An Efficient Methodology to Verify Floating Point Matrix Multiplication

Gerardo Nahum, Nicolae Tusinschi, Salaheddin Hetalani, Siemens EDA

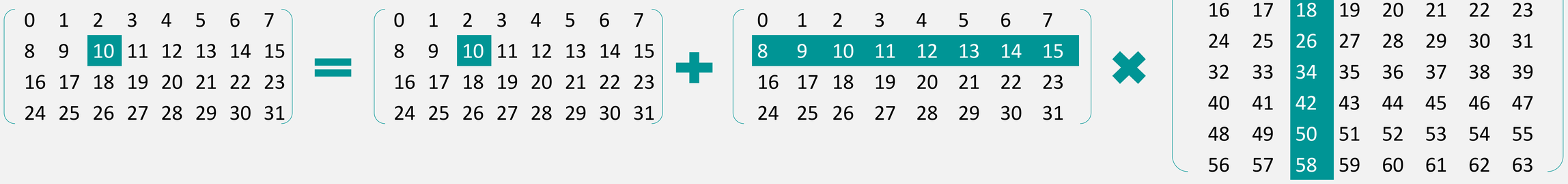
SIEMENS

Motivation

- Verification of matrix multiplication hardware is a massive computation effort
- Exhaustive check for interesting cases and different types of operands and operations is a must
- Siemens EDA has developed a solution to enable formal verification of matrix multiplication results

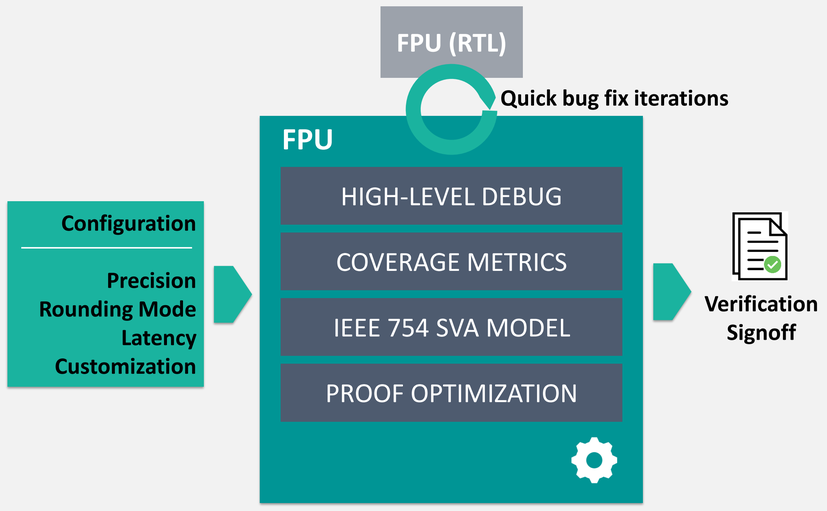
Floating point matrix multiplication

- In order to calculate element 10 of the matrix:
  - R10 = ACC10 + X8*Y2 + X9*Y10 + X10*Y18 + X11*Y26 + X12*Y34 + X13*Y42 + X14*Y50 + X15*Y58



Use model of FPU automated checks

- Supports broad array of FPU functionality
  - Half/single/double bfloat16 and custom precisions
  - All 5 rounding modes and 5 exceptions flags
  - ADD, SUB, MUL, FMA, ABS, NEG functions
  - Conversion and comparison operations
- Parameters specify ambiguities in the standard
- Easy to model intended deviations from the IEEE-754 standard
- Enables RISC-V configuration



Use case

- Each element of the resulting matrix is calculated as follows
  - R[i] = ACC[i] + Row Matrix X * Column Matrix Y
- This is a vector fuse multiply and accumulate (VFMA) operation, which requires to be populated with the relevant row and column elements of the matrixes
- We've built a new VFMA operation as follows
  - VFMA = ACC + Xc0*Yr0 + Xc1*Yr1 + Xc2*Yr2 + Xc3*Yr3 + Xc4*Yr4 + Xc5*Yr5 + Xc6*Yr6 + Xc7*Yr7
  - Becomes a basic building block to check each result
  - Support different floating-point types

The details

User Data

```
property check_op (input integer k) ;
                        ieee_t local_prod ;
                        ieee_t local_acc  ;
                        ieee_t expected   ;
                        ##0 operation = MAC
                        ##1 (1, local_prod = prod [k])
                        ##1 (1, local_acc  = acc[k] )
                        ##1 (1,expected = vfma (.op(local_acc), .prod(local_prod), .rm(roundmode) )
                        ##X operation = NOP
                        |->
                        ieee_check_result (.expected(expected), .actual (design_result_with_flags)  );
endproperty
...
genvar element,i
generate
for (element =0 ; element < 16 ; element++)
                        for(i=0;i<8;i=i+1) begin:
                                                prod[element][2*i]   = MX[element/8*8+i];
                                                prod[element][2*i+1] = MY[element%8+i*8];
                                                acc[element] = design_acc_vector[32*(i+1):32*i];
                        end
                        asrt_element : assert property check_op (element);
  endgenerate
```

Runtime results

Error detected when having a small accumulator exponent and large product exponent but zeros on mantissa

Operation	Unrestricted before fix	Has special numbers (NaN or Inf)	Unrestricted after fix	Restricted single multiplication after fix	Restricted two multiplications after fix
VADD, VSUB, VNEG	No fails	20 sec	20sec		
VMUL (Accumulator is zero)	1 min	4min	Hold bounded	10min for full prove	4h for full prove
VFMA	1 min	4min	Hold bounded	9h for full prove	Hold bounded

Summary

- Siemens EDA's QOS FPU app accelerates IEEE-754 verification and proves correctness with easy setup, without C++ model
- QOS FPU solution has the building blocks to construct simple readable properties
- Provers and disprovers performance enables bug finding in minutes