WORKING PAPER · JULY 2026

Augmentation, Not Obsolescence

AI, the Jevons Paradox, and the Bifurcation of the Engineering Labour Market

Bidya Bhushan Nanda

B.Tech, Electronics Engineering
Maharaja Agrasen Institute of Technology (MAIT), Delhi
bidyabhushannanda@gmail.com

A synthesis of labour-market data, capability benchmarks, and production economics. All quantitative claims verified against originating sources in July 2026.

Abstract

A prevailing industry narrative holds that autonomous coding agents will render human software and hardware engineers obsolete. This paper tests that claim against labour-market data, capability benchmarks, and the economics of production, and finds it mis-specified. The evidence does not support obsolescence; it supports **recomposition** — a redistribution of work between humans and machines, and a sharp redistribution of opportunity between engineers at different career stages.

Three findings are developed. First, the demand for software is elastic rather than fixed, so falling production costs expand rather than contract aggregate output — the Jevons Paradox — which sustains demand for engineers who architect, integrate, and govern that output. Second, the capability gap between benchmark performance and production engineering, long the strongest argument for human indispensability, has narrowed dramatically: frontier models scored roughly 23% on SWE-bench Pro in late 2025 and between 59% and 80% by mid-2026 depending on evaluation harness. The durable human advantage is therefore *not* implementation capacity but organisational context, architectural intent, and accountability. Third — and most consequentially — the tasks AI automates best are precisely the tasks that constituted the engineering apprenticeship. Employment for software developers aged 22–25 has fallen roughly 20% from its 2022 peak while employment for developers aged 35–49 has grown, producing a two-speed labour market and a structural threat to the senior talent pipeline of the 2030s.

Hardware engineering is treated as a control case. VLSI and chip design have absorbed AI aggressively at the tooling layer, yet the economics of silicon — where a single escaped logic bug costs upward of $10 million in respin and mask costs — impose a determinism requirement that probabilistic models cannot satisfy alone. Formal verification, and the engineers who direct it, remain structurally necessary.

The conclusion is not that AI is harmless to engineering employment, nor that it is fatal to it. It is that AI raises the floor of required competence while leaving the ceiling of human contribution intact — and that the transition costs of this shift are being paid almost entirely by people at the start of their careers.

Keywords: artificial intelligence · Jevons Paradox · software engineering labour markets · autonomous coding agents · VLSI · formal verification · entry-level employment · SWE-bench

1. Introduction

The integration of large language models and agentic systems into engineering workflows has produced a confident and widely repeated prediction: that the human engineer is a transitional figure. In its strong form, the claim is that as coding agents approach autonomy, human involvement in web development, application development, full-stack

engineering, and even domains as physically constrained as Very Large-Scale Integration (VLSI) will become unnecessary.

This paper argues that the strong form of the claim is wrong, but that the popular rebuttal — *AI is just a tool, nothing changes* — is also wrong, and in a way that is more dangerous because it is more comforting.

The argument proceeds in three moves.

The first is economic. Automation reduces the marginal cost of producing software. If demand for software were fixed, this would mechanically reduce headcount. It is not fixed. Software demand is elastic, and in several segments super-elastic: an enormous reservoir of internal tooling, workflow automation, and niche applications has historically gone unbuilt only because engineering labour was too expensive to justify it. Lowering that cost unblocks the reservoir. This is the Jevons Paradox, and Section 2 develops it.

The second is technical. Section 3 assesses what coding agents actually achieve, and reports a finding that complicates the standard "AI can't do real engineering" defence. Through 2025, the collapse in model performance between curated benchmarks and enterprise-realistic ones was the single most-cited evidence for human indispensability. That gap has closed substantially within twelve months. Any argument for the enduring value of human engineers that rests on model incapability is therefore built on a depreciating asset. Sections 4 and 5 relocate the argument onto sturdier ground: organisational context, architectural intent, cost governance, and — in hardware — mathematical provability.

The third is distributional, and it is where this paper departs most sharply from optimistic accounts. Section 6 shows that aggregate stability conceals a severe cohort-level shock. The routine work that AI absorbs most completely is the same work that historically converted graduates into engineers. The industry is, in the phrase used by several analysts, eating its seed corn.

A note on the scope of the claim. This paper does not argue that every engineer is safe. It argues something narrower and more uncomfortable: that AI does not destroy engineering as a discipline, but does destroy a particular *mode* of engineering employment — the one in which value came from producing implementation volume — and that the people who occupied that mode were disproportionately at the beginnings of their careers.

2. Theoretical Framework: The Jevons Paradox in Digital Production

2.1 Origins

In *The Coal Question* (1865), William Stanley Jevons observed that improvements in the efficiency of coal use did not reduce British coal consumption. James Watt's steam engine made steam power dramatically cheaper per unit of work, which made it economically

viable across industries that had never used it, and total consumption rose. Efficiency gains, Jevons argued, are absorbed by expanded application rather than by reduced consumption, whenever demand for the underlying service is sufficiently elastic.

The elasticity condition is doing the real work in that sentence, and it is the condition that determines whether the paradox applies to software.

2.2 Why software demand is elastic

The software that exists in the world is not the software the world wants. It is the subset of desired software that cleared a historical cost-benefit threshold. Everything below that threshold — internal dashboards that were never worth a sprint, workflow automations that were never worth a headcount, regional-language interfaces that were never worth a localisation budget, small-market vertical tools that could never amortise a development team — sat unbuilt not because nobody wanted it but because engineering labour was expensive.

Lowering the cost of production moves the threshold. Work that was previously below the line becomes viable.

The organisational consequence is the part that most directly contradicts the obsolescence thesis. An engineer who can now ship in one day what previously took five does not become one-fifth as valuable. Firms do not hold ambition constant and harvest the savings as headcount reduction; they expand ambition, because the constraint that limited it has moved. The competitive dynamic reinforces this: a firm that banks the efficiency gain as cost savings is out-shipped by a firm that reinvests it as scope.

