Preliminary (Draft 2.3)

A High Resolution Time to Digital Converter

Users Manual

Manuel Mota[1]
LIP - Lisboa, ECP/MIC - CERN - Geneva (mota@sunvlsi.cern.ch)

1.0 Introduction.

This document briefly describes the architecture and functionality of a Time-to-Digital Converter (TDC) demonstrator chip developed at the ECP/MIC division of CERN. This circuit was designed taking in consideration the specific resolution needs of the read-out of a Time-of-Flight (TOF) detector such as the “Pestov Spark Counters”, being considered by the ALICE experiment.

The TDC demonstrator is based on the concept of an Array of Delay Locked Loops (ADLL), with which it is possible to achieve the high resolution required, together with a considerable dynamic range in a self-calibrating circuit built in standard digital CMOS technologies. Using an 80 MHz reference clock, a bin size of 89.3 ps and a dynamic range of 3.2 μs is obtained. An RMS (root mean square) resolution of ~35 ps has been measured.

The time-stamp corresponding to a Hit in one of the four TDC channels integrated into the chip, is converted into a binary format and then stored in a common derandomizing FIFO where it is stored, ready to be read-out from the circuit.

1.1 The TOF detector of the ALICE experiment.

In a TOF detector, particles are identified by measuring the time they use to transverse a given part of the detector volume. The “Pestov Spark Counter” is a good candidate for TOF measurements in the Heavy Ion Experiment (ALICE), because of it’s very good time resolution (25-50 ps) and to the high output pulse amplitude (larger than 0.5V) [1].

Such characteristics mean that there is no need to use a pre-amplifier, but require compatible time resolution from the discriminator and TDC connected to it. For the TDC, an RMS resolution smaller than 50 ps is needed in order not to degrade the detector’s signal intrinsic proprieties.

For this experiment, the interaction rate envisaged is not very high (100 KHz in the case of p-p collisions, but only 8 KHz for Pb-Pb and Ca-Ca collisions) and the TOF detector’s occupancy is kept bellow 10% by defining a suitable detector granularity. These conditions ease the requirements from the data-acquisition circuitry, since the read-out of each channel can be done at correspondingly low rates.

The front-end electronics (discriminators, TDCs and QDCs), mounted on top of the detector, will be housed inside the same pressure vessel that encapsulates the counters. This results in a controlled environment, where severe temperature variations are not expected during the working periods. Power dissipation remains an issue, though; cooling of the vessel by convection is sufficient to remove 50 mW/channel, but more heat may be generated by the detector and front-end electronics.

1. Supported by a grant from the Junta Nacional de Investigação Científica e Tecnológica (JNICT) under the “Sub-Programa Ciência e Tecnologia do 2º. Quadro Comunitário de Apoio”.

2.0 Architecture overview.

Delay Locked Loops (DLL) are the core of the architecture implemented. Single DLL though have a time resolution limited to the basic gate delay. An array of DLLs with a small phase shift between each DLL is used in order to achieve a resolution that is a fraction of the gate delay.

2.1 The Delay Locked Loop.

The use of the intrinsic propagation delay of a signal through a basic gate in a Delay Line (DL) arrangement is a common procedure to achieve a good time resolution [2]. Propagation delays as low as 100 ps are attainable in standard CMOS processes. In such a scheme a reference signal edge is propagated through the DL; at the arrival of the Hit signal the DL taps are sampled into a register. The position of the edge along the DL reflects the time being measured.

Unfortunately the gate delay is very dependent on process parameters, temperature and supply voltage, leaving a big uncertainty on the effective delay of the line and requires frequent calibrations. The dynamic range of such circuits is limited by the length of the DL and the expansion of this range using a coarse counter synchronous with the reference signal is not trivial due to the difficult merging of these two measures into a coherent value.

Some of the limitations of simple uncontrolled DLs can be overcome if the delay of it's elements is continuously adjusted, using as reference a (stable) clock signal. This can be done if a voltage controlled delay line (VCDL) is included within a closed control loop, resulting in a structure similar to the Phase Locked Loops (PLL) that is called Delay Locked Loop (DLL).

In a DLL, the reference clock is propagated through the VCDL and the phase of the signal arriving at the end of the line is compared with the input reference phase. If a phase difference different from one clock cycle is detected, the closed loop will try to correct it by changing the control voltage of the VCDL. The time measurement stored at the arrival of a Hit is, consequently, related to the reference clock, resulting in a "time-stamp" measurement. "Start-stop" measurements can be obtained if a second channel is used to store the start value, which is then subtracted from the Stop measurement.

Such a structure, that can easily be integrated into a complex chip, exhibits a very good time resolution and results in a self-calibrated circuit insensitive to environment variations, at the price of a somewhat reduced attainable time resolution.

Dynamic range expansion is readily achieved by introducing a counter synchronous to the reference clock. Since both the DLL and the coarse counter measurements are obtained in respect to the same reference, the expansion of the dynamic range of the measurements are unambiguous.

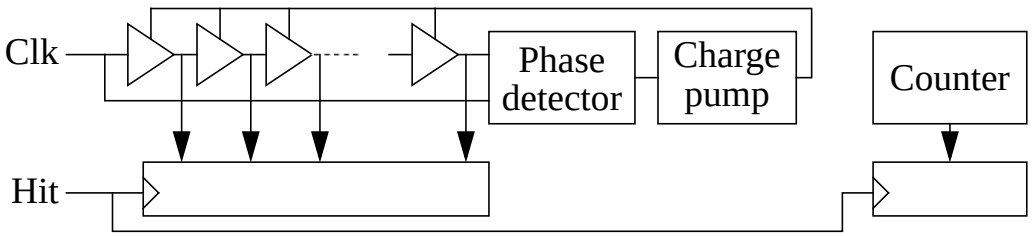

Fig. 1. Delay Locked Loop and hit registers.

To avoid metastability problems that can occur if the hit trigger arrives so close to the clock's rising edge that the counter's outputs aren't yet stable, two counters synchronous to opposite phases of the reference clock are used. Depending on the relative position of the hit arrival within the clock

period, as shown by the DLL measurement, the results of the counter that is farther from a transition will be used.

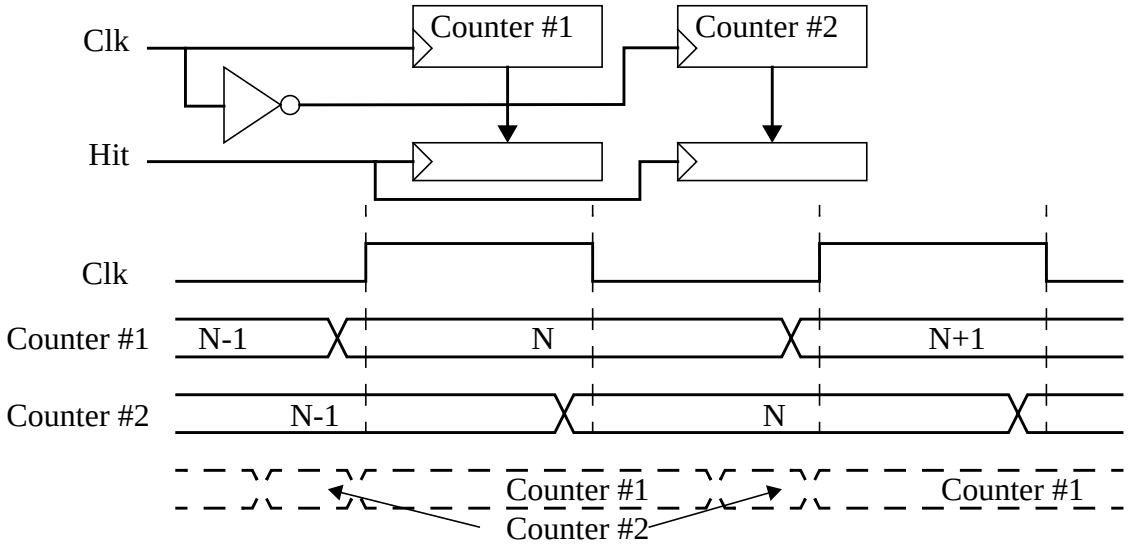

Fig. 2. Double coarse time counter.

Careful design of the delay looked loop is very important to guarantee that each delay element is exactly matched and have identical delay characteristics. A PLL configuration has the ability to filter out some of the jitter present in it's input reference, depending on the closed loop bandwidth. This is unfortunately not the case in a DLL structure, where the loop is unable to clean the reference signal's jitter which will directly influence the time measurements. A very clean reference signal should be available in order not to degrade the obtained measures.

2.2 The Array of Delay Locked Loops.

The use of an array of several uniformly offset DLLs can increase the time resolution to a fraction of the basic gate delay [3]. The problem of maintaining an uniform offset smaller than the gate delay is overcome by the use of a "phase-shifting" DLL having a smaller number of delay elements, locked to the same reference clock.

This way, a delay that is a fraction bigger than the basic cell delay is obtained. An arrangement like the one in *Fig. 3*, results in corresponding taps in consecutive DLLs being shifted by a delay that is $(1+F).\ T_b$, where F is a fraction of a unit. Due to the symmetry of the array, it is made to look like this delay shift is only $F.\ T_b$, if the tap previous to the corresponding one is used to define the limits of the time bins ($T_b.F = T_b.(1+F) - T_b$).

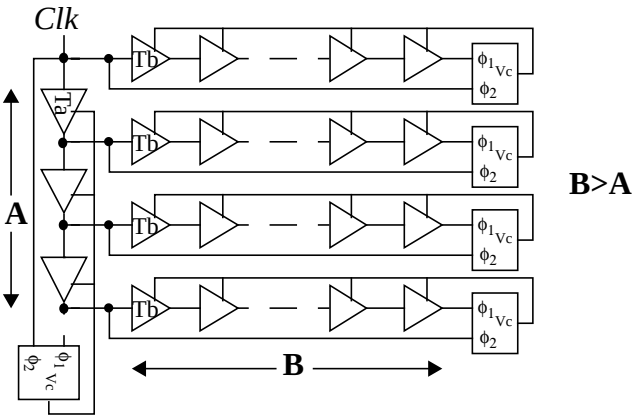

Fig. 3. Array of DLLs with phase shifting DLL.

The time bin of such an array is

$$T_{bin} = T_a - T_b = \frac{T_{clk}}{A} - \frac{T_{clk}}{B}$$

If the required time bin size is a fraction *F* of the basic cell delay of the DLLs of the array, then the relation between *A, B* and *F* can be expressed as

$$T_{bin} = \frac{T_{clk}/B}{F} = \frac{T_{clk}}{A} - \frac{T_{clk}}{B}$$

$$A = B \cdot \frac{F}{F+1}$$

where *A, B* and *F* are integers.

In this architecture, the time bins are obtained from the (small) difference between the (relatively large) delays of two delay elements in consecutive DLLs. Any mismatch of these delays will be amplified by the nature of the array and will result in a considerable degradation of the time resolution. Therefore careful design and layout of the delay elements is essential in order to achieve the best matching possible.

One disadvantage of this scheme is its inability to produce a number of bins that is a power of two. This means that the measurement obtained will not be in a binary unit of $1/2^N$, but rather in a unit of *1/(B.F)*. A special encoder that converts this code into a normal binary code must be used, before merging the fine interval measurement with the binary coarse time counter measurement.

3.0 The TDC demonstrator.

The DLL array scheme was chosen as a basis for this High Resolution TDC demonstrator, because of the many advantages it has in terms of resolution, and integration. Such an architecture can achieve the high resolution needed (rms resolution better than 50 ps)[5], without using a very high frequency clock. This enables the use of standard CMOS processes, with the inherent advantages such as low price, high integration and the possibility of realizing complex logic circuits in the same IC.

