timetagger 2.0: reference documentation

overview, theory, control reference, registers, test records

Rok Zitko, Jozef Stefan Institute

2026-11-12 (commit 6f77098)

Contents

timetagger: a ten-channel carry-chain time tagger **2**
- Before connecting equipment . . . 3
- Quick start (no Vivado needed) . . . 3
- Presets . . . 3
- Specifications . . . 4
- Hardware setup . . . 5
- Documentation . . . 5
- Changing the design . . . 5
- Source layout . . . 5
- Verification . . . 6
- Author . . . 6

timetagger: theory of operation **6**
- Delay lines . . . 7
- Input stage . . . 7
- Encoding . . . 8
- Channels . . . 8
- Conditional filter . . . 8
- Markers . . . 8
- Merging . . . 9
- Coincidences . . . 9
- Calibration lookup and time . . . 10
- Code-density calibration . . . 11
- Delays to the connectors . . . 11
- Acquisitions and records . . . 12
- Ring and stream . . . 12
- References and PPS . . . 13
- Test signals . . . 13
- Linearity and precision self-tests . . . 13
- Error budget (measured values from VERIFICATION.md; the others marked) . . . 13
- Prototype findings (0.1, 2026-10-27, die at 67 °C) . . . 14

timetagger: settings, Python methods, TCP commands, the records stream **14**
- The configuration . . . 14
- Python methods (on the board: `pynq.overlays.timetagger.DRIVER`; on a PC: the client) . . . 16
- TCP commands . . . 17
- The records stream . . . 17
- Records . . . 18

timetagger 2.0 registers **18**
- Control and acquisition . . . 18
- Sources . . . 19
- Test signals, dither, generator . . . 19
- Calibration and the 10 MHz meter . . . 19
- Ring writer and merger . . . 20
- Coincidence engine . . . 20
- The ring in DDR . . . 21

timetagger verification records **21**
2026-11-05 (evening): the coincidence engine, its gate and the conditional filter on awg's patterns 21
2026-11-05: awg on a second board (channels, markers, sorting by markers) 21
2026-10-31: the user's manual, session edges (build 0x9b2ec69e) 22
2026-10-31: the user's manual, sessions gpsdo and temp (build 0x9b2ec69e) 22
2026-10-30: the user's manual, session base (build 0x9b2ec69e) 23
2026-10-30: verify_build (build 0x9b2ec69e) 23
2026-10-30: version 2.0 (build 0x9b2ec69e, the current one) 23
2026-10-30: 2.0, the coincidence engine and the conditional filter (build 0x993102ce) 23
2026-10-29: 2.0 step 1, the markers (build 0xfd6932c8) 25
2026-10-28: verify_build (build 0x2a8ff3d5) 25
2026-10-28: connector delays, polarity offsets, the ring oscillator (build 0x2a8ff3d5) 25
2026-10-28: 1.0 on the board (build 0xf47cf07e) 26
2026-10-28: verify_build (build 0xf09c5aa0) 26
2026-10-28: ttcore (the host library) 26
2026-10-28: the ring pump (build 0xf09c5aa0) 27
2026-10-28: 1.0 on the board (build 0xf09c5aa0) 27
2026-10-28: 1.0 offline 28
2026-10-27: 0.1 prototype on the board (build 0x152a286a) 28
2026-10-27: 0.1 prototype, offline 29

timetagger: a ten-channel carry-chain time tagger

timetagger turns the Red Pitaya into a **ten-channel time tagger**: every edge on its inputs gets a timestamp with **picosecond resolution**, and the timestamps stream to a PC, where programs build correlation histograms (g2), lifetime histograms (TCSPC), coincidence counts, rates and stability analyses from them. It has

- **ten channels**: DIO0_P..DIO7_P on the E1 extension connector (3.3 V logic) and the two pairs of the **SATA S2** connector (1.8 V differential);
- per channel a **tapped delay line** in the FPGA's carry chain: 80 cells, 160 taps of about 30 ps, sampled at 250 MHz; timestamps in ticks of 4 ns / 4096 = **0.98 ps**, 60 bits (13 days);
- per channel: **rising, falling or both edges**, a **toggle input** (one transition per edge, whatever the pulse width), **dead time** (up to 262 µs), a **divider** (every N-th event, e.g. a laser's 80 MHz sync), an **offset** (±2.05 µs, for cable delays), counters of events, flagged and lost events;
- a **conditional filter** on channels 1 and 2: a laser's sync is kept only when a detector channel fires within up to 120 ns after it, so an 80 MHz sync becomes as few events as there are photons (for lifetime measurements, TCSPC, on the PC);
- **self-calibration**: an on-chip **ring oscillator** gives each delay line its code-density calibration (a table per channel and polarity) without any wiring, kept per board; while measuring, the channels' own hits refine the tables as the die's temperature changes (live calibration, with a guard against clock-locked signals); **dithered** times (no comb in histograms); the internal test signal measures each line's **polarity offsets** (a toggle input's alternate transitions) and the scale of the **delays to the connectors**, which come from Vivado's timing model with the bitstream: times refer to the connectors;
- **eight markers** on DIO0_N..DIO7_N (3.3 V logic): rising, falling or both edges timed to 1 ns (289 ps rms) beside the channels, for gates, line and frame clocks and triggers; their levels readable at any time;
- a **coincidence engine** in the FPGA: clusters of events within a window (up to 16.4 µs) on any eight of channels 1–10 (and the generator), counted per **pattern** (which channels took part: all of them at once), eight **masks** (these channels at least, or exactly these) with counters, and **virtual channels** 21–28: a record in the stream at each coincidence of a mask. A channel can feed the engine only, not the stream, so coincidences are counted at rates far beyond the stream's; a marker can **gate** it (count only while the marker is high, or low);
- **acquisitions**: start now, at a given sample, or **armed** on a channel's next event (a PPS: t = 0 on its edge); a duration stops it in hardware;
- the events go through a **ring in DDR** (16–96 MB) and a compiled **ring pump** on the board to any number of clients as **binary frames** over TCP (`RECORDS STREAM`, port 5026: 7 Mevents/s sustained, 10 Mevents/s in bursts), with every loss counted where it happens;
- references: a **10 MHz sine on IN2** (an I/Q phase meter, METER records in the stream), a tagged 10 MHz square or PPS on any channel;
- **test signals**: a phase-stepped test signal (17.86 ps steps) that can drive every channel and leave on SATA S1 (a SATA cable loops it into channels 9 and 10), and an **event generator** (random events with exactly delayed pairs) to check the stream and the analysis;

- software: the driver and the service's worker on the board; on the PC a client using only Python's standard library, numpy decoding and reference measurements, and **ttcore**, a C++ library with a Python module that measures as the events come at the stream's full rate: counters, rates, correlations and g2, start-stop histograms (TCSPC), coincidence patterns, recordings (`.ttbin`; tools/ttcore/README.md); **frequency stability** of a tagged clock or PPS (ADEV, MDEV, TDEV, MTIE with ITU-T masks, a PDF report) with the common stability tools (`tools/stability.py`).

timetagger 1.0 was the core instrument; **2.0** (this version) adds the markers, the FPGA coincidence engine with its gate and the conditional filter. Correlations and lifetime histograms are computed on the PC (ttcore): the FPGA has no room left for a correlation engine beside these. The FPGA (a Zynq 7010) is full: the planned **3.0** (discriminators on IN1/IN2 for analog and NIM pulses, pulse widths, outputs) is not built on this board, and may come later on a larger Zynq (e.g. a 7020).

Before connecting equipment

Read the board-wide rules in the root README first.

- **E1 pins are 3.3 V logic and not 5 V tolerant.** A 5 V TTL detector output must be terminated in 50 Ω at the board (which also halves its level); nothing negative (NIM pulses) may reach E1: those belong on IN1/IN2 (3.0).
- **The SATA connectors are not SATA**: they are 1.8 V FPGA pins (bank 34) without termination on the board. Connect only Red Pitaya daisy-chain cables or 1.8 V differential signals (the inputs use the FPGA's internal split termination).
- **SATA S1 drives only when the test signal's output is enabled** (`test.sata_out`); never enable it with anything else driving S1.
- The channels have pull-downs; unconnected channels stay quiet.

Quick start (no Vivado needed)

On the board, with the service running timetagger (`make install-service`):

```
import rpfpga  # must come first
from pynq.overlays.timetagger import DRIVER
tt = DRIVER()
print(tt.calibrate())                # once per board: code density on the ring oscillator
tt.load_preset("detectors")          # channels 1-8 on rising edges
print(tt.counters())                 # rates
```

From a PC (Python's standard library and numpy):

```
import sys; sys.path[:0] = ["common/python", "timetagger/control"]
from timetagger import TimeTaggerClient, RecordStream
from timetagger import analysis
with TimeTaggerClient("192.168.0.149") as tt, RecordStream("192.168.0.149") as stream:
    tt.load_preset("detectors")
    tt.start(duration=1.0)
    decode, chs, ts = None, [], []
    for frame in stream.frames(timeout=2.0):
        head, words = analysis.parse_frame(frame)
        decode = decode or analysis.Decoder(head["time_top"], head["time_hi"])
        c, t = decode(words)                # channel numbers, times in ps
        chs.append(c); ts.append(t)
```

`control/timetagger.ipynb` shows the calibration, the generator and a two-channel precision test; `make test-hw` runs the whole board test.

Presets

Preset	What it does
`off`	all channels off; generator, test signal and ring oscillator off (power-up, SAFE)
`detectors`	channels 1–8 on the rising edges of DIO0_P..DIO7_P
`all`	all ten channels on rising edges
`test`	all channels on the internal test signal: checks without wiring
`generator`	the test generator: 1 Mevent/s, half of them paired 10 ns later

Specifications

Measured on this board unless marked; docs/VERIFICATION.md has the records.

Quantity	Value
Channels	10: 8 × 3.3 V LVCMOS (E1 DIO0_P..DIO7_P), 2 × 1.8 V differential (SATA S2)
Tick	4 ns / 4096 = 0.9765625 ps; 60-bit times (13 days)
Delay line	80 CARRY4 cells, 2 taps each: 128–139 codes per 4 ns period, mean bins 28.8–31.2 ps, largest 68–110 ps, rms quantization 10.9–13.6 ps (die 49 °C, build 0x9b2ec69e; at 70 °C: 28.6–31.0, 62–120, 10.2–13.1 ps)
Precision of one channel	9.1–12.1 ps rms (least squares over the 45 pairs of channels on the test signal at one phase, no dither)
Precision on clean external edges	15.2 ps rms rising, 22.2 ps falling, one channel at all phases without dither (20.0 ps with): a function generator's square wave through coax and a fork adapter onto channels 7 and 8 (one source, the difference over √2; 14.9–15.2 ps at 1–30 MHz, 60 Mevents/s on the two without loss, 2026-10-31)
Slow edges	add 0.8–5 mV rms of the board's own noise divided by the slew rate (sines of 9.5–283 mV/ns: 155 ps at 1 MHz, 15 ps at 30 MHz); part of it is the 250 MHz sample clock; keep edges faster than 0.3 V/ns
Jitter of two channels together	5.6–29.5 ps rms (median 15.2)
Calibration agreement between two channels	5.5–13.8 ps rms over the period (the test signal at 8 phases, code density of 2 M hits per polarity)
SATA pairs through a cable	16.9–18.6 ps rms against an internal channel
Dead time	about 5 ns (the delay line), plus the programmable dead time
Rates	125 Mevents/s through the merger (design); to a PC 7 Mevents/s sustained (20 s; at 8 Mevents/s 0.4 % are lost, counted), bursts of 2 s complete at 10 Mevents/s (the 32 MB ring the service allocated; 4 s with 96 MB)
Offsets	±2.05 µs per channel in the FPGA (more on the PC)
Polarity offsets	measured per channel by `calibrate_delays()` and corrected in the FPGA (POLARITY): the toggle paths −57 to +21 ps (316–421 ps before the input stages were placed beside the lines), the test path −358 to −211 ps; after the correction the toggles' alternate transitions agree to ±14 ps at one phase (once 23.5 ps: the tables' error at that phase)
Delays to the connectors	subtracted per channel (`connectors`, 3.25–3.89 ns): Vivado's model of each path at the measured scale, 39 ps rms from the test paths' measured delays; the test signal's own paths measured exactly (the channels agree to 5 ps on it); the board's traces not included
Calibration on asynchronous edges	the time error of 10^6 edges from a function generator does not depend on their phase in the 4 ns period: 8.2 ps rms over 40 phase bins (29 ps peak to peak)
References	IN2 10 MHz sine (I/Q meter), any channel (tagged clock, PPS, armed start); a GPSDO's PPS re-expressed in its 10 MHz's time through the meter: seconds to 35 ps rms, ADEV $4.3 \cdot 10^{-11}$ at 1 s and $4.1 \cdot 10^{-13}$ at 100 s (2 h, 2026-10-31)
The board's crystal	−4.50 ppm against a GPSDO, −128 ppb per °C of the die; ADEV $1.8 \cdot 10^{-9}$ at 1 s, $1.9 \cdot 10^{-8}$ at 1000 s (following the room's temperature, fan on); timestamps are re-expressed in a reference's time scale on the PC
Temperature	die 48–71 °C (fan off, then on, 2026-10-31): bins −0.02 % per °C (live calibration follows); recalibrated, the channels' offsets move by 0.02–0.15 ps per °C
Markers	8 × 3.3 V LVCMOS (E1 DIO0_N..DIO7_N), 1 ns bins: 291 ps rms against a channel on the same edges, referred to the connectors with the channels (a measured 12.7 ns delay subtracted); pulses shorter than 4 ns are counted as glitches, not timed
Unmeasured	the sample clock's jitter alone (common to all channels; the PPS through the meter bounds it with everything else: 35 ps rms)

Hardware setup

Nothing is needed for the self-calibration and the internal tests. Detectors go on E1 (3.3 V; 5 V TTL with a 50 Ω terminator). A straight SATA cable from S1 to S2 loops the test signal into channels 9 and 10. A 10 MHz sine reference goes on IN2 (HV jumper, or 6 dB and 50 Ω for +13 dBm).

