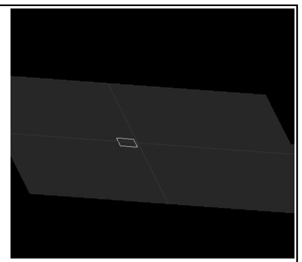
Mapping of the FDTD computations in a streaming model architecture

Adam Smyk[1], Marek Tudruj[12],
[1] Polish-Japanese Institute of Information Technology,
86 Koszykowa Str., 02-008 Warsaw, Poland
[2] Institute of Computer Science, Polish Academy of Sciences,
21 Ordona Str., 01-237 Warsaw, Poland,

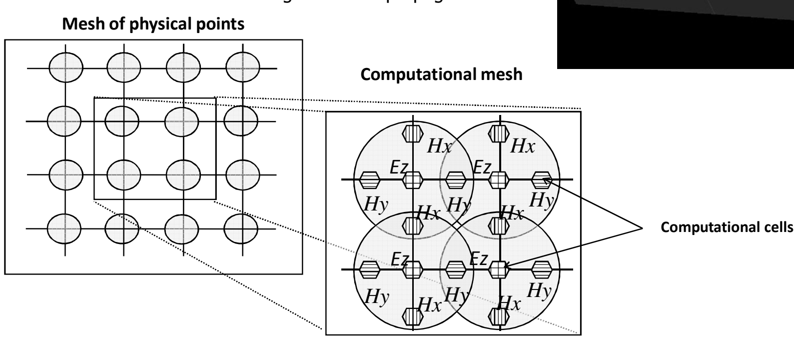
{asmyk,tudruj}@pjwstk.edu.pl

Outline

Introduction



- The Finite Difference Time Domain (FDTD) method is an iterative algorithm, which enables performing a simulation of the electromagnetic wave propagation.

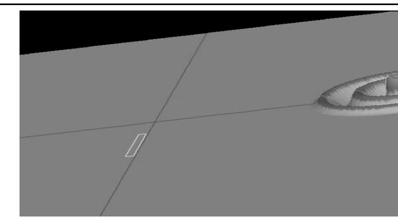


- It is done by solving Maxwell equations.

$$\nabla \times H = \gamma E + \varepsilon \frac{\partial E}{\partial t}, \qquad \nabla \times E = -\mu \frac{\partial H}{\partial t} \tag{1}$$

$$\begin{cases} \overline{H}_y^{n}(i,j) = \overline{H}_y^{n-1}(i,j) + RC \cdot [\overline{E}_z^{n-0.5}(i,j-1) - \overline{E}_z^{n-0.5}(i,j+1)], \\ \overline{H}_x^{n}(i,j) = \overline{H}_x^{n-1}(i,j) + RC \cdot [\overline{E}_z^{n-0.5}(i-1,j) - \overline{E}_z^{n-0.5}(i+1,j)], \\ \overline{E}_z^{n}(i,j) = CA_z(i,j) \cdot \overline{E}_z^{n-1}(i,j) + CB_z(i,j) \cdot [\overline{H}_y^{n-0.5}(i+1,j) - \overline{H}_y^{n-0.5}(i-1,j) + \overline{H}_x^{n-0.5}(i,j-1) - \overline{H}_x^{n-0.5}(i,j+1)] \end{cases} \tag{2}$$

Introduction cnd.



$$\nabla \times H = \gamma E + \varepsilon \frac{\partial E}{\partial t}, \qquad \nabla \times E = -\mu \frac{\partial H}{\partial t} \tag{1}$$

