counter 1.0: reference documentation

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

Rok Zitko, Jozef Stefan Institute

2026-11-10 (commit 6178370)

Contents

counter: a gated photon counter **2**
- Connecting to lab equipment: read first . . . 2
- Quick start . . . 2
- Presets . . . 3
- Specifications (design values unless marked measured; the board tests in docs/VERIFICATION.md) . . . 3
- Hardware setup . . . 4
- Files . . . 5

counter recipes **5**
- Without wiring: the test emitter . . . 6
- NV centre: a pulsed Rabi measurement with awg . . . 6
- NV centre: the readout trace . . . 7
- NV centre: pulsed ODMR . . . 7
- Ions and atoms: state detection and the threshold . . . 7
- Preselection through awg's EVENT input . . . 7
- Hanbury Brown and Twiss (antibunching) . . . 8
- Heralded single photons (Grangier) . . . 8
- Bell's inequality (CHSH) . . . 8
- Alignment by ear and eye: the rate on OUT1 . . . 8
- NIM pulses on the analog inputs . . . 9

counter: theory of operation **9**
- 1. Physics for lab users . . . 9
- 2. Definitions (the contract) . . . 10
- 3. Version 1.0 (the plan's steps 1 to 5) . . . 14

counter commands (TCP) **15**
- Configuration (`CONFIG {json}` patches, `CONFIG REPLACE {json}`, `CONFIG?`) . . . 16
- RECORDS STREAM . . . 17

counter registers **18**
- General . . . 18
- Channels (C1..C10; C9, C10 are the analog discriminators of step 4 and idle in 0.1) . . . 18
- RATE records and the alarm . . . 19
- The ring writer (owned by the service's worker, `rpfpga.ringworker`) . . . 19
- The test emitter . . . 19
- Monitors (OUT1, OUT2) and outputs (O1..O4) . . . 20
- Control inputs, shots and gates (plan step 2) . . . 20
- Decisions (plan step 3) . . . 21
- Analog channels and coincidences (plan step 4) . . . 21
- Configuration registers . . . 22

Verification record **22**
- 2026-11-06: NEXT, GATE, one word, and awg's OUT1 as a detector . . . 22
- 2026-11-05 (evening): tests on awg's edges with exact expectations . . . 23
- 2026-11-05: 1.0 with awg on a second board and an oscilloscope . . . 24
- 2026-11-04: 1.0, the release (plan step 5) . . . 25
- 2026-11-04: 0.4, analog channels and coincidences (plan step 4) . . . 25

2026-11-04: 0.3, decisions (plan step 3) . . . 27
2026-11-04: 0.2, shots (plan step 2) . . . 28
2026-11-04: 0.1, counting (plan step 1) . . . 29
2026-11-03: 0.1, the skeleton (plan step 0) . . . 30

counter: a gated photon counter

counter turns the Red Pitaya into a **gated photon counter**: it counts detector pulses in time windows set by a pulse generator, sums them per point of a sweep, decides on the board (a count against a threshold, on a pin about 105 ns after the gate), counts coincidences among four channels, and streams per-shot records to a PC. It is the detection half of pulsed experiments: NV-centre pulsed readout, ion and atom state detection, and teaching-lab coincidence counting. The pulse sequences come from elsewhere (awg on a second board, a PulseBlaster, a Pulse Streamer, ARTIQ).

Status: version 1.0. It counts eight inputs sampled every 1 ns, each with a non-paralyzable dead time and a delay, into 48-bit totals, and streams RATE records with exact intervals to a PC; it takes **shots**: a TRIG (from DIO0_N, its own timer or software) opens up to four gates (a delay, a width, a delay step per point for a time trace), the point index follows the sequence (per shot, per NEXT on DIO2_N, fixed; SEQ on DIO3_N starts a sweep, which is checked), and the counts are summed per point into 64-bit totals on the board, with optional per-shot records. A rate alarm, rate monitors on OUT1/OUT2 and a built-in test emitter (pseudo-random detector pulses, also in pairs or antibunched, bright or dark per shot from a table of the points, with a decay) complete it. Two **decisions** per shot (a count window on each of two gates, any logic of the two: ”bright”, ”both clicked”) drive output pins (a level held until the next shot, pulses, or early as soon as a count reaches its threshold) and go into the records and the per-point sums; an output can also echo a channel’s counted pulses. **IN1 and IN2** are two more count channels (C9, C10): a discriminator at a threshold in volts with a hysteresis, for positive or negative pulses (NIM), the crossing interpolated to 1 ns, aligned with the pins through the board calibration. Two **coincidence units** of up to four channels each count timetagger’s clusters (a window of 8 … 1023 ns) by their 15 exclusive patterns in 48-bit totals, optionally inside gate G1; the second can delay one channel by up to 16 µs to count the accidental coincidences directly; the patterns come in the RATE records, and a unit’s clusters can herald on an output pin. Presets and RECIPES.md set it up for NV pulsed readout (Rabi, readout traces, ODMR) with awg on a second board, ion and atom state detection with its threshold and fidelity, preselection, Hanbury Brown and Twiss, heralded single photons and Bell’s inequality; `analysis.py` has the contrasts, fits and coincidence quantities. docs/THEORY.md explains the physics (section 1) and defines every number (section 2). Everything is simulated against a bit-exact model and tested on the board (docs/VERIFICATION.md), including a session with a second board running awg and an oscilloscope (exact counts from awg’s patterns, the decision latency, the preselection loop through awg, the analog channels’ alignment) and tests on awg’s random patterns with exact expectations (sums, records and clusters, the 1 ns boundaries, accidentals, dead time with a known rate, repeat until success); a real detector was not part of any test.

Connecting to lab equipment: read first

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

- **Extension connector E1 is 3.3 V LVCMOS and not 5 V tolerant.** counter reads DIO0_P–DIO7_P (count inputs C1–C8) and DIO0_N–DIO3_N (TRIG, GATE, NEXT, SEQ: from a pulse generator such as awg on a second board), all with pull-downs. A detector module that drives 50 Ω (most give 5 V TTL into a high-impedance input) needs a 50 Ω terminator at the board, which brings it to about 2.5 V; never connect 5 V directly. NIM signals need a NIM-to-TTL converter, or go to the analog channels on IN1/IN2 (C9, C10: negative pulses below a threshold, e.g. −0.4 V, with 50 Ω termination; LV jumpers ±1 V).
- **Outputs DIO4_N–DIO7_N** (O1–O4) are high impedance after loading and after SAFE, and are driven only when enabled. Never drive an output pin from outside.
- **OUT1/OUT2** are the rate monitors: 0 V (through the board calibration) unless a monitor is on, then within about ±1 V. Terminate them as the root README says.

Quick start

With counter loaded on a board (no Vivado needed):

```
import sys, time; sys.path[:0] = ["common/python", "counter/control"]
from counter import CounterClient, RecordStream, model
from counter.analysis import parse_frame, rate_table          # numpy
```

```python
with CounterClient("192.168.0.102") as dev:
    dev.load_preset("selftest")       # the emitter on C1..C4, no wiring
    print(dev.rates())                # counts/s of C1..C8 over 0.1 s
    print(dev.counters())             # 48-bit totals, rejected edges, the emitter's events
    dev.apply({"channels": {"1": {"dead_time": 50e-9, "delay": 12e-9}}})
    with RecordStream("192.168.0.102") as stream:           # RATE records, exact intervals
        words = []
        for frame in stream.frames(timeout=1.0):
            words += list(parse_frame(frame)[1])
            if len(words) > 200:
                break
    print(rate_table(model.decode(words))["rates"])
    # shots: the internal timer, 16 points, G1 counting C1 for 2 us after each TRIG
    dev.load_preset("shots-selftest")
    dev.load_emitter_table([i / 15 for i in range(16)])      # point i bright with probability i/15
    dev.run()
    time.sleep(2)
    dev.stop()
    print(dev.results(now=True)["mean"]["n1"])           # counts per shot at each point
    # decisions: D1 "bright" (G1 >= 3) on O1 (DIO4_N) as a level until the next shot
    dev.apply({"decisions": {"1": {"gate_a": 1, "low_a": 3, "logic": "a"}}, "outputs": {"1":
    ↪ {"mode": "d1"}}})
    print(dev.decisions())                                 # taken, true, the last value
    # with a jumper from O1 to C8: dev.measure_latency(output=1, channel=8)
    # coincidences of the emitter's pairs: C2 copies half of C1's events 16 ns later
    dev.apply({"channels": {"1": {"enable": True}, "2": {"enable": True}},
               "emitter": {"lanes": {"1": {"rate": 1e6, "channel": 1}, "2": {"channel": 2}},
                           "pairs": {"1-2": {"mode": "pairs", "probability": 0.5, "delay": 16e-9}}},
               "coincidences": {"1": {"enable": True, "channels": [1, 2], "window": 20e-9}}})
    print(dev.coincidences()["1"]["exclusive"])           # {"C1": n, "C2": n, "C1+C2": n}
    # IN1 as C9: NIM pulses below -0.4 V
    dev.apply({"channels": {"9": {"enable": True, "polarity": "negative", "threshold": -0.4,
                                  "hysteresis": 0.1}}})
```

Presets

Preset	What it sets up
`off`	everything off, outputs in high impedance (the power-up state)
`rates`	C1..C8 counting, RATE records every 0.1 s, C1's rate on OUT1 at 1 V per Mcount/s
`selftest`	the emitter on C1..C4 without wiring: 1 Mcount/s on C1, half of them copied 16 ns later to C2, a split (antibunched) source of 2 Mcount/s on C3 and C4
`shots-selftest`	shots from the timer every 10 µs over 16 points, the emitter bright or dark per shot on C1 (`load_emitter_table`), G1 counting C1 for 2 µs
`nv-readout`	TRIG at each laser readout, SEQ at each sweep, one shot per point (100 points); G1 the signal (0 … 300 ns), G2 the reference (2 … 3 µs) on C1
`nv-trace`	the readout's time trace: G1 a 20 ns window stepped by 20 ns over 150 points
`ion-detection`	G1 counts C1 for 500 µs after TRIG; D1 = G1 ≥ 3 (bright) held on O1; SHOT records for the histogram
`preselection`	a 100 µs gate; D1 = G1 ≥ 2 as a 200 ns pulse on O1 for awg's EVENT input; D2 = G1 = 0 on O2
`hbt`	C1, C2: unit 1 their clusters within 8 ns, unit 2 with C2 1 µs later (the accidentals); RATE records every second
`heralded-g2`	the herald on C1, the beam splitter's outputs on C2, C3: both units (unit 2 delayed)
`bell`	Alice on C1 (+), C2 (−), Bob on C3 (+), C4 (−): the clusters of the four, unit 2 with C3 delayed

The experiments' wiring, sequences and analysis: RECIPES.md.

Specifications (design values unless marked measured; the board tests in docs/VERIFICATION.md)

Quantity	Value
Count inputs	C1–C8: 8 pins of E1 sampled every 1 ns (ISERDESE2 oversampling); pulses ≥ 2 ns always seen; rising or falling edges
Dead time	non-paralyzable, 8 ns … 65.535 µs in 1 ns steps, per channel (at most 125 Mcount/s); rejected edges counted
Delay	0 … 255 ns in 1 ns steps, per channel
Totals	48 bits per channel, never cleared (26 days at 125 Mcount/s)
RATE records	every 10 µs … 10 s; consecutive records differ by exactly one interval in every channel
Rate alarm	a set of channels below a rate in a window of 1 µs … 2 s: STATUS, records, an output pin
Rate monitors	OUT1/OUT2: offset + scale × rate in volts, windows of 128 ns … 134 ms
Test emitter	4 lanes, up to 125 Mevents/s each (one trial per 8 ns, a uniform 1 ns slot); pairs (with a delay of 0 … 248 ns) or split (antibunched)
Outputs	O1–O4: 4 pins of E1, 2 ns bits (OSERDESE2): a pattern, an emitter lane, the alarm, a gate's window, the TRIGs, D1/D2, a channel's echo (also C9/C10), a coincidence unit's herald; high impedance until enabled; measured: a channel's echo 92.5 ns after its edge, a herald 121.5 ns + W after its cluster's first event
OUT1/OUT2	the rate monitors, or pulses (8 ns at an emitter lane's events, an amplitude in volts: the analog channels' test signal)
Records	a ring in DDR through S_AXI_HP0, served to a PC by the board's ring pump (`RECORDS STREAM`)
Control inputs	TRIG, NEXT, SEQ (DIO0_N, DIO2_N, DIO3_N): 1 ns, rising or falling, hold-off 8 ns … 65.5 µs; GATE (DIO1_N): a level, minimum width 1 … 8 ns, delay 0 … 255 ns
Shots	TRIG from the pin, an internal timer (64 ns … 34 s, with SEQs every P TRIGs) or software; a shot busy for its gates (at least 64 ns), TRIGs meanwhile counted as missed; runs started at once or at a SEQ, stopped by software, a number of shots or a duration
Points and sweeps	up to 1024 points: per shot, per NEXT or fixed; sweeps from SEQ to SEQ, checked (TRIGs = points) in per-shot mode
Gates	4 per shot: delay and width 0 … 4.29 s, delay step per point 0 … 16.7 ms (all 1 ns), a channel set each, optionally only where GATE is high; 32-bit counts
Per-point sums	per point: the 4 gates' counts, the shots, D1 true, D2 true, Σ $n1^2$ (two banks on the board, collected by the service every second into 64-bit totals, none lost)
Analog channels	C9, C10 on IN1, IN2: a Schmitt trigger at a threshold in volts (board calibration, input range) with a hysteresis, positive or negative pulses, the crossing between two samples interpolated to 1 ns, then the same dead time and delay as C1–C8; aligned with the pins (Δ = 95 ns on this board's calibration, added to the pin channels' and gates' delays while an analog channel is on; measured with OUT1's pulses on IN1 against an output pin wired to C8 and an oscilloscope: within 0.5 ns, ±1 ns, for these pulses; other pulse shapes cross the threshold differently). Measured: a 10 MHz sine counted once per period; 8 ns pulses of 0.8 V from OUT1 counted above 0.25 V, two within about 16 ns merging (the converters' bandwidth)
Coincidences	2 units of up to 4 channels (any of C1–C10): timetagger's clusters, window 8 … 1023 ns (1 ns), the 15 exclusive patterns' 48-bit totals, optionally inside G1; unit 2 delays one channel by 0 … 16.4 µs (accidentals); the patterns in the RATE records (exact intervals); a herald output
Decisions	D1, D2 per shot: `low ≤ n ≤ high` on each of two gates (the counts saturated to 16 bits, as in the SHOT records), any 2-input logic; the pin a level (until the next shot), pulses on true or false (8 ns … 2 µs), or early (a gate's running count at its threshold); in SHOT records, the per-point sums and counters. Latency from the end of a gate (an edge in its last ns) to the output pin: 101 … 109 ns, 105 ns on average (where TRIG falls in the 8 ns clock; measured on the board with a jumper, `measure_latency`, and an oscilloscope: 103.5 ns); early mode: the same after the deciding edge (the plan's target was 75–100 ns)
With awg	exact counts: 39 million shots at 2 million per second from awg's patterns (TRIG, SEQ, 1–8 pulses per point on C1, C2), every per-point sum exact, every sweep with a missing TRIG flagged; the preselection loop (a decision on O1 into awg's EVENT0, awg's sequence jumping to a step that pulses a digital output): 468 ns from the gate's end to awg's output, 356 ns with awg's digital outputs at their shortest delay (awg's part 363 ns or 251 ns)
Shot records	per shot (optional): point, flags, the TRIG's time in ns, the sweep, 4 counts (16 bits); 2,000,000 records/s (40 MB/s) without loss to a client on the board; 1,500,000/s (30 MB/s) to a PC over Gigabit Ethernet (measured; at 2,000,000/s 2.4 % dropped)

Hardware setup

Pin (E1)	Use
DIO0_P … DIO7_P	count inputs C1 … C8 (the same pins as timetagger's channels 1–8)
DIO0_N	TRIG (shot start)
DIO1_N	GATE (external gate level)
DIO2_N	NEXT (next point)
DIO3_N	SEQ (sequence start)
DIO4_N … DIO7_N	outputs O1 … O4
IN1, IN2	analog count inputs C9, C10 (LV jumpers ±1 V; 50 Ω termination for NIM)
OUT1, OUT2	rate monitors or test pulses (0 V through the board calibration otherwise)

With awg on a second board (RECIPES.md): awg's digital outputs D0–D7 (its DIO0_N–DIO7_N) drive the laser, the microwave switch and counter's TRIG (DIO0_N), SEQ (DIO3_N), NEXT (DIO2_N) or GATE (DIO1_N); counter's O1 (DIO4_N) can drive awg's EVENT0 (its DIO4_P). Join the two boards' grounds. Header jumpers next to each other couple: with 20 cm wires, 8 ns pulses on the wire beside TRIG's made about 2 spurious TRIGs per million (each shifts the rest of its sweep's points and flags the sweep). Keep TRIG's wire apart from pulse wires or use coax, and set TRIG's hold-off up to just below the shortest shot spacing (450 ns removed them in that test).

