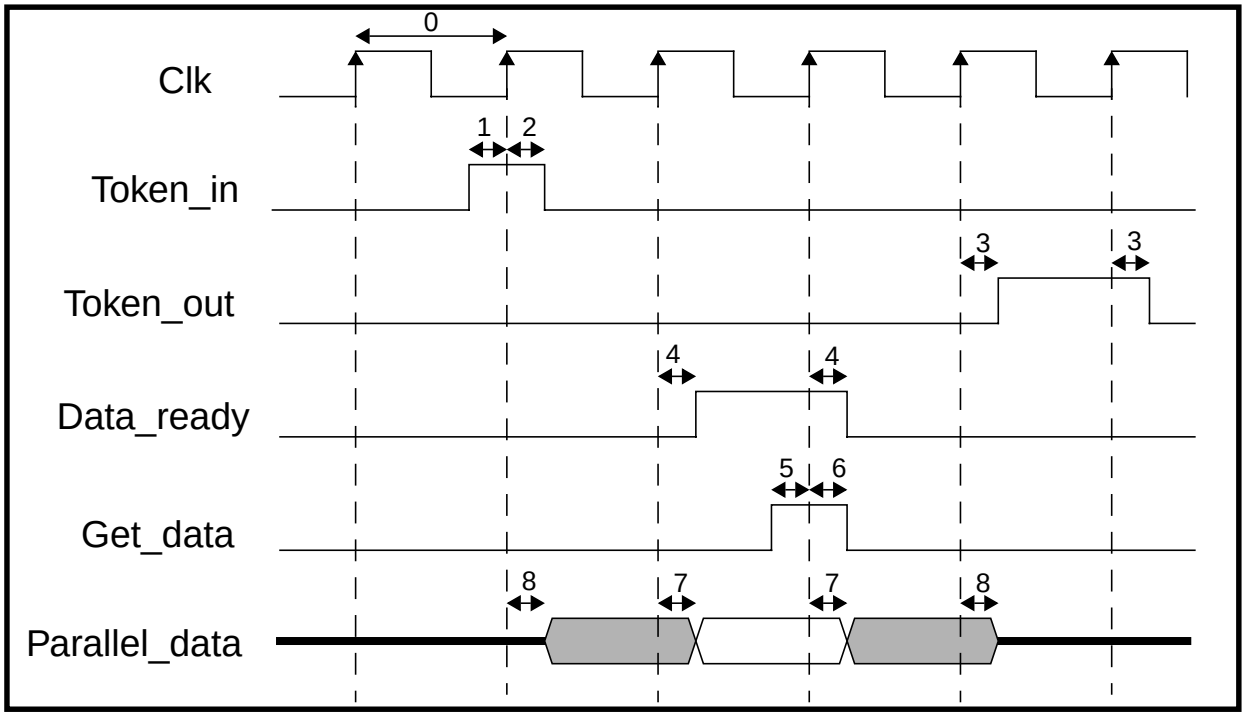


Fig. 14 :Timing of parallel read-out signals

#	Name	Description	Reference edge	Min	Max	units
10	Tsres	Reset setup	CLK+	5		ns
11	Thres	Reset hold	CLK+	2		ns
12	Tsbres	Bunch_reset setup	CLK+	5		ns
13	Thbres	Bunch reset hold	CLK+	2		ns
14	Tseres	Event_reset setup	CLK+	5		ns
15	Theres	Event reset hold	CLK+	2		ns
16	Tstrg	Trigger setup	CLK+	5		ns
17	Thtrg	Trigger hold	CLK+	2		ns
18	Tsenc	Encoded_control setup	CLK+	5		ns
19	Thenc	Encoded_control hold	CLK+	2		ns
20	Tderr	Error delay	CLK+	4	16	ns

