

Peer Review

Review of: "DRC-Coder: Automated DRC Checker Code Generation Using LLM Autonomous Agent"

Evgenii Arsentev¹

1. Independent researcher

Summary

The paper tackles a real and unglamorous problem: every new technology node forces engineers to hand-write an integrated (grid-based) DRC checker that agrees with the signoff commercial tool, and that takes weeks of reading foundry documents and debugging. DRC-Coder automates it with two GPT-4o agents in an AutoGen group chat — a Planner that turns the foundry rule description plus layout images into grid-domain conditions, and a Programmer that writes the Python `drc` function — plus three tool functions (Foundry Rule Analysis, Layout DRV Analysis, DRC Code Evaluation, Section 4). The third one is the engine of the method: it runs the candidate code over a dataset of 207 NVCell standard-cell layouts (Section 3.1) and reports precision, recall, F1, and a summary of false positives/negatives back to the agents, which then iterate. The headline claims are F1 = 1.000 on all seven design rules versus F1 = 0.631 for standard prompting, 2.3 debugging iterations, and 210 seconds per rule (Section 5.2). The framing and the engineering are good, and the domain-specific evaluation function is the part I would keep. The evidence, as reported, does not yet support the strength of the claims — mainly because the loop that produces the code optimizes directly on the metric that is then reported, and because the version I am reviewing does not contain the main results table at all.

A note on the version I received

I am reviewing exactly the text as it appears on the Qeios article page. In that rendering, **the body of Table 1 is missing**. The caption for Table 1 is present (immediately before Section 5: "Performance evaluation of DRC code generation using standard prompting and our DRC-Coder with GPT-4o and Llama3 across seven design rules... Precision (P), Recall (R), and F1 score (F) for each method... the

number of debugging iterations and runtime in seconds are also included"), but no numeric rows follow it. The only numeric block in the article — the one running from the header lines "LLM / GPT-4o", "Design rules / Multi-agent / Single-agent", "w/o vision capability / w/ vision capability", "P R F P R F" down through the Average row — has six metric columns labelled with the two ablation variants and is immediately followed by the Table 2 caption. So that block is Table 2, and the primary comparison table is simply absent from this version.

Consequence: *every headline number in this paper is unverifiable in the submitted version.* F1 = 1.000, F1 = 0.631, 2.3 iterations, 210 seconds, the 37% figure, the Llama3 42.2% figure — none of them can be checked against a table, because the table that holds them is not there. I will not invent values for it. Restoring Table 1 is the first thing to fix, and until it is restored, no reviewer can validate the central claim. Everything below that touches Table 1 numbers is therefore based on the running text of Section 5.2 and Section 5.3 only, and I flag it as such.

Does the claim hold on the data

Three problems, in descending order of severity.

1. The evaluation set is the optimization set. Section 4.3 states that DRC Code Evaluation "executes our generated code on cell layouts in the dataset and directly compares the code output with the golden DRC reports of the commercial tool," and Section 4 states that the process runs "until the DRC reports are aligned with ground truth DRVs of the commercial DRC tool." Section 5.1 then evaluates on "207 standard cell layouts and covers 7 distinct design rules." There is no held-out split anywhere in the paper. The debugging loop terminates when the reported metric reaches its ceiling, and the reported metric is that same ceiling. Under that protocol, F1 = 1.000 is closer to a description of the stopping rule than to a measurement of generalization. It is also worth noting that Section 4.1 puts "two layout examples that has the target DRVs from our dataset" directly into the prompt, so evaluation layouts appear in the input as well. The fix is straightforward and does not require new infrastructure: partition the 207 layouts into a debug set and a held-out set, let the loop see only the debug set, and report F1 on the held-out set. If the number stays at 1.000, the claim becomes far stronger than it is now. If it drops, that is the interesting result.

