Universal V3
reliable, reproducible, and energy-efficient numerics

Theodore Omtzigt
Stillwater Supercomputing, Inc.

James Quinlan
School of Mathematical and Physical sciences
University of New England

Digital Transformation

- Source: 2015 ITRS 2.0 Executive Report

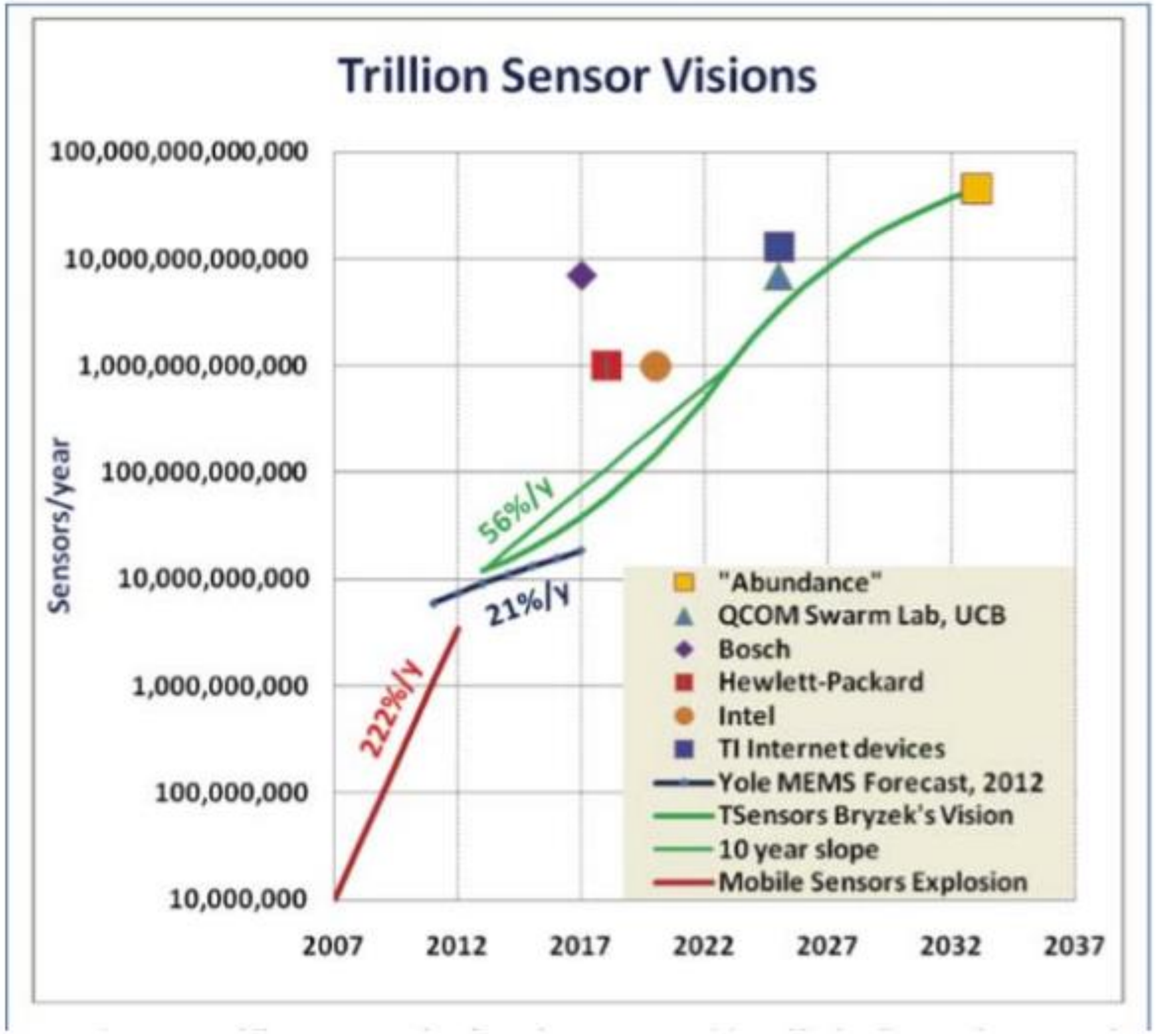
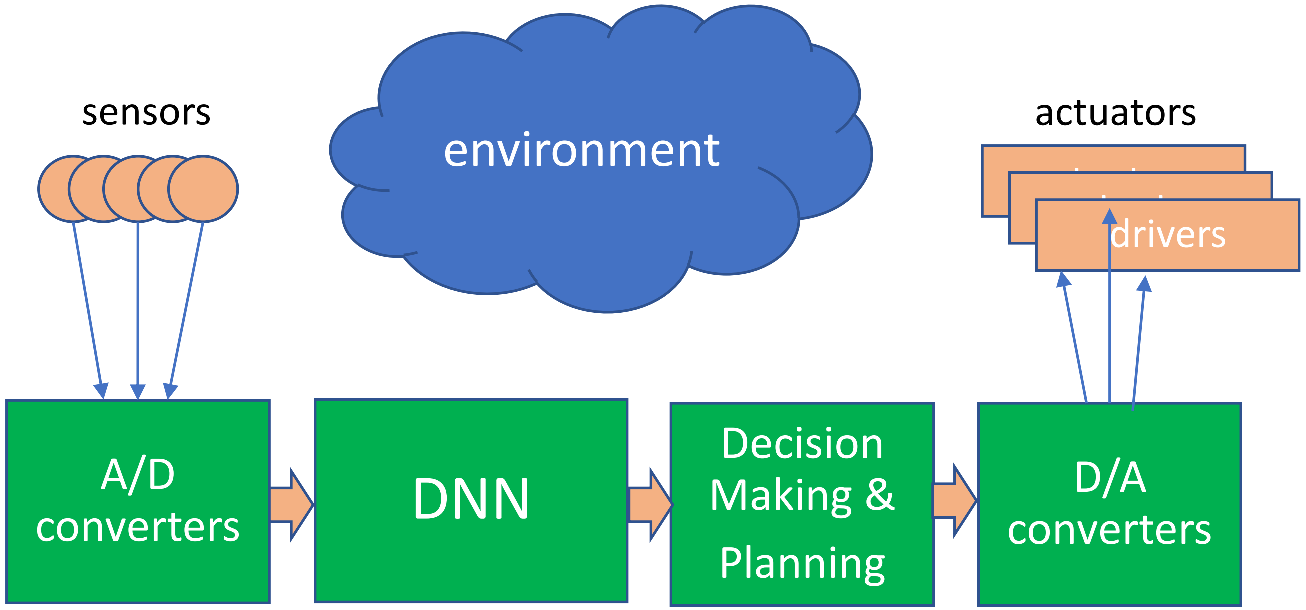