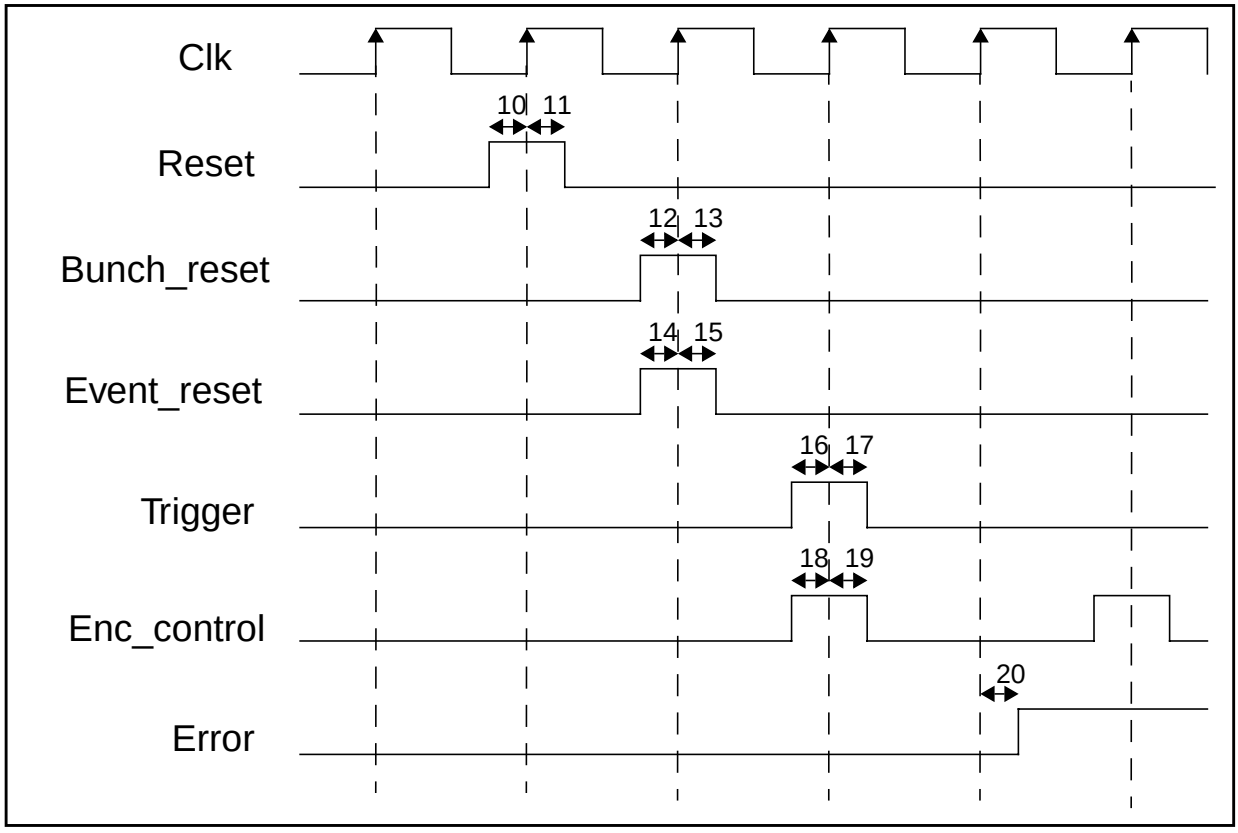


Fig. 15 :Timing of resets, trigger and error.

#	Name	Description	Reference edge	Min	Max	units
30	Tsser	Serial_in setup (f <=40 MHz)[1]	CLK+	5		ns
31	Thser	Serial_in hold (f <=40 MHz)[1]	CLK+	2		ns
32	Tdser	Serial_out delay	CLK+	4	16	ns
33	Tdstr	Strobe_out delay	CLK+	4	16	ns
34	Tsstrser	Strobe_out, serial_out skew[2,3]	Strobe+-		+/-1	ns
35	Tjser	Serial_out jitter			+/-1	ns
36	Tsskew	Serial transfer skew[1] @40mbits/s			15	ns
		@80mbits/s			5	ns
		@160mbits/s			1	ns

Note1: When connecting the serial output of one TDC into the serial input of the next TDC it can be assumed that the internal delays in the two chips are nearly equivalent. The serial transfer between two TDC's is "guaranteed" to work if ("clock skew between the two TDC's" + "delay from serial_out to serial_in") is below Tsskew [36]. If this can not be guaranteed it is possible to artificially skew the serial_in via the programming (setup[19:16]).

Note 2: The strobe signal is generated such that the strobe and the serial data changes at the same time within a skew of +/- 1ns. It is up to the receiver of the data to time the sampling of the serial-data using the available strobe. The specified skew is with identical loads on the two outputs.

Note 3: During the synthesis of the AMT0 an additional delay of one clock period was accidentally introduced on the strobe signal in the clock mode when using a serialization speed below 40 MHz. This means that in this mode the rising edge of the strobe (clock) will occur 25 ns after the change of data (see bug list).