2. The baseline runs on a different compute budget. Section 5.1 (Baselines) defines standard prompting as "using prompt in Figure 6 to directly generate the code without tool function feedback" — one shot, no execution, no feedback. DRC-Coder gets 2.3 debugging iterations with execution feedback over 207

layouts (Section 5.2). Comparing 1.000 against 0.631 across that gap does not isolate the contribution of the multi-agent design or of vision; it mostly measures "with execution feedback" versus "without." Two baselines would close this: (a) standard prompting sampled k times with the same total number of LLM calls, reporting best-of- k and pass@1 separately; (b) a single non-specialized agent with the DRC Code Evaluation loop but without the Planner/Programmer split and without the vision tools. The second is the decisive one.

3. $n = 7$ rules, one run each, no dispersion of any kind. There are no repeats, no seeds, no confidence intervals, no variance across runs, and the word "significantly" appears in the abstract and in Section 6 with no test behind it. With seven rules and LLM sampling in the loop, a second run of the same pipeline could plausibly produce a different row. The ablation is thinner still: in Table 2, four of the seven rows (M1.S.2, VIA1.S.1, M2.S.1 in both variant columns, plus M1.S.1 in the single-agent-with-vision column) are already at 1.000/1.000/1.000, so the entire discriminating signal for the ablation comes from at most three rules — M0.S.1, M0.S.2, VIA0.S.1. Attributing a design choice to a three-rule difference with a single run each is not something the data can carry.

And the ablation does not point where the text says it does. Compare the two variant columns of Table 2 row by row. Row M0.S.2: multi-agent without vision gives P/R/F = 0.929/1.000/0.956, single-agent with vision gives 0.530/1.000/0.661. Row VIA0.S.1: 1.000/1.000/1.000 versus 0.660/1.000/0.769. But row M1.S.1 flips: 0.848/0.880/0.844 for multi-agent without vision against 1.000/1.000/1.000 for single-agent with vision — dropping the Planner *improves* that rule. Row M0.S.1 also favors the single agent (0.944/0.965/0.946 versus 0.951/0.657/0.747). The two variants land at Average F1 0.935 and 0.911 respectively (Table 2, Average row), which is a 0.024 gap on seven rules with no error bars. Section 5.3's conclusion that the results "indicate the importance of visual capability" and "the advantage of our multi-agent setting" is not supported by those rows; what the rows show is that both variants, each of which retains some form of evaluation feedback, sit far above the claimed standard-prompting average of 0.631 and close to each other. That is evidence for the debug loop, not for vision or for role decomposition.

The percentages in Section 5.2 and Section 5.3 do not reconcile with each other. Section 5.2 says "DRC-Coder achieves 37% higher F1 score"; from the text's own numbers (F1 1.000 against 0.631) that is +36.9 percentage points, or +58.5% relative — so "37%" is a percentage-point difference labelled as a percentage. Section 5.3 then says the two variants are "32.5% and 30.7% higher F1" than standard prompting. Under the same percentage-point convention, and using the Average row of Table 2 (0.935 and 0.911), the differences are 30.4 and 28.0 points; under a relative convention, they are 48.2% and 44.4%.

Neither reading yields 32.5 and 30.7. I cannot tell whether the error is in the text, in a column of the missing Table 1, or in my reading of which baseline is meant, and I will not guess — but the three figures as printed are mutually inconsistent, and they need to be recomputed and stated with an explicit convention ("percentage points" or "relative"). The same applies to the Llama3 "42.2%" in Section 5.2, for which no Llama3 values are visible anywhere in this version.

The ground truth is itself a derived artifact, and it is not validated. Section 3.2 builds the golden labels by running the commercial DRC tool, then "identify[ing] the layout components that intersect with the DRV polygons" and treating those components' grid coordinates as ground truth. That polygon-to-grid projection is a heuristic. A DRV polygon can overlap components that are not actually the offending pair, and can miss a component whose relevant edge falls outside the marked region. Every P/R/F number in the paper is measured against this projection, not against the commercial tool's output. Section 6 nevertheless states that "the generated DRC checker successfully replicates the report of the commercial tool, providing signoff DRC to the layout tool" — that is a stronger claim than the experiment supports, and "signoff" is a loaded word in this field. Either validate the projection (e.g., have an engineer audit a sample of converted DRVs and report the agreement rate) or soften the wording to "replicates the grid-projected report".

