boxcar 1.0: reference documentation

overview, recipes, theory, control reference, registers, test records

Rok Zitko, Jozef Stefan Institute

2026-11-13 (commit 6af1a9b)

Contents

boxcar: a pump–probe boxcar averager and segmented scope **2**
Connecting to lab equipment: read first . . . 2
Quick start . . . 3
Presets . . . 3
Specifications (design values; measured ones marked) . . . 3
Hardware setup . . . 4
Files . . . 5

boxcar recipes **5**
Without a laser: the loopback (two cables) . . . 5
Placing the gates on a real pulse . . . 6
An amplified laser (1–10 kHz) with a chopper . . . 6
A high-repetition laser (50 kHz–2 MHz) with a modulator . . . 7
The board as the master: two lasers and an electronic delay scan . . . 7
A PC-driven delay stage . . . 7
Two independent signals . . . 8
The boxcar outputs into a recorder or a servo . . . 8
A laser emulated by awg on a second board . . . 8
Every shot on a PC . . . 9

boxcar: theory and definitions **9**
1. Pump–probe with a boxcar . . . 9
2. Definitions . . . 12

boxcar commands (TCP) **16**
Configuration (`APPLY {json}`, `CONFIG?`) . . . 17

boxcar registers **18**
Pins, the DDR writers (0.0) . . . 18
The run, the triggers, the engine (0x100) . . . 19
Pulse outputs (0x200 + 16 k, k = 0 ... 7 for P1 ... P8) . . . 20
The input path (0x300) and the test source (0x340) . . . 20
Averaged traces (0x380) and segments (0x3C0), 0.2 . . . 20
Gates (0x400 + 16 k, k = 0 ... 7 for G1 ... G8; 0.3) . . . 20
The boxcar outputs and the limits (0x4B0; 0.4) . . . 21
Block RAMs . . . 21

Verification record **21**
2026-11-10: 1.0, the oscilloscope bench . . . 21
2026-11-09: 1.0, awg on a second board as the laser . . . 22
2026-11-09: 1.0, the release (plan step 1.0) . . . 22
2026-11-09: 0.4, the boxcar outputs (plan step 0.4) . . . 24
2026-11-09: 0.3, gates and sums (plan step 0.3) . . . 25
2026-11-08: 0.2, traces and segments (plan step 0.2) . . . 25
2026-11-08: 0.1, shots and pins (plan step 0.1) . . . 26
2026-11-08: 0.0b, the averager and the pump's tap in common (plan step 0.0b) . . . 27
2026-11-08: 0.0, the skeleton (plan step 0.0) . . . 27

boxcar: a pump–probe boxcar averager and segmented scope

boxcar turns the Red Pitaya into a **boxcar averager for pump–probe experiments** with a **segmented oscilloscope**. On every laser shot it integrates eight gates on the detector signals (IN1, IN2), sorts the shots into categories (pump on and off, from a chopper's reference or from a modulation pattern it drives itself), and sums them per point of a delay scan: the pump–probe signal ΔT/T, normalized shot by shot by a reference detector. Meanwhile it averages whole traces of both inputs per category, so the gates can be placed on the averaged detector pulse, and it can store every raw segment in DDR. It follows a laser's trigger (timed to 1 ns) or is the master that fires the pump, probe and modulator pulses on eight 2 ns outputs, stepping the electronic delay from point to point.

It is meant for:

- amplified lasers at 1–10 kHz (a chopped pump, shot-to-shot ΔT/T, a mechanical delay stage stepped by the PC);
- lasers at 50 kHz–2 MHz (fibre and cavity-dumped lasers, pulse pickers);
- two lasers or a laser and a modulator fired with an electronic delay from the board.

Status: version 1.0. It takes laser shots from a TTL trigger (timed to 1 ns, with a glitch filter and hold-off), an analog level on IN1 or IN2, its own timer or software; sorts them into categories (a chopper's reference on CHOP with a guard band, or a pattern it drives itself, also onto a modulator); counts points and sweeps of a delay scan; and fires up to eight timed pulses per shot on P1–P8 (2 ns steps, per category, a delay table per point: the electronic delay scan), with markers and static levels on the other pins. It averages whole traces of IN1 and IN2 per category without dead time (up to 4096 points, 1–64 samples per point, a pre-trigger), shows single snapshots and writes every shot's raw segment into DDR. Eight gates per shot give D1 and D2 (each a gate minus a baseline gate) and their ratio Q = D1/D2 (signal over reference); pump-on/off pairs give d = Q_on − Q_off; everything is summed per delay point and category (with squares, for error bars) into POINT records, which the service totals over sweeps (`POINTS?`: means, errors and ΔT/T per point). A hold makes the board take N shots per point and wait for a PC-driven delay stage. Records (RUN, SWEEP, SHOT with the gate values, POINT, BLOCK, SEGMENT) stream to a PC. OUT1 and OUT2 show a running average of any value per category (pump on, pump off, or their difference) in volts within lockable limits, or carry a test source that emulates a laser and a sample for a cable loopback. Presets for the common setups, a Jupyter panel, a live window for a PC (`tools/live.py`), worked recipes, a user's manual with the board's recordings and a theory manual with exercises through a cable loopback come with it; docs/THEORY.md explains the physics (section 1) and defines every clock (section 2), docs/VERIFICATION.md lists what was tested (1.0 passed 40 board checks twice on 2026-11-09, with OUT1 → IN1, OUT2 → IN2 and jumpers P8 → TRIG, P7 → CHOP: among them the averaged traces equal to the sum of the raw segments from 1 kHz to 2 MHz, the gates recomputed from raw segments equal to the SHOT records and the SHOT records' sums equal to the POINT records, all exactly; a boxcar output read back within 10 mV; every preset read back; the emulated sample's ΔT/T(τ) measured within 0.0006; and with awg on a second board as an asynchronous laser with a chopper: every shot taken from 1 kHz to 1.95 MHz, ΔT/T within $4 \cdot 10^{-4}$ of awg's exact value; the pins' and the outputs' latencies with an oscilloscope). Behaviour with real lasers and detectors is unmeasured: no laser has been at hand (a second board running awg played one).

Connecting to lab equipment: read first

Read the equipment-safety section of the root README. For boxcar in particular:

- **Extension connector E1 is 3.3 V LVCMOS and not 5 V tolerant.**
  - boxcar reads DIO0_P (TRIG), DIO1_P (CHOP), DIO2_P (ARM) and DIO3_P (AUX), all with pull-downs.
  - A 5 V TTL source (a laser's or chopper's sync output) needs a 50 Ω terminator at the board, which brings it to about 2.5 V. Never connect 5 V directly.
- **Outputs DIO0_N–DIO7_N** (P1–P8) and **DIO4_P–DIO7_P** (M1–M4) are high impedance after loading and after SAFE. They are driven only when enabled. Never drive an output pin from outside.
  - A released pin has only the FPGA's weak pull-down; before the FPGA is configured the pins are not defined at all.
  - **Fit an external pull-down (e.g. 10 kΩ) on any line that fires a laser, a Pockels cell or a shutter**, so nothing fires while the board boots or reloads.
- **OUT1/OUT2** are at their DC levels (0 V) after loading and SAFE. They carry the test source's detector-like pulses or the boxcar outputs only when set; give a boxcar output limits (and lock them) before it drives a servo or a laser's modulation input.
- **IN1/IN2** take ±1 V on the LV jumpers (±20 V on HV). Photodiode amplifiers and PMTs can exceed that: check the detector's output range first.

Quick start

With boxcar loaded on a board (no Vivado needed) and two cables, OUT1 → IN1 and OUT2 → IN2: the test source plays a laser's signal and reference pulses and a sample.

```
import sys, time; sys.path[:0] = ["common/python", "boxcar/control"]
from boxcar import BoxcarClient
with BoxcarClient("192.168.0.102") as dev:
    dev.load_preset("loopback-demo")          # 50 kHz, dT/T 10 % on category 1, four gates, traces
    dev.points_clear(); dev.run(); time.sleep(2)
    print(dev.points()[0]["dtt"])             # dT/T from the sums: 0.0993 (the emulated sample
     ↪  0.0998)
    blk = dev.trace_block()                   # averaged traces per category, in volts
    dev.arm_snapshot(); snap = dev.read_snapshot()        # one shot
    seg = dev.segments(100)                   # 100 raw shots
    dev.stop()
    dev.load_preset("loopback-delay-scan")    # a pump-probe curve: 41 delays, the emulated decay
    dev.points_clear(); dev.run(); time.sleep(5); dev.stop()
    for p, r in dev.points().items():
        print(p, r["dtt"]["ratio"], r["dtt"]["ratio_sem"])
```

In the board's Jupyter, `control/boxcar.ipynb` opens the panel (`import_plugin("boxcar").make_gui(dev)`: run control, one shot's timing, the traces with the gates, the pump–probe curve, the gates and the boxcar outputs). On a PC, `python3 boxcar/tools/live.py --host 192.168.0.102` draws the traces and the curve live (needs matplotlib). RECIPES.md has the experiments with their wiring.

Presets

Preset	Setup
`off`	all outputs in high impedance, the test source off
`loopback-demo`	no laser (cables OUT1 → IN1, OUT2 → IN2): the timer at 50 kHz fires P1, which fires the test source (ΔT/T 10 % on category 1, 20 % shared noise); four gates, Q, traces, a POINT record every 1000 shots
`loopback-delay-scan`	no laser: the probe P1 stepped 0.2 … 8.2 µs in 41 points, the gates following, the pump P2 at 1.2 µs; the emulated ΔT/T = 5 % exp(−τ / 2 µs)
`amplified-chopper`	1–10 kHz: TRIG the laser's sync, CHOP the chopper (1 µs guard); signal and baseline on IN1, reference and baseline on IN2, Q; every SHOT record, POINT every 1000 shots, traces; OUT1 Q's difference, OUT2 D1 (within ±0.5 V)
`high-rep-modulator`	50 kHz–2 MHz: TRIG the sync, categories 1, 0, 1, 0, … with P1 driving the modulator (the next shot's category), 48 ns gates, POINT every 65,536 shots
`master-delay-scan`	the board as the master at 10 kHz: P1 the pump (category 1) at 2 µs, P2 the probe at 0 … 20 µs in 101 points, the gates following the probe
`stage-scan`	a PC-driven stage: TRIG and CHOP as `amplified-chopper`, 2000 shots per point, then hold until NEXT (ARM or the host), 100 points
`two-signals`	D1 on IN1 and D2 on IN2 as two independent signals, their pump-on − pump-off differences on OUT1 and OUT2
`scope`	TRIG; an averaged trace of IN1 and IN2 of 4096 samples from 2 µs before the trigger, snapshots and segments of 1024 samples

Specifications (design values; measured ones marked)

Quantity	Value
Triggers	TTL on DIO0_P (1 ns timestamps, rising or falling, a glitch filter 1 … 8 ns, a hold-off), a Schmitt threshold in volts on IN1 or IN2 (8 ns), the internal timer (8 ns … 34 s), software; a trigger while a shot is busy is counted as missed
Shots	categories from CHOP (with a guard band: shots near a chopper edge are *invalid*) or a pattern of up to 16 bits; up to 1024 points per sweep (per shot, every N shots, on ARM's edges, or set by the host), run limits (shots, sweeps, a duration)

Quantity	Value
Pulse outputs	P1–P8 (DIO0..7_N): per shot a pulse at a delay (1 ns resolution, 2 ns slots) and width, per category; a delay table per point (the electronic delay scan); 16 pending pulses each (a pulse may come after the next trigger). Measured with an oscilloscope: the edge at the pin delay + 21.7 ns after the TTL edge at TRIG (±0.6 ns rms, every 2 ns step exact), the shortest on-time delay 88 ns (110 ns after the edge; a shorter one is counted late and starts at about 106 ns); level outputs (markers, the next category) 74 ns after the edge (±2.2 ns rms)
Other outputs	M1–M4 (DIO4..7_P) and any P: static levels or patterns, markers of shots, points, sweeps, triggers, the category, the next pattern bit (a modulator's drive), run, the test source's sync and chopper; inversion
Snapshot	up to 4096 samples of IN1 and IN2 at 8 ns around a shot, up to 65 µs before it (an 8,192-sample delay line shared with the traces and segments)
Averaged traces	IN1 and IN2 per category (2 × 2048 points) or one trace (4096 points), 1–64 samples per point, from up to 65 µs before the shot; 36-bit sums in two halves (one averages while the host reads the other: no dead time), swaps after N traces (optionally only at a point's first shot) or on request; a trace per shot while it ends 5 samples before the next one starts
Segments	every accepted shot (or the first M of each point): a 4-word header (time, shot, point, sweep, category) and up to 8192 samples of IN1 and IN2, written into DDR through S_AXI_HP1 at up to 500 MB/s (e.g. 28 samples at 2 MHz, 4092 at 30 kHz)
Test source	OUT1/OUT2: a detector-like pulse (a shape at 1 ns, e.g. rise and decay times) per emulated laser shot (its own timer or fired by a pulse output), amplitude per category with ΔT/T per point, a common noise, a sync and a chopper level on pins
Gates	8 per shot, each on IN1 or IN2, a delay (8 ns steps, from 65 µs before the shot) and a width of 1 … 65,536 samples (8 ns … 524 µs), optionally moved per point (a gate table); exact 32-bit integrals and an overload flag; a shot's gates are taken if they start after the previous shot's have ended (at least 8 clocks) and 48 clocks after the previous shot's gates ended (up to 2.6 MHz)
Derived values	D1, D2 = a gate minus (its width / a baseline gate's width) × the baseline gate; Q = D1/D2 (24 fractional bits); d = Q_on − Q_off for consecutive pump-on/off shots
Point sums	per point (and every 65,536 shots) and category: the shots, Σx and Σx^2 of G1 … G8, D1, D2, Q, ΣD1·D2, the pairs' Σd, Σd^2, exact (48/80-bit); totalled over sweeps by the service (`POINTS?`)
Records	RUN, SWEEP, SHOT (every M-th shot; with the gates, D1, D2, Q chosen), POINT (106 words), BLOCK (a frozen half), SEGMENT into a ring in DDR through S_AXI_HP0, streamed to a PC (`RECORDS STREAM`, the ring pump)
Boxcar outputs	OUT1, OUT2: an exponential average over 1 … 2^{20} shots of a gate, D1, D2 or Q, per category (pump on, off, or their difference), scaled to volts, updated (measured with an oscilloscope) 0.90 µs (OUT1) and 1.08 µs (OUT2) after the end of the shot's last gate at the IN connector, with a delay line of 7 clocks (each clock more, from a longer pre-trigger anywhere, adds 8 ns); limits in volts on every output word, lockable until the FPGA is reloaded
Measured through a cable loopback	ΔT/T of the test source's emulated sample within 0.5 % of its exact value (`emulated_dtt`; at 0.3 V, a differential gain of the converters; 0.02 % at 0.15 V and 0.6 V); a 5 % decay over 41 delays within 0.00044; Q = D1/D2 removes 20 % shared noise to 0.16 %; every shot's gates taken at 2.016 MHz; averaged traces equal to the sum of the raw segments and the gates equal to the segments' sums, exactly (docs/VERIFICATION.md)
Measured with awg as the laser	every shot taken from 1 kHz to 1.95 MHz (4 million shots), the categories exact; ΔT/T within $4 \cdot 10^{-4}$ of awg's exact value for pulses up to 0.68 V at IN1; a gate holding the whole pulse independent of the laser's phase to ±0.05 %; a chopper edge 60 ns before the gate adds $4 \cdot 10^{-4}$ (keep edges half a period away); awg's pulse 392 ns after P1 in master mode

Hardware setup

Pin (E1)	Use
DIO0_P	TRIG (the laser's trigger; 1 ns timestamps)
DIO1_P	CHOP (a chopper's reference: pump on or off)
DIO2_P	ARM (a delay stage's "in position", or the next point)
DIO3_P	AUX (spare input)
DIO0_N ... DIO7_N	pulse outputs P1 ... P8 (pump and probe triggers, pulse picker, chopper or AOM drive, shutter)
DIO4_P ... DIO7_P	markers M1 ... M4 (shot, point, sweep, category, ...)