2.3 Empirical evidence: the recomposition of knowledge work

The most rigorous available evidence on generative AI and knowledge-work productivity comes from Brynjolfsson, Li, and Raymond's study of 5,179 customer-support agents at a Fortune 500 software firm, exploiting the staggered rollout of a generative AI conversational assistant (NBER Working Paper 31161; published in the *Quarterly Journal of Economics*, 2025).

Table 1. Productivity effects of generative AI assistance by worker skill tier

MEASURE	FINDING
Average productivity increase	14% increase in issues resolved per hour
Novice / low-skilled workers	34% improvement; markedly faster traversal of the experience curve
Experienced / high-skilled workers	Minimal gains
Mechanism	The model captures and disseminates the tacit knowledge and practices of high performers
Secondary effects	Improved customer sentiment; increased employee retention; evidence of worker learning

Source: Brynjolfsson, Li & Raymond (2023/2025).

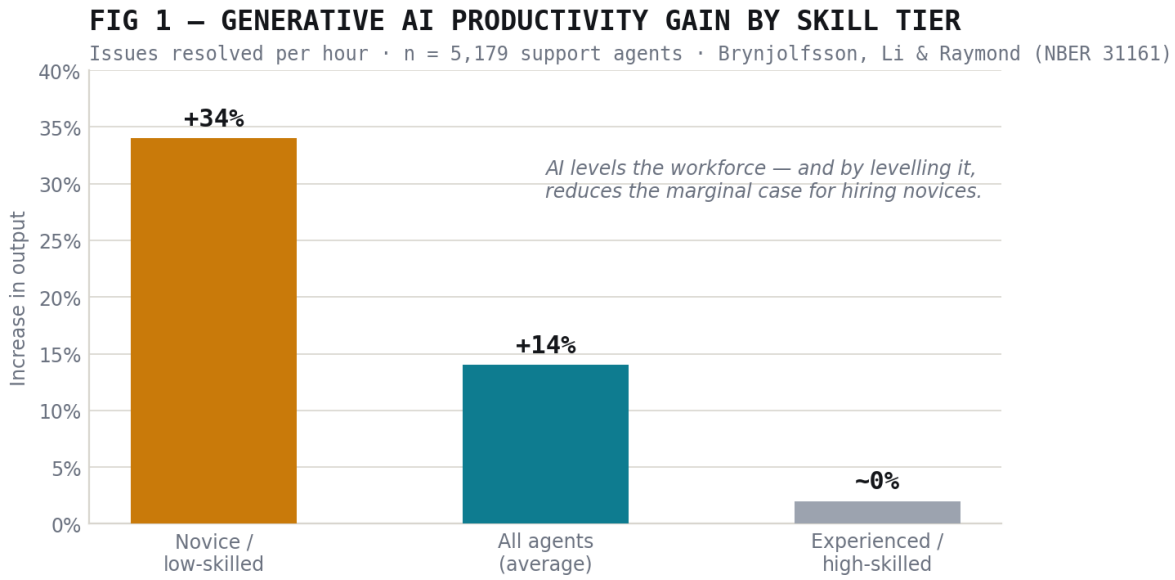

Figure 1. Generative AI productivity gain by worker skill tier. Source: Brynjolfsson, Li & Raymond, NBER 31161 / QJE 2025.

Two implications carry over to engineering.

The first is levelling. The gain is concentrated among the least experienced, because what the model supplies — best-practice patterns, idiomatic structure, the accumulated tacit knowledge of strong performers — is exactly what novices lack and experts already have. This is genuinely good news for individual junior workers who hold a job.

The second is the trap, and it is why Section 6 is necessary. If AI compresses the productivity distance between a novice and an expert, then from the employer's side the *marginal* value of hiring a novice falls relative to equipping an expert. Levelling within the workforce and exclusion at its boundary are the same mechanism viewed from two directions.

On the output side, the expansion is observable: GitHub reports year-over-year growth in code pushes and a surge in new application deployments. As output volume grows, the aggregate requirement for humans to architect, review, deploy, secure, and maintain that output grows with it. The role does not disappear; its centre of gravity shifts from *producing* code to *judging* it.

3. Capability Assessment: What Autonomous Coding Agents Actually Achieve

3.1 Adoption

Adoption is no longer marginal. GitHub Copilot surpassed 20 million cumulative users by mid-2025 and had approximately 4.7 million paid subscribers by January 2026, up 75% year-over-year. Roughly 90% of Fortune 100 companies use it. Among active users it now generates an average of 46% of code written — 61% among Java developers, up from 27% at launch. Controlled studies involving approximately 4,800 developers report task completion up to 55% faster.

Whatever else is true, the counterfactual world in which engineers do not use these tools no longer exists.

3.2 The benchmark landscape — and a correction

Benchmarks measure different things, and the differences matter more than the scores.

Table 2. AI coding benchmarks and what they measure (scores as of July 2026)

BENCHMARK	WHAT IT MEASURES	FRONTIER PERFORMANCE	INTERPRETATION
HumanEval	Single-function generation; isolated algorithmic logic	~100%	Saturated. Basic syntax and standard algorithms are solved.
SWE-bench Verified	Human-validated GitHub issues in open-source Python repositories	~96% (Claude Opus 5)	Approaching saturation; top models clustered within ~1 point.
Terminal-Bench	Agentic terminal use, command execution, filesystem navigation	Contested; still differentiates frontier models	Environment operation is largely competent but not solved.
SWE-bench Pro	Long-horizon, multi-file resolution across proprietary, uncontaminated enterprise codebases	~59% on Scale AI's standardised public set (June 2026); up to ~80% on other harnesses	The hard case — but no longer the wall it was twelve months ago.

Sources: Scale AI SWE-bench Pro leaderboard; llm-stats and BenchLM leaderboard aggregations, July 2026. Cross-harness comparisons are directional only; scaffolds and configurations differ materially between leaderboards.

This is the paper's most important revision to the received narrative. As recently as late 2025, frontier performance on SWE-bench Pro sat near 23%, and the collapse from ~70% on Verified to ~23% on Pro was the single most-cited quantitative evidence that AI could not perform real engineering. Within roughly twelve months that figure has risen to somewhere between 59% and 80% depending on evaluation harness. The wall did not hold.