Files

Path	Contents
`src/`	the core (`ct_core`), channels (`ct_chan`), delays (`ct_delay`), totals, RATE records and the alarm (`ct_rates`), the emitter (`ct_emit`), the monitors (`ct_mon`), the control inputs (`ct_ctl`), the shots (`ct_shots`), the gates (`ct_gate`), the decisions (`ct_decide`), the analog discriminators (`ct_analog`), the coincidence units (`ct_coinc`), the per-point sums (`ct_sums`), the shots' records (`ct_shotrec`), the records' merge (`ct_merge`, `ct_fifo`)
`control/counter/`	driver, client, protocol, the bit-exact model, the ring worker and stream hub, analysis (contrasts, fits, thresholds and fidelities, g2, alpha, CHSH), the notebook panel (the plugin)
`RECIPES.md`	the experiments: NV Rabi, readout trace and ODMR with awg, state detection, preselection, HBT, heralded g2, CHSH, alignment, NIM pulses
`control/counter.ipynb`	the notebook
`docs/`	THEORY (definitions), REGISTERS, PROTOCOL, VERIFICATION; `docs/data/` the board tests' receipts
`docs/manual/`	the user's manual (`user_manual.pdf`, LaTeX): the instrument, the theory (counting statistics, detectors, readout, coincidences, the FPGA) and counting in quantum labs (NV centres, ions and atoms, photons), with measured and computed figures
`tools/`	host-only: the manual's board recordings (`record_manual_data.py`) and figures (`manual_figures.py`, `app_figures.py`)
`scripts/test_hw.py`	the board test (`make test-hw`): the emitter, dead time, RATE records, alarm, monitors, shots, gates, per-point sums, decisions and their records, the analog channels (the emitter, the idle inputs' noise), the coincidences against a simulation and the RATE records with patterns without wiring; outputs looped back to inputs if wired (patterns, an emitter lane, a decision's pulses and its latency, a channel's echo, a herald), OUT1 → IN1 if cabled
`tests/`	model, driver/protocol/client/stream and analysis tests; the XSim testbenches and their vectors (`vectors.py`)

counter recipes

Worked examples in Python on a PC (`dev = CounterClient("192.168.0.102")`) or on the board (`dev = DRIVER()`, after `import rpfpga`): the methods are the same. Times are in seconds at the connectors, volts at the connectors (through the board calibration). The physics behind each one is in docs/THEORY.md section 1; the presets in README.md.

```
import sys, time; sys.path[:0] = ["common/python", "counter/control", "awg/control"]
import numpy as np
from counter import CounterClient, RecordStream, analysis, model
dev = CounterClient("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 detector

module that drives 50 Ω (5 V TTL) needs a 50 Ω terminator at the board (about 2.5 V then); NIM goes to IN1/IN2 (C9, C10) or through a NIM-to-TTL converter. Join the grounds of every instrument (and of a second Red Pitaya).

Without wiring: the test emitter

```
dev.load_preset("selftest")                    # C1: 1 Mcount/s, C2: half of them 16 ns later, C3/C4:
↪ split
print(dev.rates(1.0)["channels"])
dev.load_preset("shots-selftest")              # shots every 10 us over 16 points, C1 bright or dark
↪ per point
dev.load_emitter_table([i / 15 for i in range(16)])
dev.run(); time.sleep(2); dev.stop()
print(dev.results(now=True)["mean"]["n1"])     # rises from 0.5 to 5.4 counts per shot
```

The emitter imitates detectors on the channels (2 ns pulses through the same edge and dead-time logic); its counts are binomial per 8 ns word, not Poisson: the variance is n p (1 − p) per word.

NV centre: a pulsed Rabi measurement with awg

A second Red Pitaya running awg makes the pulse pattern on its digital outputs (D0–D7 on its DIO0_N–DIO7_N, one 8-bit word every 8 ns, a pattern of up to 32768 words repeated):

awg board	to	
D0 (DIO0_N)	the AOM driver's TTL input	the laser
D1 (DIO1_N)	the microwave switch	the MW pulse
D2 (DIO2_N)	counter's DIO0_N (TRIG)	the readout's start
D3 (DIO3_N)	counter's DIO3_N (SEQ)	the sweep's start
the APD's output (terminated)	counter's DIO0_P (C1)	the photons

```
from awg import AwgClient
awg = AwgClient("192.168.0.150")                          # the second board
LASER, MW, TRIG, SEQ = 0x01, 0x02, 0x04, 0x08
taus = [20e-9 * k for k in range(25)]                     # MW pulses of 0 .. 480 ns
words = []
def hold(word, seconds):
    words.extend([word] * max(1, round(seconds / 8e-9)))
for k, tau in enumerate(taus):
    if tau:
        hold(MW, tau)                                     # the MW pulse
    hold(0, 100e-9)
    hold(LASER | TRIG | (SEQ if k == 0 else 0), 16e-9)  # the readout: TRIG at the laser's start
    hold(LASER, 3e-6 - 16e-9)                             # 3 us: readout, then polarization
    hold(0, 1e-6)                                         # the metastable state empties
awg.save_pattern("rabi", words)                           # (14,375 words: 115 us per sweep)
awg.set_digital(run="continuous", program="rabi", enable=0x0F,
                programs={"rabi": {"mode": "memory", "pattern": "rabi", "rate": 125e6}})

dev.load_preset("nv-readout")                             # G1 0 .. 300 ns: signal, G2 2 .. 3 us:
↪ reference
dev.apply({"shots": {"points": len(taus)}})
dev.run(); time.sleep(60); dev.stop()
r = dev.results(now=True)                                 # per point: 64-bit sums of n1, n2, shots
c = analysis.contrast(r["n1"], r["n2"])                   # 1 - S/R with Poisson errors
fit = analysis.fit_rabi(taus, c["ratio"], sigma=c["error"])
print(fit["pi_time"], fit["frequency"], r["shots"][0])
```

Each shot's point comes from its position in the sweep (per shot, SEQ resets it); the `SWEEP` records and `run_status()["bad_sweeps"]` flag a sweep whose TRIGs were not 25 (a lost or extra TRIG would shift the points). The service collects the per-point sums every second, so a run can last hours without the PC. The reference window normalizes the laser power's drifts.

NV centre: the readout trace

The fluorescence's time trace during the readout pulse, without histograms: a 20 ns window stepped by 20 ns per point (the gate-delay scan), one point per shot.

```
dev.load_preset("nv-trace")                              # G1: 20 ns at 0, 20, 40 .. 2980 ns after TRIG
dev.run(); time.sleep(30); dev.stop()
r = dev.results(now=True)
t = np.arange(150) * 20e-9
fit = analysis.fit_exponential(t, r["mean"]["n1"])       # the polarization's decay
```

With the Rabi pattern above (no MW pulse: `taus = [0.0]`), the trace shows the bright start of the readout and its decay to the polarized level; choose the signal window G1 where the contrast is largest (typically the first 200–300 ns).

NV centre: pulsed ODMR

Fix the MW pulse at π (from the Rabi fit) and step the synthesizer's frequency once per point: its list mode advanced by a trigger from awg's D4 (DIO4_N), one pulse per point in the pattern (`hold(0x10 | ..., 16e-9)` at each point's start). counter stays in `nv-readout` (per-shot points, SEQ at the list's start).

```
freqs = np.linspace(2.85e9, 2.89e9, 81)                  # the synthesizer's list
r = dev.results(now=True)
c = analysis.contrast(r["n1"], r["n2"])
dip = analysis.fit_lorentzian(freqs, c["ratio"], sigma=c["error"])
print(dip["f0"], dip["fwhm"], dip["contrast"])
```

Ions and atoms: state detection and the threshold

```
dev.load_preset("ion-detection")      # G1 500 us after each TRIG; D1 = G1 >= 3 on O1; SHOT records
words = []
with RecordStream("192.168.0.102") as stream:
    for frame in stream.frames(timeout=1.0):
        words += list(analysis.parse_frame(frame)[1])
        if len(words) > 5 * 20000:
            break
h = analysis.count_histogram(model.decode(words), gate=1)
mix = analysis.two_poisson(h)                            # dark and bright means, the bright fraction
th = analysis.threshold(mix["dark"], mix["bright"])
dev.apply({"decisions": {"1": {"low_a": th["threshold"]}}})
print(th["threshold"], th["fidelity"])
```

Prepare the states alternately (or use a mixed state) so that the histogram has both Poisson components. The decision D1 then marks every shot as bright or dark on O1 (DIO4_N), in the SHOT records and in the per-point sums (`results()["d1"]`: the bright probability per point is d1 / shots). The SHOT records' counts are saturated to 16 bits like the decisions', so the PC's thresholds and the board's agree exactly.

Preselection through awg's EVENT input

The `preselection` preset pulses O1 (DIO4_N) for 200 ns when G1 counts at least 2: wire it to awg's DIO4_P (EVENT0). awg's sequence waits in a step that ends on the event (see awg's RECIPES, "A triggered sequence with a jump"): a detection step whose marker or digital pattern gives the laser pulse and counter's TRIG, then a waiting step that jumps to the experiment on EVENT0 and otherwise repeats the detection. Start awg's waiting step before counter's gate ends: awg drops an event that comes before the waiting step's first point. The loop time, measured (2026-11-05, 20 cm jumpers): counter's decision 101 … 109 ns after the gate's last nanosecond (105 ns on average), awg's EVENT0 to its digital output 363 ns (its outputs aligned with its analog outputs) or 251 ns (`digital.delay` −112 ns, the shortest): 468 ns or 356 ns from the gate's end to awg's output. `dev.measure_latency()` measures counter's part with a jumper from an output to a quiet channel (O4 to C8 are neighbouring pins).

```
# (LASER, MW, TRIG as in the Rabi recipe) one step starts the detection pattern and waits for
# EVENT0 (102 us, longer than the 100 us gate plus the decision); without an event it detects again
awg.save_segment("z64", [0.0] * 64)                          # 512 ns of silence at 125 MSa/s
awg.save_pattern("detect", [LASER | TRIG] * 2 + [LASER] * 12498)   # 100 us of light, TRIG at its
 ↪ start
awg.save_pattern("go", [MW] * 25)                            # the experiment's first pulse (200 ns)
```

```
awg.save_sequence("presel", [
    {"segment": "z64", "repeat": 200, "digital": "detect", "on_event": "now", "event_target": 1,
 ↪  "next": 0},
    {"segment": "z64", "repeat": 4, "digital": "go", "next": 0}])
prog = lambda p: {"mode": "memory", "pattern": p, "rate": 125e6, "repeats": 1, "end": "idle"}
awg.apply({"arb": {"sequence": "presel", "rate": 125e6, "run": "continuous",
                   "jump": {"source": "event0", "edge": "rising"}},
           "outputs": {"1": {"arb": {"on": True, "amplitude": 0.0}}},
           "digital": {"run": "sequencer", "enable": 0x07, "delay": -112e-9,
                       "programs": {"detect": prog("detect"), "go": prog("go")}}})
dev.load_preset("preselection")
dev.run()
```

Hanbury Brown and Twiss (antibunching)

Two detectors behind a 50:50 beam splitter on C1 and C2.

```
dev.load_preset("hbt")                # unit 1: C1, C2 within 8 ns; unit 2: the same, C2 1 us later
a = dev.coincidences(); time.sleep(60); b = dev.coincidences()
n12 = b["1"]["inclusive"]["C1+C2"] - a["1"]["inclusive"]["C1+C2"]
acc = b["2"]["inclusive"]["C1+C2"] - a["2"]["inclusive"]["C1+C2"]
print(analysis.g2_zero(n12, accidentals=acc))           # g2(0) < 0.5: a single emitter
```

The delayed unit counts the accidental coincidences of the same rates directly, so no rate model is needed (`g2_zero(n12, rates=..., window=8e-9, duration=60)` computes them instead). The window W must cover the detectors' jitter and the two channels' relative delay (set the channels' `delay` so that the coincidences' peak sits inside it); counter's 1 ns times cannot resolve g2(τ) itself: use timetagger for that.

Heralded single photons (Grangier)

The herald (the idler of a pair source) on C1, the signal's beam splitter outputs on C2, C3.

```
dev.load_preset("heralded-g2")
a = dev.coincidences(); time.sleep(300); b = dev.coincidences()
inc = {k: b["1"]["inclusive"][k] - a["1"]["inclusive"][k] for k in b["1"]["inclusive"]}
print(analysis.heralded_alpha(inc["C1"], inc["C1+C2"], inc["C1+C3"], inc["C1+C2+C3"]))
```

alpha < 1 shows a heralded single photon (alpha = 1 for coherent light).

Bell's inequality (CHSH)

Alice's polarizing beam splitter outputs on C1 (+) and C2 (−), Bob's on C3 (+) and C4 (−); four settings of the analysers (a, b), (a, b'), (a', b), (a', b').

```
dev.load_preset("bell")
E = []
for setting in ("a b", "a b'", "a' b", "a' b'"):
    input("set the analysers to %s, then press Enter" % setting)
    x = dev.coincidences(); time.sleep(60); y = dev.coincidences()
    d = {k: y["1"]["inclusive"][k] - x["1"]["inclusive"][k] for k in ("C1+C3", "C2+C4", "C1+C4",
 ↪  "C2+C3")}
    E.append(analysis.correlation(d["C1+C3"], d["C2+C4"], d["C1+C4"], d["C2+C3"]))
print(analysis.chsh(*E))              # |S| > 2: a violation (quantum mechanics: up to 2.83)
```

Unit 2 (C3 delayed by 1 µs) counts the accidentals of the same pairs: subtract them before `correlation` when the rates are high.

Alignment by ear and eye: the rate on OUT1

```
dev.load_preset("rates")              # C1..C8 counting; C1's rate on OUT1 at 1 V per Mcount/s
dev.apply({"monitors": {"1": {"scale": 1e-5, "window": 0.01}}})      # 0.1 V per 10 kcount/s, 10 ms
```

OUT1 follows C1's rate in volts (on a meter, an oscilloscope, or an audio amplifier for a pitch); a rate alarm (`alarm`) can light an output pin when an ion is lost.

NIM pulses on the analog inputs

IN1 and IN2 (LV jumpers, ±1 V, 50 Ω termination at the board for NIM) are C9 and C10:

```
print(dev.analog(1024))              # the inputs' level and noise in volts
dev.apply({"channels": {"9": {"enable": True, "polarity": "negative", "threshold": -0.4,
                              "hysteresis": 0.1, "dead_time": 20e-9}}})
```

The crossing is interpolated to 1 ns between the converters' samples. The converters' bandwidth limits the pulse-pair resolution: two 8 ns pulses less than about 16 ns apart are counted once (measured with OUT1's test pulses into IN1). While C9 or C10 is on, every pin channel, gate and GATE gets the analog lag (95 ns on this board's calibration, measured) added, so all times stay referred to the connectors (`describe()["alignment"]`).

counter: theory of operation

counter is a gated photon counter. It counts detector pulses in time windows (gates) that a pulse generator sets (awg on a second board, a PulseBlaster, a Pulse Streamer, ARTIQ), sums them per point of a sweep, decides on the board (a count against a threshold, on a pin within about 100 ns), counts coincidences among four channels, and streams per-shot records to a PC. It is the detection half of pulsed experiments: NV-centre readout, ion and atom state detection, and teaching-lab coincidence counting.

Status: 1.0 (2026-11-04), the approved plan's release. This version counts (channels with their dead time and delay, 48-bit totals, RATE records, a rate alarm, the test emitter, the rate monitors, the outputs: step 1), takes shots (TRIG, NEXT, SEQ and GATE inputs, runs, points and sweeps, four gates with a delay scan, the per-point sums, SHOT, SWEEP, SWAP and RUN records, the emitter's shot states: step 2), decides (D1, D2 on output pins, in the records and the sums; a channel's echo on an output: step 3), discriminates IN1 and IN2 (C9, C10, aligned with the pins) and counts coincidence clusters in two units (one with a delayed channel; the patterns in RATE records, a herald on an output: step 4), with presets, analysis and recipes for NV readout, state detection and coincidences (1.0; VERIFICATION.md for what was tested). The definitions below are the **contract** for the bit-exact model (`control/counter/model.py`) and the RTL: they fix what every number means before it is built. Section 3 describes what 1.0 actually contains.

1. Physics for lab users

1.1 Counting detectors: the dead time

A single-photon detector (an APD, a PMT, an SNSPD) gives a pulse per detected photon and is then blind for its own dead time. counter adds a **non-paralyzable** dead time τ per channel (8 ns ... 65 µs): after a counted edge, edges within τ are rejected (and counted as rejected), and the next edge after τ counts. A true rate R then reads as

$R_{read} = R / (1 + R\,\tau)$, so $R = R_{read} / (1 - R_{read}\,\tau)$

(`analysis.dead_time_corrected`). Set τ at least to the detector's own dead time (and above its afterpulsing), so that the correction uses one known τ: e.g. 22 ns for an SPCM, 50 ns for a PMT with ringing. At 1 Mcount/s with τ = 22 ns the correction is 2.2 %.

1.2 Counts in a gate: Poisson and binomial

Photons of a steady source arrive at random: the counts n in a gate of fixed length follow Poisson statistics, mean µ, variance µ, relative error $1/\sqrt{\mu}$. Per-point sums over N shots have the variance of the sum (N µ); `analysis.contrast` propagates these errors. The built-in test emitter makes one trial per 8 ns word with a probability p, so its counts are **binomial** (variance N p (1 − p) per N words): close to Poisson for small p, narrower near p = 1. The board tests check the emitter against the binomial law.

A gate is **half-open**, [start, start + width): an event exactly at the end belongs to the next gate, so adjacent gates (a signal window and a reference window, or the steps of a gate-delay scan) never count an event twice and never miss one between them.

1.3 State discrimination: two Poisson states and a threshold

Fluorescence detection of a trapped ion or atom gives few counts when dark (background, mean μ_d) and many when bright (μ_b). A shot is called bright when $n \geq t$; the errors are

$\varepsilon_d = P(n \geq t \mid \mu_d)$, $\varepsilon_b = P(n < t \mid \mu_b)$, fidelity $F = 1 - (\varepsilon_d + \varepsilon_b) / 2$.

`analysis.two_poisson` fits the two means and the bright fraction to a histogram of SHOT records (EM), `analysis.threshold` finds the t of the largest F; set it as D1's `low_a` and the board decides every shot (on a pin within about 114 ns of the gate's end, in the records and the per-point sums). Real ions depart from two Poissons (a bright ion can be pumped dark during detection): F from the fit is then an upper bound; check with prepared states.

1.4 NV readout windows and the contrast

The NV centre's fluorescence at the start of a green readout pulse depends on its spin state (m_s = 0 brighter by about 30 % for 200–300 ns), then decays to the polarized level within about 1 µs. counter's gate G1 takes the signal (the first ~300 ns), G2 a reference late in the pulse, both per point; the normalized contrast C = 1 − S/R removes the laser's and the collection's drifts. With S, R ~ 0.03 counts per shot, a point needs ~3 × 10^5 shots for a 1 % error: the per-point sums on the board make that cheap (no per-shot records needed). The gate-delay scan (a 20 ns window stepped per point) shows the readout's time trace for choosing the windows.

1.5 Coincidences, clusters and accidentals

Two detectors behind a beam splitter count a **coincidence** when both click within a window W. counter groups events into **clusters** exactly as timetagger does: a cluster opens at an event and takes every later event within W of its first; its pattern is the set of channels in it, and the 15 exclusive patterns of four channels are counted. Inclusive counts ("C1 and C2, with or without C3") are sums of exclusive ones. Two independent sources of rates R1, R2 give **accidental** coincidences at about 2 W R1 R2 (a cluster opened by either one); the second unit, with one channel delayed far beyond the correlations, counts exactly those accidentals with the same rates and the same dead times, so

