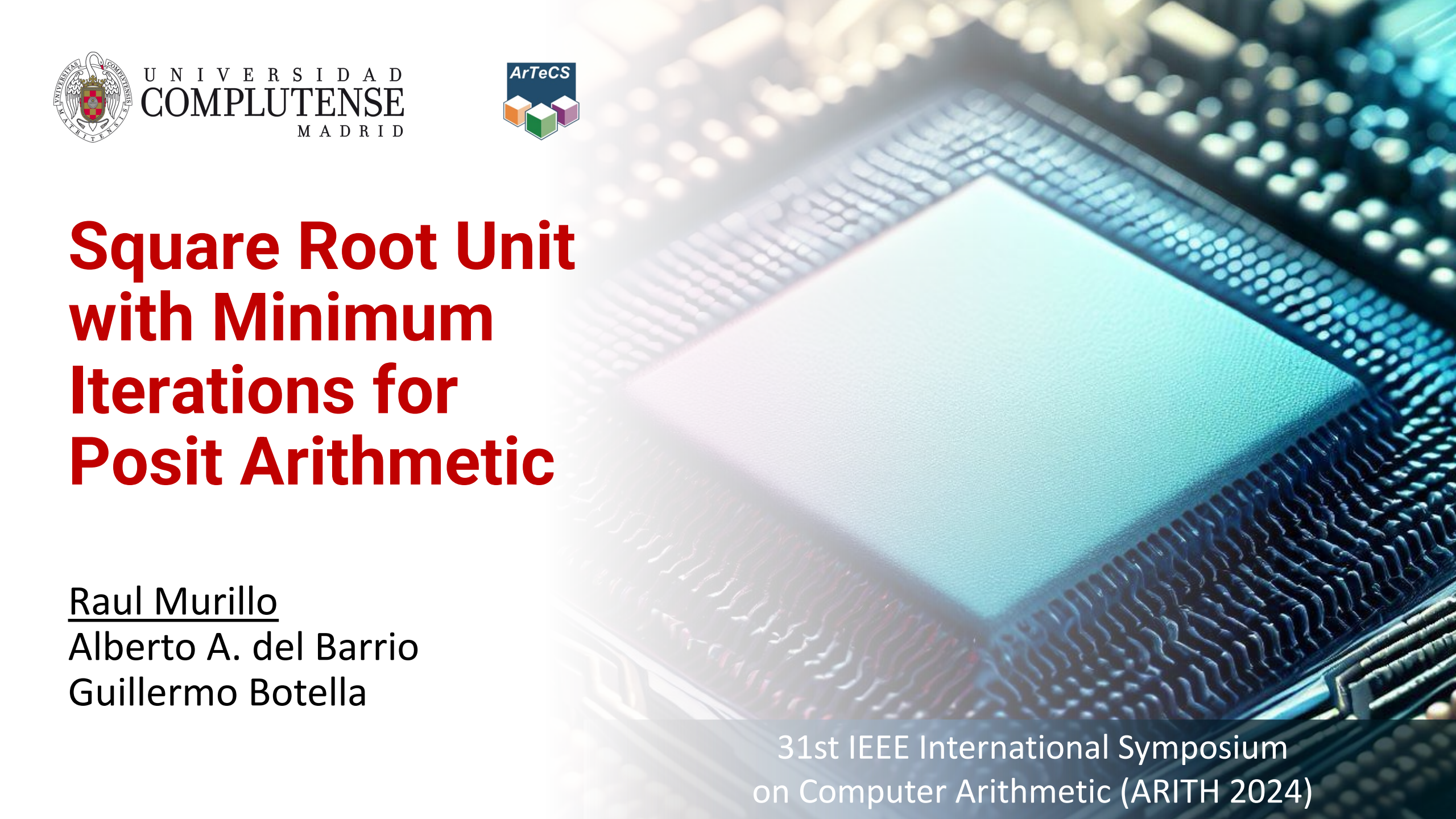


UNIVERSIDAD COMPLUTENSE MADRID

ArTeCS

Square Root Unit with Minimum Iterations for Posit Arithmetic

Raul Murillo
Alberto A. del Barrio
Guillermo Botella

31st IEEE International Symposium
on Computer Arithmetic (ARITH 2024)

Outline

- **Introduction**
- Posit square root unit design
- Hardware evaluation
- Conclusions