The methodological caution is worth stating plainly: cross-leaderboard comparisons are unreliable, because harnesses differ in scaffolding, retry policy, and configuration, and the spread between reported figures for the same models is wide enough to be interpretively significant. But no reading of the data supports the claim that enterprise-realistic performance is stuck in the low twenties.

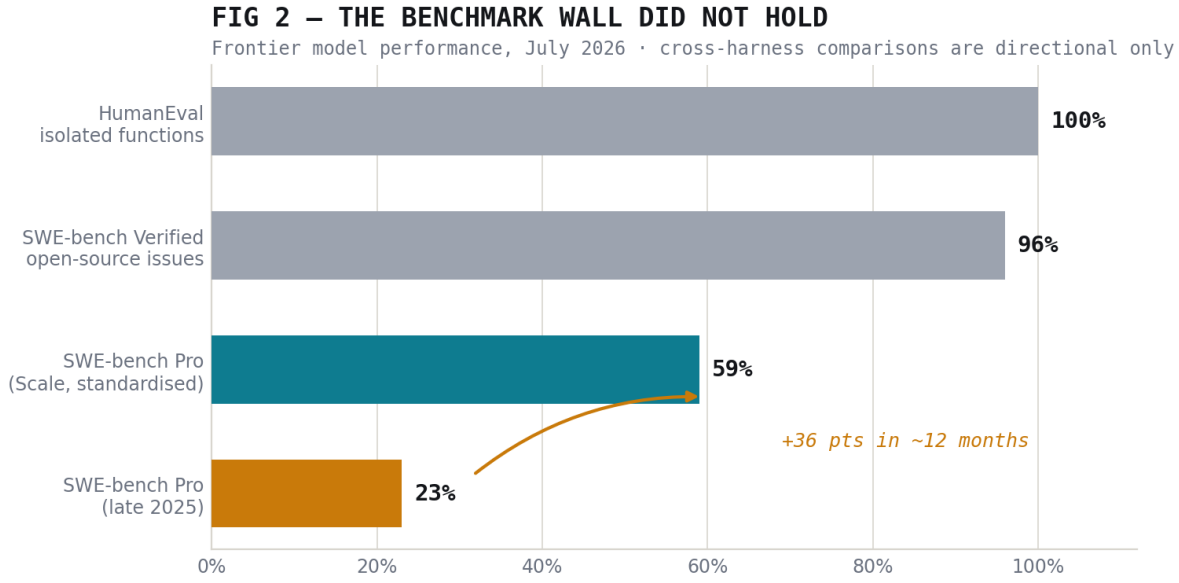

Figure 2. Frontier model performance across coding benchmarks, July 2026. Cross-harness comparisons are directional only.

3.3 Relocating the argument

The honest consequence is that any defence of human engineering that rests on *models cannot do this yet* is a defence with a short half-life. Every version of that argument made since 2023 has been overtaken within eighteen months.

The durable claims are different in kind. Production engineering spans multiple services, packages, schemas, and deployment configurations; it involves internal packages, bespoke CI pipelines, secrets management, and undocumented legacy constraints. An agent can trace a failing test, identify related files, and propose a coordinated refactor. What it cannot supply is the organisational context that determines whether that refactor serves the business objective, satisfies a security or regulatory standard, and interacts safely with downstream systems nobody documented.

Nor can it hold accountability. Someone has to be answerable for the change. This is not a capability gap that scales away, because it is not a capability gap.

The defensible position, then, is not that AI cannot implement. It is that implementation was never the scarce input. Architectural intent, contextual judgement, and accountability are — and they remain human.

4. Full-Stack Development Under AI-First Workflows

4.1 Tiny teams and role consolidation

The organisational expression of these productivity effects is team compression. Gartner projects that **60% of organisations will deploy smaller software engineering teams at scale by 2029, up from 15% in 2026** — a four-fold shift in three years. These "tiny teams" are typically four to five versatile engineers, each covering ground that previously required multiple specialists, supported by AI agents handling routine work and by platform engineering teams supplying standardised workflows.

Gartner is explicit that this is not primarily a cost-reduction tactic but a restructuring intended to exploit human and AI capabilities where each is strongest. That framing matters, because the two readings imply very different hiring behaviour.

The consolidation cuts across traditional role boundaries. As AI absorbs routine UI implementation, frontend specialists are pushed toward full-stack breadth. Demand for narrowly scoped UI production declines while demand for UX research and human-in-the-loop design rises. SDET and QA functions are increasingly absorbed into general development roles as automated tooling covers unit, integration, and regression testing. The emergent professional profile is a generalist — technical execution plus domain depth plus the ability to direct multiple agents.

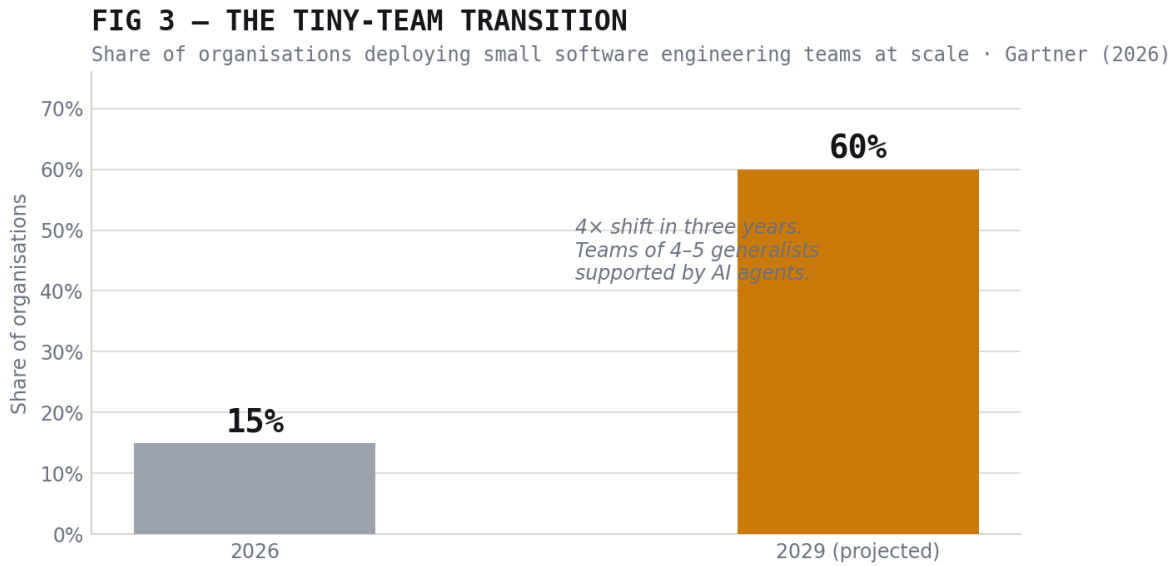

Figure 3. Projected adoption of small software engineering teams at scale. Source: Gartner (2026).