Documentation

Document	Contents
docs/manual/user_manual.pdf	the user's manual (LaTeX): Part I the instrument, chapter by chapter with measured figures (calibration, precision, stream, markers, coincidences, filter, references, self-tests, specifications); Part II time tagging in quantum labs (histograms and their statistics, g2, TCSPC, HOM, heralding; entanglement-based QKD in depth: clock offsets from the photons, windows, QBER and key rates; prepare-and-measure QKD, quantum time transfer; clock comparison against a GNSS-disciplined oscillator, measured)
docs/PROTOCOL.md	every setting, Python methods, TCP commands, the records stream and the record format
docs/THEORY.md	how it works: delay lines, input stage, encoding, channels, markers, merging, coincidences, calibration, records, ring and stream, references, test signals, error budget, the prototype's findings
docs/REGISTERS.md	the hardware registers and the ring's layout in DDR
docs/VERIFICATION.md	what was tested, when; resource use and timing
../common/README.md	the shared framework: TCP service, common API, build workflow
NOTICE.md	sources
AGENTS.md	notes for maintainers and coding agents

Changing the design

Rebuilding needs Vivado 2025.1 on a build machine (../common/SETUP.md). From this directory:

```
make                      # create the project (first time), build, export the overlay
make test                 # offline tests: model, driver, decoding, stream, worker, protocol, client
make sim                  # simulation: system ID and the core against the model, word for word
make test-remote          # upload, program the board, check identity and power-up state
make install-service      # the control service, with timetagger active
python3 ../common/tests/verify_build.py --design timetagger
```

`make test-hw` runs `scripts/test_hw.py` on the board (no wiring needed; optional checks with a SATA cable from S1 to S2 and with edges on DIO0_P).

Source layout

Path	Contents
`design.tcl`	design description for the common recipe: ID "TTAG", RTL, one `tt_core` cell, HP0, the pins
`src/tt_core.v`	top: registers, time base, acquisitions, pins, the ten channels
`src/tt_front.v`, `src/tt_strip.v`, `src/tt_enc.v`, `src/tt_popcount.v`	input stage, delay line (80 CARRY4), encoder
`src/tt_chan.v`, `src/tt_fifo.v`, `src/tt_count.v`	channel logic, its FIFO, DSP48 counters
`src/tt_merge.v`, `src/tt_table.v`	merger, calibration lookup, event queue, records
`src/tt_mark.v`	the markers (ISERDES at 1 ns)
`src/tt_sort.v`, `src/tt_coinc.v`, `src/tt_count48.v`	the coincidence engine: its sorter, clusters, patterns, masks, virtual events; 48-bit DSP48 counters
`src/tt_writer.v`	the ring writer (AXI3 over HP0, mailbox, watchdog)
`src/tt_cal.v`, `src/tt_ring.v`, `src/tt_test.v`, `src/tt_gen.v`, `src/tt_xoro.v`, `src/tt_meter.v`	calibration histogram, ring oscillator, test signal, generator, random words, IN2 meter
`constraints/pins.xdc`, `constraints/strips.tcl`	pins and I/O standards; placement of the delay lines and timing exceptions

Path	Contents
`control/timetagger/`	Python package: bit-exact model (`model.py`), driver, the service's worker and stream hub, decoding and measurements (`analysis.py`), TCP commands, client, Jupyter GUI
`control/timetagger.ipynb`	the notebook
`scripts/test_hw.py`	the board test (`make test-hw`)
`tools/ttcore/`	host-only: the C++17 library and pybind11 module (decoding, measurements, the stream client, recordings) with its tests
`tools/`	host-only: the prototype's figures (`p1_figures.py`) and recordings; the stability tool (`stability.py`); the manual's recordings (`record_manual_data.py`), measured figures (`manual_figures.py`), computed figures (`app_figures.py`) and simulated photon sources, detectors and clocks (`qsim.py`)
`docs/manual/`	the user's manual: `user_manual.tex`, chapters `um_*.tex` (Part I) and `ap_*.tex` (Part II), `values.tex` and `figures/` (generated), `data/` (the board recordings), `references.bib`
`tests/`	`tb_core.sv`, `tb_coinc.sv`, `tb_accept.sv` (simulation), `vectors.py` → `tt_vectors.sv`, `tt_sy_*.mem` (the model's vectors), Python tests

Verification

The current build (2.0: the markers, the coincidence engine, the conditional filter) passes the offline tests, the simulations (the core's ring words and counters against the model, word for word; the coincidence engine; the acquisition's start and stop) and all 89 board checks (2026-10-30: calibration, the ring oscillator's frequency, the polarity offsets and the delays' scale, precision and agreement on the test signal after the corrections, the generator through the stream, dead time, divider, acquisitions with exact starts and stops, the coincidence engine against the PC's clusters (recorded, engine-only, on the test signal, gated by a marker), the conditional filter, the SATA loop, the markers against a channel on the same edges). Build, timing and board results: docs/VERIFICATION.md.

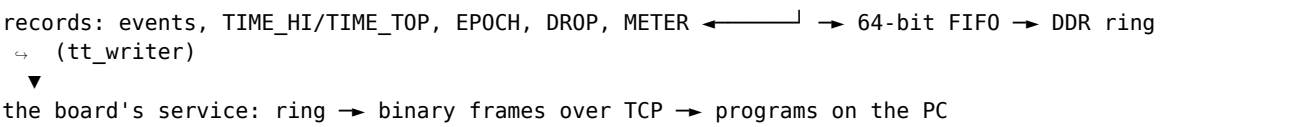
Author

Rok Zitko, Jozef Stefan Institute (Ljubljana, Slovenia), rok.zitko@ijs.si. Credits: NOTICE.md.

timetagger: theory of operation

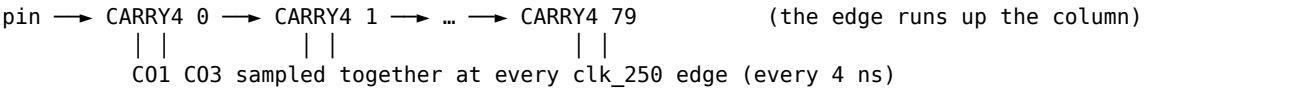
This page explains how timetagger times an edge to picoseconds with an FPGA whose clock ticks every 4 ns, how it calibrates itself, and how the events reach a program. Registers are in REGISTERS.md, commands and the record format in PROTOCOL.md, measured results in VERIFICATION.md. The bit-exact model of the digital part is `control/timetagger/model.py`; the RTL is checked against it word for word.

```
DIO0_P..DIO7_P (E1), SATA S2 pairs A/B
  │  input stage: the level, or a toggle flip-flop per chosen edge; or the ring oscillator,
  │  the test signal (tt_front, placed beside the line's first cells)
  ▼
delay line: 80 CARRY4 cells, 160 taps, sampled every 4 ns (tt_strip)
  ▼
encoder: hit, code = popcount, polarity, dbl/ovf (tt_enc)          ┐
  ▼                                                                 │ 250 MHz, per channel
channel: edges, dead time, divider, counters, (1, 2) the conditional  │
         filter (tt_chan) ─► FIFO 512                              ┘
  ▼                                    (+ the test generator's two sources, the markers: 1 ns)
merger: release in order of the 4 ns sample, round robin (tt_merge)
  ▼
calibration lookup: table [channel][polarity][code] + dither, offset, polarity offset (tt_table)
  ├─► (the lanes' events, the gate's marker) sorter: time order (tt_sort) ─► coincidence engine
   ↪  (tt_coinc)
  ▼                                                      │ virtual events
```



Delay lines

The FPGA's adders contain a fast carry chain: each CARRY4 cell passes a carry through four multiplexers and its four outputs CO0..CO3 show where the carry has got to. A chain of cells in one column of the chip is a **tapped delay line**. When an edge enters the chain it runs up the taps; at the next rising edge of the 250 MHz sample clock, flip-flops next to the taps record which taps it has already passed. If the edge entered a time Δ before the clock edge, the taps with delay below Δ show the new level: **their number measures Δ**, with a resolution of one tap.



The coarse time comes from counting clock edges; the delay line supplies the time within the 4 ns period. Three facts make this harder than it sounds:

- **The taps are not equal.** Inside a CARRY4 the four outputs are not in delay order (CO1 leads the others by 24–52 ps on this board), and cells, the boundary between clock regions and the sampling clock's routing add their own steps. Some bins are nearly empty, others two or three times the average.
- **The delays drift** with die temperature and supply voltage: on this board the mean bin narrows by 0.02 % per °C (the carry chains get faster as the die warms, while the ring oscillator's LUTs and routes get slower by 0.018 % per °C; 48–71 °C, 2026-10-31).
- **The chain must cover a whole period at every temperature**, plus the margin the hit detection needs (below).

timetagger 1.0 uses **80 cells and two outputs per cell, CO1 and CO3**: 160 taps of about 30 ps on average (the 0.1 prototype measured 60–64 ps per cell at a 67 °C die, 62–67 cells per period, and found two taps per cell nearly as precise as four, with the slices to fit ten channels). Each channel's cells go up one slice column next to its I/O bank, placed by hand (`constraints/strips.tcl`), with each tap's first flip-flop in the same slice: the tap-to- flip-flop paths are the short dedicated ones. Channels 1–8 use columns X43..X29 rows 20–99, channels 9–10 (SATA) columns X27, X25 rows 0–79; every chain crosses one clock-region boundary.

The first flip-flop rank samples an asynchronous signal and may go metastable; a second rank gives it a full period to settle before the encoder looks.

Input stage

A channel's chain takes (`tt_front`, `channels.n.input` and `toggle`):

- **the pin's level** (`toggle` False): every edge enters the chain with its own polarity; needed for `edge = "both"`;
- **a toggle of the chosen edges** (`toggle` True, the default): a flip-flop clocked by the pin flips at every rising (or falling) edge, so each chosen edge puts exactly **one** transition into the chain whatever the pulse's width (a 2 ns pulse in level mode puts two transitions into the chain, closer than its length). The transitions alternate in polarity;
- the **ring oscillator** (calibration), the **test signal**, or the test signal through the channel's own toggle flip-flop (`test_toggle`): the same toggle path then times a periodic signal, whose alternate transitions measure the offset between the line's two polarities.

The toggle adds the flip-flop's clock-to-output delay to the path (common to all events of the channel: an offset) and its own jitter (small against the line's quantization; unmeasured). A toggle's transitions alternate in polarity, and the two polarities take different times through the flip-flop, the input mux and the carry chain: 316–421 ps apart on the 1.0 board (2026-10-28), where the stages sat far from their lines. Uncorrected, a toggle channel's times alternate by that much; the **polarity offset** (POLARITY, measured by `calibrate_delays()`) is added to the falling transitions' times.

Each channel's input stage has its own hierarchy and sits in the CLB of its line's first two cells (a hard pblock in `constraints/strips.tcl`), so that the paths from the pins into the lines are short. Left to the placer, the stages sat in the middle of the die and the paths took 8.1–10.1 ns at the slow corner, 1.9 ns apart; placed, 4.5–6.6 ns (the rest is each pin's route from its I/O bank to its line's first cell).

Encoding

A **hit** is a transition passing tap M = 2·m + 1 (CO3 of cell m, `hit_cell`, default 0) between two samples. Each transition passes M exactly once, so every edge gives exactly one hit, in the first sample after it reached M, whatever the bubbles elsewhere. The **code** is the number of taps showing the new level in that sample, a population count (`tt_enc`): it does not depend on the order of the taps. The code lies between the taps before M and the taps within M's delay plus one period, so the chain must be at least one period longer than M's delay; 80 cells leave about 15 cells of margin at 67 °C.

Two checks flag invalid codes. In the hit's sample the chain beyond the transition must still show the old level. If it does not, either the previous sample still held an older transition (**dbl**: two edges closer than the chain's length, about 5 ns: the channel's dead time) or the previous sample was settled and the chain is too short (**ovf**). Both are counted (FLAGGED) and dropped: a timestamp is never made from an invalid code.

The polarity v of the transition selects the calibration table (rising and falling transitions run up the chain differently).

Channels

Per channel and sample (`tt_chan`, model `Channel`):

- **edge**: with a level input the transition's polarity is the edge; with a toggle input every transition is the chosen edge. Edges not accepted (`edge`) are ignored;
- **dead time**: after an accepted event, events are ignored for `dead_time` (whole samples; an event exactly at its end is accepted);
- **events** are counted (EVENTS: accepted edges, before the divider);
- **divider**: every N-th event goes on;
- the event enters the channel's **FIFO** (512 entries) as {sample k mod 2^16, code, polarity, edge}; a full FIFO counts a **drop**.

Each channel takes at most one event per 4 ns sample; with the line's own dead time of about 5 ns the highest rate on one channel is about 200 Mevents/s for a moment and the sustained rate is set by the merger and the ring (below).

Conditional filter

A pulsed laser's sync at 80 MHz is more than the merger takes from one source (an event every two clocks: 62.5 Mevents/s) and far more than the stream (about 7 Mevents/s). For lifetime measurements (TCSPC) only the syncs that a photon follows matter. Channels 1 and 2 have a **conditional filter** (`channels.n.filter`; `tt_chan` FILTER, model `Core.filter_decide`):

- a filtered channel's events (after edges, dead time and divider) wait N + 3 samples in a shift register before its FIFO;
- each one enters the FIFO only if a **trigger** channel (`filter.triggers`, any of 1–10) had an event (after its own divider) at a sample in [k, k + N], k being the filtered event's sample; the others are dropped (EVENTS still counts them);
- N = `filter.window` / 4 ns, 0..30: up to 120 ns, the period of an 8.3 MHz laser (slower syncs stream without a filter).

The decision is made per sample, so the syncs kept are a superset of those with a trigger within N · 4 ns after them: a trigger up to 4 ns before a sync in its sample keeps it, and so does one up to (N + 1) · 4 ns after it. The timestamps themselves are exact; the PC computes the TCSPC histogram from them. A shared count-down marks "a trigger in the last N + 1 samples"; the filtered events' delay is aligned with it, so the decision needs no comparison per channel. Channels 1 and 2 take the delay also with the filter off; it is invisible, since an event is released 1024 samples after its sample.

Markers

The eight pins DIO0_N..DIO7_N are **markers** (`tt_mark`, model `Markers`): edges timed to 1 ns, for gates, line and frame clocks, triggers and other slow signals beside the channels. A PLL from the 125 MHz clock makes 250 MHz at 0° and 90°; each pin's ISERDESE2 in oversampling mode takes four samples per 4 ns sample, 1 ns apart. For each 4 ns window and pin, with the last sample of the window before (p):

- the level after the window is its last sample; where it differs from p there is an **event** at the last change in the window (between samples i − 1 and i, the time (k − k0)·4096 + 1024 i − 512 ticks: the middle of that nanosecond), its edge the new level, if that edge is accepted (`markers.n.edge`);
- where the level ends as it was but changed within the window (a pulse shorter than 4 ns), there is no event: a **glitch** is counted;
- the window's events form a batch; a serializer puts one event per sample into the markers' FIFO (the merger's source 12), the lowest pin first; it holds two batches, and a third that comes while two wait is dropped and counted (eight pins changing in every window for longer than about 8 ns, rare for markers).

