Legal FPGA Routing at 799,403 Nets on a Single GPU

A technical brief from Voxell. Author: Jonathan Corners, Voxell Inc. Contact: voxell.ai/mi-go **Platform:** NVIDIA RTX PRO 6000 Blackwell (sm_120, 96 GB). CPU baseline host: Intel Core i9-14900KF, 192 GB DDR5. **Patent status:** U.S. Patent Application No. 19/731,882, filed July 2026.

Abstract

Routing is the dominant stage of FPGA compilation, and the field has long treated it as intrinsically sequential. Parallel routers either fail to reach a legal result or reach one by reconstructing a serial order underneath. mi-go is a GPU-native FPGA detailed router that reaches fully legal routes, with every capacity constraint satisfied and the parallel routing itself doing the work, on a single GPU. It is demonstrated across a 200x range of design size: `picosoc` (4,055 nets) legal in about half a second against a 1.23 second reference, `neuron` (50,835 nets) legal in about twelve seconds at the same channel width against a 15.77 second reference, and the two Titan-class designs `sparcT2` (182,090 nets) and `bitcoin_miner` (799,403 nets) legal on a single 96 GB card. Every legality claim in this brief is empirically legal on our tests: an independent check re-derives resource occupancy directly from the emitted routes and confirms full source-to-sink connectivity at zero capacity overuse, so the router's own state is never trusted alone. To our knowledge, no prior GPU router of any architecture has reached zero capacity overuse at even one tenth of this scale. Legal first, optimal second.

Hardware, baseline, and designs

- **Hardware.** A single NVIDIA RTX PRO 6000 Blackwell GPU (sm_120, 96 GB). No multi-GPU, and no CPU co-solver in the routing loop for any legal result. The CPU baseline ran on an Intel Core i9-14900KF (24C/32T) with 192 GB DDR5.
- **Baseline.** VPR (VTR 9), the standard open-source CPU FPGA router, timed on its single fixed-width route pass. This is the realistic head-to-head, not the academic minimum-width search.
- **Designs.** Four designs spanning a 197x range in net count. The routing-resource graph (RRG), not the net count, is what governs the router's memory footprint and search cost; it grows with both design size and channel width.

Design	Class	Nets	RRG nodes	Channel width	Role
`picosoc`	small SoC	4,055	185 K	W=250	correctness and speed target

`neuron`	Titan23	50,835	3.0 M	W=200	first scale target
`sparcT2`	Titan23 (Stratix-IV class)	182,090	6.1 M	W=440	first Titan-mid target
`bitcoin_miner`	Titan23	799,403	13.7 M (173 M edges)	W=400	largest attempted, 16x `neuron`, 197x `picosoc`

Results

Design	Nets	mi-go (single GPU)	VPR 9 (CPU)	Verification
`picosoc`	4,055	about 0.5 s, LEGAL	1.23 s	independent check, 4,055/4,055 connected, zero overuse
`neuron`	50,835	about 12 s, LEGAL	15.77 s (same width)	independent check, 50,835/50,835 connected, zero overuse
`sparcT2`	182,090	about 326 s, LEGAL (W=440)	81.7 s (W=400)	independent check, 3 of 3 runs legal, 182,090/182,090 connected, zero overuse
`bitcoin_miner`	799,403	about 259 s, LEGAL	124.8 s (W=400)	independent check, 5 of 5 runs legal, 799,403/799,403 connected, zero overuse

Context for the comparison. The reference router embodies roughly three decades of continuous optimization, from PathFinder in 1995 through nine major VPR releases. mi-go is a first-generation engine whose program to date had a single target: fully legal routing on a GPU, the property the field lacked. The numbers are printed side by side and left to speak. The property that has no counterpart in the table is that no published GPU router, of any architecture, has reached zero capacity overuse at any of these scales.

Note on the `neuron` comparison. At the same channel width (W=200) both routers reach a fully legal, independently verified route: VPR in 15.77 s, mi-go in about 12 s. Separately, VPR's full minimum-width search (nine route attempts that find the tightest feasible width, 130) takes about 697 s, a different and much larger task that mi-go does not perform. One asymmetry to state plainly: VPR routes down to that minimum width and mi-go currently does not, it needs about 1.5x minimum. VPR is more width-capable, mi-go is faster at the width it routes.

Note on the Titan-class comparison. `bitcoin_miner` was routed by both at the same channel width (W=400): VPR in 124.8 s and mi-go in about 259 s. `sparcT2` mi-go routes legally at W=440, its 1.33x-minimum width, against a VPR reference of 81.7 s, so at Titan scale mi-go needs more channel width than the reference and is slower here, consistent with the width premium noted below. The largest result is not a ceiling: fitting a 13.7M-node instance in 96 GB was footprint engineering, and disclosed route-storage layout in the filed application extends single-GPU capacity well beyond it.

Test conditions (so the numbers reproduce)

- **Same design, same placement, same width, both legal** wherever a head-to-head is claimed. A pure routing comparison, not a placement or width-search comparison.
- **VPR (CPU baseline).** Open-source VPR 9 (VTR 9), a single fixed-width route (`--route_chan_width 200` on `neuron` , `400` on the Titan designs, no minimum-width search), reusing a fixed placement. On `neuron` it converged in 18 routing iterations, routing stage 15.77 s in isolation (the full pack, place, and route flow was about 46 s). On `picosoc` the single route was 1.23 s.
- **mi-go.** One routing pass over the same-width routing-resource graph on a single RTX PRO 6000 Blackwell GPU. Reported times are the median of five warm runs (`neuron` 12.27 s, `picosoc` about 0.5 s). Titan-class times are as measured on the verified runs.
- **Verification, every design.** Every legality figure passed an independent host-side check that re-derives resource occupancy directly from the emitted routes and confirms end-to-end source-to-sink connectivity at zero capacity overuse. The check is parallelized so it scales with the designs it gates, and the router's own internal state is never trusted on its own.