4.2 Token economics: efficiency becomes a measurable line item

A constraint has appeared that did not exist in previous automation waves: the marginal cost of the automation itself is high, variable, and attributable to individuals.

Gartner projects that **by 2028, enterprise AI coding costs will exceed the average developer's salary**, driven by token consumption and the shift to consumption-based pricing. This is not a distant projection. Nearly a quarter of technology leaders already report spending $200–$500 per developer per month on coding tokens; reported bills have escalated from $20–$100 to $2,000–$5,000 per developer per month, with extreme cases reaching $20,000.

Gartner's observation about the mechanism is the sharpest point in the finding: token discipline will not emerge from developer choice, because developers optimise for speed and convenience rather than cost. Without a governed operating model, spend escalates faster than the productivity it purchases.

This produces something historically unusual. Two engineers with identical output can now have materially different costs to the firm, and the difference is legible in a billing dashboard. Efficiency has become measurable at the individual level in a way it never was when the input was salaried time. The practical disciplines — context engineering to reduce token consumption without degrading output, and intelligent model routing that reserves frontier models for high-value architectural work while directing high-frequency tasks to smaller ones — are becoming part of the baseline definition of engineering competence.

5. Hardware as Control Case: Why VLSI Resists Full Automation

Hardware is the useful control case for the obsolescence thesis, because it has adopted AI aggressively while remaining structurally resistant to autonomy. If the resistance were merely a matter of AI not having arrived yet, hardware would look like software with a lag. It does not.

5.1 AI has already reshaped chip design

Modern SoCs integrate tens of billions of transistors, and the Power-Performance-Area (PPA) design space is far too large for manual optimisation. EDA vendors have responded aggressively: Cadence Cerebrus and Synopsys DSO.ai apply reinforcement learning and generative methods to floorplanning, layout optimisation, and design-space exploration. Cadence reports over 1,000 completed designs using AI-driven tools and claims acceleration of chip delivery timelines by five to ten times against manual methods. Vendor-reported case studies describe compressions of comparable magnitude at advanced nodes — one cites a 6nm networking chip's tapeout timeline falling from six weeks to two. These are vendor figures and should be read as such, but the direction of travel is corroborated across the industry.

Domain-specialised models exist at the HDL layer. NVIDIA's ChipNeMo applies domain-adaptive pretraining and custom tokenisation to internal hardware design data, specifications, and documentation, supporting EDA script generation, bug summarisation, and architectural Q&A. Frameworks such as LLM-VeriPPA use two-stage LLM pipelines to generate Verilog with attention to syntactic correctness and PPA constraints.

The adoption is not partial. It is deep and production-grade.

5.2 The determinism requirement

What makes hardware different is not the tooling. It is the cost structure of being wrong.

Software defects are patchable at near-zero marginal cost after deployment. Silicon defects are not. HDLs are also structurally hostile to probabilistic generation in a way that sequential code is not: they are declarative and concurrent, every statement executes simultaneously, and a line that reads as syntactically plausible can introduce race conditions, pipeline hazards, or cross-clock-domain deadlocks that no amount of surface plausibility will reveal.

The economics compound this along the design cycle.

Table 3. Approximate cost of defect detection by semiconductor design stage

DESIGN STAGE	PRIMARY DETECTION METHOD	APPROX. COST TO FIX	RISK PROFILE
RTL design	Designer inspection / linting	~$100	Negligible
Block verification	Unit simulation / directed tests	~$1,000	Low
System verification	Full-chip emulation / regression	~$10,000	Moderate
Post-silicon (lab)	Validation boards / logic analysers	~$10,000,000+	Catastrophic
In the field	Customer return / product recall	~$100,000,000+	Existential

Order-of-magnitude figures reflecting semiconductor defect economics; exact costs vary by node, volume, and product segment.

A logic bug that escapes into tapeout at a 5nm node invalidates mask sets and forces a respin costing upward of $10 million, with a schedule impact measured in months. Because semiconductor products are sold into narrow market windows, a six-month delay can forfeit a substantial share of a product's lifetime gross profit.

And first-time success is already rare and getting rarer. The 2024 Siemens EDA / Wilson Research Group Functional Verification Study found that **only 14% of ASIC/SoC projects achieved first-silicon success — the lowest figure in more than two decades of tracking.** Logic and functional flaws remain the leading cause of respins. Verification is estimated to consume roughly 70% of total project cost, and in processor development the ratio of verification engineers to design engineers runs about five to one.

The industry's tolerance for hallucination in this pipeline is therefore not low. It is zero — and it is zero at a moment when the discipline is already failing to hit first silicon 86% of the time.