The time saving excludes the pipeline's own setup cost. The abstract and Section 6 contrast "four minutes per design rule" with weeks (Section 1, Section 5.2) or days (Section 6) of manual effort — note that the human baseline itself is stated as "several weeks" in Section 1 and as "days" in Section 6, and is never measured. More importantly, the four minutes covers only code generation. Producing the dataset requires 207 layouts run through the commercial tool, and Section 2.3 gives the commercial tool as 215 seconds for a cell with 22 devices. Taking those two stated numbers at face value, that is roughly twelve hours of commercial-tool runtime alone, before the polygon-to-grid conversion and the grid visualizations of Section 3.2 — and it has to be redone for each new node, which is exactly the recurring cost the paper sets out to remove. I am presenting this as an order-of-magnitude estimate derived from the paper's own two figures, not as a measurement. Please report the end-to-end setup cost alongside the per-rule number.

Finally, on the VLM comparison in Section 2.2 and Figure 2: it is one design-rule image and one layout image, GPT-4o against Phi-3. The text calls it an experiment ("Based on this experiment, we observe.."). It is a motivating anecdote, and it should be labelled as one — it cannot support a claim about VLM capability in this domain.

Reproducibility

Section 5.1 does one thing right that most LLM papers get wrong: it names the model snapshot, "GPT-4o using the OpenAI API version 2024-05-13". Credit for that. Almost everything else needed to rerun this is missing.

Not reported: temperature, top_p, or any decoding parameters for either agent or for the VLM calls; random seed; the AutoGen version (Section 5.1 names the toolkit but not the release, and its group-chat behavior has changed substantially between versions); the maximum number of debugging iterations, and whether any rule hit that cap; the Llama3 variant and size (Section 5.2 cites [23], the Llama 3 herd report, which covers 8B through 405B) and how it was served; the Phi-3 variant used for Figure 2 — reference [7] is the general Phi-3 report, and image input implies a vision variant that is not named; token counts or API cost per rule; and the prompt text itself beyond the schematic in Figure 6.

Dataset and code are not released. Given a sub-3nm node, foundry documentation under NDA, and NVCell being proprietary, I do not expect the rules or the layouts to be published, and I would not hold that against the paper. But the framework itself is separable from the protected data. Releasing the agent harness, the prompt templates, the DRC Code Evaluation implementation, and a small synthetic rule set with synthetic layouts would let others reproduce the *mechanism* even if they cannot reproduce the numbers. Without something along those lines, nothing here is externally checkable.

Two further gaps that affect interpretation more than replication. First, the aggregation level of P, R, and F1 is never defined. Section 4.3 says they are measured "between the DRVs detected by the commercial tool and our generated code," but it is not stated whether that is micro-averaged over all grid instances in all 207 layouts or macro-averaged over layouts. This is not a pedantic question because the reported numbers are not internally consistent with a single aggregation: in Table 2, row M0.S.1 under multi-agent without vision gives $P = 0.951$ and $R = 0.657$, whose harmonic mean is 0.777, while F is printed as 0.747; row M1.S.1 in the same column gives 0.848 and 0.880 (harmonic mean 0.864) against a printed F of 0.844; row M0.S.2 under single-agent with vision gives 0.530 and 1.000 (harmonic mean 0.693) against a printed 0.661; row VIA0.S.1 in that column gives 0.660 and 1.000 (harmonic mean 0.795) against a printed 0.769. The direction is consistent, so I assume F1 is averaged per layout while P and R are aggregated differently, or all three are averaged per layout — but the reader should not have to assume. State the formula. The Average row, for what it is worth, is a plain arithmetic mean over the seven rules in every column; I checked all six, and they match, so that part is clean.