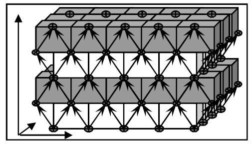
$$\begin{cases} \overline{H}_y^{n}(i,j) = \overline{H}_y^{n-1}(i,j) + RC \cdot [\overline{E}_z^{n-0.5}(i,j-1) - \overline{E}_z^{n-0.5}(i,j+1)], \\ \overline{H}_x^{n}(i,j) = \overline{H}_x^{n-1}(i,j) + RC \cdot [\overline{E}_z^{n-0.5}(i-1,j) - \overline{E}_z^{n-0.5}(i+1,j)], \\ \overline{E}_z^{n}(i,j) = CA_z(i,j) \cdot \overline{E}_z^{n-1}(i,j) + CB_z(i,j) \cdot [\overline{H}_y^{n-0.5}(i+1,j) - \overline{H}_y^{n-0.5}(i-1,j) + \overline{H}_x^{n-0.5}(i,j-1) - \overline{H}_x^{n-0.5}(i,j+1)] \end{cases} \tag{2}$$

```
For(t=0; t<numberOfIterations; t++) {
  For(i=0; i<getMaxX(); i+=2)
    For(j=0; j<getMaxY(); j+=2)
       T[i][j]=C*T[i][j]+D*(T[i+1][j]-T[i-1][j]-T[i][j+1]-T[i][j-1])

  For(i=0; i<getMaxX(); i+=2)
    For(j=1; j<getMaxY()-1; j+=2)
       T[i][j]=T[i][j]+F*(T[i][j-1]-T[i][j+1])

  For(i=1; i<getMaxX(); i+=2)
    For(j=1; j<getMaxY(); j+=2)
       T[i][j]=T[i][j]+F*(T[i-1][j]-T[i+1][j])
}
```

Loop tiling theory



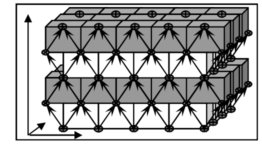
In „Loop titling for paralellism" (Jingling Hue):

Loop tiling is one of the most important iteration-reordering loop transformation.

- Maximize parallelism
- Improve memory hierarchy performane
- Reduce communication and synchronization time
- It is beneficial for both parallel machines and uniprocessors with multi-levels cache

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

Loop tiling theory



Example: Matrix-vector multiplication

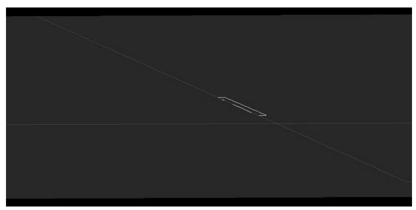
```
for (i=0; i<N; i++)
        for (j=0; j<N; j++)
                c[i] = c[i]+ a[i][j]*b[j];
```

Idea: Loop tiling partitions a loop's iteration space into smaller chunks or blocks

```
for (i=0; i<N; i+=3)
   for (j=0; j<N; j+=3)
      for (ii=i; ii<min(i+3,N); ii++)
         for (jj=j; jj<min(j+3,N); jj++)
            c[ii] =c[ii]+ a[ii][jj]*b[jj];
```

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

Loop tiling theory – Dependency set



```
For(t=0; t<numberOfIterations; t++) {
  For(i=0; i<getMaxX(); i+=2)
    For(j=0; j<getMaxY(); j+=2)
      T[i][j]=C*T[i][j]+D*(T[i+1][j]-T[i-1][j]-T[i][j+1]-T[i][j-1])

  For(i=0; i<getMaxX(); i+=2)
    For(j=1; j<getMaxY()-1; j+=2)
      T[i][j]=T[i][j]+F*(T[i][j-1]-T[i][j+1])

  For(i=1; i<getMaxX(); i+=2)
    For(j=1; j<getMaxY(); j+=2)
      T[i][j]=T[i][j]+F*(T[i-1][j]-T[i+1][j])
}
```

D_{ez}={(1,0,0),(1,-1,0),(1,1,0),(1,0,-1),(1,0,1)}

D_{hy}={(1,0,0),(1,0,1),(1,0,-1)}
D_{hx}={(1,0,0),(1,1,0),(1,-1,0)}

Dependency set for the two dimensional FDTD problem

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

FDTD tiling theory – iteration space graph

Dependency set

D_{ez}={(1,0,0),(1,-1,0),(1,1,0),(1,0,-1),(1,0,1)}

D_{hy}={(1,0,0),(1,0,1),(1,0,-1)}
D_{hx}={(1,0,0),(1,1,0),(1,-1,0)}

Distance Vectors (Dependency vectors)

Direction value

D_{ez}={…,…,…,(1,0,-1),…}

Time distance value

```
For(t=0; t<numberOfIterations; t++){
   ............
```

