Formal Verification of Floating-Point Matrix Multiply-Add Designs

Emiliano Morini, Christopher Poole

Accelerated Computing Systems & Graphics

intel

Floating-Point Matrix Multiply-Add

$$\underset{R^{m,p}}{\begin{pmatrix} r_{11} & r_{12} & \dots & r_{1p} \\ r_{21} & r_{22} & \dots & r_{2p} \\ \vdots & \vdots & \dots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mp} \end{pmatrix}} = \underset{A^{m,n}}{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}} \cdot \underset{B^{n,p}}{\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \dots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{pmatrix}} + \underset{C^{m,p}}{\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \vdots & \vdots & \dots & \vdots \\ c_{m1} & c_{m2} & \dots & c_{mp} \end{pmatrix}}$$

One of the most common operations required by algorithms, widely used in Machine Learning, Computer Graphics and Network Theory.

$$r_{ij} = \sum_{k=0}^{n} a_{ik} \cdot b_{kj} + c_{ij}$$

DPn.5

$2n + 1$ floats with precision t
4 rounding modes $\Rightarrow$ $2^{(2n+1)t+2}$ input values for each r_{ij}

Hard to verify due to the **massive input space** and the **arithmetic** involved

DPn.5	DP4.5	DP4.5	DP4.5	DP8.5	DP8.5
Precision	16	32	64	32	64
Input Space					

Proof of Correctness: Equivalence Checking (EC)

Formally check that each output r_{ij} is equivalent to the output of a validated independent DPn.5 C++ model

TRADITIONAL METHODOLOGY

1. Complete the C2RTL proof for r_{11}
2. Replicate the same proof for all the other r_{ij}

Serial approach
(no progress until step 1 is completed)
+
Risk not diversified

NEW "SYMMETRY INVARIANCE" METHODOLOGY

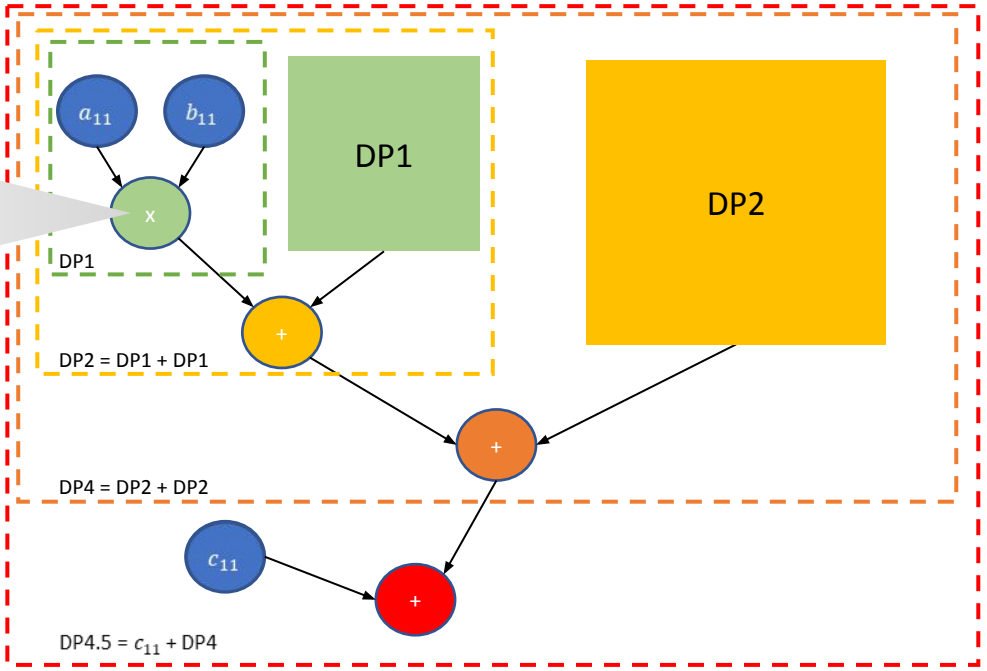
1. Complete the C2RTL proof for r_{11}
2. Prove that the other values r_{1j} in the first row are computed as r_{11} with an RTL2RTL **(column symmetry invariance for the first row)**
3. Prove that the values r_{ij} in the following rows are computed as r_{1j} with an RTL2RTL **(row symmetry invariance for other rows)**

Parallel approach
(the 3 steps progress at the same time)
+
Risk diversified

C2RTL for r_{11}

No commercial equivalence checking tool converged out of the box,
proof decomposition required

Automatic Theorem Proving FV with a commercial EC tool



Internal equivalence points between C and RTL are unlikely, due to the RTL optimizations.

