FP-ANR:
Another representation format to handle cancellation at run-time

David Defour
LAMPS, Univ. de Perpignan (France)
Motivations

• Number’s representation really matters

• List of problems
  • **Uncertainty**
    Imprecision from physical measure
    \[x = 123.45678, \ U = 10^{-5}, \ x \in [123.455;123.458]\]
  
  • **Cancellation**
    \[x=1.0, \ y=10^{100}, \ z=(x+y)-y = \{1.0 ; 0.0\}\]
  
  • **Rounding**
    Discretization of real number
Outline

• Background (Significance arithmetic, unum, ...)
• Representation format
• Operations
• Results
• Conclusion
Background

• How to handle uncertainty

• Tools
  • CADNA, VERIFICARLO, PRECIMONIOUS, ...

• Formats
  • Unum v1, v2, v3, v4 ....
  • Significance arithmetic
Three ways to express a big number

Avogadro’s number: \( \sim 6.022 \times 10^{23} \) atoms or molecules

Sign-Magnitude Integer (80 bits):

IEEE Standard Float (64 bits):

Unum (29 bits):

Self-descriptive "utag" bits track and manage uncertainty, exponent size, and fraction size.
Background: Significance arithmetic

• How to represent numbers?
  Old problem (1\textsuperscript{st} computers)
  • Fixed-point
    “exact” computation
  • Floating-point
    deal with large/small numbers, but there are problems
  • Significance arithmetic
    keep track of the number of significant digits
    • Denormalized number (add ‘0’ in the leading digits of the mantissa, ex: \texttt{0.000000123})
    • Index of significance (Argonne FLIP based on a triplet \((x_f,x_p,x_i)\), ex: \((0.123,-6,3)\))
      • \(x_f\): fractional part; \(x_p\): associated power; \(x_i\): significant figures
    • Rely on an unused pattern in BCD format (A:1010 / B:1011) (ex: 1.23A000)
FP-ANR: concept

• Implementation of significant arithmetic

• Don’t:
  • Throw away IEEE-754 arithmetic
    Why: good properties, backward compatibility,…
  • Use extra fields
    Why: memory and data transfer are expensive
  • Use varying length
    Why: Make addressing, memory allocation, and hardware implementation difficult

• Do
  • Be IEEE-754 backward compatible
FP-ANR: Just a bit in the mantissa

• 2 patterns
  • 0 1 1 1 ... 1 1 1
  • 1 0 0 0 ... 0 0 0

• Example (Binary32)
  • Value 1: (IEEE-754)
    0 01111111 00000000000000000000000

  • Value 1: (FP-ANR, accurate to 23 bits)
    0 01111111 000000000000000000000001

  • Value 1: (FP-ANR, accurate to 13 bits)
    0 01111111 0000000000001000000000000
Impact on IEEE-754

- The right most « 1 » of the mantissa is called the *significance flag*

- *Significance flag* represents cancellation & uncertainty
  - Only for normal & subnormal numbers

- Others special cases remain unchanged.
  - +/- 0
  - +/- Inf
  - Nan
A new concept: Representation of error

- When all bit of the mantissa are insignificant, it does not mean that we have a ‘0’ (which is the case with IEEE-754) it means that we have the order of magnitude of the error

$$0 \ 01111111 \ 00000000000000000000000$$

(This number correspond to an exact 1 in IEEE-754)
Operation: Addition

• Similar to IEEE-754 addition + insignificant flag propagation
• Rule:
  Let $A$, $B$, and $R$ be 3 FP-ANR numbers with respectively $a$, $b$ and $r$ significant bits. The number of significant bits $r$ of the result $R = A +/- B$ is determine as follow:

\[
r = \exp_R - \text{MAX}((\exp_A - a), (\exp_B - b))
\]
Operation: Multiplication

• Similar to IEEE-754 Multiplication + insignificant flag propagation

• Rule:
  Let $A$, $B$, and $R$ be 3 FP-ANR numbers with respectively $a$, $b$ and $r$ significant bits. The number of significant bits $r$ of the result $R = A (\ast,/) B$ is approximated as follow:

$$r = \text{MIN}(a,b)$$
Rounding

• Rounding mode
  • Similar to IEEE-754 ....
  • Except that it has to be done before the position of the insignificant flag

