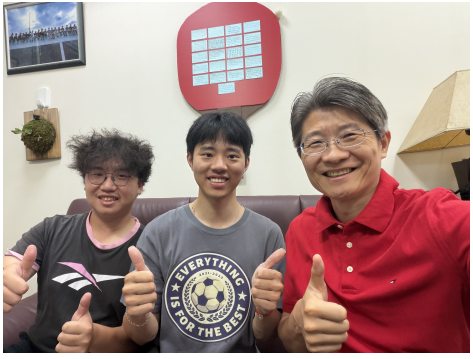
AI-Assisted AIG Optimization for IWLS 2026 Programming Contest

Wuqian Tang[1], Fangzhou Liu[2], Ruijie Wang[1]
Prof. TingTing Hwang[1], Prof. Bei Yu[2], Prof. Chun-Yao Wang[1]

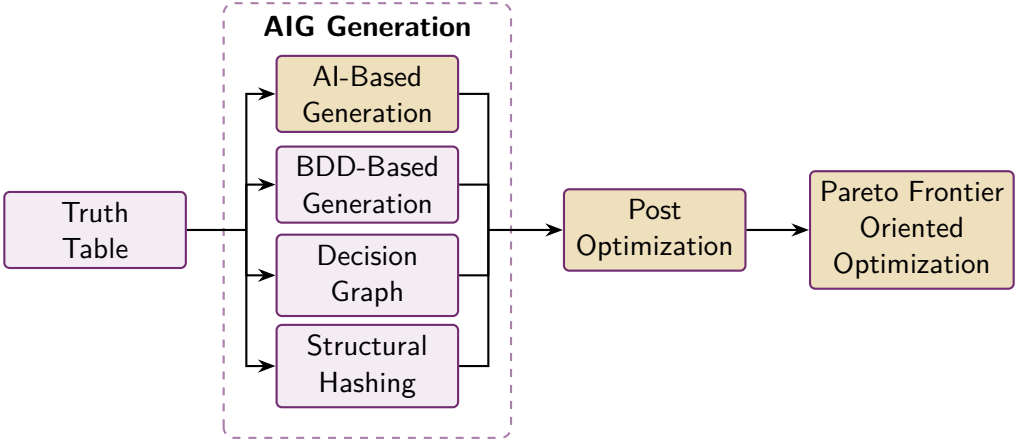
[1]National Tsing Hua University; [2]The Chinese University of Hong Kong

Jun 11, 2026

Team Members



Overall Flow



Core Components

1 AI-Based AIG Generation

AI-Based Reverse Engineering
AI-Based High-Level Verilog Rewriting

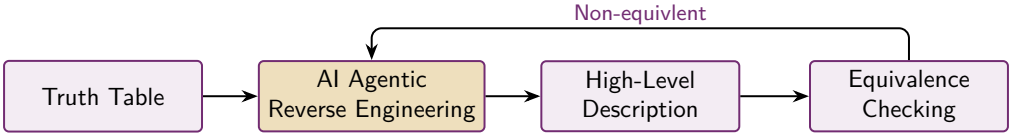
2 Post Optimization

Adaptive Evolutionary Algorithm for Logic Synthesis
ABC / ABC9 / MockTurtle / CULS

3 Pareto Frontier Oriented Optimization

Cross-AIG Structural Hashing
Cliff/Gap Repair in the Pareto Frontier

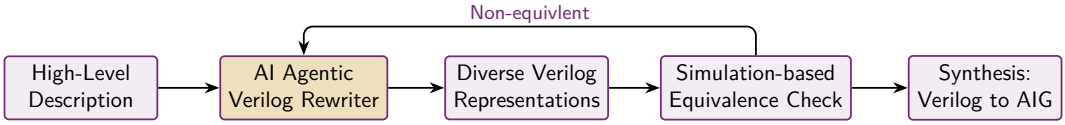
AI-Based AIG Generation: Reverse Engineering



Reverse Engineering Results

Range	Description
ex200 – ex219	BF16 Unary Operations (e.g., exp, log, sin, sqrt)
ex220 – ex239	FP16 Unary Operations (e.g., tanh, sigmoid, x^2, $1/x^3$)
ex240 – ex254	Float Conversion (e.g., fp16 to fp8, bf16 to fp8)
ex255 – ex279	Basic Arithmetic (e.g., NxN mul, N-bit div, integer sqrt)
ex280 – ex299	Structured Logic Families (e.g., Quartic Event Generator, Lossy Routing, Rot2)

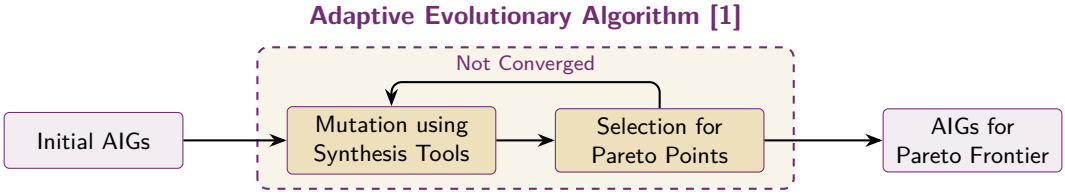
AI-Based AIG Generation: High-Level Verilog Rewriting



Diverse Verilog Representations Yield Different Pareto Points

Case	Function	RTL Strategy	Initial AIG (Level/Area)
ex226	$\sin(x)$	Exact quarter-wave key/magnitude/correction	114 / 174563
ex226	$\sin(x)$	Odd/sign-split shared BDD	29 / 46265
ex226	$\sin(x)$	Partitioned small-identity / mid-delta / high-table	66 / 44043
ex207	$\tan(x)$	Dictionary-coded tail factorization	48 / 37795
ex207	$\tan(x)$	Passthrough-front & tail BDD	33 / 48645
ex207	$\tan(x)$	Exact passthrough & flat nonlinear tail	27 / 34869

