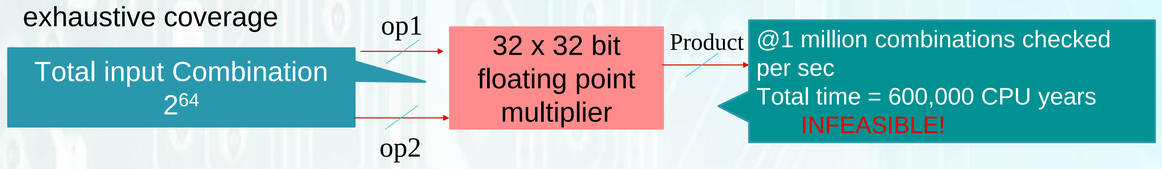
# Accelerating Datapath Verification using Formal Techniques

**Bhavya Dasari, Jaaneshwaran A, Karthik Rajakumar, Craig Deaton, Uday Kumar**

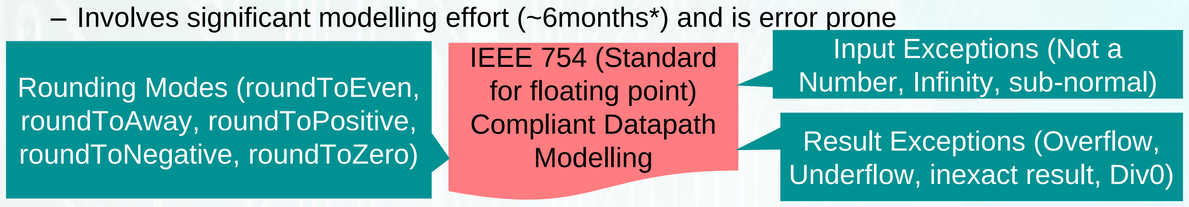
60 DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW

## 1. Motivation & Problem Statement (1/2)
### CPU Datapath Verification

- Modern CPUs have multiple hand-optimized Datapaths
  - To further the application performance
  - Achieve stringent low power targets
- Datapath verification is complex due to the number of combinations to be checked
- Simulation based verification / Dynamic verification (DV) is infeasible to provide exhaustive coverage

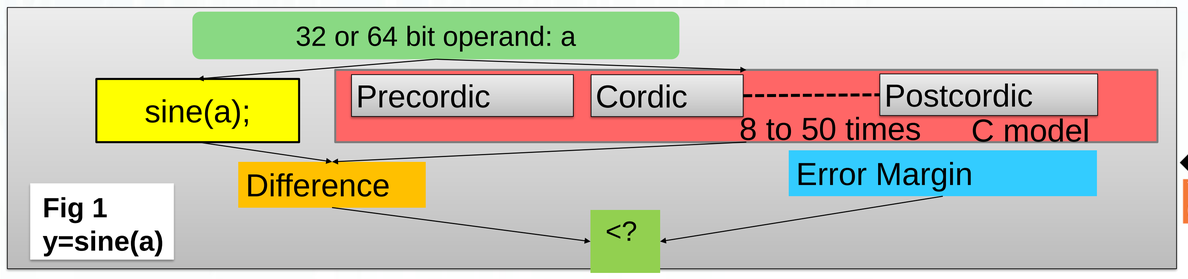



- Formal property verification (FPV) provides comprehensive coverage
  - Involves significant modelling effort (~6months*) and is error prone

