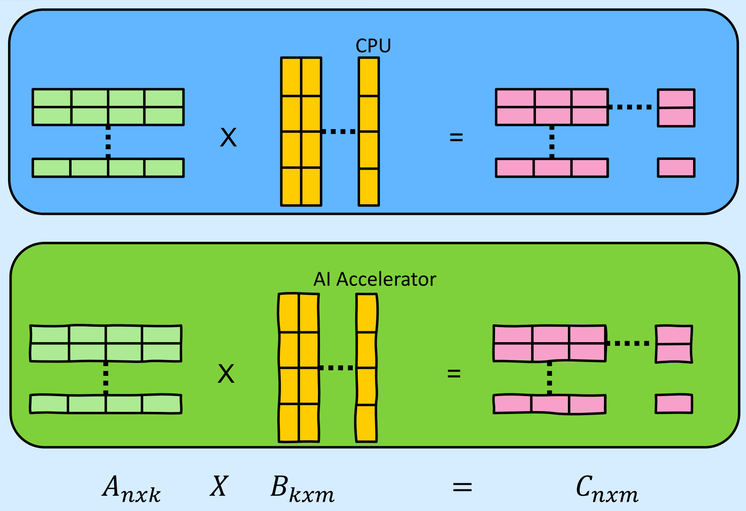
Precision Unveiled: Formal Methods in Dot Product Accumulate ULP Analysis

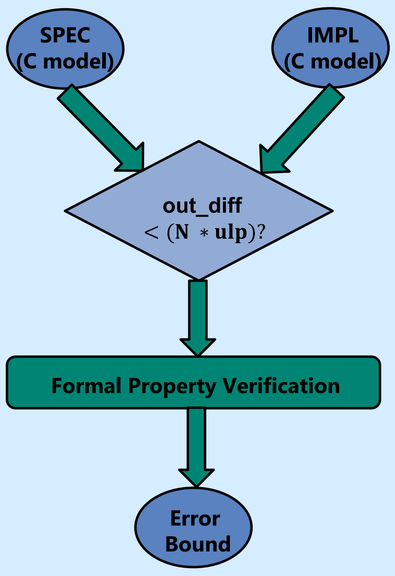
Suraj Kamble, Mohit Choradia, Vichal Verma, Kyung-Nam Han

Microsoft Technology Pvt Ltd

Introduction/ Problem Statement

- **Dot Product Accumulate**
  - A dot product multiplies corresponding elements of two vectors and sums the results. It's a core operation in machine learning, especially in neural networks and matrix computations.
- **Unit in the Last Place (ULP)**
  - Measure of the smallest possible difference between two representable floating-point numbers near a given value.
  - Quantify how close a computed result is to the mathematically correct one.
  - High ULP errors can lead to significant deviations in dot products, affecting the performance and reliability of AI models.
- **Precision Discrepancy Across Platforms**
  - ML inference algorithms show varying numerical precision when executed on CPUs versus accelerators due to differences in floating-point formats and execution models.
- **Need for Acceptable Error Bounds**
  - It is essential to define and enforce acceptable precision error margins to ensure consistent and reliable inference across heterogeneous hardware.
- **Role of Formal Verification**
  - Formal methods can rigorously model and bound precision errors, providing guarantees that inference results remain within safe and acceptable limits.