With an 80 MHz reference clock, a bin size of 89.3 ps is obtained when the DLLs of the array have 35 delay elements (*B=35*) and the phase shifting DLL has only 28 elements (*B=28*). The clock period is therefore divided into 140 time taps. Using an 8 bit coarse time counter synchronous with the same reference clock, the Dynamic range of the measurements is extended to 3.2µs.

To obtain such a high resolution from this circuit, tight constraints must be imposed to the reference and hit signals. Any jitter in these signals would degrade the overall resolution. It is required that these time critical inputs be driven trough differential (Pseudo-ECL) signals, in order to avoid any common mode noise coupling into the signals.

When a hit arrives to a channel the time information is stored in the respective hit register, coded in a 140 bits long fine time plus the two 8 bit coarse time words. These registers must then be read-out in order to free the channel to acquire new hits. In this demonstrator a simple arbitration scheme is used to extract data from the four implemented channels and store it in a common derandomizing read-out FIFO.

In order to reduce hit rejection rate, two hit registers were implemented per channel. This way the channel will be able to acquire a new hit even if the previous one was not extracted because the extraction circuitry is busy serving another channel.

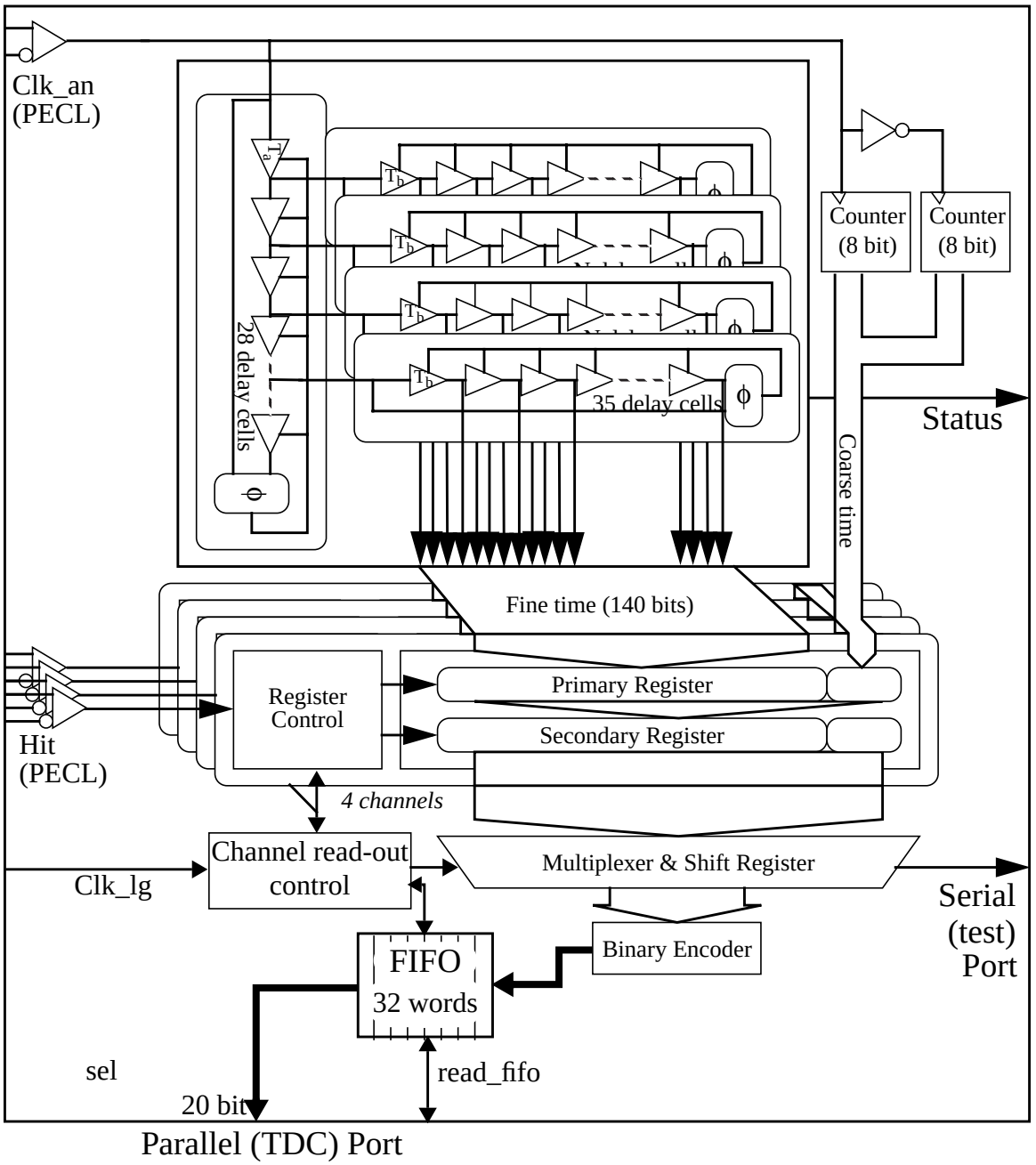

Fig. 4. Block diagram of the HRTDC demonstrator circuit.

The low hit rate and channel occupancy expected eases the speed and complexity requirements of the read-out circuitry. All the read-out logic can be made synchronous to a lower frequency clock (*Clk_lg* in *Fig. 4*) without any noticeable degradation of the circuit's performance. The read-out circuitry can run at a clock frequency of up to 40 MHz.

A binary encoder is included in the data path from the channel hit registers into the common FIFO. It's function is to encode the fine time word, which is in a 140 bits long thermometer like code, into a binary code. Unfortunately the encoded measurement does not relate to the same binary unit as the coarse counters, but to one that is 255/140 (~=1.82) times the coarse time binary unit.

The information of the time of the hit arrival in relation to the clock edge, available in the fine time measurement is used to select the correct coarse time counter word. A dedicated coarse counter reset pin (*reset_coarse*) is available to correctly initialize the counter for each measurement, if necessary. This reset signal must be synchronous to the reference clock. Time stamp measurements, for example, should use this feature. Relative measurements, obtained from the difference between two channel measurements (for ex., start/stop measurements) don't need to use it.

After encoding and coarse counter selection, all the data related to a hit is stored as a 20 bits wide word in the 32 words deep derandomizing FIFO. These data are then available to be extracted from the chip by an external read-out controller, using a simplified protocol: to the exterior the chip looks like a FIFO.

An independent *reset_buffers* signal can be used to discard data on all the buffers and restart the read-out control logic without having to re-initialize the Array, a process that may take more time than available.

3.1 Array Initialization.

The initialization procedure is started by the assertion of the *reset_dll* signal. It is a critical phase for any DLL-like architecture. During initialization the closed control loop must obtain correct lock to one reference clock period.

For lock to be obtained correctly, the reference clock period should be within the locking range of the DLLs, otherwise the loop might try to lock on 'zero' delay or on the double of the delay. In this chip several tracking and locking delay ranges were implemented, and a small state machine is included to force the loop to lock in the correct delay. The choice of delay range (and the necessity of forcing the loop into the locking range) are dependent on process parameters and expected environment conditions. These will be determined when the circuit is characterized, prior to utilization.

The successful locking is signalled by the assertion of two signals, *lock_dllB3* and *lock_dllA*. After the circuit is locked into the correct range, the closed loop will be able to track temperature and supply voltage variations of more than $15^{o}C$ and $300mV$ in any locking range, resulting in a self-calibrated circuit.

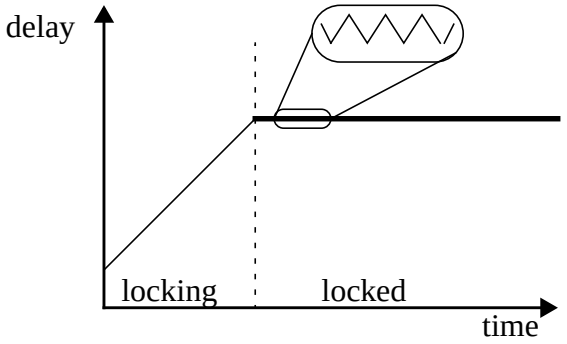

Fig. 5. 'Bang-bang' behaviour of the DLL.

The architecture chosen for the DLL results in a 'bang-bang' configuration [4]. This configuration has the behaviour highlighted in *Fig. 5*, with the total delay oscillating between two values when the DLL is locked. It will inevitably generate some small variations of the time bin sizes, decreasing the resolution of the time measurements. This jitter can be kept small by limiting the current levels flowing in and out of the filter capacitor through the charge pump. Unfortunately, since the same cur-

rent levels are used to move the DLL through the locking range to the desired working point, the locking time may become quite slow (less than 10 μs).

The different current levels are programmed into the chip in order to make it possible to select the lowest current level for which the DLL is still functioning properly in the given environment conditions. The programming settings will also be pre-determined and given to the user. Appendix A. describes to procedure to obtain the correct program and an example of a typical program.

3.2 Timing performance.

Converters are usually characterized using a given set of metrics, such as Differential and Integral non-linearity, Gain error, Offset. Although very much used, some of these metrics have different definitions depending on the application, therefore a set of appropriate definitions is given, and discussed:

- Differential Non-Linearity (DNL) is the deviation of the output bin size from the ideal value of one least significant bit (LSB). The result is usually presented as an histogram representing all the bins, together with its standard deviation.
- Integral Non-Linearity (INL) is the deviation of the input/output characteristic and a straight line of ideal gain (slope) that best fits the curve[1]. Using this definition, Gain error is always zero, because its effect is included in the INL result. The INL histogram is usually presented, together with its standard deviation.
- Gain error is the deviation of the slope of the line used in the INL calculation from its ideal value. As stated before, the definition of INL used results in null gain error.
- Offset is the vertical intercept of the line used in the INL calculation. In our case, this definition results in a relative offset of the transfer curve. An absolute offset would have to take into account the offset due to different signal paths of the reference and hit signals within (and outside) of the circuit. Since an absolute offset value depends on the system where the TDC is incorporated, and is anyway easily measured, no further mention is made of this metric.

Another important characteristic of the converter, which reflects its behaviour in the presence of random error sources such as loop jitter, electrical noise or quantization noise is the Conversion error:

- Conversion Error is the deviation of the input/output characteristic from a straight line of ideal gain (slope) that best fits the curve. The result is presented as an histogram of the error, and its standard deviation is defined as the converter RMS Resolution.

This definition is quite similar to the INL definition, the difference being on the method by which the transfer curve is obtained. In this case it is obtained via a linear time sweep over the dynamic range, while the INL histogram is obtained using randomly generated hits in Code Density Tests.

This metric reflects a different way of characterizing the circuit, very appropriate for Particle Physics Experiments, where most of the detectors have a statistical response, with a given standard deviation, to a particle crossing.

1. The best fit to the curve is, in this sense, the straight line with a determined slope that results in the smallest least square error. It is not necessarily the best least squares fit to the curve since this might have a different slope that would minimise the least square error.

3.2.1 Linearity Histograms.

The histograms shown in this section relate only to the time interpolation within the 12.5 ns clock period performed by the array. Since dynamic range extension is performed simply by appending the number of clock cycles elapsed, the DNL and INL of the full 3.2μs dynamic range is the replication of the single clock period histogram along the 256 clock cycles of the dynamic range. A DNL histogram over a few clock cycles is also shown, demonstrating the periodicity of the linearity.

The linearity histograms are obtained by collecting of ~840,000 randomly generated hits. Note that this is a kind of statistical averaging method which, by its nature excludes all error sources of random nature, such as noise or jitter.