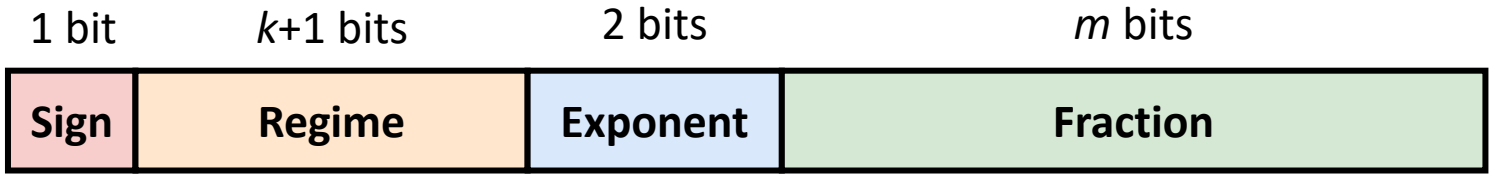
Float & posit formats

- Floating-point:

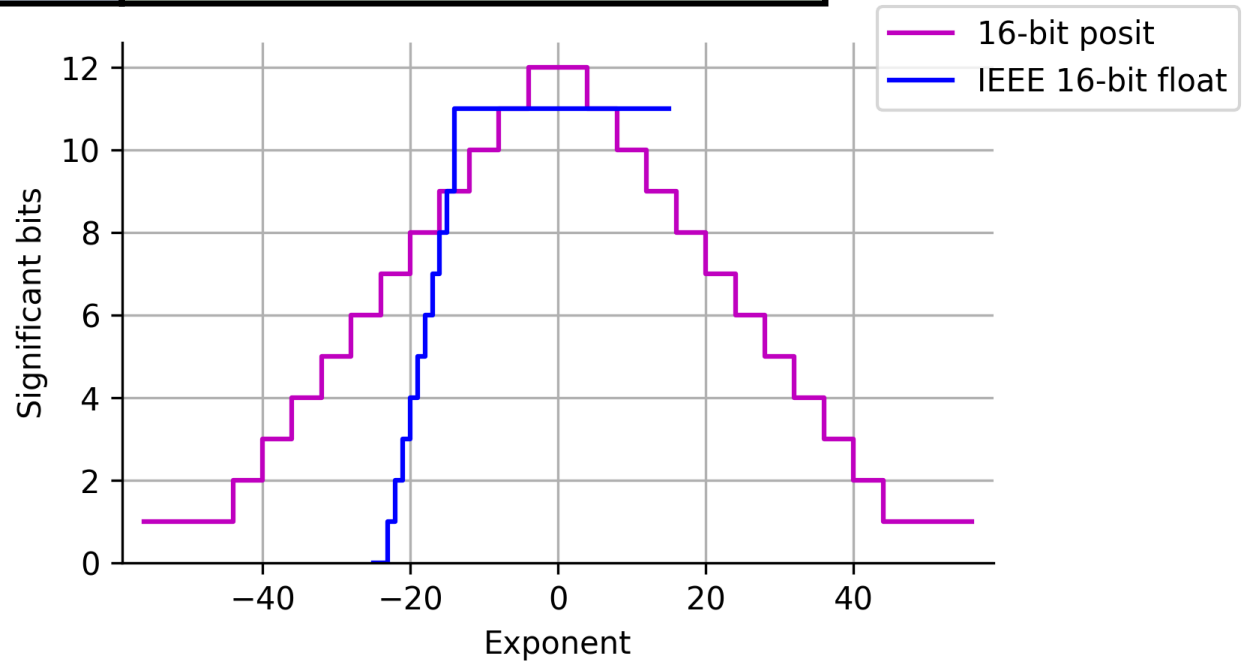
$$X = (-1)^{s} \times 2^{e} \times (1 + f)$$

The effect of the new regime field

- Variable-length field: sequence of *k* identical bits (ending with a negated bit)