The width behavior, characterized

For a long time `neuron` plateaued short of legality, and the finding of that campaign is why. The original instance sat exactly at the minimum feasible channel width, the narrowest routing graph on which the design can route at all, with zero routing slack. Regenerating the graph at a modestly wider channel and routing it with the same engine, no new algorithm, reaches full legality. The banked, independently verified result is legality at W=200, which is 1.54x the minimum feasible width. The same behavior repeats at Titan scale, where `sparcT2` legalizes near 1.33x its reference width. Across the size range the engine's width requirement is a stable, characterized property, roughly 1.5x minimum, rather than an instance accident.

Design	Minimum feasible width	mi-go legal width	Premium
`neuron`	W=130	W=200	1.54x
`sparcT2`	~W=330	W=440	1.33x

The premium is characterized, not incidental: it is the channel slack mi-go currently needs above the tightest width on which the design can route at all.

Honest caveat: the width premium

Published routers typically reach legality nearer 1.3x minimum width, so mi-go currently needs more slack than the best CPU routers do. This is stated plainly as the open item. It is a width property, not a solver gap:

below about 1.5x minimum, the conflicts that remain are genuine local corridor congestion, small clusters of nets that must share a channel that is physically full, where no single move frees room. That diagnosis was certified by an exact solver, not inferred from the parallel router's behavior. Closing the premium toward 1.3x is the remaining work. Reaching full legality at scale on a single GPU is done.

Declared scope: routability first

mi-go's program to date is routability-driven. Legality at scale was the property the field lacked, so it was the target, and the results in this brief are legality and runtime results. No timing or wirelength quality claims are made here. Timing-driven and wirelength-driven operation are on the roadmap, and they are a cost-model change rather than an architecture change: delay and criticality enter through the route cost function the engine already optimizes, while the parallel machinery that reaches legality is unchanged. These are stated as scope, not discovered as omissions.

Verification

Correctness is not self-reported. Every time mi-go declares a route legal, an independent host-side check re-derives resource occupancy directly from the emitted routes and confirms end-to-end source-to-sink connectivity, requiring zero capacity overuse. All four results above passed this check: `picosoc` (4,055/4,055 connected), `neuron` (50,835/50,835, zero overuse, on repeated independent runs), `sparcT2` (182,090/182,090 connected, on three independent runs), and `bitcoin_miner` (799,403/799,403 connected, zero overuse, zero broken, zero unplaced, on five independent runs).

mi-go is non-deterministic by design: it is a parallel negotiation, and which of two contenders keeps a contested wire and which one detours is settled by the parallel hardware, not by a fixed order. The reliability claim is therefore not that a run reproduces a particular route, it is that every run reaches a legal one, confirmed independently. A run that does not reach legal reports that and exits with a failure status, so a non-legal result can never be mistaken for a legal one. The result is empirically legal on every test reported here.

What makes it work, at the level we can state

Three findings carried the campaign. Each is named here at the "what" level, with the construction reserved to the filed application.

A reservation, before routing begins. mi-go performs a resource-reservation step before any routing occurs and enforces it throughout, which removes a class of avoidable early contention. The effect is decisive and was isolated by a single-variable control on `picosoc`: the identical engine without the reservation stalled with dozens of nets left unrouted, and with the reservation, under the same configuration, it reached a fully legal route. The mechanism is claimed in the filed application and is not described further here.

A hierarchical scheme, at scale. The Titan-class results are produced by a hierarchical scheme that keeps the parallel negotiation stable as the design grows. It is claimed in the filed application and not described further

here.

The bottleneck is latency, not bandwidth. A central engineering finding is where the time goes. mi-go's core graph search is latency-bound on scattered memory access: throughput is limited by the round-trip time of dependent, irregular loads, not by raw memory bandwidth and not by arithmetic. This was confirmed across hardware, where performance tracked clock rate rather than the far larger memory bandwidth, the signature of a latency-bound workload. The speedups above came from a small, targeted change that followed directly from the measurement, with no change to what the router computes. Measurement identified the real bottleneck before any code was rewritten, and a smaller change captured the gain.

What this enables

A router that reaches verified legality in seconds to minutes on one workstation card changes how routing is used, not just how long it takes. Route iterations become interactive where overnight batch was the norm. Seed and strategy sweeps run in parallel on one GPU. Width and what-if exploration happen at conversational speed. The width characterization in this brief was itself produced that way, by routing the same design at six widths in the time a single conventional attempt would take. None of this requires the router to win every wall-clock comparison. It requires the router to finish legally on hardware an engineer already has.

Novelty and status

Single-GPU, GPU-native FPGA routing to legality at this scale is, to our knowledge, unprecedented. A literature review found no GPU router of any architecture reaching zero capacity overuse at even 50K nets. In the most recent FPGA routing contest, run with GPUs supplied to entrants, every top finisher was a CPU PathFinder variant and the sole multi-GPU entry did not finish. The demonstrated results here run to 799,403 nets. The underlying mechanisms are the subject of U.S. Patent Application No. 19/731,882, filed July 2026, with additional disclosed territory reserved for continuation practice.

Verify it yourself

A reproduction kit exists for qualified evaluators: the binary, the independent verifier, and the routing-resource graphs used above, sufficient to reproduce the legality results on your own hardware without source access, and to run the verifier against the emitted routes yourself. There is also a standing offer, which is the fastest way to evaluate this: send a netlist and a placement, and we return a verified legal route. Contact: voxell.ai/mi-go.

All figures are measured results on the hardware and designs named above. Run logs and repository evidence are available under NDA. Implementation details are intentionally omitted to protect pending intellectual property.