Space distance values

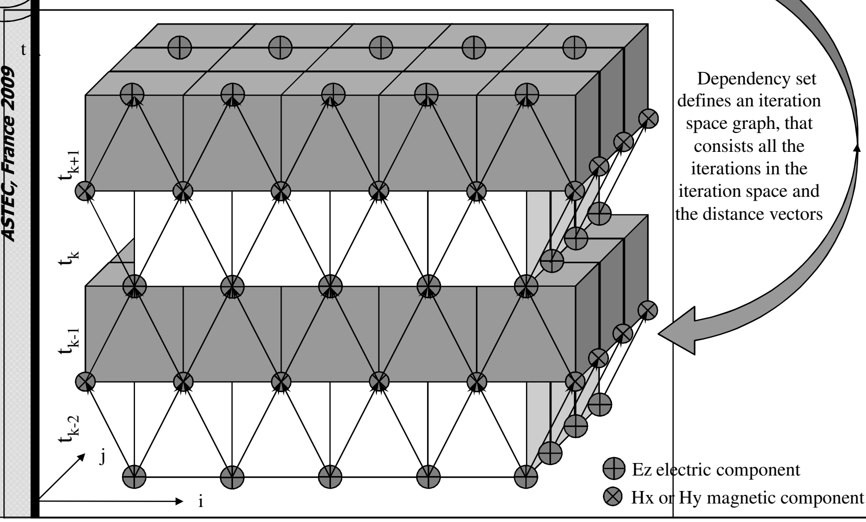
```
For(i=1; i<getMaxX(); i+=2) {
      For(j=1; j<getMaxY(); j+=2) {
         .........
      }
}
```

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

FDTD tiling theory – iteration space graph

D_{ez}={(1,0,0),(1,-1,0),(1,1,0),(1,0,-1),(1,0,1)}

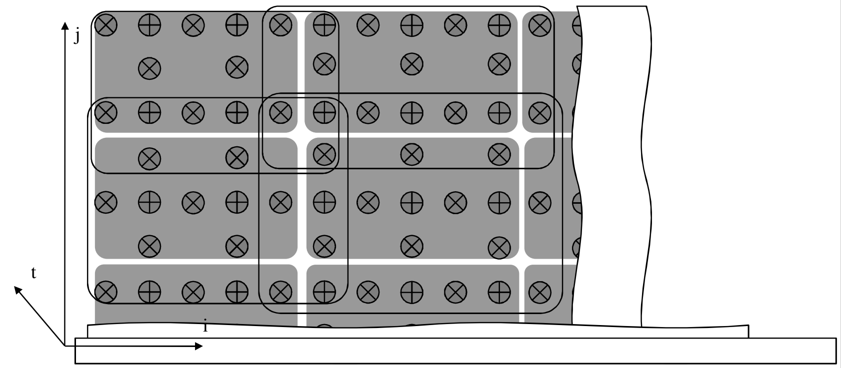
D_{hy}={(1,0,0),(1,0,1),(1,0,-1)}
D_{hx}={(1,0,0),(1,1,0),(1,-1,0)}



Dependency set defines an iteration space graph, that consists all the iterations in the iteration space and the distance vectors

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

FDTD tiling theory – loop space tiling



5x3 rectangular tiling of the FDTD ISG.

D_{ez}={(1,0,0),(1,-1,0),(1,1,0),(1,0,-1),(1,0,1)}

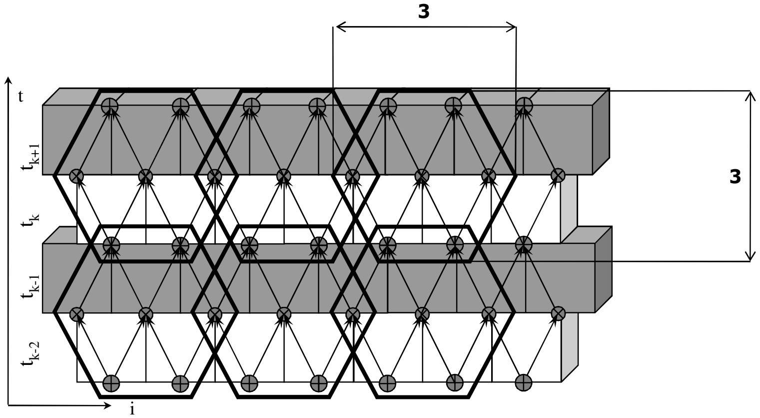
D_{hy}={(1,0,0),(1,0,1),(1,0,-1)}
D_{hx}={(1,0,0),(1,1,0),(1,-1,0)}