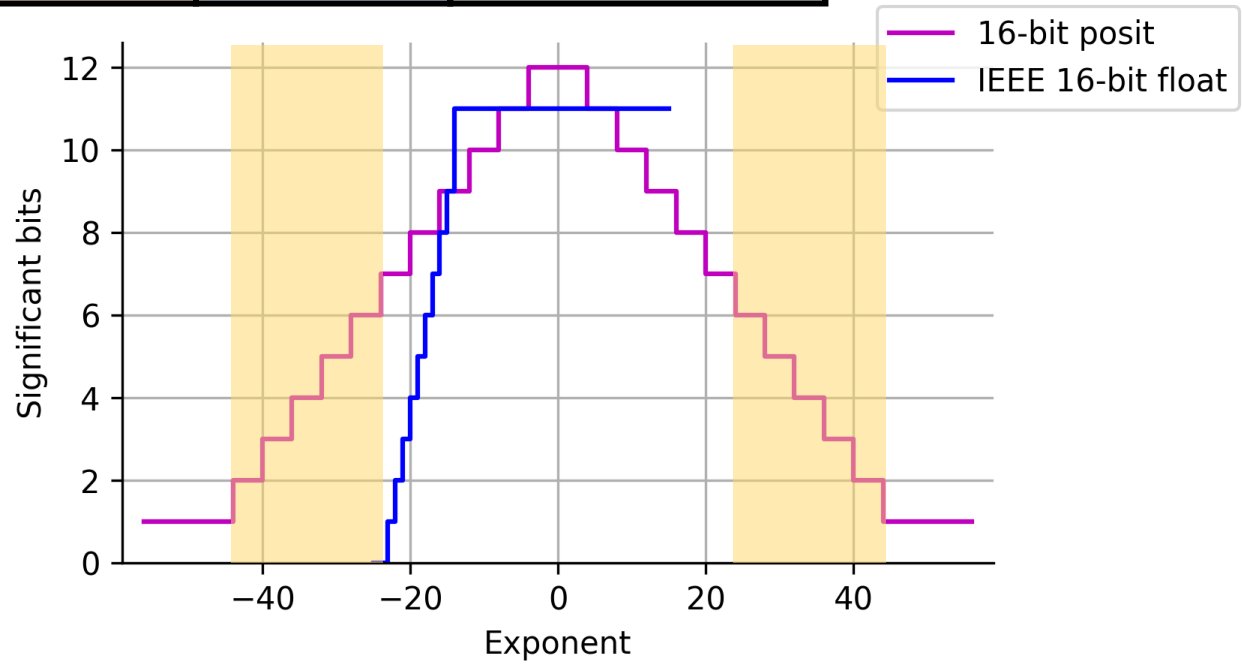
- Coarse-grained scaling factor
- Trade-off:
  precision – dynamic range



The effect of the new regime field

- Variable-length field: sequence of *k* identical bits (ending with a negated bit)

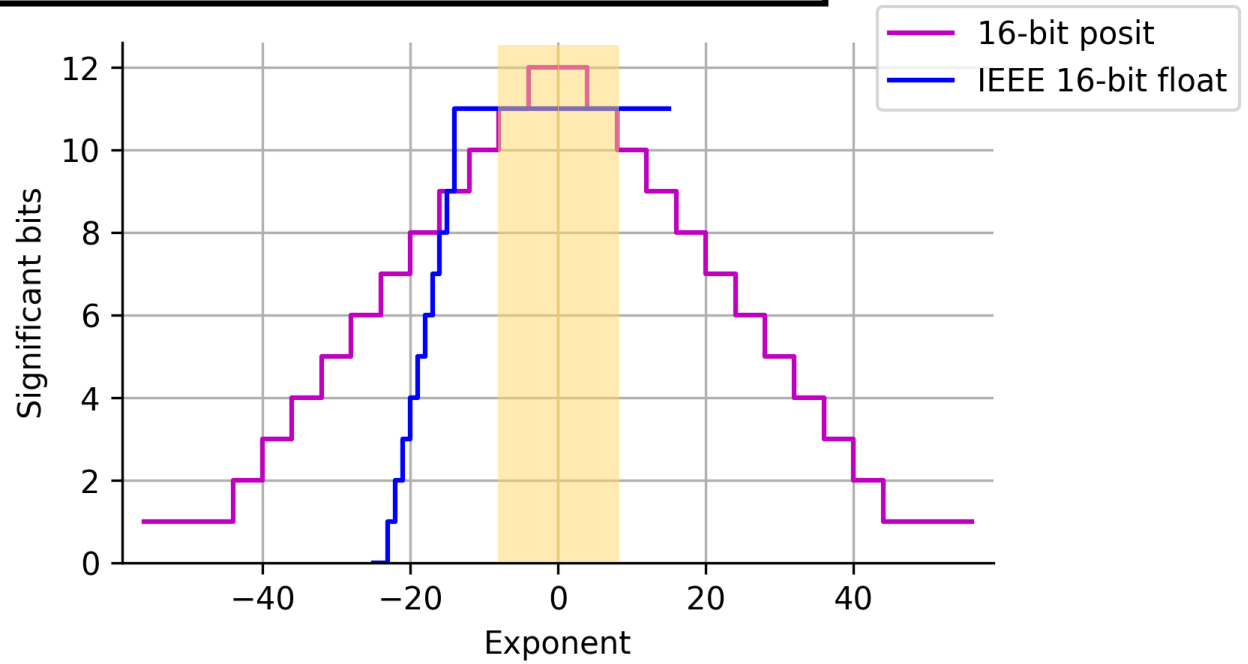
- Coarse-grained scaling factor
- Trade-off:
  precision – dynamic range



