2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Navigating Complexity to Convergence: Formal Verification for Single Precision FMAs

Swaresh Phadke(swaresh.phadke@intel.com), Madhurima E(madhurima.eranki@intel.com)

***Abstract*- Single-precision fused multiply-add (FMA) units are fundamental building blocks in AI/ML accelerators, offering high arithmetic throughput and reduced rounding errors critical for training and inference workloads. Their increased adoption in performance-driven architecture demands rigorous and complete formal convergence to ensure functional correctness across all corner cases. However, verifying these designs poses significant challenges: FMAs feature deep arithmetic pipelines and extended exponent widths, both of which enlarge the symbolic state space and exacerbate corner-case sensitivity. Traditional verification strategies—such as assume-guarantee reasoning, helper assertions or case-splits often struggle in this context. These methods rely on clean modularity and local reasoning, which break down in FMAs due to tightly coupled Datapath stages, nonlinear interactions between operand paths, and fused rounding logic. This complexity leads to state-space explosion, fragile constraints, and frequent tool timeouts, making convergence infeasible through standard approaches.**

We demonstrate a targeted proof decomposition methodology that systematically partitions the algorithm into smaller, logically coherent verification units such as pre-alignment, multiplication, normalization, and rounding stages—based on their structural and arithmetic characteristics. A key aspect of our approach is constraint migration, which ensures that high-level design assumptions and properties are seamlessly transferred and preserved throughout the verification process at the RTL level, maintaining consistency and correctness across abstraction layers.

By aligning verification sub-problems with the solver strengths of different tools and leveraging constraint migration, we achieved exhaustive convergence in 4–6 weeks—cutting down the formal closure timeline by over 50% compared to traditional approaches that typically require 12–14 weeks for similar complexity designs. Applied to an industrial-grade single-precision FMA design with extended exponent width, our methodology demonstrates practical scalability and complete end-to-end signoff viability in real-world Datapath verification scenarios.

I. INTRODUCTION

Fused Multiply-Add (FMA's) are fundamental components in modern computing architectures, particularly in the context of AI accelerators, GPUs, and scientific computing platforms where precision and performance are critical. The Single-Precision (SP) FMA operation, which performs a multiplication followed by an addition with a single rounding step, introduces inherent challenges in both implementation and verification due to its tightly coupled arithmetic datapath and corner-case behavior involving IEEE-754 floating-point compliance. Ensuring functional correctness of such units is non-trivial, especially as various RTL optimization techniques introduce structural differences between high-level algorithmic descriptions and their low-level Register Transfer Level (RTL) implementations. These structural variations often fall outside the scope of traditional equivalence checking methods, making formal datapath verification a complex and convergence resistant task.

C-to-RTL formal equivalence checking offers a methodology to rigorously compare the behavior of a high-level C/C++ specification against its synthesized RTL implementation. Unlike traditional RTL-to-RTL equivalence, which expects near-identical structural mapping, C-to-RTL supports semantic equivalence despite significant microarchitectural transformations. This makes it particularly attractive for datapath-heavy components where the RTL may be heavily pipelined or optimized, while the C model remains flat and algorithmic. However, this added flexibility comes with verification complexity—particularly when control and data paths diverge due to custom datapath optimizations and structural variations in implementation across functionally equivalent blocks.