Rectangular tile 5x3

Overlapping boundary

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

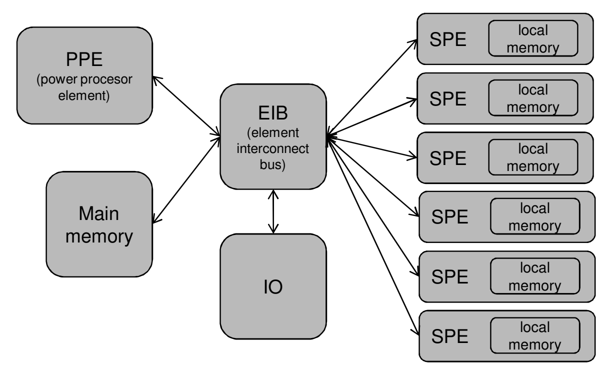
FDTD tiling theory – loop time tiling



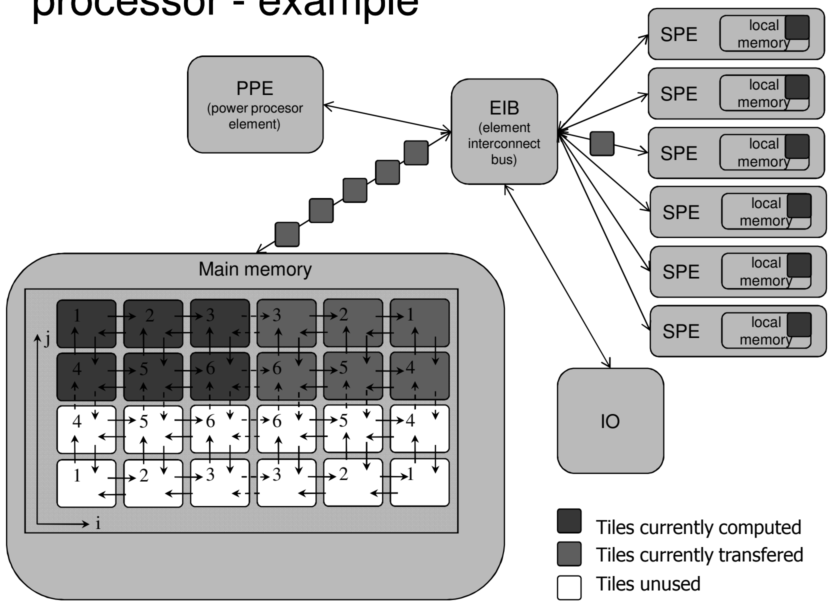
Hexagon shape time tiling for the FDTD 2D (height=3, width=3).

Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

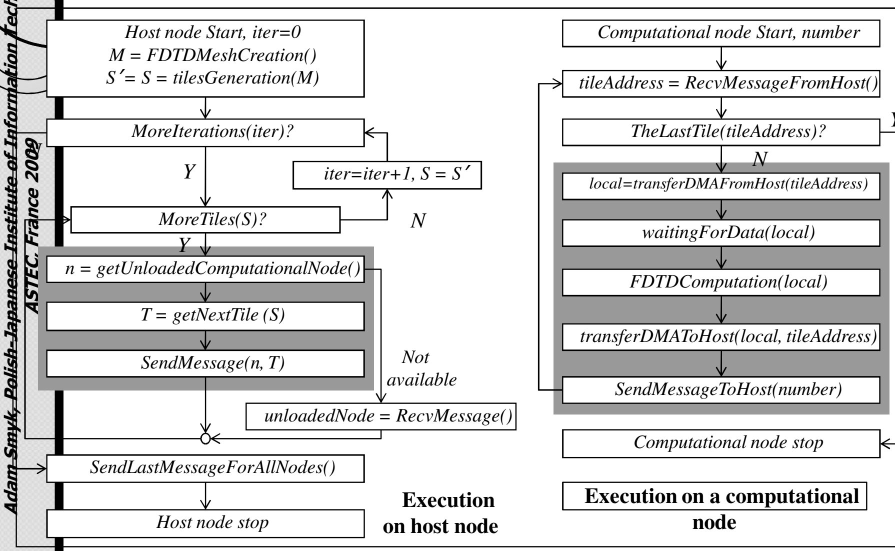
Streaming model architecture Cell/BE processor - example