The effect of the new regime field

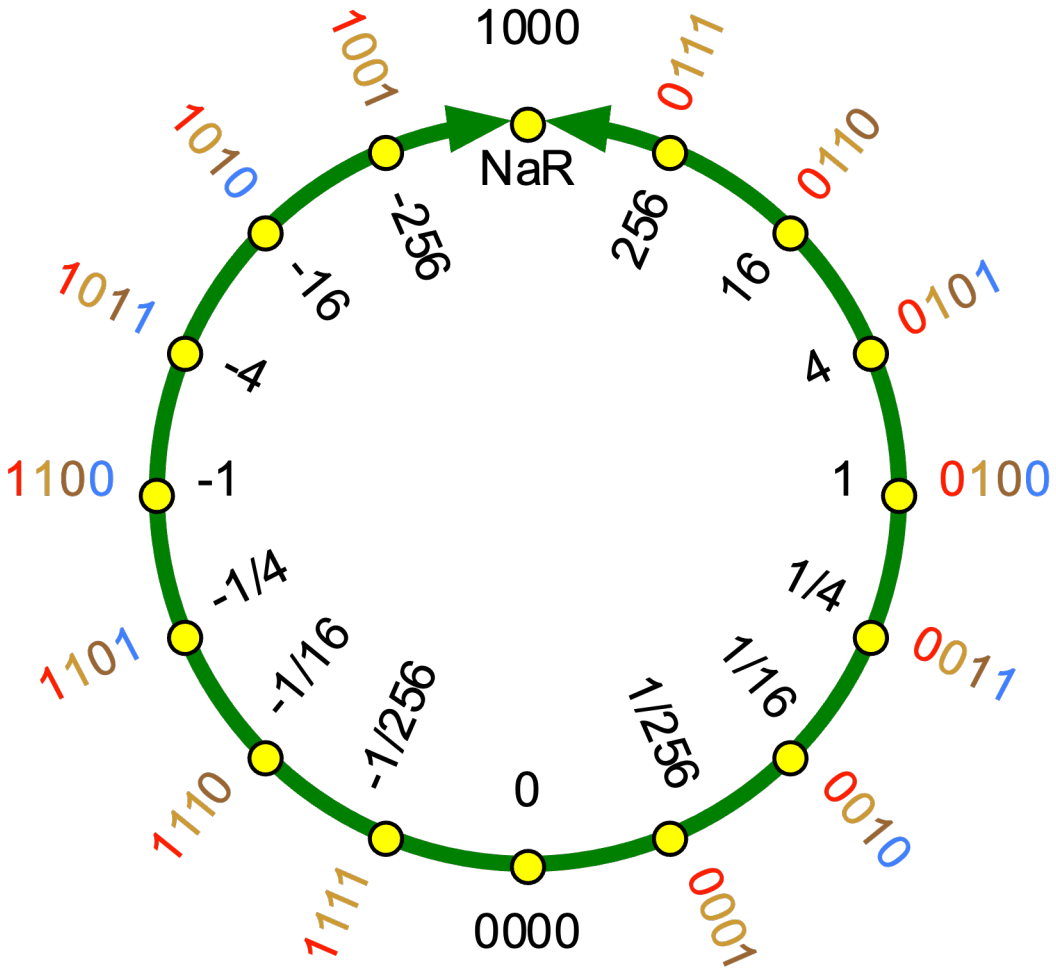
- Variable-length field: sequence of *k* identical bits (ending with a negated bit)

- Coarse-grained scaling factor
- Trade-off:
  precision – dynamic range
- Not compatible with FP units



Some properties of posits

- Higher accuracy around 2^0
  - For 32-bits, around $[2^{-16}, 2^{16}] \approx [10^{-5}, 10^4]$
- Single zero and single exception (NaR)
  - Floats have -0 and +0
- Two's complement design
  - Comparison as for integer arithmetic
- Fused arithmetic
  - Accumulation of multiple products with no intermediate rounding