Files

Path	Contents
`src/bx_core.v`	the core: registers, pins, the output modes, the DDR writers
`src/bx_trig.v`, `src/bx_engine.v`, `src/bx_pulse.v`	the triggers, the shot engine, a pulse output
`src/bx_in.v`, `src/bx_test.v`, `src/bx_rec.v`	the input path (the delay line's stream) and snapshot, the test source, the records
`src/bx_q.v`, `src/bx_trace.v`, `src/bx_seg.v`	the consumers' start queues, the averaged traces (into the common `rp_avg`), the segments (into the common `rp_seg_writer` on S_AXI_HP1)
`src/bx_gate.v`	the gates, D1, D2, Q, pairs, the point sums, the SHOT and POINT records' FIFOs
`src/bx_out.v`	the boxcar outputs: averages per category, codes
`control/boxcar/model.py`	the bit-exact model (the RTL's reference)
`control/boxcar/`	driver (with the presets), client, protocol, the service's record worker and stream, the Jupyter panel (`gui.py`)
`control/boxcar.ipynb`	the notebook
`tools/live.py`	a live window on a PC (host-only, matplotlib)
`RECIPES.md`	experiments with their wiring and code
`docs/manual/`	the user's manual and the theory manual (LaTeX, PDFs), their board recordings (`data/`)
`tools/record_manual_data.py`, `tools/manual_figures.py`, `tools/theory_figures.py`	the manuals' recordings (through the client) and figures (host-only)
`docs/`	THEORY (physics and definitions), REGISTERS, PROTOCOL, VERIFICATION
`scripts/test_hw.py`	the board test (`make test-hw`)
`tests/`	the driver, protocol, client, GUI, tools and manual tests, the XSim testbenches

boxcar recipes

Worked examples in Python on a PC (`dev = BoxcarClient("192.168.0.102")`) or in the board's Jupyter (`dev = import_plugin("boxcar").CLIENT("127.0.0.1")`): the methods are the same. Times are in seconds at the connectors, volts at the connectors (through the board calibration). The physics behind each recipe is in docs/THEORY.md section 1, the commands in docs/PROTOCOL.md, the presets in README.md.

```
import sys, time; sys.path[:0] = ["common/python", "boxcar/control"]
from boxcar import BoxcarClient, model
dev = BoxcarClient("192.168.0.102")
```

Read the root README's equipment-safety section first. The DIO pins are 3.3 V and **not 5 V tolerant**: a laser's or a chopper's 5 V TTL output that drives 50 Ω needs a 50 Ω terminator at the board (about 2.5 V then); one that does not drive 50 Ω needs a divider or a level shifter. The inputs IN1/IN2 take ±1 V on the LV jumpers (±1.13 V on the test board): an attenuator in front of a detector with more output. Join the grounds of every instrument. The pulse outputs P1–P8 and the markers M1–M4 stay in high impedance until a preset or a setting turns them on.

Without a laser: the loopback (two cables)

Two cables, OUT1 → IN1 and OUT2 → IN2. The test source plays a laser's signal pulse (OUT1) and its reference (OUT2) with 20 % shared shot-to-shot noise, and a sample whose ΔT/T it is given.

```
dev.load_preset("loopback-demo")              # 50 kHz, dT/T 10 % on category 1, four gates, traces
dev.points_clear(); dev.run(); time.sleep(2); dev.stop()
r = dev.points()[0]
print(r["Q"]["mean"], r["dtt"])              # dT/T: 0.0993 measured, 0.0998 emulated (0.1
 ↪  programmed)

dev.load_preset("loopback-delay-scan")        # a probe delay scan, 41 points, the emulated decay
dev.apply({"shots": {"sweeps": 20}})
dev.points_clear(); dev.run()
while dev.state()["running"]:
    time.sleep(0.1)
pts, cfg = dev.points(), dev.config()
for p, r in pts.items():
    print("%.1f us  dT/T %.5f +- %.5f  (emulated %.5f)" % (cfg["points"]["delays"][p] * 1e6,
     ↪  r["dtt"]["ratio"],
                                                          r["dtt"]["ratio_sem"], cfg["test"]["dtt"][p]))
```

The test source's sample has a ΔT/T a little below its table (its amplitudes are integer codes and the product is truncated): `driver.emulated_dtt(dev.config(), dev.calibration())` gives the exact values, and the board measured the decay within 0.00044 of them at all 41 points (docs/VERIFICATION.md, 1.0). The shot that ends a point plays the next point's ΔT/T (THEORY 2.9): a step of the table shows as a $1/n_1$ part of it one point early. On a PC, `python3 boxcar/tools/live.py --host 192.168.0.102` draws the traces and this curve live.

Placing the gates on a real pulse

Before gating, look at the detector's pulse: the `scope` preset averages 4096 samples of IN1 and IN2 from 2 µs before the trigger.

```
dev.load_preset("scope")                      # TRIG (DIO0_P) the laser's sync
dev.run()
blk = dev.trace_block()                       # the next frozen block (1000 scans), in volts
dev.arm_snapshot(); snap = dev.read_snapshot()  # one shot
dev.stop()
t = blk["t"]                                  # s from the trigger, each point's centre
peak = max(range(len(t)), key=lambda i: blk["in1"][i])
print("IN1 peaks at %.0f ns, %.3f V" % (t[peak] * 1e9, blk["in1"][peak]))
```

Put the signal gate over the whole pulse (its edges on the flat baseline, not on the slopes: THEORY 1.2) and the baseline gate before it, both on the same input; the same on the reference. `timing_svg` in the panel, or the live window, draws them over the traces.

```
dev.apply({"gates": {"G1": {"enabled": True, "input": "in1", "delay": 120e-9, "width": 320e-9},
                     "G2": {"enabled": True, "input": "in1", "delay": -400e-9, "width": 320e-9},
                     "G3": {"enabled": True, "input": "in2", "delay": 80e-9, "width": 320e-9},
                     "G4": {"enabled": True, "input": "in2", "delay": -400e-9, "width": 320e-9}},
           "derived": {"d1": {"signal": "G1", "baseline": "G2"}, "d2": {"signal": "G3", "baseline":
            ↪  "G4"}}})
print(dev.status()["warnings"])               # e.g. shots closer than the gates' span
```

An amplified laser (1–10 kHz) with a chopper

from	to	
the laser's sync (TTL)	TRIG, DIO0_P	50 Ω terminator if it drives 5 V into 50 Ω
the chopper's reference	CHOP, DIO1_P	high = pump passes (else `categories.invert`)
the signal detector (after the sample)	IN1	LV jumper, within ±1 V
the reference detector (before the sample)	IN2	LV jumper
OUT1, OUT2	a chart recorder, a scope	optional

```
dev.load_preset("amplified-chopper")          # 4 gates, Q = D1/D2, POINT every 1000 shots, traces
dev.apply({"gates": {"G1": {"delay": 120e-9}, "G3": {"delay": 80e-9}}})    # your pulses' positions
dev.points_clear(); dev.run(); time.sleep(10)
r = dev.points()[0]                           # the totals of all POINT records so far
```

```
print("shots", r["n"], "invalid", r["n_invalid"])
print("dT/T %.2e +- %.1e (ratio), %.2e +- %.1e (pairs)" % (r["dtt"]["ratio"], r["dtt"]["ratio_sem"],
                                                           r["dtt"]["pairs"], r["dtt"]["pairs_sem"]))
```

The chopper's edges invalidate the shots within 1 µs of them (`categories.guard`); a chopper at half the laser's rate alternates the categories shot by shot. Lock the chopper to the laser (its controller's sync input), so that no shot falls on an edge; a free-running chopper works too, with a few invalid shots. OUT1 shows Q's difference (on − off, 10 V per unit of Q, 256 shots), OUT2 the signal D1 of the pump-on shots (1 V/V), both within ±0.5 V.

A high-repetition laser (50 kHz–2 MHz) with a modulator

from	to	
the laser's sync	TRIG, DIO0_P	
P1, DIO0_N (3.3 V)	the modulator's gate (AOM driver, pulse picker)	high: pass the pump
the signal detector	IN1	
the reference detector	IN2	

```
dev.load_preset("high-rep-modulator")         # categories 1, 0, 1, 0, ...; P1 the next shot's
 ↪ category
dev.points_clear(); dev.run(); time.sleep(5)
r = dev.points()[0]
print(r["n"], r["dtt"]["ratio"], r["dtt"]["ratio_sem"])
```

P1 (mode *next*) changes after each trigger to the next shot's category, so the modulator has the laser's period minus the trigger-to-pin latency (74 ns from the TTL edge at TRIG to the level at the pin, ±2.2 ns rms, measured with an oscilloscope) to open or close for the next pump pulse: check on a scope that the modulator settles in time. The gates are 48 ns wide with their baselines 64 ns before; at these rates the POINT records (every 65,536 shots) carry the result, not SHOT records. `shots.busy` (480 ns) keeps the traces and gates consistent up to 2.08 MHz; above that, shorten the traces or reduce `busy`.

The board as the master: two lasers and an electronic delay scan

from	to	
P1, DIO0_N	the pump laser's trigger input	fired on category-1 shots
P2, DIO1_N	the probe laser's trigger input	its delay stepped by the table
the signal and reference detectors	IN1, IN2	

```
dev.load_preset("master-delay-scan")          # 10 kHz, the probe 0..20 us in 101 points, gates
 ↪ following
grid = [0.0] + [10 ** (-8 + 3 * i / 99) for i in range(100)]          # 10 ns .. 10 us, logarithmic
grid = [round(d / 8e-9) * 8e-9 for d in grid]                         # whole samples (gates follow)
dev.apply({"points": {"count": len(grid), "delays": grid, "gate_delays": grid},
           "outputs": {"P1": {"delay": 1e-6}}})                        # the pump 1 us after the
            ↪ trigger
dev.points_clear(); dev.apply({"shots": {"sweeps": 10}}); dev.run()
while dev.state()["running"]:
    time.sleep(0.2)
pts = dev.points()
curve = [(grid[p] - 1e-6, r["dtt"]["ratio"]) for p, r in sorted(pts.items())]   # (pump-probe delay,
 ↪ dT/T)
```

The probe's delay is d_0 + T[p] with T the table (ns resolution, 2 ns pins); the gates' table must hold whole samples (multiples of 8 ns): round the grid as above. The gates' span (their latest end plus the table's largest entry) must stay below the shots' spacing (100 µs here; the driver warns). A shuffled grid (`random.shuffle` the table, keep both tables equal) decorrelates a drift from τ. Delays may exceed the repetition period: each output keeps 16 pending pulses.

A PC-driven delay stage

ARM (DIO2_P) from the stage controller's "in position" output, or the host's `next_point()`:

```
def move_stage(position_mm):
    pass                                        # your stage's driver

def wait_held():
    while not dev.state()["held"] and dev.state()["running"]:
        time.sleep(0.01)

dev.load_preset("stage-scan")               # 2000 shots per point, then hold until NEXT; 100 points
positions = [0.005 * p for p in range(100)]
move_stage(positions[0])
dev.points_clear(); dev.run()
for p in range(1, len(positions)):
    wait_held()                                 # point p - 1 is done
    move_stage(positions[p])
    dev.next_point()                            # point p starts
wait_held(); dev.stop()
pts = dev.points()
```

Each point's 2000 shots come after the stage arrived; the shots while the board is held count as triggers only. A stage that signals "in position" on ARM releases the hold itself (then only the moves are the PC's).

Two independent signals

```
dev.load_preset("two-signals")              # D1 = G1 - G2 on IN1, D2 = G3 - G4 on IN2, each on its
 ↪  own
dev.points_clear(); dev.run(); time.sleep(5)
r = dev.points()[0]
for name in ("D1", "D2"):
    on, off = r[name]["mean"][1], r[name]["mean"][0]
    print(name, "pump on - off: %.3e V" % (on - off), "relative %.3e" % (on / off - 1))
```

OUT1 and OUT2 show the two differences (10 V/V, 256 shots). Q = D1/D2 is formed anyway; ignore it.

The boxcar outputs into a recorder or a servo

```
dev.apply({"test": {"out1": {"mode": "off"}},
           "analog": {"out1": {"mode": "boxcar", "source": "Q", "show": "difference", "average":
            ↪  1024,
                               "scale": 20.0}},
           "limits": {"out1": {"low": -0.2, "high": 0.2}}})
dev.lock_limits()                               # until the FPGA is loaded again
```

The average runs over about `average` shots of each category (time constant average / f_category); OUT1's level changes 0.90 µs and OUT2's 1.08 µs after the end of the shot's last gate at the IN connector (measured with an oscilloscope, a delay line of 7 clocks; THEORY 2.9). A locked limit holds against later settings: protect a servo's input or a laser driver's modulation input this way.

A laser emulated by awg on a second board

Tested on 2026-11-09 (docs/VERIFICATION.md; receipt `docs/data/board-1.0-awg-session-2026-11-09.json`). A second Red Pitaya running awg plays an independent laser with a chopper. Wiring (A: awg, B: boxcar): A D0 (DIO0_N, pin 4) → B TRIG (DIO0_P, pin 3); A D1 (DIO1_N, pin 6) → B CHOP (DIO1_P, pin 5); A OUT1 → B IN1, A OUT2 → B IN2 (coax); pin 26 → pin 26 (ground). awg loops two shots, pump on and pump off, each with a pulse on OUT1 (1 + dT/T on the pump-on shot) and OUT2, and with each shot a digital pattern: the sync on D0 and the chopper on D1, its edges half a period from the shots.

```
import math
from rpfpga.client import connect
awg = connect("192.168.0.104")
n, dtt = 1250, 0.02                                 # samples per shot (100 kHz), dT/T
pulse = [(1 - math.exp(-k * 0.8)) * math.exp(-k * 0.2) for k in range(48)]    # rise 10 ns, decay 40
 ↪  ns
peak = max(pulse)

def shot(height):
```

```
    v = [0.0] * n
    for k, p in enumerate(pulse):
        v[4 + k] = height * p / peak
    return v

half = n // 2
on = [0x03] * 8 + [0x02] * (half - 8) + [0x00] * (n - half)     # D0 sync, D1 the chopper
off = [0x01] * 8 + [0x00] * (half - 8) + [0x02] * (n - half)
awg.save_segment("lon", shot(0.8 * (1 + dtt)), values2=shot(0.8))
awg.save_segment("loff", shot(0.8), values2=shot(0.8))
awg.save_pattern("pon", on)
awg.save_pattern("poff", off)
awg.save_sequence("laser", [{"segment": "lon", "digital": "don"},
                            {"segment": "loff", "digital": "doff", "next": 0}])
prog = {"mode": "memory", "rate": 125e6, "repeats": 1, "end": "hold"}
awg.apply({"arb": {"sequence": "laser", "rate": 125e6, "run": "continuous"},
           "outputs": {"1": {"arb": {"on": True, "amplitude": 0.6, "filter": "smooth"}},
                       "2": {"arb": {"on": True, "amplitude": 0.6, "filter": "smooth"}}},
           "digital": {"run": "sequencer", "idle": 0, "enable": 0x03,
                       "programs": {"don": dict(prog, pattern="pon"), "doff": dict(prog,
                        ↪  pattern="poff")}}})
dev.load_preset("amplified-chopper")                 # TTL, CHOP, four gates; place them on the
 ↪  traces
```

Measured that way: every shot taken from 1 kHz to 1.95 MHz (4 million shots), the categories exact, awg's crystal 4.3 ppm off boxcar's; dT/T within $4{\cdot}10^{-4}$ of awg's exact value (its quantized points) for pulses up to 0.68 V at IN1. With boxcar's P1 (DIO0_N) wired to awg's TRIG A (DIO0_P) and awg playing a triggered sequence (`"run": "triggered", "trigger": {"source": "trig_a"}`), boxcar is the master: awg's pulse comes 392 ns after P1, and a delay scan with the gates following keeps D1 within ±0.05 %.