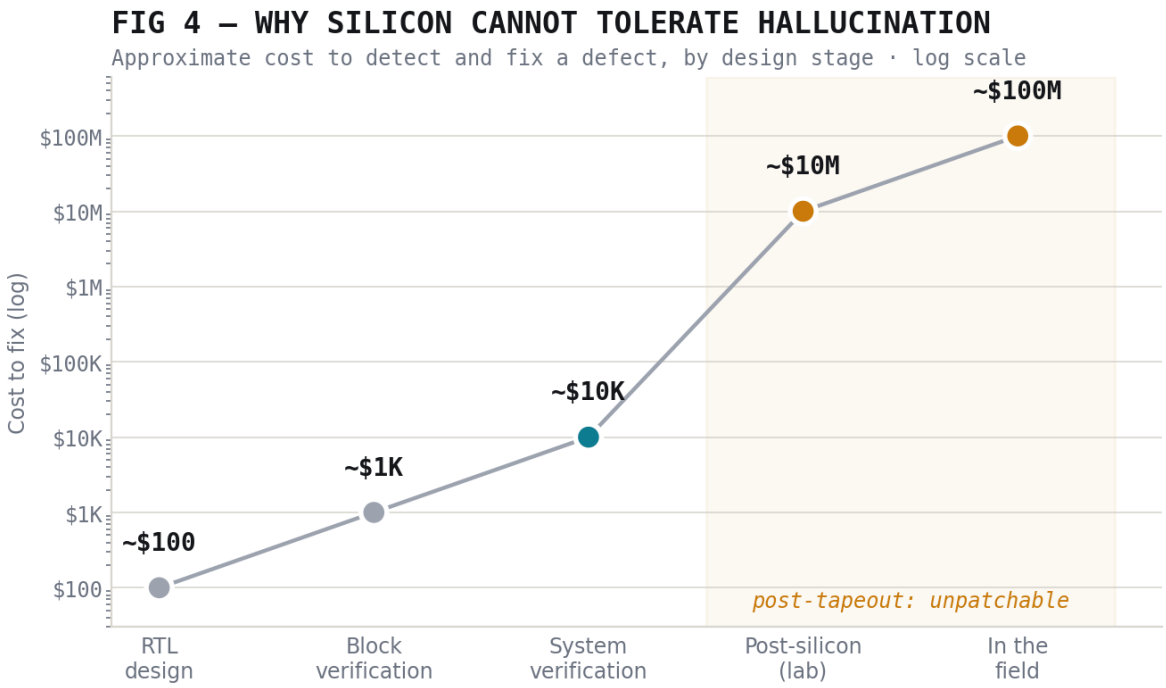

Figure 4. Approximate cost to detect and fix a defect by semiconductor design stage (log scale). Order-of-magnitude figures.

5.3 Formal verification as the deterministic judge

An LLM performing probabilistic token prediction cannot guarantee mathematical correctness. A model may generate RTL that passes ten thousand directed simulation tests and still miss a simulation-resistant corner case that deadlocks under a specific interrupt-and-branch sequence.

Formal verification addresses precisely this: a static mathematical proof process, typically using SMT solvers, that exhaustively explores the state space to prove the *absence* of a class of bug rather than merely failing to find one. Simulation samples behaviour; formal verification quantifies over it.

The productive division of labour in modern hardware workflows follows directly. AI is highly effective at generating initial RTL, drafting SystemVerilog Assertions, and identifying coverage gaps. Formal verification is the deterministic judge of whether that output is correct. The AI proposes; the solver disposes.

The human role shifts accordingly — away from writing boilerplate Verilog, toward architectural micro-judgement, requirements traceability, and review of debugging evidence. The hardware engineer's irreducible function is translating ambiguous product specifications into mathematically provable intent. That translation is the step AI cannot perform, because the ambiguity is in the human requirement, not in the code.

6. The Structural Collapse of the Junior Pipeline

Sections 2 through 5 support a reassuring conclusion at the aggregate level. This section is where the aggregate conclusion breaks down.

The thesis that AI "only destroys those who use it inefficiently" contains an assumption that does not survive contact with the data: that everyone gets the chance to be efficient. Entry-level engineers are, by definition, not yet efficient. They are the population the mechanism cannot protect.

6.1 The evidence

The 2026 Stanford AI Index, drawing on ADP payroll records covering millions of workers at tens of thousands of firms from 2021 through mid-2025, found that **employment for software developers aged 22–25 has fallen nearly 20%**, with the decline concentrated after late 2022. Developers aged 26 and above saw stable or growing employment over the same period; secondary analyses of the same dataset place growth in the 35–49 band in comparably AI-exposed occupations at roughly 6–9%.

The same divergence appears across the hiring funnel:

- **Entry-level software engineering postings in the US fell approximately 67% between 2023 and 2024** (Stanford Digital Economy Lab).
- Tech-specific internship postings declined by roughly 30%.
- Junior developers fell to about **7% of tech hiring, from 15% three years earlier**.
- The fifteen largest tech firms cut entry-level hiring by approximately 25% between 2023 and 2024 (SignalFire).
- Roughly **22% of CHROs** report that at least one business leader in their organisation has stopped hiring for entry-level roles entirely due to AI automation (Gartner).
- Computer science graduates carry a 6.1% unemployment rate; computer engineering graduates, 7.8%.
- In the UK, tech graduate roles fell 46% in 2024.

The critical observation is that **this is not a demand contraction.** Total software demand is expanding, and senior employment is growing. A uniform downturn would depress every cohort. This one depresses exactly one — which points to substitution rather than recession.