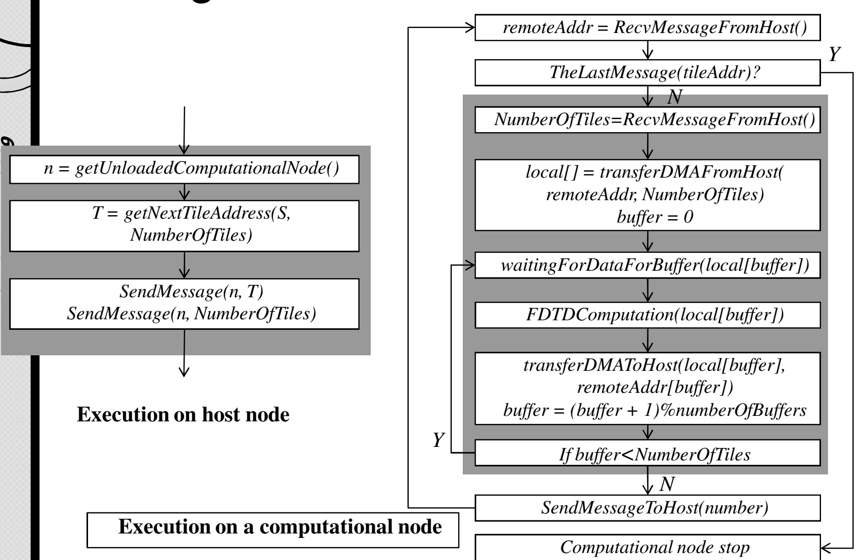
Streaming model architecture Cell/BE processor - example



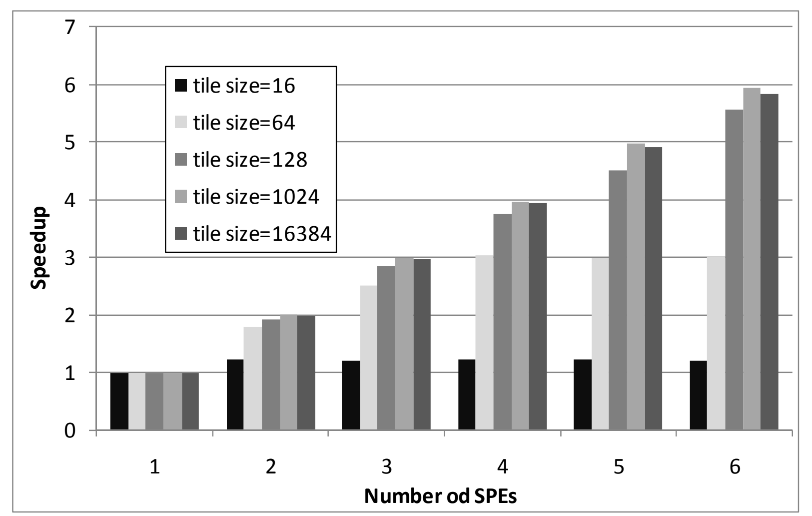
The FDTD computation scheme.



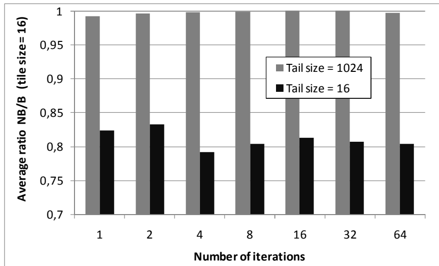
The FDTD computation scheme with rotating buffers modification.

