# Synth + PnR Error Class Quick-Reference

1,349 words · ~6 min read

---

*VCA-CSA-101 + VCA-CSA-201 cross-chapter quick-reference handout. Anchors: §5.10.5 (sim-then-silicon discipline) + Lab 5.6 (sim toolchain pre-flight) + the Phase 2C KB doc-set ( + `kb/build-out/*` ). *

**Purpose:** the canonical catalog of synthesis + place-and-route + bitstream-generation error classes you will hit while bringing up Virtus HDL on real silicon, with a fix recipe for each. Print and pin during Labs 5.5-5.6, Labs 12.1-12.5, and any IP Pack integration round.

**Three toolchains addressed in parallel** (Vivado for Ultra96 dev-rig validation; yosys + nextpnr-himbaechel + gowin_pack for Tang Nano 20K production target; Gowin EDA as commercial backup). Most error classes manifest similarly across all three with tool-specific phrasing.

**Discipline rule** (also in `kb/build-out/sim-as-debugger-pattern.md`): **sim catches ~24-36× faster than silicon**. If a synth error arises, ask first: "did sim catch this?" If yes. Fix sim-side; resynth confirms. If no. Sim coverage gap; uplift sim before continuing.

---

## At a glance. Error class triage

| Error class | Where it surfaces | Cost to fix | Fix recipe section |
|---|---|---|---|
| Multi-driven net | Synth elaboration | 5 min | §1 |
| Latch inferred | Synth elaboration | 5-15 min | §2 |
| Missing sensitivity list | Synth elaboration | 5 min | §3 |
| CDC violation (report_cdc) | Synth post-elab | 30-60 min | §4 |
| Timing failure (negative slack) | Place-and-route | 30 min, 2 hr | §5 |
| BRAM inference fallthrough | Synth elaboration | 30-60 min | §6 |
| LUT/FF resource over-budget | Synthesis impl | 1-3 hr | §7 |
| AXI handshake hang at sim | Sim runtime | 5 min if you read KB first | §8 |
| Address decode silent drop | Sim post-write read-back | 15-30 min | §9 |
| FSM unreachable state | Synth | 5 min | §10 |
| Reserved-keyword collision | Synth elaboration | 5 min | §11 |
| Cocotb 2.x ReadOnly RuntimeError | Sim runtime | 5 min | §12 |
| Bitstream load fail (FPGA mgr) | Post-bitstream | 30 min, 2 hr | §13 |

---

## §1: Multi-driven net

**Error phrasing:**

- Vivado: `[Synth 8-690] multi-driven net 'X' on pin <module>/<port>`
- yosys: `ERROR: Multiple drivers found for net 'X' (driven from <module1> and <module2>)`

**Cause:** two `assign` statements OR two `always` blocks both write to the same wire/reg. Common pitfalls: copy-paste of an assign without renaming; resetting the same register from two `always` blocks.

**Fix recipe:**

```
1. grep -n 'assign\s\+X\s*=' <source>.v
2. grep -nE '^\s*X\s*<=' <source>.v
3. Delete or rename one of the duplicates.
4. Re-elaborate.
```

**Pre-emption:** at module top, `wire [N-1:0] X;` declarations should be unique; prefer single `always_comb` block per output.

---

## §2: Latch inferred (combinational always block missing default)

**Error phrasing:**

- Vivado: `[Synth 8-327] inferring latch for variable 'X'`
- yosys: `Warning: Latch inferred for variable 'X' from <module>:<line>`

**Cause:** combinational `always @(*)` (or `always_comb`) block doesn't assign `X` in every code path. Missing `default:` in case; missing `else` in if-else chain.

**Fix recipe:**

```
// BAD - latch inferred when sel is unknown:
always @(*) begin
    case (sel)
        2'b00: X = A;
        2'b01: X = B;
    endcase
end

// GOOD - default covers all cases:
always @(*) begin
    X = 0;              // default first; subsequent assignments override
    case (sel)
        2'b00: X = A;
        2'b01: X = B;
    endcase
end
```

**Pre-emption:** start every combinational block with default assignments, then override conditionally. Treat any "latch inferred" warning as an error. It's almost always a bug, not desired latching.

---

## §3: Missing sensitivity list

**Error phrasing:**

- Vivado: `[Synth 8-2766] sensitivity list of always block contains <signal>` (warning; usually still synthesizes)
- yosys: silent (yosys often promotes to `always @(*)` automatically)

**Cause:** old-style `always @(A or B)` sensitivity list missing some signal that's read inside the block. Sim mismatch with synthesis; latches inferred at synthesis.