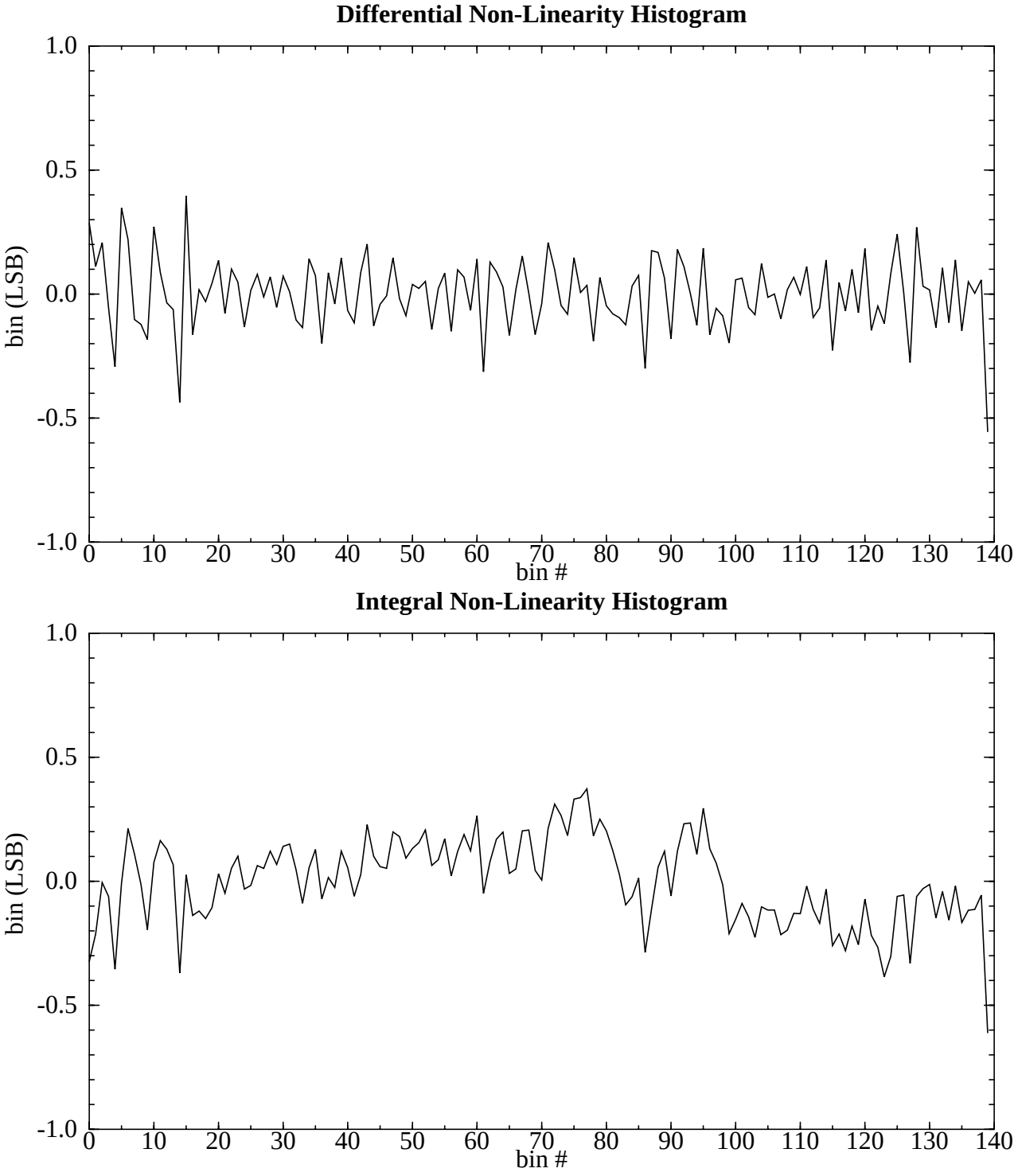

Fig. 6. Differential and Integral Non-Linearity Histogram (σ=0.14 LSB and σ=0.17 LSB, respectively).

The DNL histogram in *Fig. 6* shows that most of the bins in the array fall well within the half LSB limit, with the exception of the last bin (0.55 LSB). This bigger error, together a noticeable linearity degradation in the first 15 bins of the array are due to de fact that they are generated from the interpolation of taps in both extremities of consecutive DLLs. Phase errors and boundary effects on the extremities of the DLLs will, by the nature of the interpolation, be accumulated in this bins.

These effects can also be seen in the INL histogram of *Fig. 6.* Again a single bin in the array falls out of the half LSB limit (0.61 LSB).

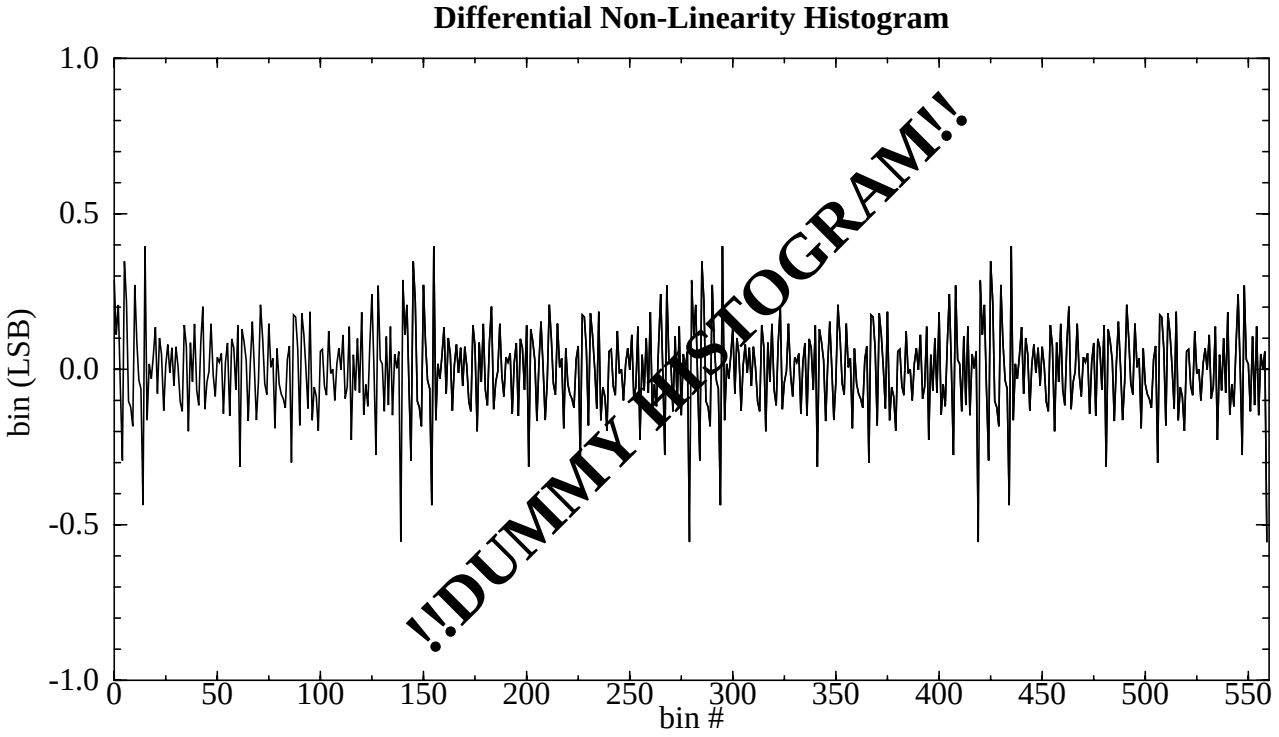

Fig. 7. Extended dynamic range Differential Non-Linearity Histogram.

The histogram of *Fig. 7* shows the clock periodicity of the DNL curve, demonstrating that dynamic range extension does not affect the linearity of the converter.

3.2.2 Time Sweeps.

Two sets of linear time sweeps were used to obtain the Conversion error of the TDC and to demonstrate the correct functionality of the dynamic range extension circuitry. The plot of *Fig. 8* shows the result of a linear time sweep performed with a very small time step (measurements every five steps ~2.75ps).

These time sweeps were performed while running all the internal logic at the full 40 MHz frequency, therefore displaying a representative picture of the operation of the circuit. The resolution measure obtained from these tests should be compared with the $89.3ps/\sqrt{12} = 25.8ps$ RMS resolution obtainable from an ideal converter with the same bin size (thus only affected by the quantization error).

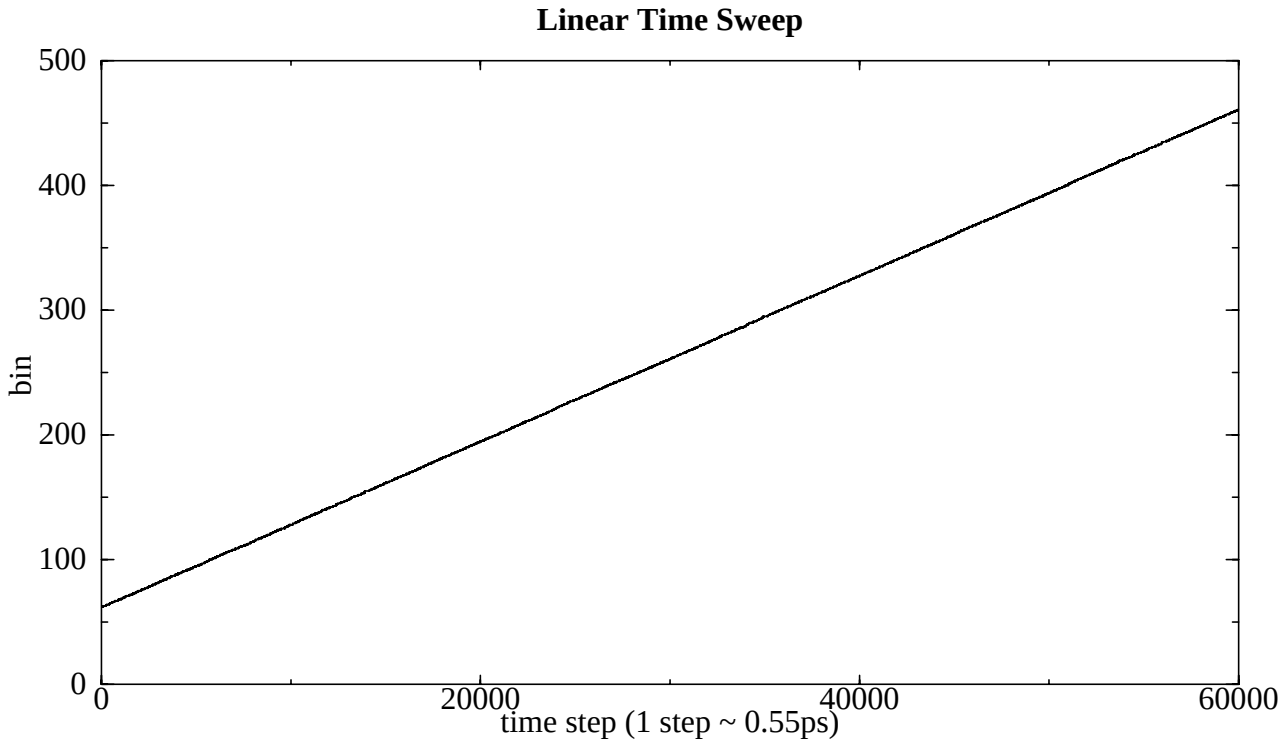

Fig. 8. Linear Time Sweep across three clock transitions.

The Conversion error is then the difference between this curve and the ideal conversion curve, as shown in *Fig. 9*. This plot takes into account all noise sources that affect the converter, including sources of random nature, such as noise and jitter (from the reference clock and internal), and systematic errors, such as non-linearities.