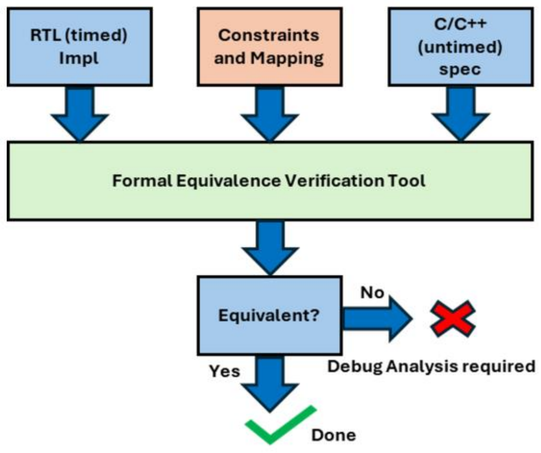

Fig 1. Datapath Formal Equivalence

In this work, we investigate the application of C-to-RTL equivalence checking to formally verify the correctness of SP FMA datapaths. We identify key bottlenecks that hinder convergence, including exponent-dependent path conditions, mantissa alignment and normalization subtleties, rounding-mode interactions, and corner-case divergences. To address these challenges, we present a systematic approach combining constraint migration & targeted proof decomposition. Our findings highlight how these methods can reduce solver load, improve proof convergence, and ensure bit-level compliance across a wide range of inputs and design optimizations.

II. RELATED WORK

The formal verification of floating-point and Datapath-intensive designs has received substantial attention, especially as modern architecture grows more complex [1]. Pouarz and Agrawal [2] demonstrated exhaustive FMA validation using Sequential Equivalence Checking (SEC), though their approach relies heavily on strict structural alignment. Wang and Zhu [3] proposed recursive reasoning for C-to-RTL verification, yet it involves significant manual decomposition. More recent works [4] explore convergence acceleration via multiple formal tools but remain tool-specific and lack abstraction-agnostic proof integration. In contrast, our approach leverages tool-independent proof decomposition and constraint migration to achieve scalable and robust formal convergence for complex arithmetic pipelines. Our methodology thus enables scalable and modular formal closure without being constrained by a specific tool or abstraction alignment requirement

III. APPLICATION

1. Design Overview

A single-precision Fused Multiply-Add (FMA) unit executes the operation (A × B) + C as a single, uninterruptible pipeline with a unified rounding step, adhering to IEEE 754 specifications. Each operand is represented in 32-bit IEEE 754 format, comprising a 1-bit sign, 8-bit biased exponent, and 23-bit fraction field with an implicit leading 1 for normalized values. This fused execution enhances numerical precision and computational efficiency by eliminating intermediate rounding.

The operation begins by unpacking inputs A, B, and C into their sign, exponent, and mantissa components. The mantissas of A and B are multiplied to produce a 48-bit product. The resulting exponent is computed by adding the exponents of A and B and adjusting for bias and normalization. Simultaneously, operand C is aligned by shifting its mantissa based on the exponent difference with the product. Guard, round, and sticky bits are introduced to preserve precision. The aligned mantissas are then added or subtracted, depending on operand signs, resulting in an extended-width intermediate sum. This sum undergoes final normalization and a single rounding step (typically round-to-nearest-even), after which the sign, exponent, and mantissa are reassembled into a 32-bit IEEE 754 compliant result as shown in figure 2 below.

In contrast to conventional single-precision FMA units, the design utilized in this work incorporates an internally extended exponent path, enabling a substantially increased dynamic range for intermediate computations beyond the constraints of the IEEE 754 single-precision format.

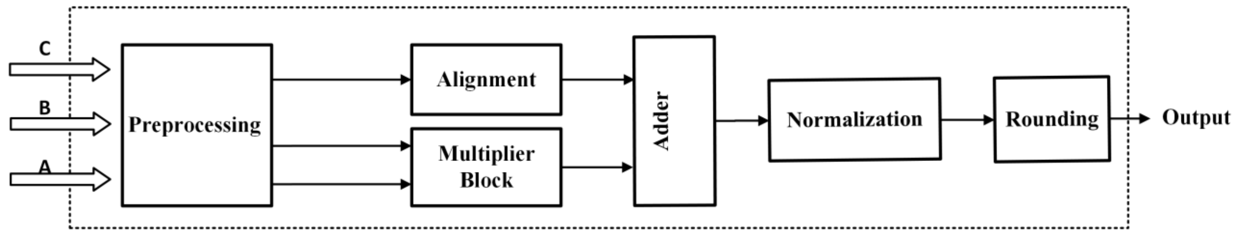

Fig 2. FMA Block Diagram

2. Problem Statement

The complexity of formally verifying single-precision FMA units is driven by both the breadth of the Datapath—up to 49 bits internally—and the intricacies of floating-point logic. The Datapath includes tightly coupled stages for normalization, operand alignment, rounding, and IEEE 754 exception handling, each of which introduces data dependencies that are challenging to model symbolically.