**Fix recipe:**

```
// BAD - old style sensitivity list:
always @(A or B) begin
    Y = A & B & C;        // C missing from sensitivity list
end

// GOOD - let tool infer sensitivity:
always @(*) begin
    Y = A & B & C;
end

// BETTER (SV-2012):
always_comb begin
    Y = A & B & C;
end
```

**Pre-emption:** never use explicit sensitivity lists. Always use `always @(*)` or `always_comb`.

---

## §4: CDC violation

**Error phrasing:**

- Vivado: `[CDC-1]` through `[CDC-8]` warnings from `report_cdc`
- yosys: no built-in CDC report; verify in cocotb sim

**Cause:** signal generated by clock-A FF directly drives clock-B FF without a synchronizer. Multi-bit registers crossing domains without async FIFO + Gray pointers.

**Fix recipe:** see `kb/synthesis/clocking-cdc-pattern.md` for the canonical 2FF synchronizer + async FIFO patterns. Quick triage:

```
For 1-bit cross: insert 2FF synchronizer.
For multi-bit cross: replace with async FIFO + Gray-coded pointers.
For low-rate multi-bit: use req/ack handshake.
```

**Pre-emption:** annotate `(* ASYNC_REG = "TRUE" *)` on synchronizer FFs. Run `report_cdc` after every synth.

---

## §5: Timing failure (negative slack)

**Error phrasing:**

- Vivado: `WNS = -X.XXX ns` (Worst Negative Slack) at impl_1; failed timing
- nextpnr-himbaechel: `Max delay X.XX ns > Y.YY ns clock period`

**Cause:** combinational path between two FFs (or from input to FF, or FF to output) exceeds the clock period.

**Fix recipe** (in priority order):

```
1. Identify the failing path: report_timing -from <source> -to <sink>
2. Common fixes:
   a. Pipeline the path - insert a register stage in the middle.
   b. Reduce LUT depth - refactor wide ANDs / combinational decode trees.
   c. Reduce fan-out - duplicate driving FF; replicate slow inputs.
   d. Try one-hot FSM encoding (kb/synthesis/fsm-encoding-pattern.md §2.2).
3. If WNS only marginally negative (< 0.5 ns), try:
   - Vivado: switch impl strategy to "Performance_Explore" or
"Performance_NetDelay_low"
   - nextpnr: --placer heap --router router2 (slower but tighter)
```

**Pre-emption:** target ~80% of the clock period as the design's longest combinational path. Leaves headroom for impl variation.

---

## §6: BRAM inference fallthrough

**Error phrasing:**

- Vivado: `[Synth 8-2933] RAM not inferred for 'X' due to <reason>`
- yosys: `Warning: Block RAM inference failed for memory 'X' (using fallback distributed RAM)`

**Cause:** memory declaration violates BRAM inference rules. Async read on a sync-only BRAM, mixed-width ports without explicit width, write enable per byte without proper byte-write template, etc.

**Fix recipe:** see `kb/synthesis/ram-inference-canonical-patterns.md` (Phase 1) for canonical inference patterns. If inference can't be fixed, switch to explicit `xpm_memory_*` primitive per `kb/synthesis/xpm-memory-primitive-decision-guide.md` (Phase 2A). Pick from the 4 silicon-cert'd canonical patterns.

**Pre-emption:** declare memories using the canonical templates from the KB doc; verify inference success in synth log before assuming BRAM was used.

---

## §7: LUT/FF resource over-budget

**Error phrasing:**

- Vivado: `[Place 30-687] Insufficient resources to place: X LUTs required, Y available`
- nextpnr: `ERROR: Unable to fit design on device <part>`

**Cause:** total design exceeds the device's LUT4/FF/BRAM/DSP budget. On Tang Nano 20K (GW2AR-LV18 Arora II has 20,736 full user-logic LUT4 per D120 silicon-correction 2026-05-01. Earlier docs framed this as "GW1NSR-LV4 with ~5K effective" which was a Cat 1 mis-identification confused with Tang Nano 4K). Tang Primer 25K (GW5A-25 Arora V) has ~23K LUT4. Both Tangs have comparable headroom for M0-2 designs.