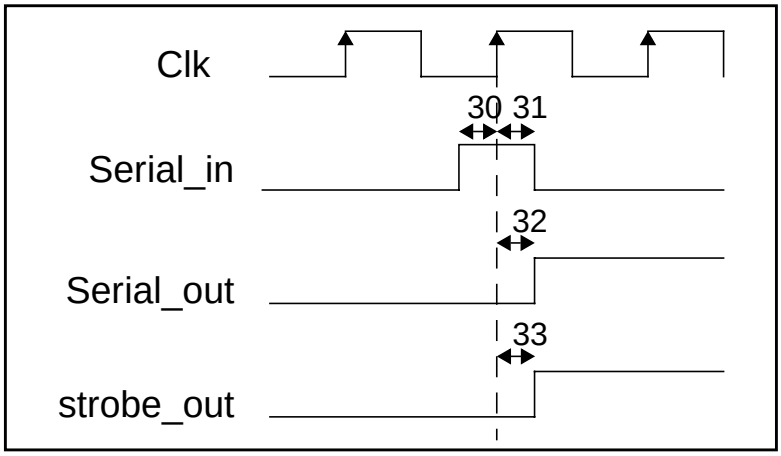


Fig. 16 :Timing of serial data in relation to external TDC clock

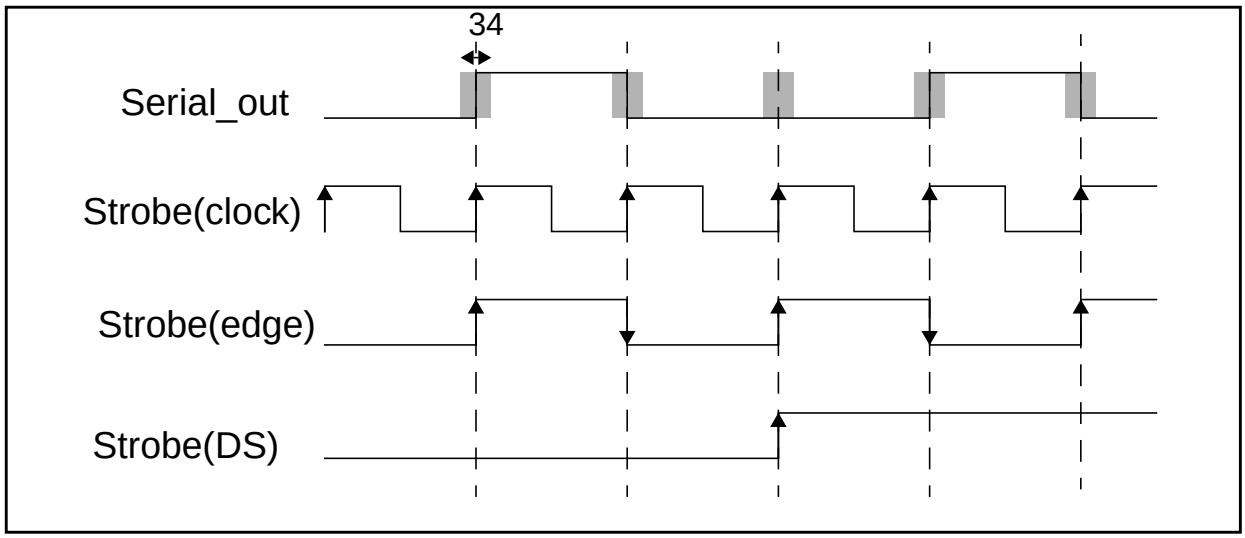


Fig. 17 :Timing of strobe and serial data signals

#	Name	Description	Reference edge	Min	Max	units
40	Tptck	tck clock period		50		ns
41	Twtrst	trst width		25		ns
42	Trtrst	trst recovery	TCK+	5		ns
43	Tstms	tms setup	TCK+	5		ns
44	Thtms	tms hold	TCK+	2		ns
45	Tstdi	tdi setup	TCK+	15		ns
46	Thtdi	tdi hold	TCK+	4		ns
47	Tdtdo	tdo delay	TCK-	4	16	ns
48	Tztdo	tdo tristate delay	TCK-	4	16	ns

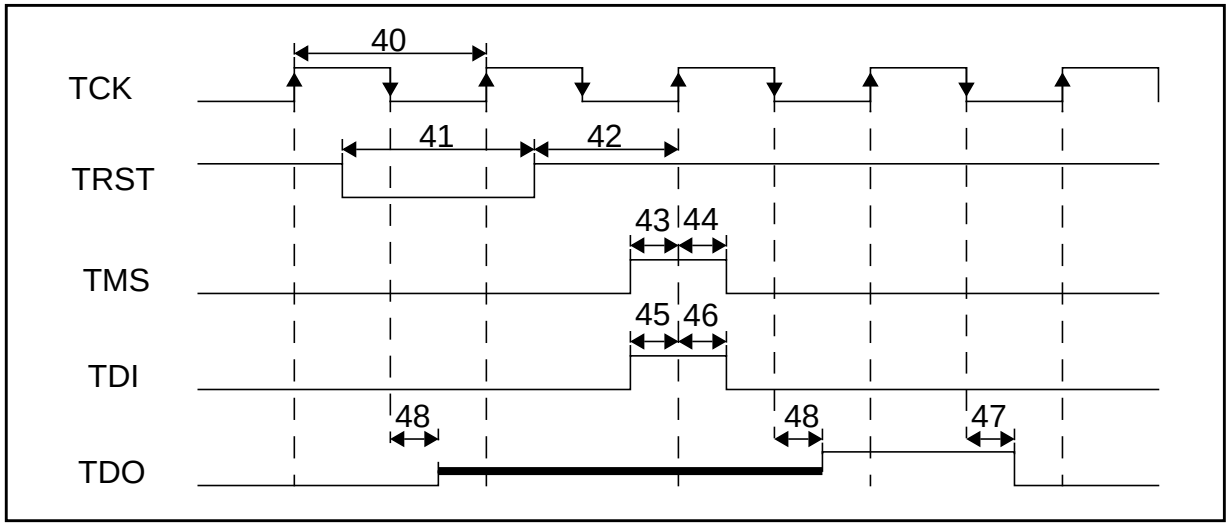


Fig. 18 :Timing of JTAG signals.

15. Pin Layout

The AMT0 will be packaged in a 180 pin PGA for prototype evaluation (AMT 0.0).