2.1 Numerical Complexity

The numerical range of standard single-precision floating-point numbers spans approximately 1.18×10^{-38} to 3.4×10^{38}. Increasing the number of exponent bits exponentially expands this dynamic range while maintaining a fixed 23-bit mantissa. This broader range helps mitigate early overflows and underflows, particularly in deeply pipelined floating-point units used in matrix and tensor computations. However, it also increases rounding sensitivity near the extremes of the range, introducing more intricate corner-case behaviors. Since precision remains constant, the extended range leads to a denser distribution of representable values near boundaries, complicating normalization and rounding logic during verification.

Table 1. Numerical Complexities in SP FMA

Complexity	Cause	Example Behavior
Single rounding	IEEE-754 fused rule	fma ≠ mul + add
Denormal	Small mantissa/exponent	Requires custom LZC/LZA
Cancellation	Opposite signs, same magnitude	ULP-sensitive result
Overflow/Underflow	Extreme exponents	Generates INF or 0
NaNs/Infinities	Special IEEE-754 rules	NaN propagation
Exponent alignment	Different scales	Shifts, sticky/guard bits
Rounding mode	Dynamic rounding	Affects tie-breaking and edge cases

2.2 Datapath Verification Complexity

Verifying Datapath designs with extended exponent widths presents several core challenges. First, the enlarged exponent increases the symbolic state space exponentially, placing a heavy burden on formal engines and making exhaustive exploration difficult within practical bounds. This results in longer convergence times and greater tool resource consumption. Second, the broader dynamic range introduces a denser set of numerical corner cases—such as denormal, overflows, and precision-critical rounding scenarios—that require fine-grained assertions and extensive constraint modeling. These edge cases are harder to isolate and verify due to their sensitivity to operand patterns and timing behavior. Constraint tuning and migration also become increasingly complex. As the operand range expands, constraints must be accurately propagated across pipeline stages and verification blocks, or they risk over-constraining or under-constraining the design—both of which hinder convergence or lead to missed bugs. Finally, monolithic proof strategies that treat the entire Datapath as a single verification problem often become infeasible. They struggle to scale under the weight of growing complexity. This necessitates the adoption of more modular and scalable methods, such as targeted proof decomposition and constraint migration, to achieve formal closure in high-complexity arithmetic designs.

Table 2. Datapath Formal Verification Complexities in SP FMA using C2RTL

Challenge	Description	Effects
Structural Mismatch	RTL is pipelined and optimized; C is functional and sequential.	Requires signal alignment and structural correlation for convergence.
Operand-Dependent Optimizations	RTL uses gating, early-out, or fast paths.	Causes path divergence; C-to-RTL needs input partitioning to isolate cases.
RTL Optimizations	Techniques like partial product sharing or custom rounding logic.	Bit-accurate C uses full computation; correlation points are needed.
Corner Case Convergence	Cases like denormal, cancellations, and rounding ties.	Solver complexity spikes: input partitioning or helper lemmas often needed.
Solver Scalability	Wide Datapath and deep pipelines stress formal tools.	Proofs may timeout without decomposition or helper strategies.

3. Methodology

As a first step, we execute the end-to-end C-to-RTL equivalence check without applying any convergence strategies, allowing the formal engines to explore the design exhaustively for a bounded time window, typically up to 48 hours until the proof reaches an inconclusive state.

Eventually, we shift our focus to corner-case scenarios that structurally bypass the mantissa multiplication logic, such as zero, NaN etc. However, due to the FMA unit's highly optimized microarchitecture targeted for precision-critical and robustness-sensitive workloads standard convergence strategies prove insufficient to establish full end-to-end equivalence, even for semantically simpler bypass scenarios where mantissa multiplication is elided.