Second, and related: with per-layout averaging, layouts containing zero ground-truth DRVs for a given rule have undefined precision and/or recall. How are they handled — excluded, or counted as 1.0? If excluded, the averages are taken over a subset that varies by rule, which is the classic "average over survivors" problem and would need to be disclosed. The paper also never reports class balance in numbers: Section 4.3 says the dataset is "inherently imbalanced, with DRC-violating grids being significantly less than compliant ones," but no counts are given anywhere. A rule with three positive instances reaching $F1 = 1.000$ means something very different from a rule with three hundred.

Specific points

1. **Restore Table 1.** In the version on the article page, the Table 1 caption is present but no data rows follow it; the only numeric table in the article is Table 2 (six metric columns, headers "Multi-agent / w/o vision capability" and "Single-agent / w/ vision capability"). Every primary claim — $F1 = 1.000$, standard prompting $F1 = 0.631$, 2.3 iterations, 210 seconds, the Llama3 results — rests on a table that does not render. This alone blocks acceptance.
2. **Add a held-out evaluation.** Section 4.3's DRC Code Evaluation optimizes against the golden DRVs of the same 207 layouts (Section 3.1, Section 5.1) that Section 5.2 reports on, and Section 4 terminates the loop when alignment is reached. Split the layouts, debug on one part, report on the other. Also state whether the two in-prompt layout examples (Section 4.1) are drawn from the debug split only.
3. **Add an equal-budget baseline.** Standard prompting as defined in Section 5.1 gets a single shot with no feedback. Report (a) best-of- k standard prompting at matched LLM-call count, and (b) a single generic agent with the DRC Code Evaluation loop but no Planner/Programmer split and no vision tools. Without (b), the paper cannot separate the effect of the architecture from the effect of the feedback loop.
4. **Reconcile the percentages.** Section 5.2's "37% higher $F1$ " is a percentage-point difference ($1.000 - 0.631 = 36.9$ points; relative improvement is 58.5%). Section 5.3's "32.5% and 30.7% higher $F1$ " match neither convention against the Table 2 Average row values of 0.935 and 0.911 (which give 30.4 / 28.0 points, or 48.2% / 44.4% relative). Recompute all of them and state the convention explicitly. Same for the Llama3 "42.2%" in Section 5.2.
5. **Define the P/R/ $F1$ aggregation, and reconcile Table 2 internally.** Four rows of Table 2 print an F that is 0.02–0.03 below the harmonic mean of the P and R on the same row (M0.S.1 and M1.S.1 under multi-agent without vision; M0.S.2 and VIA0.S.1 under single-agent with vision). This is explicable

under per-layout macro-averaging, but the paper never says which averaging is used, and a reader checking the table will read it as an arithmetic error. Add the formula to Section 4.3 and state how layouts with no ground-truth DRVs for a rule are treated.