Outline

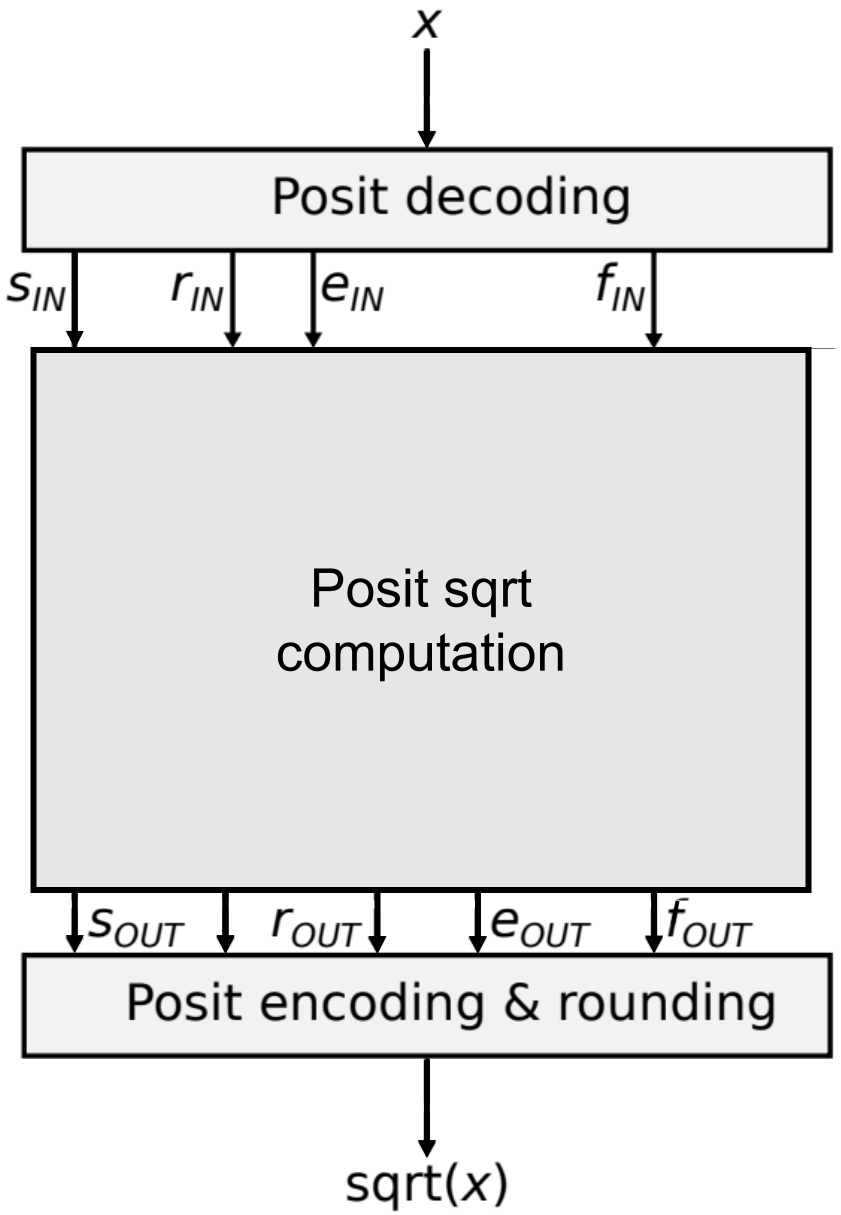
- Introduction
- **Posit square root unit design**
- Hardware evaluation
- Conclusions

Posit square root

$$X = (-1)^{s} \times 2^{4r+e} \times (1 + f)$$

$$\sqrt{X} = 2^{\frac{4r+e}{2}} \times \sqrt{1 + f}$$

- Odd exponent
  - Subtract 1, multiply significand by 2
- Sqrt algorithm requires the operand to be in [0.25, 1)
  - Divide significand by 4



The square root algorithm

- Digit-recurrence methods vs. iterative methods
  - Provide a fixed number of correct digits per iteration (from MSB to LSB)
  - Low complexity
  - Small area and power
  - High latency

- Non-restoring algorithm
  - Employ radix-2
  - Uses the quotient-digit set {-1, 0, 1}

The Non-restoring algorithm

1: **procedure** NON-RESTORING(X)
2: $R_0 \leftarrow X$
3: $Q_0 \leftarrow 0$ ▷ $Q = (q_0.q_1 q_2 \dots q_m)$
4: **for** $i = 0, \dots, m-1$ **do**
5: **if** $R_i \geq 0$ **then**
6: $Q_{i+1}[i+1] \leftarrow 1$
7: $R_{i+1} \leftarrow 2R_i - (2Q_i + 2^{-i})$
8: **else**
9: $Q_{i+1}[i+1] \leftarrow -1$
10: $R_{i+1} \leftarrow 2R_i + (2Q_i - 2^{-i})$
11: **return** Q_m

Initialization

Recurrence step (all asignments in parallel)

