Code Comparison Standard For-Loop FDTD vs CUDA FDTD

1. Standard For-Loop Code

```
void updateH2d(Grid *g);
void updateE2d(Grid *g);

int main() {
        Grid *g;
        ALLOC_1D(g, 1, Grid);       // Allocate memory for grid
        gridInit(g);                // Initialize the grid
        ezIncInit(g);               // Initialize source

        for (Time = 0; Time < MaxTime; Time++) {
                updateH2d(g);// Update the magnetic field
                updateE2d(g); // Update the electric field
                Ez(SizeX / 2, SizeY / 2) = ezInc(Time, 0.0);
        }
        return 0;
}

void updateH2d(Grid *g) {
    int mm, nn;
    for (mm = 0; mm < SizeX; mm++) {
        for (nn = 0; nn < SizeY - 1; nn++)
                Hx(mm, nn) = Chxh(mm, nn) * Hx(mm, nn) - Chxe(mm, nn) *
             (Ez(mm, nn + 1) - Ez(mm, nn));  }
    for (mm = 0; mm < SizeX - 1; mm++) {
          for (nn = 0; nn < SizeY; nn++)
              Hy(mm, nn) = Chyh(mm, nn) * Hy(mm, nn) + Chye(mm, nn) *
              (Ez(mm + 1, nn) - Ez(mm, nn)); }
    return;
}

void updateE2d(Grid *g) {
    int mm, nn;
    for (mm = 1; mm < SizeX - 1; mm++) {
          for (nn = 1; nn < SizeY - 1; nn++)
              Ez(mm, nn) = Ceze(mm, nn) * Ez(mm, nn) +
                  Cezh(mm, nn) * ((Hy(mm, nn) - Hy(mm - 1, nn)) -
                  (Hx(mm, nn) - Hx(mm, nn - 1))); }}
    return;
}
```

2. CUDA FDTD Code

```
__global__ void HxHyUpdate_Kernel(Grid *g, int M, int N);
__global__ void EzUpdate2D_Kernel(Grid *g, int DIM);
void updateH2D_CUDA(Grid *g, int M, int N, dim3 BLK, dim3 THD);
void updateE2D_CUDA(Grid *g, int M, int N, dim3 BLK, dim3 THD);

int main() {
    struct Grid *g;
    cudaCalloc1((void**)&g, sizeof(struct Grid), 1);
    int src_x_pos = (int)(0.85 * M);
    int src_y_pos = (int)(N / 2);
    int Tx =32; int Ty = 32;
    int Bx = (M + (Tx - 1))/Tx;
    int By = (N + (Ty - 1))/Ty;
    dim3 BLK(Bx, By, 1);
    dim3 THD(Tx, Ty, 1);

    gridInit(g);
    ezIncInit(g);
    for (int time = 0; time < maxTime; time++) {
        updateH2D_CUDA(g, M, N, BLK, THD);
        updateE2D_CUDA(g, M, N, BLK, THD);
        updatesource_CUDA(g, src_x_pos, src_y_pos, M, time);
        }
        return 0;
}

void updateH2D_CUDA(Grid *g, int M, int N, dim3 BLK, dim3 THD) {
    HxHyUpdate_Kernel << <BLK, THD >> >(g, M, N);
}

void updateE2D_CUDA(Grid *g, int M, int N, dim3 BLK, dim3 THD) {
    EzUpdate2D_Kernel << <BLK, THD >> >(g, M);
}

__global__ void HxHyUpdate_Kernel(Grid *g, int M, int N) {
    __shared__ double Che;
    int size_Hx = M * (N - 1);
    int size_Hy = (M - 1) * N;
    // Map from threadIdx/blockIdx to cell position
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    int offset = row * blockDim.x *gridDim.x + col;

    if (threadIdx.x == 0)        Che = 0.0018769575507639;
    __syncthreads();
```

```
    int top = offset + blockDim.x * gridDim.x;
    int right = offset + 1;

    // Calculate Hx
    if ((row == M - 1))top -= M;
    if (offset < size_Hx)
        g->hx[offset] = 1.0 * g->hx[offset] - Che * (g->ez[top] - g->ez[offset]);
    else g->hx[offset] = 0.0;

    // Calculate Hy
    if ((col == M - 1) || (col == M - 2))         right--;
    if (offset < size_Hy)
        g->hy[offset] = 1.0 * g->hy[offset] + Che * (g->ez[right] - g->ez[offset]);
    else g->hy[offset] = 0.0;
}

__global__ void EzUpdate2D_Kernel(Grid *g, int DIM) {
    __shared__ double Ceze, Cezh;
    // Map from threadIdx/blockIdx to cell position
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    int offset = row * blockDim.x *gridDim.x + col;
    int total = DIM*DIM;
    int left = offset - 1;
    int right = offset + 1;
    int top = offset + blockDim.x * gridDim.x;
    int bottom = offset - blockDim.x * gridDim.x;

    if (threadIdx.x == 0) {
        Ceze = 1.0;
        Cezh = 266.3885498084424;
        }
    __syncthreads();

    if (col == 0)                 left++;
    if (col == DIM - 1)           right--;
    if ((row == DIM - 1))         top -= DIM;
    if (row == 0)                 bottom += DIM;

    if ((col == 0) || (col == (M - 1)) || (row == 0) || (row == (N - 1)))
        g->ez[offset] = 0.0;
    else {
        if (offset < total)
            g->ez[offset] = Ceze * g->ez[offset] +
                Cezh * ((g->hy[offset] - g->hy[left]) -
                (g->hx[offset] - g->hx[bottom]));
        else     g->ez[offset] = 0.0;
        }
}
```