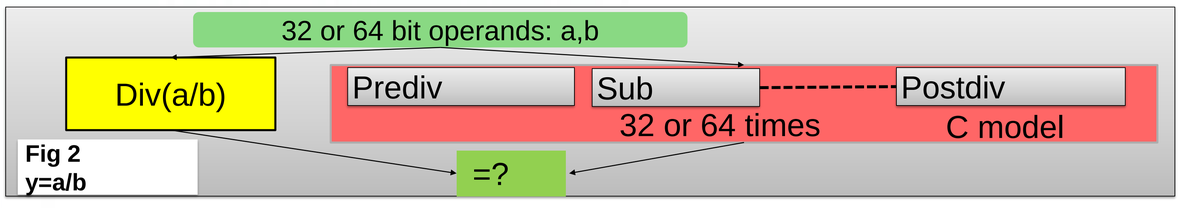



## 2. Motivation & Problem Statement (2/2)
### Architectural Evaluation

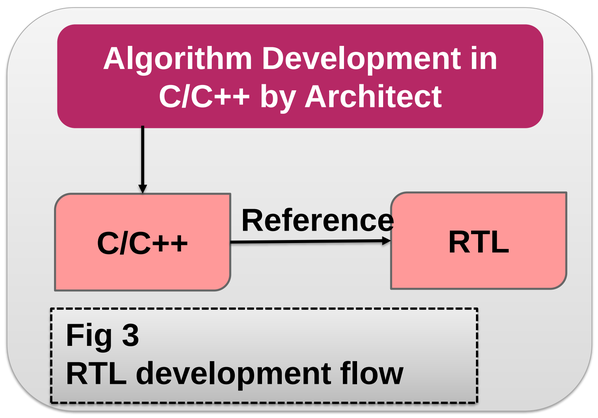
- Custom Datapaths by Architects not IEEE compliant and have some error margin (Fig 1: Sine)
  - CORDIC ↑ → Accuracy ↑
- Area/performance advantage with acceptable tolerance
- Need to ensure result difference is within error margin




- Complex operations split into multiple single cycle operations (Fig 2: Division)
- Enables shorter non-interruptible boundary
- Need to ensure C model is correct

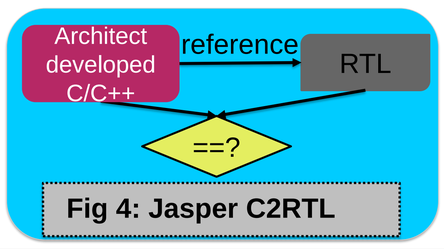


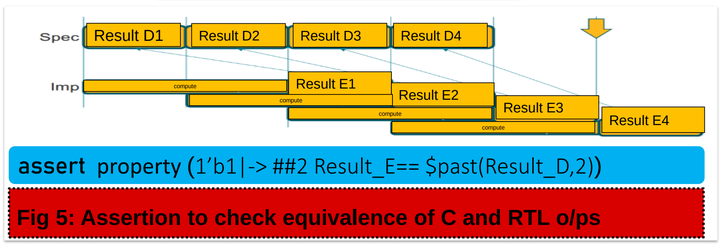

- C code used as reference for RTL development
- **Prior art-** C equivalence can be verified
  - –For 32 bit operands – by iterating 2^32*2^32 values, **slower**
  - –For 64 bit operands – impossible to iterate all 2^64*2^64 values
- **Proposed methodologies using Jasper C2RTL Formal app will help**
  - **Verifying Datapath Intensive RTLs**
  - **Architectural Evaluation**




## 3. Proposed Solution (1/3)

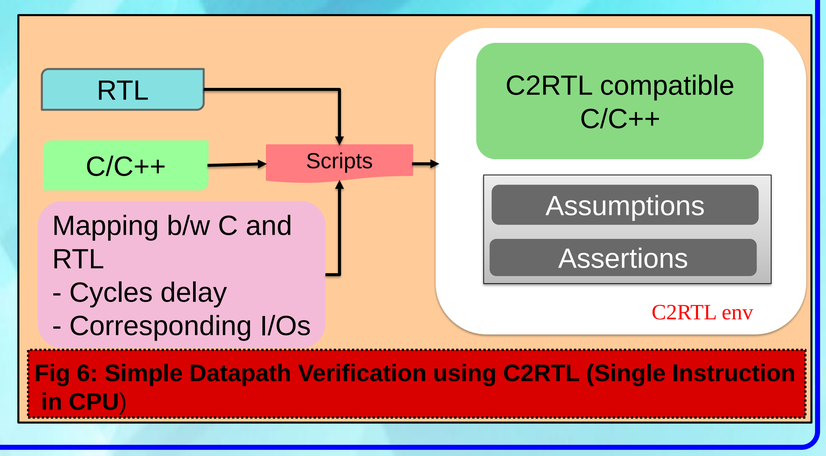
- Re-use C model by Architect -> No modelling effort
- **Jasper C2RTL app** checks equivalence b/w C and RTL (Fig 4)
  - Built on FPV app, provides exhaustive coverage
  - Better computational engines
  - Convergence is faster