Every shot on a PC

SHOT records (with the gates' values) stream to the PC through the service:

```
dev.load_preset("amplified-chopper")          # every SHOT record with every value
dev.apply({"shots": {"limit": 10000}})
mask = sum(1 << model.SUM_NAMES.index(v) for v in dev.config()["records"]["values"])
with dev.records() as stream:
    dev.run()
    words = stream.collect(timeout=1.0)       # until no record came for 1 s (the run's end)
shots = [model.shot_fields(info, ws, mask)
         for typ, info, ws in model.parse_records(words) if typ == model.REC_SHOT]
q1 = [s["values"]["Q"] for s in shots if s["category"] == 1]
```

Each SHOT record carries the trigger's time (1 ns), the point, the category, the sweep, the shot number and the chosen values (G1–G8, D1, D2, Q as raw sums: THEORY 2.8); `point_results` converts the POINT totals to volts. At high rates use `records.shot_every` to send every M-th shot.

boxcar: theory and definitions

This document has two parts. Section 1 explains, for lab users, what a boxcar averager does in a pump–probe experiment and what limits it. Section 2 is the **contract**: the definitions the RTL, the bit-exact model (`control/boxcar/model.py`, from 0.1) and the driver follow. A step of the plan fixes its subsections before its RTL is written; subsections marked *planned* are the approved plan's design, not yet built.

1. Pump–probe with a boxcar

This section explains the measurement boxcar makes and what limits it, with the numbers the board measured through a cable loopback (docs/VERIFICATION.md; no laser has been at hand, so every statement about lasers and detectors here is the physics, not a measurement). Section 2 defines every register and clock; worked setups are in RECIPES.md, the presets in README.md.

1.1 Pump–probe

A **pump** pulse excites the sample; a weaker **probe** pulse, delayed by τ, measures a property that the excitation changed: the transmission T, the reflectivity R, a fluorescence or a photocurrent. Scanning τ maps the sample's response in time: a carrier population that decays, a coherent oscillation, a heat pulse that diffuses. The quantity is the relative change

```
ΔT/T (τ) = (T_pump on − T_pump off) / T_pump off,
```

often 10^{-2} … 10^{-6}, which is why the measurement compares shots with the pump on and off, divides by a reference, and averages many shots. Femtosecond to nanosecond delays come from an optical delay line (a translation stage; the board waits for it, 1.6); nanoseconds to milliseconds from two lasers fired electronically, with the board's pulse outputs (2 ns steps, 1.6).

1.2 The boxcar: gated integration

A detector turns each probe pulse into a voltage pulse x(t), from a few ns (a fast photodiode) to microseconds (an integrating preamplifier). A **gate** sums the samples of x in a window of W samples (8 ns each) at a fixed delay from the shot's trigger: g = Σ x, a number per shot proportional to the pulse's area when the window holds the whole pulse. A **baseline gate** before the pulse measures the offset and the slow drifts of the same input, and D = g_a − (W_a / W_b) g_b removes them (2.7). Everything between the pulses is ignored: at 1 kHz with a 400 ns pulse the gate is open 0.04 % of the time, so the noise of the other 99.96 % never enters, which is the boxcar's advantage over a lock-in at the same repetition rate.

The gate's width: averaged over a gate of duration T_g = 8 ns · W, white noise of density e_n (V/√Hz) leaves about e_n / √(2 T_g) on the gate's mean level, while a pulse of area A (V·s) gives the mean level A / T_g once the window holds it: the signal-to-noise ratio A √2 / (e_n √T_g) is largest for the shortest window that still holds the whole pulse (wider only adds noise, narrower loses signal). The inputs' noise floor on this board is about 1 µV/√Hz at kHz and 0.09 µV/√Hz above 1 MHz (the root README, measured with lockin); a detector's or a laser's noise is normally far larger.

Sampling: the samples are 8 ns apart and the analog front end passes about 60 MHz. With an asynchronous laser the pulse falls at a random phase within the 8 ns; a window that holds the whole (band-limited) pulse sums to nearly the same area at any phase, but a pulse shorter than about three samples, or a window edge on the pulse's slope, makes the integral depend on the phase: keep the gates' edges away from the pulse. The trigger's time is recorded to 1 ns in every SHOT record (2.3), so the phase can be checked or corrected afterwards. With awg on a second board as an asynchronous laser (its band-limited pulses walking through the samples), a gate holding the whole pulse varied by ±0.05 % with the trigger's nanosecond, a 2-sample gate by ±35 %.

1.3 Normalization by a reference detector

An amplified laser's pulse energy fluctuates by about 1 % from shot to shot, far more than the signal. A **reference** detector that sees the probe before the sample measures the same fluctuation: with D1 the signal (probe after the sample) and D2 the reference, the per-shot ratio Q = D1 / D2 cancels the common factor (2.7). Through the loopback with 20 % shared noise on both outputs the board measured a relative spread of 11 % for D1 and 0.16 % for Q (0.3). The ratio does not cancel the detectors' own noise, a different nonlinearity of the two detectors, or fluctuations that reach only one of them (beam pointing, an aperture): match their linear ranges and keep both beams' paths similar.

The board forms Q per shot (a mean of ratios) and also keeps ΣD1·D2 per point, so the ratio of means ⟨D1⟩/⟨D2⟩ and its standard error follow from the same sums (the covariance term) when the reference is too noisy for a per-shot division.

1.4 Chopping and categories

Slow drifts (the laser's power, the alignment, the temperature) would be confused with a signal if the pump-on and pump-off measurements were made minutes apart. The pump is therefore **modulated** shot by shot: a mechanical chopper at half the repetition rate (its reference on CHOP; the guard counts to the clock where CHOP is read, 2 clocks after the trigger's period: make it about 16 ns longer than the edge distance to reject) or a modulator driven by the board (an AOM, a pulse picker: an output in mode *next* shows the next shot's category, 2.4). Every shot gets a **category**: 1 pump on, 0 pump off, or *invalid* for a shot within the guard time of a chopper's edge (a partly blocked pump). Then

```
ΔT/T = ⟨Q⟩₁ / ⟨Q⟩₀ − 1        ("ratio")
```

from the sums per category. The board also forms **pairs** d = Q_on − Q_off of consecutive shots of different categories (2.7): ⟨d⟩/⟨Q⟩₀ (”pairs”) rejects noise slower than the shot rate, because a pair’s two shots see nearly the same laser: with a shot-to-shot correlation ρ the pair’s variance is $2\sigma^2(1-\rho)$ instead of $2\sigma^2$. For a 1/f-dominated laser the pairs estimate has the smaller error; for white noise both agree.

1.5 Statistics: what the sums give

For every point and category the board keeps exact sums: the count n, Σx and Σx² of the eight gates, D1, D2 and Q (2.7). The service adds them over sweeps (`POINTS?`), and `point_results` gives each mean, standard deviation and standard error σ/√n, and ΔT/T with its error to first order:

$(\delta r / r)^2 = (\mathrm{sem}_1 / \langle Q\rangle_1)^2 + (\mathrm{sem}_0 / \langle Q\rangle_0)^2, \qquad r = \langle Q\rangle_1 / \langle Q\rangle_0.$

With a per-shot relative spread σ_Q of the ratio and N shots per category the error of ΔT/T is about σ_Q √(2 / N): σ_Q = 10^{-3} and N = 10^6 (about 17 minutes at 2 kHz with a chopper) give $1.4 \cdot 10^{-6}$. Shot noise of the detected photons sets a floor of σ_Q ≈ 1/√N_photons per shot. The sums are exact integers (48-bit Σx, 80-bit Σx²); a point that runs longer than 65,536 shots is reported in parts (partial POINT records) that the service adds, which also gives live updates during a long point.

1.6 Delay scans

Electronic (the board as the master, or with two lasers that accept triggers): a pulse output fires the probe laser at d_0 + T[p] after the shot, T the delay table (1024 entries: linear, logarithmic for decays over decades, or shuffled so that a drift does not follow τ), and another fires the pump on category-1 shots; the delay is exact to the 2 ns step and its jitter is the lasers’ (measured with an oscilloscope: a pulse at delay d leaves the pin d + 21.7 ns after the TTL edge at TRIG, ±0.6 ns rms; delays from 88 ns are on time). The gates follow the probe’s detector pulse with their own table (`points.gate_delays`). A pulse may come after the next trigger (16 pending pulses per output), so delays longer than the repetition period, and negative delays at high rates, are possible.

A PC-driven stage: the board takes N shots per point and waits (the hold, 2.4) until NEXT: the stage controller’s ”in position” on ARM, or the host’s `next_point()` after it moved the stage. The POINT record of each point carries its sums.

Repeating the scan in **sweeps** and adding them (the service does) averages slow drifts over the whole curve instead of letting them distort one part of it. The test source can emulate a sample with a programmed ΔT/T(τ): the `loopback-delay-scan` preset measured its 5 % exponential decay within 0.00044 of the emulated sample’s exact values (its table less the test source’s truncation, 2.9) at all 41 points (1.0).

1.7 High repetition rates

At 50 kHz … 2 MHz (oscillators, fibre lasers, pulse-picked systems) the board keeps up without dead time: the gates’ processor needs 48 clocks (384 ns) per shot, so every shot is integrated up to about 2.6 MHz (2.016 MHz measured: 100,000 of 100,000 shots, 0.3); the traces must be shorter than the period (the driver warns), and SHOT records to the PC stop being practical above about 0.5 M shots/s: the sums carry the result. With only ~500 ns between shots the pump is modulated shot by shot (pattern categories with a modulator on an output in mode *next*), and the detector’s pulse must decay within the period.

1.8 The scope: traces and segments

The gates are placed on a waveform, so boxcar also records waveforms. **Averaged traces** (2.5) sum IN1 and IN2 per category without dead time while the gates run: the pulse, the baseline, the pump’s scattered light (a pump-on trace with the probe blocked), ringing, saturation. **Segments** (2.6) write every raw shot to the board’s memory for analysis on a PC: the gates recomputed from the segments equal the SHOT records exactly (0.3), so a new gate choice can be tried offline. A single snapshot (2.5) shows one shot.

1.9 The boxcar outputs

OUT1 and OUT2 can show a running average of one value per category (2.9): A ← A + (s − A)/2^τ per shot of that category, an exponential average over about 2^τ shots (time constant 2^τ / f_cat, f_cat the category’s shot rate), shown as category 1 (*on*), category 0 (*off*) or *on* − *off*, in volts per unit of the value. Measured with an oscilloscope, OUT1 changes 0.90 µs and OUT2 1.08 µs after the end of the shot’s last gate at the IN connector (with a delay line of 7 clocks). This is the classic boxcar output for a chart recorder, a strip chart of a slow experiment, or a servo

(a lock-in-style feedback on a pump–probe signal). ΔT/T itself is formed on the host (the user's cut of 2026-11-09). Limits on every word (lockable) protect the instrument downstream.

1.10 What limits the measurement

- The laser: its shot-to-shot fluctuations (reduced by the reference, 1.3) and its slow drifts (cancelled by the categories and pairs, 1.4); pointing changes reach the two detectors differently.
- The detectors: their noise, their linear range (a saturated detector makes ΔT/T wrong, not noisy: check the traces), and pump light scattered onto the probe detector (measure with the probe blocked, subtract).
- Logic edges near the gates: a chopper's or a modulator's edge shortly before the gates on one category's shots couples into the inputs ($4{\cdot}10^{-4}$ of ΔT/T measured for an edge 60 ns before the gate; nothing when the edge is half a period away). The converters' linearity: against awg's exact ΔT/T the measurement stayed within $4{\cdot}10^{-4}$ for pulses up to 0.68 V at IN1.
- The board: 14-bit converters (138 µV per step on LV), the inputs' noise (1.2), the 8 ns gate edges (no sub-sample gates), the front end's 60 MHz, pulses shorter than about three samples (1.2). Pick-up of the pulse outputs' edges on IN1/IN2 is not measured (keep the trigger cables away from the inputs).
- Statistics: the error falls as $1/\sqrt{N}$ (1.5); the sums do not saturate (48/80 bits), the traces' 36-bit sums can (the driver warns, the flag is reported).

2. Definitions

2.1 Time and pins

- The **sample clock** runs at 125 MHz; sample n is at t = 8 n ns, counted from the core's reset (the core's counter `now`, TIME_LO/TIME_HI).
- **Inputs** DIO0..3_P (TRIG, CHOP, ARM, AUX) are sampled every 1 ns by ISERDESE2 in oversample mode (`../../common/rtl/optional/rp_pins1ns.v`): pin p's word of sample period n, w8[8p +: 8] (bit 0 the earliest nanosecond), is available in clock n + 4. LEVELS holds each pin's latest sample; SEEN is set by a rising edge anywhere in a word (0 then 1 between consecutive nanoseconds, the last nanosecond of the previous word included) and stays set until cleared. A pulse of 2 ns or more is always seen. Open inputs read low (pull-downs).
- **Outputs** P1..P8 (DIO0..7_N) and markers M1..M4 (DIO4..7_P) are serialized with 2 ns bits (`../../common/rtl/optional/rp_oserdes2ns.v`): a 4-bit word per sample period, bit 0 first. A disabled output is high impedance (OBUFT), with the FPGA's weak pull-down. In 0.0 each output is off, low, high or a fixed 4-bit pattern.
- *Planned (0.1):* the shot's time t_T of a TRIG edge in slot j of word m is 8 m + j ns (connector time, referred to OUT1 through the board calibration's `pins_delay_s`); a shot starts on the next sample; pulse outputs fire at t_T + d with width w in 2 ns steps.

2.2 The DDR masters

- **The record ring** (S_AXI_HP0): `../../common/rtl/optional/rp_ring_writer.v`, 32-bit words into a ring in DDR with a mailbox and a watchdog on the reader's heartbeat (`../../common/README.md`), magic "BOXC". In 0.0 a test source feeds it words counting up from 1, one every PERIOD + 1 clocks; from 0.1 the records (2.8).
- **The segment port** (S_AXI_HP1): a finite writer (`../../common/rtl/optional/rp_seg_writer.v` from 0.2, 0.0's `bx_hp1`; structured like ajku's `iq_capture`). From S_BASE it writes S_WORDS 32-bit words, two per 64-bit beat, in 16-beat (128-byte) bursts with up to four responses outstanding, from a 2048-beat FIFO. The base must be 128-byte aligned and at least 1 MiB, the count a nonzero multiple of 32 (at most 2^{26}), the end within DDR (0x20000000); otherwise the start is refused (error bit 0). An abort or an error stops new bursts but never retracts an offered one; busy stays set until every issued burst has its response. Its words: the segments (2.6; S_CTRL bit 2 at the start), or 0.0's test pattern S_SEED + k, one per clock (500 MB/s).
- The two ports write into two separate allocations (contiguous, uncached) at once.