When bypass scenarios fail to converge, A deeper analysis of the design revealed that the primary sources of complexity impeding formal closure originated around the FMA boundary

3.1 Constraint Migration

In complex Datapath architectures, Fused Multiply-Add (FMA) units are rarely verified in isolation from their operational context. Instead, they often function as subcomponents within broader execution units that implement full instruction semantics, including decoding, operand formatting, exception management, and architectural flag generation. While the FMA's core responsibility is to perform a numerically correct A × B + C operation under IEEE-754 semantics, its behavior can be significantly influenced by external constraints originating from upstream or downstream logic.

These contextual constraints may include:

- Operand range limitations, such as disallowing denormal inputs in certain micro-ops.
- Exception gating, where invalid operations are architecturally masked and should not trigger local assertion failures.
- Encoding-specific behaviors, such as specific NaN payload values, zeroing of inactive vector lanes, or fused instructions with implicit operand patterns.

Although these constraints are logically external to the FMA datapath, they directly influence its local correctness and the reachability of its internal states. If not accounted for, the formal tool may waste effort exploring unreachable states, leading to poor convergence, false counterexamples, or timeouts.

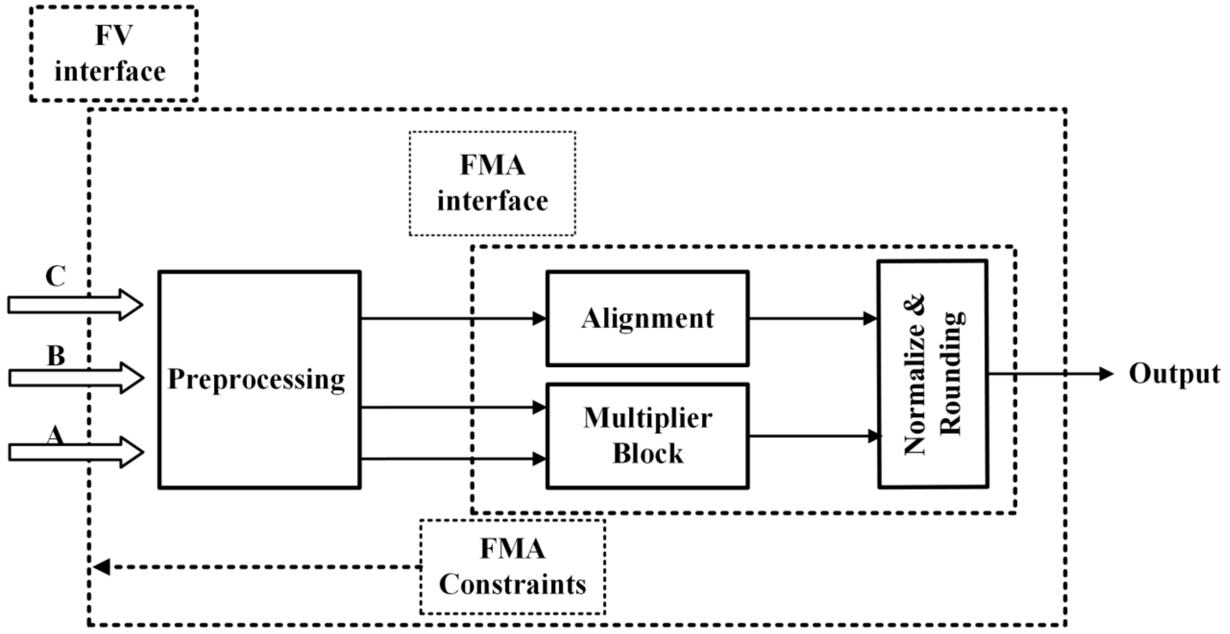

Fig 3. Constraint Migration

To address this, a constraint migration methodology is employed. This involves:

- Identifying external constraints relevant to the FMA's operational scope.
- Validating these constraints at the interface boundary of the FMA using local formal properties (e.g., assume/guarantee or assertion checks).
- Once proven valid, removing the internal duplication of these constraints within the FMA's formal model, and instead applying them externally in the formal environment or testbench.

