Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic Using the Binary Encoding Format

Marius Cornea, Cristina Anderson, John Harrison, Peter Tang, Eric Schneider, Evgeny Gvozdev, Charles Tsen

June 25, 2007

Decimal Floating-Point Applications

- Applications that involve financial computations: banking, telephone billing, tax calculation, currency conversion, insurance, accounting in general
- Current feedback indicates that decimal computations take a small fraction of the total execution time
- No indication that scientific computation will migrate to decimal arithmetic in the near future

- IEEE 754R addresses the need for good quality decimal arithmetic, and defines three basic formats: _Decimal32, _Decimal64, _Decimal128

Decimal Floating-Point Applications

- Example of decimal floating-point computation, performed with the Intel IEEE 754R Decimal Floating-Point BID library from GCC 4.3:

```
float f1 = 7.0, f2 = 10.E3, f3;
_Decimal32 d1 = 7.0, d2 = 10.E3, d3;
f3 = f1 / f2; f3 = f2 * f3;
printf ("f3 = 0x%8.8x = %f\n", *(unsigned int *)&f3, f3);
d3 = d1 / d2; d3 = d2 * d3;
printf ("d3 = 0x%8.8x = %f\n", *(unsigned int *)&d3, d3);

f3 = 0x40dfffff = 7.000000 (6.9999997504 with other compilers)
d4 = 0x32000046 = 7.000000
```

IEEE 754R Decimal Floating-Point Encoding Methods

- For example _Decimal64 numerical values are:
  $v = (-1)^s \cdot \text{significand} \cdot 10^{\text{exponent}}$
  (up to 16 digits; exp. range = [-383,384], bias = 398)

- Decimal Encoding Method: based on the Densely Packed Decimal (DPD) method - up to three decimal digits are encoded in 10-bit fields named declets (non-linear mapping)
  - the encoding is “s G E T”:
  - s = 1-bit sign
  - G = 5-bit combination field: encodes the leading decimal digit and the top two exponent bits
  - E = 8-bit exponent field - the lower 8 bits of the biased exponent
  - T = 50 lower bits of the coefficient (significand), consisting of 5 declets

IEEE 754R Decimal Floating-Point Encoding Methods

- Binary Encoding Method: based on Binary Integer Decimal (BID); the coefficient C (significand, scaled up) is a binary integer
  - the encoding is "s E C_{52-0}" if the coefficient C = $d_0 d_1 \ldots d_{15}$ represented as a binary integer fits in 53 bits
  - the encoding is "s 11 E C_{50-0}" otherwise, and C_{53-51} = 100
  - The biased exponent field E takes 10 bits

- The BID format does not require a costly conversion to/from binary format on binary hardware, which matters especially when the decimal arithmetic is implemented in software

Rounding Binary Integers to a Given Number of Decimal Digits

- Occurs in addition, subtraction, multiplication, fused-multiply add, and conversions that use the BID encoding

- Example: round the decimal value

  C = 1234567890123456789

  stored as a binary integer, from q = 19 to p = 16 decimal digits; need to round off x = 3 digits
- Straightforward method
- Better: multiply by $10^{–3}$
- If $k_3 \approx 10^{–3}$ is calculated with sufficient accuracy and rounded up, then

  floor $(C \cdot k_3)$ = 1234567890123456

  with certainty

Rounding Binary Integers to a Given Number of Decimal Digits

- Method 1: Calculate $k_3 \approx 10^{-3}$, y-bit approximation of 10^{-3} rounded up

  $\text{floor}(C \cdot k_3) = 1234567890123456 = \text{floor}(C/10^3)$

- Method 1a: Calculate $h_3 \approx 5^{-3}$, y-bit approximation of 5^{-3} rounded up

  $\text{floor}((C \cdot h_3) \cdot 2^{-3}) = 1234567890123456 = \text{floor}(C/10^3)$

- Method 2: Calculate $h_3 \approx 5^{-3}$, y-bit approximation of 5^{-3} rounded up

  $\text{floor}(\text{floor}(C \cdot 2^{-3}) \cdot h_3) = 1234567890123456 = \text{floor}(C/10^3)$

- Method 2a: Calculate $h_3 \approx 5^{-3}$, y-bit approximation of 5^{-3} rounded up

  $\text{floor}(\text{floor}(C \cdot h_3) \cdot 2^{-3}) = 1234567890123456 = \text{floor}(C/10^3)$

Basic Property for Decimal FP Arithmetic on Binary Hardware

- **Property 1:** Let $q \in N$, $q > 0$, $C \in N$, $10^{q-1} \leq C < 10^{q}-1$, $x \in \{1, 2, 3, \ldots, q-1\}$, and $\rho = \log_2 10$.