```
g2(0) ≈ N_12 / N_12,delayed
```

needs no rate model (`analysis.g2_zero`). For heralded photons, Grangier's α = N_h N_htr / (N_ht N_hr) < 1 is non-classical; for entangled pairs, the correlation E = (N++ + N−− − N+− − N−+) / Σ of four settings gives CHSH's S (|S| ≤ 2 classically, up to 2√2). Keep W small against 1 / R (accidentals) and large against the detectors' jitter.

1.6 counter and timetagger

timetagger time-stamps every event (ps resolution, 7 Mevents/s to the PC) and computes histograms and correlations on the PC; counter counts on the board (1 ns, 125 Mcount/s per channel) and decides within ~114 ns, keeping only sums, totals and optional per-shot records. Use timetagger for g2(τ), lifetimes and TCSPC; counter for pulsed sequences (per-point sums over hours, decisions in real time, gated counting) and for coincidence counts at high rates. Both use the same cluster definition, so their coincidence counts agree to counter's 1 ns.

2. Definitions (the contract)

2.1 Time

- The **sample clock** runs at 125 MHz; one **word** is one sample period (8 ns) and holds eight **slots** of 1 ns, bit 0 the earliest. A time is `t = 8 n + j` ns for slot j of word n, counted from the core's reset.
- Every input is sampled every 1 ns at the pins (an ISERDESE2 per pin in oversample mode, at 250 MHz in 0° and 90°: `common/rtl/optional/rp_pins1ns.v`); pin p's word of period n reaches the logic in clock n + 4. A pulse at least 2 ns wide is always seen; one shorter than 1 ns can fall between two samples.
- Times are referred to the connectors with the board calibration's `pins_loop_s` (an output pin through a jumper to a 1 ns input; this board 4.5 ns) and `pins_delay_s` (a pin against OUT1; −28.6 ns), as feedback does (`common/CALIBRATION.md`, "Delay of the digital pins").

2.2 Lanes and events

A **lane** carries the events of one input after its conditioning: the count channels C1–C10, TRIG, NEXT and SEQ (GATE is a level, carried as its 8-bit word). **Events in a lane are at least 8 ns apart** (the channels' dead time and the controls' hold-off are at least 8 ns, in hardware). So a lane holds at most one event per word, an event is `{valid, slot}`, a constant delay keeps one event per word, and every block works at 125 MHz.

2.3 Count channels

- **C1–C8 (DIO0_P–DIO7_P).** An edge at slot j of word n is a 0→1 change between the samples of slots j − 1 and j (slot −1 is the previous word's slot 7) of the channel's word: the pin's samples, inverted for a channel set to falling edges, ORed with the test emitter's pulses (2.11). So the emitter imitates a detector of the channel's polarity. Setting falling edges on a pin that idles low (or rising edges back on one that idles high) is itself one edge.
- **Dead time** D = 8 … 65535 ns (1 ns steps), **non-paralyzable**: an edge at t is counted if `t ≥ t_last + D`, where t_last is the last *counted* edge; the others are counted as rejected (a diagnostic). At most 125 Mcounts/s per channel. A disabled channel counts nothing, neither counted nor rejected edges.
- **C9, C10 (IN1, IN2).** A Schmitt trigger on the ADC samples, one per word (codes; the driver converts volts at the connector through the board calibration and the input's range). For negative pulses (NIM: −0.8 V into 50 Ω) the samples are inverted first (y = −x − 1, the codes' one's complement), else y = x. With a threshold T and a hysteresis H ≥ 1 code: the trigger is armed by a sample y ≤ T − H and fires at the next sample y[n] ≥ T (an event in word n), which disarms it. Between the two samples the crossing is interpolated: with a = T − y[n−1] and d = y[n] − y[n−1] (1 ≤ a ≤ d), the event's slot is `j = ceil(8a / d) − 1` (0 … 7: three compare-subtract steps). The event's time 8n + j lies 7 … 8 ns after the straight line's crossing; that constant is part of the analog lag. Events are at least two words apart (a sample must re-arm), and the event's word (a 1 at slot j, ORed with the test emitter's pulses for the channel) goes through the same edge and dead-time logic as C1–C8. A disabled analog channel counts nothing.
- **Alignment.** The analog lanes come Δ later than the pins' lanes for pulses that reach the connectors together (the converters' pipelines and filters): Δ = `adc_delay − pins_loop_s`
  - `pins_delay_s` (the board calibration, 0 unless measured; the DAC's own delay cancels) + A, the analog lanes' lag after the ADC port against the pins' timing as `pins_loop_s` defines it. A = 7.95 ns was measured on 2026-11-05 (OUT1's 8 ns pulses of 0.8 V on IN1 above 0.25 V, against an output pin wired to C8, with an oscilloscope: docs/VERIFICATION.md); the RTL's simulated lag after the port, 23.5 ns, timed C9 15.5 ns early. The difference (two clocks, within the measurement) is not explained: the pins' input pipeline counted against feedback's definition of `pins_loop_s` is the likely place. Pulses of another shape cross the threshold a little differently between the samples. While C9 or C10 is enabled the driver adds Δ (rounded to 1 ns) to every pin channel's delay, every gate's delay and the GATE level's delay, so that all settings and times refer to the connectors; decisions then come Δ later.

2.4 Delays and hold-off

- Every count channel's lane has a delay of 0 … 255 ns in 1 ns steps (cables, detectors, the analog alignment). The coincidence unit's delayed lane has 0 … 16 µs. The control inputs have none: a gate's delay sets its timing from TRIG.
- TRIG, NEXT and SEQ (each on rising or falling edges) have a **hold-off** (8 ns … 65.535 µs, default 24 ns): an edge within the hold-off of the previous accepted edge is ignored (bounces); the channels' dead-time rule. Their events of pin word n reach the shots two clocks before the channels' lanes of the same word, so a gate can start at its TRIG's own slot.
- GATE (active high or low) has a **minimum width** W (1 … 8 ns): the level at a slot is L if the W samples up to it are all L, else the previous slot's level, so pulses and gaps shorter than W never appear and a level appears W − 1 ns after it starts; then a delay of 0 … 255 ns.

2.5 Shots, points and sweeps

- **Runs.** Shots are taken only while a run runs. A run starts at once, or (armed) at the next SEQ, from that SEQ's slot; it stops by software, after a number of shots or after a duration (the automatic stops take effect one clock after their condition; a shot in progress completes).
- A **shot** starts at a TRIG (the pin, the internal timer, or software) when no shot is busy. A shot with TRIG in word n_T is busy for the words n_T … n_T + max(7, its gates' last words): at least 64 ns. A TRIG while a shot is busy is **missed** (counted, flagged in the busy shot's record). The internal timer gives a TRIG every period (64 ns … 34 s) at slot 0, and optionally a SEQ with every P-th TRIG (a self-contained sweep).
- The **point** index (0 … P − 1, P ≤ 1024) is latched at the TRIG. Modes: *per shot* (it advances after every TRIG, also a missed one, and SEQ resets it: one shot per τ of an NV sweep), *per NEXT* (NEXT advances it, SEQ resets it: N shots per point), *fixed* (a register: set by software). After P − 1 comes 0. Events of one word apply in slot order; at equal slots NEXT, then SEQ, then TRIG (a marker that starts a sequence and its first shot together gives point 0). A run starts at point 0 (fixed: the register).
- A **sweep**: a run opens sweep 0; a SEQ closes the open sweep if it had TRIGs and opens the next (a SEQ on an empty sweep keeps it). A closed sweep gives a SWEEP record (its TRIGs with the missed ones, the missed);

in per-shot mode it is checked: its TRIGs must equal P, else it is flagged bad (the record and a counter). A TRIG in the SEQ's word at a later or equal slot is the next sweep's.

2.6 Gates

- Up to four gates G1–G4 per shot. Gate g of a shot with TRIG at t_T and point i covers the half-open interval `[t_T + d_g + i·s_g, t_T + d_g + i·s_g + w_g)` (delay d, width w, 0 … 4.3 s in 1 ns steps; the delay step s of 0 … 16.7 ms per point for the **gate-delay scan**). An event at t is in the gate if t is in the interval: adjacent gates never count an event twice, overlapping gates both count it. A gate is *timed* (the interval) or *timed with the GATE level* (only where the interval and the GATE level, after its filter and delay, are both high: a gate pulse from the sequencer within a timed envelope). Width 0: an empty gate.
- A gate counts the events of its **channel set** (any of C1–C10, summed). The shot's count of gate g is n_g (32 bits inside, 16 bits saturating in records).

2.7 Decisions

- D1 and D2 each look at two gates A and B of the shot (any of G1–G4, also the same one twice): a = `low_a` ≤ n_A ≤ `high_a` and b = `low_b` ≤ n_B ≤ `high_b` (the counts saturated to 16 bits, as the SHOT records carry them, so that a PC recomputes every D from the records; a `high` of 65535 is no upper limit), and D = table[2b + a], any of the 16 functions of two bits (the driver names them: "a", "a and b", "a xor b", "not (a or b)", ...). For example "bright" in ion detection is D = a with a = G1 ≥ 3; "both detectors clicked" is a and b with a = G1 ≥ 1, b = G2 ≥ 1.
- A decision is **taken once per shot**, when the results of both its gates are in (a gate's result is in at its end; the gates' results come in the shots' order). D goes into the shot's SHOT record (2.12), into the per-point sums (D1 true, D2 true: 2.8) and into the decision's counters (taken, true, the last value; since loading).
- Its **pin** (an output that selects D1 or D2, 2.10):
  - *level* (the default): the last decision, held until the next shot's decision: for a receiver that reads a level when it needs it (a TTL input sampled by a sequencer). It gives an edge only when the decision changes; awg's sequence jumps on edges (rising, falling or both), so for a jump on every true decision use *pulse-true*;
  - *pulse-true*, *pulse-false*: a pulse of `pulse` (8 ns … 2.048 µs in 8 ns steps) on each true (false) decision; a decision during a pulse restarts it;
  - *early*: high from the moment gate A's running count of the shot reaches `low_a` (gate A's lower limit only; with `low_a` 0 from the gate's start) until the next shot's gates start: a "bright" signal ahead of the gate's end.
- **Latency** (from an edge in a gate's last ns to the output's edge; DIO channels at zero delay): 113.6 ns in the simulation of the RTL (the behavioural 1 ns sampler and the OSERDESE2 model, no pin buffers); the early pin follows the `low_a`-th counted edge with the same lag. On the board `measure_latency` (a jumper from the output to a count input) measures it from the gate's end to the output's connector: 101.5 ns with the timer's TRIGs (on the clock's grid); with a TRIG from a pin the decision waits for the clock, so the latency is 101 … 109 ns (105 ns on average) by where TRIG falls in the 8 ns word. An oscilloscope (TRIG's pin to the output's) gave 103.5 ns on average, within the jumpers' and probes' uncertainty (2026-11-05). The early pin: 105.5 ns after the deciding edge.

2.8 Per-point sums

- For every point i the FPGA sums over shots: n_1 … n_4, the shots, D1 true, D2 true, and Σ n_1^2 (64 bits; n_1 saturated to 2^24 − 1 for the square). Two banks: one counts, the other is frozen for the service, which reads it, adds it into 64-bit totals and releases it; a released bank is zeroed (8 µs) and is then ready to count. A swap happens between two shots' updates on request (the service's worker, every second), at a sweep's end if so set, and automatically before any 32-bit word of the counting bank could overflow; only a released and zeroed frozen bank can be swapped in (a request waits for it). A shot that would overflow without a possible swap is added with saturation (flagged). The SWAP record names the sweep and point at the swap.

2.9 Coincidences

The **cluster definition of timetagger** (`timetagger.analysis.coincidences`), with 1 ns times. Each of two units takes four lanes L0–L3, each any of C1–C10 after its delay (or none):

- the events of the four lanes in time order (equal times together); a cluster opens at an event that does not join the open cluster; an event at t joins the cluster that opened at t_0 if `t − t_0 ≤ W` (W = 8 … 1023 ns); the cluster closes at the first event with `t − t_0 > W`, which opens the next, or, if no event comes, with the word that holds t_0 + W (no later event can join);
- its **pattern** is the set of lanes it saw (bit i: L_i; repeated events of a lane count once); the unit counts each of the 15 **exclusive** patterns in 48-bit totals, never cleared (their sum is the clusters). Inclusive counts ("A and B, with or without C") are sums of exclusive ones, computed on the PC: the same numbers as timetagger's analysis of the same events;
- **gated by G1** (optional): only the events inside the shot's G1 window (and its GATE level, 2.6) take part (a cluster open at the gate's end closes as usual);
- unit 2 can **delay** one of its lanes by 0 … 16,383 ns (1 ns steps) before the unit: with a delay longer than the correlations it counts the accidental coincidences of the same rates directly (a lane delayed by D sees a channel's event of time t at t + D);
- a unit switched off drops its open cluster; switched on it starts empty.

With W ≥ 8 a word holds at most one closing and one opening cluster, so a unit closes at most one cluster per clock.

2.10 Outputs

