Microdisk Cavity FDTD Simulation on FPGA using OpenCL

Tobias Kenter, Christian Plessl

Paderborn Center for Parallel Computing and Department of Computer Science
Paderborn University
Microdisk Cavity

• Microdisk cavity in perfect metallic environment
  – Well studied nanophotonic device
  – Point-like time-dependent source (optical dipole)
  – Known analytic solution (whispering gallery modes)

Simulations can help to investigate other nanophotonic setups

---

Experimental setup: microdisk cavity

Result: energy density
• **Physics:** Maxwell's partial differential equations
  – Electric field $E$
  – Magnetic field $H$
  – Material constants (electric permittivity $\varepsilon$, magnetic permeability $\mu$)

\[
\frac{\partial H}{\partial t} = -\frac{1}{\mu} \nabla \times E - \frac{1}{\mu} (M_{\text{source}} + \sigma^* H)
\]

\[
\frac{\partial E}{\partial t} = \frac{1}{\varepsilon} \nabla \times H - \frac{1}{\varepsilon} (J_{\text{source}} + \sigma E)
\]

• **Simulation:** FDTD stencils
  – Stencil for dielectric material in 2D

```plaintext
updateE(*ex, *ey, *hz) {
    ex[x,y] = ca * ex[x,y] + cb * (hz[x,y] - hz[x,y-1]);
    ey[x,y] = ca * ey[x,y] + cb * (hz[x-1,y] - hz[x,y]);
}
```

```plaintext
updateH(*ex, *ey, *hz) {
    hz[x,y] = da * hz[x,y] + db * (ex[x,y+1] - ex[x,y] + ey[x,y] - ey[x+1,y]);
}
```
**FPGA Pipeline for FDTD**

- Inside time step
  - Regular + parallel update operations
  - Can form customized loop pipeline on FPGA
  - Locality + predictable memory access
  - Can prefetch and stream data

- E and H are must be updated alternately (leap-frog)
  - Reusing local results is key to performance
  - Unrolling several time steps increases computational intensity

**Diagram:**
- updateE → MEM → updateH
- updateH → MEM → updateE

  2-fold unrolled, overlap processing for 2 iterations
OpenCL for FPGAs

• OpenCL
  – Covers parallelism and awareness of memory locations
  – Base of familiar developers (mostly GPU)
  – Suitable to generate competitive FDTD design on FPGA?

• OpenCL-based SDAccel tool flow
  – OpenCL source-to-source transformation
  – Vivado HLS step
  – Vivado synthesis place + route
  – SDAccel Version 2016.1

• Target system
  – ADM-PCIE-7V3 board with Xilinx Virtex-7 XC7VX690T + 2x 8GB DDR3 memory
Design Steps

1. Wrap main loop into OpenCL kernel
   – First FPGA design up and running after few hours
   – ~1000x slower than CPU

2. Generate FPGA pipeline for E and H updates
   – Burst transfers to local memory
   – Compute from local memory
   – Pipeline main loop with low initiation interval

3. On the way…
   – Separate compute + transfer kernels, coupled through pipes
   – Code transformations in compute kernel

4. Unroll as many time steps as resources permit
   – Allow data reuse
   – Instantiate many individual buffers
OpenCL-based FPGA Design

Global Memory (DDR3 on ADM-PCIE-7V3 board)

- Burst transfers

Compute Kernel

- Read E_x
- Local Memory (BRAM)
- E_y
- H_z
- ... more Pipes

- Pipe

Stage 1

- Local Memory
- ... more ... more

Stage 2

- Local Memory
- ... more ... more

... more

Stage 36

- Pipe

- Write E_x
- E_y
- H_z
- ... more Pipes

- Burst transfers
• 36 pipeline stages, initiation interval 2
• 140MHz (down from original target 200MHz)
• Resulting design with OpenCL is very competitive
• Code is adapted to FPGA target and current tool capabilities
  – Much lengthy boilerplate may go away with maturing tools and better understanding of them
  – Performance portability not explored (currently design with single work-item)
Thank you!