On-the-fly conversion

Termination

Optimizations

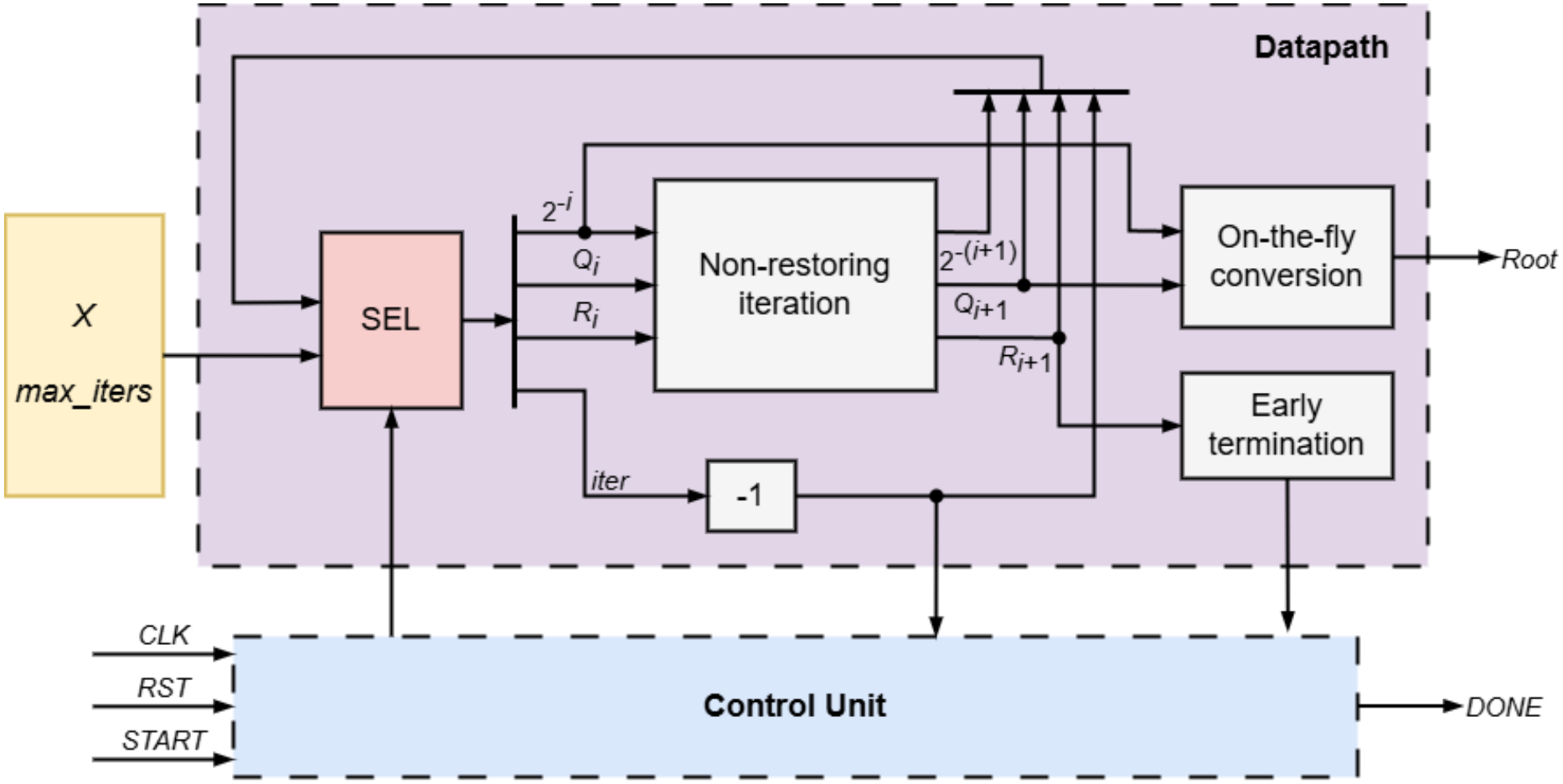
1. **Early termination**
   If $R_{i+1} = 0$ ➔ Stop

p_1 := 0 110 10 11 = 112
$\sqrt{p_1}$:= 0 10 11 011 = 11

p_2 := 0 0001 01 1 = 7.324×10^{-4}
$\sqrt{p_2}$:= 0 001 10 11 = 2.734×10^{-2}

2. **Variable number of iterations**
   - 1 correct digit per iteration
   - Floating-point has fixed fraction length
   - Posit has variable fraction length!
   - Compute only as many iterations as needed

Sqrt unit implementation



How many iterations can we reduce?

$$X = (-1)^{s} \times 2^{r \times 4 + e} \times (1 + f) \quad \Rightarrow \quad \sqrt{X} = 2^{(r \times 4 + e)/2} \times \sqrt{1 + f}$$

- It depends on the regime: r given by the **length** of the regime field (r_s)

	Fraction bits	Radix-2 iterations	Sqrt unit latency
Posit16	4 – 11	6 – 13	8 – 15
Posit32	12 – 27	14 – 29	16 – 31
Posit64	28 – 59	30 – 61	32 – 63
Float16	10	12	14
Float32	23	25	27
Float64	52	54	56