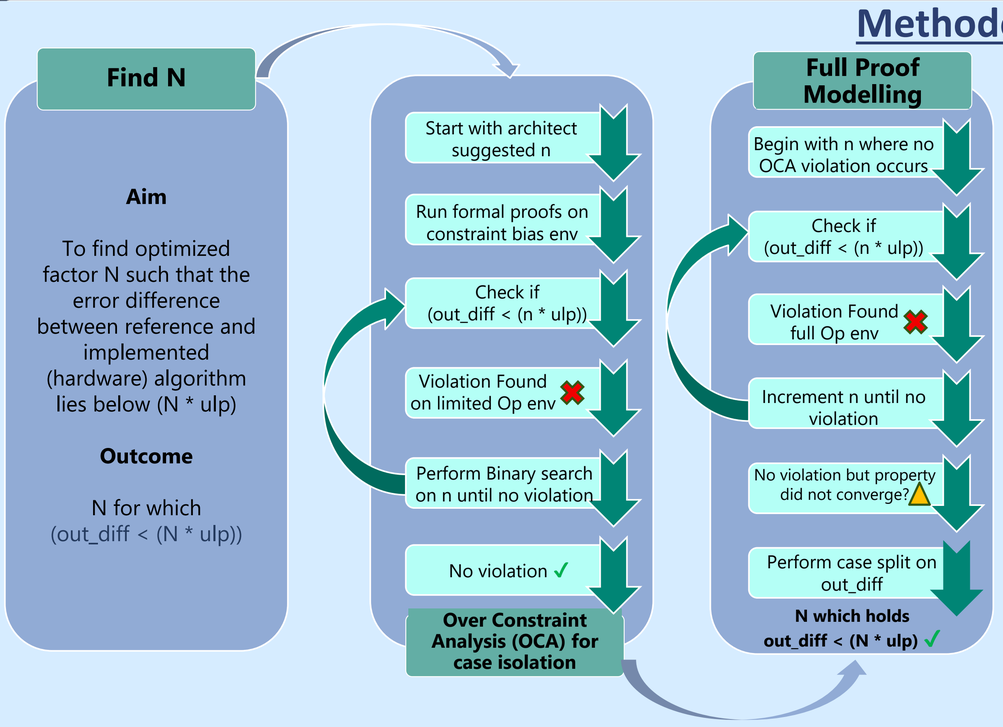




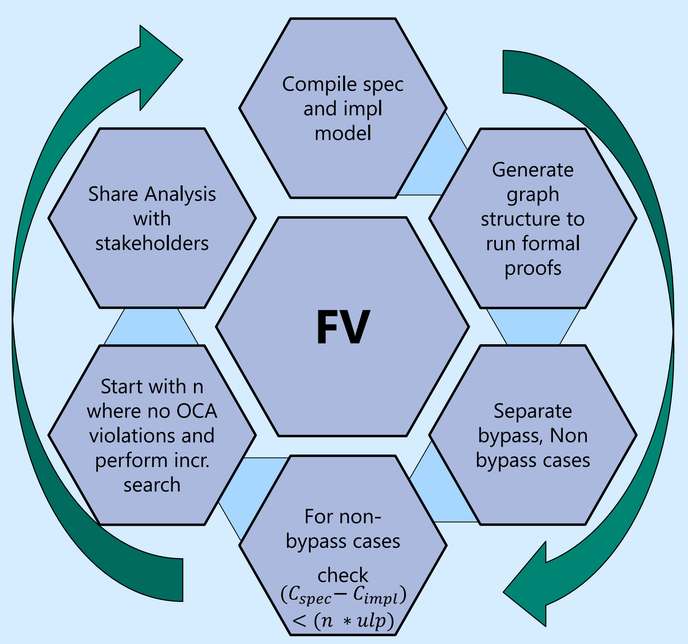
Formal Methods

Challenge & Impact	Solution & Benefit
Multiple MulAdd ops and symbolic error bound forms huge formal model results blocking of proof execution.	hardcode (assume) error bound to enable proofs
Exhaustive trials for error bound are resource-heavy	Use OCA for case isolation strategy for efficient convergence

Methodology and Implementation Details



- **Architectural analysis**
  - The architect- documented n based on design intent and expected behavior, guiding initial OCA Formal runs.
- **Over constraint Analysis (OCA) for case isolation**
  - Leverages its speed to quickly eliminate invalid bounds, reducing the search space for full proof formal verification.
- **Binary search on OCA Environment**
  - Efficiently narrows down the viable range of n, avoiding brute-force exploration.
- **OCA for case isolation-pruned formal analysis**
  - Ensuring that only promising values of n are subjected to deeper analysis.
- **Case splitting and incremental search**
  - Formal convergence techniques are applied only when necessary, optimizing proof effort.