2.3 Triggers (0.1)

Clock c is the sample period c; S is a run's first clock (running from S on). The pins' words: W_p(n) (pin p's word of period n, bit k = nanosecond 8n + k) is in the samplers' output during clock n + 4. Times of events are **pin times** in ns, t = 8 n + j: n the period, j the slot. Exactly one source is selected (TRIG_CFG); its events reach the shot engine (2.4) at most one per clock:

- **TTL** on TRIG (DIO0_P): the bits b_k of W_TRIG(n) (inverted for falling edges), b_{-1} the last bit of W_TRIG(n − 1). A transition at slot k (b_k = 1, b_{k−1} = 0) is an edge if the bits k … k + g of the 16 bits {W(n + 1), W(n)} are all 1 (the glitch filter, g = 0 … 7 ns: a pulse must last g + 1 ns); the first such k is the edge, t = 8 n + k. Decided in clock n + 5 (W(n + 1) arrives then), the event is in the engine in clock n + 6. **Hold-off** H (clocks): after an edge of period n the edges of periods n + 1 … n + H are ignored (counted, IGNORED). Edges are detected also without a run, but count only in one.
- **Analog**: x(a), the offset-corrected sample of IN1 or IN2 (2.5) of the ADC port's clock a, through a Schmitt trigger (rising: armed once x ≤ T − H, fires at the first armed x ≥ T, then disarmed; falling: on −x): the event, in the engine in clock a + 2, has t = 8 (a − DA) (DA: the driver's alignment of the ADC port to pin time, from the board calibration).
- **Timer** (the master): ticks in clocks S, S + P, S + 2 P, … (P ≥ 8); a tick of clock τ is in the engine in clock τ + 1 with t = 8 τ.
- **Software**: a write of RUN_CTRL bit 2 committed in clock w is in the engine in clock w + 1 with t = 8 w (only with the software source selected).
- **A run's start** (the clock W = S − 1) restarts the hold-off, the Schmitt trigger (disarmed until a sample x ≤ T − H) and the timer; the run's events are those decided from S on (one decided in W, from the state before the run, is dropped). An edge or a pulse already on its way at the start can be the run's first shot with t a little before 8 S: down to 8 (S − 5) for TTL (an edge of period S − 5, decided in S) and 8 (S − DA) for the analog trigger (a pulse that reached the connector before the start and the ADC port after it). A free-running laser does this, and so does the test source (2.9), which keeps running after a run ends.

2.4 Shots, points, categories and pulse outputs (0.1)

The engine, in a clock c of a run with an event of time t:

- the trigger is counted (TRIGGERS) and the **pattern** position q advances (q + 1 mod L): the event's pattern bit is PAT[q] (before advancing);
- it is **accepted** as a shot if the busy counter is 0 and (with the ARM gate) ARM's latest sample (bit 7 of W_ARM(c − 4)) is 1; a shot sets the busy counter to LB − 1 (LB ≥ 1 clocks: events in clocks c + 1 … c + LB − 1 are **missed**, counted, if they would have been accepted otherwise); the counter counts down each clock;
- the shot's **category**: from the pattern, PAT[q]; or from CHOP: its latest sample (bit 7 of W_CHOP(c − 4)) XOR the inversion, or *invalid* (2) if CHOP changed in the last G clocks: a clock "has a CHOP edge" if its word W_CHOP(c − 4) differs from its own previous bit somewhere (bit 7 of the word before included); the age of the last edge is 0 in such a clock, else the previous clock's age + 1 (saturating at 255); invalid if age < G (G = 0: never);
- the shot takes the current **point** p and sweep s, and the counts: SHOTS + 1.

Points (POINT_CFG): P points per sweep (1 … 1024); the point advances after a shot (per shot), after every N shots, at each rising edge of ARM (NEXT: any rising transition in W_ARM(c − 4) in clock c; a shot in the same clock still takes the old point; ARM then gates nothing), or only when the host sets it (fixed). Advancing from P − 1 wraps to 0 and counts a sweep. **Hold** (POINT_CFG [13], with points every N; 0.3): after a point's N-th shot (its advance) no shot is accepted (the events count as triggers only) until NEXT (ARM's rising edge, or the host's NEXT), which releases without advancing: the board takes N shots per point and waits for a PC-driven stage. **The point epoch** (0.3) counts the point's changes: it increments at each advance, NEXT, POINT_SET, and at a run's start and end; a shot takes the epoch of its point. **Run limits**: a shot count (the run's last shot), a sweep count (the run ends with the wrap that completes it), a duration D (the last running clock S + D − 1), or the host's STOP (a write committed in clock w: the last running clock w). A run's state (points, sweeps, counts) starts at 0.

Pulse outputs P1–P8 (mode *pulse*): an accepted shot of category k with the output's enable bit k set schedules a pulse from F = t + d0 + (scan ? T[p] : 0) to E = F + w (ns, 32-bit modulo 2^{32}: delays and widths below 2^{31} ns); T is the delay table (1024 words, ns; the driver fills it with any grid: linear, logarithmic, shuffled). The entry is pushed into the output's FIFO (16 entries; a full FIFO drops it, counted) in clock c + 3. A generator plays the FIFO's head: in clock c the 2 ns slot q (time t_q = 8 c + 2 q) is high if F ≤ t_q < E (32-bit signed differences); a head is replaced in the clock its E ≤ 8 c + 8 (the next entry, if any, plays from the next clock: pulses of one output need E of one ≤ F of the next − 8 ns to keep their own starts); a head that becomes current with F < 8 (c + 1) starts late (counted, LATE). The pin's word of clock c is driven in the following period (d4 in clock c + 1), so an output and an input jumpered together see the loop's latency; pin times are referred to the connectors by the board calibration (`pins_delay_s`, `pins_loop_s`).

Output modes (all twelve outputs P1–P8, M1–M4; OUT_MODE, each with its inversion): off (high impedance), low, high, a 4-bit pattern, *pulse* (P1–P8 only), and levels of whole 8 ns words: shot, point, sweep and trigger markers (high for MARK_W clocks from the clock after an accepted shot, the first shot of a point, the first shot of a sweep,

any event of a run), the last shot's category (high for 1), the next pattern bit PAT[q] (a chopper's or modulator's drive), run (high while running), and the test source's sync and chopper levels (2.9).

2.5 The input path, the snapshot (0.1) and the averaged traces (0.2)

- **Offsets**: x_k(a) = sat16(IN_k(a) − OFF_k) per input (the ADC's 16-bit words; OFF from the board calibration); **overload** when IN_k(a) = −32768 or ≥ 32764 (the 14-bit code at a rail).
- **The delay line**: x(a) (both inputs, with their overload bits) into a ring of 8,192 samples, read as one **stream** y(τ) = x(τ − 7 − DL) in clock τ (DL = 2 … 8000 clocks). Every consumer of a shot takes its samples from the stream: a consumer with offset OFF started by an accepted shot of clock c (event period n) has the **start** st = n + OFF, or st = c + 1 if n + OFF ≤ c (*late*); its sample i is x(st − DL + i), in the stream in clock st + 7 + i. A consumer decides in clock st + 2 whether it takes the shot. With DL and the offsets from the board calibration the driver puts a consumer's sample `pre` at the IN connector's time of the shot; DL serves the consumer with the most samples before the shot, the others wait longer after it.
- **Waiting starts**: the traces and the segments each keep up to 16 shots waiting for their starts (in the order of the shots; a shot enters in clock c + 2 and leaves at its decision): a shot of clock c that finds 16 earlier shots of that consumer with starts st ≥ c has none (counted as missed).
- **The snapshot**: armed by the host, the next shot's K samples (1 … 4096) of {x_2, x_1} from its start (offset SOFF), into a block RAM the host reads.
- **Averaged traces** (0.2): a trace for each accepted shot of category 0 or 1 (category mode, into the bank of its category; invalid shots have none) or of any category (single mode, one bank of 4,096 points: the points from 2,048 on in bank 1). A trace has its start at offset TOFF and K points (1 … 2,048 in category mode, 1 … 4,096 single), each the sum over k samples (1 … 64) of each input: point m = (Σ_{i<k} x_1(f + mk + i), Σ_{i<k} x_2(f + mk + i)), f = st − DL its first sample. It is **taken** (decided in clock st + 2) if its first sample comes at least 5 samples after the last of the previous taken trace (the averager reads a point's address again no sooner than 4 clocks later); otherwise it is missed (TR_MISSED). Point m goes into the averager in clock st + 7 + (m + 1) k.
- **The averager** is the common `rp_avg` (AW 11, IN_W 24, SUM_W 36, scans per bank, automatic host reads; `../../common/README.md`): two halves of both banks. A taken trace counts its shot in its bank (category mode: its category; single mode: bank 0) and goes into the half active in its decision clock. A **swap** (the host's request, or N scans: AVG_N over both banks of the active half) happens at a taken trace's decision if its shot may swap (any shot, or with "swap at points" only a point's first shot), into a free half only; without a run a request swaps as soon as the other half is free. The closed half **freezes** when its last trace's last point has gone in (a BLOCK record 32 clocks later); the host reads it (TR_ADDR, then TR_DATA, three words per point: I [31:0], Q [31:0], {Q [35:32] at bit 8, I [35:32] at bit 0}, bank 0's 2,048 points, then bank 1's) and releases it; it is cleared (2,048 clocks without a point to write) and free again. Sums saturate at $\pm 2^{35}$ (flagged). The averaged trace of a bank is the sum over its scans: in volts I / (scans · k) through the board calibration.

2.6 Segments (0.2)

Armed by the host (S_CTRL with segments selected) with a base address, a count N, a length L (1 … 8,192 samples), an offset GOFF and a number M per point, the accepted shots that follow (all of them, or with M ≠ 0 the first M shots of each point since the arming) each write one segment through the segment port (S_AXI_HP1) into the host's allocation: S = 32 ⌈(L + 4)/32⌉ words (whole 128-byte bursts), segment j at base + 4 S j:

Word	Contents
0	[31:24] 0xB5, [23] late, [22:21] the category, [9:0] the point
1	the shot's number (from 1, in its run)
2	t [31:0] (ns)
3	{sweep [11:0], t [51:32]}
4 … L + 3	the samples i = 0 … L − 1 from its start (offset GOFF): {x_2 [31:16], x_1 [15:0]}
L + 4 … S − 1	0

A segment is **taken** at its decision (clock st + 2) if the capture is armed, fewer than N segments were taken, and the previous segment has ended: segment words go one per clock, word j of a segment decided in clock τ in clock τ + 1 + j (its samples straight from the stream), so a segment is taken if the previous one was decided at least S clocks earlier; otherwise it is missed (SEG_MISSED). After N segments the capture takes no more; it is done when the N S words have all been written and acknowledged (the port writes up to 500 MB/s; a segment ends no sooner than its last sample).

2.7 Gates, derived values and point sums (0.3)

Gates. Eight gates k = 1 … 8, each on or off, on IN1 or IN2, with a start d_k (0 … 65,535 samples after the gates' start) and a width w_k (1 … 65,536 samples), optionally moved per point by the gate table (scan: d_k' = d_k + TG[p], TG 1024 unsigned 16-bit words, samples; else d_k' = d_k). The gates are a consumer of the stream (2.5) with offset G_OFF and their own queue of 16 starts: with the shot's start st and point p,

g_k = Σ_{i < w_k} x_in(k)(st − DL + d_k' + i) (32-bit, exact: |g_k| ≤ 65,536 · 32,768),

and o_k = 1 if any of those samples is an overload. The shot's **span** L = max over the gates on of d_k' + w_k (0 with no gate on); its last gate ends in clock st + 7 + L − 1.

Taken. A shot is decided in clock st + 2 (with no gate on, every shot is taken): taken if its gates start after the previous taken shot's (start st_A, span L_A) have all ended, at least 8 clocks after it, and the processor (P = 48 clocks after a shot's gates end) is free when its own end: st ≥ st_A + max(L_A, 8) and st + L ≥ st_A + L_A + P. Otherwise it is missed (GATE_MISSED), like a shot that finds 16 waiting; a missed shot has no values, no SHOT record and no part in the sums.

Derived values of a taken shot (j = 1, 2; gate indices a_j, b_j and ratios R_j, unsigned 16.16, from D_CFG, D_R1, D_R2; R_j = w_a/w_b subtracts a baseline gate of another width, 0 none):

D_j = sat32(g_a(j) − ⌊R_j · g_b(j) / 2^{16}⌋), Q = sat32(sgn(D_1) sgn(D_2) ⌊|D_1| · 2^{24} / |D_2|⌋) (D_2 = 0: Q = 0 if D_1 = 0, else 2^{31} − 1 for D_1 > 0 and −2^{31} for D_1 < 0),

so Q is D_1/D_2 (signal over reference) with 24 fractional bits; sat32 clamps to [−2^{31}, 2^{31} − 1]. **Pairs**: two consecutive accepted shots (shot numbers m, m + 1), both taken, of categories 0 and 1 in either order, of the same point epoch, the first not already in a pair: d = sat32(Q of category 1 − Q of category 0).

Point sums. The taken shots are summed in their order into the sums of the open point (the epoch of its first shot): for c = 0, 1 the shots N_c and, for x = g_1 … g_8, D_1, D_2, Q, the sums Σx (48-bit) and Σx^2 (80-bit), and ΣD_1D_2 (80-bit); the invalid shots N_inv (their values not summed); the pairs N_d, Σd, Σd^2. None can overflow (at most 65,536 shots of 32-bit values). The point **closes** (a POINT record, if on; with no gate on the sums are 0 and the counts N_c, N_inv remain) when a taken shot of another epoch comes, when the gates' path is idle (no shot accepted in the last two clocks, none waiting, none being processed) and the engine's epoch is newer (the point ended: an advance, NEXT, POINT_SET, the run's end), or after its 65,536th taken shot (a partial record: the point goes on in new sums). Sums with no shot close without a record. A record that cannot be copied before the next close is lost (POINT_LOST; REC_LOST and an ERROR record 1): points need about 110 clocks each.