Fig. 5.2 Sensors will populate the world of the IoE

Embedded Intelligence

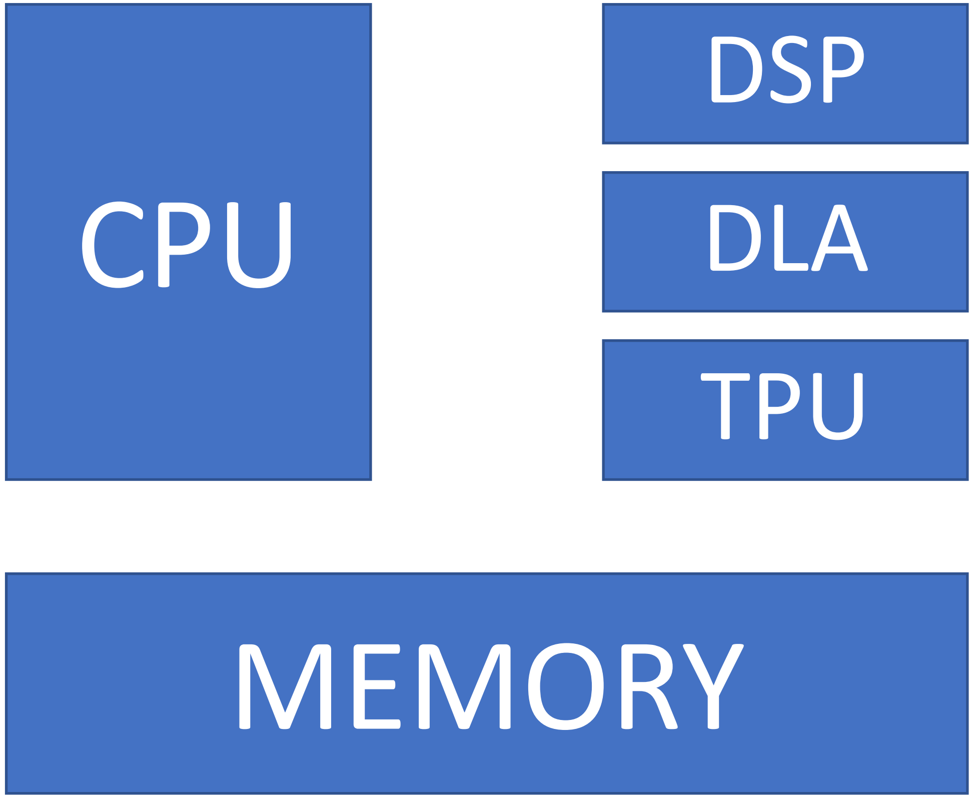
- Deep Learning has taken over many signal and image processing task
- Deep Learning (DL) and Decision Making & Planning (DMP) is compute-intensive
- Energy-efficiency of DL & DMP computation is essential
- Specialized accelerators offer high-performance and energy-efficiency.



