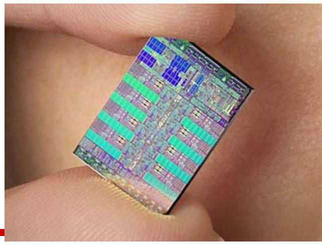
Experiments with the Cell for Linear Algebra Operations

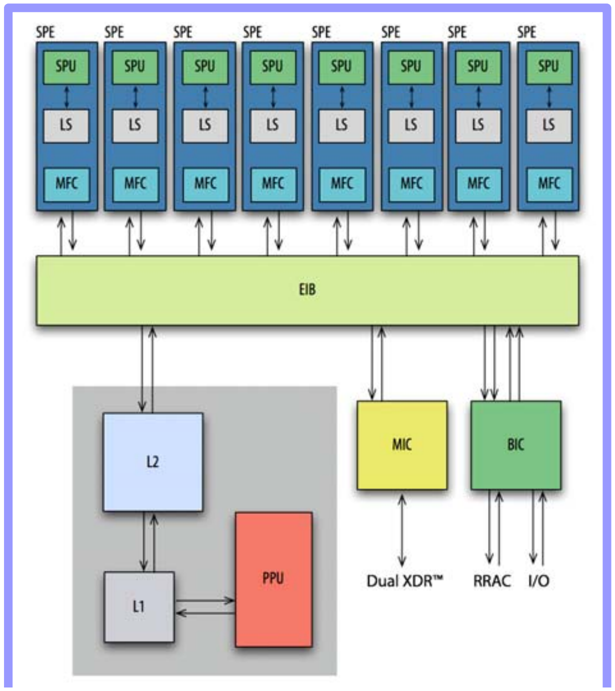
Jack Dongarra
INNOVATIVE COMPUTING LABORATORY

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

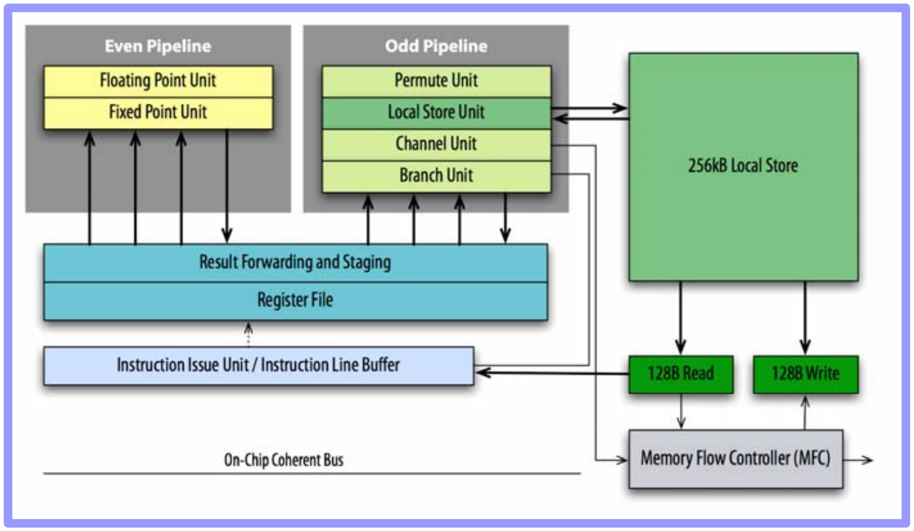
Exploiting Mixed Precision



- **Current Version of the Cell has > a factor of 10 between single precision and double prevision performance (204 GFlop/s to 14 GFlop/s)!**
  - **Next version this will narrow to a factor of 2 (as in most common processors today)**
- **We became interested in looking for ways to exploit the speed of SP but still retain the accuracy of DP.**



SPE ~ 25 Gflop/s peak



Idea Goes Something Like This…

- **Exploit 32 bit floating point as much as possible.**
  - **Especially for the bulk of the computation**
- **Correct or update the solution with selective use of 64 bit floating point to provide a refined results**
- **Intuitively:**
  - **Compute a 32 bit result,**
  - **Calculate a correction to 32 bit result using selected higher precision and,**
  - **Perform the update of the 32 bit results with the correction using high precision.**