2.8 Records (0.1, 0.2, 0.3)

A record is a header word (bits 31..27 the type, 26..20 the record's words with the header, 19..0 its fields) and its data words, through the ring (2.2). In 0.1:

Type	Words	Fields
1 RUN	4	[0] 1 start, 0 stop; [3:1] the stop's reason (0 host, 1 shots, 2 sweeps, 3 duration); words: clock [31:0], clock [63:32], shots
2 SWEEP	3	[19:0] the sweep; words: t [31:0], t [51:32] (its first shot's time, ns)
3 SHOT	4, or 5 + n	[9:0] the point, [11:10] the category, [12] values follow; words: t [31:0], {sweep [11:0], t [51:32]}, the shot's number (from 1); with a gate on: {o_8 … o_1} at [7:0], then the n values chosen by REC_MASK in the order g_1 … g_8, D_1, D_2, Q (0.3)
4 POINT	106	[9:0] the point, [10] partial; words: the sweep (mod 2^{16}), N_0, N_1, N_inv, N_d, then for c = 0, 1 and x = g_1 … g_8, D_1, D_2, Q, D_1D_2 four words {Σx [31:0], {Σx^2 [15:0], Σx [47:32]}, Σx^2 [47:16], Σx^2 [79:48]} (for D_1D_2 Σx = 0 and Σx^2 = ΣD_1D_2), then Σd, Σd^2 the same (0.3)
5 BLOCK	4	[0] the half, [1] saturated; words: the scans of bank 0, of bank 1, the blocks so far (0.2)
6 SEGMENT	3	[19:0] the segment's index; words: the shot's number, t [31:0] (0.2)
7 ERROR	2	[3:0] 1 records lost (word: their count), 2 late pulses (word: a mask of the outputs whose heads started late in that clock, bit k = P(k+1))

SHOT records are optional, every M-th accepted shot (REC_CTRL; the count runs at acceptance). With no gate on a SHOT record (4 words) is made at acceptance, as in 0.1; with a gate on, when the gates' path is done with the shot, in the order of the shots, for taken shots only, through a 64-word FIFO (a record that does not fit is lost). POINT records are optional (REC_CTRL [17]). A source that fires again before its record is copied loses the earlier one

(counted, REC_LOST, and reported by an ERROR record).

2.9 Analog outputs: the test source (0.1) and the boxcar outputs (0.4)

The test source is an emulated laser and sample for a cable loopback (OUT1 → IN1, OUT2 → IN2):

- **Its shots**: its own timer (laser times t_l = t_0 + i · T_PER ns, T_PER ≥ 128 ns; t_0 = 8 S + T_START with "start with the run", else 8 (the clock after it is turned on) + T_START; it keeps running after a run ends, until the next run's start or until it is turned off), or the pulses of a pulse output P_k (master mode: each entry pushed into P_k's FIFO is a laser time t_l = F; a new one replaces a pending one). A shot is processed in the first clock c with t_l − 8 c ≤ 64 ns (64 ns ahead): its category is its own pattern's bit TPAT[q_t] (q_t then advances), its noise n the low 16 bits (signed) of its own xoroshiro128+ word (stepped once per shot; seeded from T_SEED when the source is turned on), and its amplitudes a_k = ((A_k · g) >> 16), g = 65536 + ((σ · n) >> 15), for OUT k in *reference* mode, and a_k = (((A_k · g) >> 16) · (65536 + Δ)) >> 16, Δ = c_t ? DT[p] : 0 (the acquisition's current point p in the processing clock; DT a table of 1024 signed 16-bit values, ΔT/T in units of 2^{-16}), in *signal* mode. Fired by a pulse output, the shot that ends a point (points every N, per shot) is processed after the point has advanced and plays the next point's Δ: if of category 1, it moves the point's mean ΔT/T by (DT[p + 1] − DT[p]) / n_1 (n_1 the point's category-1 shots; seen on the board: +0.0005 = 0.05/100 at the point before a step of 0.05). The amplitudes are integer codes, so the sample's ΔT/T is a little below DT[p]/2^{16} (the product A·Δ/2^{16} truncated: 0.00969 for a programmed 0.01 at A = 2167, 0.3 V): `model.test_dtt` gives the exact value (averaged over the noise word) and `driver.emulated_dtt` the values of a configuration, which a measurement through the cables should give.
- **Its pulses**: OUT k in clock c + 4 is sat14(DC_k + round(a_k · Sh[i] / 2^{15})) · 4 with i = 8 c − (t_l + d_k) ns while 0 ≤ i < 1024, else sat14(DC_k) · 4; Sh is a shape table of 1024 signed 16-bit values at 1 ns (8× oversampled: the trigger's nanosecond moves the pulse within a sample), DC_k the output's level (codes, through the calibration). A newly processed shot replaces the playing pulse.
- **Its pins**: *test sync* (slot q of clock c high while 0 ≤ 8 c + 2 q − t_l < 8 · MARK_W) and *test chop* (the category of the processed shot, switching to the next shot's T_SW ns after t_l, in whole clocks).

The boxcar outputs (0.4): OUT k in mode *boxcar* (T_OUT 3) shows a running average of one of a shot's values (2.7), per category, in volts:

- **Each taken shot** (a gate on) of category c = 0 or 1 gives x_k, its value number SRC_k (g_1 … g_8, D_1, D_2, Q), scaled to DAC codes with 16 fractional bits: s_k = sat40(⌊x_k · G_k / 2^(4 h_k)⌋) (G_k signed 18-bit, h_k = 0 … 7: up to 2 codes per unit of x; the driver picks them from the volts per unit).
- **The averages** A_{k,0}, A_{k,1} (40-bit): A_{k,c} ← A_{k,c} + ⌊(s_k − A_{k,c}) / 2^τ_k⌋ (τ_k = 0 … 20: an exponential average over about 2^τ_k shots of that category); the first shot of category c after a run's start sets A_{k,c} = s_k (until then A_{k,c} counts as 0). Invalid shots and missed shots change nothing; the averages hold after the run ends.
- **The output**: v = A_{k,1} (category 1: *on*), A_{k,0} (*off*) or A_{k,1} − A_{k,0} (*difference*); the code sat14(DC_k + ⌊v / 2^{16}⌋). (ΔT/T is not formed on the board: the user's choice of 2026-11-09, for room; the POINT totals give it on the host.)
- **Timing**: a shot's value reaches the outputs in clock X = st + L + 59 (st its start, L its span); OUT 1's word changes in clock X + 23, OUT 2's in X + 46 (τ ≤ 20 shifts fit in a slot of 23 clocks): about 1 µs after the shot's last gate.

Limits (0.4): every word to OUT k, in any mode, is clamped to LO_k … HI_k (14-bit codes; power-up the full range). Writing 1 to LIM_LOCK locks the limits until the FPGA is loaded again.

boxcar commands (TCP)

The board's service (`rp-overlay.service`, port 5025) takes the common commands of every design (`*IDN?`, `STATUS?`, `CONFIG?`, `APPLY`, `PRESET`, `SAFE`, `LOAD`, `CAL ...`; `../../common/README.md`) and, while boxcar runs, these (version 1.0):

Command	Reply
`RUN`	`OK {state}`: starts a run (its counts, point and sweep start at 0; with traces on, the averager's halves are cleared first: a frozen half released, the active half swapped out and released)
`STOP`	`OK {state}`
`TRIGGER`	`OK`: a software trigger (the software source)
`NEXT`	`OK`: advances the point (a PC-driven stage's step)

Command	Reply
STATE?	OK {json}: running, triggers, shots, missed, ignored (hold-off), late (pulses), dropped (a full pulse FIFO), point, sweep, t_last_ns, start_clock, records_lost, snapshot {armed, busy, done, late}, traces {taken, missed}, segments {taken, missed, armed, busy}, gates {taken, missed, points, points_lost}, held, stop_reason
SNAPSHOT ARM	OK: the next shot's snapshot
SNAPSHOT DATA?	OK {json}: t_ns, late, dt, pre, in1, in2 (volts) once taken (the core's SNAPSHOT? is the configuration and state, as for every design)
PINS? [CLEAR]	OK {json}: for TRIG, CHOP, ARM, AUX: {"pin", "level", "seen"}
HARDWARE?	OK {json}: abi, locked, the ring writer (writer_running, writer_tripped, written, bursts, stalls), the HP1 writer (segment_busy, segment_done)
TRACE SWAP	OK: the averager's active half closes at the next taken trace that may swap (without a run: at once, if the other half is free)
TRACE RELEASE	OK: releases the frozen half (cleared and free 16 µs later)
TRACE STATUS?	OK {json}: active, frozen (the frozen half or null), request_pending, closing, saturated, scans [bank 0, bank 1], blocks, this run's taken, missed
TRACE? [SWAP]	OK {json}: the next frozen half (with SWAP, a swap requested first), then released: mode, half, scans, saturated, blocks, decimation, dt, t (each point's centre, s from the shot), in1, in2 (volts: category mode [category 0's trace, category 1's], single mode one trace)
POINTS? [CLEAR]	OK {json}: the service worker's totals of the POINT records (added over sweeps) per point: records, sweeps, n [category 0, 1], n_invalid, n_pairs, and for each gate on and D1, D2 {mean, sem} per category (volts: the gate's mean level), Q (D1/D2), dtt {ratio ($\langle Q\rangle_1/\langle Q\rangle_0 - 1$), ratio_sem, pairs ($\langle d\rangle/\langle Q\rangle_0$), pairs_sem}; CLEAR starts new totals
POINTS CLEAR	OK: new totals
WORKER?	OK {json}: the record worker's counters (bytes, polls, the pump, records per type)
LIMITS LOCK	OK: locks the outputs' limits until the FPGA is loaded again (later changes of limits are ignored by the hardware)
RECORDS STREAM	the records on this connection: OK {header} then binary frames (magic BXR1, sequence, bytes, bytes lost; the records as little-endian words: model.parse_records); also served by the ring pump on port 5026 (BoxcarClient.records())
SEGMENTS? n [s]	OK {json}: captures the next n segments (timeout s, default 5; at most 1,000,000 samples in all): count, missed, t_ns, shot, category, point, sweep, late (per segment), dt, pre, in1, in2 (volts, a list per segment)

Configuration (APPLY {json}, CONFIG?)

Times in seconds, levels in volts (the board calibration), as the driver applies them (rounded to the hardware's steps; a minimum time rounds up):

- trigger: source (ttl, analog, timer, software), edge (rising, falling), min_width (TTL: 1 … 8 ns), holdoff (TTL), input (in1, in2), level, hysteresis (analog), period (timer).
- shots: busy (the time a shot blocks), limit, sweeps, duration (0: none), arm_gate.
- points: mode (shot, every, next, fixed), count (1 … 1024), every, delays (seconds per point: the delay table for outputs with scan).
- categories: source (chop, pattern), invert, guard, pattern (a string of bits, the first for the first event, up to 16).
- outputs: P1 … P8, M1 … M4: mode (off, low, high, pattern, pulse, shot, point, sweep, trigger, category, next, run, test_sync, test_chop), pattern, invert, and for pulses delay, width, categories (a list of 0, 1, "invalid"), scan.
- markers: width. records: on, shot_every. snapshot: samples, pre.
- traces: enabled, mode (categories: a trace per category of up to 2048 points; single: one of up to 4096), points, decimation (samples per point, 1 … 64), pre, scans (swap after this many traces; 0: only on request), swap_at_points (swaps only at a point's first shot).
- segments: length (samples, 1 … 8192), pre, per_point (the first n shots of each point; 0: every shot). The count is the capture's (SEGMENTS?).
- gates: G1 … G8: enabled, input (in1, in2), delay (of the gate's first sample after the shot at the IN connector; negative before it), width, scan (add the gate table's delay of the point). derived: d1, d2 {signal, baseline (a gate or null)}: D = signal − (w_signal/w_baseline) baseline, Q = D1/D2.
- points: also hold (with every: N shots per point, then wait for NEXT) and gate_delays (seconds per point, added to the scanned gates). records: also points (POINT records) and values (the values in SHOT records: G1 … G8, D1, D2, Q).

- The snapshot, the traces and the segments read one delay line (65 µs): the one with the most samples before the shot sets it, the others wait for their start. `STATUS?` shows the clocks (`delay_line_clocks`, `snapshot_offset_clocks`, `trace_offset_clocks`, `segment_offset_clocks`, `gate_offset_clocks`) and `warnings` (a trace longer than the shots' least spacing, sums that may saturate).
- `test`: `on`, `fired_by` (null: its own timer, or `P1` … `P8`), `with_run`, `start`, `period`, `pattern`, `chop_switch`, `noise` (relative), `seed`, `out1`/`out2` {`mode` (`off`, `signal`, `reference`), `amplitude`, `delay`}, `shape` ({`rise`, `decay`} or 1024 values at 1 ns), `dtt` (ΔT/T per point).
- `dc`: `out1`, `out2` (the outputs' levels; a boxcar output's zero).
- `analog`: `out1`, `out2`: `mode` (`off`, `boxcar`: needs `test.outK.mode` off), `source` (`G1` … `G8`, `D1`, `D2`, `Q`), `show` (`on`: category 1, `off`: category 0, `difference`), `average` (shots, rounded to a power of 2, 1 … 2^{20}), `scale` (volts at the output per unit of the source: a gate's or D's mean voltage at the input, or Q). ΔT/T is not formed on the board (POINTS?).
- `limits`: `out1`, `out2` {`low`, `high`} (volts; null: the output's range): every word to the output in any mode stays within them.

Presets (`PRESET <name>`, `PRESETS?`; README.md, *Presets*): `off`, `loopback-demo`, `loopback-delay-scan`, `amplified-chopper`, `high-rep-modulator`, `master-delay-scan`, `stage-scan`, `two-signals`, `scope`.

boxcar registers

`bx_core` is a 256 KiB AXI4-Lite window at **0x43C00000** (32-bit words, byte offsets). Access it only through the driver (`rpfpga.Overlay`): a read or write while the FPGA is not configured hangs the board (`../../common/README.md`, "Safe loading"). The common sysid block is at 0x40000000 (design ID `BOXC`, version 1.0).

Layout version (ABI) 5, boxcar 1.0 (the layout of 0.4). A read takes three clocks. The control registers are below 0x1000, the tables and the snapshot above. Unused offsets read 0 and ignore writes. The definitions are in THEORY section 2; `control/boxcar/model.py` (`register_writes`) is the one encoding of every field, used by the driver and the benches.