O1–O4 on DIO4_N–DIO7_N, 2 ns bits (an OSERDESE2 per pin), high impedance until enabled. Each selects a source: D1 or D2 (their pins, 2.7), a gate G1–G4 (its window, 2 ns bits), TRIG (the accepted TRIGs as 4 ns pulses), a channel's **echo** (its counted edges after its delay and dead time, as 4 ns pulses: a detector's clicks as clean TTL, also C9/C10's discriminated pulses), a **herald** (a 4 ns pulse from slot 0 of the word in which a unit closes a cluster whose pattern holds all lanes of the output's mask), the rate alarm (high while on), the test emitter (a lane's event at slot j as a 4 ns pulse from 2 ⌊j / 2⌋ ns: two 2 ns bits) or a fixed pattern of four 2 ns bits per word. The 4 ns pulses start at 2 ⌊j / 2⌋ ns of their event's slot j, so events 8 ns apart stay 8 ns apart.

2.11 Test emitter

Four lanes of pseudo-random events (two xoroshiro128+: in clock c their words c − 1, 0 in clock 0), each drawing one trial per word: an event if u < p (24 bits each: p / 2^24 per word, a rate of p / 2^24 · 125 MHz) at a uniform slot (3 bits); lane 1 takes u = r0[63:40] and the slot r0[39:37], lane 2 r0[36:13] and r0[12:10], lanes 3 and 4 the same bits of r1. The pulses of pin word n come from the draw in clock n. Each event is a 2 ns pulse (slots j and j + 1, into the next word from slot 7) ORed into its channel's word (2.3), so the edge and dead-time stage is tested too; pulses 1 or 2 ns apart merge into one edge. Counts per gate are binomial (variance N p (1 − p) per word), not Poisson. The lanes pair as (1, 2) and (3, 4), each pair in a mode: independent; pairs (lane 2 copies each event of lane 1 with probability p_2, at the same slot, 0 … 31 words later: heralded pairs); split (each event of lane 1 goes to lane 2 with probability p_2, else stays on lane 1: never both, an antibunched source behind a beam splitter). **Shot states**: at each accepted TRIG the state is drawn, bright with probability table[point] (1024 entries of 17 bits loaded by the PC: 65536 is always), from 16 further random bits; the decay d starts at 1.0 (2^16) and falls by d −= d >> k per word (k = 0: none; τ ≈ 2^k × 8 ns). Each lane's probability is q = p + s · ((Δp · d) >> 16), clamped to 0 … 2^24 − 1 (Δp signed, per lane; s = 1 bright, 0 dark), two clocks after its terms: the state reaches the draws of pin word n_T + 11, so it takes effect 88 ns after a TRIG at slot 0 (the gates' delays should allow for it). Lanes can also go to an output pin, or (from step 4) to OUT1 as pulses (volts) for the analog channels.

2.12 Records

32-bit little-endian words in a ring in DDR (`common/rtl/optional/rp_ring_writer.v`), served by the ring pump (`rpfpga.ringworker`, `rpfpga.ringpump`). A record starts with a header word: bits 31..27 the type, 26..20 the number of words that follow, 19..0 type-specific. Types: 1 SHOT (point, flags D1/D2/missed/saturated, the TRIG's time in ns (48 bits), the sweep, the gates' counts saturated to 16 bits), 2 RATE (the 48-bit totals of channels and patterns, read at fixed offsets after an interval's end so that differences give exact intervals), 3 SWEEP (index, TRIGs, missed, flags), 4 SWAP (the bank frozen, why, sweep, point), 5 RUN (start or stop, its reason, the run's index, time, shots, missed). Type 31 is reserved: 0xF8000000 is the writer's padding (NOP). The layouts: PROTOCOL.md "RECORDS STREAM". The shots' records (RUN, SWEEP, SWAP, SHOT, in this priority) and the RATE records are kept in FIFOs of whole records and merged into the ring one record at a time: records of different FIFOs may be out of time order by a few records; each carries its time.

- **RATE** (12 words, + 15 for each coincidence unit that is on): while on, a record every T words (T ≥ 64; the first T words after on; made while the ring writer runs and the record FIFO has room, else dropped and counted); its sequence number counts every firing. E (48 bits) is the record's time in pin words; channel k's value (the

low 32 bits of its total) counts its events of (delayed) pin words < E + k − 1 (the totals are read one per clock). Then the low 32 bits of the pattern totals 1 … 15 of unit 1, then of unit 2 (each unit that is on; flags in the record), read one per clock after the channels'. Consecutive records therefore differ by the counts of exactly T words in every channel and every pattern: exact rates, without a gap or an overlap.

2.13 Rate alarm and monitors

- **Alarm**: while on, windows of W words count the events of a set of channels (after their delays); at each window's end the alarm is on if the count was below a level, else off (a lost ion, a lost focus). It shows in STATUS, in RATE records and on an output pin.
- **Monitors** (OUT1, OUT2): while on, windows of 2^k words (k = 4 … 24: 128 ns … 134 ms) count a set of channels; after each window the output is offset + scale × rate in volts (through the board calibration), clipped to the output's range: an analog rate meter for alignment. **Pulses** (the analog channels' test): the output is offset + amplitude for the one sample (8 ns) of each word in which a test emitter lane has an event, else the offset. Off: the offset (0 V by default).

3. Version 1.0 (the plan's steps 1 to 5)

The pipeline, in clocks c of the core (the bit-exact model steps the same registers):

Stage	Module	In clock c
pins	`rp_pins1ns` (12 ISERDESE2, oversampling at 250 MHz, 0° and 90°)	pin word c − 4
emitter	`ct_emit` (two `rp_xoro`, the modes, the pairs' delay)	the pulses of pin word c − 4
analog	`ct_analog` × 2: the Schmitt trigger, three compare-subtract steps	the event word five clocks after the sample on the port; its lane 23.5 ns (23 … 24) after the interpolated crossing at the port in simulation (the alignment uses the measured A = 7.95 ns: 2.3)
channel words	`ct_core`: pin ^ falling, OR the pulses of the lanes aimed at the channel	pin word c − 5
channels	`ct_chan` × 10: edges, the dead time, rejected edges	events of pin word c − 6
delays	`ct_delay` × 10: the slot shift (a carry into the next word), then 0 … 31 words in an SRLC32E	(delayed) pin word c − 8
totals	`ct_rates`: `rp_count48` (a DSP48) per lane, rejected-edge counts	pin words < c − 9
RATE records	`ct_rates` → `ct_fifo` (64 words, distributed RAM) → `rp_ring_writer`; the units' patterns through their record port	E = the firing clock − 5
alarm, monitors	`ct_rates`, `ct_mon` × 2 (the product in a DSP48)	windows of the lanes
controls	`ct_ctl`: TRIG, NEXT, SEQ through `ct_chan` (hold-offs), GATE's filter and delay	events of pin word c − 6; GATE's level of pin word c − 9
shots	`ct_shots`: runs, the TRIG's fate, the point, sweeps, the timer	a TRIG of pin word c − 6 accepted in clock c; done in c + end_k + 5
gates	`ct_gate` × 4: d + p·s kept with the point (a DSP48 after writes), word counters	the lanes of pin word c − 9 (registered once more), added in c + 1
per-point sums	`ct_sums`: 2 banks × 9 columns × 1024 (18 BRAM36), read-modify-write per shot	an update in clocks e + 1 … e + 5 of a done in e
coincidences	`ct_coinc` × 2: the lanes' selection, unit 2's slot shift and ring (a block RAM), the clusters, 15 × 48-bit totals in distributed RAM	the gates' lanes registered once more (pin word c − 10) with G1's window; unit 2 four clocks later; a close counted in the next clock
decisions	`ct_decide` × 2: the windows on the gates' results, the table, the pin's mode	taken in the clock after the later gate's result (`fin`); D in the sums and the SHOT record one clock after the shot's done
records	`ct_shotrec` (RUN, SWEEP, SWAP, SHOT) and `ct_merge` (with the RATE FIFO)	whole records into the ring writer
outputs	`ct_core` → `rp_oserdes2ns` × 4	the pattern, an emitter lane, the alarm, a gate's window, the TRIGs, a channel's echo, a decision's pin (without the outputs' register: one clock less)

- **C9, C10**: the ADC streams' samples (16-bit codes, one per clock) into `ct_analog`; its word is ORed with the emitter's pulses for the channel. The measured lag A = 7.95 ns (the simulated 23.5 ns was 15.5 ns too long against the pins) enters the alignment Δ (2.3); with this board's calibration (`adc_delay` 119.9 ns, `pins_loop_s` 4.5 ns, `pins_delay_s` −28.6 ns) Δ = 95 ns, measured within 0.5 ns with a cable and a pin jumper (2026-11-05).
- **Monitors**: the rate's scale shift moves in steps of 4 (two levels of the shifter fewer; the gain keeps at least 13 bits).
- **Counters in DSP48s** (`rp_count48`, its increment registered first): the channels' totals and rejected edges, the emitter's events, the decisions' counts (the last three one clock later than in 0.3; their low 32 bits are read). The 7010's slices are nearly all used, and each such counter frees about 8 slices.
- **The decision's path**, for an edge in pin word w (the gate's last): the lanes of w are in clock w + 8 after the delays and in w + 9 at the gates; the gate's two counting stages and its result (`fin`) follow, and the decision's registers change in clock w + 12 (the model's `Decide`, `test_model.DecisionTests`). A decision's pin drives its serializer directly, which puts it on the pin within the next word. The early pin compares gate A's running count (`count`, `cnt_on`) with `low_a` every clock, on the same path.
- **The register window** is 128 KiB: the emitter's table at 0x4000, the frozen bank at 0x10000 (REGISTERS.md).
- **The service** runs the record ring (`worker.py`, a subclass of `rpfpga.ringworker`; the stream's hub is `rpfpga.ringhub`): the FPGA makes records only while the writer runs. Every 0.1 s the worker reads a frozen bank (if a swap gave one) into the driver's 64-bit totals and releases it, and every second it requests a swap (`RESULTS?`).
- Resources and timing of the build: VERIFICATION.md.

counter commands (TCP)

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

Command	Reply
`PINS?`	`OK {json}`: for C1..C8, TRIG, GATE, NEXT, SEQ: `{"pin", "level", "seen"}` (the latest 1 ns sample; a rising edge since the last clear)
`PINS? CLEAR`	the same, then clears the rising edges
`HARDWARE?`	`OK {json}`: `abi`, `locked` (the samplers' PLL), `alarm`, `writer_running`, `writer_tripped`, `written`, `bursts`, `stalls`, `rate_records` (firings since on), `records_dropped`
`COUNTERS?`	`OK {json}`: `time_s` (the core's clock), `channels`: {"1".."10": {`total` (48 bits, never cleared), `rejected` (32 bits)}}, `emitter`: {"1".."4": events (32 bits)}, `alarm`: {`on`, `last_count`}. Each total is exact (its high bits latched with its low word), but the channels are read one after another, some microseconds apart: at MHz rates two channels' differences between two readings differ by a few counts (measured: up to ±10, ±170 when the board's CPU was interrupted). For exact simultaneous intervals use RATE records
`RATES? [<s>]`	`OK {json}`: `interval_s`, `channels`: {"1".."10": counts/s} from two readings of the totals `<s>` apart (0.001 ... 10 s, default 0.1)
`WORKER?`	`OK {json}`: the service worker's statistics (`bytes`, `polls`, `restarts`, `pump`, `banks`)
`RUN [KEEP]`	start a run (at once, or at the next SEQ if `shots.start_at_seq`); the per-point results are cleared first unless `KEEP`; `OK {json}` as `RUN?`
`STOP`	stop the run (a shot in progress completes); `OK {json}` as `RUN?`
`TRIG`, `NEXT`, `SEQ`	a software TRIG (with `shots.trigger` software), NEXT or SEQ, at slot 0 of the word the write reaches
`CLEAR`	discard the per-point results (both banks zeroed, the totals 0)
`RUN?`	`OK {json}`: `state` (idle, armed, running), `shot_busy`, `point`, `shots`, `missed`, `sweeps`, `bad_sweeps`, `sweep`, `swaps`, `bank`, `released`, `zeroing`, `saturated`, `records_dropped` (the counts are the run's, 32 bits, wrapping: `shots` after $4.29 \cdot 10^9$, use the per-point sums' 64-bit shots for long runs; `records_dropped` since loading, not cleared by a run: compare differences)

Command	Reply
`RESULTS? [NOW]`	`OK {json}`: the per-point totals since the last clear (64 bits): `shots`, `n1`...`n4`, `d1`, `d2`, `n1_sq` (lists over the points), `banks` (banks read), `mean` {`n1`...`n4`: counts per shot}. The service's worker collects the frozen banks (a swap every second); `NOW` freezes and reads the counting bank first
`TABLE [p0, p1, ...]`	the emitter's bright probability of each point (0 ... 1, up to 1024; the rest 0; not kept across a load); `OK <n>`
`DECISIONS?`	`OK {json}`: {"1", "2": {`taken`, `true` (32 bits, wrapping, since loading), `last`}}
`LATENCY? [<output> <channel> [<jumper s>]]`	with O`<output>` wired to C`<channel>` (default 1 8): the decision latency (`measure_latency`: D1 pulses on each shot's G1 end, a 1 ns gate-delay scan finds the edge); `OK {json}`: `found`, `latency_s` (from the gate's end to the output's connector, the jumper's delay subtracted), `arrival_s`, `edges_per_shot`, `shots`, `means`. About 0.1 s; the configuration is restored
`COINC?`	`OK {json}`: for units "1", "2": `enable`, `channels`, `window`, `counts` (16: the 48-bit totals by the lanes' bits, never cleared), `exclusive` {"C1+C2": n, ...}, `inclusive` {"C1+C2": the clusters with at least these channels}
`ANALOG? [<n>]`	`OK {json}`: {"C9", "C10": {`mean`, `min`, `max`, `std`}}: IN1's and IN2's latest samples in volts, read n times (1 ... 4096, default 256; microseconds apart: for choosing a threshold, not a waveform)
`RECORDS STREAM`	the records (below)

Configuration (`CONFIG {json}` patches, `CONFIG REPLACE {json}`, `CONFIG?`)

Times in seconds, rates in counts (or events) per second, volts at the connectors (through the board calibration). A patch changes only what it names.