6. **Soften or substantiate the "replicates the commercial tool / signoff DRC" claim** (Section 6). The comparison target is the grid projection of Section 3.2, not the commercial tool's report. Report a validation of that projection — a manual audit of a sample of converted DRVs, with an agreement rate — or restate the claim in terms of the projected labels.
7. **Report the end-to-end cost, not just the per-rule cost.** The "four minutes per design rule" (abstract, Section 6) excludes dataset construction: 207 layouts (Section 3.1) through the commercial tool, which Section 2.3 clocks at 215 seconds for a 22-device cell, plus the polygon-to-grid conversion and grid visualizations (Section 3.2). By the paper's own two numbers, that is on the order of half a day of tool runtime per node, before any human time. Report actual measured setup cost. Also reconcile the human baseline, given as "several weeks" in Section 1 and Section 5.2 but "days" in Section 6, and say whether it is an estimate or a measurement.
8. **Report the iteration distribution and any non-convergence.** Section 5.2 gives an average of 2.3 debugging iterations over seven rules. Give the per-rule counts, the iteration cap, and whether any rule failed to converge and was excluded. If nothing was excluded, say so — with F1 = 1.000 as the headline, readers will want that stated.
9. **Report dataset statistics per rule.** Section 4.3 asserts imbalance without numbers. Add a table: for each of the seven rules, the number of layouts containing that rule's DRVs, the number of positive grid instances, and the number of negatives. This determines how much weight a per-rule F1 can bear.
10. **Add repeats.** Run the full pipeline at least three to five times per rule with different seeds and report the spread. Absent this, "robustness" (Section 5.2) and "significantly" (abstract, Section 6) should be dropped from the text.
11. **Rewrite the ablation conclusion in Section 5.3 to match Table 2.** The row-level evidence is mixed: removing the Planner helps M1.S.1 (0.844 → 1.000 F) and hurts M0.S.2 (0.956 → 0.661) and VIA0.S.1 (1.000 → 0.769); four of seven rows are saturated at 1.000 in both variants. The honest statement is that the two ablations are within noise of each other on Average F1 (0.935 vs 0.911) and that the experiment cannot attribute the gain to vision or to multi-agent decomposition. Also report iterations and runtime for the two variants — without them, the ablation compares systems at unknown and probably different budgets.

12. **Label Figure 2 as an illustration, not an experiment** (Section 2.2). One image per model is not a basis for the conclusion that "the VLM, especially GPT-4o, can help in explaining complex design rules and layouts". Either scale it up to the full rule set with a scoring rubric, or state plainly that GPT-4o was chosen after informal inspection.
13. **Baseline against [13]**. Section 1 distinguishes DRC-SG 2.0 [13] on the grounds that it "only focuses on extracting key design rule components rather than generating completed codes". Fair — but the only baseline in the paper is the authors' own standard prompting. Either run [13]'s extraction as the front end feeding a code generator, or state explicitly in Section 5.1 why no prior system is runnable on this task.
14. **Drop or support the C++ generalization claim** in Section 6 ("DRC-Coder can be generalized to generate codes using other programming language, e.g., C++"). Nothing in the paper tests it. Move it to future work, phrased as a conjecture.
15. **Reproducibility details to add to Section 5.1**: temperature and decoding parameters, seed, AutoGen version, iteration cap, Llama3 size and serving setup, the Phi-3 variant used in Figure 2, and token/API cost per rule. The GPT-4o snapshot (2024-05-13) is already there and should stay.
16. **Language pass**. "Abalation" appears three times (Section 5.1 intro, Section 5.1 Baselines, Section 5.3 heading) and should be "ablation". Other items: "manufacturbility" → "manufacturability" (Section 5.1); "unsatisfied performance" → "unsatisfactory performance" (Section 5.2); "this process need to be repeated" → "needs" (Section 1); "existing frameworks only processes" → "process" (Section 2.1); "These tools provides" → "provide" (Section 4); "provides a answer" → "an answer" (Section 4.2); "reduces the coding time from days of manual effort an average of four minutes" is missing "to" (Section 6); "highly outperforming" in the abstract reads oddly — "substantially outperforming" would be cleaner.

Recommendation

Major revisions. The problem is well chosen, the multi-agent decomposition with a domain-specific evaluation function is a sensible design, and the DRC Code Evaluation feedback report is a contribution I would like to see published — but the central result is measured on the same layouts the debugging loop optimizes against, the baseline runs at a fraction of the compute budget, the improvement percentages in Sections 5.2 and 5.3 do not reconcile with each other, and in this version, the main results table does not render at all, so none of the headline numbers can be checked. A held-out split, a feedback-matched

baseline, corrected percentages, and a restored Table 1 would, I think, leave the paper's core conclusion intact and make it defensible.

Declarations

Potential competing interests: No potential competing interests to declare.