+2 +2

Output fraction bits:

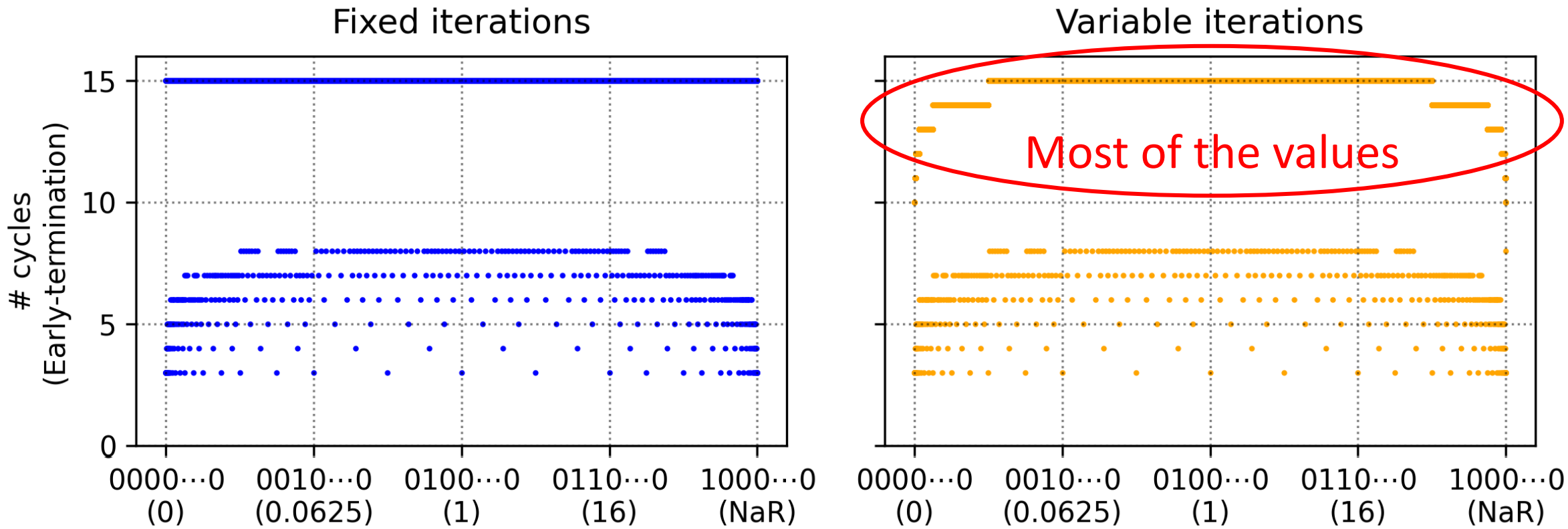
$$\widehat{f_s} = n - 2 - \left\lceil \frac{r_s}{2} \right\rceil - e_s$$

Outline

- Introduction
- Posit square root unit design
- **Hardware evaluation**
- Conclusions

Latency reduction

- HDL simulation of exhaustive testbench (Posit16)



Latency reduction in real applications

- PolyBench/C benchmark with Posit32
- Square root operations computed via HDL simulation (ModelSim)

Max sqrt latency: 31 cycles

Early termination

Larger dynamic range

Number of cycles

	correlation				gramschmidt			
Design	MINI	SMALL	MEDIUM	LARGE	MINI	SMALL	MEDIUM	LARGE
# operations	924	8080	62640	1681200	30	80	240	1200
Fixed iterations	28644	42480	1941840	52117200	930	2480	7440	37183
Variable iterations	28637	42400	1879200	50434800	847	2292	6803	34291
Reduction	-7 (0.02%)	-80 (0.02%)	-62640 (3.23%)	-1682400 (3.23%)	-83 (8.92%)	-188 (7.58%)	-637 (8.56%)	-2892 (7.77%)

Synthesis results

- ASIC synthesis
- Synopsys DC
  - 28 nm TSMC
  - 1 GHz

Small overheads

Bits	Design	Area (μm^2)		Power (mW)		Slack (ns)
16	Fixed iterations	625.21	2%	0.0810	3%	0.46
	Variable iterations	638.19		0.0836		0.46
32	Fixed iterations	1296.79	1%	0.1501	2%	0.06
	Variable iterations	1314.43		0.1534		0.06
64	Fixed iterations	3256.47	0.8%	0.3047	1%	0.19
	Variable iterations	3282.80		0.3089		0.19