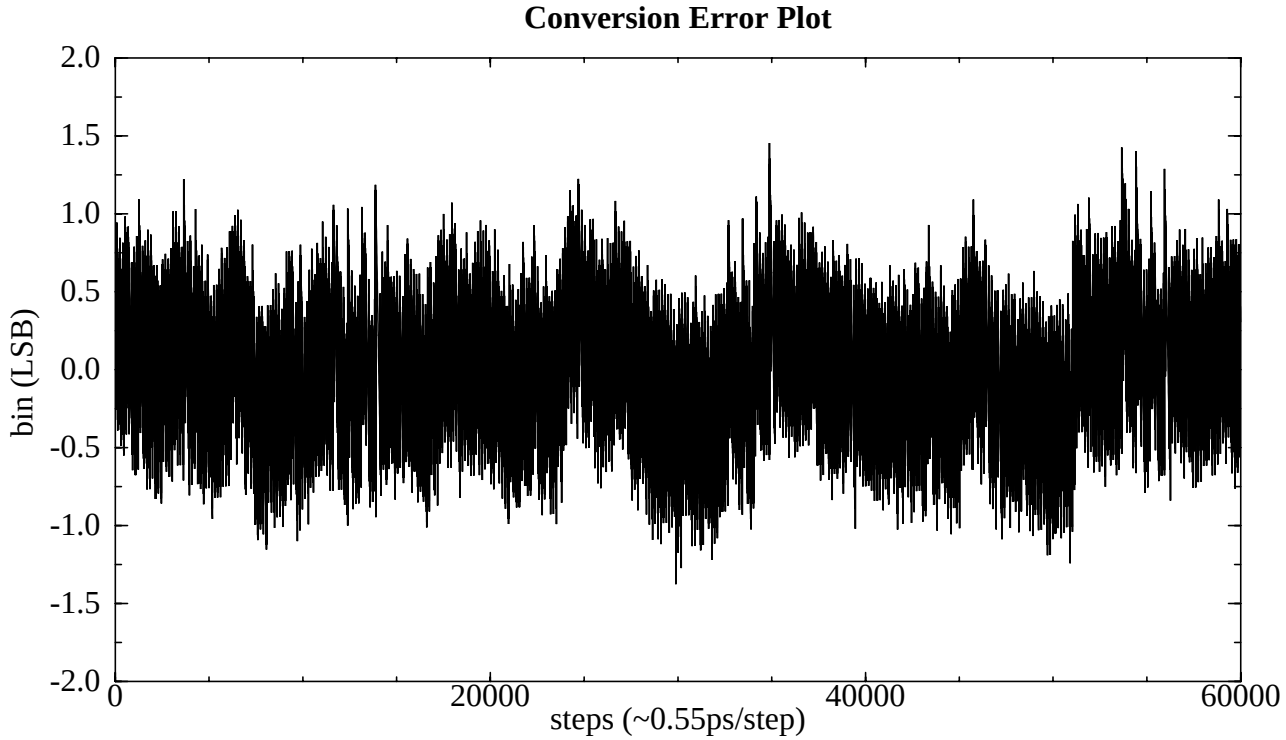

Fig. 9. Conversion Error Plot, includes three clock transitions.

This test is very sensitive to the test setup used. Any non-linearity of the test-setup, will show up in the result as an error and the separation of errors in the converter from errors in the test setup may not be trivial. In this case the test is performed using a mechanical coaxial phase shifter, driven by a step motor. This apparatus is very linear, but has a very short dynamic range. The alignment procedure used to extend the range into meaningful values may result in some small misalignment that will accumulate and be included in the converter error results.

The histogram obtained from this plot has a σ of 0.38 LSB (see *Fig. 10*) which corresponds to an RMS Resolution of 34.3 ps.

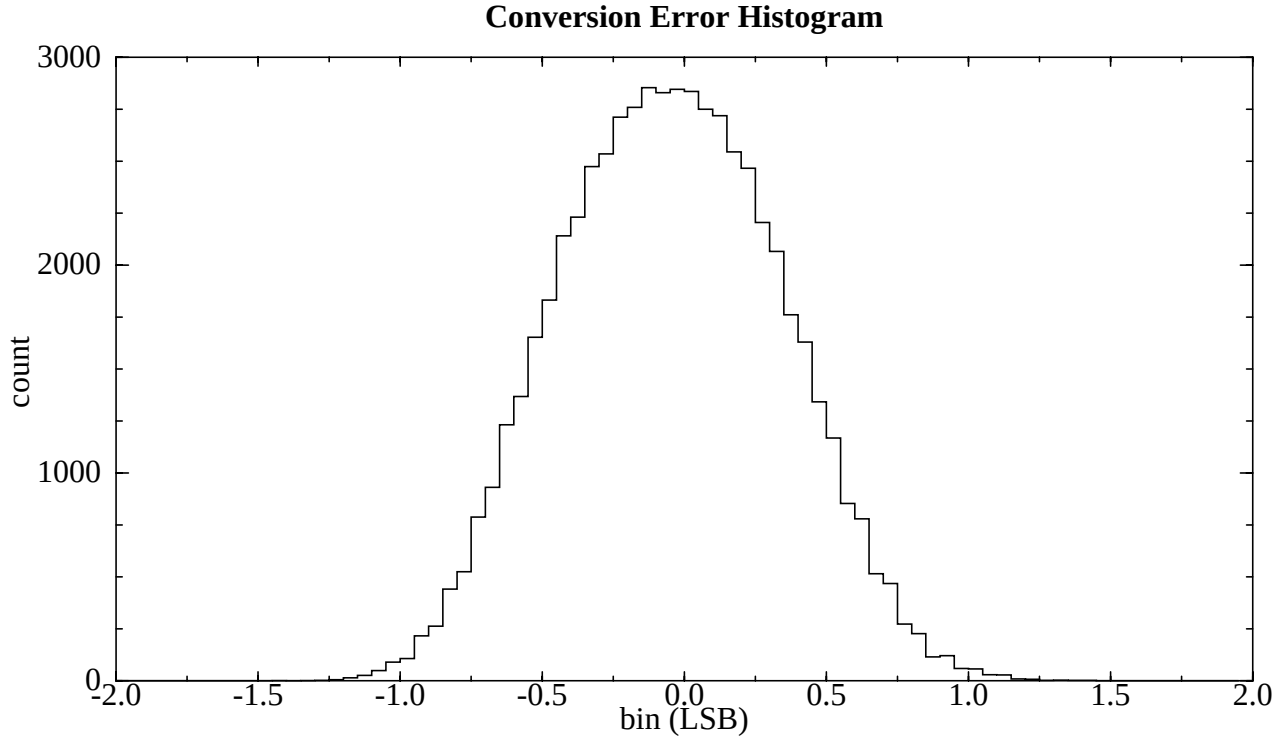

Fig. 10. Conversion Error Histogram, σ= 0.38 LSB (RMS resolution).

The tails of the distribution extend less than 1.5 LSB. This histogram is a picture of the error of the TDC, since it includes all the effects that can influence the conversion (and also non-linearities of the test setup), it's standard deviation is defined as the RMS resolution of the TDC and is 34.3 ps.

The user should understand that this resolution is obtained when the measurement is performed relative to the reference clock, in a common Start (or common Stop) mode. If the measurement is to be obtained relative to another random signal, then both Start and Stop signals have to be measured and their difference calculated. In this case the resolution of the final measurement is limited by the two measurements performed to $\sqrt{34.3^2 + 34.3^2} = 48.5$ ps.

Another factor that may influence the final system resolution is the quality of the reference clock provided. The results presented here were obtained using a very stable ECL crystal oscillator, with very small jitter. DLL loops are, by their nature, incapable of cleaning up the clock to which they lock. Therefore any jitter in the reference clock will proportionally degrade the system resolution.

The plot in *Fig. 11* results from a coarse time sweep (~1ns steps) across the full dynamic range of the converter, proving the correct functionality of the dynamic range extension circuitry:

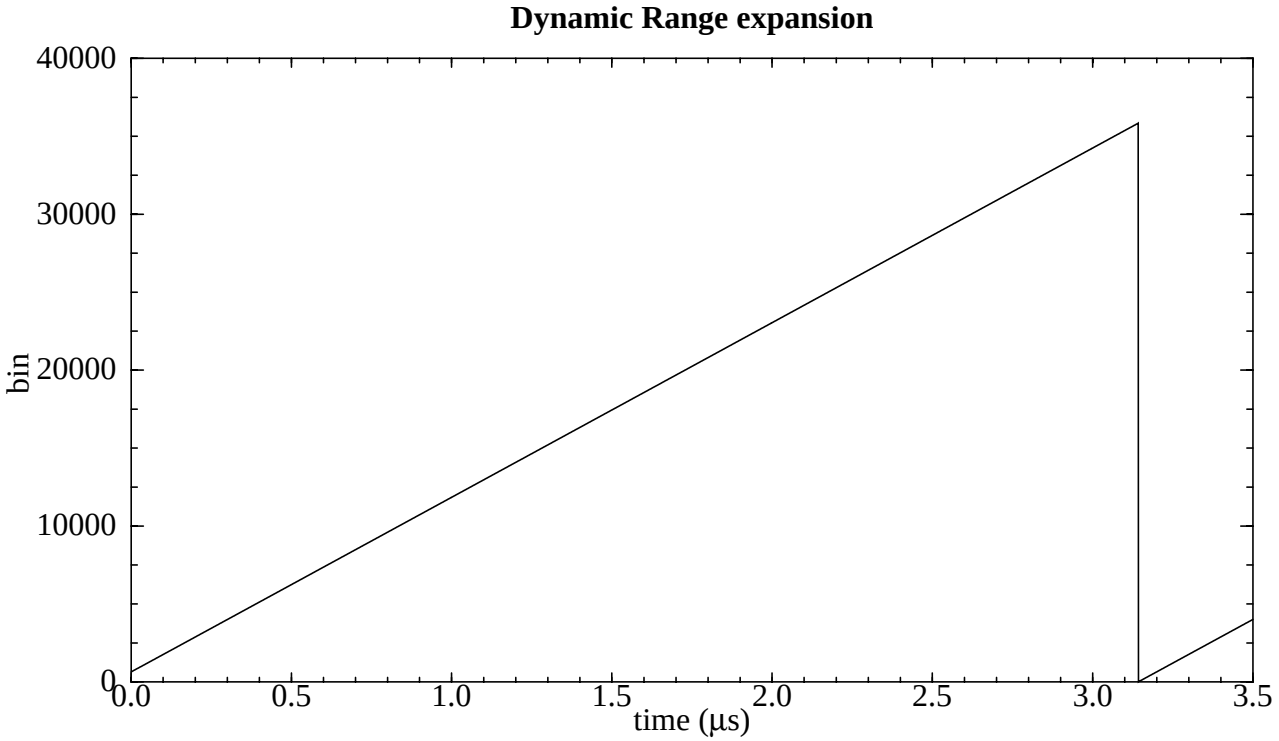

Fig. 11. Dynamic Range expansion up to 3.2 μs.

It is clear that the expansion is performed correctly, the step in the plot corresponding to the end of the dynamic range and the corresponding overflow of the coarse time counter.

3.2.3 Other timing characteristics.

A summary of the most important timing characteristics of the TDC prototype is given next:

resolution:	34.3 ps (RMS).
dynamic range:	3.2 μs.
reference clock frequency:	80 MHz.
bin size:	89.3 ps.
DNL (σ):	12.8 ps.
INL (σ):	15.3 ps.
crosstalk:	$< \pm 2$ bin.