The markers share one offset (`marker_offset`), have no dead time or divider, and are timestamped like the channels (records 10..17, channels 11..18 and −11..−18 for falling edges in programs). Their rms quantization is 289 ps (1 ns bins). A window's samples are labelled with the sample in which they reach clk_250, about three samples after they were taken: on the same edges as channel 1 the markers came 12.705 ns late (290 ps rms, 2026-10-29), which the driver subtracts with `connectors` (`MARKER_DELAY_S`). The levels of all eight pins are readable at any time (MARK_LEVELS, `hardware()`).

Merging

The merger (`tt_merge`, model `Core.clock`) runs at 125 MHz: clock n holds samples 2n and 2n + 1. Thirteen sources (ten channels, the generator's two and the markers) offer their FIFOs' heads. A head of sample k is **due** when

```
now − k ≥ 1024 + floor(offset / 4096)          (now = 2n, in samples)
```

i.e. 4.1 µs after its sample, shifted by the source's offset in whole samples (offsets reach ±2.05 µs: the delay stays positive). In each clock the first due source in round-robin order is popped. An event outside the acquisition (before k0, or at or after a stop) ends there (DISCARDED); the others are **released** into the lookup pipeline, which needs the pipeline and the event queue to hold fewer than 16 events (engine-only sources' events do not enter the queue and need no room). A source popped in one clock is not due in the next (its next head is not seen yet). Inside or outside is decided by the event's sample, so an offset can move its time a little before 0 or past the stop (a generator pair's second event by up to the pair offset). The comparison with k0 and the stop sample is pipelined: nothing is released in the two clocks after a start (every head due then is from before the new k0, which lies 256 samples ahead or at the arming event), and a stop counts from two clocks after its command. (Before this was handled, those two clocks accepted every head due after a start, events up to 5 µs before k0, and discarded every head due after a stop's command; found on the board, 2026-10-30.)

The FIFOs keep only the low 16 bits of an event's sample, enough for 131 µs. When the output is blocked (the ring full, or the writer not yet started), events wait in the FIFOs, and a head that waited longer would look young again. So a head older than 16384 samples (65 µs) is **stale**: it is popped and counted (STALE), never timed; and a head that ends at its release (discarded or stale) never waits for room in the pipeline. The merger therefore keeps draining while the output is blocked, and an event is either timed correctly or counted as lost.

So events leave in the order of their sample plus coarse offset, to within the few clocks that sources due at the same time take in turn; the fine order within a sample is not restored (the PC sorts: ttcore). One event per clock is 125 Mevents/s through the merger; the ring and the network are slower (below). The coincidence engine sorts its own input (”Coincidences”).

Coincidences

The coincidence engine (`tt_coinc`, model `Coincidence`) counts patterns of clusters of events in the FPGA. It uses the same definition as `analysis.coincidences` and ttcore's `Coincidences`, so the FPGA's counts and the PC's agree exactly on the same events.

- Its lanes are any 8 of channels 1–10 and the generator's two sources (for checks without wiring); 7 with a gate. The sorter below holds 8 events and must hold all the lanes' events that come together (the 2026-10-29 board test: with ten channels on the test signal, 9 % of its clusters lost channel 9 or 10, whose events came late).
- An event of an enabled lane that is not in the open cluster starts one. The following events with t − start ≤ window join it, their lanes ORed into its **pattern**. Falling edges count as their channel.
- The cluster closes at the first lane event beyond the window, when no event can still come within it (the horizon below, with nothing left in the sorter), or at an epoch.

The definition needs the events in time order, which the merger keeps only to within a few clocks. A **sorter** (`tt_sort`, model `Sorter`) between the lookup and the engine restores it for the lanes' events:

- It holds up to 8 events in time order and emits its oldest once that is older than the **horizon**: a time no later event can precede.
- The horizon comes from B, the earliest due sample a lane's event can still have. That is 2n less the largest overdue time (age − delay, in samples) among the lanes' due heads and the lane head popped in the previous clock, two samples more now. Every later release has t ≥ (B − 1024 − k0) · 4096 − 6656 ticks (a fine time is at most 4608 ticks, a polarity offset at least −2048). The sorter uses B of the release six clocks before, the lookup's length.
- The overdue times are kept to 4 bits. While one reaches 15 samples (a backlog), B is held, a conservative bound.
- When the sorter is full and an event arrives, it emits its oldest early. An event it then emits older than one before it is **late**: counted (SORT_LATE) and skipped by the engine.
- After a start it flushes for 24 clocks; then the engine closes its cluster.

Gate. A marker can gate the engine (`coincidence.gate`: marker 1–8, `invert`). The gate pin's events pass the sorter with the lanes' (the marker source then counts in the horizon), so the engine sees them in time order. The gate opens at the marker's rising edge (falling with `invert`) and closes at the next opposite edge. It is closed at every start until its first opening edge, so a level already there at the start does not open it, and a partial period at the start is not counted. A lane event on the very tick of a gate edge is taken in the order the two reached the sorter (equal times stay in arrival order), so it may count either way. Lane events count only while it is open; a cluster still open when the gate closes ends as usual, with the events that came while it was open. The eight markers share one merger source: when together they come near its 62.5 Mevents/s, the horizon is held and events can come late (counted, skipped; the bench's scenario, whose markers run at about 56 Mevents/s, has 12).

In simulations of round-robin release with Poisson events and 30 % pairs on ten channels, the 8 entries kept every event in order up to 0.52 events per clock (65 Mevents/s in all; scratch tools, 2026-10-29). At 0.65 events per clock, 66 of 39113 came late. A single lane near its own limit (a source is popped at most every second clock: 62.5 Mevents/s) backs up its FIFO and holds the horizon: on the board (2026-10-29), the generator engine-only with its pairs gave no late event at 15 and 30 Mevents/s (G1 alone 10 and 20 Mevents/s), 961 of 6·10^7 at 60 (G1 40), and at 90 (G1 60) its FIFO overflowed. A lane held back by a full ring while engine-only lanes go on (no room for recorded events) also makes late events: such lanes should be engine-only too.

A closing cluster:

- increments its pattern's count: 4096 patterns × 48 bits in block RAM, a read-modify-write through the RAM's output register. The read misses the two writes before it, whose counts are forwarded. A mirror RAM takes the same writes and serves the register reads, so the engine never waits for the bus.
- increments CLUSTERS.
- is compared with eight masks: inclusive (the pattern contains the mask's lanes) or exclusive (equals them). Each matching mask counts (a 48-bit DSP48 counter).
- for each matching mask marked virtual, makes a record: source 20 + mask, host channel 21–28, rising, at the cluster's start. These go through a 16-cluster queue to the formatter (after DROP reports, before the event queue); when the queue is full they are lost and counted (VIRTUAL_DROPS). They come up to the window late in the record stream, which the PC's sorter absorbs.

Clusters close at most once per clock, so the engine takes the merger's full rate. A source may feed the engine only (ENGINE_ONLY: not recorded), so coincidences can be counted at rates the stream (about 7 Mevents/s) cannot carry. Clearing takes 4096 clocks (33 µs), and clusters closing meanwhile are not counted.

An exact time order for all sources in the merger (oldest first, with a sorter in the record path) was tried first, on 2026-10-29. It kept every event in order up to 0.85 events per clock, but cost about 2,400 LUTs, filled the slices (99.9 %) and missed timing by 3.5 ns. So the records keep 1.0's order and only the engine sorts.

Calibration lookup and time

For a channel's event, the table entry [channel][v][code] holds the bin's **left edge** and **width**, both in 1/16 tick (tick = 4 ns / 4096 = 0.9765625 ps). The fine time, the time from the edge's arrival at the chain to the sample's clock edge, is

```
fine = (left + (width · u >> 8) + 8) >> 4        (ticks)
```

with u an 8-bit **dither** byte (xoroshiro128+) when dithering, else 128 (the bin's centre). Dithering spreads the times uniformly over the bin: histograms of time differences get no comb from the bins. The event's time is

```
t = (k − k0) · 4096 − fine + offset (+ polarity, for v = 0)        (ticks, 60 bits: 13 days)
```

with k0 the acquisition's start sample, offset the channel's (22 bits signed: the user's offset less the delay from the connector, "Delays to the connectors") and polarity the offset of the line's falling transitions (12 bits signed). The generator's events carry their fine time instead of a code.

The lookup is a 5-clock pipeline: the head's fields; k − k0 and the offsets (and the table's read); the polarity offset added (the table's output register); the fine time (a DSP multiplier); t.

Code-density calibration

If edges arrive at phases uniformly distributed over the clock period, each code's share of the hits is its bin's width: w_c = T · h_c / Σh with T = 4 ns, and the left edge of code c is Σ_{c'<c} w_c'. The **ring oscillator** (`tt_ring`: a NAND and 16 inverters in a loop, free-running, unrelated to the sample clock, 132.7 MHz) provides such edges, divided to a rate the histogram can follow. Unrelated must mean far from the sample clock's strong subharmonics: a ring near 125 MHz is pulled towards half the 250 MHz clock (by the supply and the substrate), its phase lingers near the locking point and slips quickly elsewhere, and its edges stop falling uniformly. With 13 stages the ring ran at 119 MHz in one placement and at 124.9–125.0 MHz in another (2026-10-28): there the tables came out wrong by up to ±150 ps, although the bins looked normal (mean, largest, rms quantization), while the test signal's phase sweep showed it at once (residuals 84 ps rms against 20). The ring now has 17 stages in a hard pblock in a corner without 250 MHz logic; `calibrate()` reports its frequency and warns within 3 MHz of 125, 83.3 or 62.5 MHz, and `scripts/test_hw.py` checks it. `calibrate()` switches the chosen channels to it for a second, reads the **calibration histogram** (`tt_cal`: 5120 bins, [channel][v][code]), turns each channel's two histograms into tables (`driver.table_words`), loads them and keeps them in the design's data directory per board and bitstream (they depend on the placement). At start the service's worker loads the kept tables, or calibrates when there are none.

The histogram looks at one channel per clock in turn and takes its latest hit (an update takes 4 clocks), also while no acquisition runs. With sparse hits it takes every hit; with dense periodic ones (the ring oscillator at 16 M edges/s per channel) the fixed order of its looks beats against the signal's period and the sample is not uniform: tables made that way were off by ±100 ps (2026-10-28). `calibrate()` therefore divides the ring to about 1 M edges/s per channel (4 s: 2 M hits per polarity, about 3 ps of statistical INL).

From a table follow the differential non-linearity DNL_c = w_c / w̄ − 1, the integral non-linearity INL (the table's deviation from a straight line) and the rms quantization of a uniformly random arrival, sqrt(Σ w_c^3 / 12 T): with equal bins of width w it is w/√12, 8.7 ps for 30 ps bins.

Live calibration

The delays follow the die's temperature. While measuring, the channels' own hits feed the histogram (`calibrate` True): every 5 s the worker turns the histogram of each channel and polarity with at least 1 000 000 hits into a new table. Only random arrivals give a code density: a signal locked to the sample clock (e.g. a clock derived from the board's crystal) fills only a few codes. The worker therefore keeps a histogram only if no more than 10 % of the codes that the previous table says are wider than 5 ps are empty, and counts the others (`WORKER?` "rejected"). Channels on internal inputs are never taken.

Delays to the connectors

A channel's time is the moment its transition entered the delay line, against the sample clock's edge at the line's flip-flops. Both differ from channel to channel: the path from the pin through the input buffer, the routing and the input stage (4.5–6.6 ns at the slow corner), and the clock's arrival (a spread of 80 ps). Vivado's timing model knows every path. The build exports it (`design_export` in `design.tcl`, run by the common build on the routed design): `timetagger.json` "delays" holds per channel, in ns at the slow (max) and fast (min) corner, the clock's arrival at the line's first flip-flop and the paths to it from the pin (level input; the toggles: pin to the flip-flop's clock plus the flip-flop to the line), from the test signal (its counter's bit) and from the test signal through the toggles, and the test signal's paths to the SATA S1 pins.

The board's actual delays lie between the corners, at about the same fraction λ of the way from the fast to the slow one for every path. The driver subtracts, for each enabled channel and its input path, d(λ) = [path_f + λ (path_s − path_f)] − [clock_f + λ (clock_s − clock_f)] from its times (in the OFFSET register; `connectors` true, the default; `status()` lists the corrections). Times then refer to the connectors (up to a constant common to all channels), and the user's `offset` adds to that.

`calibrate_delays()` (part of `calibrate()`, about 7 s) measures, with no wiring:

- **λ**: the test signal reaches every channel's input stage. With it at each channel's level input, the code histograms per polarity (`tt_cal`) give each channel's mean time at the test signal's phase, through the channel's table (the bins' centres, a circular mean over the 4 ns period). Their differences between channels, fitted to the model's differences of the test paths as a + fast + λ (slow − fast), give λ, and their residuals how well the model describes the paths. The test signal is moved over 8 phases of the period so that the tables' local errors average.
- **the polarity offsets**: with the test signal clocking a channel's own toggle (rising, then falling edges: the same flip-flops and paths as from the pin), the transitions alternate in polarity at one phase: the difference of the two polarities' mean times is the offset to add to the falling ones (`toggle_rise`, `toggle_fall`). At the level input the test signal's rising and falling edges fall at the same phase (both are made at the same edge of its clock), which gives the test path's own offset (`test`). A pin's level input has no calibration: the input buffer's two edges, and the source's, are not on the test path.

The results are kept with the tables (`tables-<build>.json`: "polarity", "delay_scale", "delay_residual_ps", "test_residual_ps") and applied at once. The model has one delay per path for both edges, so λ is fitted to the mean of each channel's rising and falling edges; the test inputs then take their rising edges' own offsets from the model (`test_residual_ps`), so that on the test signal the channels agree to the measurement (within 5 ps, `scripts/test_hw.py`), and only the pins' paths rely on the model. Measured (2026-10-28, 70 °C): λ = 0.02 with 48 ps rms residuals (the routes near the fast corner; build 0x2a8ff3d5), λ = 0.28–0.35 and 50–56 ps on the build before the input stages were placed (0xf47cf07e, the rising edges alone), and 0.27–0.31 for the carry cells (58–62 ps against the model's 39 and 114). Before a calibration λ is 0.1.

What the correction does not include: the board's traces from the connectors to the FPGA's pins, the source's own delays and cables, and the model's error on each path (the residuals above). For the best agreement between channels, split one signal into both with equal cables and swap them (the source's skew cancels), or measure a known delay.

Acquisitions and records

An acquisition starts (START) 256 samples after the command, at a given sample (START AT), or at the next event of an armed channel (ARM, e.g. a PPS: that event is at t = 0); it ends at a STOP, at a given sample, or after a duration set at the start. Each start sends an **EPOCH** record with its count and k0 and resets the time context.