Requirements tailored to environment, sensors, and actuator dynamics.

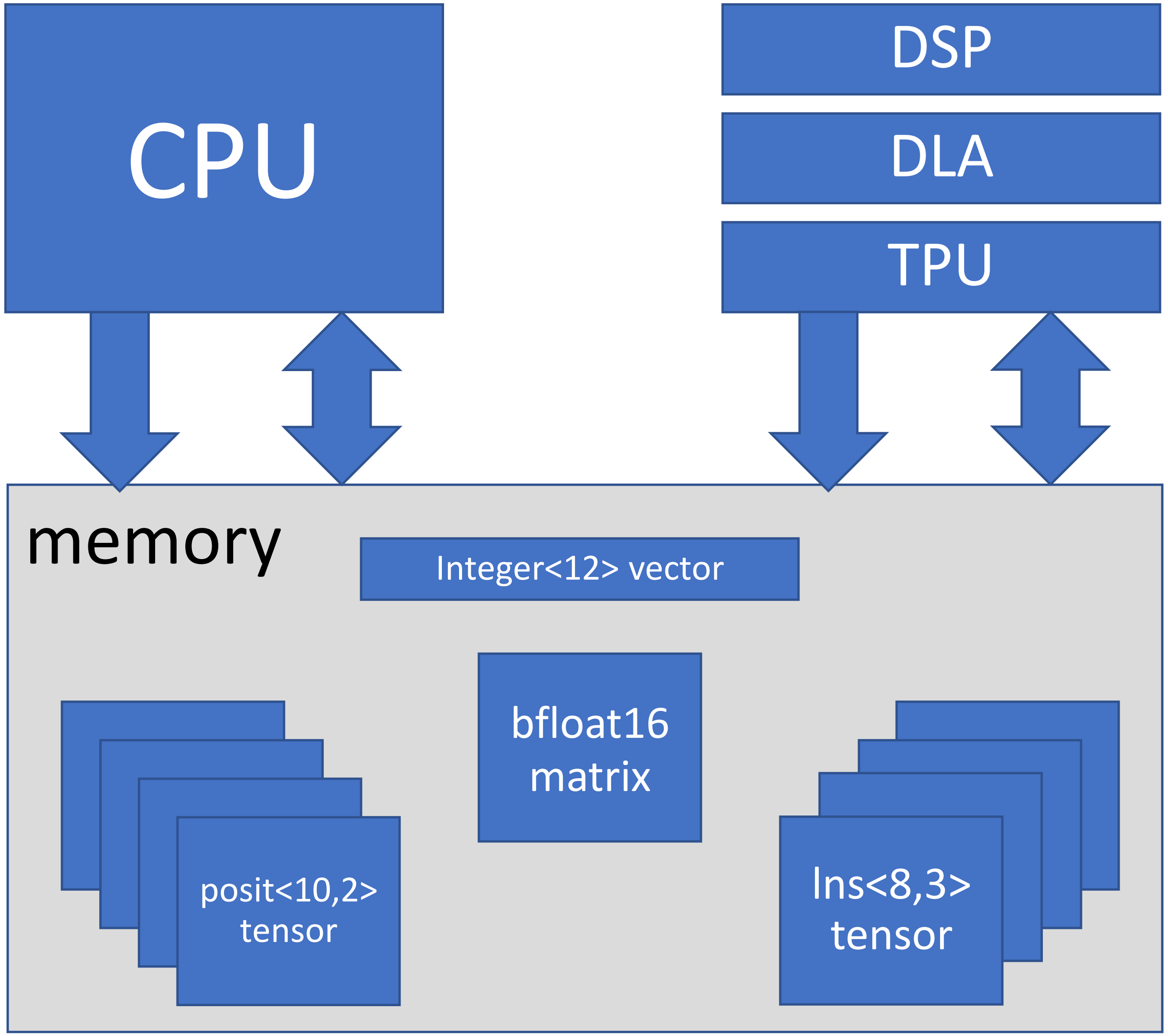
System Architecture: CPU <-> Accelerator