```
{"channels": {"1".."8": {"enable": false, "edge": "rising", "dead_time": 8e-9, "delay": 0.0},
              "9", "10": {"enable": false, "polarity": "positive", "threshold": null, "hysteresis":
               ↪  0.0,
                          "dead_time": 8e-9, "delay": 0.0}},
 "coincidences": {"1": {"enable": false, "channels": [], "window": 8e-9, "gate": false},
                  "2": {"enable": false, "channels": [], "window": 8e-9, "gate": false, "delayed":
                   ↪  null,
                        "delay": 0.0}},
 "controls": {"trig": {"edge": "rising", "holdoff": 24e-9}, "next": {...}, "seq": {...},
              "gate": {"active": "high", "min_width": 1e-9, "delay": 0.0}},
 "shots": {"trigger": "pin", "timer_period": 1e-6, "timer_seq": false, "points": 1,
           "point_mode": "per-shot", "fixed_point": 0, "records": false, "swap_at_sweep": false,
           "start_at_seq": false, "shot_limit": 0, "duration": 0.0},
 "gates": {"1".."4": {"mode": "off", "channels": [1], "delay": 0.0, "width": 0.0, "step": 0.0}},
 "emitter": {"lanes": {"1".."4": {"rate": 0.0, "channel": 0, "bright": 0.0}},
             "pairs": {"1-2": {"mode": "independent", "probability": 0.0, "delay": 0.0},
                       "3-4": {"mode": "independent", "probability": 0.0, "delay": 0.0}},
             "decay": 0.0},
 "rates": {"enable": false, "interval": 0.1},
 "alarm": {"enable": false, "channels": [1], "window": 0.1, "below": 1000.0},
 "monitors": {"1": {"mode": "off", "channels": [1], "window": 0.001048576, "scale": 1e-6, "offset":
  ↪  0.0},
              "2": {...}},           (and "lane": 1, "amplitude": 0.0 for the pulses)
 "decisions": {"1", "2": {"gate_a": 1, "low_a": 0, "high_a": null, "gate_b": 1, "low_b": 0,
  ↪  "high_b": null,
                          "logic": "false", "output": "level", "pulse": 8e-9}},
 "outputs": {"1".."4": {"mode": "off", "pattern": 0, "lane": 1, "gate": 1, "channel": 1, "unit": 1,
  ↪  "herald": []}}}
```

- **channels**: `edge` rising or falling; `dead_time` 8 ns ... 65.535 µs and `delay` 0 ... 255 ns, both rounded to 1 ns (non-paralyzable dead time: docs/THEORY.md 2.3). **C9, C10** (IN1, IN2): `polarity` positive (fires at the first sample ≥ `threshold` after one ≤ `threshold` − `hysteresis`) or negative (≤ `threshold`, armed ≥ `threshold` + `hysteresis`); `threshold` in volts at the connector (through the board calibration and the input's range; `null`: never); `hysteresis` ≥ 0 V (at least one code; one code reads 0). While C9 or C10 is

enabled, the pin channels', the gates' and GATE's delays get the analog lag Δ added (`describe` shows it): their settings stay referred to the connectors, a pin channel's delay is then at most 255 ns − Δ.

- **coincidences**: each unit's `channels` (up to four of C1 … C10, distinct; their order is the lanes' L0 … L3), `window` W (8 … 1023 ns, rounded to 1 ns), `gate`: only events inside G1's window; unit 2's `delayed` (one of its channels, or `null`) is seen `delay` later (0 … 16.383 µs, 1 ns steps).
- **emitter**: each lane's `rate` (0 … 125 Mevents/s × (1 − 2^−24), one trial per 8 ns word, rounded to 125 MHz / 2^24 = 7.45 /s) and `channel` (0: none, 1 … 10: ORed into that channel). A pair's `mode`: `independent`; `pairs` (lane 2 copies each event of lane 1 with `probability`, `delay` later: 0 … 248 ns in steps of 8 ns); `split` (each event of lane 1 goes to lane 2 with `probability`, else stays on lane 1: lane 1's `rate` is the source's). In `pairs` and `split` lane 2's own `rate` is unused (reads back 0).
- **rates**: RATE records every `interval` (10 µs … 10 s, rounded to 8 ns) while the service's worker runs the ring.
- **alarm**: on when the selected channels together count fewer than `below` × `window` (rounded, at least 1) in a window (1 µs … 2 s).
- **monitors**: `mode` off (OUT = `offset`) or rate: OUT = `offset` + `scale` × the rate of the selected channels (V per count/s), updated every `window` (2^k × 8 ns, k = 4 … 24: 128 ns … 134 ms), within the outputs' range (about ±1 V); or pulses: OUT = `offset` + `amplitude` for 8 ns at each event of emitter lane `lane` (the analog channels' test signal; a cable from OUT1 to IN1).
- **outputs**: `mode` off (high impedance), low, high, pattern (4 bits per 8 ns, bit 0 first, 2 ns each), emitter (`lane`'s events as 4 ns pulses), alarm (high while the alarm is on), gate (`gate`'s window, 2 ns bits), trig (the accepted TRIGs as 4 ns pulses), d1, d2 (the decision's pin), echo (`channel`'s counted edges as 4 ns pulses; C9, C10 too), herald (a 4 ns pulse when coincidence unit `unit` closes a cluster that holds all of `herald`'s channels, which must be the unit's; [] any cluster).
- **decisions**: D = `logic` of a (`low_a` ≤ the count of G`gate_a` ≤ `high_a`, the count saturated to 16 bits as in the SHOT records; `low` 0 … 65535, `high` 0 … 65534 or `null`: no limit) and b (the same with `gate_b`); `logic`: "false", "true", "a", "b", "not a", "not b", "a and b", "a or b", "a xor b", "a == b", "a and not b", "b and not a", "a or not b", "b or not a", "not (a and b)", "not (a or b)", or the table 0 … 15 (D = bit 2b + a). `output`: the pin of an output in mode d1/d2: level (the last decision, held), pulse-true, pulse-false (a pulse of `pulse`, 8 ns … 2.048 µs, rounded to 8 ns), early (high as soon as G`gate_a`'s running count reaches `low_a`, until the next shot). Taken once per shot at its gates' end; D1, D2 in the SHOT records and the per-point sums (`RESULTS? d1`, `d2`).
- **controls**: TRIG (DIO0_N), NEXT (DIO2_N), SEQ (DIO3_N) on rising or falling edges with a hold-off of 8 ns … 65.535 µs (1 ns steps); GATE (DIO1_N) active high or low, its minimum width 1 … 8 ns and its delay 0 … 255 ns.
- **shots**: `trigger` pin, timer (every `timer_period`, 64 ns … 34 s, rounded to 8 ns; `timer_seq`: a SEQ with every `points`-th TRIG) or software (`TRIG`); `points` 1 … 1024; `point_mode` per-shot, per-next or fixed (`fixed_point`); `records`: SHOT records; `swap_at_sweep`: the per-point banks swapped at each sweep's end; `start_at_seq`: a run starts at the next SEQ; `shot_limit` (0: none) and `duration` (0: none; rounded up to 2.048 µs) stop a run.
- **gates**: `mode` off, timed, gate (timed and where the GATE level is high); the `channels` counted (summed); `delay` and `width` 0 … 4.29 s and `step` (added to the delay per point: the gate-delay scan) 0 … 16.7 ms, all in 1 ns steps.
- **emitter** (states): each lane's `bright` (the rate added in a shot's bright state, signed); `decay`: the bright state's decay time (0: none; the nearest of −8 ns / ln(1 − 2^−k)). The bright probability of each point: `TABLE`.

Presets: `off` (all off: the power-up state), `rates` (C1..C8 counting, RATE records every 0.1 s, C1's rate on OUT1 at 1 V per Mcount/s), `selftest` (the emitter on C1..C4 without wiring: 1 Mcount/s on C1, half of them copied 16 ns later to C2, a split source of 2 Mcount/s on C3 and C4; RATE records every 0.1 s), `shots-selftest` (shots from the timer every 10 µs over 16 points with its SEQs, the emitter's lane 1 on C1 at 0.5 Mcount/s with 5 Mcount/s more when bright, decaying with 1 µs, G1 counting C1 for 2 µs from 100 ns; load a table with `TABLE`).

RECORDS STREAM

On its own connection: `RECORDS STREAM`, the reply line `OK {header json}` and then binary frames until the client sends `STOP` or closes. The board's ring pump serves the same stream on port 5026 (`RecordStream(host)` connects there first). The header: `format` "CNR1", `word_s` 8e-9, `channels` 10, `records` (the types), `ring_bytes`.

A frame is a 24-byte header, little-endian u32: magic "CNR1", sequence, payload bytes, bytes lost before this frame (a client that falls behind loses whole frames), 0, 0; then the payload: records of 32-bit little-endian words, each a header word and its words. NOP words (0xF8000000, the writer's padding) may come between records. `counter.model.decode(words)` reads them; `counter.analysis.parse_frame(frame)` splits a frame (numpy).

Bits of the header word	Meaning
31..27	type (1 SHOT, 2 RATE, 3 SWEEP, 4 SWAP, 5 RUN, 6 TIME; 31 reserved)
26..20	the number of words that follow
19..0	type-specific

RATE (type 2, 12 words + 15 per coincidence unit that is on): header [19:0] the record's sequence number (mod 2^20, from 0 at `rates` on; every firing counts, so a gap shows a lost record), then E[31:0], {[18:17] the units in the record, [16] the alarm, [15:0] E[47:32]}, then the low 32 bits of the totals of C1 … C10, then those of patterns 1 … 15 of unit 1 and of unit 2 (each that is on). E is in pin words of 8 ns (the core's clock); C k's value counts its events (after its delay) of pin words $< E + k - 1$. Consecutive records differ by exactly the interval in every channel and pattern, so `(b − a) mod 2^32` is the exact count of an interval (`model.rate_differences`, `analysis.rate_table`, its `patterns`). A record in progress when `rates` is switched off completes.

SHOT (type 1, 4 words; with `shots.records`): header [9:0] the point, [10] a TRIG was missed during the shot, [11] a count saturated (above 65535), [12] D1, [13] D2; then the TRIG's time in ns [31:0], {the sweep's index [31:16], the time [47:32]}, {n2, n1}, {n4, n3} (each 16 bits, saturated). D1 and D2 are the shot's decisions (from 32-bit counts).

SWEEP (type 3, 4 words): header [15:0] the closed sweep's index, [16] checked (per-shot points), [17] bad (its TRIGs differ from the points); then the SEQ's time in words of 8 ns [31:0], [47:32], the sweep's TRIGs (with the missed), the missed.

SWAP (type 4, 3 words): header [0] the bank frozen, [1] requested by software, [2] at a sweep's end, [3] automatic (a word could overflow), [4] the frozen bank had a saturated word; then the time in words [31:0], [47:32], {the sweep [31:16], the point [9:0]}.

RUN (type 5, 4 words): header [0] a stop (0: a start), [3:1] the stop's reason (1 software, 2 the shots' limit, 3 the duration), [19:4] the run's index; then the time in words [31:0], [47:32], the run's shots, the missed (a stop; 0 at a start).

The coincidences' patterns are in the RATE records (above).

counter registers

`ct_core` is a 128 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 `CNTR`, version 1.0).

Layout version (ABI) **5**, counter 1.0 (as 0.4): the counting of the plan's step 1 (0x000 .. 0x18C), the shots of step 2 (0x200 .. 0x2D4, the emitter's table at 0x4000, the frozen per-point bank at 0x10000), the decisions of step 3 (0x300 .. 0x33C, OUT_SEL's sources 5 and 6) and the analog channels and coincidences of step 4 (0x340 .. 0x4FC, OUT_SEL's source 7, MON_CFG's pulses).

Timing. A write whose address and data the bus presents in clock c takes effect in clock c + 2; a read presented in clock c returns the value of clock c + 1 (the bit-exact model, `control/counter/model.py`, and `tests/tb_front.sv` use exactly this). Clock c is the value of TIME (the core's `now`).

General

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 rate alarm
0x008	LEVELS	R	each input's latest 1 ns sample: [7:0] DIO0..7_P (C1..C8), [11:8] DIO0..3_N (TRIG, GATE, NEXT, SEQ)
0x00C	SEEN	R / W1C	a rising edge since the last clear, same bits; writing 1 clears a bit
0x018	TIME_LO	R	clocks (8 ns) since the core's reset, bits 31..0 (reading it latches TIME_HI)
0x01C	TIME_HI	R	bits 63..32, latched by the last TIME_LO read

Channels (C1..C10; C9, C10 are the analog discriminators of step 4 and idle in 0.1)

Offset	Name	Access	Meaning
0x024	CH_CTRL	RW	[9:0] channel enabled; [25:16] falling edges (the pin inverted)
0x080 + 4 i	CH_CFG i	RW	channel i + 1: [15:0] dead time − 8 (ns; the dead time 8 … 65,535 ns), [23:16] delay (0 … 255 ns)
0x0C0 + 8 i	TOTAL_LO i	R	lane i's 48-bit total, bits 31..0 (reading it latches its bits 47..32)
0x0C4 + 8 i	TOTAL_HI i	R	[15:0] bits 47..32, latched by the last TOTAL_LO read
0x110 + 4 i	REJECTED i	R	channel i + 1's edges not counted (within its dead time), 32 bits, wrapping

The totals count each lane's events after its delay and are never cleared (2^48 counts: 26 days at 125 Mcount/s); in clock c they hold the events of (delayed) pin words < c − 9.

RATE records and the alarm

Offset	Name	Access	Meaning
0x028	RATE_PERIOD	RW	T − 1: a RATE record every T clocks (10 µs … 10 s from the driver; values below 63 act as 63, so at most one record per 64 clocks; power-up 12,499,999: 0.1 s). The driver keeps it while records are off: written before RATE_CTRL, a short period would otherwise fire in the clocks between the two writes
0x02C	RATE_CTRL	RW	[0] on; off resets the sequence count (the first record T clocks after on)
0x030	ALARM_CTRL	RW	[9:0] the lanes counted; [16] on
0x034	ALARM_WINDOW	RW	W − 1: windows of W clocks
0x038	ALARM_LEVEL	RW	the alarm is on after a window with fewer counts than this
0x03C	ALARM_LAST	R	the last window's count (kept when the alarm is turned off)
0x060	REC_DROPS	R	RATE records not made for want of room in the record FIFO (the mailbox counts these and the shots' records' drops)
0x064	RATE_SEQ	R	the RATE firings since on (also those without a record while the writer is off)

The ring writer (owned by the service's worker, `rpfpga.ringworker`)

Offset	Name	Access	Meaning
0x040	W_CTRL	RW	[0] the ring writer runs (a rising edge restarts it: written position 0, base and size taken)
0x044	W_BASE	RW	the ring's physical address (4 KiB mailbox, then the data area)
0x048	W_SIZE	RW	the ring's size in bytes
0x04C	W_RD	RW	the consumer's position (bytes, mod 2^32); every write is the watchdog's heartbeat
0x050	W_WD	RW	the watchdog in clocks (0: off)
0x054	W_WR	R	the written position (bytes, mod 2^32), after the bursts' responses
0x058	W_BURSTS	R	data bursts written
0x05C	W_STALLS	R	clocks with a full burst waiting for room

Records are made only while the writer runs. The mailbox (the ring's first 4 KiB) carries the magic 0x434E5452 ("CNTR"), its sequence, the written position, the core's time and REC_DROPS.

The test emitter

Offset	Name	Access	Meaning
0x140 + 4 l	EM_P l	RW	lane l + 1: [23:0] p, the probability of an event per word × 2^24 (a rate of p / 2^24 × 125 MHz); in the modes pairs and split, lane 2's (4's) p is the copy probability or the fraction
0x150	EM_CFG	RW	[4l+3:4l] lane l + 1's channel (0 none, 1 … 10); [17:16] lanes 1, 2 and [19:18] lanes 3, 4: 0 independent, 1 pairs, 2 split
0x154	EM_DELAY	RW	in pairs mode: [4:0] lane 2's delay, [12:8] lane 4's (words of 8 ns, 0 … 31)

Offset	Name	Access	Meaning
0x160 + 4 l	EM_COUNT l	R	lane l + 1’s events (32 bits, wrapping)

Monitors (OUT1, OUT2) and outputs (O1..O4)

Offset	Name	Access	Meaning
0x170 + 4 o	MON_CFG o	RW	OUT(o + 1): [9:0] the lanes counted; [20:16] k (windows of 2^k clocks, 4 … 24); [24] on; [25] pulses (instead of the rate); [27:26] the pulses’ emitter lane − 1
0x178 + 4 o	MON_GAIN o	RW	the rate: [17:0] the gain g (signed); [28:24] the shift s (in steps of 4: bits 25:24 ignored). Pulses: [15:0] the amplitude (signed code)
0x180 + 4 o	MON_OFFSET o	RW	[15:0] the offset code (signed; read sign-extended)
0x188 + 4 o	MON_CODE o	R	the output’s code (signed)
0x010	OUT_EN	RW	[3:0] O1..O4 (DIO4..7_N) driven; 0 = high impedance (power-up, SAFE)
0x014	OUT_PAT	RW	[4k+3:4k] output k + 1’s bits of each sample period, bit 0 first, 2 ns each
0x020	OUT_SEL	RW	[8k+2:8k] output k + 1’s source: 0 the pattern, 1 an emitter lane ([8k+5:8k+4]: the lane − 1; 4 ns pulses), 2 the rate alarm (high while on), 3 a gate’s window ([8k+5:8k+4]: the gate − 1; 2 ns bits, a bit high when either of its two slots is in the window), 4 the accepted TRIGs (4 ns pulses), 5 a decision’s pin ([8k+4]: 0 D1, 1 D2), 6 a channel’s echo ([8k+7:8k+4]: the channel − 1, 0 … 9; its counted edges as 4 ns pulses), 7 a herald ([8k+7:8k+4]: the lanes that a cluster must hold; the unit in OUT_HERALD; a 4 ns pulse from slot 0 of the word after the cluster closed)

A monitor’s code is `sat16(offset + ((last · g) >> s))` four clocks after each window’s end (`last`: the window’s count, saturated to 2^24 − 1); with pulses it is `sat16(offset + amplitude)` in the clock after each event of the emitter lane, else the offset; while off it is the offset (the driver’s calibrated 0 V by default).

Control inputs, shots and gates (plan step 2)

Offset	Name	Access	Meaning
0x200	CTL_CFG	RW	[0] TRIG on falling edges, [1] GATE active low, [2] NEXT falling, [3] SEQ falling; [10:8] GATE’s minimum width − 1 (ns); [23:16] GATE’s delay (0 … 255 ns)
0x204	HOLD_TN	RW	[15:0] TRIG’s hold-off − 8, [31:16] NEXT’s (ns; power-up 16: 24 ns)
0x208	HOLD_S	RW	[15:0] SEQ’s hold-off − 8 (ns; power-up 16)
0x20C	SHOT_CFG	RW	[1:0] TRIG: 0 the pin, 1 the timer, 2 software only; [3:2] the point: 0 per shot, 1 per NEXT, 2 fixed; [4] SHOT records; [5] a swap at each sweep’s end; [6] runs start at the next SEQ; [7] the timer gives a SEQ with every P-th TRIG
0x210	POINTS	RW	[10:0] P (1 … 1024; power-up 1); [25:16] the fixed point
0x214	TIMER	RW	the timer’s period − 1 (clocks; power-up 124: 1 µs)
0x218	RUN_CTRL	W	pulses: [0] start, [1] stop, [2] TRIG, [3] NEXT, [4] SEQ (software; slot 0), [5] swap, [6] release the frozen bank, [7] clear (between runs: zero both banks)