The records are 32-bit words (PROTOCOL.md "Records"). An event carries the low 26 bits of t (65.5 µs); **TIME_HI** (t[48:26]) precedes the first event whose high bits differ and **TIME_TOP** (t[59:49]) the first whose top bits differ, so a steady stream costs one word per event. A **DROP** record reports a source's losses (one source checked every 256 clocks), a **METER** record a window of the IN2 phase meter.

Ring and stream

The writer (`tt_writer`) packs the records in pairs into 64-bit beats, queues up to 2048 beats and writes 128-byte bursts over the PS7's HP0 port into a **ring** in DDR (16–96 MB of contiguous memory, the first 4 KiB a mailbox) that the service's worker allocates. A burst is written only when the worker's position, which it writes back 100 times a second, leaves room for two more; otherwise the writer waits (W_STALLS), its queue fills, the merger stops releasing (its output is not taken), the channels' FIFOs fill and their events are **dropped and counted** there (DROPS, or STALE for events that waited longer than 65 µs): nothing is overwritten unread. The worker's position write is also a **heartbeat**: when it stops for 5 s the writer stops, so that the FPGA never writes into memory the service has freed for longer (the worker restarts a stopped writer once it has taken the last records: a heartbeat comes late while another process holds the hardware, e.g. 1.3 s while it constructs its driver). After 100 µs without records a partial burst is completed with NOP records and written, so a slow source is not delayed for long.

The records leave the ring through the **ring pump** (`common/python/rpfpga/ringpump.c`, built on the board by the worker): it reads the ring itself and sends frames of up to 256 KiB to each client of port 5026 (port 5025's RECORDS STREAM relays it), each with its own queue of 16 MB: a client that falls behind loses whole frames, counted in the next frame's header (the FPGA is never held up by a slow client). The worker (`worker.py`) only passes the FPGA's written position to the pump and the pump's position back to the FPGA. Without a compiler the worker copies the records itself into the stream's hub (Python, slower).

The rates: the merger takes 125 Mevents/s (one per clock); the ring writer and DDR port keep up with it (measured to 60 Mevents/s: two channels at 30 MHz for 33 ms, no drops, 2026-10-31; unmeasured beyond); the pump sends about 27 MB/s, 6.7 Mevents/s, sustained to a PC (2026-10-28; 7 MB/s through Python; 7 Mevents/s for 20 s on 2026-10-30), and the ring holds a burst beyond that (10 Mevents/s for 4 s with 96 MB, for 2 s with the 32 MB the service allocates on the current SD card).

References and PPS

The board's own crystal clocks the hardware; timetagger does not lock it to an external reference. Instead the timestamps are related to a reference afterwards:

- **a 10 MHz sine on IN2**: the phase meter (`tt_meter`, from atomic_clock) correlates IN2 with the exact 2/25 sub-harmonic of the sample clock over windows of 25·M samples (whole periods: exact DC rejection): $I = \Sigma\, x \cdot C$, $Q = \Sigma\, x \cdot S$ with 18-bit cosine and sine tables, one METER record per window. A sine $A\cos(2\pi \cdot 0.08 \cdot n + \psi)$ gives $I = N \cdot A \cdot \cos\psi/2 \cdot 131071$ and $Q = -N \cdot A \cdot \sin\psi/2 \cdot 131071$: the phase ψ per window measures the board's time against the reference;
- **a tagged 10 MHz square or a PPS on any channel** (with the divider for fast clocks): their timestamps measure the board's time directly;
- **an armed start on a PPS** puts t = 0 on the PPS edge.

On the PC, `analysis.reference_edges` (a tagged clock or PPS: each edge numbered by the elapsed time, its offset smoothed over neighbouring edges, non-causally) or `analysis.meter_offsets` (the METER records: the unwrapped phase of each window, placed at the window's centre) give the offset reference time − board time along the acquisition, and `analysis.to_reference` re-expresses any timestamp by interpolating it. A 10 ppm crystal error is removed to within the reference edges' jitter averaged over the smoothing (a test with 10 ps of edge jitter and 51 edges: 5 ps).

Test signals

- The **ring oscillator** (above).
- The **test signal** (`tt_test`): the second MMCM multiplies the 125 MHz clock to a 1 GHz VCO and divides it to 25 MHz; its **dynamic phase shift** moves that output by 1/56 of the VCO period, 17.857 ps on average, per step (224 steps per 4 ns). A counter on it makes the test signal (12.5 MHz / 2^n), whose transitions all sit at one programmable phase of the sample clock; it can drive all channels and leave on the SATA S1 pairs.
- The **generator** (`tt_gen`): in every sample, with probability p, an event with a random fine time enters the generator's first source (record source 28, channel 31 in programs), and with probability q the same event enters its second (29, channel 32), whose offset sets the pairs' delay. It exercises the merger, ring, stream and analysis at a known rate with known coincidences, independent of the delay lines.

Linearity and precision self-tests

The MMCM's steps are exact only on average: its phase interpolator has an error of its own, about 20 ps rms over a period on this board (VERIFICATION.md, 2026-10-27), so the phase sweep is no linearity ruler. What it provides is the same transition, at every phase of the period, to all ten delay lines at once: the **difference of two channels' calibrated times** does not depend on the test signal at all. Its rms at one phase is the two channels' combined single-shot precision (the sample clock's jitter is mostly common to both), and the spread of its mean over the phases is how well the two calibrations agree (`scripts/test_hw.py`).

Error budget (measured values from VERIFICATION.md; the others marked)

Term	Size
quantization of 29–31 ps bins	10.2–13.1 ps rms (code density, two builds, 2026-10-28)
calibration error, two channels compared over the period	3.0–14.9 ps rms (1.0, two builds, 2026-10-28; 3.5–4.1 ps in 0.1 with four taps)
jitter of two channels at one phase (no dither)	6.9–23.7 ps rms (1.0, two builds): roughly 5–17 ps per channel
one channel on clean external edges	15.2 ps rms rising, 22.2 ps falling at all phases, quantization included, no dither (one source on channels 7 and 8 through coax, 1–30 MHz, 2026-10-31)
slow edges	0.8–5 mV rms of the board's own noise at the inputs divided by the slew rate (155 ps at 9.5 mV/ns); part of it is the 250 MHz sample clock (a 163 ps rms swing over the period at 9.5 mV/ns); negligible above 0.3 V/ns
the sample clock's jitter (MMCM from the 125 MHz crystal)	mostly common to the channels; unmeasured alone

Term	Size
temperature drift between calibrations	the bins −0.02 % per °C: a table made at 48 °C is about 20 ps off at the end of the period at 71 °C (live calibration follows); recalibrated, the channels' offsets move by 0.02–0.15 ps per °C (2026-10-31)
the input buffers, the toggle flip-flop	the toggle paths' two polarities differ by −34 to +19 ps (placed beside the lines; 316–421 ps before), calibrated to ±14 ps at one phase (`calibrate_delays()`); on clean external edges the level input gives 16.2 ps rms rising, 18.3 falling per channel, the toggle 15.2 and 22.2 (2026-10-31)
the delays to the connectors	Vivado's model at the measured scale λ: 48 ps rms on the test paths (up to ±104 ps per channel); the board's traces are not included
calibration on asynchronous edges	the time error does not depend on the arrival's phase within 5.9–7.6 ps (10^6 edges, 40 bins)

Prototype findings (0.1, 2026-10-27, die at 67 °C)

- A CARRY4 cell takes 60–64 ps: 62–67 cells per period. Inside a cell CO1 leads the other three outputs by 24–52 ps, so the bins of all four taps are very uneven (0 to 74 ps).
- 64 cells are too short (4.5 % of the hits found the chain full); 80 cells cover the period with margin (a cooler die makes the cells faster and the chain shorter in time).
- With CO1 and CO3 the bins are 29–32 ps on average and the rms quantization 10–11 ps, against 6–9.5 ps with all four; the single-shot precision changes little.
- After code-density calibration two independent delay lines agree to 3.5–4.1 ps rms over the period; one line's single-shot precision is about 8–9 ps rms with four taps.
- Rising and falling transitions need their own tables and have their own offsets (up to hundreds of ps between paths).
- The SATA pairs work as DIFF_SSTL18_I inputs with the FPGA's split termination (the board wires them straight to the FPGA), with the same precision; a straight SATA cable from S1 to S2 loops the test signal into channels 9 and 10.

timetagger: settings, Python methods, TCP commands, the records stream

Times are **seconds** in the configuration and **picoseconds** in the decoded data. Channels are numbered 1–10: 1–8 are DIO0_P..DIO7_P on the E1 connector, 9 and 10 are the SATA S2 connector's pairs A and B. The common commands and methods (CONFIG, SNAPSHOT?, PRESET, SAFE, STATUS?, ...) are in ../../common/README.md.

The configuration

`tt.config()` returns, and `tt.apply(patch)` merges, a JSON-compatible dict. The returned values are the realized ones (dead times in whole 4 ns samples, offsets in ticks of 0.9765625 ps).

```
{"channels": {"1": CHANNEL, ..., "10": CHANNEL},
 "generator": {"rate": 0.0, "pair_fraction": 0.0, "offset": 0.0, "pair_offset": 0.0, "record":
  ↪  True},
 "test": {"enable": False, "divider": 0, "sata_out": False, "phase_steps": 0},
 "ring": {"enable": False, "divider": 3},
 "markers": {"1": MARKER, ..., "8": MARKER},
 "marker_offset": 0.0,
 "marker_record": True,
 "coincidence": {"enable": False, "channels": [], "window": 1e-9,
                 "masks": {"1": MASK, ..., "8": MASK}, "gate": {"marker": 0, "invert": False}},
 "filter": {"triggers": [], "window": 0.0},
 "dither": True,
 "connectors": True,
 "reference": {"meter_window": 0.0}}
```

Key	Default	Meaning
`channels.n.enable`	False	the channel produces events
`channels.n.edge`	`"rising"`	`rising`, `falling` or `both`
`channels.n.toggle`	True	a flip-flop toggled by the chosen edge drives the delay line: one transition per edge whatever the pulse width (narrow pulses), at the cost of the toggle path. False: the pin itself (needed for `both`)
`channels.n.input`	`"pin"`	`pin`; `ring` (the ring oscillator), `test` (the test signal, both edges), `test_toggle` (the test signal through the channel's own toggle path): internal sources for calibration and self-tests
`channels.n.dead_time`	0.0	s, 0 … 262 µs: events closer than this after an accepted event are ignored (the delay line's own is about 5 ns)
`channels.n.divider`	1	keep every N-th event, 1 … 65536 (e.g. a laser's sync)
`channels.n.offset`	0.0	s, ±2.05 µs: added to the channel's times (cable delays)
`channels.n.hit_cell`	0	0 … 15: the delay-line cell where a transition counts as a hit (expert)
`channels.n.calibrate`	True	the channel's hits feed the calibration histogram (live calibration)
`channels.n.record`	True	the channel's events go into the records. False: to the coincidence engine only (rates the stream cannot carry)
`channels.n.filter`	False	channels 1 and 2 only: the conditional filter keeps an event only if a `filter.triggers` channel has an event within `filter.window` after it (THEORY.md "Conditional filter"); needs triggers, not the channel itself
`filter.triggers`	[]	the conditional filter's trigger channels (1–10), e.g. the detectors
`filter.window`	0.0	s, a multiple of 4 ns, 0 … 120 ns: decided per 4 ns sample, so the syncs kept are a superset of those with a trigger within the window after them (one up to 4 ns before, or up to the window + 4 ns after, keeps them too)
`markers.n.enable`	False	marker n (n = 1..8, DIO0_N..DIO7_N) produces events: edges to 1 ns (gates, line and frame clocks)
`markers.n.edge`	`"rising"`	`rising`, `falling` or `both`; a pulse shorter than 4 ns is a glitch (counted, no event)
`marker_offset`	0.0	s, ±2.05 µs: added to every marker's times
`marker_record`	True	the markers' events go into the records
`generator.rate`	0.0	events/s of the test generator (source 31 in programs), 0 … 250 M
`generator.pair_fraction`	0.0	the fraction of them copied to source 32 …
`generator.offset`, `generator.pair_offset`	0.0	… with source 31's offset and source 32's extra delay (s, ±2.05 µs)
`generator.record`	True	the generator's events go into the records (False: the coincidence engine only)
`coincidence.enable`	False	the FPGA's coincidence engine counts (THEORY.md "Coincidences")
`coincidence.channels`	[]	its lanes: any 8 of channels 1–10 and the generator's 31, 32 (7 with a gate: the engine's sorter holds 8 events that come together)
`coincidence.window`	1e-9	s, 0 … 16.4 µs: an event joins the open cluster within this of the cluster's first event
`coincidence.masks.m.channels`	[]	mask m = 1..8: channels among `coincidence.channels` ([]: off)
`coincidence.masks.m.exclusive`	False	the cluster's channels equal the mask's (else: include them)
`coincidence.masks.m.virtual`	False	a matching cluster is also an event of channel 20 + m (21..28) in the records, at the cluster's first event
`coincidence.gate.marker`	0	1–8: the engine counts lane events only while marker n's gate is open: from its rising edge to its next falling edge, closed at every start until its first rising edge (0: no gate)
`coincidence.gate.invert`	False	the gate opens at the marker's falling edges and closes at its rising ones (open while low)
`test.enable`	False	the phase-stepped test signal (25 MHz from the second MMCM, divided)
`test.divider`	0	0 … 15: 12.5 MHz / 2^divider
`test.sata_out`	False	the test signal also drives both SATA S1 pairs (a straight SATA cable from S1 to S2 loops it into channels 9 and 10)

Key	Default	Meaning
`test.phase_steps`	0	the test signal's phase, steps of 1/56 ns (17.857 ps on average)
`ring.enable`, `ring.divider`	False, 3	the ring oscillator (96–133 MHz depending on the build: `counters()["ring_oscillator_hz"]`) divided by 2^(divider+1), for calibrations
`dither`	True	a calibrated time is placed at random within its bin (else at the bin's centre)
`connectors`	True	each channel's times refer to its connector: the delay of its input path (Vivado's model at the measured scale, THEORY.md "Delays to the connectors") is subtracted. False: the times as the delay lines see them
`reference.meter_window`	0.0	s: the IN2 10 MHz phase meter's window (0 off, ≥ 100 µs): METER records in the stream

Presets: `off`, `detectors` (channels 1–8 on rising edges), `all` (1–10), `test` (all channels on the internal test signal), `generator` (1 Mevent/s, half of them paired 10 ns later).