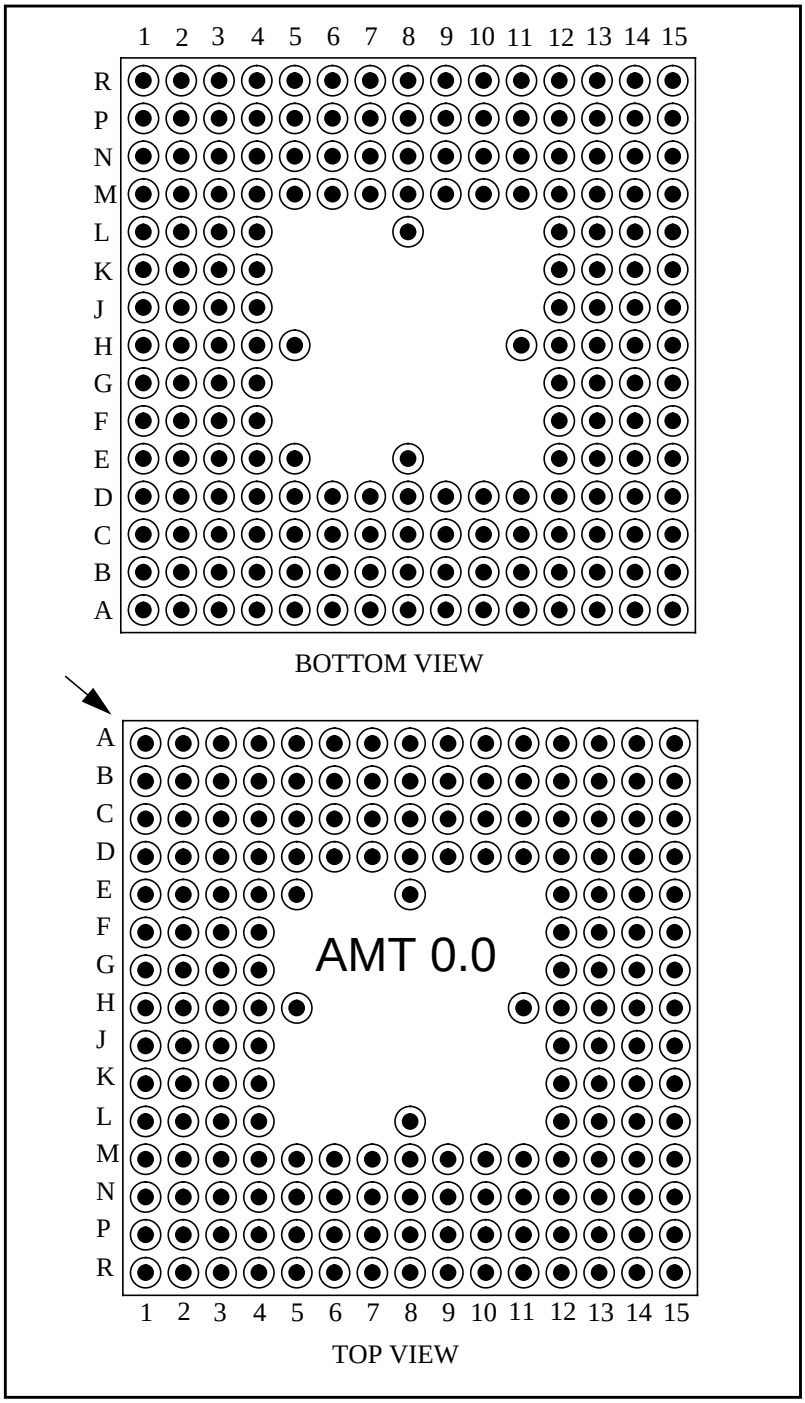


Fig. 19 :180 pin PGA package for prototypes

Small scale production chips will be packaged in a 160 pin plastic QFP.

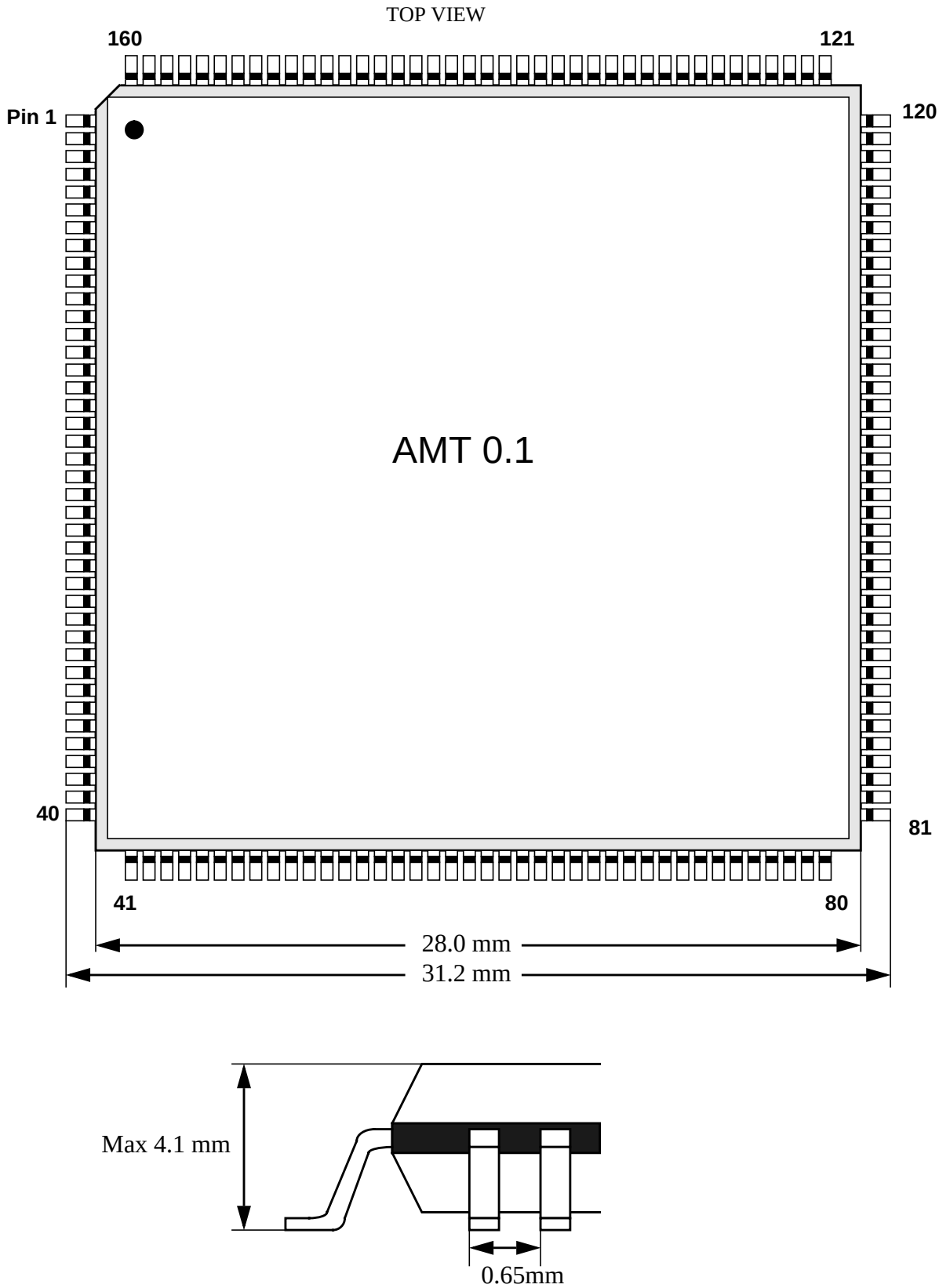


Fig. 20 :160 pin QFP package.