  If $y \in N$, $y \geq \text{ceiling}(\{\rho \cdot x\} + \rho \cdot q)$ and k_x is a y-bit approximation of 10^{-x} rounded up, i.e.

  $$k_x = (10^{-x})_{RP,y} = 10^{-x} \cdot (1 + \varepsilon), \qquad 0 < \varepsilon < 2^{-y+1}$$

  then

  $$\text{floor}(C \cdot k_x) = \text{floor}(C / 10^{x})$$

Correction Step for Rounding to Nearest

- **Property 2:** Let $q \in N$, $q > 0$, $x \in \{1, 2, 3, \ldots, q-1\}$,
  $C \in N$, $10^{q-1} \le C < 10^q - 1$, $C = 10^x \cdot H + L$,
  $H, L \in N$, $H \in [10^{q-x-1}, 10^{q-x} - 1]$, $L \in [0, 10^x - 1]$,
  $f = C \cdot k_x - \text{floor}(C \cdot k_x)$,
  $\rho = \log_2 10$, $y \in N$,
  $y \ge 1 + \text{ceiling}(\rho \cdot q)$,
  $k_x = 10^{-x} \cdot (1 + \varepsilon)$ $\quad 0 < \varepsilon < 2^{-y+1}$

  Then the following are true:
  (a) $C \cdot 10^{-x} = H$ iff $0 < f < 10^{-x}$
  (b) $H < C \cdot 10^{-x} < (H + 1/2)$ iff $10^{-x} < f < 1/2$
  (c) $C \cdot 10^{-x} = (H + 1/2)$ iff $1/2 < f < 1/2 + 10^{-x}$
  (d) $(H + 1/2) < C \cdot 10^{-x} < (H + 1)$ iff $1/2 + 10^{-x} < f < 1$

Reducing the Length of Constants k_x

- Property 2 also helps reduce the length of some of the constants k_x
- Reduce the accuracy of k_x one bit at a time, and verify that for $H = 10^{q-x} - 1$:

(a) $H \cdot 10^x \cdot k_x < H + 10^{-x}$
(b) $(H + 1/2 - 10^{-x}) \cdot 10^x \cdot k_x < H + 1/2$
(c) $(H + 1/2) \cdot 10^x \cdot k_x < H + 1/2 + 10^{-x}$
(d) $(H + 1 - 10^{-x}) \cdot 10^x \cdot k_x < H + 1$

- For example k_3 is reduced from $y = 65$ to $y = 62$ bits

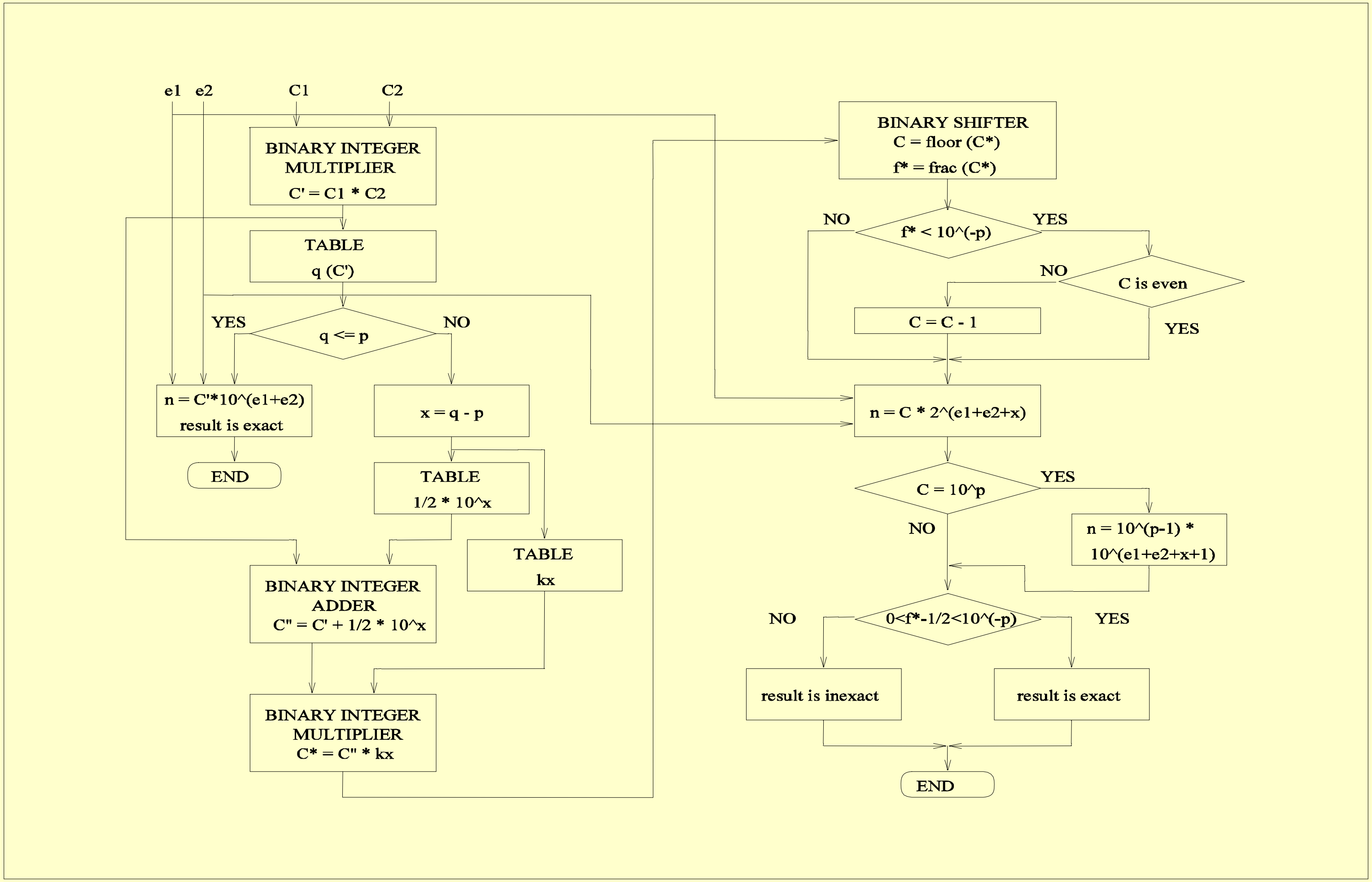
Software Implementation of the IEEE 754R Decimal FP Arithmetic