This constraint migration shifts complexity out of the FMA module and into the formal harness, which helps the solver:

- Prune unreachable symbolic states early
- Avoid unnecessary case splits based on invalid input combinations,
- Enhance convergence by restricting analysis to valid and functionally relevant input spaces

3.2 Targeted Proof Decomposition

After applying state space reduction techniques, the verification effort is partitioned based on Datapath behavior. For bypass or special-case scenarios—where the mantissa multiplier is structurally bypassed (e.g., zero, NaN, or subnormal operands)—standard convergence strategies such as multi-level case splits on bypass conditions are sufficient to achieve full proof. The bypass conditions of the mantissa multiplier depend on specific combinations of the exponents and mantissas of the multiplier inputs A and B. Initial convergence attempts using a single-level case split on inputs proved insufficient. Introducing a second-level case split focused on exponent patterns enabled full proof convergence for all bypass scenarios within 5-6 hours. For non-bypass paths involving full mantissa multiplication, we leverage multiplier-aware formal engines to establish correctness of the mantissa multiplier block in isolation.

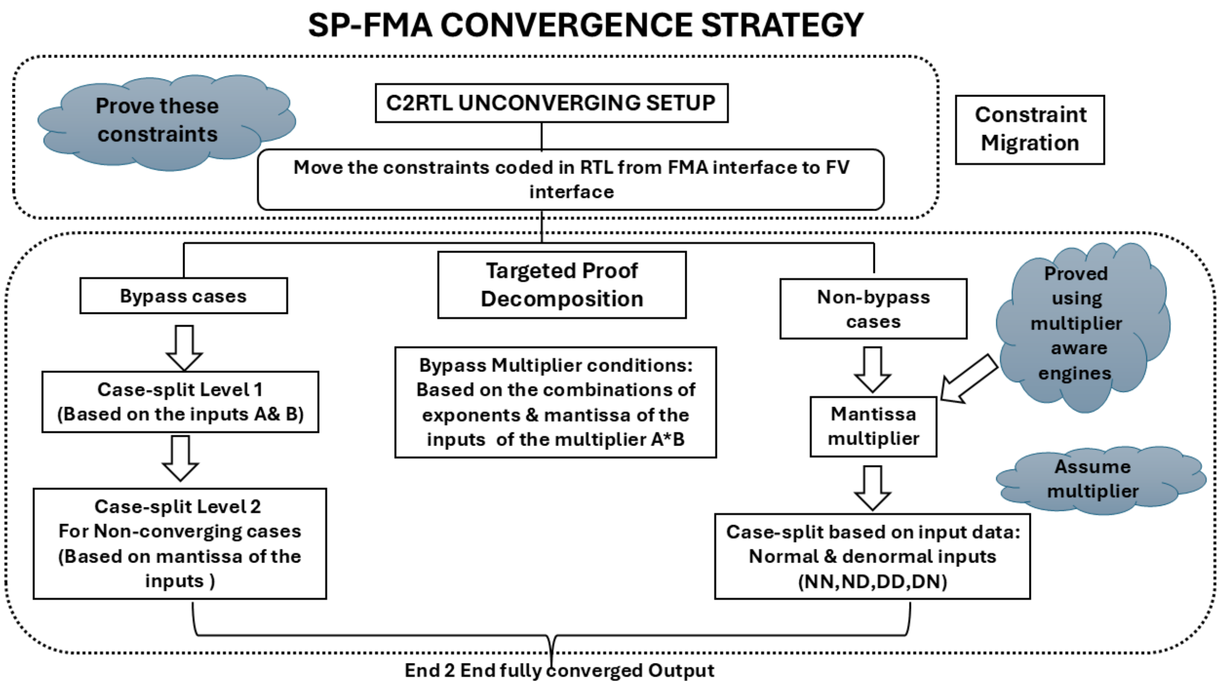
SP-FMA CONVERGENCE STRATEGY
Prove these constraints
C2RTL UNCONVERGING SETUP
Move the constraints coded in RTL from FMA interface to FV interface
Constraint Migration
Bypass cases
Targeted Proof Decomposition
Non-bypass cases
Proved using multiplier aware engines
Case-split Level 1 (Based on the inputs A& B)
Bypass Multiplier conditions: Based on the combinations of exponents & mantissa of the inputs of the multiplier A*B
Mantissa multiplier
Assume multiplier
Case-split Level 2 For Non-converging cases (Based on mantissa of the inputs)
Case-split based on input data: Normal & denormal inputs (NN,ND,DD,DN)
End 2 End fully converged Output