Mixed-Precision Iterative Refinement

- **Iterative refinement for dense systems, $Ax = b$, can work this way.**

L U = lu(A)	$O(n^3)$
x = L\(U\b)	$O(n^2)$
r = b – Ax	$O(n^2)$
WHILE || r || not small enough	
z = L\(U\r)	$O(n^2)$
x = x + z	$O(n^1)$
r = b – Ax	$O(n^2)$
END	

  - **Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.**

Mixed-Precision Iterative Refinement

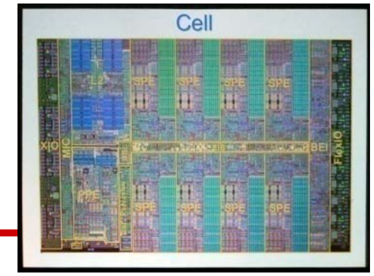
- **Iterative refinement for dense systems, $Ax = b$, can work this way.**

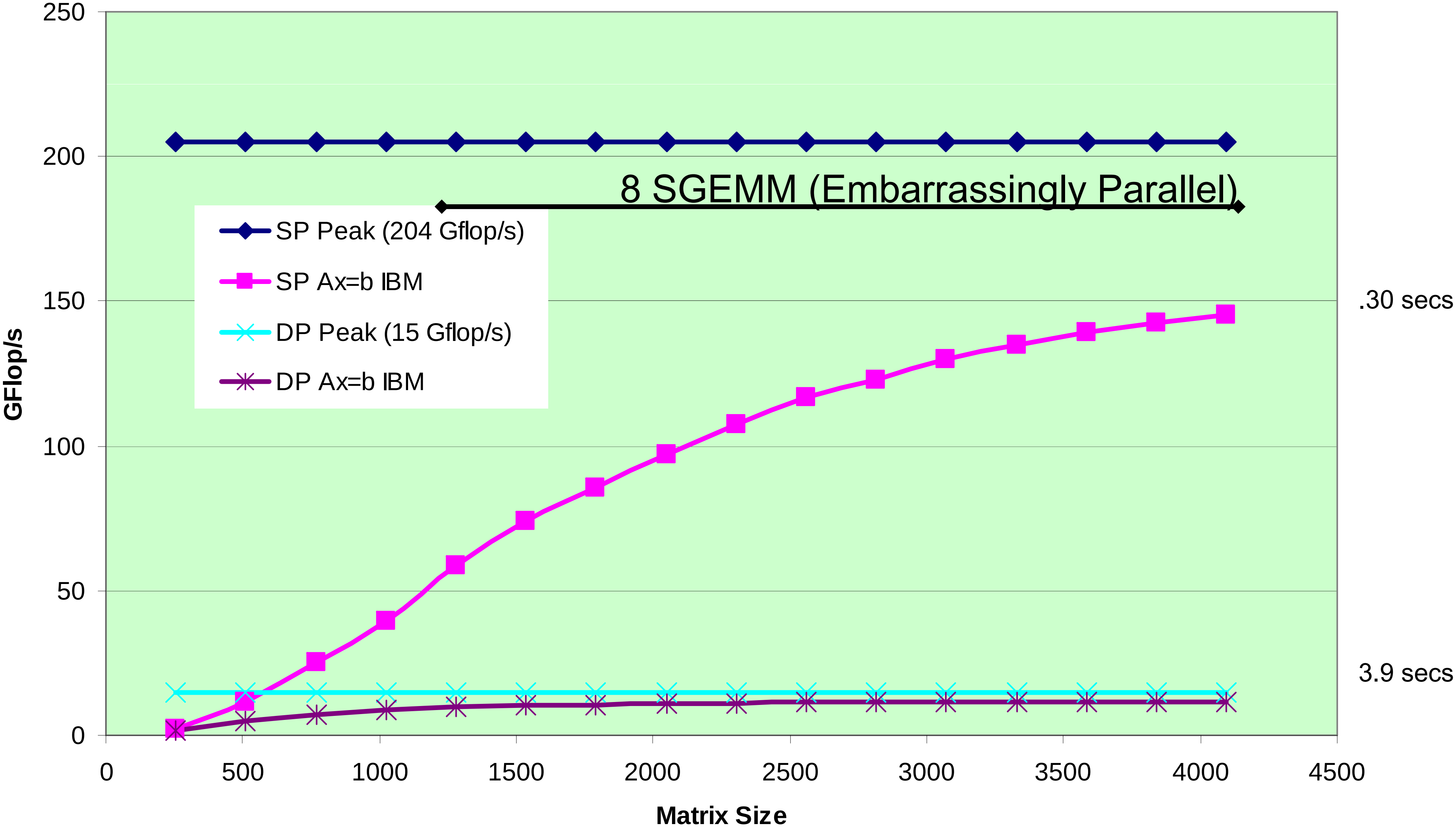
L U = lu(A)	SINGLE	$O(n^3)$
x = L\(U\b)	SINGLE	$O(n^2)$
r = b – Ax	DOUBLE	$O(n^2)$
WHILE || r || not small enough		
z = L\(U\r)	SINGLE	$O(n^2)$
x = x + z	DOUBLE	$O(n^1)$
r = b – Ax	DOUBLE	$O(n^2)$
END		