Pins, the DDR writers (0.0)

Offset	Name	Access	Meaning
0x000	ABI	R	the register layout's version (5)
0x004	STATUS	R	[0] the 1 ns samplers' PLL locked; [3] the ring writer runs; [4] its watchdog tripped; [5] the HP1 writer busy; [6] done
0x008	LEVELS	R	each input's latest 1 ns sample: [0] TRIG (DIO0_P), [1] CHOP (DIO1_P), [2] ARM (DIO2_P), [3] AUX (DIO3_P)
0x00C	SEEN	R / W1C	a rising edge since the last clear, same bits; writing 1 clears a bit
0x010	OUT_EN	RW	[7:0] P1..P8 (DIO0..7_N), [11:8] M1..M4 (DIO4..7_P) driven; 0 = high impedance (power-up, SAFE)
0x014	OUT_PAT	RW	P1..P8 static patterns, 4 bits each (bit 0 the first 2 ns of every 8 ns)
0x018	OUT_PAT_M	RW	M1..M4 static patterns, [15:0]
0x01C	OUT_INV	RW	[11:0] each output inverted (not in mode off)
0x020, 0x024	TIME_LO, TIME_HI	R	sample periods since the core's reset (reading LO latches HI)
0x028	OUT_MODE0	RW	P1..P8 modes, 4 bits each: 0 off, 1 low, 2 high, 3 pattern, 4 pulse, 5 shot, 6 point, 7 sweep, 8 trigger, 9 category, 10 next pattern bit, 11 run, 12 test sync, 13 test chop
0x02C	OUT_MODE1	RW	M1..M4 modes, [15:0] (pulse not available)
0x040–0x064	W_CTRL … W_STALLS, TEST, TEST_DROPS		the ring writer on S_AXI_HP0 and its test words (0.0; the records wait while TEST is on)
0x080	S_CTRL	W	the segment port (S_AXI_HP1, `rp_seg_writer`): [0] start (when idle), [1] abort, [2] with start: the segments (THEORY 2.6); else 0.0's counting pattern S_SEED, S_SEED + 1, … one per clock
0x084	S_BASE	RW	the allocation's physical address (128-byte aligned, ≥ 1 MiB)

Offset	Name	Access	Meaning
0x088	S_WORDS	RW	words to write (a nonzero multiple of 32, ≤ 2^{26}; segments: N S)
0x08C	S_SEED	RW	the pattern's first word
0x090	S_STATUS	R	[0] busy, [1] done, [2] a segment capture armed (fewer than N taken), [15:8] errors: [8] bad configuration, [9] aborted, [10] FIFO overflow, [11] a word missing (pattern), [12] a write response error
0x094, 0x098	S_ACCEPTED, S_ACKED	R	words taken into the FIFO, words written and acknowledged
0x09C	S_HIGH	R	the FIFO's largest occupancy (64-bit beats of 2048)

The run, the triggers, the engine (0x100)

Offset	Name	Access	Meaning
0x100	RUN_CTRL	W	[0] start (when stopped), [1] stop, [2] software trigger, [3] arm the snapshot, [4] advance the point
0x104	RUN_STATUS	R	[0] running, [1] snapshot armed, [2] snapshot busy, [3] snapshot done, [4] snapshot late, [7:5] the last stop's reason (0 host, 1 shots, 2 sweeps, 3 duration), [8] held (waiting for NEXT, 0.3)
0x108	TRIG_CFG	RW	[1:0] source (0 TTL, 1 analog, 2 timer, 3 software), [2] TTL falling, [5:3] the glitch filter g (a pulse must last g + 1 ns), [6] analog on IN2, [7] analog falling
0x10C	HOLDOFF	RW	[15:0] TTL hold-off (clocks)
0x110	AN_THR	RW	[15:0] the analog threshold T (signed, offset-corrected ADC words; on −x when falling)
0x114	AN_HYS	RW	[15:0] its hysteresis H (armed at x ≤ T − H)
0x118	AN_DA	RW	[15:0] the analog events' alignment: t = 8 (a − DA)
0x11C	TIMER	RW	the timer's period P (clocks, ≥ 8)
0x120	BUSY	RW	[15:0] LB: a shot blocks the next LB − 1 clocks
0x124	SHOTS_LIM	RW	the run's shots (0: no limit)
0x128	SWEEPS_LIM	RW	the run's sweeps (0: no limit)
0x12C	DURATION	RW	the run's clocks (0: no limit)
0x130	POINT_CFG	RW	[1:0] mode (0 per shot, 1 every N, 2 NEXT on ARM, 3 fixed), [11:2] P − 1, [12] the ARM gate, [13] hold (every N: wait for NEXT after a point's N shots; 0.3)
0x134	POINT_N	RW	[15:0] N − 1
0x138	CAT_CFG	RW	[0] source (0 CHOP, 1 pattern), [1] CHOP inverted, [9:2] guard G (clocks), [13:10] pattern length − 1
0x13C	PATTERN	RW	[15:0] the pattern (bit 0 the first event's)
0x140	MARK_W	RW	[7:0] the markers' and the test sync's width (clocks)
0x144	REC_CTRL	RW	[0] records on, [16:1] SHOT records every M-th shot (0: none), [17] POINT records (0.3)
0x148	POINT_SET	W	[9:0] the point (the host's)
0x14C–0x154	TBL_ID, DT_ID, SHAPE_ID	RW	the tables' identities (the driver writes a table only when its identity changes; TG_ID at 0x494)
0x160–0x174	TRIGGERS, SHOTS, MISSED, IGNORED, LATE, DROPPED	R	the run's counts (from its start)
0x178, 0x17C	POINT, SWEEP	R	the current point and sweep
0x180, 0x184	TLAST_LO, TLAST_HI	R	the last shot's time t (ns, from the core's reset)
0x188, 0x18C	START_LO, START_HI	R	the run's first clock S
0x190	REC_LOST	R	records lost
0x194, 0x198	SNAP_TLO, SNAP_THI	R	the snapshot's shot time

Pulse outputs (0x200 + 16 k, k = 0 … 7 for P1 … P8)

Offset	Name	Meaning
+0x0	P_D0	the delay d0 (ns)
+0x4	P_W	the width w (ns)
+0x8	P_CFG	[2:0] the categories enabled (0, 1, invalid), [3] scan: add the delay table's word of the point

The input path (0x300) and the test source (0x340)

Offset	Name	Meaning
0x300, 0x304	IN_OFF1, IN_OFF2	the inputs' offsets (ADC words, from the calibration)
0x308	DL	the delay line's delay (clocks, 2 … 8190)
0x30C	SNAP_K	the snapshot's samples (1 … 4096)
0x310	SNAP_OFF	its start after the shot's period (clocks; late if not after the shot's clock)
0x340	T_CTRL	[0] on, [1] master (fired by output P(k+1)), [4:2] k, [5] start with the run (power-up 1)
0x344	T_START	the first laser time after its start (ns)
0x348	T_PER	the laser's period (ns, ≥ 128)
0x34C	T_PAT	[15:0] the test pattern, [19:16] its length − 1
0x350	T_SW	the chopper level's switch after each laser time (ns)
0x354	T_SIGMA	[15:0] σ: the noise's amplitude (65536 = ±100 %)
0x358, 0x35C	T_SEED_LO, T_SEED_HI	the noise generator's seed
0x360	T_OUT	[1:0] OUT1, [3:2] OUT2: 0 off (the DC level), 1 signal, 2 reference, 3 the boxcar output (0.4)
0x364, 0x368	T_A1, T_A2	the amplitudes (14-bit codes, signed)
0x36C, 0x370	T_D1, T_D2	the pulses' delays after the laser time (ns)
0x374, 0x378	OUT_DC1, OUT_DC2	the outputs' levels (14-bit codes)

Averaged traces (0x380) and segments (0x3C0), 0.2

Offset	Name	Access	Meaning
0x380	TR_CFG	RW	[0] traces on, [1] single mode (one bank of 4096 points; else a bank per category), [2] swap only at a point's first shot, [13:8] k − 1 (samples per point)
0x384	TR_LEN	RW	[11:0] K − 1 (points per trace)
0x388	TR_OFF	RW	[15:0] TOFF (clocks after the shot's period)
0x38C	AVG_N	RW	swap after N scans of the active half (both banks; 0: on request only)
0x390	TR_CTRL	W	[0] request a swap, [1] release the frozen half
0x394	TR_STATUS	R	[0] the active half, [1] a half frozen, [2] which, [3] a request pending, [4] the old half closing, [5] the frozen half saturated, [31:16] blocks frozen (since power-up)
0x398, 0x39C	TR_SCANS0, TR_SCANS1	R	the frozen half's scans of bank 0 and bank 1
0x3A0	TR_ADDR	RW	the next point to read {bank [11], point [10:0]}; a write restarts at word 0
0x3A4	TR_DATA	R	the frozen half's next word (I [31:0], Q [31:0], {Q [35:32] at bit 8, I [35:32] at bit 0}; then the next point); 0 while none is frozen
0x3A8, 0x3AC	TR_TAKEN, TR_MISSED	R	this run's traces taken, missed (a full queue or the 5-sample rule)
0x3C0	SEG_CFG	RW	[12:0] L − 1 (samples per segment), [31:16] M (the first M shots of each point; 0: every shot)
0x3C4	SEG_N	RW	N, the segments of a capture
0x3C8	SEG_OFF	RW	[15:0] GOFF (clocks after the shot's period)
0x3CC, 0x3D0	SEG_TAKEN, SEG_MISSED	R	the capture's segments taken, missed (a full queue or the previous segment still written)

Gates (0x400 + 16 k, k = 0 … 7 for G1 … G8; 0.3)

Offset	Name	Meaning
+0x0	G_CFG	[0] on, [1] on IN2 (else IN1), [2] scan: add the gate table's word of the point to the start
+0x4	G_D	[15:0] the start d (samples after the gates' start)
+0x8	G_W	[15:0] the width w − 1 (samples)

Offset	Name	Access	Meaning
0x480	D_CFG	RW	the derived values' gates: [2:0] a1, [5:3] b1, [8:6] a2, [11:9] b2 (0 = G1; power-up D1 = G1 − G2, D2 = G3 − G4)
0x484, 0x488	D_R1, D_R2	RW	the baseline ratios R (unsigned 16.16: w_a / w_b; 0 no baseline)
0x48C	G_OFF	RW	[15:0] the gates' offset (clocks after the shot's period)
0x490	REC_MASK	RW	[10:0] the values in SHOT records: G1 … G8, D1, D2, Q
0x494	TG_ID	RW	the gate table's identity
0x4A0, 0x4A4	G_TAKEN, G_MISSED	R	this run's shots taken and missed by the gates (the rule or a full queue)
0x4A8, 0x4AC	POINTS, PT_LOST	R	this run's points closed (with shots), POINT records lost

The boxcar outputs and the limits (0x4B0; 0.4)

Offset	Name	Access	Meaning
0x4B0, 0x4B4	B_CFG1, B_CFG2	RW	OUT1, OUT2: [3:0] the source (0–7 G1–G8, 8 D1, 9 D2, 10 Q), [5:4] the mode (0 category 1, 1 category 0, 2 their difference), [10:6] τ (the average over about 2^τ shots, 0–20), [13:11] h (the scale's shift 4 h)
0x4B8, 0x4BC	B_G1, B_G2	RW	[17:0] the gains G (signed): codes × 2^{16} per unit of the source = G / 2^(4 h)
0x4C0, 0x4C4	OUT_LIM1, OUT_LIM2	RW	the limits of every word to OUT1, OUT2: [13:0] low, [29:16] high (signed codes; power-up −8192, 8191); ignored once locked
0x4C8	LIM_LOCK	RW	[0] writing 1 locks the limits until the FPGA is loaded again

Block RAMs

Offset	Name	Access	Meaning
0x08000	DTABLE	W	the delay table: 1024 words (ns)
0x09000	DT	W	the test source's ΔT/T per point: 1024 signed 16-bit (2^{-16})
0x0A000	SHAPE	W	the test pulse: 1024 signed 16-bit at 1 ns (32767 = 1)
0x0B000	TG	W	the gate table: 1024 unsigned 16-bit (samples) added to the scanned gates' starts per point (0.3)
0x10000	SNAP	R	the snapshot: 4096 words {x2 [31:16], x1 [15:0]} (offset-corrected ADC words)

Verification record

What was tested on boxcar, when, and with what result, newest first. Only results that were actually obtained are listed; anything not yet done is said so. Every step is tested on the board before the next one starts (the user's decision, 2026-11-08).

How to repeat the checks: `PATH=~/miniforge3/envs/base-tools/bin:$PATH make test`, `make sim`, `make`, `python3 ../common/tests/verify_build.py --design boxcar`; on the board `make upload`, `make install-service`, `make test-hw` (twice).

2026-11-10: 1.0, the oscilloscope bench

Build 0xe4e10263 with awg on the second board (10 kHz sync on D0 into TRIG, asynchronous: 4.3 ppm) and a Siglent SDS2504X Plus (1 GSa/s, crossings interpolated at 1.65 V). Probes: CH1 on awg D3 (a copy of the sync in the same pattern words), CH2 on boxcar P3 (DIO2_N, a free pin); their skew, both on copies of the sync first, +0.01 ns (±0.02 ns rms); CH3, CH4 coax (1 MΩ) on OUT1, OUT2 (the cables were found swapped and identified with

DC levels of +0.3 V and −0.3 V). Receipt `data/board-1.0-scope-bench-2026-11-10.json`; scripts `.build/session/sb*.py` (untracked).

- **Pulse outputs**: P3's edge came the programmed delay + 21.7 ns after the TTL edge for every delay from 88 ns to 10 µs (+21.5 … +21.9 ns; ±0.6 ns rms, the 2 ns slots); 202, 204, 206 ns gave 2 ns steps within ±0.2 ns. Delays up to 80 ns are late (every shot counted LATE): the pulse still starts, at about 106 ns, ±2.3 ns rms (whole clocks). A delay of 0 with a 40 ns width never appears (its end has passed when it is loaded).
- **Level outputs** (mode next with the pattern 10, the shot and trigger markers): 74 ns after the TTL edge, ±2.2 ns rms (whole clocks).
- **The boxcar outputs** (OUT1 and OUT2 showing D1 of the last pump-on shot, average 1; awg's pump-on pulses alternating between two heights): OUT1's step 1,109 ns after the TTL edge, OUT2's 183.9 ns later (the design: 23 clocks), ±2.3 ns rms; the signal gate ended 208 ns after the edge, so OUT1 follows the end of the last gate at IN by 0.90 µs and OUT2 by 1.08 µs: the processing's 75 and 98 clocks (600 and 784 ns), the delay line (7 + 7 clocks here) and about 190 ns of the two converters.
- On the way: the scope's SCPI server wedged when a script was interrupted during a waveform transfer (it accepted connections and answered nothing, on ports 5025 and 5024) and needed a power cycle; the session's acquisition helper now confirms the SINGLE command (`*OPC?`) before polling for the end of an acquisition (it could read the previous record's "Stop").

2026-11-09: 1.0, awg on a second board as the laser

A second Red Pitaya (192.168.0.104, awg 3.0, build 0x97cda2f3, nominal calibration) played an independent laser for build 0xe4e10263. Wiring (A: awg, B: boxcar): A D0 → B TRIG, A D1 → B CHOP, A D2 → B ARM, B P1 → A TRIG A, ground, A OUT1 → B IN1, A OUT2 → B IN2 (each checked from its receiving side first; OUT2 → IN2 had to be reseated). awg looped two shots (pump on, off) of its sequencer: a pulse (rise 10 ns, decay 40 ns, smooth filter) on each output, OUT1 2 % larger on the pump-on shot, and per shot a digital pattern (the sync on D0, the chopper on D1, its edges half a period from the shots). Receipt `data/board-1.0-awg-session-2026-11-09.json`; scripts `.build/session/` (untracked).

- **Rates**: 1 kHz, 10 kHz, 100 kHz, 1 MHz, 1.95 MHz (2,000 to 4,000,000 shots): every trigger a shot, every shot's gates taken, none missed, ignored or invalid; the categories alternated exactly with awg's chopper (SHOT records); the shots' intervals within 1 ns, awg's crystal 4.3 ppm fast.
- **ΔT/T against awg's exact value** (from its quantized points): 0.02024–0.02030 for 0.019984 at the five rates. Against the pulse height (0.14–0.80 V at IN1) and the sign of ΔT/T the deviation is an offset of up to $\pm 4 \cdot 10^{-4}$ that changes with the height ($8.6 \cdot 10^{-4}$ at 0.80 V), not a gain; with ΔT/T programmed 0 it is $3–4 \cdot 10^{-5}$: the converters' (awg's and boxcar's) linearity at the 10^{-4} level.
- **A chopper edge near the gate**: with the chopper a short pulse at the pump-on sync (its falling edge 60 ns before the gate) ΔT/T = 0 read $2.8–4.3 \cdot 10^{-4}$; with the same awg pattern but D1 in high impedance (no edge on the wire), $1–2 \cdot 10^{-5}$. The edges couple into the inputs (D2 more than D1): the presets keep them half a period from the shots; documented in the manuals and THEORY 1.10.
- **Shared noise** written into awg's segments (a common factor per shot, 5.78 %): D1 and D2 spread 5.72 % and 5.71 %, Q 0.257 %; ΔT/T 0.01990 ± 0.000026 (awg 0.019984).
- **The sampling phase**, awg's pulses walking through boxcar's samples (4.3 ppm, 10 kHz, 20,000 shots, every nanosecond about equally): a 2-sample gate ±35 %, a gate holding the whole pulse ±0.05 %.
- **The chopper guard**: a CHOP edge 200 ns before each pump-on trigger: no shot invalid with a guard up to 216 ns, 9,934 of 10,000 pump-on shots at 224 ns, all at 232 ns (the guard counts to the clock where CHOP is read, 2 clocks after the trigger's period).
- **The hold** released by awg's D2 (a pulse every 100 shots): exactly 30 shots in each of 10 points, each starting at its NEXT; 948 triggers for 300 shots.
- **boxcar as the master** (P1 → awg TRIG A, awg answering with one pulse): the pulse rose at IN1 392 ns after P1's edge; a delay scan of 21 points (0.5–10.5 µs, 5 sweeps) with the gates following: D1 0.12327–0.12348 V (±0.05 %, the whole-pulse gate's phase dependence), no awg trigger missed.
- Not yet done: the oscilloscope bench (the trigger-to-pin latency, the boxcar outputs' latency).