Pos.	I/O cell	Pad	PGA pin	QFP pin	Signal
		136	D11		NC
		137	D10		NC
		138	D9		NC
top 0	289	139	A14	81	gnd_lvds_in
top 1	284	140	A13	82	vdd_lvds_in
top 2	58	141	A12	83	hit[0]
		142	A11	84	hitb[0]
top 3	59	143	A10	85	hit[1]
		144	A9	86	hitb[1]
top 4	60	145	B13	87	hit[2]
		146	B12	88	hitb[2]
top 5	57	147	B11	89	hit[3]
		148	B10	90	hitb[3]
top 6	55	149	B9	91	hit[4]
		150	C12	92	hitb[4]
top 7	54	151	C11	93	hit[5]
		152	C10	94	hitb[5]
top 8	53	153	C9	95	hit[6]
		154	D8	96	hitb[6]
top 9	56	155	D7	97	hit[7]
		156	D6	98	hitb[7]
top 10	294	157	D5	99	vdd_core
top 11	296	158	E8	100	gnd_core
top 12	63	159	D4	101	hit[8]
		160	C8	102	hitb[8]
top 13	62	161	C7	103	hit[9]
		162	C6	104	hitb[9]
top 14	61	163	C5	105	hit[10]
		164	C4	106	hitb[10]
top 15	64	165	C3	107	hit[11]
		166	B8	108	hitb[11]
top 16	66	167	B7	109	hit[12]
		168	B6	110	hitb[12]
top 17	67	169	B5	111	hit[13]
		170	B4	112	hitb[13]
top 18	68	171	B3	113	hit[14]
		172	B2	114	hitb[14]
top 19	65	173	A8	115	hit[15]
		174	A7	116	hitb[15]
top 20	47	175	A6	117	hit[16]
		176	A5	118	hitb[16]
top 21	46	177	A4	119	hit[17]
		178	A3	120	hitb[17]
		179	A2		NC
		180	A1		NC
		1	B1		NC
		2	C1		NC
		3	D1		NC
left 0	45	4	E1	121	hit[18]
		5	F1	122	hitb[18]
left 1	48	6	G1	123	hit[19]
		7	H1	124	hitb[19]
left 2	50	8	C2	125	hit[20]
		9	D2	126	hitb[20]

Pos.	I/O cell	Pad	PGA pin	QFP pin	Signal
left 3	51	10	E2	127	hit[21]
		11	F2	128	hitb[21]
left 4	52	12	G2	129	hit[22]
		13	H2	130	hitb[22]
left 5	49	14	D3	131	hit[23]
		15	E3	132	hitb[23]
left 6	42	16	F3	133	hit[24]
		17	G3	134	hitb[24]
left 7	43	18	H3	135	hit[25]
		19	E4	136	hitb[25]
left 8	295	20	F4	137	vdd_core
left 9	297	21	G4	138	gnd_core
left 10	44	22	H4	139	hit[26]
		23	H5	140	hitb[26]
left 11	41	24	J4	141	hit[27]
		25	K4	142	hitb[27]
left 12	40	26	L4	143	hit[28]
		27	M4	144	hitb[28]
left 13	39	28	J3	145	hit[29]
		29	K3	146	hitb[29]
left 14	38	30	L3	147	hit[30]
		31	M3	148	hitb[30]
left 15	24	32	N3	149	hit[31[]
		33	J2	150	hitb[31]
left 16	285	34	K2	151	vdd_lvds_in
left 17	290	35	L2	152	gnd_lvds_in
left 18	194	36	M2	153	reset
		37	N2	154	resetb
left 19	195	38	P2	155	bunch_reset
		39	J1	156	bunch_resetb
left 20	196	40	K1	157	event_reset
		41	L1	158	event_resetb
left 21	197	42	M1	159	trigger
		43	N1	160	triggerb
		44	P1		NC
		45	R1		NC
		46	M5		NC
		47	M6		NC
		48	M7		NC
bot. 0	198	49	R2	1	encoded_control
		50	R3	2	encoded_controlb
bot. 1	199	51	R4	3	token_in
		52	R5	4	token_inb
bot. 2	200	53	R6	5	token_bypass_in
		54	R7	6	token_bypass_inb
bot. 3	201	55	P3	7	serial_in
		56	P4	8	serial_inb
bot. 4	202	57	P5	9	serial_bypass_in
		58	P6	10	serial_bypass_inb
bot. 5	286	59	P7	11	vdd_lvds_in
bot. 6	291	60	N4	12	gnd_lvds_in
bot. 7	193	61	N5	13	clk
		62	N6	14	clkb
bot. 8	287	63	N7	15	vdd_lvds_in