3.3 The read-out and programming interfaces.

Two ways of reading-out the time measurements acquired in the four channels of the chip were implemented. Via the parallel port, data stored in the read-out FIFO can be read as a 20 bit long word. A serial read-out port, mainly used for testing and debugging, is available to read-out data before being binary encoded.

3.3.1 The parallel (TDC) port.

This port accesses the derandomizing FIFO included in the chip. Data are stored in this FIFO as 20 bit words per time measurement. These words are made of the binary encoded fine time measurement (1 byte), the selected coarse time measurements (1 byte), channel identification (2 bits) and two error flags indicating if the encoding algorithm found two or more rising edge transitions (1 bit) or didn't find any transition (1bit) in the fine time word.

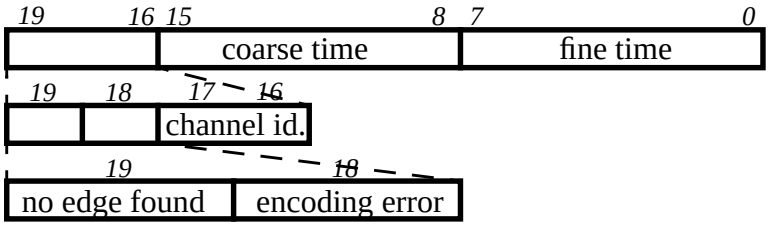

Fig. 12. Read-out word description.

The read-out interface is made of 2 asynchronous control signals: *readout_empty* and *get_data*. The *readout_empty* signal indicates if there is any data available to be read-out. In this case, the external read-out controller should start the read-out procedure by asserting the *get_data* signal. The rising edge of this signal forces the FIFO to make available the requested measurement word. An output enable (*oe_b*) is available to simplify the connection to a shared bus. The read-out procedure is schematically shown in *Fig. 13*.

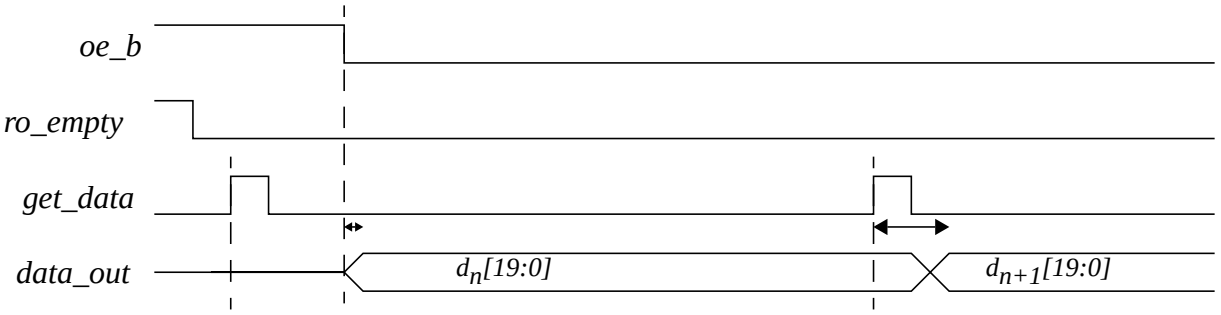

Fig. 13. The parallel read-out protocol[1].

The simple arbitration scheme implemented in this demonstrator to sort data from channels does not guarantee that measurements will be available to be read-out in temporal order.

The measurement data-path from the hit registers to the FIFO is implemented as a 3 level deep pipeline. Taking into account the time required to synchronize the hit arrival time to the logic clock cycle, a minimum 5 *clk_lg* period latency is spent before the measurement data is available in the output port. If the read-out arbitration and logic circuitry are already busy serving other channels, the latency will be larger.

3.3.2 The serial (test) port.

This read-out port was implemented having in mind the testing and debugging of the circuit. It is, therefore, very simple but slow. When data is available the *serial_data_valid* signal is asserted. After this, an asynchronous read signal (*serial_rd*) can be used to serially shift out the data (via the *serial_data_out* line). To warn the user of the end of the read-out data, the *serial_data_valid* signal is

1. All signal timings will be given in a different section of the manual.

deasserted one bit before the last one. This scheme is driven by the receiver side and can be used in a completely independent way from the parallel port. Therefore it can be assessed by a slow control port (for example a PC port) for test purposes, while the parallel read-out port is being used at normal speed.

These data are presented in a raw format: all the time bins are read without binary encoding, together with the 4 dummy taps, the 2 coarse time counter results and channel identification:

Bit n^o.:	*Description:*
bit139-0	Fine time measurement, 140 time taps (not encoded).
bit143-140	Initial, dummy, time taps of the four timing DLLs (for chip test only).
bit151-144	Coarse time measurement (counter1 - synchronous to reference clock).
bit159-152	Coarse time measurement (counter2 - synchronous to the opposite phase of the reference clock).
bit161-160	Channel identification. Uniquely identifies the channel where the measurement was obtained.

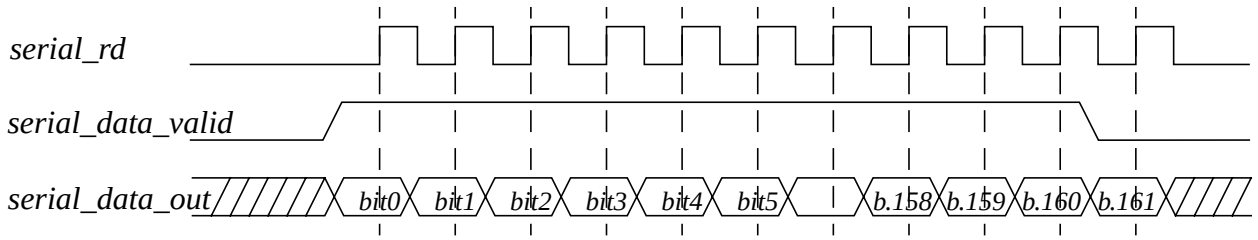

Fig. 14. Parallel read-out wave diagram.

If the programming register is to be read-out, then *sel_strobe_program* should be asserted, forcing the serial port to clear it's register and prepare to read-out this word. Toggling twice the *sel_store_program* signal can also be used force the serial port to reload a new word into it's register. This feature is useful to discard an old measurement that is stored in the serial port register, but has not been read-out. This way a more recent measurement is made available.

3.3.3 The DLL status outputs.

Three multiplexed status outputs are available in order to facilitate the test of the circuit. Using these outputs, one is able to access the most important internal signals that describe the status of each DLL: *Last_cellA_b*, *Last_cellB0_b*, *Last_cellB3_b*, *Analog clock*, *Up_downA_b*, *Up_downB0_b*, *Up_downB3_b*, *Logical zero*, available in each test output:

Name:	*Description:*
Last_cellA_b	Output of the last delay element of the phase shifting DLL (inverted).
Last_cellB0_b	Output of the last delay element of the first timing DLL - #0 (inverted).
Last_cellB3_b	Output of the last delay element of the last timing DLL - #3 (inverted).
Analog_clock	Reference clock after the differential to single-end conversion.
Up_downA_b	Output of the phase shifting DLL's phase detector (inverted).

Up_downB0_b	Output of the first timing DLL's phase detector - #0 (inverted).
Up_downB3_b	Output of the last timing DLL's phase detector - #3 (inverted).
Logical zero	Test output grounded.

The three output multiplexers can be selected independently via the programming word. If these outputs are not to be used they should be stopped by selection of *Logical zero* as the multiplexed output). This way undesirable noise generated by the large output buffers can be avoided.

3.3.4 The programming port.

The programming port is a simple asynchronous serial port used to access all the programmable features. At the rising edge of the *program_wr* signal, the data at the *program_in* input will be shifted into the programming shift-register. To signal the end of the programming word shift in, the *sel_store_program* signal must be asserted at the *program_wr* rising edge corresponding to the last bit being shifted in (bit #50). This way the data present on the shift register will be stored in the program register, completing the programmation cycle.

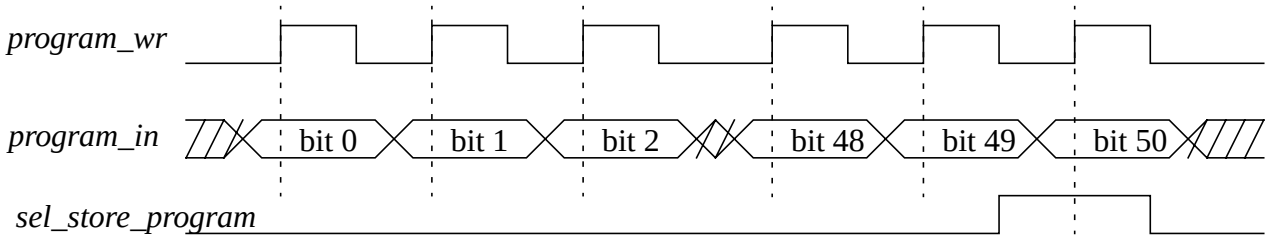

Fig. 15. Timing diagram for the register programming port.

The register functionality is listed bellow:

Register n^o.:	*Name:*	*Description:*
bit4-0	IselA_b	Current level for the charge-pump (phase-shifting DLL).
bit9-5	IselB_b	Current level for the charge-pump (timing DLLs).
bit11-10	SlopeA_b	Gain selection for the assigned offset (phase-shifting DLL).
bit16-12	OffsetA_b	Offset selection (phase-shifting DLL).
bit18-17	SlopeB_b	Gain selection for the assigned offset (timing DLLs).
bit23-19	OffsetB_b	Offset selection (timing DLLs).
bit24	DummyA_b	dummy cells offset control voltage source selection (phase-shifting DLL).
bit25	DummyB_b	dummy cells offset control voltage source selection (timing DLLs).
bit26	InitA_b	Initialization procedure (phase-shifting DLL).
bit27	InitB_b	Initialization procedure (timing DLLs).
bit32-28	ResetDLL_b	Force DLLs to run slow (1 bit per DLL) - for debugging.

bit37-33	SetDLL_b	Force DLLs to run fast (1 bit per DLL) - for debugging.
bit40-38	Test0	Status signal selection for Test 0 output.
bit43-41	Test1	Status signal selection for Test 1 output.
bit46-44	Test2	Status signal selection for Test 2 output.
bit50-47	Channel enab.	Individual channel enable (channel#3-channel0). These signals are logically ANDed with the master hit enable.

Alterations to the circuit's programmation can be performed while the circuit is running without disturbing its functionality (except in the way pretended by the alteration, of course). This is achieved by using a shift register independent of the program register, this way the process of shifting in the program will not disturb the normal work of the circuit.

In order to allow access to the previously programmed registers for debugging purposes, these registers are available in the serial (test) port. To access this information the *sel_store_program* selection signal should be activated; the serial port will then contain the programming register data, that can be read-out the usual way. The user should be careful to avoid using simultaneously the programming input port and the serial output port, because of the different significance of the *sel_store_program* signal to both ports.

At system level it may be required to load different circuits with different programs (for example, if some channels are to be disabled). To facilitate this task, avoiding the board area waist of having to distribute independent programmation signals to different chips, a *program_out* pin, where the end of the program shift register is available, to enable the daisy-chaining of several circuits into the same programmation interface.

The necessary program word should be provided to the user, together with the chip. It is obtained when at circuit's testing characterization level.

3.4 The differential inputs.

To obtain the high resolution required from this circuit, special care has been taken to ensure that the handling of time critical paths, such as the reference clock and the hit signals, does not compromise the overall resolution. Any jitter in the reference clock will couple directly into the time measurements, therefore slow and noise sensitive TTL (or CMOS) signalling levels should be avoided.