**Fix recipe:**

```
1. Run utilization report:
   - Vivado: report_utilization
   - yosys: synth_gowin -top X (utilization in log; or use abc_dff_log for FF
count)
2. Identify the largest consumers (top-down by module).
3. Apply one of:
   a. Use BRAM instead of LUTRAM for any > 1 KB memory (saves ~600 LUT4 per KB).
   b. Switch to one-hot FSM only for >8-state FSMs; binary for ≤4 states.
   c. Reduce parameterized widths where possible.
   d. Defer or scope-cut a peripheral (drop HDMI tile-map → minimal text-mode;
defer audio).
```

**Pre-emption for Tang Nano 20K:** see `kb/build-out/tang-dual-target-portability-pattern.md` for the LUT4 budget fit analysis. M0-2 baseline (~2.6K LUT4) fits ~12.5% of GW2AR-LV18's 20,736 LUT4 fabric. Comfortable headroom (D120 silicon-correction 2026-05-01; was previously framed as ~52% of a wrong-silicon ~5K budget).

---

## §8: AXI handshake hang at sim (cocotb)

**Error phrasing:** cocotb test hangs at `axi_write(...)` waiting for AWREADY+WREADY both high. No traceback; test eventually times out.

**Cause:** AXI4-Lite slave has registered (not combinational) AWREADY/WREADY/ARREADY. See `kb/build-out/axi4lite-handshake-canonical.md` §1.

**Fix recipe:**

```
// Replace:
output reg s_axi_awready;        // ← BAD: registered
// With:
output wire s_axi_awready;       // ← GOOD: combinational
assign s_axi_awready = (w_state == W_IDLE);
```

Drop the registered handshake assignment from the FSM `always` block. Same fix for WREADY and ARREADY.

**Pre-emption:** start every new AXI slave from the canonical skeleton in `kb/build-out/axi4lite-handshake-canonical.md` §2. Validated 5-of-5 across R1T9.5 + 4 M0-2 IP Pack rounds.

---

## §9: Address decode silent drop

**Error phrasing:** no synth error; cocotb test passes individual writes (BVALID always asserts) but read-back returns wrong value or zeros for a subset of addresses.

**Cause:** AXI register-map region is not power-of-2 aligned at its size. See `kb/build-out/axi-address-range-decode.md` §1.

**Fix recipe:**

```
1. List all regions in the slave's register map with (BASE, SIZE).
2. For each, verify: (BASE & (SIZE - 1)) == 0.
3. If misaligned, relocate to next aligned boundary.
4. Update decode mask to single-bitmask: waddr[N-1:M] == BASE_HI.
5. Update PORT_DELTAS.md or equivalent forensic doc.
```

**Pre-emption:** every AXI register-map design starts with the §4 alignment pre-flight checklist. Validated in M0-2 manifold (16-bit aligned per slave).

---

## §10: FSM unreachable state

**Error phrasing:**

- Vivado: `[Synth 8-3848] Net 'X' has unreachable state encoding`
- yosys: `Warning: FSM 'X' has N unreachable states; remove with -dont_care`

**Cause:** FSM state register is wider than `log2(N_states)` and some encodings aren't covered in the case statement.

**Fix recipe:** add `default:` clause that recovers to a safe state:

```
case (state)
    S_IDLE:  // ...
    S_WORK:  // ...
    S_DONE:  // ...
    default: state <= S_IDLE;     // ← REQUIRED
endcase
```

**Pre-emption:** see `kb/synthesis/fsm-encoding-pattern.md` §4, every FSM has explicit reset state + `default:` clause.

---

## §11: Reserved-keyword collision

**Error phrasing:**

- iverilog (cocotb's default):
  `<file>:<line>: error: Reference to '<keyword>' is illegal in this context`
- Vivado XSim: `[Synth 8-2715] syntax error near '<keyword>'`

**Cause:** identifier in source collides with SV-2012 reserved keyword (`null`, `event`, `time`, `string`, `assert`, etc.) when source is compiled under `-g2012` (cocotb default; Vivado default).

**Fix recipe:** see `kb/build-out/sv-reserved-keyword-pattern.md` §1.3. Rename the identifier with a `_src` / `_val` / `_sig` suffix; record in PORT_DELTAS.md.

```
# Pre-port grep:
grep -nwE '\b(null|event|time|string|assert|byte|bit|int|void|this|priority|unique|
final|local)\b' \
    src/*.v hdl/*.v
```

**Pre-emption:** the §3.1 pre-port grep on every third-party HDL component before pulling into cocotb harness.

---

## §12: Cocotb 2.x ReadOnly RuntimeError

**Error phrasing:**

```
RuntimeError: Attempting settings a value during the ReadOnly phase.
  at line: dut.<signal>.value = <expr>
```

**Cause:** test code uses `await ReadOnly()` followed (within the same coroutine OR via composition) by `dut.X.value = Y`. cocotb 2.x strictly forbids signal-set during ReadOnly phase.