Pos.	I/O cell	Pad	PGA pin	QFP pin	Signal
bot. 9	292	64	M8	16	gnd_lvds_in
bot. 10	288	65	M9	17	vdd_lvds_out
bot. 11	293	66	M10	18	gnd_lvds_out
bot. 12	205	67	M11	19	token_out
		68	L8	20	token_outb
bot. 13	203	69	M12	21	serial_out
		70	N8	22	serial_outb
bot. 14	204	71	N9	23	strobe_out
		72	N10	24	strobe_outb
bot. 15	298	73	N11	25	vdd_core
bot. 16	300	74	N12	26	gnd_core
bot. 17	302	75	N13	27	vdd_out
bot. 18	306	76	P8	28	gnd_out
bot. 19	269	77	P9	29	token_out_ttl
bot. 20	270	78	P10	30	serial_out_ttl
bot. 21	271	79	P11	31	strobe_out_ttl
bot. 22	34	80	P12	32	tck
bot. 23	35	81	P13	33	tms
bot. 24	36	82	P14	34	tdi
bot. 25	37	83	R8	35	trst
bot. 26	22	84	R9	36	tdo
bot. 27	21	85	R10	37	error
bot. 28	19	86	R11	38	data_ready
bot. 29	31	87	R12	39	get_data
bot. 30	310	88	R13	40	gnd_core
		89	R14		NC
		90	R15		NC
		91	P15		NC
		92	N15		NC
		93	M15		NC
right 0	107	94	L15	41	parallel_data_out[31]
right 1	106	95	K15	42	parallel_data_out[30]
right 2	105	96	J15	43	parallel_data_out[29]
right 3	104	97	H15	44	parallel_data_out[28]
right 4	2	98	N14	45	parallel_data_out[27]
right 5	101	99	M14	46	parallel_data_out[26]
right 6	102	100	L14	47	parallel_data_out[25]
right 7	103	101	K14	48	parallel_data_out[24]
right 8	108	102	J14	49	parallel_data_out[23]
right 9	303	103	H14	50	vdd_out
right 10	307	104	M13	51	gnd_out
right 11	109	105	L13	52	parallel_data_out[22]
right 12	110	106	K13	53	parallel_data_out[21]
right 13	111	107	J13	54	parallel_data_out[20]
right 14	115	108	H13	55	parallel_data_out[19]
right 15	114	109	L12	56	parallel_data_out[18]
right 16	113	110	K12	57	parallel_data_out[17]
right 17	112	111	J12	58	parallel_data_out[16]
right 18	299	112	H12	59	vdd_core
right 19	301	113	H11	60	gnd_core
right 20	124	114	G12	61	parallel_data_out[15]
right 21	125	115	F12	62	parallel_data_out[14]
right 22	126	116	E12	63	parallel_data_out[13]
right 23	127	117	D12	64	parallel_data_out[12]

Pos.	I/O cell	Pad	PGA pin	QFP pin	Signal
right 24	131	118	G13	65	parallel_data_out[11]
right 25	130	119	F13	66	parallel_data_out[10]
right 26	304	120	E13	67	vdd_out
right 27	308	121	D13	68	gnd_out
right 28	129	122	C13	69	parallel_data_out[9]
right 29	128	123	G14	70	parallel_data_out[8]
right 30	123	124	F14	71	parallel_data_out[7]
right 31	122	125	E14	72	parallel_data_out[6]
right 32	121	126	D14	73	parallel_data_out[5]
right 33	120	127	C14	74	parallel_data_out[4]
right 34	116	128	B14	75	parallel_data_out[3]
right 35	117	129	G15	76	parallel_data_out[2]
right 36	118	130	F15	77	parallel_data_out[1]
right 37	119	131	E15	78	parallel_data_out[0]
right 38	305	132	D15	79	vdd_out
right 39	309	133	C15	80	gnd_out
		134	B15		NC
		135	A15		NC

NC = Not Connected

16. Technical Specifications.

Number of channels:	32 (special 24 channel mode supported)
Clock frequency:	40 MHz nominal 20 - 60 MHz for typical chips
Time bin size:	0.78 ns @ 40 MHz
Differential non linearity:	Max = +/-0.16 ns, RMS = 0.071 ns
Integral non linearity:	Max= +/-0.23 ns, RMS = 0.11 ns
Time resolution:	0.29 ns RMS
Difference between channels:	Maximum one time bin
Variation with temperature:	Maximum one time bin
Cross talk:	Maximum +- 2 bins from 31 simultaneous changing channels to one channel.
Dynamic range:	12 + 5 = 17 bit
Double pulse resolution:	15 ns (if both hit registers are free).
Max. recommended Hit rate:	500 KHz per channel, all 32 channels used 1 MHz per channel, 16 channels used.
Event buffer size:	256
Read-out buffer size.:	32
Trigger buffer size.:	8
Power supply:	4.75 - 5.25 volt Typical 100 mA, max 200 mA
Temperature range:	-40 - 80 Deg. Cent.
Hit inputs:	LVDS or TTL