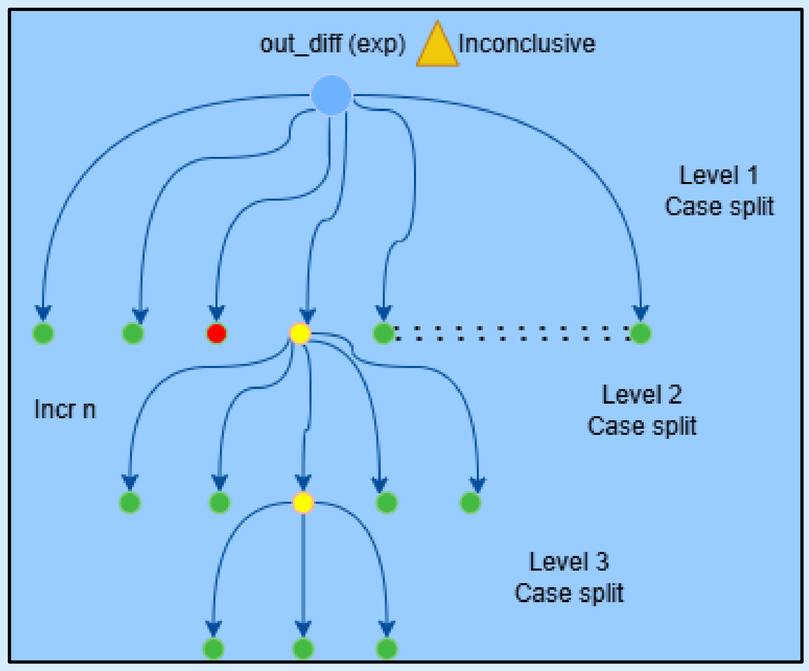


- **Over Constraint Analysis (OCA) for case isolation** enables rapid validation when input space is limited, making it ideal for early error bound exploration. **Full proof formal verification** ensures exhaustive correctness and starting it from OCA-cleared bounds optimizes time and memory utilization. Architect-suggested values guide this flow, aligning verification with design intent and accelerating hardware development cycles.

Results

The methodology aimed to rigorously constrain the error in dot product computations to remain within the architect-defined bound of $n \times ULP$. The following key outcomes were observed:

- **Architect-Suggested Bound Violations**
  - Initial OCA runs using the architect-defined value of n revealed counterexamples (CEX) where the error exceeded (n × ULP).
  - These violations were shared with the architect to reassess the feasibility of the original bound.
- **Binary Search for Fast Convergence in OCA for case isolation**
  - Quickly finds the smallest n where no constraint bias violations occur. Speeds up validation and memory utilization before full proofs.
- **Case Split Strategy in Formal Proofs**
  - When end-to-end proofs didn't converge, splitting cases based on exponent of algorithms output difference value helped isolate hard scenarios. This revealed violations in specific cases, guiding deeper analysis and improving overall proof coverage.
- **Formal in early design cycle**
  - It enables early validation, reduces risk, and improves overall design system quality.
- **Future Work :- Algorithmic Error Attribution**
  - By limiting operations, the experiment can be performed to pinpointed high-error segments in the algorithm—guiding architects to focus optimization where it matters most.



Conclusion

This work establishes a formal upper bound on output error based on architectural specifications, enabling reliable system design. OCA guides formal analysis efficiently, and the resulting bound helps architects iteratively refine specifications. **Formal methods thus evolve from bug detection to shaping robust hardware early in the design cycle.**

References

1. Goldberg, D. (1991). "What Every Computer Scientist Should Know About Floating-Point Arithmetic." *ACM Computing Surveys*, 23(1), 5-48.
2. Robert D. Skeel. Roundoff error and the patriot missile. SIAMNews, 25(4):11, Jul.1992.
3. Erik Seligman, Tom Schubert, and MV Achutha Kiran Kumar. 2015. Formal verification: an essential toolkit for modern VLSI design. Morgan Kaufmann.