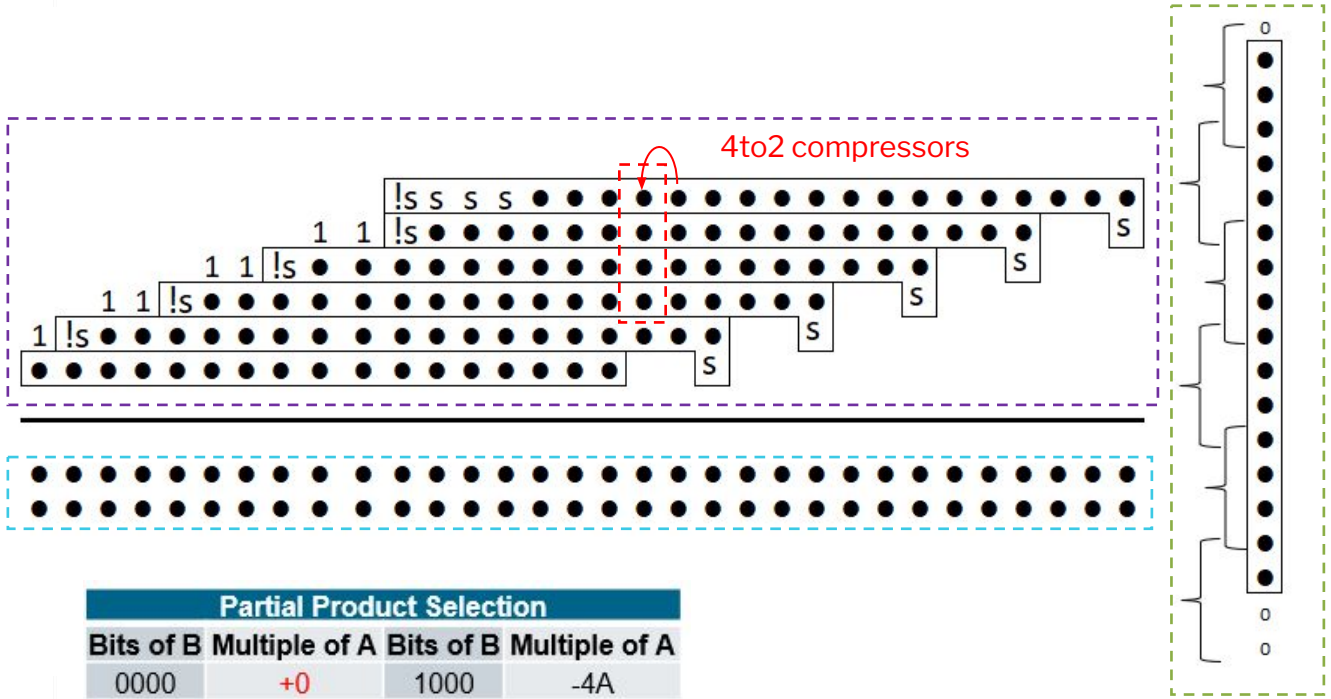
Relations among multiple signals are more likely:

```
impl.prod ==
(spec.prod << spec.renorm.val)
```

Black-box the blocks that are not part of the sub-proof you are working on

Cut the input of the block under test, assuming what has been proven in the previous level of blocks

Automatic Theorem Proving for Multipliers



Partial Product Selection			
Bits of B	Multiple of A	Bits of B	Multiple of A
0000	+0	1000	-4A
0001	+A	1001	-3A
0010	+A	1010	-3A
0011	+2A	1011	-2A
0100	+2A	1100	-2A
0101	+3A	1101	-A
0110	+3A	1110	-A
0111	+4A	1111	-0
S = 0		S = 1	

Prove the correctness of the multiplication by checking these intermediate steps:

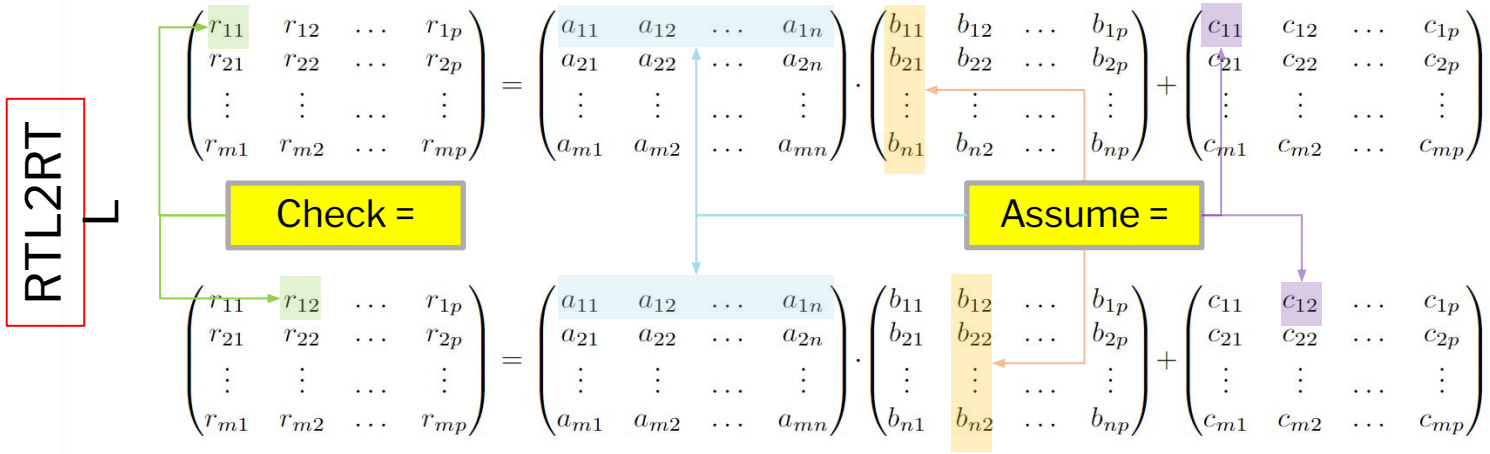
- Encoding
- Array creation
- Array reduction
- CSA form

$$a \cdot b = a \cdot \sum b_i \cdot 2^i = a \cdot \sum(-4b_{3i+2} + 2b_{3i+1} + b_{3i} + b_{3i-1})2^{3i}$$

Presented at SNUGSV23

RTL2RTL Symmetry Invariance

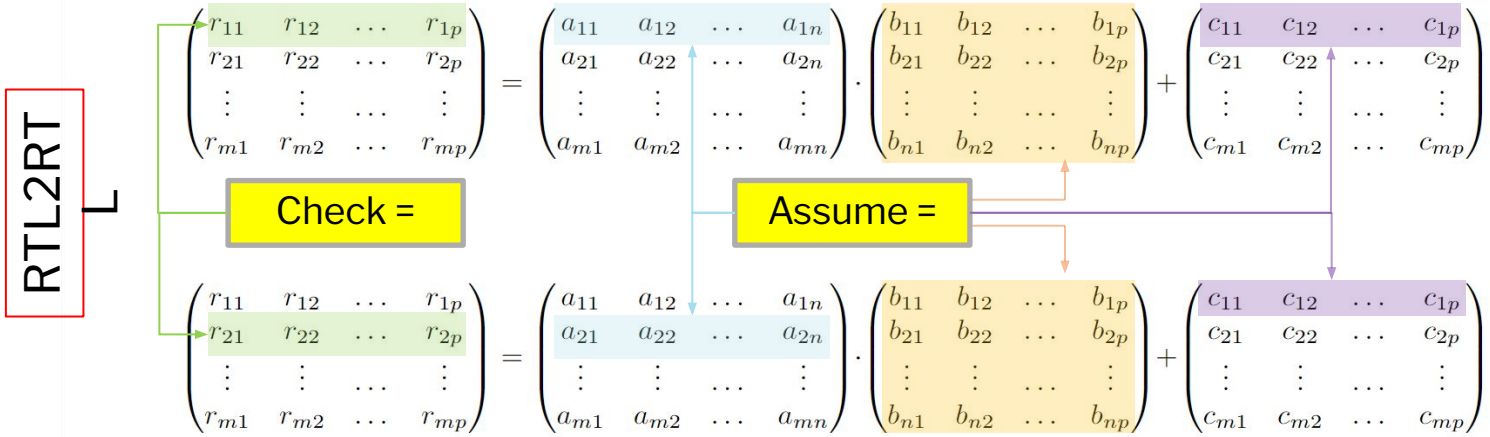
- Column symmetry invariance for the first row:



r_{12} is correct
if
r_{11} is correct

Proves the first row of R

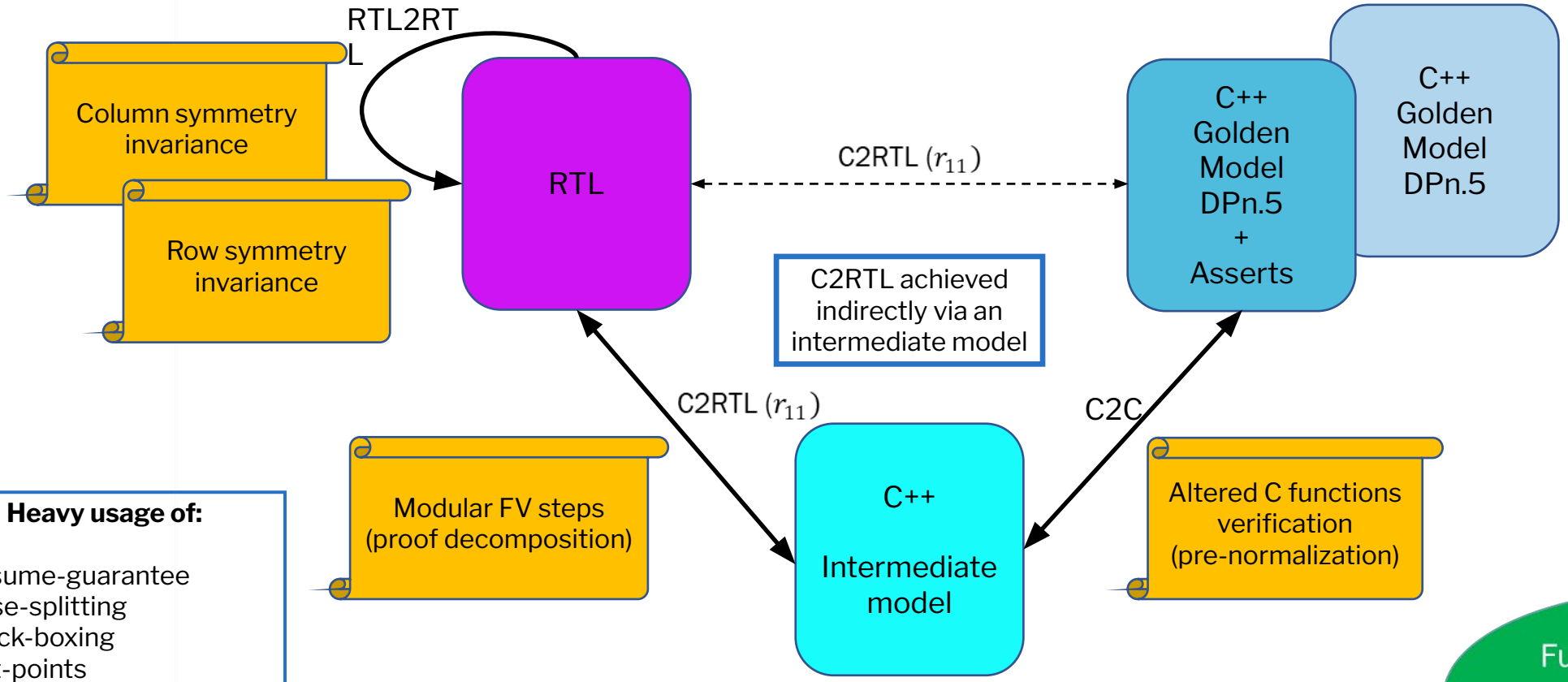
- Row symmetry invariance for the other rows:



Row r_{2j} is correct
if
Row r_{1j} is correct

Proves the other rows of R

Successful FV Landscape



Heavy usage of:

- Assume-guarantee
- Case-splitting
- Black-boxing
- Cut-points
- Innovative tool super usage for multiplications FV

Fully convergent for more than 2^{578} input values per output!

Results

- Complete formal equivalence between RTL and independent C++ model
  - **Proven the absence of bugs** compared to the golden reference

- Bugs found during the process
  - Good initial results with bug hunting approaches, like constraining most of the mantissas of the inputs to 0
  - Some bugs found happen with a probability of less than 2^{-50}, **practically impossible to find via random sim!**

- Overall runtime for our tests: < 24h using multiple licenses

- Creation of parametrizable FV code, reusable in multiple contexts

Conclusions

- We presented a **new symmetry invariance methodology** which achieved
  - full verification of floating-point matrix multiply-add hardware components
  - parallel progress among FV engineers and
  - risk diversification
- This approach
  - scales up to designs with **hundreds of input bits** and
  - can be easily adapted to **verify similar components**
- We **developed a new technique** to use commercial EC tools as Automatic Theorem Provers
- The overall proof requires advanced knowledge of RTL, computer arithmetic and formal tools to **obtain results not achievable out of the box**