17. Time Resolution

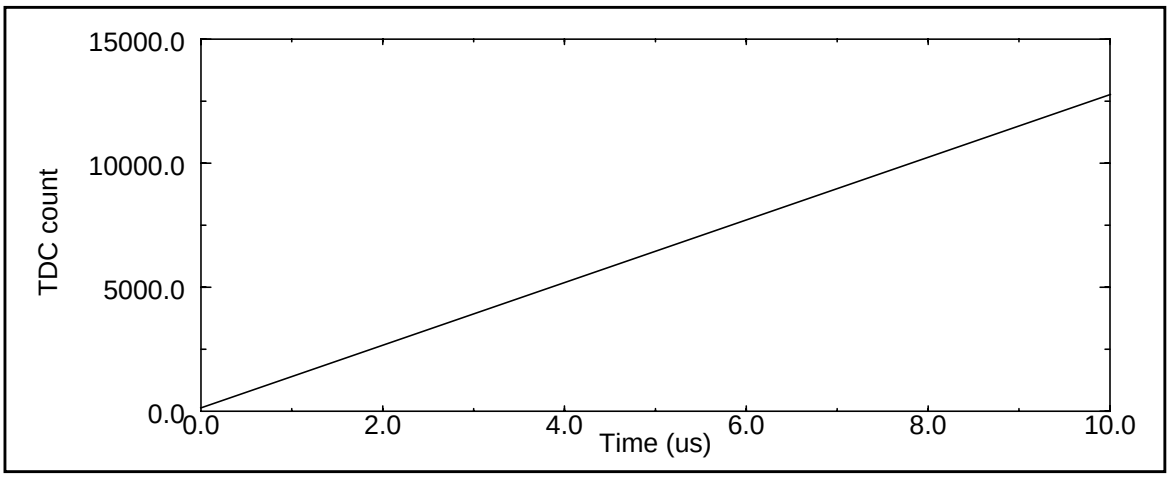


Fig. 21 :10 us Time sweep

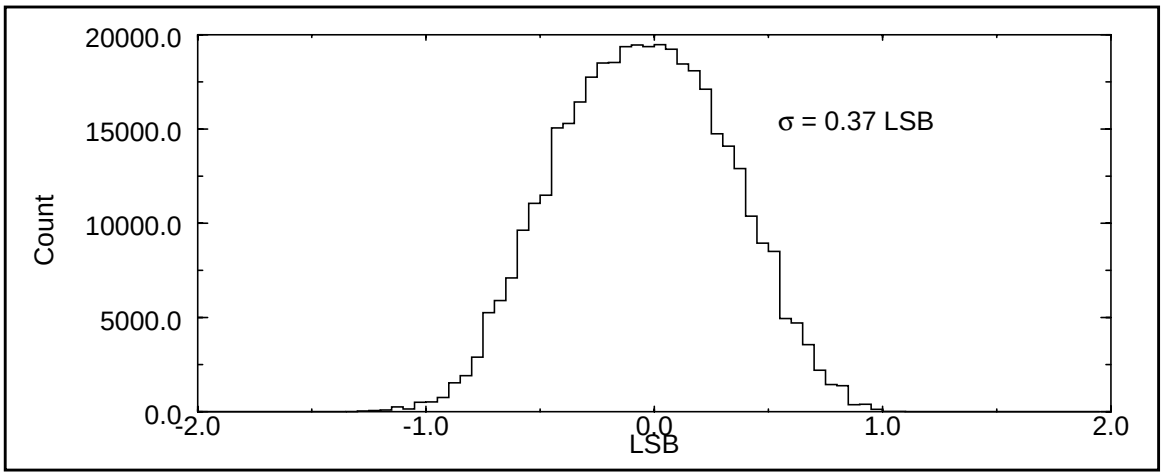


Fig. 22 :Error histogram

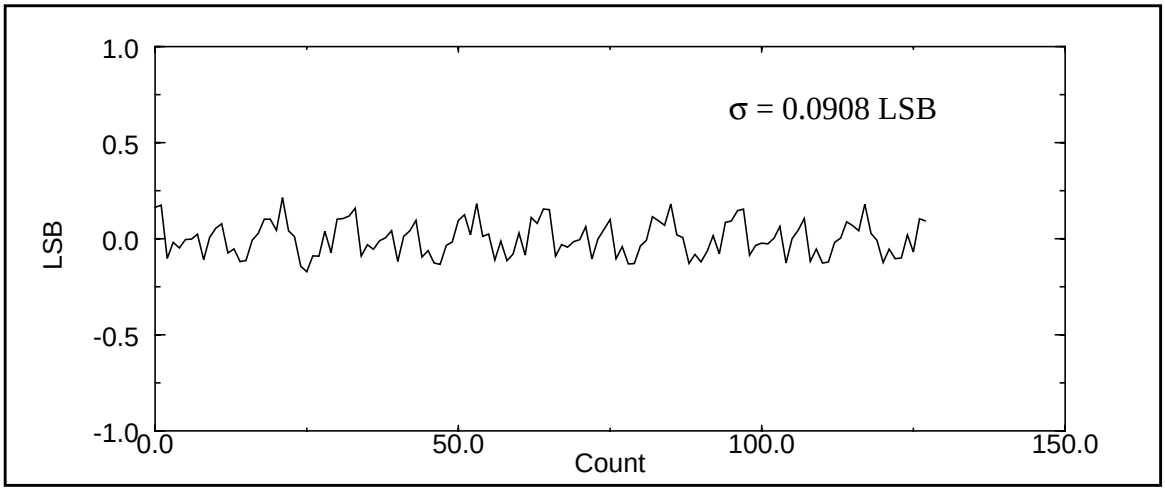


Fig. 23 :Differential nonlinearity

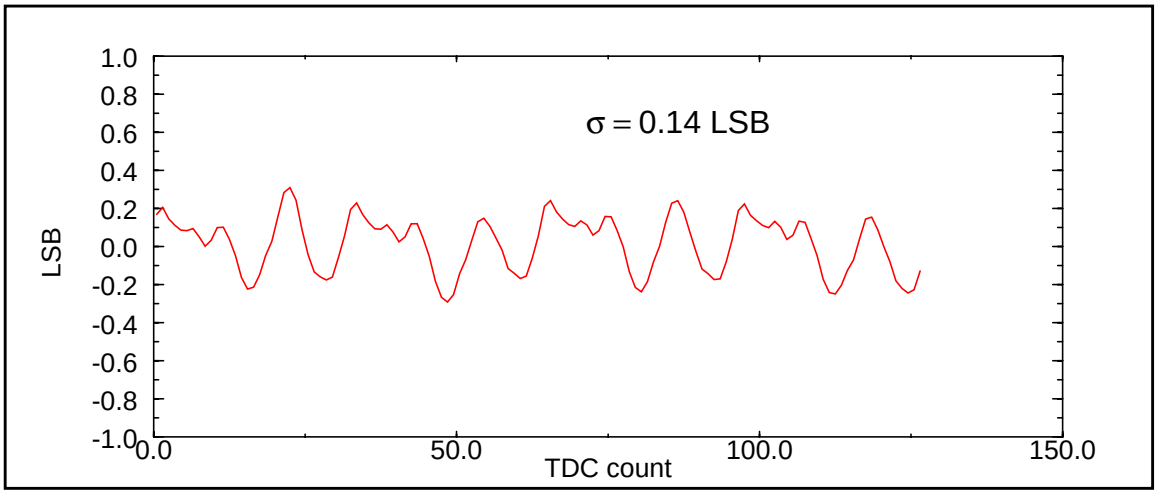


Fig. 24 :Integral non linearity

18. Bug list

18.1. Single edge measurement

The first prototype version (AMT 0.0) could not perform single edge measurements. This has been corrected in the production version (AMT 0.1).

