multiple of 16 as illustrated in Fig 2. The modified size of the FDTD domain becomes $Nxx \times Nyy \times Nz$, where Nxx, Nyy, and Nz are number of cells in *x*, *y*, and *z* directions, respectively.

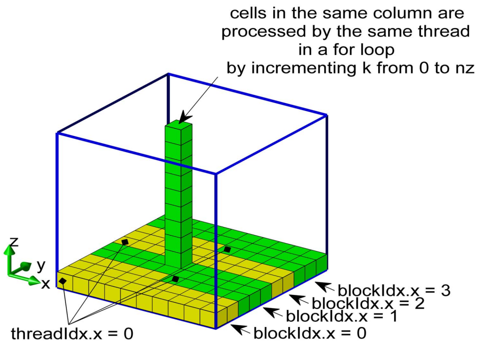

Figure 1. Mapping of threads to cells of an FDTD domain.

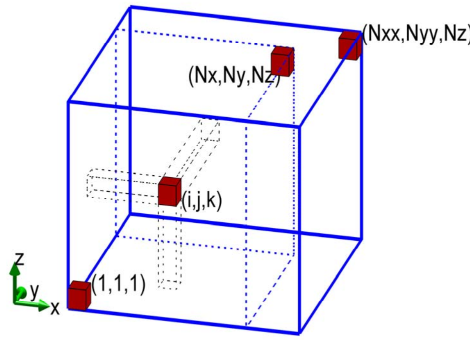

Figure 2. An FDTD problem space padded with additional cells.

```
__global__ void update_magnetic_fields_on_kernel(...)
{
extern __shared__ float sEz[];
int ci = blockIdx.x * blockDim.x + threadIdx.x;
int j  = ci/nxx;
int i  = ci - j*nxx;
int si = threadIdx.x;
int cizp;
float ex, exzp, ey, eyzp;
ey = Ey[ci];
ex = Ex[ci];
for (int k=0;k<nz;k++)
{
       cizp   = ci + nxx*nyy;
       exzp   = Ex[cizp];
       eyzp   = Ey[cizp];
       sEz[si]= Ez[ci];
       if (threadIdx.x<16)
              Ez[blockDim.x+threadIdx.x]  = Ez[ci+blockDim.x];
       __syncthreads();

       Hx[ci]  = Chxh[ci]*Hx[ci]+ Chxey[ci]*(eyzp-ey)
                              + Chxez[ci]*(Ez[ci+nxx]-sEz[si]);
       Hy[ci]  = Chyh[ci]*Hy[ci]+ Chyez[ci]  *  (sEz[si+1]-sEz[si])
                              + Chyex[ci]  *  (exzp-ex);
       ...
       ci = cizp;
       ey = eyzp;
       ex = exzp;
       }
}
```

Listing 2. A section of CUDA code to update magnetic field components.

After ensuring the coalesced memory access, data reuse in a *for* loop in *z* direction and appropriate use of shared memory a CUDA code is developed. A section of this code is shown in Listing 2. The developed algorithm is tested on an