- **Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.**
- **It can be shown that using this approach we can compute the solution to 64-bit floating point precision.**

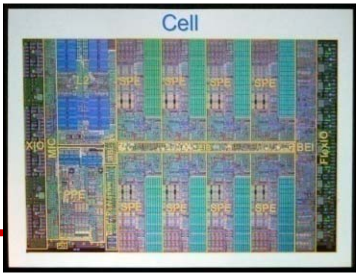
> - **Requires extra storage, total is 1.5 times normal;**
> - **$O(n^3)$ work is done in lower precision**
> - **$O(n^2)$ work is done in high precision**
>
>
>
> - **Problems if the matrix is ill-conditioned in sp; $O(10^8)$**

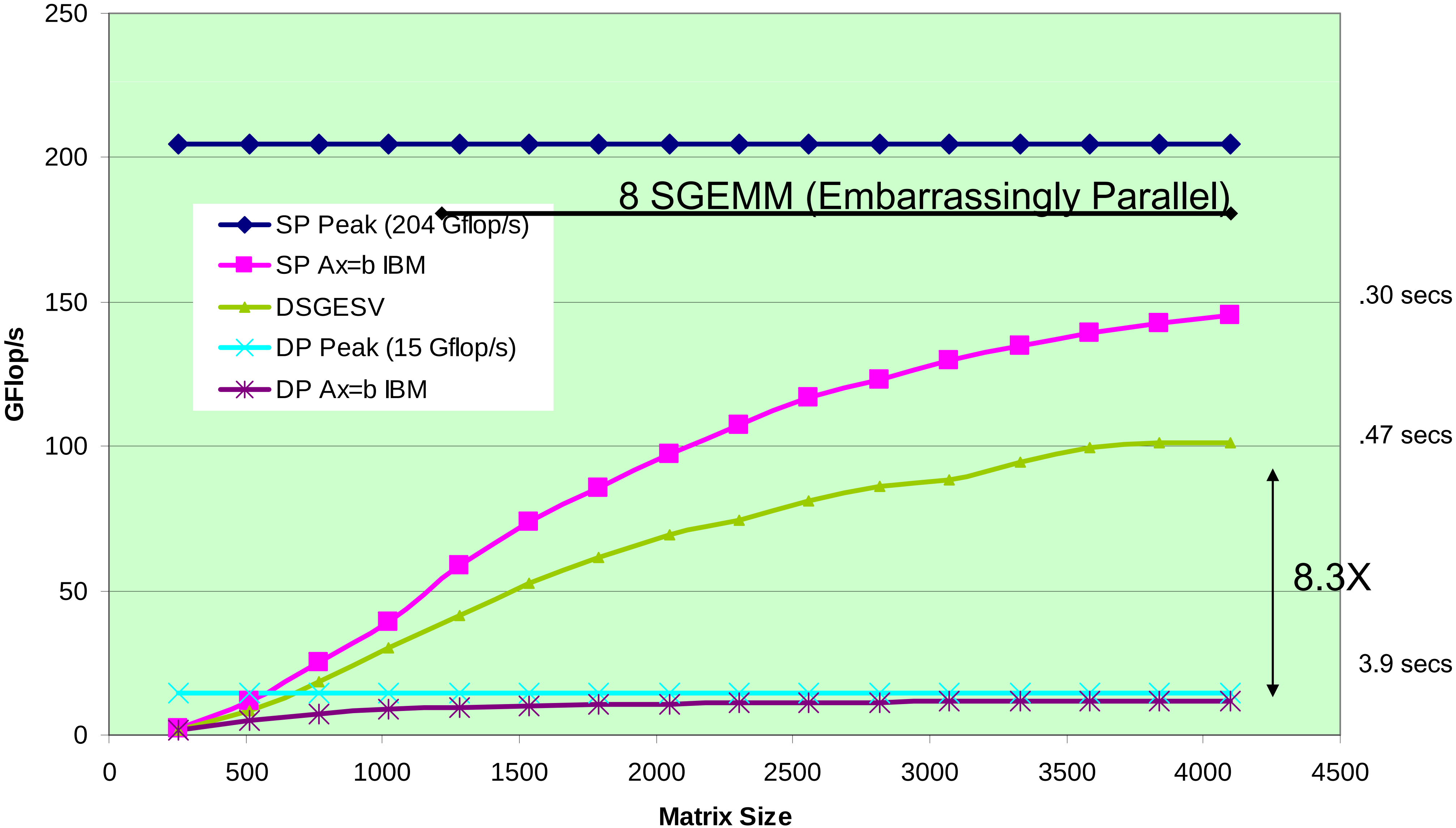
IBM Cell 3.2 GHz, Ax = b



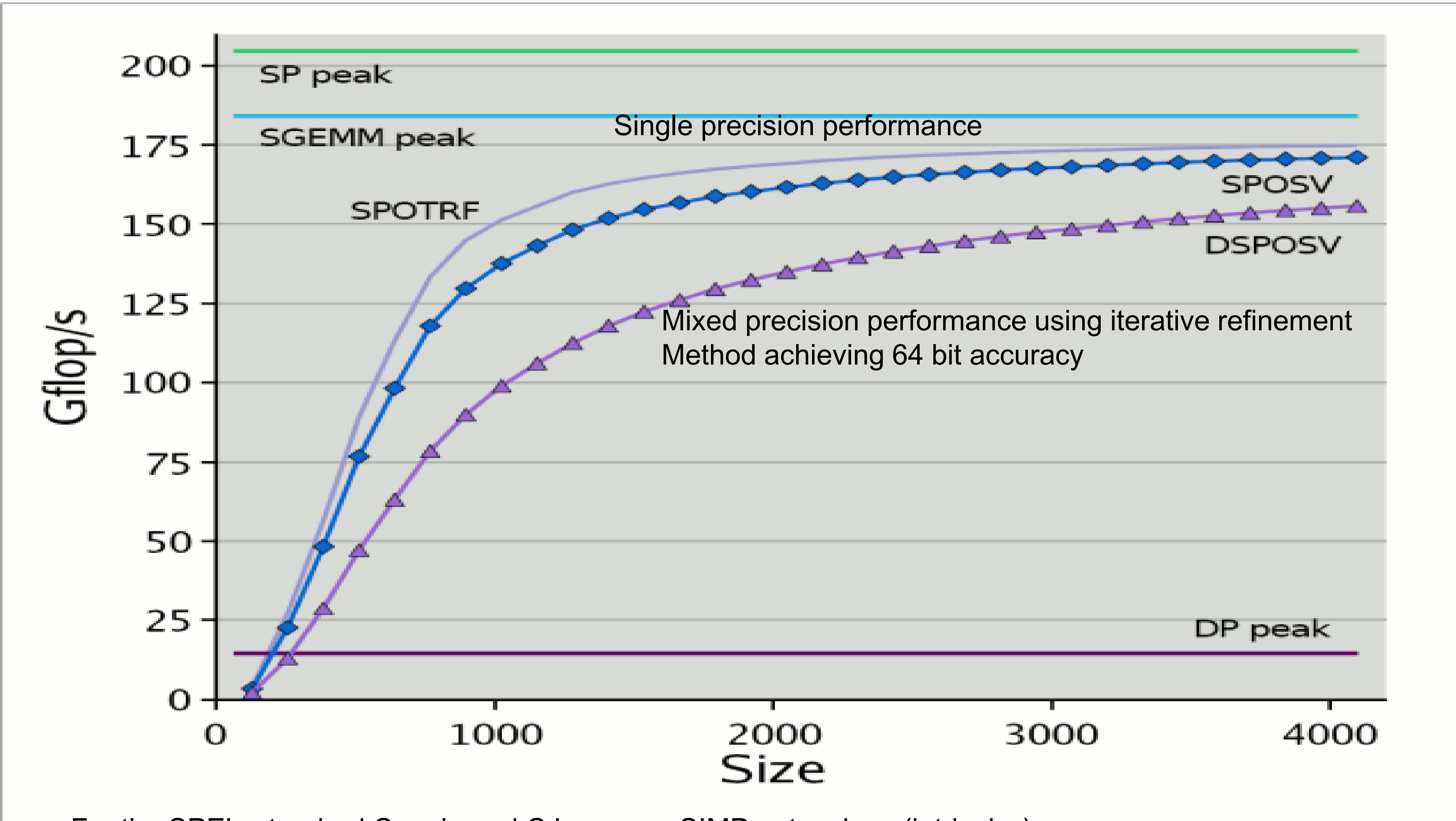


IBM Cell 3.2 GHz, Ax = b



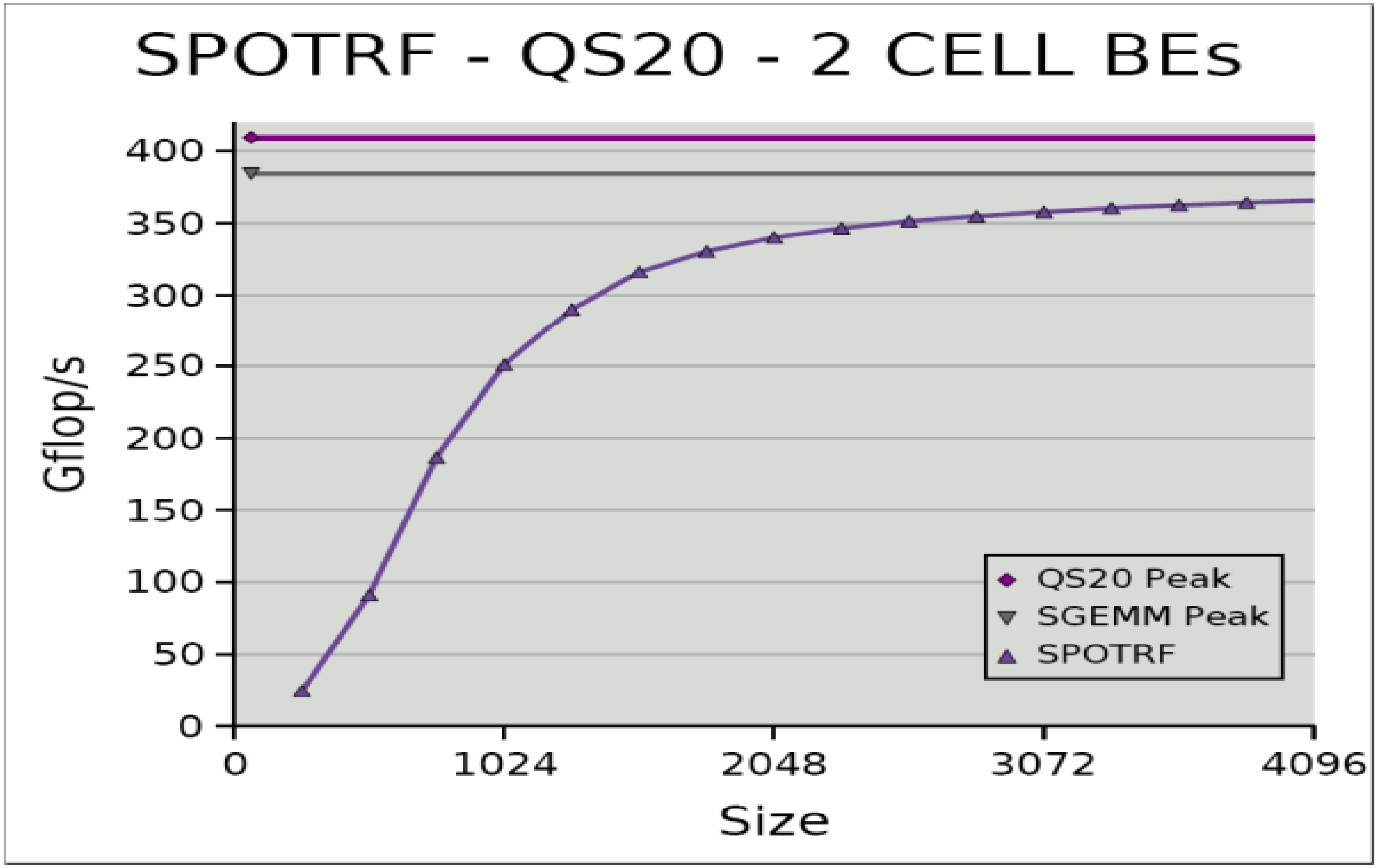


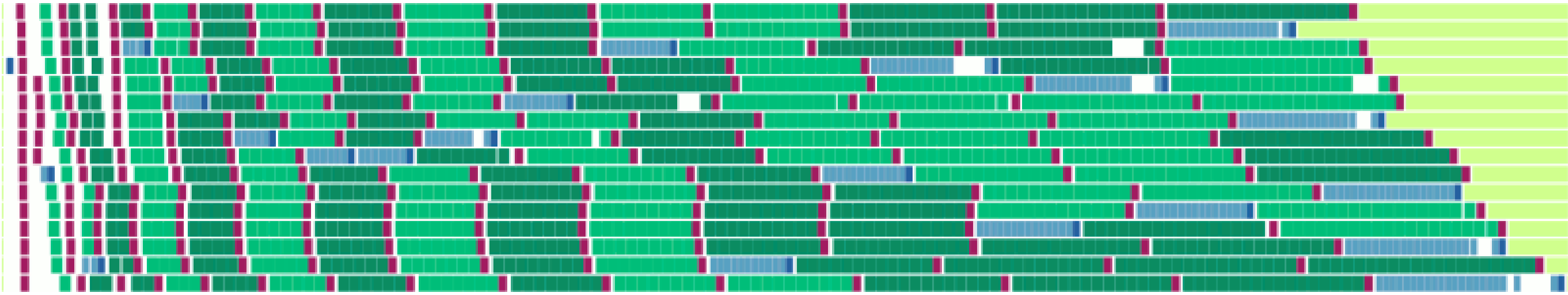
Cholesky on the Cell, $Ax=b$, $A=A^T$, $x^TAx > 0$



For the SPE's standard C code and C language SIMD extensions (intrinsics)

Cholesky - Using 2 Cell Chips





Intriguing Potential

- **Exploit lower precision as much as possible**
  - **Payoff in performance**
    - **Faster floating point**
    - **Less data to move**
- **Automatically switch between SP and DP to match the desired accuracy**
  - **Compute solution in SP and then a correction to the solution in DP**
- **Potential for GPU, FPGA, special purpose processors**
  - **What about 16 bit floating point?**
    - **Use as little you can get away with and improve the accuracy**
- **Applies to sparse direct and iterative linear systems and Eigenvalue, optimization problems, where Newton's method is used.**

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} - x_i = -\frac{f(x_i)}{f'(x_i)}$$

Correction = - A\(b – Ax)