Python methods (on the board: `pynq.overlays.timetagger.DRIVER`; on a PC: the client)

Method	Returns
`start(at=None, duration=None, arm_channel=None)`	start an acquisition now (t = 0 about 1 µs ahead), at a 250 MHz sample `at`, or at the next event of `arm_channel` (e.g. a PPS); `duration` s stops it in hardware. `{"acquisition", "k0"}`
`stop()`	stops the acquisition
`acquisition()`	`{"running", "armed", "stopping", "acquisition", "k0", "k_stop"}`
`now()`	the current sample (250 MHz count since the design was loaded)
`counters()`	per channel `events` (accepted edges, before the divider), `flagged` (closer than the delay line can separate, or the line full: dropped), `drops` (FIFO full); the merger's `released`, `discarded`, `stale` (events of an acquisition lost because the output was blocked for more than 65 µs), `queue`, `late` (lane events the coincidence engine's sorter delivered out of time order); the ring's `ring_bytes`, `ring_bursts`, `ring_stalls`; `ring_oscillator_hz`; `markers`: `events`, `glitches` (pulses shorter than 4 ns), `drops`
`clear_counters()`	
`calibrate(channels=None, seconds=4.0, delays=True)`	the code-density calibration on the ring oscillator (docs/THEORY.md "Code-density calibration"); the tables are loaded and kept per board and bitstream. Per channel and polarity: `hits`, `codes`, `mean_bin_ps`, `max_bin_ps`; `ring_hz` (and `ring_warning` near 125, 83.3 or 62.5 MHz, where the sample clock pulls the ring); with `delays`, then `calibrate_delays()` (its reply under `"delays"`)
`calibrate_delays(channels=None)`	with the test signal (no wiring, about 7 s): each channel's polarity offsets (`polarity_ps`: `test`, `toggle_rise`, `toggle_fall`) and the delays' scale between Vivado's corners (`delay_scale`, the test paths' `residual_ps` and `rms_ps`); kept with the tables and applied (THEORY.md "Delays to the connectors")
`delays()`	Vivado's delays of the channels' paths exported with the bitstream (`timetagger.json`), ns at the slow and fast corners
`status()`	also `connector_delays_s` and `polarity_offsets_s` per enabled channel (the corrections applied), `delay_scale`, `delays_known`
`histogram()`	the live calibration histogram `{channel: {v: [counts per code]}}`
`tables()`, `load_tables()`, `write_tables(tables)`	the stored tables; load them; write tables
`set_phase(steps)`	moves the test signal's phase; returns the position reached

Method	Returns
`coincidences()`	the coincidence engine's counts since its last clear: `patterns` {"1,2": the clusters whose channels were exactly 1 and 2, ...} (the nonzero ones), `masks` {"1": count, ...}, `clusters`, `virtual_drops` (virtual events lost: their queue full), `late` (lane events the engine's sorter delivered out of time order, skipped by the engine; counted since the bitstream was loaded: 0 below about 65 Mevents/s in all, unless a recorded lane was held back by a full ring or, while gated, the eight markers together come near 60 Mevents/s), `gate_open`
`clear_coincidences()`	zero the patterns (33 µs: clusters closing meanwhile are not counted), the masks' counts, `clusters`, `virtual_drops`
`hardware()`	status bits, the ring writer, the meter (`meter_windows_lost`, `meter_start_clock`: the 8 ns clock of window 0's first sample, for `analysis.meter_offsets`), the layout, the markers' levels now (`marker_levels`) and their PLL (`marker_pll_locked`), the die temperature (°C, PS XADC)

TCP commands

Command	Reply
`ACQ START [AT <sample>] [FOR <s>]`	JSON `{"acquisition", "k0"}`
`ACQ ARM <channel>`	JSON (k0 null until the event)
`ACQ STOP`	`OK`
`ACQ?`	JSON as `acquisition()`
`COUNTERS?` / `COUNTERS CLEAR`	JSON as `counters()` / `OK`
`HARDWARE?`	JSON as `hardware()`
`CALIBRATE [<channel> ...] [SECONDS <s>]`	JSON as `calibrate()` (a long command)
`CALIBRATE DELAYS [<channel> ...]`	JSON as `calibrate_delays()` (a long command)
`HISTOGRAM?`	JSON as `histogram()`
`TABLES?`	JSON as `tables()`: the calibration tables kept for this board and bitstream, {channel: {v: [word per code]}}; a word is the bin's width (bits 27:16) and left edge (bits 15:0), both in 1/16 tick
`COINCIDENCES?` / `COINCIDENCES CLEAR`	JSON as `coincidences()` / `OK`
`PHASE <steps>`	`OK <steps reached>`
`WORKER?`	the service worker's statistics: `bytes`, `polls`, `calibrations`, `rejected`
`RECORDS STREAM`	the records, on a connection of its own (below)

The records stream

`RECORDS STREAM` answers one line `OK {header}` and then sends binary **frames** until the client sends `STOP` or closes the connection. The board's **ring pump** serves the same stream on **port 5026** (any first line), at full speed (about 28 MB/s, 7 Mevents/s, to a PC); port 5025 relays it. `RecordStream` connects to 5026 when it answers, else to 5025. The header (JSON) has `format` "TTR1", `tick_s`, `channels`, `generator_sources` and `ring_bytes`. Each frame is a 24-byte header of little-endian u32 fields — magic "TTR1", sequence, payload bytes, bytes lost before this frame, TIME_TOP and TIME_HI in force at the frame's first record — followed by the records. A client that falls behind (32 MiB queued) loses whole frames, counted in the next frame's header; the time context lets it decode the next frame on its own.

```
import sys; sys.path[:0] = ["common/python", "timetagger/control"]
from timetagger import TimeTaggerClient, RecordStream
from timetagger.analysis import Decoder, parse_frame
with TimeTaggerClient("192.168.0.149") as tt, RecordStream("192.168.0.149") as stream:
    tt.load_preset("detectors")
    tt.start(duration=1.0)
    decode = None
    for frame in stream.frames(timeout=2.0):
        head, words = parse_frame(frame)
        decode = decode or Decoder(head["time_top"], head["time_hi"])
        channels, times_ps = decode(words)        # int16 channel numbers, int64 ps
```

Channels in the decoded data: 1..10 for rising edges (or the chosen ones), −1..−10 for falling edges, 11..18 and −11..−18 for the markers M1..M8, 21..28 for the coincidence engine's virtual events (masks 1..8), 31 and 32 for the generator. The C++ library `tools/ttcore` does the same at full rate and measures as the events come (README there); `analysis.reference_edges`, `meter_offsets` and `to_reference` re-express times in a reference's time (THEORY.md "References and PPS").

A `.ttbin` recording (ttcore's `StreamClient.record`) is `TTBIN1\n`, the stream's header JSON and `\n`, then the frames as received. `analysis.read_recording(path)` reads one in Python (header, channels, times in ps, sorted, controls); `analysis.time_error(t, period[, reference])` turns a periodic signal's edges into one time-error sample per period, against the board's clock or a reference channel, and `tools/stability.py` analyzes that with the common stability tools (ADEV, MDEV, HDEV, TDEV with confidence intervals, the power-law fit, Totdev, ThéoBR, MTIE with ITU-T masks: a PDF report):

```
python tools/stability.py report run.ttbin --channel 3 --period 1 -o pps.pdf
```

Records

Little-endian 32-bit words; times are ticks of 4 ns / 4096 from the acquisition's start (t = 0 at sample k0), 60 bits.

Word	Content
event	[31:27] source (0–9 channels 1–10, 10–17 markers M1–M8, 20–27 the coincidence masks' virtual events, 28–29 the generator), [26] edge (1 rising), [25:0] t mod 2^26
control	[31:27] = 31, [26:23] type, [22:0] payload
NOP (0)	padding
TIME_HI (1)	t[48:26] of the events that follow (sent before the first event whose bits differ)
TIME_TOP (2)	t[59:49] (then TIME_HI follows)
EPOCH (3)	an acquisition starts: payload its count; then two words: k0 [31:0], k0 [47:32]. The time context resets
DROP (4)	{source [22:18], events lost [17:0] (saturated)} since the last report of that source (one source every 256 clocks; source 10 for the markers together)
METER (5)	an IN2 meter window: payload its index; then I [31:0], I [63:32], Q [31:0], Q [63:32] (signed 64-bit sums, docs/THEORY.md "References")

Picoseconds are ticks × 125 / 128, rounded. Events are in order of their coarse time (4 ns samples) after each channel's offset; within a few samples, events of different channels may be out of order by up to their offsets' fine parts and polarity offsets. Virtual coincidence events come when their cluster closes, up to the window plus a few clocks after their time (ttcore's sorter puts everything in order).

timetagger 2.0 registers

The core's AXI4-Lite window is at **0x43C00000** (256 KiB, 32-bit words). Use the driver (`rpfpga.Overlay`): an access to an FPGA address without a loaded design hangs the board. The common sysid block at 0x40000000 reads ID "TTAG", version 2.0.

Units: a **sample** is one period of the 250 MHz sample clock (4 ns); a **tick** is 4 ns / 4096 (0.9765625 ps); a **clock** is one period of the 125 MHz register clock (two samples).

Control and acquisition

Offset	Name	Access	Content
0x000	CTRL	W	strobes: [0] START, [1] STOP, [2] ARM, [3] clear the counters, [4] clear the calibration histogram (5120 clocks), [5] empty the FIFOs
0x004	STATUS	R	[0] acquiring, [1] armed, [2] a stop is set, [3] ring writer running, [4] writer stopped by its watchdog, [5] histogram clearing, [6] test MMCM locked, [7] phase shifter busy
0x008	ACQ_CFG	RW	[3:0] ARM channel (0..9), [8] START at ACQ_AT (else 256 samples after the strobe), [9] STOP at STOP_AT (else 256 samples after the strobe)
0x010, 0x014	ACQ_AT	RW	the start sample (48 bits: low word, then [15:0] of the high word)
0x018, 0x01C	STOP_AT	RW	the stop sample (48 bits)
0x020, 0x024	K0	R	the acquisition's start sample k0 (t = 0)

Offset	Name	Access	Content
0x028, 0x02C	NOW	R	the current sample (2n, n the clock count); reading the low word latches the high word
0x030, 0x034	K_STOP	R	the stop sample (valid with STATUS[2])
0x038	ACQ_COUNT	R	acquisitions started since power-up (23 bits; also in the EPOCH record)

An acquisition keeps the events with k0 ≤ k < k_stop. ARM starts it at the next event of the chosen channel that enters its FIFO (after edge selection, dead time and divider): that event's sample is k0. START or ARM while acquiring starts a new acquisition (a new EPOCH).

Sources

Thirteen sources: channels 1–10 are sources 0–9 (DIO0_P..DIO7_P, SATA S2 pairs A and B), the test generator's two outputs are sources 10 and 11 (record sources 28 and 29), the markers M1–M8 (DIO0_N..DIO7_N) together source 12 (record sources 10–17). Source s has eight words at **0x100 + 32 s**:

Offset	Name	Access	Content
+0x00	CFG	RW	[0] enable, [2:1] edge mode (0: the line's polarity, a level input; 1: rising; 2: falling: toggle inputs), [4:3] accepted edges ([3] falling, [4] rising), [5] live calibration (hits enter the histogram), [6] the conditional filter (channels 1 and 2 only; THEORY.md "Conditional filter"), [10:8] input (0 pin level, 1 toggle at rising edges, 2 toggle at falling edges, 3 ring oscillator, 4 test signal, 5–7 none), [11] the toggle is clocked by the test signal, [15:12] hit cell m (a hit when a transition passes tap 2m + 1)
+0x04	DEAD	RW	extra dead time after an accepted event, samples (16 bits)
+0x08	DIV	RW	keep every (DIV + 1)-th event (16 bits)
+0x0C	OFFSET	RW	the source's offset, ticks (22 bits signed: ±2.05 µs); added to its times. Its whole samples delay the source's release in the merger
+0x10	EVENTS	R	accepted events (before the divider)
+0x14	FLAGGED	R	hits closer than the delay line can separate (dbl) or with the line full (ovf): counted and dropped
+0x18	DROPS	R	events lost at a full FIFO (512 entries)
+0x1C	POLARITY	RW	the offset of the channel's falling transitions in its delay line (v = 0), ticks (12 bits signed: ±2.0 ns); added to their times after OFFSET. The driver writes what `calibrate_delays()` measured for the channel's input path (0 for the generator's sources)

The generator's sources use only OFFSET and DROPS (their CFG reads 0x18). The markers' (0x280): CFG [15:0] their accepted edges (pin p: bit 2p + 1 rising, bit 2p falling; 0 at power-up), OFFSET (one for the eight pins), EVENTS (accepted marker events), FLAGGED (glitches: a pin that changed and came back within one 4 ns window), DROPS (events the serializer or the full FIFO lost); DEAD, DIV and POLARITY are unused (0). OFFSET and POLARITY read back sign-extended to 32 bits.