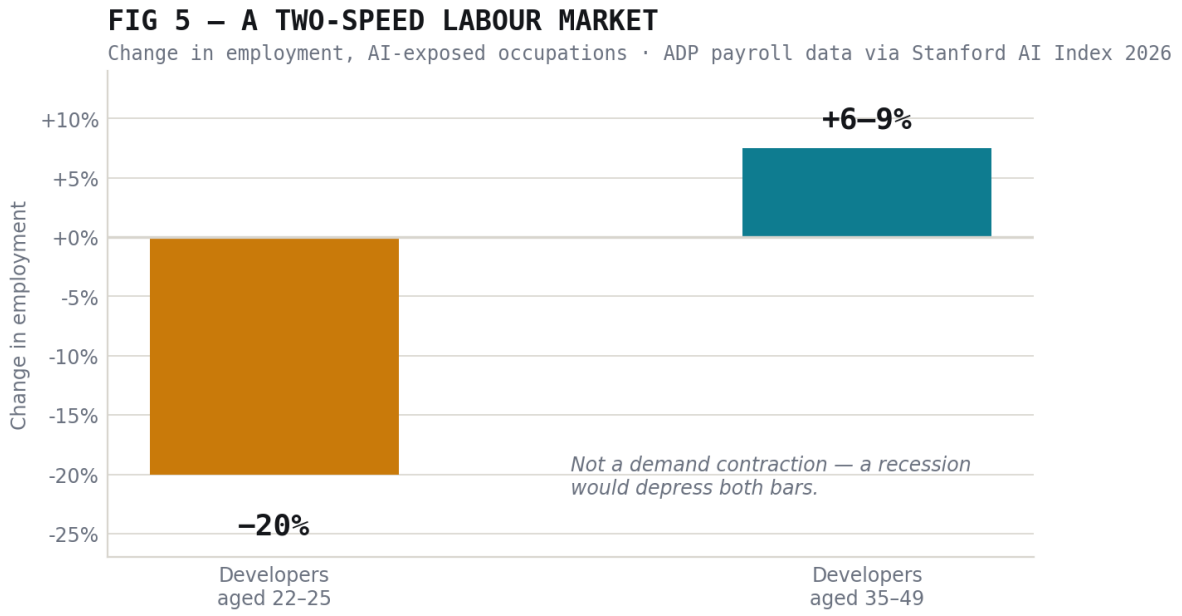

Figure 5. Change in employment by age cohort in AI-exposed occupations. Source: ADP payroll data via Stanford AI Index 2026.

6.2 The mechanism

The substitution is legible in the task composition of junior work. Junior engineers were historically hired to write boilerplate, implement standard CRUD operations, produce unit tests, and fix well-scoped bugs. These were not merely how juniors contributed; they were the apprenticeship — the means by which a graduate absorbed a codebase, internalised conventions, and accumulated the tacit knowledge that eventually produces architectural judgement.

Those tasks are the ones coding agents perform most reliably. Section 3 documents near-saturation on exactly this class of work.

The consequence is a widened distance from graduation to net-positive contribution. A team of five senior engineers with agent tooling can produce the output that previously required eight including three juniors. Firms respond rationally in the short term: freeze junior recruitment and demand three to five years of experience for nominally entry-level roles, ensuring new hires arrive with the judgement to validate AI output on day one.

Every firm making this decision individually is behaving sensibly. The aggregate result is that the industry stops producing the seniors it is now competing for.

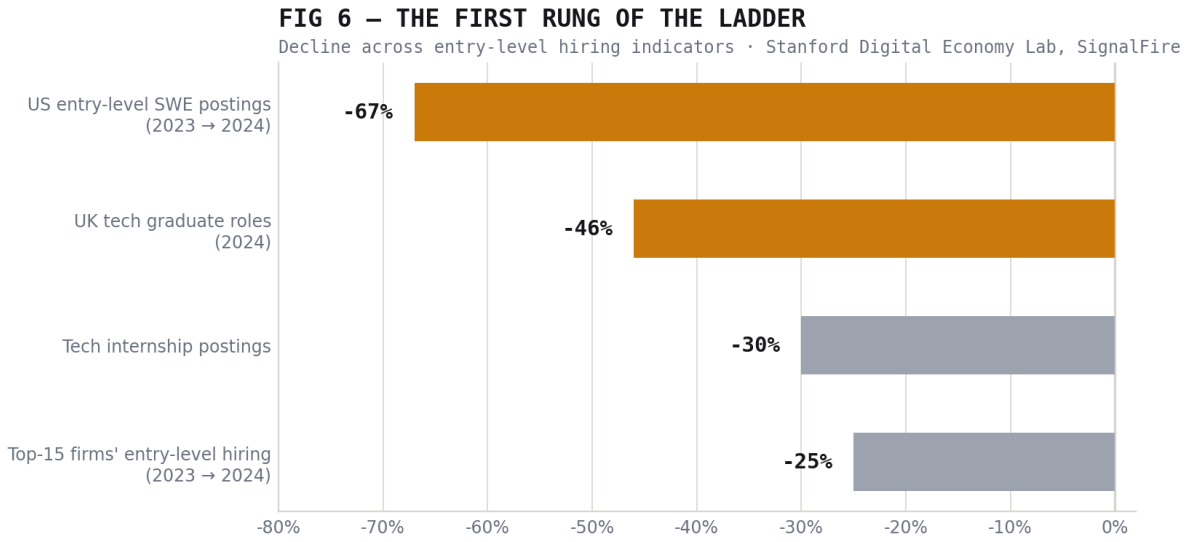

Figure 6. Decline across entry-level hiring indicators. Sources: Stanford Digital Economy Lab; SignalFire.

6.3 Macroeconomic amplification

AI is the direct substitution mechanism, but macroeconomic conditions sharpened it, and attributing the entire effect to AI overstates the case.

The zero-interest-rate environment through 2021 made speculative hiring cheap; firms could absorb the six-to-twelve-month runway before a junior developer became productive. The shift to a high-rate environment demanded near-term ROI and training budgets were cut early.

A second distortion is expectation mismatch. Enterprise leadership frequently anticipates generative AI productivity gains near 25% across the board. Measured organisational gains — net of integration friction, review overhead, and rework — are typically far lower. When the gap appears in results, AI is available as a scapegoat: the tools are assumed to work, so the shortfall is attributed to the people, and entry-level hiring is tightened further to protect quarterly earnings.

Disentangling the AI effect from the rate-cycle effect is genuinely difficult, and this paper does not claim to have done so. The cohort-specific pattern of the decline is the strongest available evidence that AI substitution is doing real work, but the honest position is that AI is a substantial contributor to an effect it did not solely cause.

6.4 The leadership vacuum

Gartner's warning is that organisations relying on AI to eliminate junior roles are hollowing out their own engineering talent pipeline, inhibiting knowledge transfer and restricting future recruitment to an expensive and contested senior market.

The paradox is exact: the path into engineering is narrowing at precisely the moment senior engineers become most valuable. It is not possible to produce a senior architect

without first employing a junior developer, and the lead time on that transformation is measured in years. A hiring freeze imposed in 2024 produces a seniority shortage around 2032.