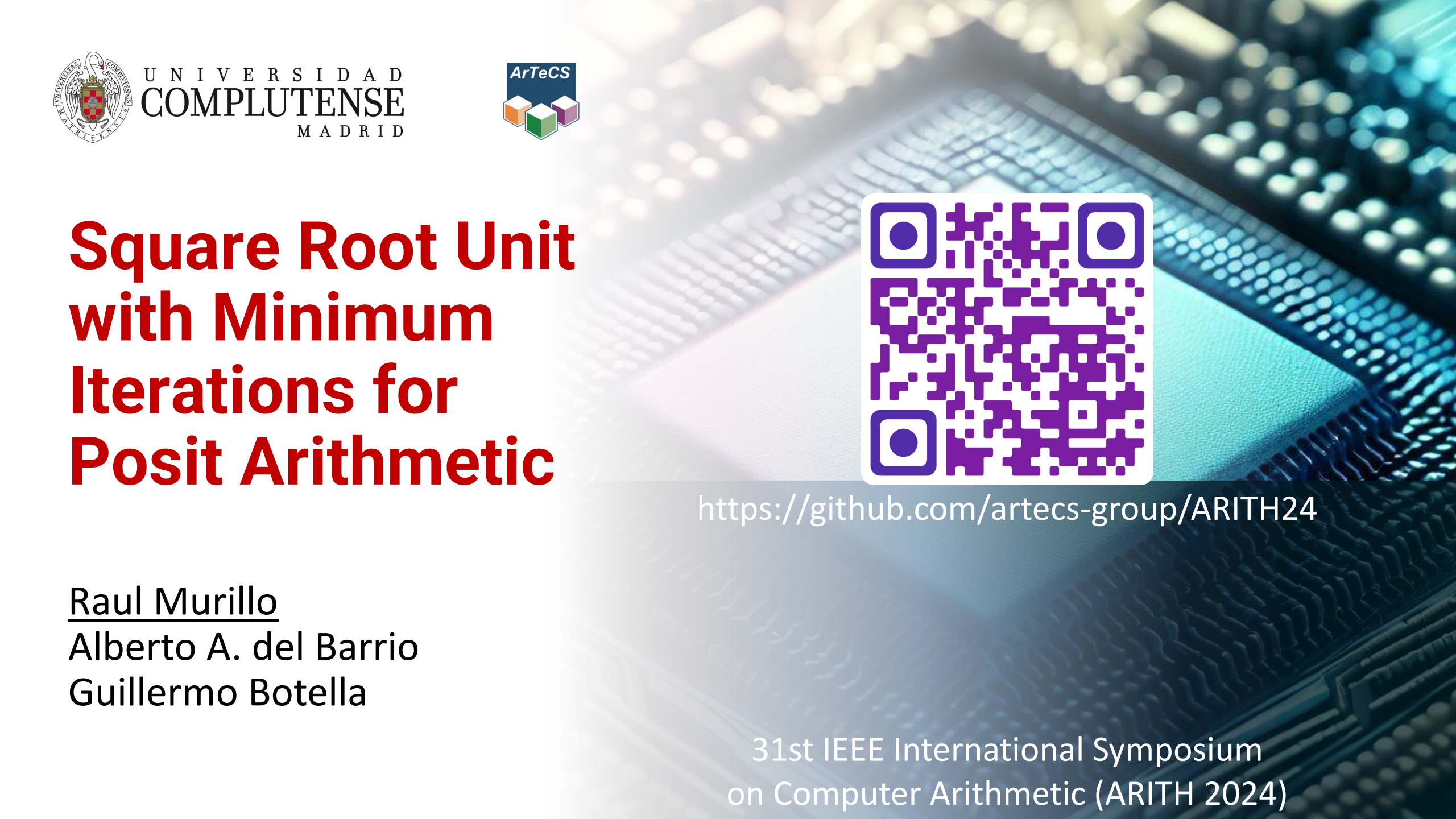
Outline

- Introduction
- Posit square root unit design
- Hardware evaluation
- **Conclusions**

Conclusions

- Posit arithmetic as alternative to IEEE 754 FP format
  - Variable length fields
  - Tradeoff: dynamic range – precision
  - Presented in 2017: lack of computer arithmetic units
- Proposed a sqrt unit for posit arithmetic
  - Radix-2 non-restoring algorithm
  - Optimized for posit arithmetic → minimize iterations
- Results
  - Latency reduction up to 8.92% (depends on the application)
  - Marginal area-power overhead (< 3%)
- Implementation is freely available: github.com/artecs-group/ARITH24

GitHub



UNIVERSIDAD COMPLUTENSE MADRID

ArTeCS

Square Root Unit with Minimum Iterations for Posit Arithmetic

https://github.com/artecs-group/ARITH24

Raul Murillo
Alberto A. del Barrio
Guillermo Botella

31st IEEE International Symposium
on Computer Arithmetic (ARITH 2024)