**Fix recipe:** see `kb/build-out/cocotb-2x-readonly.md` §1.3. Drop the `await ReadOnly()` call; use `await RisingEdge(dut.clk)` alone. Post-edge values are settled in cocotb 2.x.

**Pre-emption:** the `kb/build-out/cocotb-2x-readonly.md` §3.2 migration grep on legacy cocotb 1.x test code.

---

## §13: Bitstream load fail (FPGA mgr)

**Error phrasing (Ultra96 PYNQ):**

```
RuntimeError: Failed to load bitstream onto fabric
fpga_mgr: state=2 (operating), bitstream loaded but ip_dict empty
```

**Error phrasing (Tang Nano 20K openFPGALoader):**

```
Programming Failed: USB error
or: Bitstream loaded but device not responding to JTAG
```

**Cause classes:**

1. Bitstream targets wrong device (Tang Primer 25K bitstream loaded on Tang Nano 20K, etc.).
2. Bitstream corrupted at staging (incomplete copy).
3. PYNQ Overlay() doesn't load the .hwh sidecar (Ultra96).
4. USB connection unstable (intermittent enumeration).

**Fix recipe:**

```
1. Verify device match: openFPGALoader --detect → expected JTAG ID
   (Tang Nano 20K: 0x0001081B; Tang Primer 25K: 0x0001A001)
2. Re-stage bitstream with byte-level integrity check:
   md5sum bitstream_in_workspace.fs
   md5sum bitstream_on_target.fs
3. Ultra96: verify .hwh sidecar present alongside .bit:
   ls overlay.bit overlay.hwh
4. Retry programming with -vvv verbose mode for openFPGALoader.
5. Power-cycle the board if USB enum is intermittent.
```

**Pre-emption:** see `kb/build-out/integrated-bitstream-checklist.md` (Phase 1), 8-stage canonical sequence with exit gates for each stage. Stage 6 covers staging integrity; stage 7 covers PYNQ overlay load.

---

## §14: When to escalate to KB doc authoring

If you hit a synth/PnR/bitstream error class **not covered in this handout**, the discipline rule is:

1. **Search KB first**: `kb/synthesis/` + `kb/build-out/` + `kb/gowin/` + `kb/integration/`.
2. **Search references**: `references/UG901-vivado-synthesis.md`, `references/yosys-synthesis-flow.md`, `references/PG058-xpm-memory-library.md`.
3. **If neither covers it AND the fix takes >30 min OR >2 iterations**, you have a NEW canonical bug-class. **Stop and write the KB doc first** (per `kb/build-out/integrated-bitstream-checklist.md` discipline rule). Phase 2C produced 9 docs from a single 's iteration arcs; the discipline scales.

---

## §15: Cross-references

- `kb/synthesis/fsm-encoding-pattern.md`, §10 deeper coverage
- `kb/synthesis/clocking-cdc-pattern.md`, §4 deeper coverage
- `kb/synthesis/ram-inference-canonical-patterns.md`, §6 deeper coverage (Phase 1)
- `kb/synthesis/xpm-memory-primitive-decision-guide.md`, §6 deeper coverage (Phase 2A)
- `kb/build-out/axi4lite-handshake-canonical.md`, §8 canonical pattern

- `kb/build-out/axi-address-range-decode.md`, §9 canonical pattern
- `kb/build-out/sv-reserved-keyword-pattern.md`, §11 canonical pattern
- `kb/build-out/cocotb-2x-readonly.md`, §12 canonical pattern
- `kb/build-out/sim-as-debugger-pattern.md`, Phase 2A doctrine; sim catches faster than silicon
- `kb/build-out/integrated-bitstream-checklist.md`, Phase 1; 8-stage post-bitstream discipline
- `kb/build-out/tang-dual-target-portability-pattern.md`, Phase 2C; Tang Nano 20K + Tang Primer 25K dual-target discipline
- `kb/gowin/yosys-040-upgrade-pattern.md`, Phase 2B; yosys 0.40+ for >4K-word BRAM elaboration
- `references/UG901-vivado-synthesis.md`, Vivado synth attribute / message reference
- `references/yosys-synthesis-flow.md`. Yosys synth flow + invocation
- `cross-chapter-sim-toolchain-quick-ref.md`, Phase 2A; sim toolchain handout (this handout's complement on the silicon-side)

---

*Synth + PnR error class quick-reference (Phase 2C) 2026-04-30. 13 error classes catalogued from 8+ months of Virtus build-out iteration arcs (R1C-CPU.1 through M0-2 IP Pack sim-cert). Every class has a fix recipe + pre-emption rule + KB cross-reference. Pin during any HDL lab on a real silicon target.*

---