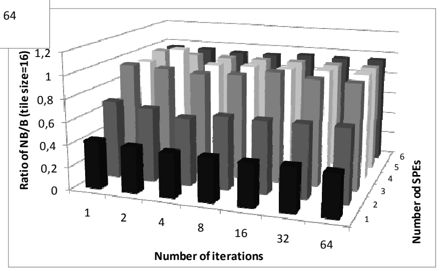


Speedup for FDTD computations for various tile sizes (space rectangular tiling).



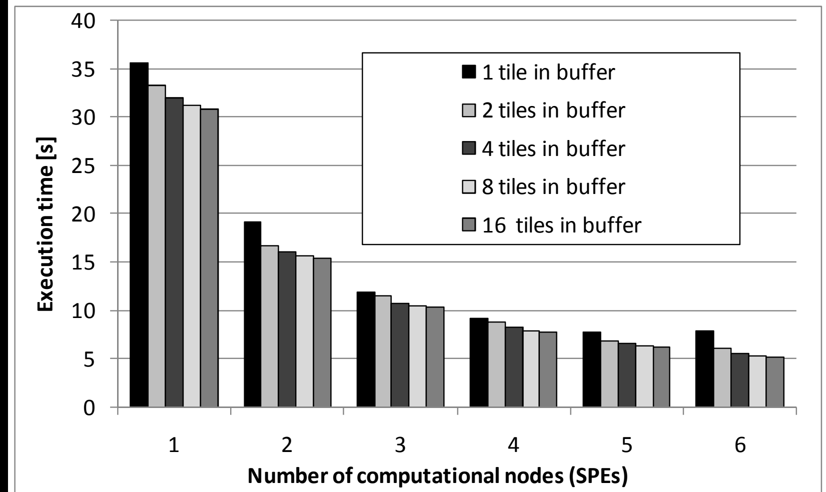
Comparison of efficiency of the FDTD computations with non-blocking and blocking mailbox communication.





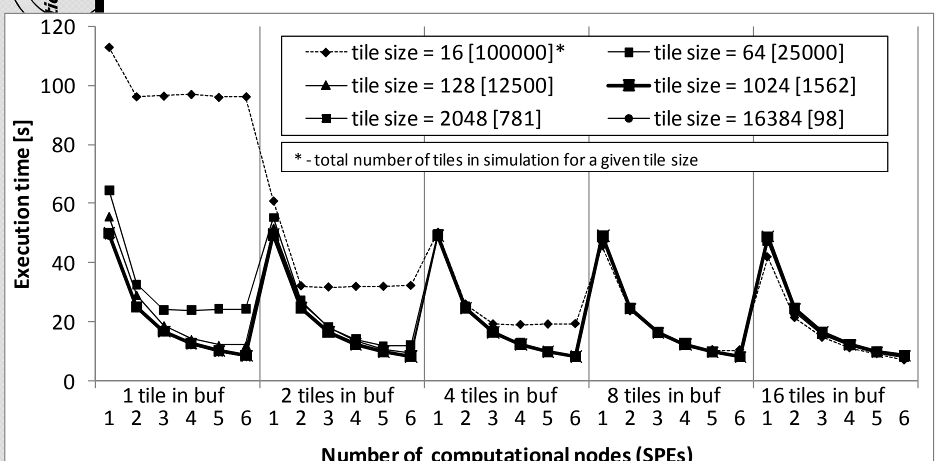
Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

Execution time of the FDTD computations with the rotating-buffers optimization (for tile size = 128)



Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

Execution time for FDTD computation for various "tile" configurations.



Adam Smyk, Polish-Japanese Institute of Information Technology
ASTEC, France 2009

Conclusions

- In the paper, the method of the FDTD simulations has been presented.
- Our idea is combining three optimization methods:
  - loop tiling theory,
  - streaming model computations
  - rotating buffers mechanism.
- We can consider a given problem from a programming structures point of view and from an architectural point of view:
  - The loop tiling method defines atomic portions of data spreading among all available computational nodes.
  - The rotating buffers mechanism has been used to create a pipeline of computations and communication, which provides a constant stream of data for computational nodes
- All these methods have been tested on the Cell/BE processor and they have given very promising results.
- Our experiments show that fine grain computations can be efficiently executed the Cell/BE processors using the streaming computational model.

Thank you !!!