0x21C	SHOT_LIMIT	RW	a run stops after this many shots (0: none)
0x220	DUR_LIMIT	RW	a run stops after this many × 256 clocks (0: none)
0x224	RUN_STATUS	R	[1:0] 0 idle, 1 armed (waits for a SEQ), 2 running; [2] a shot busy; [3] the counting bank; [4] the frozen bank released; [5] zeroing a bank; [6] a word saturated; [25:16] the point register
0x228	SHOTS	R	accepted TRIGs of the run (32 bits, wrapping: $4.29 \cdot 10^9$ shots, 4.9 h at 244,000 shots/s; the per-point sums’ shots are 64-bit)
0x22C	MISSED	R	TRIGs while a shot was busy (32 bits, wrapping)
0x230	SWEEPS	R	sweeps closed (32 bits, wrapping)

Offset	Name	Access	Meaning
0x234	BAD_SWEEPS	R	sweeps closed with TRIGs ≠ P (per-shot points)
0x238	SWAPS	R	the banks' swaps
0x23C	SREC_DROPS	R	the shots' records not made (their slot full or no room), since the bitstream was loaded (a run does not clear it)
0x240 + 0x20 g	G_CFG g	RW	gate g + 1: [1:0] 0 off, 1 timed, 2 timed and the GATE level; [25:16] the channels counted
0x244 + 0x20 g	G_DELAY g	RW	the delay d (ns)
0x248 + 0x20 g	G_WIDTH g	RW	the width w (ns)
0x24C + 0x20 g	G_STEP g	RW	[23:0] the delay's step s per point (ns)
0x250 + 0x20 g	G_LAST g	R	the gate's count of the last shot (32 bits, saturating)
0x2C0 + 4 l	EM_DP l	RW	[24:0] lane l + 1's step in the bright state (signed, per word × 2^24)
0x2D0	EM_DECAY	RW	[4:0] k: the bright state's decay d −= d >> k per word (0: none)
0x2D4	SWEEP_IDX	R	[15:0] the open sweep's index
0x4000 + 4 i	EM_TABLE i	W	[16:0] point i's bright probability × 65536 (65536: always)
0x10000 + 0x1000 c + 4 i	BANK c, i	R	the frozen bank: column c (0 n1, 1 n2, 2 n3, 3 n4, 4 shots, 5 D1 true, 6 D2 true, 7 $n1^2$ low, 8 $n1^2$ high) of point i

A gate of a shot with TRIG at t_T (in ns of the pin time) and point i covers [t_T + d + i·s, t_T + d + i·s + w); a shot is busy for its TRIG's word and the next max(7, the gates' last words) words, and its gates' results are final four clocks after its last word (docs/THEORY.md 2.5, 2.6). Writes of a gate's registers, of POINTS or SHOT_CFG make the gate compute d + p·s again (from 4 clocks after the last such write).

Decisions (plan step 3)

Offset	Name	Access	Meaning
0x300 + 0x20 k	D_CFG k	RW	D(k + 1): [1:0] gate A − 1, [3:2] gate B − 1, [7:4] the table (D = bit 2b + a), [9:8] the pin: 0 level, 1 pulses on true, 2 pulses on false, 3 early; [23:16] the pulse's words − 1
0x304 + 0x20 k	D_LOW_A k	RW	[15:0]: a = LOW_A ≤ n_A ≤ HIGH_A (n_A saturated to 16 bits)
0x308 + 0x20 k	D_HIGH_A k	RW	[15:0] (power-up 0xFFFF: no limit)
0x30C + 0x20 k	D_LOW_B k	RW	[15:0]: b = LOW_B ≤ n_B ≤ HIGH_B
0x310 + 0x20 k	D_HIGH_B k	RW	[15:0] (power-up 0xFFFF)
0x314 + 0x20 k	D_COUNT k	R	decisions taken since loading (wrapping)
0x318 + 0x20 k	D_TRUE k	R	of them true
0x31C + 0x20 k	D_LAST k	R	[0] the last decision

A decision is taken in the clock after the later of its two gates' results (each gate's result is in at its end); D enters the per-point sums and the SHOT record one clock after the shot's done. The early pin is gate A's running count ≥ LOW_A while that gate counts and until the next shot's gates are loaded (docs/THEORY.md 2.7).

Analog channels and coincidences (plan step 4)

Offset	Name	Access	Meaning
0x340 + 0x20 u	CO_CFG u	RW	unit u + 1: [0] on, [1] only inside G1's window; [25:16] W (ns, 8 … 1023; below 8 acts as 8; power-up 8; a new W applies to the clusters that open from the clock after its write takes effect)
0x344 + 0x20 u	CO_LANES u	RW	[4i+3:4i] lane i's channel − 1 (0 … 9; 10 … 15: none; power-up 0xFFFF)
0x368	CO_DELAY	RW	unit 2: [13:0] D (ns, 0 … 16,383), [17:16] the delayed lane

Offset	Name	Access	Meaning
0x380	OUT_HERALD	RW	[k] output k + 1's herald: 0 unit 1, 1 unit 2
0x3A0 + 0x10 k	AN_CFG k	RW	C(9 + k) on IN(1 + k): [0] negative pulses (the samples inverted: y = ~x)
0x3A4 + 0x10 k	AN_THRESH k	RW	[16:0] T (signed, in y; fires at y ≥ T; power-up 0x08000: never)
0x3A8 + 0x10 k	AN_HYST k	RW	[15:0] H (armed by y ≤ T − H; 0 acts as 1; power-up 1)
0x3AC + 0x10 k	AN_SAMPLE k	R	[15:0] the latest sample x (16-bit stream code)
0x400 + 0x80 u + 8 p	PAT u, p	R	unit u + 1's total of the exclusive pattern p (1 … 15: the lanes' bits), bits 31..0 (reading it latches bits 47..32)
0x404 + 0x80 u + 8 p	PAT_HI u, p	R	bits 47..32, latched by the last PAT read

The analog channel's event word enters its count channel (CH_CFG 8, 9: the dead time and the delay) five clocks after its sample was on the converter's port; the lanes of a pulse at the connector come Δ later than the pins' (docs/THEORY.md 2.3): the driver adds Δ to the pin channels', the gates' and GATE's delays while an analog channel is enabled. The units take the gates' lanes one clock later (pin word c − 10); unit 2's lanes, G1's window and its delayed lane 4 clocks after that (its delayed lane D ns more: a slot shift, then a ring of 4096 words in a block RAM). A closed cluster's pattern is counted one clock after its close (distributed RAM); the RATE records read the totals one per clock after the channels'.

Configuration registers

Restored and checked by `rpfpga.Overlay` (the default configuration compiles to the power-up values with the nominal calibration: 0 but RATE_PERIOD, HOLD_TN, HOLD_S, POINTS, TIMER, the decisions' HIGH_A, HIGH_B, CO_CFG, CO_LANES, AN_THRESH and AN_HYST as above), written in this order: CH_CTRL, CH_CFG 0..9, EM_P 0..3, EM_DP 0..3, EM_DECAY, EM_DELAY, EM_CFG, RATE_PERIOD, RATE_CTRL, ALARM_WINDOW, ALARM_LEVEL, ALARM_CTRL, MON_GAIN, MON_OFFSET, MON_CFG (both monitors), CTL_CFG, HOLD_TN, HOLD_S, TIMER, SHOT_LIMIT, DUR_LIMIT, POINTS, each gate's DELAY, WIDTH, STEP, CFG, each decision's LOW_A, HIGH_A, LOW_B, HIGH_B, CFG, each analog channel's THRESH, HYST, CFG, unit 1's LANES, CFG, unit 2's LANES, DELAY, CFG, OUT_HERALD, SHOT_CFG, OUT_PAT, OUT_SEL, OUT_EN. The ring registers belong to the service's worker; RUN_CTRL's pulses and the emitter's table are operations, not configuration.

Verification record

What was tested on counter, 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 modelled, simulated, built and then tested on the board (`make install-service`, `make test-hw`; the user's decision of 2026-11-04, which replaced "board tests only at 1.0"); the session with awg on a second board and an oscilloscope was on 2026-11-05.

How to repeat the checks: `PATH=~/miniforge3/envs/base-tools/bin:$PATH make test`, `make sim`, `make`, `python3 ../common/tests/verify_build.py --design counter`, then on the board `make install-service` and `make test-hw`.

2026-11-06: NEXT, GATE, one word, and awg's OUT1 as a detector

- **Setup:** the 2026-11-05 wiring plus awg's D5 → NEXT (E1 pins 14 → 8) and D7 → GATE (18 → 6), and awg's OUT1 cabled to IN1 (counter's OUT1 cable removed). Every wire checked first by static levels: each of awg's six outputs raised exactly its input and nothing else; O1/O4 to EVENT0 and C8; awg's DC offset ±0.20 V read ±0.198 V on IN1. C2 and C8 carried no signal in the tests below and counted nothing in any run (no crosstalk). Receipt: `data/board-1.0-awg-tests-2026-11-06.json`; the manual's figure `analog` and section "The tests of the next day".
- **NEXT** (per-NEXT points, 4 points × 5 shots with k + 1 pulses on C1 at point k, 20,000 sweeps): exact in all three runs (shots, n1, $\Sigma n1^2$). Double NEXT pulses 16 ns apart: one advance with the default 24 ns hold-off (rising and falling edges), two with 8 ns (points [200000, 0, 200000, 0] as predicted).
- **GATE** (an 80 ns GATE pulse, C1 pulses before, at, inside, after it; G1/G2 isolating the edge pulses, G3 all in GATE, G4 timed), 400,000 shots per setting: exact in all 27 settings (G4 = 7 per shot, the inner pulses always

in). W = 1, delay 0: the pulse at GATE's rise inside in 0.748 of the shots, the one at its fall in 0.804 (GATE's edges 0.25 and 0.80 ns after C1's: the pulse reads 0.55 ns long); any shift ≥ 1 ns: 0 and 1. Pulses 8 ns after GATE's edges flip exactly at a shift of 8 ns for W 1 + delay 8, W 8 + delay 1, W 4 + delay 5, W 3 + delay 6 (0.748, 0.809 each: the same skews): the shift is (W − 1) + delay. Active low: the complements.

- **One word:** SEQ a word before the first TRIG: no bad sweep in 49,999. SEQ and TRIG in one word: 1.1 % (falling edges) and 1.5 % (rising) of the sweeps had TRIG one slot before SEQ (each flags two sweeps; points 0 and 3 off by exactly that many). NEXT in the word of a point's first TRIG: exact (NEXT never later). NEXT, SEQ and TRIG in one word: 1 shot in 400,000 moved.
- **awg's analog pulses on C9** (a 16 ns-rise, photomultiplier-like shape; awg's D0 marking each pulse on C1): four heights 0.18–0.81 V give a staircase of 1, ¾, ½, ¼, 0 counts per pulse, each step at 1.06 × the nominal height, 0.02–0.04 V wide (the sampling phase). A photomultiplier-like spectrum (2048 pulses: 69 % around 0.45 V, the rest small noise pulses) gives the plateau curve and, by its derivative, the height spectrum. Time walk at 0.08, 0.15, 0.17 V: 8.2, 11.8, 13.6 ns from the 0.81 V to the 0.18 V pulse. NIM-like −0.75 V pulses at −0.40 V: every pulse counted (SHOT records: 1,467,300 shots × 4 pulses, each in its window); a positive threshold of +0.05 V counts each pulse's overshoot (1.00000 per pulse). C9 sees awg's analog pulses 33.5 ns after the digital marks awg emits with them (awg on nominal calibration, plus the cable).
- **`counters()` across channels:** C1 − C9 between consecutive readings scattered by ±10 counts (once ±170) at 1 pulse per µs: the channels' totals are read one after another (each one latched exactly). Ratios from two readings near a boundary therefore differ by a few counts (the first NIM ratios showed 2 and 10 "missing" in 2 million); RATE records hold one instant. PROTOCOL says so.

2026-11-05 (evening): tests on awg's edges with exact expectations

- **Setup:** as the afternoon's session (below), the oscilloscope unused; the boards' crystals unlocked (awg 5.02 ppm fast against counter, measured by the fits below). awg's patterns are sources whose every edge is known, so every count has an exact expected value, computed on the PC from THEORY's definitions (half-open gates, the dead-time rule, timetagger's clusters). Receipt: `data/board-1.0-awg-tests-2026-11-05.json`; the manual's figures `boundaries`, `delayscan`, `deadknown`, `rus` and section "The tests on awg's edges".
- **Random patterns (differential test):** 7 random patterns of 64 shots × 512 words (TRIG and SEQ idling high, on falling edges; random 8 ns pulses on C1, C2; G1–G4 with edges 4 ns off the 8 ns grid, G4 with a delay scan; D1 a count window, D2 an xor of two gates; unit 1 C1+C2 W = 12 ns, unit 2 W = 20 ns inside G1), 2000 passes each: 896,000 shots, 129 million edges, SHOT records on. Five patterns exact in every per-point sum (n1–n4, Σn1², D1, D2), every SHOT record, both units' 15 pattern totals, the channels' totals, no rejected edge. Two were not: a false **TRIG** edge 463 ns before point 45's real TRIG in 4 % of the passes (seed 4: the shot started early, the real TRIG fell into the 3 µs hold-off, nothing flagged), and a false **SEQ** edge in 1.4 % of the passes (seed 5: 52 bad sweeps, the points renumbered). Both came about 1 ns after D0 and D1 rose in the same word, at one place each of about 1000 such words per pass: jumper crosstalk on lines idling high. Every one of the two patterns' 256,000 SHOT records equalled the counts computed at its own TRIG time (the times from the records; 78 at false TRIG times), the renumbered points included: counter computed exactly what its inputs gave. With a 200 ns hold-off the same pattern gave 407 missed TRIGs and 340 bad sweeps (more false edges got through). The manual's hold-off and crosstalk sections now say so.
- **1 ns boundaries** (a 512 ns pattern with no two outputs changing in one word, 2,000,000 passes per setting, every count exact: shots = N, C1 total + rejected = 3N, C2 = 2N): gates stepped 1 ns per point around C1's and C2's edges; the windows starting at an edge count it in 48.0 % (C1) and 53.1 % (C2) of the shots and the adjacent windows ending there in exactly the others (n1 + n2 = shots at every point): the edges arrive 0.52 and 0.47 ns before TRIG + 64 ns (sub-ns from the fraction, the clocks being unlocked). Dead time 14, 15 ns: none of a pair 16 ns apart rejected; 16 ns: 10.6 %; 17, 18 ns: all. Unit 1 (C1@64, C2@80): W = 14, 15: 0; W = 16: 0.946 of the pairs (0.949 predicted from the two edges' offsets); W = 17, 18: all. Unit 2, W = 12, C1 delayed 2..6 ns: 0, 0, 0.946, 1, 1 (the delayed lane at 1 ns). Edge offsets of single pulses on one wire by the same method: ±95 ps depending on the pulse 8–16 ns before (ringing).
- **Latencies through O4 → C8** (TRIG from awg, 1 ns windows scanned over 512 points, 1,000,000 shots each): a channel's echo 92.5 ns after C1's edge (155–157 ns after TRIG); a herald 121.5 ns + W after the cluster's first event for W = 8, 16, 24, 32, 64, 128 ns (121.49–121.51), spread evenly over 8 ns; D1 pulse-true 104.5 ns after its gate's end.
- **Accidentals and a delay scan:** pairs (C2 40 ns after C1, 348 per pass) on independent backgrounds (4.7 and 4.8 Mevents/s), 400 passes per setting; unit 2 (W = 12) with C1 delayed 0..80 ns and 1 µs, unit 1 without delay at W = 8..128 ns: 61 of the 164 settings lie at least 2 ns from every window edge and all 61 were exact in all four pattern counts; the others lie between or near the bounds for C2 0 or ±1 slot late (the sub-ns offsets vary by ±0.1 ns with the pattern). Peak 419 per pass, accidentals 139 (144 from R1·R2 on the 8 ns grid): CAR 2.01.

- **Dead time against a known rate:** random patterns on C1, 1.27–28.3 Mevents/s, dead times 20, 52, 100, 500, 1004 ns, 1000 passes each: accepted = the pattern's simulation and accepted + rejected = its edges in all 25 runs. The Poisson correction $R = m/(1 - m\tau)$ is within 3 % at 1.27 Mevents/s and over-corrects by 14 % at 11.3 and up to 50 % at 28 Mevents/s (awg's grid and merging pulses: not a Poisson process).
- **Repeat until success** through awg (step 0: TRIG every 4.096 µs, jump on EVENT0; counter: the emitter on C1 at 1.09 Mevents/s, G1 96 ns, D1 = G1 ≥ 1 pulse-true on O1; step 1 answers on C2): 2,047,844 attempts, 204,167 successes, p = 0.09970 (expected 0.09976, −0.3 σ); every success answered once (G2 on C2), no failure answered; attempts per success geometric (χ^2 71.6 for 78 dof), memoryless; the attempt after a success 2608 ns after its TRIG (4096 after a failure). No record lost.
- **SHOT records to a PC** (Gigabit Ethernet, the timer's TRIGs, 5 s each): lossless at 0.25, 0.5, 1.0 and 1.5 million records/s (30 MB/s); at 2.0 million the FPGA's record FIFO dropped 2.4 % (the ring full: the network path's limit, about 39 MB/s).
- **Overnight run** (12 h, 2026-11-05 23:05 to 11:05; awg's sweep of 64 shots × 512 words, 164,794,921 passes exactly, no two outputs changing in one word; counter: four gates, two decisions, two coincidence units, SHOT records, RATE records every 1 s with C10 counting the GPSDO; the PC checked every SHOT and SWEEP record as it arrived and the cumulative per-point sums every minute (a bank swap each time), and read the die temperature every 30 s): 10,546,874,944 SHOT records (214 GB over the network, 244,000 per second), **none wrong and none lost** (frames' lost 0, the record FIFO's drops 0); 164,794,920 SWEEP records, none bad; 0 missed TRIGs; the per-point sums exact at all 720 minutes and at the end (164,794,921 shots per point); both units' pattern totals and the channels' totals exactly N × the pattern's; 0 rejected edges. `run_status()["shots"]` read 1,956,940,352: the run's shot count is 32 bits and wraps (now documented in REGISTERS and PROTOCOL; `records_dropped` is since loading, not cleared by a run).
- **The crystal against the GPSDO** (C10 in the same run: 43,207 intervals of exactly 1 s of the board's clock): the crystal runs 5.70 ppm slow on average, 5.76 in the first hour, 5.66 in the last, 5.66–5.77 ppm in minute averages; the die at 54.8–56.2 °C. No simple coefficient: a 0.8 K warming after 8 h slowed the crystal by 0.08 ppm at once, which recovered within about 1.5 h while the die stayed warm. Allan deviation of the board's time against the GPSDO $4.4 \cdot 10^{-8}$ at 1 s (one count of 10^7 in a second), $2.1 \cdot 10^{-9}$ at 54 s (minimum; MDEV $1.4 \cdot 10^{-9}$ at 24 s), $1.5 \cdot 10^{-8}$ at 4320 s. The manual's figure `crystal`.

2026-11-05: 1.0 with awg on a second board and an oscilloscope

- **Setup:** counter on 192.168.0.102 (build 0x894f27b0); awg 3.0 (0x97cda2f3, nominal calibration) on a second board, 192.168.0.104; Dupont jumpers of about 20 cm: awg's D0, D1, D2, D3 (DIO0..3_N) to C1, C2, TRIG, SEQ; counter's O1 (DIO4_N) to awg's EVENT0 (DIO4_P); O4 to C8 (DIO7_N to DIO7_P, neighbouring pins, 2 cm); the grounds joined; OUT1 cabled to IN1 (0.5 m) and a 10 MHz sine on IN2 as before. A Siglent SDS2504X Plus (1 GSa/s per channel, 10× probes, CH1 and CH2 of high bandwidth) read over SCPI; the probes on free pins carrying copies (awg's D6 with TRIG, D4 with its answer, counter's O2 with O1/O4), since a pin with a jumper takes no probe. Each wire was checked first by static levels both ways. Receipt: `data/board-1.0-awg-scope-session.json` (all numbers below, the histograms and the scope's acquisitions).