- Tight collaboration between CPU and Accelerator
- Handshake through memory
  - Streams
  - Vectors
  - Matrices
  - Tensors
- CPU and Accelerator need to collaborate using custom numerics
  - ***Universal V3 provides all services required***



Universal Design Pattern:

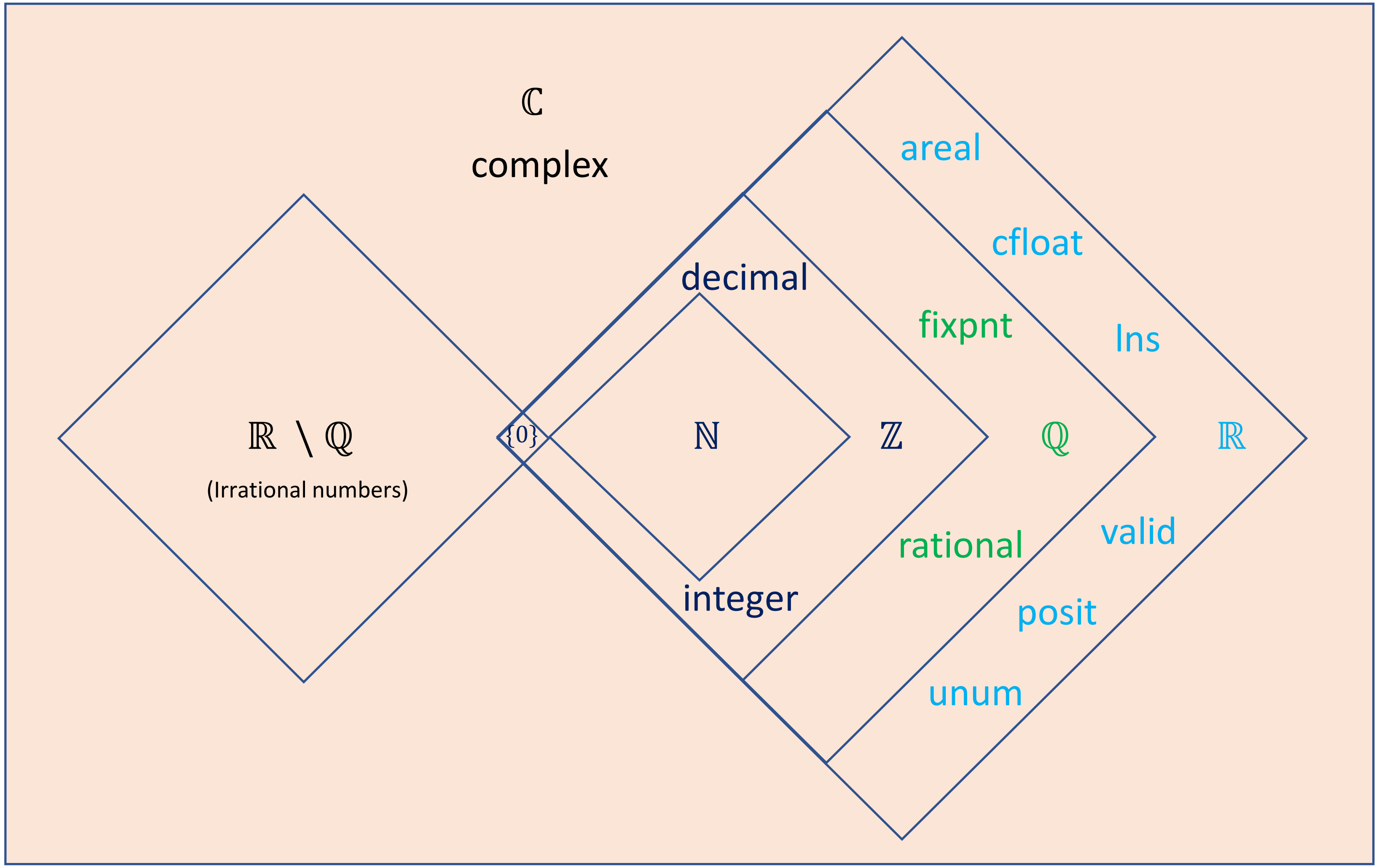
Collaborate Through Memory



Universal V3

A versatile **Arithmethic Type** Library

Universal arithmetic types



Infinite Precision

- decimal
- rational
- adaptiveint
- adaptivefloat
- adaptiveposit

$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$

$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$

- Parameterize algorithms along data type
- Explore refinement
  - Precision
  - Dynamic range
  - Arithmetic
  - Sampling Profile (linear, exponential, logarithmic, custom, etc.)
- Leverage *Fused Dot Product*
  - Theoretical max benefit is a 2x reduction in operand bandwidth
- Measure numerical error
  - Forward error propagation
  - Theoretical predictions tend to be too conservative

Directory of Arithmetic Types

Replace

- integer
- fixpnt
- cfloat
- posit
- lns
- unum

Reliable

- areal
- interval
- valid

Precision

- decimal
- rational
- adaptint

Oracle

- priest
- lazyexact

Number System:

integer

```
template<size_t _nbits,
    typename BlockType = uint8_t>
class integer;
```

- Arbitrary Precision, fixed-size integer of *nbits*
- 2's complement signed integer
- User-directed block size
  - Minimum set of blocks to contain the integer
  - Optimal memory layout for linear-algebra
  - Default block size is 8bits
- Smallest integer is integer<2>
- Largest integer is limited by memory
  - We have tested nbits = 1,048,576

Number System:

fixpnt

```
template<size_t nbits, size_t rbits,
    bool arithmetic = Modulo,
    typename bt = uint8_t>
class fixpnt;
```

- Arbitrary Precision, fixed-point number of *nbits* with radix point at *rbits*
- 2's complement signed integer with *rbits* normalizing shift