18.2. Timing of clock strobe signal when serial read-out speed below40 MHz

During the synthesis of the AMT0 an additional delay of one clock period was accidentally introduced on the strobe signal in the clock mode when using a serialization speed below 40 MHz. This means that in this mode the rising edge of the strobe (clock) will occur 25 ns after the change of data.

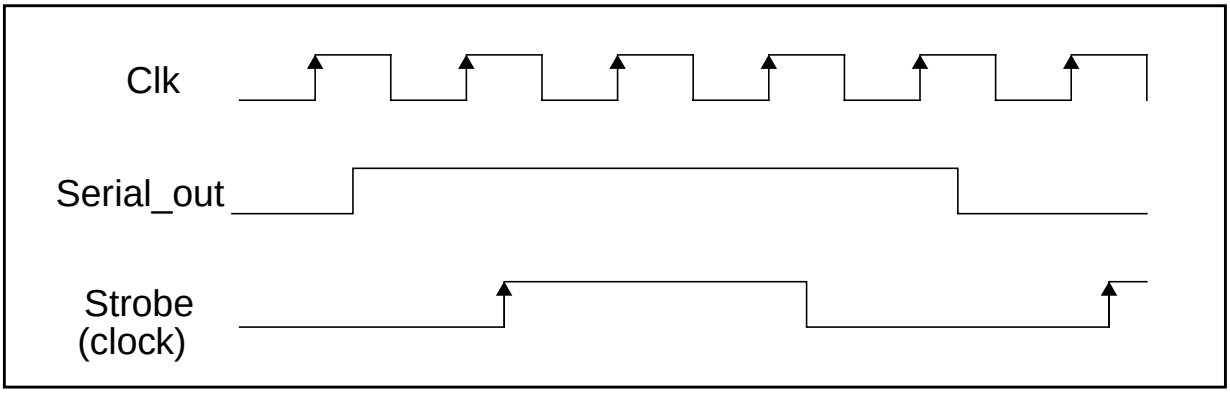


Fig. 25 :Delayed clock strobe at serial speed lower than 40Mbits/s (shown at 10mbits/s)

18.3. CMOS input levels on get_data input pin

By mistake the get_data input buffer is of CMOS type having an input threshold voltage of Vdd/2.

18.4. Potential problem changing charge pump currents after lock of DLL.

After locking of the DLL has been obtained it may be required to change the charge pump current to reduce the jitter in the DLL. To load a new charge pump current value the setup scan path must be used. Some of the bits in the setup scan chain is also used to define if the clock input is LVDS or TTL levels. During the shifting of the new setup data the internal clock for the DLL may therefore get corrupted and there is a potential risk that the DLL will lose lock. The slower the loading of the setup data is performed the higher is the risk that the DLL may loose lock. If this problem is seen it will be necessary to use the same charge pump currents for locking and during operation. If small charge pump currents are used the DLL locking may not be performed correctly if the input clock has jitter. If large (max) currents are used the rms resolution of the TDC during operation will be reduced by approximately 50ps.

18.5. Potential problem with false DLL locking when using clock with jitter

It has been found that the DLL locking may not be performed correctly if the input clock have a significant amount of jitter (~100ps). The sensitivity to this problem is inversely proportional to the charge pump currents used in the DLL during locking. The DLL lock status bit (status[29]) will be set as if the DLL has obtained correct lock, but when the first hit measurement is performed the TDC will detect that the DLL is in fact not in correct lock. This will set the vernier error status (status[0]) bit. If single edge measurements is performed the hit error bit (bit 17) will mark the detection of the measurement being corrupted. For combined measurements this error bit is not available in the data read out because of the limited number of bits available for the double measurement.

19. Recommendations for future versions of AMT

19.1. Problem with trigger matching algorithm if pairing and falling edge generated very late.

If paired time measurements of leading and trailing edge are performed and the trailing edge of the hit is generated long time after the leading edge (caused by some malfunction of the analog front-end) it may prevent the correct hits to be extracted from the L1 buffer. The problem is that the base for the trigger matching is the time of the leading edge but a paired measurement is not written into the L1 buffer before the trailing edge also have been detected. If an excessive delay from the leading edge to the trailing edge is generated on a channel the time order in the L1 buffer is seriously affected. The trigger matching can accept a certain “dis-order” in the l1 buffer by using the programmable search window. If the disorder gets bigger than the search window one may risk to loose hits in the trigger matching.

If the occurrence of this kind of malfunction of the analog front-end is estimated to be likely the trigger matching algorithm can with minor changes be made to skip this kind of late hits in the L1 buffer.

19.2. Implementation of BIST on memories.

In the AMT0 the internal memories are tested via the internal JTAG scan path. The test patterns to perform an exhaustive test of all the memories via this serial access though becomes very long and may give problems on production testers with limited test vector depth. The use of Built in self test of internal memories would solve this problem.

19.3. Read-out of trigger fifo and read-out fifo occupancies.

A special read-out packet has been defined to be capable of reading out in real time the occupancies of internal memories. In AMT0 only the occupancy of the L1 buffer is implemented because implementation specific details of the FIFO macros used for the trigger and read-out FIFO. It is recommended to include the read-out of the occupancy of the other buffers if possible.