- **Exact counts:** awg's pattern (8 points per sweep, 512 ns apart: SEQ, TRIG, then k + 1 pulses of 8 ns on C1 and 8 − k on C2 at point k): 39,075,760 shots in 20 s, 4,884,470 at each point, every point's sums exact (G1 = (k + 1) × shots, G2 = (8 − k) × shots, Σ $G1^2$ exact: 176 million counts), no bad sweep, no missed TRIG. With point 5's TRIG left out every one of 1,222,473 sweeps was flagged.
- **Jumper crosstalk:** with TRIG's default hold-off (24 ns) the first run had about 3.5 extra TRIGs per second (2 per million TRIGs), each shifting the rest of its sweep's points (a few ppm off in the sums, 16 bad sweeps in 5 s). Varying the wires: TRIG and SEQ alone 0 extra, with C1's pulses 0, with C2's pulses 87 in 8 s (C2's wire runs beside TRIG's: awg pins 6 and 8), none with a TRIG hold-off of 450 ns: the edges were real, coupled in the jumpers. The README's hardware setup now says so.
- **Decision latency** (D1 = G1 ≥ 1 at G1's end, G1 [0, 100 ns)): `measure_latency(4, 8)` (timer TRIGs on the clock's grid): 101.54, 101.57, 101.59 ns. With awg's TRIGs on the pin (asynchronous) counter timed O4's edge at C8 201 … 209 ns after TRIG, evenly over the 8 ns word, 205.0 ns on average: **101 … 109 ns from the gate's end, 105.0 ns on average**. The scope (TRIG's copy on awg's pin to O2's 50 % point, 900 acquisitions): 204.5 ns (200.3 … 208.8), 103.5 ns after the gate's end at TRIG's pin with 1 ns for the jumper: within 1.5 ns of counter's own (the jumpers, the pins' skews and the thresholds). Simulated: 113.6 ns (no pin buffers); the plan's target was 75–100 ns. **Early mode:** O4 105.5 ns after the deciding photon (photon at 40.5 ns, O4 at 146.0 ns on average).
- **Preselection loop** (awg's sequence: a step starts the detection pattern, TRIG and one photon on C1, and waits for EVENT0, jumping at once to a step that pulses D1 into C2; 3.8 million shots each, D1 true in every shot, timed by counter's 1 ns gate scans and by the scope): awg's answer at C2 **567.8 ns** after TRIG with awg's digital outputs aligned with its analog outputs (on awg's 8 ns grid: 91 % at 568, 8 % at 560; scope

567.83 ns after correcting CH4's probe by its measured +1.09 ns), **455.8 ns** with awg's `digital.delay` −112 ns (scope 455.43 ns). So from the gate's end to awg's output **468 ns, or 356 ns** at awg's shortest delay; awg's part (EVENT0 to the output pin) 363 ns or 251 ns. With D1 early: the answer 468 ns after the photon. awg's marker 1 (the target step's start pulse) came within 0.3 ns of D4.

- **Wired checks** (`make test-hw` finds O4 → C8 and runs them: their first board run): C8 counts O4's 125 MHz pattern exactly on both edges and with 20 ns of dead time, the emitter lane's 4 ns pulses (0.4 % merging, as pulses within 2 ns do), D1's pulses one per shot, the latency, and **C1's echo, the herald of unit 1's clusters and C9's echo exactly** (up to 5 Mevents/s, none rejected). The first runs found three faults in `scripts/test_hw.py` itself, never run before: an undefined name, totals read while the emitter ran (read one register at a time, the source's and C8's totals were taken up to milliseconds apart: −1083, −6486, +245), and a ±0.01 bound on edges per shot that the windows' statistics (±0.025) break; all fixed. **Board test** with this wiring (`make test-hw` twice, the driver with the measured analog lag): **45 checks passed** both times (receipts `data/board-1.0-session-run1.json`, `-run2.json`).
- **Analog alignment:** one emitter event as an 8 ns pulse of 0.8 V on OUT1 (into IN1 by the 0.5 m cable, C9 at 0.25 V) and on O4 (into C8) and O2 (the scope). Counter's unit 2 (C8, C9, W 8 ns, one delayed in 1 ns steps) put C9 **6.6 ns** after C8 with Δ = 110 ns. The scope (OUT1's cable on CH3 through a T at 1 MΩ, as IN1; CH2's probe calibrated in the T against the coax: +7.04 ns) put OUT1's 0.25 V crossing at the cable's end **22.25 ns** after O2's edge, so 22.15 ns after C8's: counter timed C9 15.5 ns early. The analog lag constant in the driver (`A_LAG`, was the RTL's simulated 23.5 ns) is now the measured 7.95 ns: Δ = 95 ns on this board, and the scan then put C9 **21.70 ns** after C8 (−0.45 ns, ±1 ns: the probe's calibration, the pins' skews, the 1 ns scan; 0.24 ns of it Δ's rounding). The 15.5 ns (two clocks within the error) are not explained; THEORY 2.3 says where they likely are. Other pulse shapes (a fast NIM edge) cross the threshold differently between the samples: not measured.
- **Not measured:** a real detector; NIM pulses; awg's EVENT jitter beyond its 8 ns grid; the herald's and echo's latencies on the scope.

2026-11-04: 1.0, the release (plan step 5)

- **Board** (192.168.0.102, build 0x894f27b0: the FPGA of 0.4 with version 1.0; `make install-service`, then `make test-hw` twice: **36 checks passed** both times; receipts `data/board-1.0-run1.json`, `-run2.json`; OUT1 cabled to IN1, a 10 MHz reference on IN2): the same results as 0.4 (unit 1's patterns within 1.7 σ of the simulation over 13.6 million clusters, unit 2's within 0.05 σ; IN2's 10 MHz one count per period, +5.4 ppm; OUT1's pulses on IN1 99.84 % at 0.1 Mevents/s). All eleven presets loaded on the board through the service (each applied and read back); with C9 enabled the alignment is +110 ns on this board's calibration.
- **New in 1.0, host only**: the experiments' presets (`nv-readout`, `nv-trace`, `ion-detection`, `preselection`, `hbt`, `heralded-g2`, `bell`), `analysis.py` (contrast, weighted Levenberg- Marquardt fits of Rabi, Ramsey, exponential and Lorentzian curves, the two-Poisson mixture by EM, the threshold of the best fidelity, g2(0), Grangier's α, visibility, E and CHSH's S), `RECIPES.md`, the GUI's timing diagram, count distribution and coincidence table, THEORY's physics section, the site entry (`common/tests/test_site.py`) and the SD image's inclusion (`common/tests/test_sd_image.py`).
- **`tests/test_analysis.py`** (new, 8 tests): every fit recovers its parameters from noisy synthetic data within the expected errors (the Rabi fit's χ^2/dof within 0.5 of 1); the EM fit of a 6000 + 4000 shot mixture of Poisson 0.4 and 9 gives the means within 0.03 and 0.15; the threshold (3) and its fidelity equal the hand computation; g2, α, visibility and S of known counts; the RATE records' patterns in `rate_table`.
- **`make test`** (counter): 59 tests passed; `make sim`: the six benches passed (the RTL is 0.4's but the version comment). The common uniformity, site and SD-image tests pass.
- **`verify_build.py --design counter`** (an isolated source-only copy): all 8 stages passed; the copy's builds have the same ID as the working tree's, 0x894f27b0.
- **Build 0x894f27b0**: routed WNS +0.235 ns, TNS 0, WHS +0.017 ns; DRC: no errors (the DSP48 advisories and RTSTAT-10); 11,335 LUTs, 3,849 slices (87.5 %), 23 BRAM36, 40 DSP48: within the caps.
- **Not done yet**: the board session with awg on a second board and an oscilloscope (the plan's 1.0 session, with the user's OK): the loop times through awg's EVENT input, the decision latency and the early mode on a scope, a herald and C9's echo on wired outputs, the analog alignment Δ measured (a cable and a pin jumper together). No real detector was used in any test.

2026-11-04: 0.4, analog channels and coincidences (plan step 4)

- **Board** (192.168.0.102, build 0xc8bcfb61, `make install-service` with counter active, then `make test-hw` twice: **36 checks passed** both times; receipts `data/board-0.4-step4-run1.json`, `-run2.json`; OUT1

was cabled to IN1 and a 10 MHz reference sine was on IN2, no output wired back). Step 3's 28 checks as before, and:
  - C9 and C10 count the emitter's pulses (the discriminators never firing) as simulated (0.97842 and 0.95738 per event, simulated 0.97668 and 0.95735);
  - C9 on the idle IN1 (mean −0.5 mV, σ 1.5 mV): no count with the threshold 50 mV beyond the noise, 1.2 × 10^7 counts/s at it, both polarities (IN2 was not idle: its noise check was not run);
  - coincidences of the emitter's pairs (C2 copies half of C1's events 16 ns later) and an independent C3: unit 1's 7 exclusive patterns of C1, C2, C3 (W 20 ns, 13.5 million clusters) within 1.8 σ of a simulation of the same process with timetagger's cluster definition, and unit 2's of C1 and C2 delayed by 2 µs (the accidentals, 12.1 million clusters) within 0.3 σ;
  - RATE records with both units (43 words) every 10 ms through the ring pump: none lost, exact intervals, the patterns' rates as the totals' (C1 + C2 3.517 × 10^6 /s);
  - OUT1 → IN1: OUT1's pulses (0.8 V for 8 ns at emitter lane 1's events) at 0.1 Mevents/s counted by C9 above 0.25 V: 99.83 % and 99.85 %; at 1 Mevents/s 1.6 % are lost, as two pulses within about 16 ns merge in the converters' analog paths (the channel's pulse-pair resolution with these pulses);
  - IN2's 10 MHz: C10 at 0 V with a hysteresis of 0.2 V counts one per period, 100,000 or 100,001 in every 10 ms interval (10,000,055 Hz: the board's crystal at +5.5 ppm against the reference). The first run (`data/board-0.4-step4-attempt1.json`, 35 of 36) failed the OUT1 pulses at 1 Mevents/s against 99 % (98.4 %: the pulse-pair resolution above), now checked at 0.1 and reported at 1 Mevents/s. Before it, a run stopped: AN_THRESH reads back sign-extended and the driver wrote a negative code's 17 bits (fixed; the driver test's fake registers now read back as the RTL does, with a negative code). **Not measured on the board**: the analog alignment Δ (110 ns from the calibration; a cable and a pin jumper together), the herald and C9's echo on a wired output, the decision latency.
- **The model against the definitions** (`tests/test_model.py`, 26 tests; 4 new): both units' clusters equal timetagger's `coincidences` on the lanes' events (four lanes, W 20 ns and 13 ns, a lane delayed by 37 ns), also gated by G1 with TRIGs; the analog discriminator's events equal an independent reference (the Schmitt trigger, `ceil(8a/d) − 1` with exact fractions) at one fixed lag over pulses with random onsets, every slot; the coincidences' scenario covers RATE records with 0, 1 and 2 units, every pattern of both units, every analog slot.
- **The driver** (`tests/test_control.py`, 25 tests; 4 new): the analog channels (thresholds in volts through the calibration, polarity, hysteresis, a negative code) and the alignment (+24 ns with the nominal calibration on the pin channels, the gates and GATE), the units, the herald, the monitors' pulses compiled, derived and described; invalid settings refused; the pattern totals (exclusive and inclusive) and the samples; the driver's registers in the bit-exact model: unit 1 on the emitter's pairs and an independent lane against the clusters of the lanes' events, unit 2 with C2 2 µs later against the same events shifted.
- **`tb_coinc`** (new, bit-exact): pins on C1..C4, the emitter's pairs and lanes into C5, C6, C9, C10, ADC samples clock by clock (pulses with noise on both inputs, and 600 crafted crossings over every 1 ≤ a ≤ d ≤ 24: the interpolation's boundaries), thresholds and hysteresis changed while running, unit 1 gated by G1 and with other lanes and W, unit 2 delayed by 37 ns, then 5,000 ns, switched off and on, W up to 1023 ns; **269 reads and all 2,076 ring words (52 RATE records of 13, 28 and 43 words) equal the model's, and both discriminators' event words every clock (23,999)**. Checked to fail with a cluster's join at `s < rem`, with the pattern words' index off by one, with an interpolation step's ≥ as > (after the crafted crossings were added: noise alone never hit the boundary) and with the window's `W + s` in 10 bits (after the W = 1023 phase was added).
- **`tb_core`**: unit 1's heralds on O1 looped back into C8: 785 pulses for 785 clusters.
- **`make sim`** (the final sources): `tb_sysid`, `tb_core`, `tb_front`, `tb_shots`, `tb_sums`, `tb_coinc` passed.
- **Found and fixed**: switching RATE records off reset a record in progress, leaving a cut record in the FIFO (since 0.1; with 43-word records likely): a record now completes.
- **Timing and resources**: the first build (0x48d03122) failed (WNS −0.412 ns: unit 2's delayed lane put into the cluster logic in the same clock) and exceeded the LUT cap (11,533). Fixed over six builds without removing a feature: unit 2's lanes registered with the delayed lane put in one clock earlier (the same lag); the cluster's `rem ≥ 8` registered and the new cluster's from a per-slot mask; the monitors' shift in steps of 4; the decisions' windows on the counts saturated to 16 bits (as the SHOT records carry them: a PC recomputes every D) from copies the gates register; the rejected edges, the emitter's events and the decisions' counts in DSP48s (the last counted from the registered decision); the timer's `tidx == 0` registered. Builds 0xe5a9f609 (−0.422 ns), 0xb72aa99f (−0.414 ns, 92.5 % of the slices), 0x1b5a3590 (−0.072 ns), 0x465293ff (−0.140 ns), 0x381264be (+0.077 ns).
- **`make test`** (counter): 51 tests passed. The common uniformity tests pass.
- **`verify_build.py --design counter`** (an isolated source-only copy): all 8 stages passed; the copy's builds have the same ID as the working tree's, 0xc8bcfb61.
- **Build 0xc8bcfb61** (the one tested on the board): routed WNS +0.272 ns, TNS 0, WHS +0.021 ns, THS 0;

DRC: no errors (the DSP48 advisories and RTSTAT-10). Utilization: 11,347 LUTs (64.5 %; 356 as memory), 11,622 registers, 3,793 slices (86.2 %), 23 BRAM36, 40 DSP48, the common MMCM and one PLL: within the 1.0 caps (LUT ≤ 11,500, slices ≤ 3,960, BRAM36 ≤ 36, DSP ≤ 64). No feature of the plan's cut order was cut.

2026-11-04: 0.3, decisions (plan step 3)

- **Board** (192.168.0.102, build 0x8a6d58dc, `make install-service` with counter active, then `make test-hw` twice: **28 checks passed** both times; receipts `data/board-0.3-step3-run1.json`, `-run2.json`; no wiring). Step 2's 26 checks as before, and the decisions: the emitter at 40 Mevents/s on C1, shots from the timer every 2 µs over 4 points with SHOT records, G1 [0, 104 ns) and G2 [104, 200 ns) on C1; D1 = (G1 in 3 … 5), D2 = (G1 ≥ 5) xor (G2 ≥ 4) with pulses on true. In every one of the 100,000 records of each run D1 and D2 are what the record's counts give (0 wrong; D1 true 69,985 and 69,905 times, D2 47,757 and 47,789); the decisions' counters (taken 100,000 each, true as the records) and the per-point D1, D2 sums equal the records'. **Not measured on the board** (no output was wired back to an input): the decisions' pins, the channel echo and the decision latency (`test_hw` runs those checks, with `measure_latency`, when it finds a jumper from an output to a count input).
- **The model against the definitions** (`tests/test_model.py`, 22 tests; 3 new): D1, D2 of shots with random emitter counts against THEORY 2.7 computed from each SHOT record's counts (two windows, "a and not b", "a xor b" with pulses); the counters, the last value and the frozen bank's D1/D2 columns as the records; the pin's modes: pulses on true of 3 words (one per true decision), pulses on false, the level (it changes exactly where consecutive decisions differ) and early (one rise in each true shot, never after the decision, at a fixed lag of 11 words after the `low_a`-th counted edge); the sums scenario's swaps (below).
- **The driver** (`tests/test_control.py`, 21 tests; 4 new): the decisions and the outputs d1, d2, echo compiled, derived and described; invalid settings refused; the 16 logic names equal their tables; the counters; the driver's registers in the bit-exact model (D1 = G1 ≥ 5, D2 = G1 in 2 … 4 or G2 ≥ 1, against 400+ SHOT records); `DECISIONS?`, `LATENCY?`'s arguments.
- **`tb_shots`** (bit-exact): a fifth phase with decisions (D1 a level, D2 pulses, shots back to back so that a shot's last results come after the next TRIG): **68 reads and all 1,781 ring words (239 SHOT records with their D bits) equal the model's**. Checked to fail with a window's upper limit exclusive (ring words) and with the table's index swapped (`tb_core`).
- **`tb_core`**: D1 (G1 ≥ 1, pulses on true) on O1 looped back into C8: 8 pulses for the 8 true decisions of 12 shots; **simulated latency 113.6 ns** from a C1 edge in G1's last ns to O1's rising edge (121.6 ns before a decision's pin bypassed the outputs' register; the plan's target was 75–100 ns); C2's echo on O1: C8 counted all 804 of C2's counted edges.
- **`tb_sums`** (new; `ct_sums` alone, bit-exact against the model's `Sums`, 9,873 clocks): counts near 2^32 that the core's scenarios cannot reach: automatic swaps, saturation while the frozen bank is not released (flagged at the next swap), software swaps in the clock of a shot's result and in the clock after it, a clear in the clock of a result, a sweep-end swap, the zeroing, $n1^2$ with n1 at and above 2^24; every clock's swap state and 135 frozen words equal the model's. Checked to fail without the swap's and without the clear's zeroing of the totals' sums (the timing fix below).
