Interval arithmetic on the Cell processor

Stef Graillat    Jean-Luc Lamotte    Siegfried M. Rump
Svetoslav Markov

LIP6/PEQUAN, P. and M. Curie University, Paris
Institute for Reliable Computing, Hamburg University of Technology
Institute of Mathematics and Computer Science, Bulgarian Academy of Sciences

13th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numerical Computations SCAN’08
El Paso, Texas, USA, September 29 - October 3, 2008
Overview

The Cell processor

Interval Arithmetic on Cell processor

Conclusions
The Cell processor

SP > 200 GFlops, DP=15 Gflops, 25GB/s memory BW, 300 GB/s EIB
Synergistic Processing Element SPE (1/2)

The SPE is a small processor with a vectorial unit.

- small memory (256 KB) for instructions and data, named “local store” (LS)
- 128 registers of 128 bits
- 1 SPU “Synergistic Processing Unit”
  - 4 units for single precision computation
  - 1 unit for double precision computation
- MFC “Memory Flow Controller” which manages memory access through DMA
Synergistic Processing Element SPE (2/2)

128-bit registers:

- 16 integers of 8 bit,
- 8 integers of 16 bit,
- 4 integers of 32 bit,
- 4 single precision floating point numbers,
- 2 double precision floating point numbers.

The SIMD processor is based on FMA and is fully pipelined in SP:

- Peak performance SP: $4 \times 2 \times 3.2 = 25.6 \text{GFLOPs}$

Not fully pipelined in double precision:

- Peak performance in DP: $2 \times 2 \times 3.2/7 = 1.8 \text{GFLOPs}$
Parallelism on Cell

3 levels of parallelism:

1. processes run on Cell processors, exchange with a MPI library,
2. Data distribution and communication on the 8 SPE,
   - ALF, Dacs
   - POSX thread, CELL thread,
   - mailing box, exchange through DMA
   - data need to be aligned on quadword
   - double buffering technique
3. inside a thread
   - only 256 KB
   - Altivec programming
   - code and data dependencies: not to break the SIMD pipeline
The performance price on SPE

No division
1/x and 1/√x : only the 12 first bits are exact.

SPU float arithmetic is not IEEE compliant :

- only rounding mode to zero (truncation).
- The highest exponent (128) is used not for Infinity or NaN, but is used to extend the range of the floating point.
- Inf and NaN are not recognized by arithmetic operations.
- Overflow results saturate to the largest representable positive or negative values, rather than producing +/-IEEE Infinity.
- No denormalized results : +0 instead.
SPU double arithmetic is IEEE compliant except:

- FP trapping is not supported.
- Denormalized operands are treated as 0.
- NaN results are always the default QNaN (Quiet NaN)
Reliable computing on Cell processor

- difficult to implement interval arithmetic.
- possible to “emulate” a rounding mode toward $+\infty$
  if $r \in \mathbb{R}$ non-negative, $\text{fl}_0(r) \leq r \leq \text{succ}(\text{fl}_0(r))$
  and
  \[
  \text{succ}(f) = \max\{\text{fl}_0((1 + 2u)f), \text{fl}_0(f + u)\}.
  \]
  where $u$ is the relative rounding error and $\underline{u}$ the underflow unit

On the Cell processor, no underflow

- $\text{succ}(f) = \text{fl}_0((1 + 2u)f)$ if $f > 0$
- $\frac{1}{2}\underline{u}u^{-1}$ if $f = 0$
Interval with a rounding mode toward zero

Three representations:

• endpoint
• center-radius
• leftpoint-diameter
Let $A = [a_{inf}, a_{sup}]$, $B = [b_{inf}, b_{sup}]$ and $C = [c_{inf}, c_{sup}]$ be three intervals, $C = A + B$ is defined by:
Let $\oplus$ and $\otimes$ be the floating point addition and multiplication with rounding toward zero.