- The values k_x for all x of interest are pre-calculated and are stored as pairs (K_x, e_x) with K_x and e_x positive integers, and $k_x = K_x \cdot 2^{-ex}$.
- The algorithms and operations presented here represent the core of a generic implementation in C of the IEEE 754R decimal floating-point arithmetic
- Test runs for several hardware configurations, operating systems, compilers, little/big endian, build options

Software Implementation of the IEEE 754R Decimal FP Arithmetic

- Several decimal floating-point operations, in particular addition, subtraction, multiplication, fused multiply-add, and most conversions could be implemented efficiently using operations in the integer domain
- An important property is that when rounding the exact result to p digits, the information necessary to determine whether the result is exact (in the IEEE 754 sense) or perhaps a midpoint, is available in the product $C \cdot k_x$ itself
- For division and square root, the algorithms are based on scaling the operands so as to bring the results into desired integer ranges, in conjunction with a few floating-point operations and one or two refinement iterations

Example: Decimal floating-point multiplication with rounding to nearest using hardware for binary operations. From n1 = C1 · 10^{e1} and n2 = C2 · 10^{e2} the product n = $(n1 \cdot n2)_{RN,p}$ = C · 10^{e} is calculated.



Software Implementation of the IEEE 754R Decimal FP Arithmetic

- Mixed-format floating-point operations, e.g. with operands of precision N0 and result of precision N (N0 > N), are replaced by:
  - similar, existing operation with operands of precision N0 and result of precision N0
  - conversion from precision N0 to precision N
  - logic to avoid double rounding errors
- Conversions between binary and decimal floating-point formats
  - There is a finite, and relatively small number of (decimal, binary) exponent pairs that can occur in conversions
  - For each pair use continued fractions to show that the relative error when a binary floating-point number is approximated by a decimal one (or vice-versa) for inexact conversions, has a lower bound which sets an upper bound on the intermediate precision needed to achieve correct IEEE conversion

Performance Results - Clock Cycle Counts for a Subset of Decimal FP Arithmetic Functions (Intel Xeon 5100)

Oper.	Min	Max	Med
add64	14	140	80
mul64	22	140	40/130
fma64	61	307	200
div64	58	269	170
sqrt64	35	192	180
add128	80	224	150
mul128	121	655	550
fma128	299	1036	650
div128	157	831	550
sqrt128	227	947	900

Operation	Min	Max	Med
bid64_to_bid128	8	12	8
bid128_to_bid64	125	174	145
dbl_to_bid128	123	375	375
bid128_to_dbl	160	185	160
int64_to_bid128	5	5	5
bid128_to_int64	31	138	121
bid64_quiet_less	31	69	34
bid128_quiet_less	8	114	60

Conclusion

- Beta version available for download at http://www3.intel.com/cd/software/products/asmo-na/eng/219861.htm
- Next release in July 2007
- Opportunity for improving performance exists
- Possible future work:
  - Implement optional parts of IEEE 754R
  - Implement specific operations required by C/C++ Standards TRs on Decimal Floating-Point Arithmetic
  - Optimize