- Either Saturating or Modulo arithmetic
- User-directed block size
  - Minimum set of blocks to contain the fixpnt
  - Optimal memory layout for linear-algebra
  - Support for *Fused Dot Product*
  - Default block size is 8bits
- Smallest fixpnt is fixpnt<2,2>
- Largest fixpnt is limited by memory

Number System:

cfloat

```
template<size_t nbits, size_t es,
 typename bt = uint8_t, bool hasSubnormals,
 bool hasSupernormals, bool isSaturating>
     class cfloat;
```

- Arbitrary Precision, fixed-size classic float of *nbits* and an exponent field of size *es* bits
- Selectable gradual underflow (subnormals), gradual overflow (supernormals), and saturation
- 1 bit Signalling/Quiet NaN, 1bit +/-infinite
- + and - zero
- User-directed block size
  - Minimum set of blocks to contain the cfloat
  - Optimal memory layout for linear-algebra
  - Support for *Fused Dot Product*
  - Default block size is 8bits
- Smallest cfloat is cfloat<3,1,uint8_t,true,true>
- Largest *es* is currently 11

Number System:

posit

```
template<size_t _nbits, size_t _es,
         typename bt = uint8_t>
class posit;
```

- Arbitrary Precision, fixed-size tapered float of *nbits* and *es* exponent bits
- Exponential regimes of *useed* = $2^{2^{es}}$
- Saturating arithmetic to *minpos*/*maxpos*
- 1 code for NaR, 1 code for 0
- Maximum precision at 1.0 using (*nbits – es – 1)* fraction bits
- User-directed block size
    - Minimum set of blocks to contain the posit
    - Optimal memory layout for linear-algebra
    - Default block size is 8bits
- Smallest posit is posit<2,0>
- Largest posit is posit<4096,9>

Number System:

lns

```
template<size_t nbits, float base,
    typename bt = uint8_t>
  class lns;
```

- Arbitrary Precision, fixed-size logarithmic number system of *nbits*
- Values represent $\log_b(|x|)$
- Represented as 2's complement number
- User-directed block size
  - Minimum set of blocks to contain the lns
  - Optimal memory layout for linear-algebra
  - Default block size is 8bits