Mitigation requires deliberate redesign of early-career roles rather than their elimination: skills-based rather than tenure-based advancement, AI-assisted simulation and guided practice environments, and structured apprenticeship programmes that compress the acquisition of foundational expertise. The alternative is to trade a short-term margin improvement for a long-term architectural capability deficit.

7. Sectoral Case: The Restructuring of India's IT Industry

India's technology sector offers a large-scale natural experiment, because its historical business model was the one most directly exposed to the mechanism this paper describes.

NASSCOM projects Indian tech industry revenue at **$315 billion in FY26**, growing 6.1%, with direct employment near **6 million** after adding approximately 135,000 people — a 2.3% increase. AI-related revenues are estimated at $10–12 billion. The sector remains a net hirer.

The revenue composition is instructive: IT services $149B, engineering R&D $63B, business process management $59B, software products $23B, hardware $21B.

The structural change sits in the delivery model rather than the headline number. The industry was built on scale-led growth and Full-Time Equivalent billing — revenue as a function of human hours. As AI-driven productivity materialises, providers are shifting toward outcome-based, risk-sharing constructs. Clients increasingly decline to pay for hours spent producing boilerplate; they pay for a delivered system.

This is the Jevons mechanism operating at national scale, and it is worth noting which part of it held. Aggregate employment grew. Aggregate revenue grew. What changed was the *basis* on which labour is valued — from volume of hours to value of outcome. The response has been correspondingly large: over 2 million professionals upskilled in AI during FY26, including 200,000–300,000 in advanced capabilities, supported by initiatives such as FutureSkills Prime.

The Indian case therefore supports the paper's central claim in both directions. AI did not destroy the sector's employment. It did destroy the pricing model that employment was built on, and the sector's continued growth is conditional on a transition it is still in the middle of.

8. Discussion: A Two-Speed Labour Market

Synthesising the preceding sections yields a picture more differentiated than either the obsolescence narrative or its dismissal.

Table 4. Differential exposure to AI substitution by engineering role profile

PROFILE	PRIMARY VALUE CONTRIBUTION	EXPOSURE	TRAJECTORY
Entry-level generalist developer	Implementation volume, boilerplate, standard tests	**Severe**	Postings down ~67%; cohort employment down ~20%
Mid-level specialist (narrow scope)	Framework-specific implementation	**High**	Role consolidation pressure toward full-stack breadth
Senior engineer / architect	System design, integration judgement, accountability	**Low**	Employment growth of ~6–9% in AI-exposed occupations
Platform / infrastructure engineer	Standardised workflows enabling tiny teams	**Low**	Structurally reinforced by the tiny-team model
Hardware / verification engineer	Provable correctness, requirements traceability	**Very low**	Protected by determinism requirements and defect economics

The two speeds are not fast and slow versions of the same trajectory. They are opposite trajectories, and they are separated by a barrier — the apprenticeship — that AI has substantially removed.

Three propositions follow.

Aggregate demand is not the risk; access is. Every indicator of total engineering demand points upward. The risk is concentrated entirely at the entry point, and aggregate figures conceal it by construction.

Arguments from model incapability depreciate; arguments from accountability do not. Section 3 demonstrated how quickly a capability-gap argument can be overtaken. The durable human functions — setting architectural intent, supplying organisational context, bearing responsibility for outcomes, and in hardware, translating ambiguous specification into provable intent — are not tasks awaiting a better model. They are structurally human.

Efficiency has become measurable, and therefore consequential. Token economics converts what was previously an unobservable difference in working style into an attributable cost. This is the strongest available support for the claim that AI penalises inefficiency — not through a vague competitive disadvantage, but through a line item.

9. Limitations

Several constraints bound the confidence of these findings.

Benchmark volatility. Section 3's central figures moved by a factor of three within twelve months. Any conclusion drawn from current capability measurements has a short shelf life, and this paper's own benchmark table should be treated as a July 2026 snapshot rather than a stable fact.

Cross-harness incomparability. Reported SWE-bench Pro figures for July 2026 range from approximately 59% to 80% depending on the evaluation leaderboard. Scaffolding, retry policy, and configuration differ materially. Comparisons across leaderboards are directional at best.

Attribution. Section 6.3 addresses this directly: the entry-level collapse coincides with a major interest-rate regime change and post-2021 over-hiring correction. The cohort-specific pattern is strong evidence for AI substitution, but no study reviewed here fully isolates the AI effect.

Transfer from the NBER study. Table 1's findings come from customer support, not software engineering. The mechanism — AI encoding and distributing expert tacit knowledge — plausibly transfers, but the magnitudes should not be assumed to.

Forecast dependence. Several load-bearing claims (60% tiny-team adoption by 2029; AI coding costs exceeding developer salaries by 2028) are analyst projections, not observations. They describe expectations, which can be wrong.

Cost figures are order-of-magnitude. Table 3's semiconductor defect costs vary substantially by node, volume, and product segment. The exponential shape is well established; the specific values are illustrative.

Survivorship in productivity data. Self-reported and vendor-reported productivity gains (46% of code generated, 55% faster completion) derive largely from populations that adopted the tools and continued using them. They should not be read as population-level effects.

10. Conclusion

The evidence assembled here supports a qualified version of the thesis that AI will not destroy full-stack development, computer architecture, or VLSI engineering — with the qualification carrying more weight than is usually acknowledged.

AI is not a substitute for the engineering profession. It is an abstraction layer. In software, it lowers the cost of production, and because demand for software is elastic, the resulting expansion in output sustains aggregate demand for engineers who architect, integrate, and govern it. In hardware, the economics of silicon respins and the inability of probabilistic models to guarantee mathematical correctness make deterministic formal verification — and the engineers who direct it — structurally necessary rather than merely currently useful.

But the argument for human indispensability has to be made carefully, because one common version of it is failing. Between late 2025 and mid-2026, frontier performance on enterprise-realistic coding benchmarks rose from roughly 23% to between 59% and 80%. Any case for the human engineer that rests on what models cannot yet do is a case that has been overtaken repeatedly and will be again. The defensible case rests elsewhere: on

organisational context, architectural intent, cost governance, and accountability — none of which are capability gaps.