## 4. Proposed Solution (2/3)

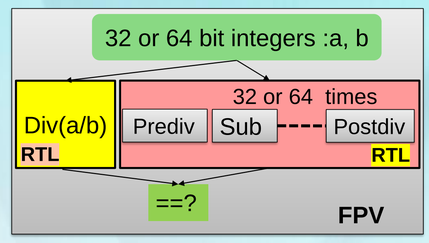
- Automation developed to generate C2RTL environment (Fig 6)
  - Reduces verification time
- Inputs needed for automation
  - C code, RTL
  - Instruction execution delay
  - Corresponding Input/Outputs b/w C/RTL

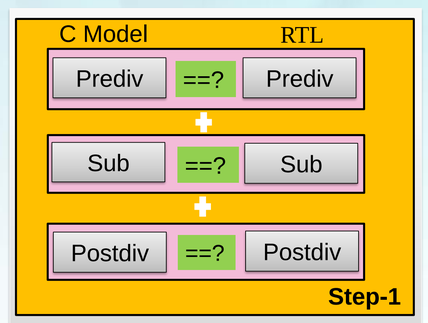


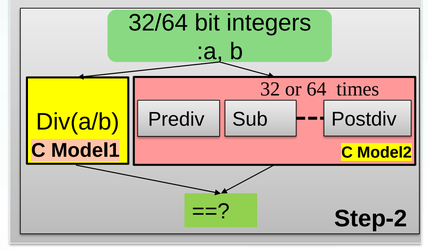

## 5. Proposed Solution (3/3)

- Sequence of instructions for 32 or 64 bit integer division
  - Prediv -> Sub - - - (32 or 64 times) -> Postdiv
- Huge sequential depth resulting in state space complexity
- General verification approach (Fig 7) using FPV has **convergence** issues
- **Solution - Move the convergence problem from RTL to C**
  - **Step-1**: Prove equivalence b/w C and RTL for each of the instructions
  - **Step-2**: Prove equivalence b/w two C models
    - C Model 1: Standard division is calculated
    - C Model 2: Sequence of instructions used to compute division
- Convergence can be achieved faster
  - Takes 1 cycle to compute both sequence of instructions and actual instruction

**Step-2 can be used for Architectural evaluation**

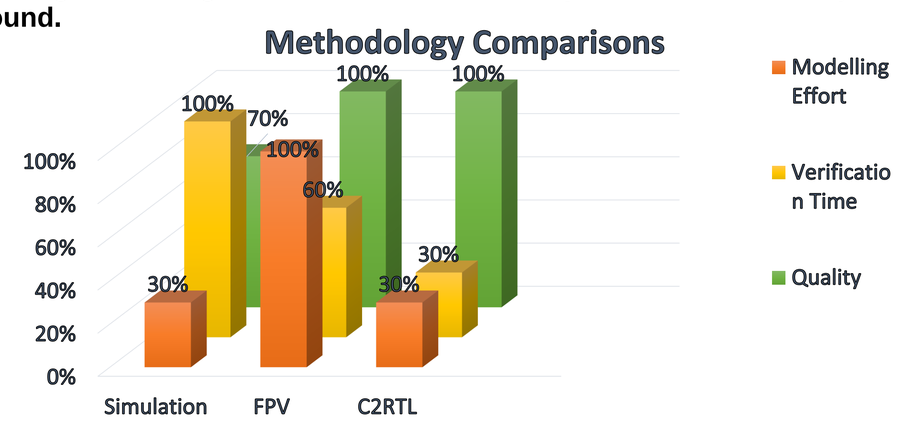



## 6. Evidence and Results

- The highlighted novel methods were deployed to verify complex operations such as floating point and trigonometric math units in next gen CPU
- **Multiple bugs including corner case scenarios of (1 in 10 million cases) were found.**




**Methodology comparisons**

- **Quality :** C2RTL, FPV – 100% as both exercise all possible inputs including corner case
- **Modelling effort:** C2RTL, Simulation has less modelling effort as C model is readily available
- **Verification Time:** C2RTL is relatively faster compared to FPV/Simulation
  - Automated proposed methodologies & Better engines compared to FPV.

## 7. Conclusions

- Proposed methodologies can be used in the **Verification** of any Datapath intensive RTL.
- Additionally, these methodologies can make **Architectural evaluation** faster and easier.
  - Ensures reference C/C++ model used for RTL development is Bug free.
  - Avoid late bugs.
- Overall **~50-70%** reduction in Verification effort can be achieved using proposed methodologies

## Acknowledgements

- Naveen Kothuri, Prasanth Viswanathan Pillai of Texas Instrument for their support in this work

TEXAS INSTRUMENTS