2026-11-09: 1.0, the release (plan step 1.0)

- **No RTL change** but the sysid version (1.0): build 0xe4e10263 (the build ID hashes the sources, the version among them), WNS +0.137 ns, WHS +0.026 ns, DRC without errors (DSP pipelining warnings, a SmartConnect reset net without loads, advisories), 4,270 slices (97.05 %), 12,581 LUTs (71.5 %), 43.5 block RAM tiles, 32 DSP48; placement ExtraNetDelay_low as 0.4. `verify_build` 8 of 8 (a fresh build from the recipe reproduced 0xe4e10263 with the same timing). `make sim`: tb_core, tb_shot, tb_avg, tb_trig passed.

- **Host side:** nine presets (`off`, `loopback-demo`, `loopback-delay-scan`, `amplified-chopper`, `high-rep-modulator`, `master-delay-scan`, `stage-scan`, `two-signals`, `scope`; each loads without the driver's warnings and reads back from the registers, the tables checked in the registers: `test_presets_load_and_read_back`), the Jupyter panel (`gui.py`, `GuiTests`), the PC window `tools/live.py` (`tests/test_tools.py`: a frame from a fake client and one through a local service; on the board: the loopback delay scan's traces and curve drawn live), the notebook, `RECIPES.md` (every block parses; the settings it applies pass the driver without warnings), THEORY section 1, `BoxcarRecords.collect`.
- **Found and fixed:** the driver's ADC snapshot was named `snapshot()`, which shadowed `rpfpga.Overlay.snapshot()` (the atomic configuration and state that the service returns after a change, and the core's `SNAPSHOT?`): `PRESET` through the service waited for a shot ("no snapshot in 2.0 s"), seen when the board's service first loaded a preset. Now `read_snapshot()` and `SNAPSHOT DATA?`; a client test loads a preset and reads the common snapshot through a local service. The test source plays the next point's ΔT/T for the shot that ends a point (it reads the point after the advance; THEORY 2.9 now says so): +0.0005 = 0.05/100 at the point before the delay scan's step.
- **Board (192.168.0.102, OUT1 → IN1, OUT2 → IN2, jumpers P8 → TRIG, P7 → CHOP), `make test-hw` twice: 40 of 40 passed both times** (`data/board-1.0-test_hw-1.json`, `-2.json`): the checks of 0.4, and the presets: all nine load without warnings and read back from the registers (volts within a code); `loopback-delay-scan` (10 sweeps of 41 points) measured the emulated sample's ΔT/T(τ) = 5 % exp(−τ / 2 µs) within 0.0005 and 0.0006 of the table at every point (0.0496 and 0.0494 at the pump's point, programmed 0.0500; below, the low readings explained); the boxcar output within 9.3 mV; the start check: 8 and 6 runs of 20 with an early first shot, all within DA = 19 clocks.
- **After the release commit (the manuals' recordings, `tools/record_manual_data.py`; driver only, no RTL change):**
  - *Stale traces*: the averager keeps both halves' sums across runs (THEORY 2.5), so a run's first `trace_block` could return the previous run's block, also of another trace layout (a pulse 4.75 µs away from where it was). `run()` now starts with both halves cleared when traces are on (`clean_traces`; `test_a_run_starts_with_clean_traces`).
  - *A negative TR_OFF*: traces turned off with a pre-trigger made the trace offset negative; it read back as its 16 bits and the driver's readback check refused the configuration ("Register readback mismatch"). The traces' pre-trigger now always counts in the delay line's length (as the snapshot's and the segments'); a test compiles every preset, also with its traces, gates or test source off, and requires every register value to be 0 … $2^{32} - 1$.
  - *ΔT/T through the loopback, explained*: a 10 % step of the test source's amplitude setting reads 1.0999 (D1) and 1.1002 (D2): the chain is linear. The low ΔT/T readings come mostly from the test source itself: its amplitudes are integer codes (2167 at 0.3 V) and A·Δ/65536 is truncated, so its sample's ΔT/T is below the table (0.09979 for 0.1 with 20 % noise, 0.009755 for 0.01). `model.test_dtt` computes it bit-exactly (averaged over the noise word) and `driver.emulated_dtt` for a configuration; test_hw's delay-scan check now compares with it. Measured against it: 0.5 % low at 0.3 V (0.09928, 0.0004 at the delay scan's points, 0.009720 ± 0.0000024 at 2 MHz), within 0.02 % at 0.15 V and 0.6 V (noise off): a differential gain of the converters' chain at that level, scaling with ΔT/T. The crosstalk between the two loopback channels is 0.15–0.17 % (one output pulsing, the other silent, both pulse speeds): too small to matter.
  - *2.016 MHz through the loopback*: 10 million shots in 5.3 s, every shot's gates taken, ΔT/T 0.009720 ± 0.0000024 (1 % programmed, 0.009755 emulated).
  - *The manuals* (`docs/manual/`): the user's manual (16 pages, nine measured figures from `tools/record_manual_data.py` through the client, the board test's receipts) and the theory manual (13 pages, seven computed figures, exercises with their answers); `tests/test_manual.py`: the examples run on fake registers, both value files regenerate exactly from the stored data, both PDFs build without undefined references. Measured for them: Q's per-shot spread 0.165 % with 20 % shared noise and 0.159 % without it (the ratio removes the noise down to the converters' floor); the error of ΔT/T falls as N^−0.515 up to 10,000 shots per category (σ√(2/N): $7.5 \cdot 10^{-5}$ against $8.0 \cdot 10^{-5}$ at N = 1000); the sampling phase with an asynchronous source (every nanosecond of the grid): a 2-sample gate ±17 %, a 10-sample one ±3.4 %, one holding the whole pulse ±1.5 % for a smooth pulse; a boxcar output's step response read back within 0.8 mV of the 16-shot average.
  - *The board test after these changes* (`make test-hw`, the same build 0xe4e10263): **40 of 40 twice** (`data/board-1.0-test_hw-3.json`, `-4.json`). On the way two checks were made robust: the delay-scan check now compares with the emulated values including the point lag, within 5 standard errors + 1 % (a fixed 0.0002 failed once at 3 σ before the pump); the ring check (HP0 test words with HP1 running) failed 2 of 6 runs started from the user's configuration ("OUT OF ORDER", nothing dropped, no restart; the service's pump was not running): it now starts from the default configuration, and 10 of 10 repetitions from defaults had no break (with and without HP1).

- Not yet done: the awg second-board session (an asynchronous laser with a chopper, 1 kHz–2 MHz), the oscilloscope bench (the trigger-to-pin latency, the boxcar outputs' latency).

2026-11-09: 0.4, the boxcar outputs (plan step 0.4)

- **The room decision (the user's, 2026-11-09):** the plan's cut 2: the outputs give A_on, A_off or A_on − A_off, and ΔT/T stays on the host (`POINTS?`); no on-chip division for the outputs. No other cut: the markers, the raw gates' Σx^2 and the eight gates and eight pulse outputs stay. Room without losing features: each pulse output's head counters are single adders with multiplexed operands (`late` a clock later, as documented), and 13 host-only counters (triggers, missed events, ignored edges, late and dropped pulses, the traces taken and missed, the segments missed, the gates' shots taken and missed, points, lost POINT records, lost records) are `rp_count48` DSP48 counters (common).
- **The contract and the model:** THEORY 2.9 (the boxcar outputs: s_k = sat40(⌊x_k G_k / 2^(4 h_k)⌋), the averages per category seeded by a run's first shot of each, A += ⌊(s − A)/2^τ⌋, the codes sat14(DC + ⌊v/2^16⌋) from X = st + L + 59 (OUT1's word changes in X + 23, OUT2's in X + 46), the limits on every DAC word, LIM_LOCK) and 2.3 (a run's start, below); `control/boxcar/model.py` implements them bit-exactly (`Machine._outputs`).
- **`make sim`:** `tb_shot` passed with two new scenarios besides A–J: K (shots 48 clocks apart, so the next shot's values arrive while the outputs take this one's; OUT1 the difference of D1 between the categories, τ 0, clamped by limits on both sides; OUT2 the average of G1's category 1, τ 2, 0 until the first category-1 shot) and L (OUT1 the average of G3, τ 1; OUT2 the difference of Q's averages, τ 3, a negative gain, h = 3); every clock's DAC words against the model. Scenario A's pulse outputs now load heads 0 … 7 ns before their start (P3) and 1 … 8 ns late (P4): the late test's boundary. New directed bench `tb_trig` (below). `tb_core` (ABI 5) and `tb_avg` passed.
- **Mutation:** 20 of 20 mutants of the new RTL fail the final `tb_shot` (`.build/mut04/mutate.py`: τ's shifts, seeding, the shown category, the difference's sign, the gain's shift and each output's h and gain, DC, the slot's start, the limits, the capture clocks of D1 (OUT1), Q (OUT2) and a gate (OUT1), the outputs' category from the next shot, the folded pulse counters (late boundary, both steps, the carry), the DSP counters' clears and inputs). Three survivors on the way were sharpened into the scenarios (D1's capture on OUT1: K's OUT1 now shows D1; the category from the next shot: shots 48 clocks apart; the late test's boundary: scenario A's delays). `tb_trig` fails for the three mutants of the start's handling (the event decided in W kept, the Schmitt trigger not disarmed, the hold-off not cleared; `.build/mut04/mutate_trig.sh`).
- **Found on the board, fixed:** the first of three runs on 0x6adbb148 passed 36 of 37 checks: an analog shot 1,828 ns away from its test pulse. A repeat of that check (30 runs) gave 1 stray shot in 6,000: a run's first shot, 24 ns before 8 S. Cause: the test source keeps running after a run (as designed), so a pulse can be on its way at the next start. With the test laser at 400 ns, 16 of 40 runs had an early first shot (−8 … −168 ns from 8 S); with the source turned off before each run, 0 of 40. Shots down to 8 (S − DA) are the contract (DA 19 clocks on the test board: a pulse that reached the connector before the start and the ADC port after it), but −168 ns is DA + 2 clocks: `bx_trig` accepted an event decided in the start clock W, from the trigger's state before the run, which the model never makes. The fix drops it (`if (start) ev <= 0`); THEORY 2.3 now states the rule and the boundary (TTL down to 8 (S − 5), analog to 8 (S − DA)). tb_shot drives pins and ADC words only from S on, so the directed `tests/tb_trig.sv` checks it (the unfixed trigger gives an analog event at n = S − 21 and a TTL event at n = S − 6, as on the board); test_hw's analog check now starts with the test source off, and a new check runs 20 starts with the test laser running through them.
- **Build 0x9695f9d1:** WNS +0.115 ns, WHS +0.014 ns, DRC without errors (DSP pipelining warnings, a SmartConnect reset net without loads, advisories), 4,270 slices (97.05 %), 12,581 LUTs (71.5 %), 43.5 block RAM tiles, 32 DSP48 (13 of them the counters). The first 0.4 build (0x6adbb148, before the trigger fix; ExtraNetDelay_high: WNS +0.154 ns, 96.4 % slices) was tested on the board too (below); with the fix (one gate) the same placement gave +0.025 ns, and of nine placements from its checkpoint (`.build/impl_explore04`) ExtraNetDelay_low gave +0.111 ns (+0.115 ns in the full build), now in `design.tcl`. `verify_build` 8 of 8 (a fresh build from the recipe reproduced 0x9695f9d1 with the same timing). `make test` (1,420 tests of all designs) passed.
- **Board (192.168.0.102, OUT1 → IN1, OUT2 → IN2, jumpers P8 → TRIG, P7 → CHOP), `make test-hw` twice on 0x9695f9d1: 38 of 38 passed both times** (`data/board-0.4-test_hw-1.json`, `-2.json`): the checks of 0.3, and OUT2 as a boxcar output of D1 (pump-on shots, 1 V/V) read back by a wide gate G3 on IN2 before each shot: within 10.2 mV of the level the SHOT records' D1 values give (0.216 … 0.218 V); with a 0.05 V limit OUT2 stays at 0.0500 V (readback within 4.0 mV); the new start check: with the test laser running through 20 starts, 9 and 8 runs had an early first shot, at −144 … −32 ns from 8 S, all within DA = 19 clocks (152 ns). On 0x6adbb148: 36 of 37, then 37 of 37 twice (`data/board-0.4-0x6adbb148-*.json`, with the diagnosis above in `-analog-start.json`).

- Not yet done: the output latencies with the oscilloscope (the plan's 0.4 bench item; deferred to 1.0's oscilloscope session), the scaled values' saturation at ±2^39 (sat40; not reached by the scenarios), a refresh while slots run (the bench writes settings only between runs).