The transition is also not costless, and the costs are not evenly borne. The tasks AI performs best are the foundational rungs of the engineering career ladder. A 20% employment decline among developers aged 22–25 alongside growth among those aged 35–49, and a roughly 67% collapse in entry-level postings, describe a structural crisis in how the profession reproduces itself. This is not a transitional inefficiency that clears on its own. Without deliberate redesign of early-career pathways — skills-based advancement, structured apprenticeship, guided practice environments — the industry will arrive in the 2030s having optimised its way out of the senior engineers it depends on.

The synthesis is this. AI does not replace human engineers, but it raises the floor of required competence and removes the ramp that used to lead up to it. The future belongs to engineers who combine domain depth, system-level judgement, and disciplined governance of AI agents. Those who adapt will operate at a scale of output the profession has not previously seen. Those who do not will be priced out — and, most urgently, those who have not yet been given the chance to adapt are being priced out before they begin.

References

1. Brynjolfsson, E., Li, D., & Raymond, L. R. (2023). *Generative AI at Work.* NBER Working Paper 31161. Published in *Quarterly Journal of Economics*, 140(2), 889 (2025). https://www.nber.org/papers/w31161
2. Stanford HAI. (2026). *The 2026 AI Index Report — Chapter 4: Economy.* https://hai.stanford.edu/ai-index/2026-ai-index-report/economy
3. Gartner. (2026, July 7). *Gartner Predicts 60% of Organizations Will Adopt Smaller Software Engineering Teams by 2029.* https://www.gartner.com/en/newsroom/press-releases/2026-07-07-gartner-predicts-60-percent-of-organizations-will-adopt-smaller-software-engineering-teams-by-2029
4. Gartner. (2026, June 24). *Gartner Predicts AI Coding Costs Will Surpass Average Developer's Salary by 2028 as Token Consumption Surges.* https://www.gartner.com/en/newsroom/press-releases/2026-06-24-gartner-predicts-ai-coding-costs-will-surpass-average-developer-salary-by-2028-as-token-consumption-surges
5. Gartner. (2026, July 27). *Gartner Survey Finds AI Automation Is Reducing Some Entry Level Hiring at Nearly One-Quarter of Organizations.* https://www.gartner.com/en/newsroom/press-releases/2026-7-27-gartner-survey-finds-ai-automation-is-reducing-some-entry-level-hiring-at-nearly-one-quarter-of-organizations
6. Scale AI Labs. *SWE-bench Pro Leaderboard (Public Dataset).* https://labs.scale.com/leaderboard/swe_bench_pro_public
7. llm-stats. *SWE-bench Verified Leaderboard.* https://llm-stats.com/benchmarks/swe-bench-verified
8. Siemens EDA & Wilson Research Group. (2024). *IC/ASIC Functional Verification Trends Report.* https://resources.sw.siemens.com/en-US/white-paper-2024-wilson-research-group-ic-asic-functional-verification-trend-report/
9. Semiconductor Engineering. *First-Time Silicon Success Plummets.* https://semiengineering.com/first-time-silicon-success-plummets/
10. Semiconductor Engineering. *Formal Verification First: How AI Supports But Cannot Replace It.* https://semiengineering.com/formal-verification-first-how-ai-supports-but-cannot-replace-it/
11. Liu, M. et al. (2023). *ChipNeMo: Domain-Adapted LLMs for Chip Design.* arXiv:2311.00176. https://arxiv.org/abs/2311.00176

12. *LLM-VeriPPA: Power, Performance, and Area Optimization aware Verilog Code Generation with Large Language Models.* arXiv:2510.15899. https://arxiv.org/html/2510.15899v1
13. Cadence. *Chip Design Industry Reaches an AI Inflection Point.* https://community.cadence.com/cadence_blogs_8/b/corporate-news/posts/chip-design-industry-reaches-an-ai-inflection-point
14. TTI, Inc. *AI-Driven Chip Designs Improve PPA and Productivity.* https://www.tti.com/content/ttiinc/en/resources/marketeye/categories/new-technology/me-slovick-20240625.html
15. McKinsey & Company. *Unleashing Developer Productivity with Generative AI.* https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/unleashing-developer-productivity-with-generative-ai
16. McKinsey & Company. *How an AI-Enabled Software Product Development Life Cycle Will Fuel Innovation.* https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/how-an-ai-enabled-software-product-development-life-cycle-will-fuel-innovation
17. Microsoft Research. *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.* https://www.microsoft.com/en-us/research/publication/the-impact-of-ai-on-developer-productivity-evidence-from-github-copilot/
18. NASSCOM. (2026). *Technology Sector in India: Strategic Review 2026.* https://nasscom.in/knowledge-center/publications/technology-sector-india-strategic-review-2026
19. FutureSkills Prime. https://www.futureskillsprime.in/
20. Jevons, W. S. (1865). *The Coal Question.* Macmillan & Co. Summary: https://en.wikipedia.org/wiki/Jevons_paradox
21. Press Information Bureau, Government of India. *India's Youth Dividend in the AI Era.* https://www.pib.gov.in/PressNoteDetails.aspx?id=157463

Methodology note

This paper is a synthesis of publicly available secondary sources: peer-reviewed and working-paper economics literature, industry analyst research (Gartner, NASSCOM, McKinsey), public benchmark leaderboards, and industry survey data (Siemens EDA / Wilson Research Group). No primary data was collected. All quantitative claims were verified against their originating sources in July 2026. Benchmark figures in Table 2 are explicitly time-stamped because they are volatile; readers consulting this paper at a later date should assume they have moved.

Where the underlying research draft contained figures that subsequent verification contradicted — most notably frontier performance on SWE-bench Pro — the corrected figures are reported and the implications for the argument are addressed rather than elided.

Copyright and citation

Suggested citation: Nanda, B. B. (2026). *Augmentation, Not Obsolescence: AI, the Jevons Paradox, and the Bifurcation of the Engineering Labour Market.* Working paper, July 2026.

Correspondence: bidyabhushannanda@gmail.com