Test signals, dither, generator

Offset	Name	Access	Content
0x040	DITHER	RW	[0] dither the fine time within its bin (else the bin's centre)
0x044	GEN_P	RW	[23:0] the generator's event probability per sample × 2^24
0x048	GEN_Q	RW	[7:0] the probability × 256 that an event is copied to source 11
0x04C	RING	RW	[0] ring oscillator on, [7:4] divider n: f_ring / 2^(n+1) into the channels
0x050	TEST	RW	[0] test signal on, [1] also out on both SATA S1 pairs, [7:4] divider n: 12.5 MHz / 2^n
0x054	PS_TARGET	RW	the test signal's phase, MMCM fine steps (signed 16 bits, 1/56 of 1 ns: 17.857 ps on average, 224 per 4 ns)
0x058	PS_CURRENT	R	the phase reached
0x05C	RING_FREQ	R	f_ring = value · 64 · 125 MHz / 2^24

Calibration and the 10 MHz meter

Offset	Name	Access	Content
0x060	CAL	RW	[0] the histogram takes hits
0x064	CAL_UPDATES	R	histogram updates since power-up
0x068	METER_M	RW	IN2 meter window: 25 · METER_M clocks (0: off)
0x06C	METER_LOST	R	windows lost (one not yet sent when the next ended)
0x070, 0x074	METER_START	R	the clock of window 0's first sample (64 bits)
0x4000 + 4a	TABLE	W	calibration table entry a = 512 c + 256 v + code (channel c = 0..9, transitions to level v): [27:16] the bin's width, [15:0] its left edge, both in 1/16 tick. Power-up: 160 equal bins
0x10000 + 4a	HIST	R	calibration histogram bin a (same index; 32 bits)

Ring writer and merger

Offset	Name	Access	Content
0x080	W_CTRL	RW	[0] enable (a rising edge restarts: the written position 0, base and size taken)
0x084	W_BASE	RW	the ring's physical address (4 KiB aligned)
0x088	W_SIZE	RW	its size in bytes (a multiple of 128, at least 8 KiB); data after the first 4 KiB
0x08C	W_RD	RW	the consumer's position (bytes mod 2^32); every write is the watchdog's heartbeat
0x090	W_WD	RW	watchdog: the writer stops when W_RD was not written for this many clocks (0: off)
0x094	W_WR	R	the written position (bytes mod 2^32; advanced by the bursts' write responses)
0x098	W_BURSTS	R	data bursts written
0x09C	W_STALLS	R	clocks with a burst waiting for room in the ring
0x0A0	W_LEVEL	R	64-bit beats queued (2048 at most)
0x0A4	RELEASED	R	events released into the lookup (inside acquisitions)
0x0A8	DISCARDED	R	events released outside an acquisition
0x0AC	QUEUE	R	events in the merger's queue
0x0B0	DROPS_TOTAL	R	the sum of the sources' DROPS
0x0B4	STALE	R	events of an acquisition lost because the output was blocked: a FIFO head older than 16384 samples (65 µs) is discarded, never timed with a wrapped sample number
0x0B8	MARK_LEVELS	R	[7:0] the marker pins' levels now (DIO0_N..DIO7_N), [8] their PLL locked
0x0BC	SORT_LATE	R	lane events the coincidence engine's sorter emitted older than an earlier one (out of time order: skipped by the engine; THEORY.md "Coincidences"): 0 below about 65 Mevents/s in all, unless a recorded lane was held back by a full ring
0x0C0	ENGINE_ONLY	RW	[12:0] per source (0..9 channels, 10, 11 the generator, 12 the markers): its events go to the coincidence engine only, not into the records
0x0C4	FILTER	RW	the conditional filter: [9:0] trigger channels (bit i: channel i + 1), [20:16] window N in samples (0..30): an event of a channel with CFG [6] enters its FIFO only if a trigger channel had an event at a sample in [k, k + N] of its sample k
0x0FC	CONSTS	R	{taps per cell 2, cells 80, channels 10, layout 1} from the high byte down

The counters run at the sample clock and are read synchronously (both clocks come from one MMCM); a read takes five clocks.

Coincidence engine

Offset	Name	Access	Content
0x300	COINC_CFG	RW	[11:0] lanes (bit i: source i, 0..9 the channels, 10 and 11 the generator's two; at most 8, 7 with a gate: the sorter holds 8 events), [12] enable, [16] gated by a marker, [19:17] its pin (0..7), [20] invert (the gate opens at the pin's falling edges, else rising)
0x304	COINC_WINDOW	RW	the window, ticks (24 bits: up to 16.4 µs): an event joins the open cluster within it of its start
0x308	COINC_CTRL	W	[0] clear the patterns (4096 clocks), the masks' counts, CLUSTERS and VIRTUAL_DROPS

Offset	Name	Access	Content
0x30C	COINC_STATUS	R	[0] clearing, [1] a cluster is open, [2] the gate is open
0x310	CLUSTERS	R	clusters closed (32 bits)
0x314	VIRTUAL_DROPS	R	virtual events lost: the queue (16 clusters) full
0x320 + 16 i	MASK i	RW	mask i = 0..7: [11:0] lanes, [12] exclusive (the pattern equals the lanes; else it contains them), [13] virtual (a record, source 20 + i, at the cluster's start)
0x324 + 16 i	COUNT i	R	the clusters mask i matched, low 32 bits; reading it latches the high 16 bits
0x328 + 16 i	COUNT_HI i	R	the high bits latched
0x20000 + 8 p	PATTERN p	R	clusters with exactly the lanes p (p = 1..4095), low 32 bits; reading it latches the high 16
0x20004 + 8 p	PATTERN_HI p	R	the high bits latched

The counts are 48 bits: at 100 million clusters per second they last 32 days.

The ring in DDR

The service's worker allocates the ring (contiguous memory) and gives its address and size. The first 4 KiB is the **mailbox**, written by the FPGA at the start, every 64 bursts and 100 µs after the last burst if anything was written; its 32-bit words: 0 magic 0x54544D42 ("TTMB"), 1 sequence, 2 the written position (as W_WR), 3 [0] tripped, 4–5 the sample of the mailbox (48 bits), 6 DROPS_TOTAL, 7 W_BURSTS. The data area follows: 128-byte bursts of records (docs/PROTOCOL.md "Records"), filled in order and wrapping at the end; a burst is written only when the consumer's position leaves room for it. After 100 µs without records a partial burst is completed with NOP records.

timetagger verification records

Newest first. Unmeasured quantities are marked as such.

2026-11-05 (evening): the coincidence engine, its gate and the conditional filter on awg's patterns

- **Setup:** timetagger 2.0 (0x9b2ec69e) loaded on 192.168.0.102 (the board then ran counter); awg 3.0's D0, D1, D2 to channels 1, 2 and marker 1, wiring as below. Pattern: 1024 blocks of 32 words; a sync on channel 1 at word 0 (p = 0.7), a click on channel 2 8–120 ns later (p = 0.5), a second click 160–224 ns after the sync (p = 0.2); marker 1 high over half of the blocks, its edges 16 ns from any channel edge; no two outputs changing in one word; per pass 705 syncs, 717 clicks. Receipt: `../counter/docs/data/board-1.0-awg-tests-2026-11-05.json` (`t3_timetagger`).
- **Engine** (channels 1, 2, W = 20 ns; masks [1,2] inclusive with virtual channel 21, [1] exclusive, [2] exclusive, [1] inclusive), 2000 passes: patterns "1" 1312000, "2" 1336000, "1,2" 98000, the four masks and the 98000 virtual events in the records exactly as `analysis.coincidences` on the pattern's events gives; `late` 0, `virtual_drops` 0.
- **Gated by marker 1** (both edges enabled): patterns 668000, 662000, 46000 and the masks exact against the events inside the gate; the marker's 506000 edges in the records.
- **Conditional filter** on channel 1 (triggers [2], 60 ns: N = 15 samples, so that awg's 8 ns grid never meets its edge): 340000 syncs kept = 2000 × 170 (those with a click 8–56 ns later), every click recorded; the engine sees the filtered channel. In this first run one pair of 98000 was split into "1" and "2" (97999 pairs); four further runs of 4000 passes with the filter (784000 pairs) and four without split none. Not explained; no timestamp outlier was found: 5.7 million events of channels 1, 2 against their neighbours (within 2 µs) lie within 0.59 ns of awg's grid (rms 87–225 ps with the pattern-dependent edge shifts of the jumpers).

2026-11-05: awg on a second board (channels, markers, sorting by markers)

- **Setup:** timetagger 2.0 (0x9b2ec69e) on 192.168.0.102, calibrated at the start; awg 3.0 on 192.168.0.104: its D0, D1 to channels 1, 2 and D2, D3 to markers 1, 4 (Dupont, about 20 cm); grounds joined. Receipt: `guides/combining/data/awg-timetagger-2026-11-05.json`.
- **Markers against channels:** awg's D2 16 ns after D0: marker 1 minus channel 1 16.008 ns on average over 1000005 events, 290 ps rms (the markers' 1 ns quantization: 289 ps).

- **Sorting by markers:** awg's sweeps (SEQ on marker 4, a TRIG per point on marker 1, k + 1 pulses on channel 1 at point k, 8 points, 32 ns words) for 5 s: 305177 sweeps, 2441414 points sorted on the PC from the tags, every one with its exact count, 10986385 channel-1 events.
- **awg's outputs** (awg's VERIFICATION 2026-11-05): a 1 us period 5.02 ppm short in this board's time, 30.0 ps rms per period; D1 20 ps after D0.
- **Crosstalk:** D0 and D1 rising together fired markers 1 and 4 (whose wires were idle low) and hundreds of thousands of marker glitches per second; markers have a glitch filter (under 4 ns) but no hold-off: keep marker lines apart from fast pulse lines, or idle them high.

2026-10-31: the user's manual, session edges (build 0x9b2ec69e)

`tools/record_manual_data.py --session edges --label ... --wave ... --vpp 3` (evening, die about 48 °C, fan on): an SRS DS345 (0 to 3 V at the pins: 3.00 Vpp, 1.50 V offset into 50 Ω) through one coax to a BNC fork adapter on E1 pins 15–26: one 50 Ω termination at the socket, fanned out to pins 15 and 17 (channels 7 and 8), ground on pins 25 and 26. Per setting about 10^6 edges per channel (1 s at 1 MHz, 33 ms at 30 MHz); the difference channel 8 − channel 7 on the same edges (the source's jitter cancels; one channel = rms / √2); `docs/manual/data/edges-*.json`.

- Square, 1 MHz: one channel 15.2 ps rms rising, 22.2 falling (toggle input), 16.2 and 18.3 (level input); with dither 20.0 ps (rising). The difference's mean flat over the 4 ns period to 6.7 ps rms. 10 and 30 MHz: 14.9–15.1 ps rising, 20.5–22.1 falling; 30 MHz is 60 Mevents/s on the two channels (33 ms, through the ring), no drops, no flagged events.
- The DS345's own period jitter (channel 7's intervals): 1027 ps rms at 1 MHz (rising edges), 108 ps at 10 MHz.
- Sines (slew rate π f Vpp at the threshold): 9.5 mV/ns (1 MHz) 155 ps rising / 85 ps falling; 28 mV/ns 59–96 ps; 94 mV/ns 28–54 ps; 283 mV/ns 15.2–19.3 ps. The extra jitter equals 0.8–5 mV rms at the inputs over the slew rate. On the 1 MHz sine the difference's mean swings by 163 ps rms once per 4 ns (about 1.5 mV of the sample clock at the inputs), and the rms of an acquisition changes in steps (3 MHz: 91 → 66 ps within one; 59–105 ps between repeats); the channels' thresholds differ by about 5 mV (the mean moved by 560 ps against the square wave's).

2026-10-31: the user's manual, sessions gpsdo and temp (build 0x9b2ec69e)

The bench: a GNSS-disciplined oscillator (BG7TBL, front label 2018-06-05) with its 10 MHz sine on IN2 (the meter, 10 ms windows, amplitude about 0.45 of full scale) and its PPS on channel 7; a function generator's 1 MHz square on channel 1 (divider 1000).

gpsdo, warm-up (09:37–11:39, the GPSDO switched on minutes before; die 47.1–48.2 °C, fan on): `tools/record_manual_data.py --session gpsdo --hours 2`, raw stream `~/board-data/timetagger/gpsdo-20261031-093748.ttbin` (1.0 GB), kept as `docs/manual/data/gpsdo_warmup.json`. The GPSDO's PPS stepped against its 10 MHz by +200 ns at minute 40, then moved by a few ns for about 15 s (15 intervals more than 1 ns off); a check before saw a 208 ns jump and the rising edge moving 2.6 ns/s. Otherwise as below (crystal −4.43 ppm, PPS through the meter 38 ps rms, generator jitter 118 ps rms).

gpsdo (12:30–14:32, the GPSDO on for five hours, its display reporting lock; die 47.4–48.8 °C, fan on): raw `~/board-data/timetagger/gpsdo-20261031-123052.ttbin`, reduced to `docs/manual/data/gpsdo.json`: 7208 PPS, 7218 meter seconds; one acquisition armed on the PPS.

- The first PPS tagged at −6.25 ns (the armed start). No PPS steps.
- The crystal −4.50 ppm against the GPSDO; its frequency moves by 158 ppb (10 s averages) and its time error by 81 µs (mean frequency removed), following the die's temperature (the room's, with the fan on: 0.6 °C in 2 min averages) at −139 ppb/°C (correlation −0.71).
- The PPS intervals are 4491 ns short in the board's time, varying by 26.4 ns rms; through the meter they are 1 s to 35 ps rms (ADEV $4.3\cdot10^{-11}$ at 1 s, $4.1\cdot10^{-13}$ at 100 s, white phase noise).
- The generator +3.19 ppm against the GPSDO, its edges' jitter about 141 ps rms.
- ADEV against the GPSDO: crystal $1.8\cdot10^{-9}$ (1 s), $5.4\cdot10^{-9}$ (100 s), $1.9\cdot10^{-8}$ (1000 s); generator $8.8\cdot10^{-10}$, $3.7\cdot10^{-9}$, $1.7\cdot10^{-8}$.
- Three-cornered hat: the crystal and the generator against the GPSDO follow the same ±50 ppb curve (60 s frequencies correlated 0.90), against each other 63 ppb; the hat gives the GPSDO $1.7\cdot10^{-8}$ at 1000 s (the crystal $8.0\cdot10^{-9}$, the generator negative). Both recordings gave the same. **The hat is wrong here**: session temp shows the crystal's temperature coefficient, which accounts for its wander, and the generator against the GPSDO not moving; the crystal and the generator both follow the room with nearly the same coefficient, and the hat gives their common wander to the GPSDO.

temp (19:27–20:07, die 48 → 71 → 48 °C: the fan off for 20 min, then on): `--session temp --minutes 40`, 92 steps of a 2 s calibration, the test signal on channels 1, 2, 5, 9 and a 10 s look at the crystal (meter, 1 ms windows) and the generator; `docs/manual/data/temp.json`.

- The crystal −128 ppb per °C of the die against the GPSDO and against the generator alike (−2.78 ppm over the run); the generator against the GPSDO +0.3 ppb/°C (6.4 ppb rms).
- The mean bin −0.021 %/°C (polarity tables −0.028 and −0.015), the ring oscillator −0.018 %/°C, the delays' scale λ 0.060 → 0.075 (+0.64·10⁻³/°C).
- Recalibrated at every step, channels 2, 5, 9 against channel 1 move by −0.02, −0.07 and −0.15 ps/°C (residuals 0.8–1.6 ps rms).
- The die back within 1 °C 4 min after the fan was switched on.

2026-10-30: the user's manual, session base (build 0x9b2ec69e)

`tools/record_manual_data.py --session base` (22:39–22:44, die 48–49 °C, fan on; the existing jumpers, the function generator's 1 MHz on channel 1 over header wires), reduced to `docs/manual/data/*.json` and `docs/manual/values.tex`:

- code density: 128–139 codes per 4 ns, mean bins 28.8–31.2 ps, largest 68–110 ps, rms quantization 10.9–13.6 ps; channel 1's widest bin 1.5 mean bins above the mean (DNL), INL 72 ps peak to peak;
- the test signal: one channel 9.1–12.1 ps rms (from the pairs' rms), pairs 5.6–29.5 ps rms (median 15.2), the two calibrations agree to 5.5–13.8 ps rms over the period; asynchronous edges (the generator) 8.2 ps rms over the period (29 ps peak to peak);
- connector delays 3.25–3.89 ns, the delays' scale λ 0.062 (39 ps rms); polarity offsets −57..+21 ps (toggle paths), −358..−211 ps (the test signal's level path);
- the SATA loop 16.9–18.6 ps rms against an internal channel; marker 1 against channel 1 265 ps mean, 291 ps rms;
- the stream 7 Mevents/s sustained (20 s), 10 Mevents/s in 2 s bursts, 0.4 % lost at 8;
- the engine's patterns equal the PC's on 320 814 clusters; the conditional filter on the ring oscillator (97.6 MHz, divided by two) with the test signal's edges as triggers and a 120 ns window: 6.1 events kept per trigger, nothing dropped.