• No more Table Maker’s Dilemma
  • Can be circumvented by integrating the uncertainty which occurs during hard to round case in the results.
  • Reproducibility can be ensure by setting a target accuracy function of the number of significant bit in the input number.
\[ t_0 = \frac{1}{\sqrt{3}} \]

\[ t_{i+1} = \frac{\sqrt{t_i^2 + 1} - 1}{t_i} \]

\[ \pi \sim 6 \times 2^i \times t_i \]
Conclusions

• FP-ANR
  • An alternative format to handle cancellation/uncertainty based on IEEE-754
  • Simple, easy to use, to understand, to implement
    (ie: execution state similar to rounding mode)
  • Next
    • Use AVX-512 features
    • Hardware implementation

<table>
<thead>
<tr>
<th></th>
<th>IEEE-754: 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP-ANR</td>
<td>ERR(2^{47})</td>
</tr>
<tr>
<td>Correct</td>
<td>1024.0</td>
</tr>
</tbody>
</table>

• Significant arithmetic
  • Offers a solution to some numerical problems
    (NOT ALL: EFT, …)
  • Decorrelation of variables is problematic
  • Can only come in support to traditional IEEE-754
Software implementation: conversion

```c
#include <ieee754.h>

/* Convert a binary32 number f to a FP-ANR number with p bits */
float Float2FpAnr(float f, int p){
    union ieee754_float d;
    d.f = f;

    prec = MIN(22, p);
    d.ieee.mantissa &= (0x7FFFFF<<(23-p));
    /* Set the significant flag */
    d.ieee.mantissa |= 1<<(22-p);
    return d.f;
}

/* Convert a FP-ANR number with p bits to a binary32 number */
float FpAnr2Fp(float f, int *p){
    union ieee754_float d;
    int c;
    d.f = f;

    if (d.ieee.mantissa!=0){
        c = count_trailing_zeros(d.ieee.mantissa);
        /* Remove the significant flag */
        d.ieee.mantissa ^= 1<<c;
    }
    *p = 22-c;
    return(d.f);
}
```

Use truncation: harder if other rounding mode is required
Software implementation: ADD

/* Addition: return a1 + a2 */
float FpAnrAdd(float a1, float a2){
    float res;
    int e1, e2, er;
    int p1, p2;

    res = FpAnr2Fp(a1, &p1) + FpAnr2Fp(a2, &p2);
    frexp(a1, &e1);
    frexp(a2, &e2);
    frexp(res, &er);
    return Float2FpAnr(res, er - MAX((e1 - p1), (e2 - p2)));
}

/* Multiplication: return a1 * a2 */
float FpAnrMul(float a1, float a2){
    float res;
    int p1, p2;

    res = FpAnr2Fp(a1, &p1) * FpAnr2Fp(a2, &p2);
    return Float2FpAnr(res, MIN(p1, p2));
}
Hardware implementation

• Similar with IEEE-754 operator with the integration of significant signal as mentioned previously

• Example: *significant signal* generation based on OR gate:
Operations: Others

• Use first order Taylor series expansion to approximate uncertainty propagation
  • \( R = f(X) = \sqrt{X}, \ r = x + 1 \)
  • \( R = f(X) = \exp(X), \ r = x - \log_2|X| \)
  • \( R = f(X) = \ln(X), \ r = x + \log_2|\ln(X)| \)
  • \( R = f(X) = \sin(X), \ r = x + \log_2|\sin(X)/X\cdot\cos(X)| \)
  • \( R = f(X) = \cos(X), \ r = x + \log_2|\cos(X)/X\cdot\sin(X)| \)

• However, \( f(X) \) has to be quasi-linear & quasi-gaussian on the interval \( [X-e_X; X+e_X] \)
Operations: interaction between FP-ANR & IEEE-754

• Use conversion:
  • FP-ANR -> IEEE-754: replace the significant flag with a 0
  • IEEE-754 -> FP-ANR: replace the right most bit of the mantissa with a 1

• Mixing FP-ANR & IEEE-754 without special support is possible
  • Do not lead to crash or irrelevant results
  • Only the insignificant bit of the mantissa will be different but it should not be a problem as those bit are considered as meaningless (correspond to noise)
Now include +1 and −1

The SORN is 8 bits long.

This is actually enough of a number system to be useful!

One option: more powers of 2

There is nothing special about 2. We could have added 10 and 1/10, or even π and 1/π, or any exact number.

(Yes, π can be numerically exact, if we want it to be!)