$$
c_{inf} = \begin{cases} 
- \text{succ}(|a_{inf} \oplus b_{inf}|) & \text{if } (a_{inf} \oplus b_{inf}) < 0 \\
 a_{inf} \oplus b_{inf} & \text{if } (a_{inf} \oplus b_{inf}) > 0 \\
 - \frac{1}{2} uu^{-1} & \text{if } (a_{inf} \oplus b_{inf}) = 0 
\end{cases}
$$

$$
c_{sup} = \begin{cases} 
\text{succ}(a_{sup} \oplus b_{sup}) & \text{if } (a_{sup} \oplus b_{sup}) > 0 \\
 a_{inf} \oplus b_{inf} & \text{if } (a_{sup} \oplus b_{sup}) < 0 \\
 \frac{1}{2} uu^{-1} & \text{if } (a_{sup} \oplus b_{sup}) = 0 
\end{cases}
$$
Endpoint representation — multiplication

\[ x = \min(a_{\text{inf}} \otimes b_{\text{inf}}, a_{\text{inf}} \otimes b_{\text{sup}}, a_{\text{sup}} \otimes b_{\text{inf}}, a_{\text{sup}} \otimes b_{\text{sup}}) \]
\[ y = \max(a_{\text{inf}} \otimes b_{\text{inf}}, a_{\text{inf}} \otimes b_{\text{sup}}, a_{\text{sup}} \otimes b_{\text{inf}}, a_{\text{sup}} \otimes b_{\text{sup}}) \]

\[ C = A \times B \text{ is defined by :} \]

\[ c_{\text{inf}} = \begin{cases} 
- \text{succ}(|x|) & \text{if } x < 0 \\
- \frac{1}{2}uu^{-1} & \text{if } x = 0 \\
x & \text{else}
\end{cases} \]

\[ c_{\text{sup}} = \begin{cases} 
\text{succ}(y) & \text{if } y > 0 \\
\frac{1}{2}uu^{-1} & \text{if } y = 0 \\
y & \text{if } y < 0
\end{cases} \]
Let $A = [a, \alpha]$, $B = [b, \beta]$ and $C = [c, \gamma]$ be three intervals.

Rump's algorithm:

$$c = \Box (a + b)$$
$$\gamma = \triangle (2u \cdot |c| + \alpha + \beta)$$

$C = A + B$ is defined by:

- $c = a \oplus b$
- $\gamma = \text{succ}(2u \otimes |c| \oplus \text{succ}(\alpha \oplus \beta))$
Center-radius — multiplication

Rump’s algorithm:

\[ c = \Box(a \cdot b) \]
\[ \gamma = \triangle(u + 2u \cdot |c| + (|a| + \alpha)\beta + \alpha|b|) \]

\( C = A \times B \) is defined by:

\[ c = a \otimes b \]
\[ \gamma = \text{succ}(\text{succ}(2u \otimes |c| \oplus \text{succ}(\text{succ}(|a| \oplus \alpha) \otimes \beta)) \oplus \text{succ}(\alpha \otimes |b|)) \]
Implementation of intervals vectors

typedef struct{
    float center;
    float radius;
} T_INTERVAL;

T_INTERVAL x[...]

All the SIMD operations use vectors of four 32-bit floating point numbers.
## Performances on 1 SPE

<table>
<thead>
<tr>
<th>Operations</th>
<th>MFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle (function call)</td>
<td>477.6</td>
</tr>
<tr>
<td>Add [crrcr]</td>
<td>273.5</td>
</tr>
<tr>
<td>Add [ccrr]</td>
<td>345.9</td>
</tr>
<tr>
<td>Mul [crrc]</td>
<td>244.3</td>
</tr>
<tr>
<td>Mul [ccrr]</td>
<td>255.1</td>
</tr>
<tr>
<td>Add inf-sup</td>
<td>285.7</td>
</tr>
</tbody>
</table>
Center-diameter representation

Seems to be useful with rounding toward zero!

\[ A = (a, \alpha) = \begin{cases} 
{x \in \mathbb{R} : a \leq x \leq a + \alpha} & \text{if } a \geq 0 \\
{x \in \mathbb{R} : a - \alpha \leq x \leq a} & \text{if } a < 0 
\end{cases} \]

But very difficult to implement!
Conclusions

- hard programming job
- necessity to develop complex algorithms to reach a high level of performance.
- to prepare the work for the new Cell:
  - fully pipelined double precision floating point number (100 DP GFIOPS)
  - up to now no information on the floating point quality
Rumours on the next generation

- IEEE compliant
- from 8 to 32 SPE
- over 1TFLOPS
Acknowledgment

Thanks to the CINES Center for providing IBM Cell Blade access.