2026-10-30: verify_build (build 0x9b2ec69e)

`common/tests/verify_build.py --design timetagger`: **PASS, all 8 stages** in a source-only copy (creation from the recipe 580 s, unchanged-build reuse 45 s, missing-export recovery 46 s, a vendor HDL edit 562 s, the design RTL edit of `tests/verify_build.json` 603 s, a failed build blocks the upload 171 s, cleanup, regeneration from the saved project 551 s); the build ID 0x9b2ec69e is reproduced, and the copy's `timetagger.json` equals the uploaded one but for its build time (version 2.0, the delays of all ten channels).

2026-10-30: version 2.0 (build 0x9b2ec69e, the current one)

Build 0x993102ce (below) with the sysid version 2.0 (`design.tcl`, the driver's VERSION): the same timing (WNS +0.064/+0.188 ns, WHS +0.028/+0.017 ns) and utilization. `scripts/test_hw.py --external --sata --markers`: **PASS, 89 checks** (`tools/recordings/hw20-20261030/ test-hw-0x9b2ec69e.txt`, `hw10-20261030-191450.json`), die 49 °C; start and stop exact on 1 502 434 events, the engine on the test signal 19 531 of 19 531 clusters, engine-only at 60 Mevents/s 815 late of 6·10^7 (reported), the delays' scale λ 0.059 (39.2 ps rms), marker 1 against channel 1 291 ps rms, the SATA loop 18.3 ps rms (channel 9, rising).

After `make install-service` the service did not load the new build by itself: it attached to the FPGA's old build (version 1.0) and stopped at the major-version check, which comes before the build-ID comparison that reloads (`rpfpga.overlay`); a `LOAD timetagger` loaded it. Fixed afterwards in the framework: an attaching driver now compares the build ID, and loads the uploaded files, before the identity checks (`common/tests/test_overlay.py`, `test_upload_of_a_new_major_version_loads_it`).

2026-10-30: 2.0, the coincidence engine and the conditional filter (build 0x993102ce)

New since step 1 (the scope decided on 2026-10-29: no FPGA correlation engine, g2 and TCSPC are computed on the PC):

- the coincidence engine: an 8-entry sorter after the lookup (`tt_sort`), fed only with the engine's lanes (channels 1–10 and the generator's G1/G2, at most eight at once) and released by a horizon from the lanes' overdue times; clusters as `analysis.coincidences`; 4096 patterns of 48 bits; eight inclusive or exclusive masks; virtual events (channels 21–28); engine-only sources (beyond the stream's rate); a gate on a marker pin;
- the conditional filter on channels 1 and 2: an event kept only if a trigger channel fires within N ≤ 31 samples after it (an 80 MHz laser sync reduced to the detected periods);
- the acquisition's acceptance corrected (below).

Build: WNS +0.064 ns (clk_125) and +0.188 ns (clk_250), WHS +0.028 and +0.017 ns (the post-route phys_opt report; the `_routed` report is written before that step); 64.9 % LUTs, 45.0 % flip-flops, 98.6 % slices, 53 DSP, 36.5 BRAM, both MMCMs and one PLL. Two paths were shortened without changing any result: the horizon's thermometer is registered and encoded in the clock that sets it, and the engine's lost virtual events are counted from registered flags. `make sim`: PASS (`tb_sysid`, `tb_core` against the model, `tb_coinc`, and the new `tb_accept`). `verify_build` was not run for this build.

The acquisition's start and stop (found on the board, fixed in this build). The merger compares each event's sample with k0 and the stop sample through registered ages, which follow a START or STOP two clocks after the flags `running` and `stop_set` do. In those two clocks a start accepted every event that fell due, from about 1280 samples (5.1 µs) before the new k0, and a stop discarded every one that fell due, although inside the acquisition and without counting it as lost. The old acquisition's ages are clamped (2^15 − 1 samples), so nothing else masked it. On the board it showed once in run 3 (below) as a lone channel-1 cluster; with events due in nearly every clock it is frequent: the generator at 10 Mevents/s over 40 acquisitions of 5 ms gave 5 events before the start and 5 pairs split around the driver's stop write (about 1 ms after the start), on build 0xe57df917 (`tools/recordings/hw20-20261030/diag-0xe57df917-acceptance.txt`). Now nothing is accepted in the two clocks after a start (every head due then lies before the new k0) and `stop_set` enters together with the stop sample's age. `tests/tb_accept.sv` makes a head due in each clock from three before to four after a start and a stop: against the old merger it fails in exactly the two clocks, for both. The model (which always accepted `start ≤ k < stop`) now also clears the stop at a start, as the hardware does; the vectors did not change.

`scripts/test_hw.py --external --sata --markers`: **PASS, 89 checks** (`tools/recordings/hw20-20261030/test-hw-0x993102ce.txt`, `hw10-20261030-182132.json`), die 48.6 °C (a fan is now fitted: 72 °C before), the board on a new SD card (the old one returned corrupted reads; the tagger's records never touch the card). Wiring as on 2026-10-29. New and changed checks:

- start and stop exact: the generator at 10 Mevents/s, half paired 10 ns later, 20 acquisitions of 5 ms, 1 500 608 events: none before the start, none after the stop, no pair split;
- the coincidence engine against `analysis.coincidences` on the same records: generator pairs recorded at 2 Mevents/s (319 744 clusters, equal patterns, the masks and 159 624 virtual events at the clusters' starts, none late); the test signal on channels 1–8 (19 531 clusters of all eight, equal, none late); gated by marker 1 (500 004 gate periods, equal but for one event on a gate edge's tick, which may count either way);
- engine-only generator events (not recorded; patterns and clusters per second as a simulation of the generator's draws within 5 σ): 15 Mevents/s (G1 10) none recorded and none late; 30 (G1 20) none late; 60 (G1 40) 915 late of 6·10^7 (one merger source near its limit holds the horizon), reported; 90 overflows G1's FIFO, not judged. The check of "none recorded" counts decoded events now (before, a raw count of record words by their source bits took an EPOCH's k0 word for a generator event);
- the conditional filter: the ring oscillator (98 Mevents/s) on channel 1 kept within 31 samples before the test signal's edges on channel 2: nothing dropped or lost, 12.10 kept per trigger as the ring's rate gives, all 236 358 inside the window;
- the 1.0 checks and the markers as before: the calibration of all ten channels (mean bins 28.8–31.0 ps, rms quantization 10.8–13.6 ps), the delays' scale λ 0.059 (38.9 ps rms), the test signal's channels against channel 1 (means −5.0..−0.1 ps), polarities within 11 ps, the SATA loop 15.7–19.3 ps rms, marker 1 against channel 1 291 ps rms.

The highest lossless stream rate is 5 Mevents/s, as in the earlier runs. The ring took 32 MB on the new card (96 MB before): the larger sizes did not fit in the contiguous memory then (128 MB, partly taken by the display driver at overlay loads); the backlog it holds is shorter, the rate the same.

Earlier runs on build 0xe57df917 (the old card, 72 °C): run 1, four failures (the generator's count, 431 688 of 10^6, not seen again; three engine-only checks whose simulation and judging were then corrected); run 2, 85 of 88 (ten lanes firing together overflow the 8-entry sorter: 1808 of 19 531 clusters incomplete; hence at most eight lanes, seven with the gate, enforced by the driver; and a gate edge on an event's tick); run 3, 86 of 88 (`test-hw-0xe57df917.txt`, `hw10-20261030-024135.json`): the EPOCH word counted as a generator event and the acceptance race above.

2026-10-29: 2.0 step 1, the markers (build 0xfd6932c8)

New: the eight markers on DIO0_N..DIO7_N (`tt_mark`: a PLL at 250 MHz 0°/90°, an ISERDESE2 per pin at 1 ns, events at the last change of each 4 ns window, glitches counted, a two-batch serializer into the merger's source 13 of 13); the merger widened to 13 sources. To keep the margin (WNS ≥ +0.05 ns, agreed for 2.0), three paths were shortened, none visible to the model: the merger's accepted pick from a per-source "goes to the lookup" bit, the markers' drop count from the registered popcount, and a channel's divider change detected a sample earlier; a channel's FIFO entry is loaded every sample (no enable fanning out). WNS +0.160 ns, WHS +0.010 ns; 50.8 % LUTs, 41.0 % flip-flops, 91.8 % slices, 42 DSP, 25.5 BRAM, one PLL, 22 ILOGIC. `make sim`: PASS (the scenario's markers: every edge mode, a pin off, 1.5 ns glitches, a burst on all pins: 3010 events, 233 glitches, 25 drops, as the model). The notebook panel builds on the board.

`scripts/test_hw.py --external --sata --markers`: **PASS, 73 checks** (`tools/recordings/ hw20-20261029/test-hw-0xfd6932c8.txt`, `hw10-20261029-054531.json`), die 71 °C, ring oscillator 95.7 MHz (132.7 MHz in the build before: its routing inside the pblock changed; the check keeps it away from the clock's subharmonics). Wiring as before plus a wire jumper from DIO0_P to DIO0_N, so marker 1 and channel 1 see the second board's edges:

- the markers' PLL locked; a marker-1 rising event for every one of 1 000 010 channel-1 edges (and as many falling), no glitch, no drop;
- marker 1 − channel 1 on the same edges: **290 ps rms** (the 1 ns bins: 289 ps) and, before the correction, +12.705 ns (the markers' samples reach clk_250 about three samples after they are taken: `driver.MARKER_DELAY_S`, subtracted with `connectors`; after it 19 ps);
- the channels as before: calibration, polarities, delays' scale λ 0.082 (47.1 ps rms), the phase steps 11.9 ps rms, the test signal's agreement, the SATA loop, the generator, the acquisitions.

The jumper (a wire loop on E1's header pins) degrades the source's edges: the intervals then alternate by about 1 ns and spread 178–306 ps rms per edge, with edges missing or extra (182–271 per million). The external phase test is therefore only reported when the jumper is on: 11.3 ps in this run; 352 and 305 ps in two runs before, where the check's moving average met such edges (`hw10-20261029-052702.json`; the same edges analysed with per-bin medians, flat to ±10 ps: `diag-0xfd6932c8-external.txt`). Without the jumper it stays a check (5.9–7.6 ps on two builds).

2026-10-28: verify_build (build 0x2a8ff3d5)

`common/tests/verify_build.py --design timetagger`: **PASS, all 8 stages** in a source-only copy (creation from the recipe 517 s, unchanged-build reuse, missing-export recovery, a vendor HDL edit 476 s, the design RTL edit of `tests/verify_build.json` 479 s, a failed build blocks the upload, cleanup, regeneration from the saved project 478 s); the build ID 0x2a8ff3d5 is reproduced, and the copy's `timetagger.json` carries the delays of all ten channels (the common build's `design_export` hook, new since the last run).

2026-10-28: connector delays, polarity offsets, the ring oscillator (build 0x2a8ff3d5)

Changes since 0xf47cf07e: each channel's input stage in a hard pblock beside its line; Vivado's delays of every path exported with the bitstream (`design_export`, `timetagger.json` "delays"); the POLARITY register (+0x1C: the offset of a line's falling transitions) and `calibrate_delays()`; the connector correction in OFFSET (`connectors`); the merger's round-robin mask as a prefix OR; the ring oscillator with 17 stages in a hard pblock. WNS +0.060 ns, WHS +0.020 ns; 48.7 % LUTs, 39.9 % flip-flops, 88.4 % slices. `make sim`: PASS (the vectors with polarity offsets on four channels, −2048 and +2047 among them). Build 0x2a8ff3d5 is 0x4022b20f with two comments corrected (the same netlist and timing).

`scripts/test_hw.py --external --sata`: **PASS, 69 checks** (`tools/recordings/hw10-20261028/ test-hw-0x2a8ff3d5.txt`, `hw10-20261028-194650.json`), die 71 °C, ring oscillator 132.7 MHz:

- code density: mean bins 28.6–30.5 ps, largest 68–110 ps, rms quantization 10.6–12.4 ps;
- channel 1 against the test signal's phase steps: 18.08 ps/step, **9.9 ps rms** residual (the MMCM's own error included; about 20 ps before);
- every channel against channel 1 on the test signal at 8 phases, after the corrections: means **−3.1 to +0.4 ps** (rising and falling edges), spread over the phases 2.9–15.4 ps rms, jitter at one phase 3.1–29.3 ps rms;
- polarity offsets measured: the toggle paths −34 to +19 ps (the stages beside the lines; 316–421 ps before, when they sat in the middle of the die), the test signal's level path −357 to −197 ps; after the correction the toggles' alternate transitions agree to −11.1 to +9.4 ps at one phase (the check allows 30 ps: there each polarity's table error enters; 23.5 ps on channel 10 in a run before);

- the delays' scale λ = 0.023 (the mean of the two edges' delays fitted to the model's test paths: **48.0 ps rms residual**; the rising edges alone −104 to +101 ps per channel, which the test inputs take from the measurement instead);
- SATA loop 13.9–19.5 ps rms; channel 10 − channel 9 = −320 ps, the S1 outputs' model −241 ps (the board's traces and the cable pairs are not modelled);
- the second board's 1 MHz edges on DIO0_P: their time error does not depend on the edge's phase in the 4 ns period (40 bins, largest mean **5.9 ps** over 10^6 asynchronous edges): the calibration holds for any arrival. The rest is both boards' jitter together, 110 ps rms per edge (155 ps over 8 periods), and the source alternates between consecutive intervals (also at a level input, where every transition has the same polarity: `diag-0x4022b20f-external.txt`);
- the armed start, the generator, dead time, divider, acquisitions and the stream as before.

The same on 0x4022b20f: PASS, 69 checks (`test-hw-0x4022b20f.txt`, `hw10-20261028-193015.json`; λ 0.020, 47.8 ps rms, the phase steps 9.8 ps rms, the external phases 7.6 ps).