- **`make sim`**: `tb_sysid`, `tb_core`, `tb_front`, `tb_shots`, `tb_sums` passed.
- **Found and fixed**: per-gate shot tags on the results were dropped: a mutation that ignored them passed, and the gates' results always come in the shots' order (a shot's last results by clock c′ + 4 of the next shot's TRIG, the next shot's first at c′ + 5 or later).
- **Timing closure**: the first build (0x146214d7) met timing but not the cap: WNS +0.059 ns (the sums' overflow check, a 33-bit add, the swap decision and the totals' enables in one clock, 14 levels; the shots' end word at +0.120 ns). Fixed: the totals' sums are formed in the clock of the shot's result and registered (a swap, a clear or a reset in that clock makes them the shot's counts: the same values, the model unchanged but its docstring); the gates' pairwise maxima are registered and `end_k` loaded one clock later, the done's countdown one clock after that (the done in the same clock: the model follows).
- **`make test`** (counter): 43 tests passed. The common uniformity tests pass.
- **`verify_build.py --design counter`** (an isolated source-only copy): all 8 stages passed; the copy's builds have the same ID as the working tree's, 0x8a6d58dc.
- **Build 0x8a6d58dc** (the one tested on the board): routed WNS +0.143 ns, TNS 0, WHS +0.034 ns, THS 0; DRC: no errors (the same DSP48 advisories and RTSTAT-10). Utilization: 10,117 LUTs (57.5 %), 11,199 registers, 3,568 slices (81.1 %), 22.5 BRAM36, 20 DSP48, the common MMCM and one PLL: within the 1.0 caps, 392 slices left to the 90 % cap for step 4 (the decisions took 160 LUTs; the rate monitors, 654 LUTs, are the first to slim).

2026-11-04: 0.2, shots (plan step 2)

- **Board** (192.168.0.102, build 0x6482d25d, `make install-service` with counter active, then `make test-hw` twice: **26 checks passed** both times; receipts `data/board-0.2-step2-run1.json`, `-run2.json`; no wiring). Step 1's 21 checks as before, and the shots: from the internal timer every 4 µs with its SEQs over 8 points, the emitter's states from a table (bright probability 0, 1/7 … 1), G1 counting C1 for 2 µs: each point's mean count (0.50 … 5.44 per shot) within 2.2 sigma of the table's per-word probabilities and the simulated counted fraction; all 502,396 shots in the per-point sums (three banks collected by the service's worker, swapping every second), none missed, none of the 62,799 sweeps bad; 20 software SEQs at random times: 20 sweeps flagged bad; the gate-delay scan (16 points, 48 ns windows stepped by 48 ns) along a bright state decaying with 510 ns: every point's mean within 1.7 sigma of the exact decay (the state's 11-word lag included); SHOT records through the service's ring pump at 500,000 shots/s: all 200,000, none lost, their counts equal to the per-point sums; at 1,000,000 and 2,000,000 shots/s (20 and 40 MB/s to a client on the board): no byte lost, none dropped. The first run stopped at the record-rate step: the driver refused timer periods below 512 ns (its minimum meant 64 clocks, not 64 ns; now 64 ns), and a check still expected register layout 2.
- **The model against the definitions** (`tests/test_model.py`, 19 tests; 8 new): from random pins on C1..C3 and TRIG, NEXT, SEQ pulses (hold-offs, TRIGs too close: missed), an event-level reference of THEORY 2.5/2.6 (the busy window, the point modes in slot order, the sweeps) gives the same shots (their times and points), the same missed TRIGs and sweeps (bad ones flagged), and each of four gates (timed, empty, the delay scan, a late one) counts exactly the lanes' events in its window; per shot, per NEXT and fixed points; the frozen bank's columns equal the SHOT records' sums (n1..n4, shots, $n1^2$ in 64 bits) after a software swap; a second swap waits for the release and the zeroing; a run started at a SEQ with a shots' limit (points, RUN records with their reasons); the emitter's states (a dark and a bright point); a gate with the GATE level (its minimum width, its delay).
- **`tb_shots`** (XSim, bit-exact): the pins of all twelve inputs (C1, C2; TRIG with missed TRIGs and bounces within the hold-off; NEXT, SEQ with NEXT and SEQ at the same slot and NEXT just after SEQ; GATE with glitches), the emitter's states on C3 (a table, a decay), four gates (timed, timed and GATE, the scan, empty), per-shot points with sweeps and swaps at their ends, software swaps and releases, a step changed while running, per-NEXT points started at a SEQ with a shots' limit, software NEXT and SEQ, a fixed point with software TRIGs, the timer with its SEQs and a duration limit, a clear, RATE records among the shots' records; **61 reads (counters, the gates' results, the frozen bank's words) and all 1,636 ring words (10 RUN, 217 SHOT, 51 SWEEP, 3 SWAP, 18 RATE records) equal the model's**. Checked to fail on a model with the gate's end mask wrong and with NEXT/SEQ ties in the other order (the scenario got the ties after the second mutation passed).
- **`tb_front`, `tb_core`**: passed on the new core (the emitter's probability now two clocks behind its register, the model too; register layout 3).
- **`make sim`** (the final sources): `tb_sysid`, `tb_core`, `tb_front`, `tb_shots` passed.
- **`verify_build.py --design counter`** (an isolated source-only copy): all 8 stages passed; the copy's builds have the same ID as the working tree's, 0x6482d25d.
- **Found in simulation and fixed**: `release`, `inside` are (System)Verilog keywords; an `always @*` block whose inputs never change does not run in simulation (its outputs stay X: the merger's read enables were X, so no record reached the ring), so the combinational logic is now functions with continuous assignments; the gates' register decode used address bits that were not the gate's index at 0x240 (the gates came out permuted); a 1-bit epoch per bank would have let a word of two activations before reappear: the frozen bank is now zeroed after its release instead.
- **Timing closure**: the first build failed (WNS −2.849 ns, TNS −1,725 ns): the bus's read mux, the gates' count paths (mask, popcount, a 32-bit add and saturation in one clock), the shots' timer compare in series with the TRIG's decision, the sums' overflow check, and the core's reset register (925 of the failing endpoints). Fixed: the read mux in groups over two clocks (no change to what a read returns), each gate's word count registered and added one clock later (the shot's done one clock later: c + end_k + 5), the sums' inputs registered (the update in e + 1 … e + 5), the timer's zero registered, an automatic stop one clock after its condition, the reset replicated (max_fanout 64); the model follows every change.
- **`make test`** (counter): 36 tests passed (model 19, driver 17: the controls, shots, gates and emitter states compiled and derived, invalid settings refused, run control, results collected from swapped banks and added, the emitter's table, the protocol's new commands). The common uniformity tests pass.
- **Build 0x6482d25d** (2026-11-04, the one tested on the board): routed WNS +0.196 ns, TNS 0, WHS +0.031 ns, THS 0; DRC: no errors (the DSP48 pipelining advisories 8 DPIP-1, 4 DPOP-1, 5 DPOP-2 on the emitter's products, the gates' d + p·s and the sums' square, as the other designs here have them, and RTSTAT-10). Utilization: 9,387 LUTs (53.3 %), 3,419 slices (77.7 %), 22.5 BRAM36 (the two banks' 18, the ring writer's 4, the emitter's table), 20 DSP48, the common MMCM and one PLL: within the 1.0 caps, leaving 541 slices to

the 90 % cap for steps 3 and 4 (the two rate monitors, 640 LUTs, are the first to slim if needed). Earlier builds of the step: 0x452f4d7c (WNS −2.849 ns) and 0xb622ad18 (−1.213 ns: the sweep logic's adder in series with the events' decisions, now parallel; the gates' end word, now one clock earlier; the emitter's DSP input, now registered); implementation with post-route phys_opt (AggressiveExplore), as the other full designs.

2026-11-04: 0.1, counting (plan step 1)

- **Board** (192.168.0.102, build 0x720e1f83, `make install-service` with counter active, then `make test-hw` twice: **21 checks passed** both times; receipts `data/board-0.1-step1-run1.json`, `-run2.json`). No wiring: all eight count inputs were quiet, no output was wired back (the loopback checks did not run). The emitter's four lanes at 2.5, 6.25, 12.5 and 25 Mevents/s binomial per word (|z| < 2.3 over 1.25 × 10^8 words); each channel's counted and rejected edges per event as a simulation of the same process (2 ns pulses, 8 ns dead time) to about 10^-3 (e.g. 0.91663 counted, sim 0.91727, at 25 Mevents/s); 100 ns of dead time at 62.5 Mevents/s: 8.939 Mcount/s, 0.14302 counted and 0.83351 rejected per event (sim 0.14305, 0.83363); pairs (half of lane 1 copied) and split (30 %) as set; **RATE records** through the service's ring pump every 10 ms (202 records, no gap, every interval exactly 1,250,000 words, the interval counts adding up to the totals, C1 at the simulated counted rate 6.114 M against 6.110 M) and every 10 µs (200,615 records in 2 s, none lost, none dropped); the alarm on at 2.5 Mcount/s below 3 M and off below 1 M; OUT1's monitor code within 0.2 window-sigma of 0.1 V + 1e-7 V per count/s of the measured rate (the volts at the connector not measured: no cable), OUT2 off at the calibrated 0 V (code −60); no record dropped over the whole test.
- **A bug found on the board** (first session, build 0xbff2c2c7, 19 of 20 passed: the RATE check of 10 µs failed; `data/board-0.1-step1-attempt2.json`): switching records off from a short interval dropped 18,000–29,000 records (and wrote some at a few clocks' spacing). The driver wrote RATE_PERIOD = 0 for records off, before RATE_CTRL = 0; in the 150–230 µs between the two writes the timer, once at zero, reloaded with 0 and fired every clock. Reproduced from the host; neither CPU or memory load on the board nor a stream client caused drops. Fixed twice: the driver keeps the period while records are off, and the hardware treats a period below 63 as 63 (at most one record per 64 clocks; the model and `tb_front`'s scenario too). The simulation had not covered it: its scenario never switched a short interval off. (A first board run had one more failure, a wrong expectation of the test: it compared C1's rate in the records with the emitter's rate, not with the counted fraction.)
- **Bit-exact model** `control/counter/model.py` (the core clock by clock with the RTL's registers) against the definitions (`tests/test_model.py`, 10 tests): the edges, the non-paralyzable dead time and the delays of eight channels on random pulses (glitches below 1 ns, bursts) equal an event-level reference (counted and rejected); falling edges and the emitter after the inversion; the emitter's lanes binomial per word with uniform slots, pairs (copies with their delay, the copy fraction) and split (never both, the fraction); RATE records equal the lanes' events below E + k exactly, consecutive differences over exact intervals; no records while the writer is off; the alarm and a monitor's code; the vectors current.
- **`tb_front`** (XSim, the real `ct_core` with the behavioural 1 ns sampler): 21,318 pin edges on C1..C8 (bursts within the dead times, glitches, one channel active low), all eight channels with their own dead time and delay, the emitter's four lanes into four channels (pairs with a delay, split; modes, probabilities and delays changed while running), a delay, a dead time and the enables changed while counting, RATE records in two periods (stopped and restarted), the alarm (a channel disabled for a while), both monitors; **80 register reads at exact clocks and all 858 ring words (66 RATE records) equal the model's**. The bench was checked to fail on a model changed by one clock (E), by 1 ns of dead time, or by <= for < in the emitter's trial.
- **`make sim`**: `tb_sysid`, `tb_core`, `tb_front` passed (also on the final sources).
- **`tb_core`**: the register interface, levels and sticky edges, a pattern and high impedance on the outputs (the real OSERDESE2/OBUFT models), an emitter lane on O1 looped back 0.5 ns later into C8 (272 pulses: counted plus rejected equal the pulses seen on O1), the rate alarm on O4 (on, then off).
- **The behavioural sampler** (`rp_pins1ns`, SIM only) took the first sample of a word at the clock edge where the pipeline read the previous word: a race between two always blocks. Its word is now complete 7 ns after the edge (no change to the hardware).
- **Common:** the records' hub `rpfpga.ringhub` and the PC's `rpfpga.client.RecordStream` (timetagger's, now shared; timetagger's `hub.py` and `client.py` subclass them; `common/tests/test_ringworker.py` 4 tests); timetagger's `make test` 64 passed (two hub tests now set the hub's frame and queue sizes on the instance).
- **`make test`** (counter): 25 tests passed (model 11, with the period's minimum; driver: compile/derive of channels, emitter, records, alarm, monitors in volts and outputs, invalid settings write nothing, safe, all presets, counters and rates; what the `selftest` preset's and a channel's registers do in the model; protocol; the client and RECORDS STREAM over localhost; a slow client loses whole frames; the worker drains a fake ring into the hub; loading through the fake PYNQ). The common tests: 345 ran, the one failure is `test_buffers`' known

reference-count assertion under Python 3.14.

- **Build 0x720e1f83** (2026-11-04, the one tested on the board): routed WNS +0.391 ns (within clk_125; the clk_125 → clk_250 crossing into the output serializers +0.551 ns), TNS 0, WHS +0.021 ns, THS 0; DRC: only RTSTAT-10. Utilization: 4,323 LUTs (24.6 %), 1,782 slices (40.5 %), 4 BRAM36 (the ring writer's queue), 10 DSP48 (the totals of C1..C8, the two monitors' products; C9/C10's idle totals are optimized away until step 4), the common MMCM and one PLL: within the 1.0 caps. Earlier builds of the step: 0xbff2c2c7 (WNS +0.169 ns: the crossing; a build before it had DRC DPOP-2 as well, the monitors' multipliers without the DSP48's MREG, now registered there) and 0x3b8b0ab0 (with the period's minimum; WNS +0.110 ns: Vivado had merged the four serializers' reset registers into one, whose net crossed the die through a LUT to the far serializer's tristate input; `rp_oserdes2ns` now registers the tristate in each instance, kept apart with dont_touch, so an output's enable acts one clock after its data).
- **`verify_build.py --design counter`** (2026-11-04, an isolated source-only copy): all 8 stages passed on the final sources (build 0x720e1f83, the copy's builds the same ID as the working tree's), as before on 0xbff2c2c7.

2026-11-03: 0.1, the skeleton (plan step 0)

- **Shared RTL** moved to `common/rtl/optional`: `rp_pins1ns` (1 ns samplers), `rp_oserdes2ns` (2 ns outputs with tristate), `rp_ring_writer` (timetagger's `tt_writer` with MAGIC/NOP as parameters), `rp_xoro`, `rp_count48`. The designs that had them keep their copies.
- **Common ring worker** `rpfpga.ringworker` (from timetagger's worker; timetagger's worker is now its subclass): `common/tests/test_ringworker.py` 2 tests passed; timetagger's `test_control.py` 29 tests passed unchanged.
- **`make test`** (counter): 8 tests passed (driver round trip, invalid settings write nothing, safe and presets, pins and hardware readings, protocol, client over localhost, loading through the fake PYNQ).
- **Every design's `make test`** after the shared changes (base-tools Python 3.14): ajku 132, analog_echo 18, atomic_clock 223, awg 83, counter 8, feedback 72, lockin 67, mult_add 121, synth 82, timetagger 64 tests passed (skips as before). The common tests: 343 ran, the one failure is `test_buffers`' reference-count assertion under Python 3.14, known before this change (it passes under 3.12).
- **`make sim`**: `tb_sysid`, `tb_core` passed (registers; the behavioural 1 ns sampler: a 3 ns and a 2 ns pulse seen, a held level, clearing; the real OSERDESE2/OBUFT models: 2 ns bits on an enabled output, high impedance on the others and after disabling; the ring writer's test words in order through an HP0 responder, the written position matching the bursts).
- **Build 0x161cd538** (2026-11-03, the placement check of the plan's step 0): all 12 ISERDESE2 in oversample mode placed, **also on both pins of the pairs DIO0..3 (P and N)**, which no other design here uses, and the 4 OSERDESE2 with tristate on DIO4..7_N (12 ILOGIC, 22 OLOGIC with the converters'). Routed: WNS +0.264 ns, TNS 0, WHS +0.024 ns, THS 0; DRC: only RTSTAT-10 (SmartConnect nets without loads, as in every design). Utilization: 1,381 LUTs (7.9 %), 2,193 flip-flops, 695 slices (15.8 %), 4 BRAM36 (the ring writer's queue), 0 DSP, the common MMCM plus one PLL (the samplers'). So the planned pin map stands (the fallback of the plan's risks, controls and outputs swapped, is not needed).
- **`verify_build.py --design counter`** (2026-11-04, an isolated source-only copy): all 8 stages passed (recipe build, unchanged, missing export, vendor HDL edit, the design's RTL edit, a failed build blocks the upload, clean, regeneration from the saved project); the regenerated build has the same ID, 0x161cd538.