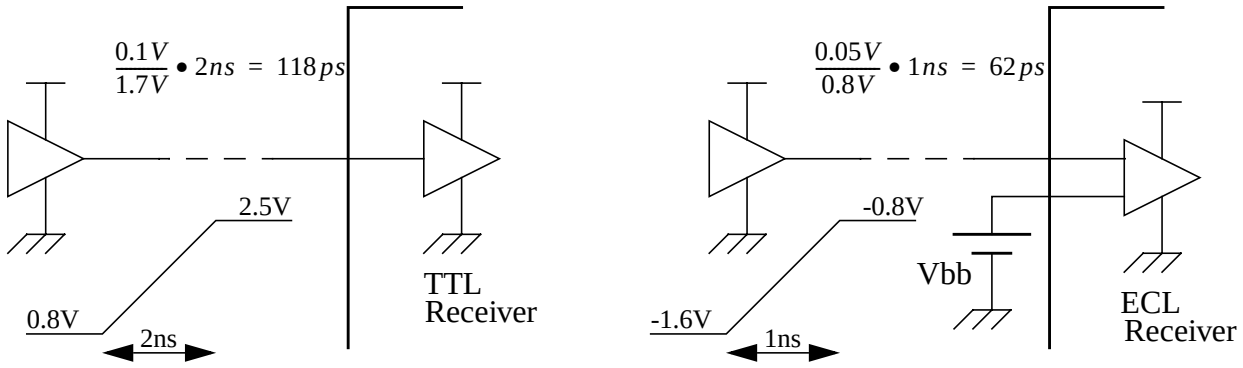

Fig. 16. Noise in single ended receivers.

To illustrate the noise sensibility of the timing accuracy in this signals, an example is given for "high speed" TTL path (see *Fig. 16*), where a Power/ground bounce of ~100mV is applied, and for

an single ended driven ECL path, with a ~50mV bounce. If we consider a 2ns as an, optimistic, approximation for the rise time in a TTL line (1ns for a modern ECL family) then, simple calculations show that for the respective typical logical levels, the noise induced variations in the rising time (~120ps and ~60ps, respectively) are unacceptable.

In order to preserve the integrity of the time information arriving to the circuit, differential lines must be used. In this chip, differential Pseudo-ECL([1]) receivers with a very good common mode rejection ratio, are used. These are capable of discriminating differential signals with a common mode ranging from 1.6V to 4V and a voltage swing as low as 200mV, without noticeable degradation of the common mode rejection ratio.

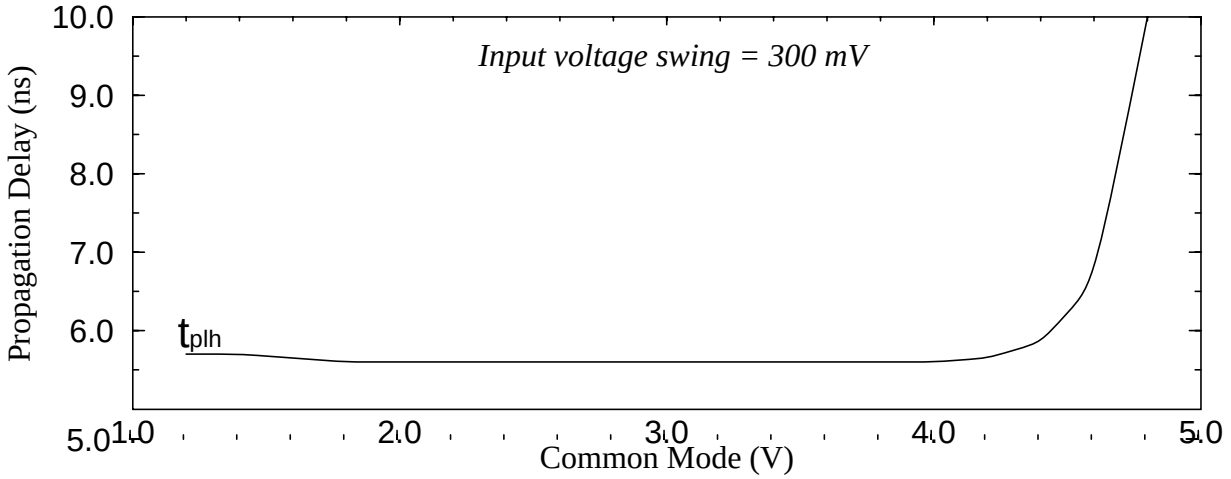

Fig. 17. Differential PECL receiver Propagation delay vs. Common mode voltage behaviour.

Because the critical timing information carried by the signals is mostly contained in the higher frequencies, it is advisable to treat the lines carrying these signals as transmission lines (especially as the length of the lines is bigger than a few centimetres). This means that coaxial cables or micro-strip lines should be used, together with the correct termination. These PECL receivers, like most commercial ECL receivers, do not include termination resistors at their inputs. The user should take care of correctly terminating the lines in order to obtain the best timing characteristics from this converter.

3.5 The technology and circuit implementation.

This demonstrator is implemented in the 0.7µm n-well CMOS technology from ES2. The design includes standard cells and a memory macro cell. The time critical core of the circuit (the array of DLLs) and the differential receivers are full custom designed cells.

The resulting demonstrator chip occupies 23 mm^2 of silicon and is packaged in a 68 pin PLCC ceramic package.

1. Pseudo-ECL, or PECL signalling levels is nothing more than the common ECL standard shifted up to positive supply levels, which enables the use of both fast ECL and slow TTL or other levels in a single supply circuit. If instead of the usual 0V to -5.2V, positive 5V to 0V supply levels are used, the resulting (P)ECL common mode level is 3.8V

3.5.1 Summary of the TDC's functional characteristics.

The functional, and other characteristics are summarized in the next table:

n^{o}. of channels:	4.
power consumption:	800 mW.
reference clock logic levels:	differential PECL.
hit inputs logic levels:	differential PECL.
buffer depth (Read-out FIFO):	32 word.
buffer depth (hit registers):	2 word/channel.
logic clock frequency:	40 MHz (max).
output word length:	20 bits.
programming port:	serial.
silicon area:	23 mm^{2}.
package	68 pins PLCC.

3.6 The pin-out definition.

(I input -- O output -- D/I differential input -- P/G power or ground)

Pin name:	*Type:*	*N^{o}.:*	*Description:*
TIMING INPUTS (15)			
Clk_ref	D/I	60	Reference clock, Rising edge used as time reference.
Clk_ref_b	D/I	59	Reference clock (inverted), Falling edge used as time reference.
Hit_trigg[0]	D/I	62	Hit input, for channel 0.
Hit_trigg_b[0]	D/I	61	Hit input (inverted), for channel 0.
Hit_trigg[1]	D/I	64	Hit input, for channel 1.
Hit_trigg_b[1]	D/I	63	Hit input (inverted), for channel 1.
Hit_trigg[2]	D/I	66	Hit input, for channel 2.
Hit_trigg_b[2]	D/I	65	Hit input (inverted), for channel 2.
Hit_trigg[3]	D/I	68	Hit input, for channel 3.
Hit_trigg_b[3]	D/I	67	Hit input (inverted), for channel 3.
Hit_enable	I	3	Enable the acquisition of hit triggers (all channels) (active high).
Reset_coarse	I	55	Coarse counter reset. resets the coarse time counter (active high). Should be synchronous to the reference clock (clk_ref)
Reset_dll	I	56	Initializes the DLLs (active high).

Clk_lg	I	8	Read-out Clock.
Reset_buffers	I	9	Buffer reset clears all buffers in the circuit and re-initializes the logic control (active high).

STATUS OUTPUTS (7)

Lock_dllA	O	49	Asserted when phase shifting DLL is in locking condition.
Lock_dllB3	O	48	Asserted when last DLL of the array is in locking condition.
Test_out[0]	O	50	Multiplexed outputs for internal DLL status signals: 0: last_cellA_b 1: last_cellB0_b 2: last_cellB3_b 3: analog clock 4: up_downA_b 5: up_downB0_b 6: up_downB3_b 7: logical zero the selection of the internal signal is performed independently for each output in the program word.
Test_out[1]	O	51	Multiplexed outputs for internal DLL status signals:
Test_out[2]	O	52	Multiplexed outputs for internal DLL status signals:
Readout_empty	O	15	Asserted if output FIFO is empty.

PROGRAMMING INPUT PORT (2)

Program-in	I	47	Serial programmation input. [4:0] charge pump current level DLL A (active low) [9:5] charge pump current level DLL B (active low) [11:10] range slope selection DLL A (active low) [16:12] range offset selection DLL A (active low) [18:17] range slope selection DLL B (active low) [23:19] range offset selection DLL B (active low) [24] use same control voltage in dummy cells DLL A (active low) [25] use same control voltage in dummy cells DLL B (active low) [26] use initialization procedure DLL A (active low)

			[27] use initialization procedure DLL B (active low) [28] reset DLL A (active low) [29] reset DLL B0 (active low) [30] reset DLL B1 (active low) [31] reset DLL B2 (active low) [32] reset DLL B3 (active low) [33] set DLL A (active low) [34] set DLL B0 (active low) [35] set DLL B1 (active low) [36] set DLL B2 (active low) [37] set DLL B3 (active low) [40:38] test output 0 selection [43:41] test output 1 selection [46:44] test output 2 selection [50:45] channel enable (channel# 3: channel# 0).
Program_wr	I	46	Asynchronous write strobe to the programming register.
Program_out	O	45	Serial output of the serial register. Allows for daisy chaining of several circuits programmation.
Sel_store_program	I	12	If asserted at positive edge of “Program_wr” program word that has been shifted in is activated. This signal is also used to make the serial interface shift out the contents of the program register.

PARALEL READ-OUT PORT (22)

Get_data	I	16	Reads data from the output FIFO. Should only be asserted when the FIFO is not empty.
Data_out[0]	O	17	Bit 0 of the read-out word. The time measurement word is: [7:0] Binary encoded vernier word. [15:8] Coarse time tag. [17:16] Channel id. [18] Decoder error: more than one transition. [19] Decoder error: no transition.
Data_out[1]	O	18	Bit 1 of the read-out word.
Data_out[2]	O	19	Bit 2 of the read-out word.

Data_out[3]	O	20	Bit 3 of the read-out word.
Data_out[4]	O	21	Bit 4 of the read-out word.
Data_out[5]	O	22	Bit 5 of the read-out word.
Data_out[6]	O	23	Bit 6 of the read-out word.
Data_out[7]	O	26	Bit 7 of the read-out word.
Data_out[8]	O	27	Bit 8 of the read-out word.
Data_out[9]	O	28	Bit 9 of the read-out word.
Data_out[10]	O	29	Bit 10 of the read-out word.
Data_out[11]	O	30	Bit 11 of the read-out word.
Data_out[12]	O	31	Bit 12 of the read-out word.
Data_out[13]	O	34	Bit 13 of the read-out word.
Data_out[14]	O	35	Bit 14 of the read-out word.
Data_out[15]	O	36	Bit 15 of the read-out word.
Data_out[16]	O	37	Bit 16 of the read-out word.
Data_out[17]	O	42	Bit 17 of the read-out word.
Data_out[18]	O	43	Bit 18 of the read-out word.
Data_out[19]	O	44	Bit 19 of the read-out word.
Oe_b	I	11	Output enable.

SERIAL READ-OUT PORT (4)

Serial_rd	I	10	Asynchronous read strobe to the serial port.
Serial_data_out	O	14	Serial output. [139:0] fine time word. [143:140] dummy time taps. [151:144] coarse time tag 1. [159:152] coarse time tag 2. [161:160] channel id. or [50:0] program word.
Serial_data_valid	O	13	Asserted if serial shifter has data to be shifted out. It is deasseted one bit before the end of the shift out of the word is complete.