Number System:

decimal

```
class decimal;
```

- Exploratory/Test number system
- Mathematical experiments on representation and precision
- Arbitrary precision and dynamic range
- Using decimal arithmetic

Number System:

rational

```
class rational;
```

- Exploratory/Test rational number system
- Mathematical experiments on representation and precision
- Adaptive precision and dynamic range
- Using decimal arithmetic

Number System:

areal

```
template<size_t nbits, size_t es,
    typename bt = uint8_t>
class areal;
```

- Arbitrary Precision, fixed-size interval float of *nbits* and an exponent field of size *es* bits
- Gradual underflow (subnormals) and gradual overflow (supernormals)
- Uncertainty bit to model (v, v+ULP) interval
- User-directed block size
  - Minimum set of blocks to contain the areal
  - Optimal memory layout for linear-algebra
  - Default block size is 8bits
- Smallest areal is areal<4,1>

Number System:

valid

```
template<size_t nbits, size_t es,
         typename bt = uint8_t>
class valid;
```

- Arbitrary Precision, fixed-sized valid of *nbits* and *es* exponent bits
- An interval type with posits as lower/upper bound values plus an uncertainty bit
- User-directed block size
  - Minimum set of blocks to contain the valid
  - Optimal memory layout for linear-algebra
  - Default block size is 8bits

Number System: unum

```
template<size_t esizesize, size_t fsizesize,
         typename bt = uint8_t>
class unum;
```

- Variable Precision floating-point of maximum *esizesize* exponent bits and maximum fsizesize fraction bits
- Experimentation type to capture precision of complex computations
- User-directed block size
  - Minimum set of blocks to contain the unum
  - Optimal memory layout for linear-algebra
  - Default block size is 8bits

Application Examples

Application Integrations:

- G+SMO: Iso Geometric Analysis Package
- FDBB: Fluid Dynamics Building Blocks
- MTL4 and MTL5 linear algebra engines
- AutoDiff: Automatic Differentiation engine
- ODEint: Boost Library for ODE solvers
- LibKet: Quantum Computer simulator

Fused DOT Product

- Chebyshev Polynomials for Approximation
- Reversible FFTs
- Perfect Matrix Inverses
- Krylov IDR(s)
- AI/DL training with 8-bit posits

Integers

- Factorization
- Irrational number approximation
- Quantum Safe Crypto
- Fully Homomorphic Encryption (FHE)

Fixpnt, posit, lns

- Quantum Expression Template Library
- DSP and Spectral Analysis
- Visuomotor applications
- Deep Learning
- Reinforcement Learning

Reals

- FPGA hardware efficiency research
- Computational Science
- Optimization
- Software-defined supercomputing

cfloat/areal & posit/valid

- Numerical error analysis research
- High Performance Computing
- Software defined supercomputing

Example: multi-precision Linear Algebra: Matrix Squeezing

- Solve Ax = b using low precision with iterative refinement in native precision (references)

Algorithm 1: Round and replace overflow with $x_{\max}$.

Input: An $n \times n$ Matrix A
Output: Rounded matrix B

$B = \mathrm{fl_p}(\mathrm{A})$
Set $a_{ij}^{p} = \mathrm{sign(a_{ij})x_{max}}$

Con: Introduces infinities

Algorithm 2: Scaled matrix entries

Input: An $n \times n$ Matrix A
Output: Rounded matrix $A^{(p)}$

$a_{\max} = \max_{i,j} |a_{i,j}|$
$\mu = x_{\max}/a_{\max}$
$A^{(p)} = \mathrm{fl_p}(\mu\mathrm{A})$