The ring oscillator was pulled by the sample clock in the build before (0x0999a501, the input stages placed, the 13-stage ring left to the placer): it ran at 124.9–125.0 MHz instead of 119 MHz, and the tables came out wrong by up to ±150 ps although the bins looked normal (mean 28.8/30.8 ps, largest 66–112 ps, rms quantization 10.5–13.4 ps). The test signal showed it: channel 1 against the phase steps 84 ps rms, the channels' spread over the phases 9–150 ps rms. 22 of 69 checks failed: 13 of the channels' agreement over the phases, 7 of the toggles' polarities (their offsets measured through biased tables), and two checks corrected since (the armed start did not allow for the connector delay; the external edges' spread is mostly the source's): `test-hw-0x0999a501.txt`, `hw10-20261028-190341.json`. Recalibrating with ring dividers 5, 6, 7 and 9 gave the same pattern (`diag-0x0999a501-ring.txt`): the ring itself, not its sampling. Its edges' phase lingers near the locking point and slips quickly elsewhere.

Before the placement (build 0xf47cf07e, `delays-0xf47cf07e.json`), Vivado's delays had already been compared with that build's records: λ 0.28–0.35 and 50–56 ps rms residuals for the test paths, against their spread of 210 ps; the pins' paths were 8.1–10.1 ns at the slow corner, 1.9 ns apart (4.5–6.6 ns placed).

Found on the way: negative channel offsets were refused on the board (the readback is sign-extended; the driver wrote 22 bits); fixed, and the tests' fake core reads back so.

2026-10-28: 1.0 on the board (build 0xf47cf07e)

Two RTL changes since 0xf09c5aa0: the calibration histogram looks at a random channel each clock (no beat against periodic inputs: live calibration is unbiased also for fast clock-like signals), and the drop total is summed in two stages; each FIFO gives the merger a fabric copy of its head's sample (the last timing path). WNS +0.009 ns, WHS +0.009 ns. The framework now imports PYNQ and parses an overlay outside the board lock: another process constructing its driver blocked every hardware transaction for 5.18 s (the ring writer's watchdog tripped), now at most 77 ms (measured with a 10 ms heartbeat while another process loads and calibrates).

`scripts/test_hw.py --external --sata`: **PASS, 57 checks** (`tools/recordings/hw10-20261028/ test-hw-0xf47cf07e.txt`, `hw10-20261028-073736.json`), die 70 °C, ring oscillator 119.0 MHz: mean bins 28.6–31.0 ps, largest 77–120 ps, rms quantization 11.0–13.1 ps, no flagged hit; two channels' jitter at one phase 6.9–22.0 ps rms; their calibration agreement over the period 3.0–13.8 ps rms; the SATA loop 13.2–17.5 ps rms; the generator, dead time, divider, acquisitions and external edges as before. The per-phase jitter limit is 30 ps (an edge straddling a wide bin alternates between two bin centres: 26.8 ps at one phase of one pair in a run before).

2026-10-28: verify_build (build 0xf09c5aa0)

`common/tests/verify_build.py --design timetagger`: **PASS, all 8 stages** in a source-only copy (creation from the recipe 519 s, unchanged-build reuse, missing-export recovery, a vendor HDL edit 508 s, the design RTL edit of `tests/verify_build.json` (the encoder's last tap) 479 s, a failed build blocks the upload, cleanup, regeneration from the saved project 501 s); the build ID 0xf09c5aa0 is reproduced.

2026-10-28: ttcore (the host library)

`tools/ttcore` (C++17, pybind11 3.1): `ctest` (every measurement against brute force on random events, the decoder on every split of hand-made records, the sorter, recordings) and `tests/test_ttcore.py` (against the numpy references: the scenario's 3616 words decoded alike, correlation and g2, start-stop, coincidences, counters, the sorter): pass. On the board's stream (the ring pump, from the PC): the generator at 5 Mevents/s with half paired

10 ns later for 3 s: 22 502 245 events, no byte lost, the correlation's peak exactly at +10 000 ps with 7 500 780 pairs (g2 2000), and the 90 MB recording read back alike.

2026-10-28: the ring pump (build 0xf09c5aa0)

The Python worker drained the ring at about 7 MB/s: one thread at 94 % CPU (copies, framing, the asyncio hand-off). With the compiled ring pump (`common/python/rpfpga/ringpump.c`, built on the board by the worker; port 5026), measured from the PC with `RecordStream`: the generator at **5 Mevents/s for 4 s: all 20 001 002 events, 20 MB/s in real time**; at 10 Mevents/s for 4 s: all 39 989 957 events, delivered at 26.7 MB/s (6.7 Mevents/s sustained, the ring holding the rest); at 20 Mevents/s the ring fills and the FPGA counts its drops and stale events. The board test passes again (57 checks) with the pump in the path. `common/tests/test_ringpump.py`: frames, the wrap, the time context, a slow client's lost frames, a restart. Found meanwhile: another process constructing its driver holds the board lock for 1.3 s, longer than the writer's 1 s watchdog (two trips in a test run, `status 0x55`): the watchdog is 5 s; no trips since.

2026-10-28: 1.0 on the board (build 0xf09c5aa0)

`scripts/test_hw.py --external --sata` through the service's stream: **PASS, 57 checks**. Wiring: a second Red Pitaya running awg on DIO0_P (1 MHz edges), a straight SATA cable from S1 to S2. Die 70 °C. Records in `tools/recordings/hw10-20261028/` (`test-hw-0xf09c5aa0.txt`: the run's output; `hw10-20261028-062520.json`: every number).

- **Build:** WNS +0.104 ns (clk_125) and +0.107 ns (clk_250), WHS +0.013 ns; 8259 LUTs (47 %), 13 556 flip-flops (39 %), 25 BRAM tiles (42 %), 39 DSP48 (49 %), both MMCMs; 3785 slices are touched (86 %: loosely packed, the LUTs and flip-flops are below half). DRC: warnings only (the allowed ring loop, the pin-clocked toggles' gated-clock note, DSP pipelining hints).
- **Ring oscillator:** 128.6 MHz (118.9 MHz in 0.1's placement).
- **Code density** (4 s, the ring divided to 1 M edges/s per channel; about 2 M hits per channel and polarity): codes 2–141 per period, mean bins 28.6–31.0 ps, largest 62–90 ps, rms quantization 10.2–12.5 ps, no flagged hit among 4.3 M ring edges per channel.
- **Calibration agreement** (the test signal on all channels at 8 phases over the period, without dither, channel n − channel 1): the mean difference varies by **3.5–14.9 ps rms** over the period (channel 10 the most, 14–15 ps; most 5–9 ps). The precision of two channels at one phase, which is their jitter: **9.9–23.7 ps rms** (the larger values at phases where the edges straddle a wide bin's boundary). The constant part (47 to 671 ps) is each channel's path from the test signal, which differs from its pin's.
- **Found and fixed on the board:** with denser ring edges (16 M/s per channel) the histogram, which looks at one channel every 10 clocks, samples the edges with a beat against the ring's period: the tables came out biased by ±100 ps (the same measurement then gave 15–80 ps rms). `calibrate()` now divides the ring to about 1 M edges/s, when every hit is taken.
- **Polarity offsets** (toggle inputs on the test signal): the two polarities of each channel's toggle path differ by 316–421 ps; calibrated per channel by the host (not yet in 1.0's FPGA).
- **Generator through the stream:** 1 Mevent/s for 1 s: 1 001 053 events, 500 404 pairs all exactly 10 ns apart (±1 ps of rounding), an EPOCH first, all within the second, nothing lost. Lossless at 2, 5 and 10 Mevents/s for 1 s (the ring holds the backlog; the board's Python worker drains about 7 MB/s, 1.75 Mevents/s, sustained); at 20 Mevents/s 1.9 % dropped in the FPGA (counted).
- **Dead time** 1 µs on 640 ns edges: exactly every other edge (78 125 of 156 250); **divider** 4: a quarter (39 063). **Start at a sample**: k0 is that sample; **duration** 0.3 s: the last event at 0.29956 s; **armed** on channel 1: its first event at −3.0 ns (within one sample of t = 0).
- **SATA loop** (the test signal out on S1, into channels 9 and 10, against channel 1 on the internal test signal; no dither): 17.8–28.6 ps rms, delay 7.5–8.1 ns (the cable and buffers).
- **External edges** (awg on DIO0_P, channel 1): 1 000 010 edges in 1 s, intervals of 0.999990 µs (the two crystals differ by 10 ppm) with 468 ps rms spread: the awg's edges themselves (its 1 ns output grid), not this board's.
- **Also found on the board, fixed before this run:** a stall of the output longer than 131 µs (the writer not yet started at boot) left FIFO heads whose 16-bit sample numbers had wrapped: they were never due again and every FIFO stayed blocked (the model did the same). Heads older than 65 µs are now stale (counted, never timed) and discards never wait for the output (model and RTL, `tests/test_model.py`). The worker's heartbeat could miss the watchdog while another process held the hardware or while it copied a large backlog from uncached memory: the histogram is now read in pieces, the ring is cacheable (invalidated region by region), the worker copies at most 256 KiB between heartbeats and restarts a stopped writer.

2026-10-28: 1.0 offline

- `make sim`: `tb_core` PASS (”time tagger core”). The scenario of `tests/vectors.py` (ten channels on behavioural delay lines of 30 ps taps: level and both toggle inputs, accepted edges, a dead time of 50 samples, a divider of 4, a later hit cell, a disabled channel, the SATA pairs, live calibration, offsets from −3000 to +123456 ticks, custom table entries, dither; the generator at 1/64 per sample with half of its events paired 40 ns later; an acquisition between two programmed samples; 6352 edges) runs through the RTL with a random AXI3 DDR model; the ring’s 3616 words (3552 events and the control records) equal the model’s word for word, and so do every channel’s event, flagged and drop counters, the generator’s drops and the merger’s released events.
- `make test`: 22 timetagger tests (the encoder, channel logic and writer of the model, a blocked output that must not strand a FIFO head, the vectors current; configuration packing and derivation, invalid settings, presets, acquisitions, tables; the fast decoder against the model, decoding in pieces, the reference measurements; the stream’s frames and time context, a slow client losing whole frames; the ring worker draining a wrapping ring and its heartbeat, the live calibration’s guard; the TCP commands, the client and RECORDS STREAM through a real server; loading) and the uniformity tests.

2026-10-27: 0.1 prototype on the board (build 0x152a286a)

`make test-hw` with `--external --sata` (scripts/test_hw.py): **PASS, 13 checks**. Wiring: a second Red Pitaya running awg on DIO0_P (edges about every 500 ns), a SATA cable from S1 to S2. Die 67–68 °C. Records in `tools/recordings/p1-20261027/` (`p1-20261027-220037.json`, `test-hw.txt`: the run’s output; `p1-diag2.json`: the two-tap emulation); figures by `tools/p1_figures.py`. `common/tests/verify_build.py --design timetagger`: PASS, all 8 stages (the hand placement in `constraints/strips.tcl` included).

- **Build:** WNS +0.016 ns, WHS +0.007 ns; 1856 slices (42 %), 3437 LUTs, 7254 flip-flops, 24.5 BRAM tiles, both MMCMs. Every CARRY4 and first-rank flip-flop sits where `constraints/strips.tcl` put it (A X43 rows 4–99, B X41 36–99, C X39 0–79). DRC: only the allowed combinational loop and the gated-clock note of the pin-clocked toggles.
- **Ring oscillator:** 118.9 MHz (13 LUTs).
- **Taps** (8192 raw snapshots per strip): 60.7 ps per CARRY4 (A), 64.2 ps (B), 59.6 ps (C): 65.9, 62.4 and 67.1 cells per 4 ns period. Inside a cell CO1 comes 24–52 ps before the other three outputs, which lie within about 10 ps of each other.
- **Chain length:** strip B (64 cells) finds the chain full on 4.5 % of the hits (ovf): too short at this temperature. A (96) and C (80) never overflow. → The instrument uses 80 cells.
- **Code density** (45 million ring-oscillator hits per strip, all four taps): mean bins 14.4–16.0 ps; largest 42–74 ps; no empty bin; DNL −1.00 … +3.6; rms quantization 6.4–9.5 ps. Transitions to 1 run more slowly than transitions to 0 (A: 259 against 274 codes per period).
- **Two taps per cell** (emulated from the raw snapshots, CO1+CO3 or CO1+CO2): 125–138 taps per period, mean bins 29–32 ps, rms quantization 9.7–11.3 ps (against 6.4–9.5 ps with four); the spread at a fixed phase grows from 18.3 to 19.6 ps (A) and 17.0 to 17.3 ps (C), the test signal’s own jitter included. → CO1+CO3.
- **Calibration:** with each strip’s code-density tables, the difference of the times of strips A and C for the phase-stepped test signal, over 112 phase steps covering the period, has a mean of 33.6 ps (transitions to 0) and 24.5 ps (to 1) that varies by only **3.5 and 4.1 ps rms** (36 and 34 ps peak to peak). Against the MMCM’s phase steps themselves each strip shows about 20 ps rms with the same shape on all strips and both polarities: that is the MMCM phase interpolator’s error, not the delay lines’. The phase steps are no linearity reference; two delay lines against each other are.
- **Precision:** A − C at one phase **10.6 and 12.7 ps rms** (two strips, four taps): about 8–9 ps per strip. With two taps the snapshot emulation gives about 9–11 ps per strip (estimate: the test signal’s jitter, about 15 ps, removed in quadrature).
- **SATA:** DIFF_SSTL18_I with the split termination works with the board’s own S1 output through the cable: A (internal) − C **12.1 and 14.2 ps rms**; delay 7.17 ns (to 0) and 6.87 ns (to 1), the cable included.
- **External edges** (awg on DIO0_P, A and B on the pin): 27.6 ps rms (rising), 85 ps (falling); strip B’s calibration is wrong where it overflows, so this says nothing yet about the pin path (1.0 repeats it with two full-length strips).
- **Polarity:** the two polarities have their own tables and also their own offsets (A − B differs by about 250 ps between rising and falling transitions; A − C by 9 ps; the SATA path by 0.3 ns): the instrument calibrates both per channel.

2026-10-27: 0.1 prototype, offline

- `make sim`: `tb_core` PASS (”delay-line prototype”) with behavioural delay lines of 20 ps taps: codes floor(D / 20 ps) for edges D before a clock edge (16 phases, both polarities, D < 80 ps one sample later), strip B as A’s partner identical, strip C on the SATA pair, a 2 ns pulse lost in level mode and timed in toggle mode, an edge 3 ns after another flagged dbl and kept out of the histogram, histograms, event capture, a raw snapshot (61 and 261 taps set), the phase stepper both ways.
- `make test`: 12 tests (configuration packing and derivation, invalid settings, presets, decoders, the analysis on synthetic lines: bins, tap delays, subsets, linearity, pairs; protocol, client, loading).