POWER and GROUND (18)

PWR(pads)	P/G	6, 24, 38, 53
GND(pads)	P/G	7, 25, 39, 54

PWR(core)	P/G	4, 32
GND(core)	P/G	5, 33
PWR(analog)	P/G	1, 41, 57
GND(analog)	P/G	2, 40, 58

TOTAL 68 used pins

3.6.1 Pin-out electrical characteristics.

Power Supply (pads)	5V
Power Supply(core)	5V
Power Supply(analog)	5V
Output signals	TTL compatible
Input signals	TTL compatible, except:
Differential Timing Inputs (*clk_a* and *hit_trig[3:0]*)	PECL[1] compatible

3.7 Signal Timing Characteristic.

Signal timings given in this section result from circuit simulations covering worst case environment and process parameters. They though only are though not measured in the prototype, thus can only be considered indicative.

3.7.1 The Read-out interface.

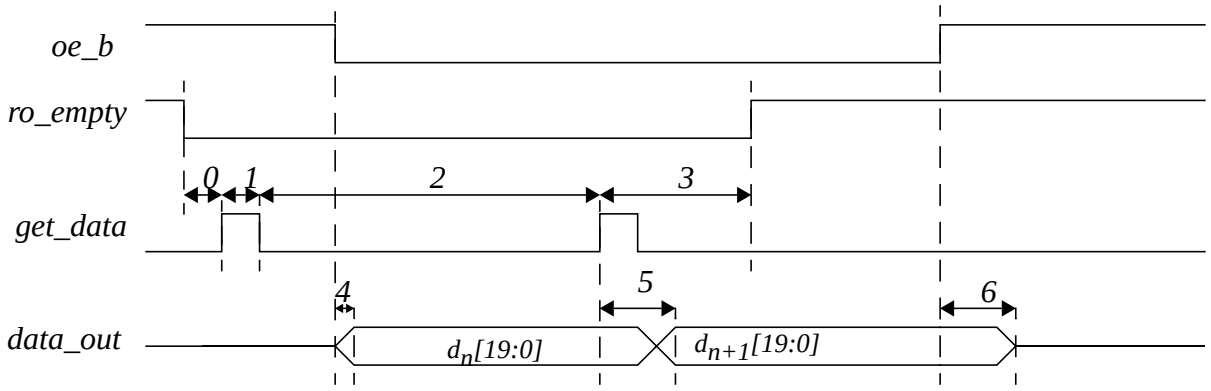

Fig. 18. Read-out interface timing diagram.

TABLE 1. Read-out timing.

#	Name	Description	Max.	Typ.	Min.
0	Tdget	get_data delay			0ns
1	Twget	get_data width			2.5ns
2	Thget	get_data hold			2.6ns
3	Tdepty	ro_empty delay	16.5ns		5.3ns
4	Tddataoe	tri-state to data delay	8.8ns		1.9ns

1. See Footnote 1., on page 17 for PECL definition.

TABLE 1. Read-out timing.

#	Name	Description	Max.	Typ.	Min.
0	Tdget	get_data delay			0ns
5	Tddataget	data delay	15.4ns		3.7ns
6	Tdhizoe	data to tri-state delay	8.8ns		1.9ns

3.7.2 Serial Test Port.

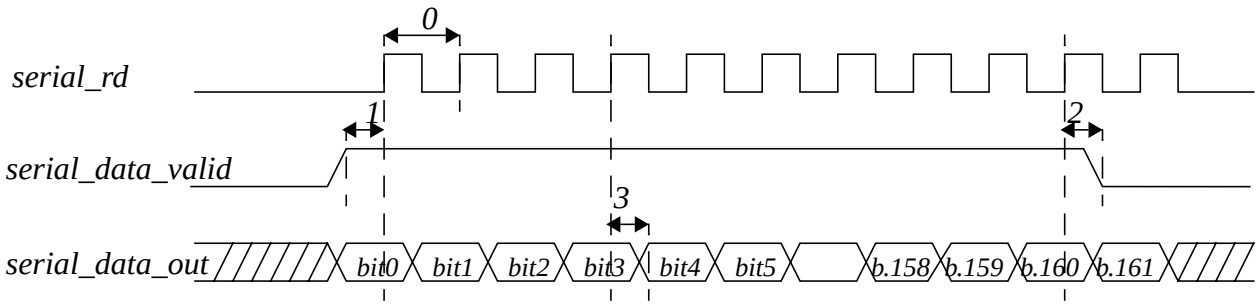

Fig. 19. Serial Test Port timing diagram.

TABLE 2. Serial Test Port timing

#	Name	Description	Max.	Typ.	Min.
0	Tprd	serial_rd period			
1	Tdrd	serial_rd delay			0ns
2	Thval	serial_data_valid hold			
3	Thdata	serial_data_out hold			

3.7.3 Serial Program Interface.

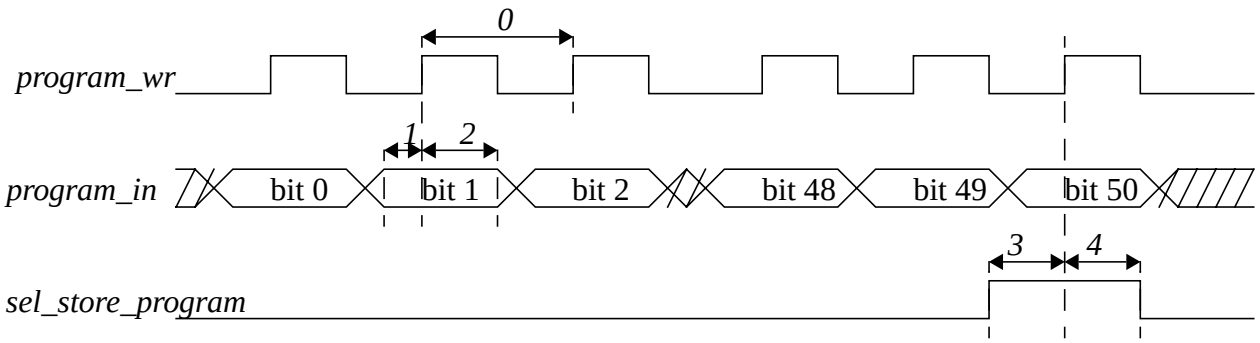

Fig. 20. Serial Program Interface timing diagram.

TABLE 3. Serial Program Interface timing.

#	Name	Description	Max.	Typ.	Min.
0	Tpwr	program_wr period			
1	Tsprg	program_in setup			
2	Thprg	program_in hold			
3	Tsstr	sel_store_program setup			
4	Thstr	sel_store_program hold			

3.7.4 Hit/Reset signals.

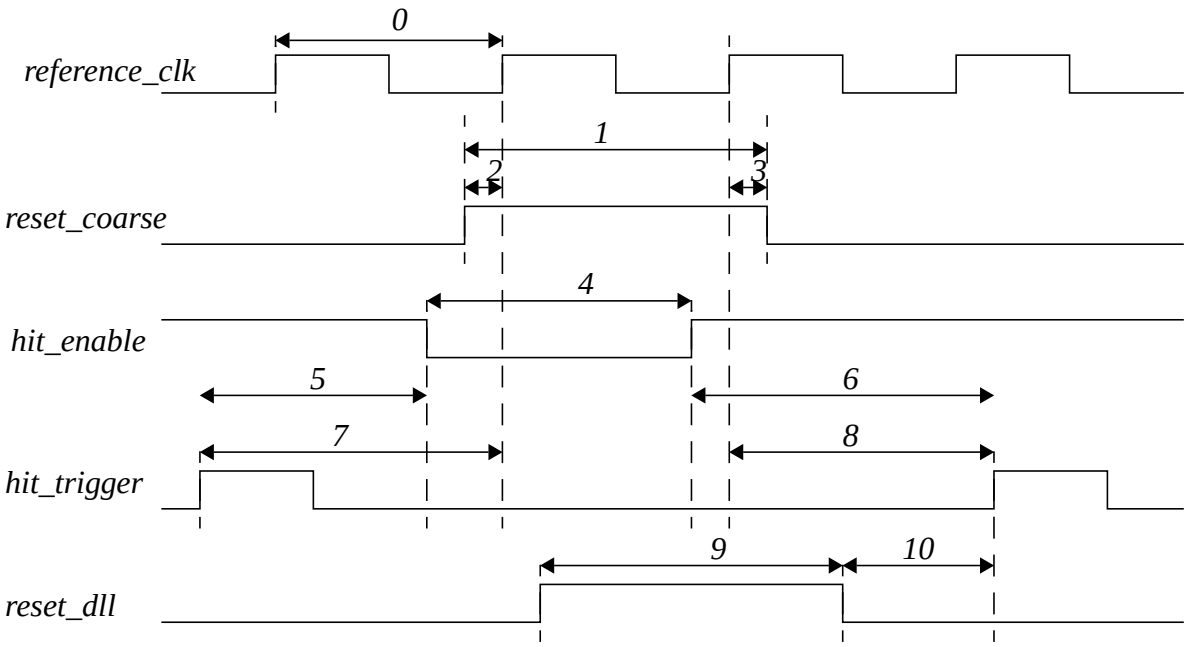

Fig. 21. Hit/Reset signals timing diagrams.

TABLE 4. Hit/Reset signals timing.

#	Name	Description	Max.	Typ.	Min.
0	Tpref	reference_clk period		12.5n	
1	Twrst	reset_coarse width			#2+#3
2	Tsrst	reset_coarse setup			1ns
3	Thrst	reset_coarse hold			5ns
4	Twenb	hit_enable width			0ns
5	Thenb	hit_enable hold (from last accepted hit)	4ns		
6	Tsenb	hit_enable setup (to next accepted hit)			4ns
7	Tdfrst	accepted hit_trigger delay to first reset clock cycle			0ns
8	Tdlrst	accepted hit_trigger delay from last reset clock cycle			#0
9	Twrstd	reset_dll width			
10	Tlock	lock procedure duration (delay to first correct measurement)	10µs		

3.7.5 Other signals.

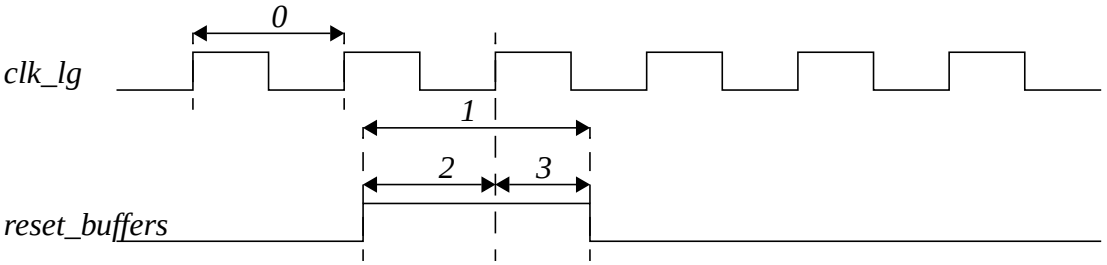

Fig. 22. Other signal's timing diagram.