Post Optimization

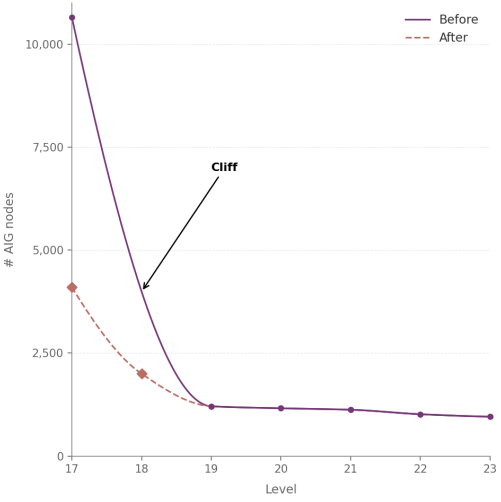


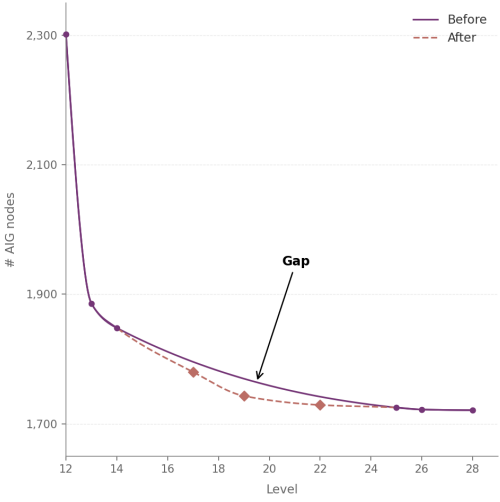
Foundational Open-Source Synthesis Tools

Tool	Our Enhancements
ABC / ABC9	Modified commands (e.g., deepsyn) to natively support pareto frontier search.
MockTurtle	Fixed Non-equivalence bugs and wrapped in an ABC-Like unified binary interface.
CULS	Ported ABC's CPU commands (e.g., rewire) to GPU CUDA versions.

[1] F. Liu, W. Tang, *et al.*, "CB-EVO: Contextual Bandit Tuning with Evolutionary Search for Logic Synthesis", *ACM TODAES*, 2025.

Pareto Frontier Oriented Optimization





Pareto-Oriented Optimization: Cross-AIG Structural Hashing

Objective

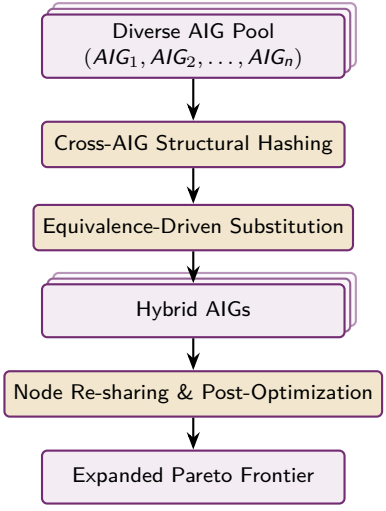
Create new Pareto points by combining structural features from a diverse AIG pool.

Sub-graph Substitution

Swap equivalent logic cones to resolve local bottlenecks (e.g., speed up critical paths or relax non-critical ones).

Functional Re-sharing

Hash the hybrids to maximize node sharing, then post-optimize to refine the expanded frontier.



Appendix: Overview of the 100 Benchmarks (ex200–ex299)

Case Range	Category	Detailed Function / Description
ex200–ex219	BF16 Unary Ops	exp, exp2, 10^x, log, log2, log10, sin, tan, sinh, tanh, sigmoid, $1/x$, x^2, $\sqrt{x}$, $1/x^2$, $1/\sqrt{x}$, x^3, $\sqrt[3]{x}$, $1/x^3$, $1/\sqrt[3]{x}$
ex220–ex239	FP16 Unary Ops	Same 20 operations as above (IEEE 754 half-precision, DAZ+FTZ)
ex240–ex253	Float Conversion	FP16 / BF16 $\rightarrow$ FP8 conversion variants (16-bit to 8-bit)
ex254	FP8 Arithmetic	FP8 addition
ex255–ex259	Basic Arithmetic	Unsigned $N \times N$ multiplier (4–8 bit)
ex260–ex264	Basic Arithmetic	Signed $N \times N$ multiplier (4–8 bit)
ex265–ex269	Basic Arithmetic	Unsigned N-bit divider with b=0 saturation (4–8 bit)
ex270–ex274	Basic Arithmetic	Unsigned N-bit squarer (8–16 bit)
ex275–ex279	Basic Arithmetic	Integer bit-by-bit restoring square root (8–16 bit input)
ex280–ex284	Unknown Family 1	Quartic GF(2) event generator + fixed $1 + z^2$ output filter
ex285–ex289	Unknown Family 2	Hamming-weight-preserving lossy routing / normalizer
ex290–ex294	Unknown Family 3	Global state/permutation-like transform with $I + R^{-2}$ skeleton
ex295, 297, 299	Unknown Family 4a	Cyclic 2-bit-symbol rot2 run/cell descriptor family
ex296, 298	Unknown Family 4b	Odd/open-boundary counterpart of the cyclic rot2 family

THANK YOU!