Fig 4. SP-FMA Convergence Strategy

Once verified, the multiplier is abstracted via an assume-guarantee interface, enabling proofs to proceed over a simplified model. Additionally, single-level case splits over multiplier operand classes (e.g., normal vs. denormal inputs) proved effective in decomposing the remaining state space, ultimately yielding complete proof convergence across all scenarios.

IV. RESULTS

The proof decomposition and constraint migration methodology were evaluated on a single-precision FMA design with extended exponent width representative of GPU datapath workloads. The setup included IEEE 754-compliant rounding logic, exception handling, and operand bypass paths. The proposed methodology is tool agnostic and verification was performed on identical compute infrastructure before and after methodology adoption.
Constraint migration reduced solver state-space, enabling earlier pruning of unreachable symbolic states and eliminating redundant case splits. Further decomposing the verification problem into solver-friendly partitions resulted in complete convergence of the FMA.
Notably, special-case handling (e.g., NaN, Inf, subnormals) benefited the most from multi-level case-splitting, where first-level splits on input operand types were supplemented by second-level splits on exponent patterns.
We compared the partitioned approach against standard verification strategy using key performance metrics as summarized in table 3.

Table 3. Comparison of Standard Convergence Methodology vs Adopted Methodology

Metric	**Standard Convergence Techniques**		**Demonstrated Methodology**	
Bypass cases	**Case-splits**		**Case-splits**	
	77	Partial. 20/77 case-split cases converged while 57/77 did not converge in 20+ hours	**156**	Complete. 4-5 hours for all the cases
Non-Bypass cases	**Did not converge in 24 hours**		**Complete. 8-12 hours for all the cases**	
Corner-case coverage	Partial		100%	
Effort required	Can take **12-14** weeks of effort for full proof		Can be completed within **4-6** weeks.	

V. CONCLUSION & FUTURE WORK

The formal verification of compute-intensive arithmetic blocks such as single precision FMA units remains a critical bottleneck in achieving verification signoff for modern, performance-driven Datapath architecture. These designs are inherently complex due to deep arithmetic pipelines, rounding logic, and architectural interdependencies. Traditional convergence techniques often struggle to scale with such complexity. In this work, we employed constraint migration strategies to reconcile specification-level constraints with RTL-level behavior, ensuring alignment throughout the C-to-RTL equivalence flow. We also leveraged a proof decomposition methodology, partitioning the FMA algorithm into logically distinct verification blocks aligned with solver strengths. Our results demonstrate that this targeted approach significantly reduces convergence time, minimizes debug iterations, and improves corner-case coverage - making it a viable and production-friendly strategy for Datapath verification signoff.

In future, the methodology can be adapted to support double-precision and mixed-precision FMA implementations, which present additional challenges due to wider datapaths and increased operand variability. In addition, automation of the constraint migration process would further reduce convergence time and improve reusability across verification projects, enabling faster onboarding of new designs. Finally, the decomposition framework can be applied to other compute-intensive architectural blocks, such as tensor cores and matrix processing units, where similar complexity and convergence bottlenecks are encountered.

VI. REFERENCES

[1] H. Foster, Trends in Functional Verification-A 2014 Industry Study, Design Automatic Conference (DAC) 2015

[2] T. W. Pouarz and V. Agrawal, Efficient and Exhaustive Floating-Point Verification Using Sequential Equivalence Checking, DVCon 2017.

[3] J. Wang and J. Zhu, Conquer Difficult C-RTL Formal Verification Problems Using Recursive Compositional Reasoning, DAC 2017.

[4] Revolutionizing Proof Convergence for Algorithmic Designs by Combining Multiple Formal Verification Tools, DVCon 2024.