TABLE 5. Other signal's timing

#	Name	Description	Max.	Typ.	Min.
0	Tclklg	clk_lg period			25ns
1	Twrstb	reset_buffers width			2*#0
2	Tsrstb	reset_buffers setup			0ns
3	Thrstb	reset_buffers hold			0ns
	Tlrstb	reset_buffers latency (after deactivation)	2*#0+ 5ns		

3.8 The package layout.

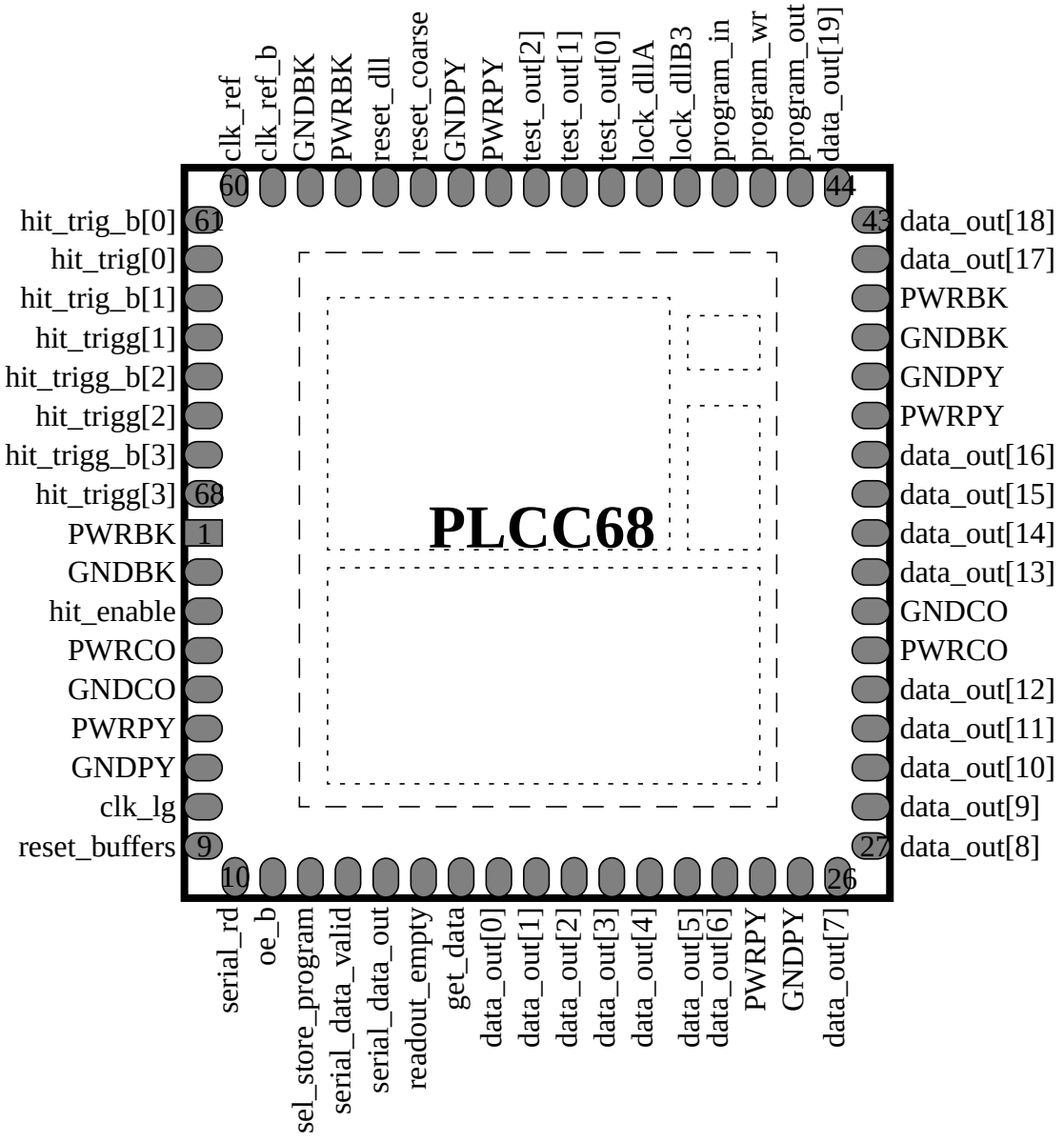

Fig. 23. High Resolution TDC, pin layout.

Appendix A. The initialization procedure.

The initialization of the TDC is the process by which the DLLs of the array are locked into the correct frequency of operation.

In this circuit the locking range was divided into several partially overlapping ranges, guaranteeing that similar matching characteristics are obtained regardless of ambient conditions and a smaller sensitivity to control voltage noise (see [4]). This characteristic makes the initialization procedure not trivial: On start-up the circuit must choose the locking range suitable for the environment where it is.

The limits of each range are determined by a different delay offset. The slope of the delay curve is defined by the gain of the delay cells, which is maximum in the middle of each range (*Fig. 24*). It sets the range length and therefore the amount of variations in ambient conditions that the DLLs are able to track. To increase the flexibility of the circuit, three different slope possibilities, corresponding to different tracking coverage are implemented. Slope selection is done in accordance with the expected environment changes.

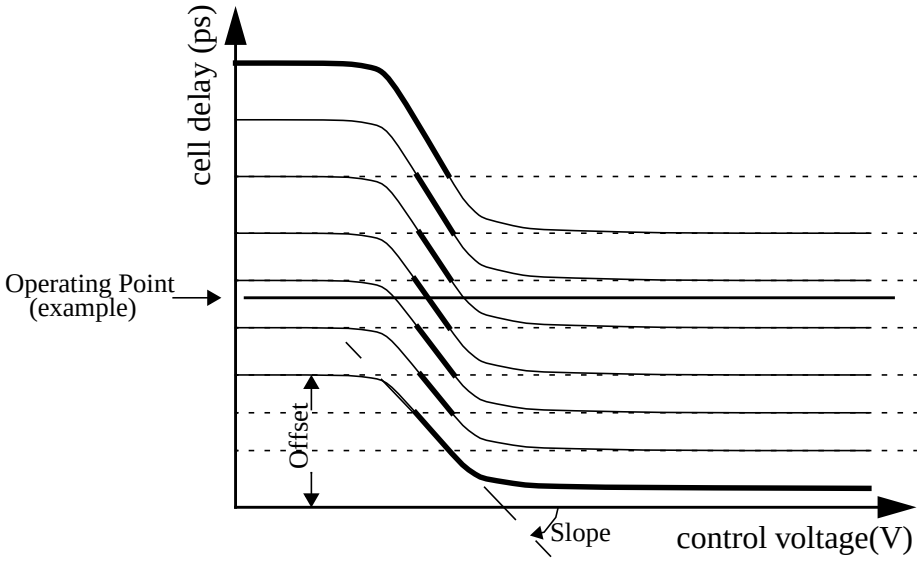

Fig. 24. Schematic representation of the range partition (for simplicity a single Slope possibility per Offset is shown).

The initialization procedure will be internally implemented in a final version of the circuit. In the present demonstrator, though, the converter should be characterized by the user and the final range selection down-loaded via the programming interface. For the typical ambient conditions, where temperature is not expected to change more than ~15^{o} C the best solution has been obtained and an example of the program to be down-loaded is shown in the end of the Appendix.

A.1 The range selection algorithm.

A.1.1 Slope selection.

For each Offset available, there are three range length possibilities (controlled by the *SlopeX_b* bit in the program word, where *X* is the DLL in question). The smaller slopes are able to track up to 15^{0}C and 300 mV supply variations in any range, medium slopes correspond to....^{0}C and... mV and large slopes to...^{0}C and... mV. In normal operating conditions, the smallest Slope should be selected. If the user expects more severe variations of ambient conditions, then the Slope selection should be performed accordingly.

A.1.2 Offset selection.

After slope selection has been made, offset selection is achieved using the criterion next explained (controlled by the *OffsetX_b* bits).

The range sub-division was implemented such that in any conditions, lock is achieved in at least three different Offsets. If the second smallest Offset is selected, then it is guaranteed that locking was achieved around the centre of the locking range, leaving a good margin on both sides of the operating point for tracking temperature and supply variations. *Fig. 24* schematically depicts this situation.

The user should try to obtain lock for every Offset in a sequential order, starting from the biggest offset, then moving towards smaller offsets. This way the correct operating range can easily be obtained. If, in corner conditions, lock can only be obtained for one or two Offsets, then the smallest Offset should be selected.

The obtainment of lock in signalled by the assertion of the outputs *Lock_dllA* (for the phase shifting DLL) and *Lock_dllB3* (for the timing DLLs). Also the signals *up_downA_b*, *up_downB0_b* and *up_downB3_b*, available on the *Test_out[2:0]* outputs can be used to verify the status of the DLLs since they show the operation of the respective Phase Detectors.

An important exception to the rule may occur in the “slower and short” ranges (higher offsets and smaller slopes). If their lower limit is bigger than the expected locking delay, the DLL may try to lock at half the frequency, resulting in a false lock. This situation is though easily identified by observing that some ranges are incompatible with the others (do not overlap), leaving non-locking ranges in the middle of a sequence of locking ranges. These ranges should, therefore, be discarded from the selection algorithm (should be considered as not achieving lock). The selected range must thus be selected among the ranges with smaller Offsets.

The following tables show the programming bits and their corresponding description (X is the DLL being programmed: A for the phase shifting DLL and B for the timing DLLs).

TABLE 6. Range offset selection programming bits.

OffsetX_b (5 bits)	Description
00000	smallest offset (faster range)
10000	
11000	
11100	
11110	
11111	biggest offset (slower range)

TABLE 7. Range slope selection programming bits.

SlopeX_b (2 bits)	Description
11	shorter slope (smaller tracking range)
10	
00	longer slope (larger tracking range)

A.1.3 Current level selection.

Another degree of freedom is introduced by allowing for the selection of the Charge Pump current level. It should be selected as low as possible to ensure correct functionality of the closed loop

control with a minimum contribution of jitter from the “bang-bang” operation of the charge pump. Typically the minimum current level is enough.

Current levels are controlled by the *IselX_b* bits on the programming word.

TABLE 8. Charge-pump current level selection programming bits.

IselX_b (5 bits)	**Description**
11111	no current (don’t use)
11110	smallest current
11100	
11000	
10000	
00000	biggest current

A.2 Programmation example.

This example shows typical values for the programming bits, in normal operation.

Bit #	*Name*	*Value*	*Description*
bit4-0	IselA_b	11110	minimum current level.
bit9-5	IselB_b	11110	minimum current level.
bit11-10	SlopeA_b	10	medium range length.
bit16-12	OffsetA_b	11111	maximum range offset.
bit18-17	SlopeB_b	00	maximum range length.
bit23-19	OffsetB_b	10000	second smallest offset.
bit24	DummyA_b	0	Separate control.
bit25	DummyB_b	0	Separate control.
bit26	InitA_b	0	Use internal initialization.
bit27	InitB_b	0	Use internal initialization.
bit32-28	ResetDLL_b	11111	Disable forced reset.
bit37-33	SetDLL_b	11111	Disable forced set.
bit40-38	Test0	111	Disable output.
bit43-41	Test1	111	Disable output.
bit46-44	Test2	111	Disable output.
bit50-47	Channel enab.	1111	Enable all channels.

Appendix B. References.

[1] ALICE Technical Proposal, CERN/LHCC 95-71, Dec. 1995.

[2] T. Rahkonen et all, The use of Stabilized CMOS Delay Lines for the digitization of short Time Intervals, IEEE J. of Solid-State Circuits, Vol. 28, No. 8, pp. 887-894, Aug. 1993.

[3] J. Christiansen et all, An Integrated CMOS 0.15 ns Digital Timing Generator for TDC's and Clock Distribution Systems, IEEE Trans. on Nuclear Science, Vol. 42, No. 4, pp. 753-757, Aug. 1995.

[4] M. Mota, High resolution time to digital converter reference manual, CERN/ECP-MIC internal note.

[5] M. Mota, Measurements of the four channel high resolution TDC prototype test, talk presented at the ALICE technical board, 14/5/97.