2026-11-09: 0.3, gates and sums (plan step 0.3)

- **The contract and the model:** THEORY 2.4 (the hold, the point epoch), 2.7 (the gates, the taken rule, D1, D2, Q, pairs, the point sums and their closes) and 2.8 (the SHOT records with values, POINT records) to the clock; `control/boxcar/model.py` implements them bit-exactly (and parses the records: `parse_records`, `shot_fields`, `point_fields`).
- **`make sim`:** `tb_shot` passed with three new scenarios besides A–G: H (seven gates on IN1/IN2, one scanned by the gate table, one on an overload; D1, D2 with baseline ratios 5/2 and 4/7; Q; pairs of alternating categories; points of 3 shots; SHOT records with every value; POINT records, the last closed at the run's end), I (Q's edges: D2 = 0 with D1 > 0, < 0 and = 0, $|D1| \geq 2^7 |D2|$ of both signs, a negative D2; a POINT record per shot), J (TTL shots late at the boundary, points of 4 shots held until ARM's edges, spans of 3 and 62 samples from the gate table: shots missed by the 8-clock floor, by the span and by the processor after a longer span; CHOP categories with invalid shots; two shots of different categories with a missed one between them, not a pair). `tb_core` (ABI 4) and `tb_avg` passed.
- **Mutation:** 36 of 36 mutants of the new RTL fail the final `tb_shot` (`.build/mut03/mutate.py`: the floor, the processor's 48 clocks, the span and processor rules, spans and starts with the gate table, a gate's start and width, its input and overload, the accumulators' load, D's floor, Q's sign, zero divisor, saturation and a divider bit, pairs (consecutive, once, the sign), closes at a new epoch and when idle, invalid shots, the SHOT mask and commit, the POINT word order, the products' alignment, the table's point, the late rule, the hold's set and release, the epoch at advances, the SHOT flag, the SHOT FIFO's pointer, the POINT/SHOT priority, the span pass, the result shift). Survivors on the way were sharpened into the scenarios (non-consecutive pairs, the late boundary); one was equivalent (a divider bit that always equals the sign).
- **Build 0xdfdf68d2:** WNS +0.215 ns, WHS +0.022 ns, DRC without errors (DSP pipelining warnings and advisories only), 4,260 slices (96.8 %), 12,486 LUTs (70.9 %), 43.5 block RAM tiles, 19 DSP48. The first build was too large (99.4 % slices, 13,417 LUTs, WNS −16.4 ns): the gates' span maximum is now computed one gate per clock, the eight gate sums are DSP48 accumulators, the results feed the items through a shift register and two shared selects, and the worst paths are pipelined (D in two clocks, the pair's difference at 33 bits, a running shot count). Of eight placements from the post-opt checkpoint (`.build/impl_explore03`) ExtraNetDelay_high with AggressiveExplore routing (+0.171 ns) was chosen in `design.tcl`. `verify_build` 8 of 8 (a fresh build from the recipe reproduced 0xdfdf68d2).
- **Board (192.168.0.102, OUT1 → IN1, OUT2 → IN2, jumpers P8 → TRIG, P7 → CHOP), `make test-hw` twice: 35 of 35 passed both times** (`data/board-0.3-test_hw-1.json`, `-2.json`): the checks of 0.2, and with four gates (signal and baseline on IN1 and IN2; the timer the master, the test source fired by P1, 20 % shared noise, ΔT/T 10 % on category 1): the gates recomputed from 2,000 raw segments equal every SHOT record's G1..G4, D1, D2 and Q exactly; the SHOT records summed per point and category equal the 10 POINT records exactly (counts, Σx, Σx^2, ΣD1D2, pairs); Q = D1/D2 has a relative spread of 0.16 % where D1 has 11 % (the shared noise removed); ΔT/T from the POINT totals 0.0988 … 0.0999 at the ten points (programmed 0.1000: about 1 % low, not investigated: the analog path; explained in 1.0: mostly the test source's own truncation); the hold takes 5 shots per point and waits for NEXT; at 2.016 MHz every one of 100,000 shots has its gates taken, in 100 POINT records of 1,000 shots, no record lost.
- The TTL loop check now allows a step's shots to spread over two adjacent nanoseconds (its most common value the same for every step, each 2 ns step exact): with 0.3's placement the jumper's edge lies near a 1 ns sampling boundary (3 of 1,000 shots at 22 ns instead of 21; counter saw the same dithering).
- Not yet done: the 65,536-shot partial POINT record (the model's unit test only: too long for the bench), POINT record losses at less than ~110 clocks per point (by design; not provoked on the board), D's saturation (needs very large ratios), RECORDS STREAM to a PC with the gates at full rate.

2026-11-08: 0.2, traces and segments (plan step 0.2)

- **The contract and the model:** THEORY 2.5, 2.6 and 2.8 define the delay line read as one stream by every consumer (the snapshot, the traces, the segments), the queues of 16 waiting starts, the traces' 5-sample rule, the averager's swaps (N scans, optionally only at a point's first shot) and freezes, the segments' layout and the BLOCK and SEGMENT records to the clock; `control/boxcar/model.py` implements them bit-exactly (the averager through `rpfpga.avg`, kept across runs).
- **`make sim`:** `tb_shot` passed with three new scenarios besides A–D: E (TTL, a trace per category into the averager's two banks, k = 3, K = 9: traces missed one sample short of the 5-sample rule and taken at it, two

starts waiting in the queue, N = 4 scans with the swap delayed to a point's first shot, a freeze and its BLOCK record, late 64-word segments missed while one is written, one decided S − 1 clocks after the previous, the snapshot from the same stream); F (single mode, 2050 points over both banks, each trace taken in the clock of the previous one's last sample and the swap at such a trace, the first segment of each point until N = 2 (the capture done), traces, segments and the snapshot late at the boundary n + OFF = c); G (shots every 4 clocks: both queues full, CHOP categories with invalid shots, K = 1 traces decided 4 clocks apart, the capture's count reached while shots still wait). Besides every pin and DAC word, the records (by type), the counters and the snapshot, the bench checks every segment word written through HP1 (and the bursts' addresses) and both averager halves read through TR_DATA against the model. `tb_core` (ABI 3) and `tb_avg` passed.

- **Mutation:** 28 of 28 mutants of the new RTL fail the final `tb_shot` (`.build/mut02/mutate.py`: the queue's depth, pop and overflow, the traces' pending and late rules, swaps at points, the capture's start, invalid categories, each point's half, single mode's banks and scans, the sums, segments per point, the segment writer's free word, the late bit, the last sample, the count, the stride, the counters, the stream's and the snapshot's clocks, the snapshot's late rule, the BLOCK record, TR_DATA, gaps in the writer, the engine's first-of-point and n − c). The first run left five alive (3, 5, 8, 13, 21); scenarios E, F and G were sharpened until they fail too. A real defect was found on the way: a trace taken in the clock of the previous trace's last sample switched the bank and half under that trace's last point (each point now carries its own).
- **Build 0x5922b221:** WNS +0.340 ns, WHS +0.018 ns, DRC without errors (DSP pipelining warnings only), 3,692 slices (83.9 %), 8,584 LUTs (48.8 %), 39 block RAM tiles (65 %), 7 DSP48. The first placement (ExtraTimingOpt) missed timing by 0.249 ns in the common ring writer's room check; eight placements from the post-opt checkpoint (`.build/impl_explore`) gave ExtraPostPlacementOpt +0.352 ns, chosen in `design.tcl`. `verify_build` 8 of 8 (a fresh build from the recipe reproduced 0x5922b221).
- **Board (192.168.0.102, OUT1 → IN1, jumpers P8 → TRIG, P7 → CHOP), `make test-hw` twice: 30 of 30 passed both times** (`data/board-0.2-test_hw-1.json`, `-2.json`): the checks of 0.1, and at 1 kHz, 10 kHz, 100 kHz, 500 kHz and 2.016 MHz (300 to 50,000 shots; the timer the master, the test source fired by P1 with ΔT/T = 10 % on category 1 and 5 % noise) every shot had its trace and its segment (none missed), the scans per category equalled the segments' categories, and the sum of the raw segments of each category, summed over each point's samples, equalled the averaged block exactly, IN1 and IN2 (largest difference 0 codes). The traces' mean IN1 differs by the test source's +10 % between the categories (e.g. 20.9 and 23.1 mV at 500 kHz); at 2 MHz the 224 ns window ends before that shot's pulse and holds the previous shot's tail, of the other category (the ratio inverted).
- Not yet done: the averager read at speed in the field (a frozen half of 2 × 2048 points through `read_repeated`; measured only by the board test's sizes), segments at the port's 500 MB/s limit with the ring busy at the same time beyond the test's rates, saturation of 36-bit sums on the board (the bench `tb_avg` covers it).

2026-11-08: 0.1, shots and pins (plan step 0.1)

- **The contract and the model:** THEORY 2.3–2.5, 2.8, 2.9 define the triggers, the engine, the pulse outputs, the input path, the records and the test source to the clock; `control/boxcar/model.py` implements them bit-exactly (`register_writes`: the one register encoding, used by the driver and the benches).
- **`make sim`:** `tb_shot` passed: four scenarios (TTL with the glitch filter, the hold-off, busy shots, CHOP categories with a guard band, points and sweeps, four pulse outputs including late pulses and the delay table, every marker and static mode, the snapshot, a shot limit; the analog trigger with hysteresis, a late snapshot, a duration; the timer, pattern categories, points every N, the ARM gate, the test source on its own timer with noise and ΔT/T; the test source fired by a pulse output, points on NEXT, falling edges, an inverted CHOP). Every clock's pin words (12 outputs) and DAC words agree with the model, and so do the records (by type), the counters, the point and sweep, and the snapshot. `tb_core` and `tb_avg` passed.
- **Mutation:** 35 of 35 mutants of the new RTL fail `tb_shot` (`.build/mut01/mutate.py`: the glitch filter, the hold-off, the periods, the Schmitt trigger, busy, the guard, the inversion, the pattern, the point modes, the ARM gate, the limits, the markers, the table, the categories' enables, the records, the snapshot, the pulse slots, pops and late heads, the offsets, the test source's noise, ΔT/T, rounding, chopper and lead; the snapshot's start, rewritten for timing, with two more).
- **`make test`:** 16 passed (the driver against the model's encoding, tables written only when they change, safe, snapshot in volts, the protocol, the client; the model's pieces and the vectors current).
- **Build 0xeb44a2fb:** the first build missed the +0.10 ns margin on the read mux (+0.058 ns): it is now two registered levels; the next missed on the snapshot's start (−0.067 ns, 21 levels: a 48-bit comparison into the read index): now a registered difference and a countdown; and the design takes spin's and counter's implementation settings (ExtraTimingOpt, post-place and post-route physical optimization). Final: WNS +0.191 ns, TNS 0, WHS +0.026 ns; 3,140 slices (71.4 %), 7,147 LUTs, 23 BRAM tiles, 7 DSP48; DRC only RTSTAT-10 and the DSP pipelining notes; no critical warning. `verify_build.py`: 8 of 8 checks passed; a fresh copy

from the recipe rebuilt 0xeb44a2fb with the same WNS (+0.191 ns) (a first run stopped at stage 05: `verify_build.json` still named the read mux's old line).

- **Board, `make test-hw` twice on 0xeb44a2fb** (OUT1 → T → IN1 and OUT2 → T → IN2 at 50 Ω, the jumper DIO7_N → DIO1_P, here P8 → CHOP): **22 of 22 passed both times** (receipts `data/board-0.1-test_hw-{1,2}.json`).
  - The timer: 1000 SHOT records through the ring exactly 3.2 µs apart, a shot limit; points per shot and 100 sweeps; with busy 1.5 periods every other tick missed (21 shots, 20 missed) and a sweep limit.
  - The analog trigger on IN1 from the test source's pulse on OUT1: 200 shots, each 12 ns after the pulse's start (the shape crosses the 0.2 V threshold at 4 ns; the events are on the 8 ns grid and the converters' loop is rounded to whole clocks).
  - A snapshot of a test pulse (rise 20 ns, decay 300 ns) fired by P1's pulse in master mode: the peak at the expected sample (67), 0.411 V of 0.400 V (the fast 6 ns pulse read 0.455 V: the analog path's high-frequency peaking).
- **Board, `make test-hw` twice more with the jumpers DIO7_N → DIO0_P (P8 → TRIG) and DIO6_N → DIO1_P (P7 → CHOP)** (the user rewired): **25 of 25 passed both times** (receipts `data/board-0.1-test_hw-jumpers-{1,2}.json`).
  - Both jumpers follow high, low and released, and see a 125 MHz pattern's edges.
  - **TTL shots from the test source's sync on P8:** 200 shots per run, each exactly 21 ns after its laser time (the loop from an output word through the pin, the jumper and TRIG's 1 ns samplers); with the laser moved by 0, 2, 4, 6 and 8 ns the shot times moved by exactly the same (21 ns in every case).
  - **Categories from CHOP**, the test source's chopper level on P7 with a 16 ns guard band: 200 shots per run, every category equal to the test pattern (0110100…), none invalid.
  - The first run's analog and snapshot checks failed on the test's own errors: a configuration merged onto the previous run's (a leftover sweep limit stopped the analog run at 21 shots) and an amplitude tolerance too tight for a 6 ns pulse.

2026-11-08: 0.0b, the averager and the pump's tap in common (plan step 0.0b)

- **`rp_avg`** (`../common/rtl/optional/rp_avg.v`), spin's averager parameterized (spin 1.1 uses it with its old parameters: spin's VERIFICATION). boxcar's `tests/tb_avg.sv` tests the new logic with boxcar's parameters (AW 11, IN_W 24, SUM_W 36, BANK_SCANS, AUTO_READ): scans per bank in the BLOCK fields and the frozen half, N scans over both banks, swaps gated by `swap_ok`, the auto-increment read across the bank boundary (and after a partial read), 36-bit saturation of I and Q after 4,097 full-scale adds (exactly −2^35 after 4,096 minimum adds, unsaturated), a released half cleared. **12 of 12 mutants** of the new logic fail it (`.build/mut/mutate.py`; the first run left one alive, the word index's reset at a new address, which every read had hidden by ending on a point boundary: a partial read now precedes one).
- **The ring pump's tap** (`../common/python/rpfpga/ringpump.c`, `ringpump.py`, `ringworker.py`): `common/tests/test_ringpump.py` checks on a file that the tap gets exactly the whole records of its types (records of 3, 7 and 6 words cut anywhere by frames of 16 words, the writer's padding skipped, across the end of the area, a half record dropped at a restart) while the TCP clients still get every word; common's 385 tests, counter's 66 and timetagger's 64 pass.
- `make sim`: tb_core and tb_avg passed. No new boxcar build (the core is unchanged; `rp_avg.v` is only listed for the bench).

2026-11-08: 0.0, the skeleton (plan step 0.0)

- **`make test`:** boxcar's 9 tests passed (the driver against a register model, the protocol, the client over localhost, loading through the fake PYNQ); `common/tests/test_designs.py` and `test_site.py` passed with boxcar in the repository.
- **`make sim`:** `tb_core` passed: the register window (no alias), the 1 ns input words (a 2 ns pulse between samples seen on TRIG and AUX), the outputs and markers (patterns, high impedance), the ring writer's test words through an HP0 responder, and the HP1 writer through a second responder with random AWREADY/WREADY stalls: 4,096 words of a pattern (every beat and burst address checked) while the ring ran, a base below 1 MiB refused, an abort that leaves no write unanswered.
- **Build 0x4d4aaaf2:** routed WNS +0.292 ns, TNS 0, WHS +0.029 ns; 858 slices (19.5 %), 1,714 LUTs, 8 BRAM tiles, 0 DSP48, one MMCM (the common clock wizard) and one PLL (the 1 ns samplers); DRC only RTSTAT-10. `verify_build.py`: 8 of 8 checks passed; a fresh copy from the recipe rebuilt 0x4d4aaaf2.
- **Board, `make test-hw` twice on 0x4d4aaaf2** (the bench as it was: OUT1 → T → IN1, OUT2 → T → IN2, a jumper DIO7_N → DIO1_P, here P8 → CHOP): **16 of 16 passed both times** (receipts `data/board-0.0-test_hw-{1,2}.json`).

- The jumper was found and checked: CHOP follows P8 high and low, reads low when P8 is released, and sees the edges of a 125 MHz pattern but none of a level.
- **The two DDR masters at once:** the ring through S_AXI_HP0 carried 2.83 M test words at 1 Mword/s in order, none dropped, no watchdog restart, while the S_AXI_HP1 writer filled a second allocation with a counting pattern at one word per clock (500 MB/s): 4 MiB in 11.6–11.8 ms and 32 MiB in 71.7–73.4 ms, every word correct, no error, the FIFO never above 16 beats (of 2,048). Both allocations (the 4 MiB ring and the 32 MiB buffer) lived in the CMA at once.
- The service ran spin; the test parked it on analog_echo and loaded spin again afterwards.