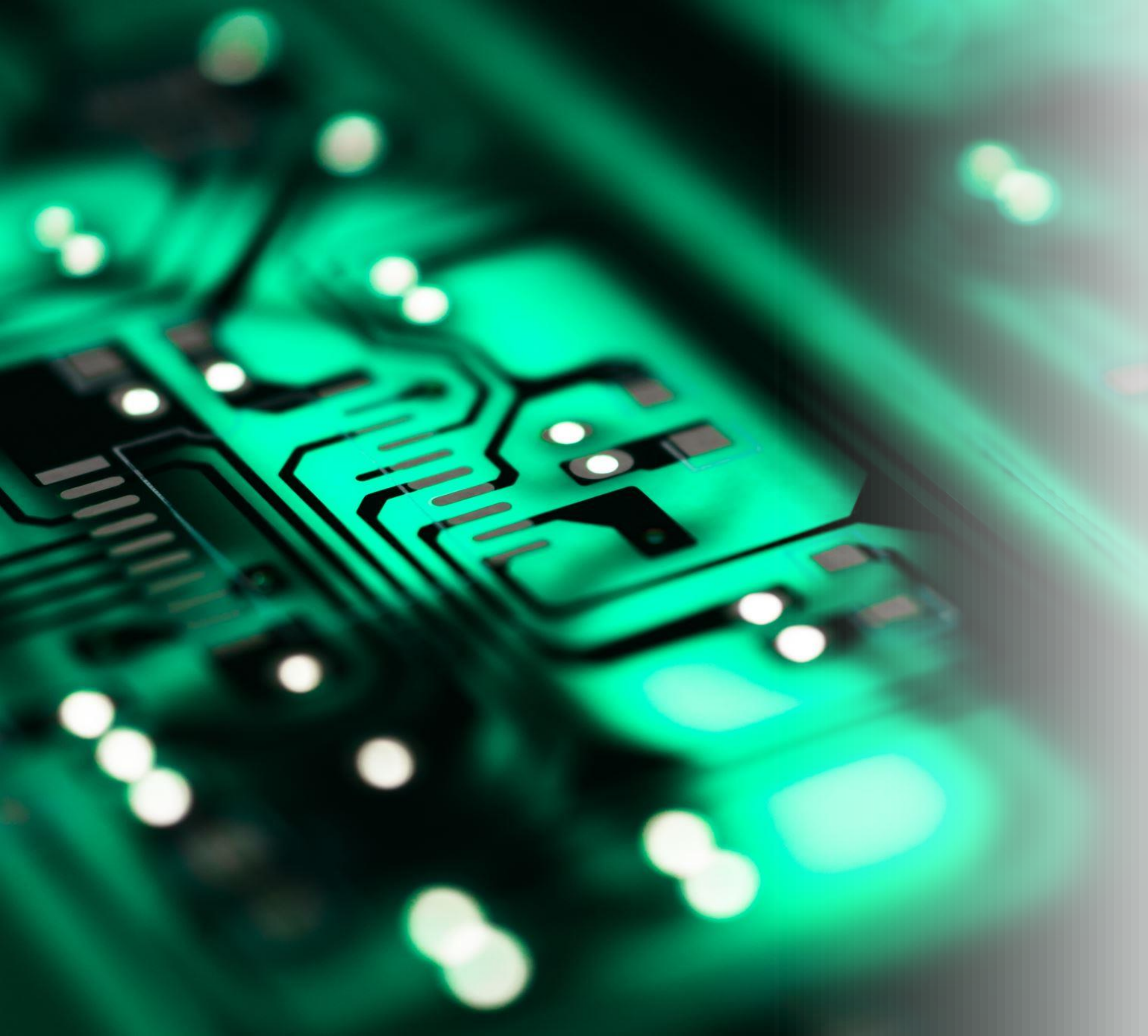
Con: Introduces underflow

Proper squeezing: scale down for stable LU followed by iterative refinement

Algorithm 3: Double side scaling using row and column equilibration

Input: An $n \times n$ Matrix A
Output: Rounded matrix $A^{(p)}$

Set $R = 0$
for $i = 1$ **to** n **do**
  $R(i,i) \leftarrow ||A(i,:)||_\infty^{-1}$
end
$B = RA$
Set $S = 0$
for $j = 1$ **to** n **do**
  $S(j,j) \leftarrow ||A(:,j)||_\infty^{-1}$
end
Set $\beta =$ maximum absolute entry in RAS
Set $\mu = x_{\max}/\beta$
Return $\mathrm{fl_p}(\mu(\mathrm{RAS}))$



Conclusions

- Energy efficiency is driving embedded intelligence applications
- Tailor arithmetic to application to maximize efficiency
- CPU and Accelerator collaborate through memory
- *Universal* is an SDK to develop and optimize mixed-precision algorithms for such CPU-Accelerator systems

Universal Resources

- Github repo: https://github.com/stillwater-sc/universal
- Gitbook: https://stillwater-supercomputing-inc.gitbook.io/universal-numbers/
- Jupiter Notebooks: https://github.com/stillwater-sc/universal-notebook
- G+SMO Iso Geometric Analysis: https://github.com/gismo