FPGA-Based High-Order Finite Difference Method for Linear Wave Modelling Problems

Chuan He, Guan Qin, Wei Zhao

Institute for Scientific Computation

Texas A&M University

http://research.cs.tamu.edu/realtime/

LACSI

FD-based Wave Equations Modeling on RC Platform

Content:

- linear wave equations
- Conventional FD-based wave modeling problems
- FPGA-based FD computing engine
- Sliding window buffering structure
- Simulation results

Linear wave equations in isotropic and inhomogeneous media

- Hyperbolic Partial Differential Equations
- Linear: the amplitude of two interacting waves is simply their summations.
- Isotropic: parameters of media are independent of directions
- Inhomogeneous: parameters are dependent of the position.
- Two kinds of problems: Forwarding and Reverse
- Transient wave phenomenon: Transmission, Reflection, diffraction, Dispersion, Dissipation, etc.
- Application Fields: Aero- or marine-acoustics, Electromagnetics, Geophysics,, etc.

Linear wave equations in first/second derivative forms

- **Acoustic wave equations:**

$$\frac{1}{\kappa(x,y,z)}\frac{\partial^2 P}{\partial t^2} = \nabla\cdot\left(\frac{1}{\rho(x,y,z)}\nabla P\right) + \vec{f}$$

$$\frac{1}{\kappa(x,y,z)}\frac{\partial P}{\partial t} = \nabla\cdot\vec{v} + \vec{F} \qquad \rho(x,y,z)\frac{\partial \vec{v}}{\partial t} = \nabla P$$

- **Maxwell's equations :**

$$\mu\varepsilon\frac{\partial^2 \vec{E}}{\partial t^2} + \nabla\times\nabla\times\vec{E} = 0$$

$$\nabla\times\vec{E} = -\mu(x,y,z)\frac{\partial \vec{H}}{\partial t} - \vec{M} \qquad \nabla\times\vec{H} = \varepsilon(x,y,z)\frac{\partial \vec{E}}{\partial t} + \vec{J}$$

- **Elastic wave equations :**

$$\rho\frac{\partial^2 \vec{u}}{\partial t^2} = (\lambda+\mu)\nabla(\nabla\cdot\vec{u}) - \mu\nabla\times\nabla\times\vec{u}$$

$$\begin{cases} \rho(x,y,z)\partial_t v_x = \partial_x\sigma_{xx} + \partial_y\sigma_{xy} + \partial_z\sigma_{xz} \\ \partial\sigma_{xx} = \lambda(x,y,z)(\partial_x v_x + \partial_y v_y + \partial_z v_z) + 2\mu(x,y,z)\partial_x v_x \\ \partial_t\sigma_{xy} = \mu(x,y,z)(\partial_x v_y + \partial_y v_x) \end{cases}$$

LACSI

The standard second-order FD scheme for 2D acoustic wave equations: A case study

- Scalar acoustic wave equation:

$$\frac{1}{v^2(x,z)}\frac{\partial^2 P}{\partial t^2} = \Delta P + f = \frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial z^2} + f(x,z,t)$$

- The standard (2, 2) FD scheme:

$$P_{i,k}^{n+1} = 2 \cdot P_{i,k}^{n} - P_{i,k}^{n-1} + (dt)^2 \cdot v_{i,k}^2 \cdot \Delta^{(2)} P_{i,k}^{n} + f_{i,k}^{n}$$

where $$\Delta^{(2)} P_{i,k}^{n} = \left(\frac{P_{i+1,k}^{n} - 2 \cdot P_{i,k}^{n} + P_{i-1,k}^{n}}{(dx)^2} \right) + \left(\frac{P_{i,k+1}^{n} - 2 \cdot P_{i,k}^{n} + P_{i,k-1}^{n}}{(dz)^2} \right)$$

- Seven pressure and velocity values are needed to update pressure fields at one grid point for one time-marching step.
- Eleven Floating-point computations. (2X operations are not counted in.)
- Computationally demanding and data-intensive.

LACSI

Numerical errors of FD approximations

- Approximation errors arise from both temporal and spatial discritizations.
- Numerical dispersion, dissipation and anisotropy, etc.
- Numerical errors cause wave components propagating in slower speeds, damped amplitudes, or wrong directions in simulations than in the reality.
- Accumulate quickly and destroy the initial shape of wavelets.
- Decreasing sampling interval alleviates this problem. But leads to more grid points.
- To simulate a wavelet in an area of 10 X its wavelength, (2, 2) FD scheme requires at least 200-point along each spatial axis for realistic simulation problems.
- An effective method is to adopt high-order FD schemes.

LACSI

High-order FD approximations in space

- **Approximate $\frac{\partial^2 P}{\partial x^2}$ to (2m) accurate order**

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_{i,k}^{(2m)} = \frac{\alpha_0^m \cdot P_{i,k} + \sum_{r=1}^{m} \alpha_r^m \cdot (P_{i+r,k} + P_{i-r,k})}{(dx)^2} + O\left((dx)^{2m}\right)$$

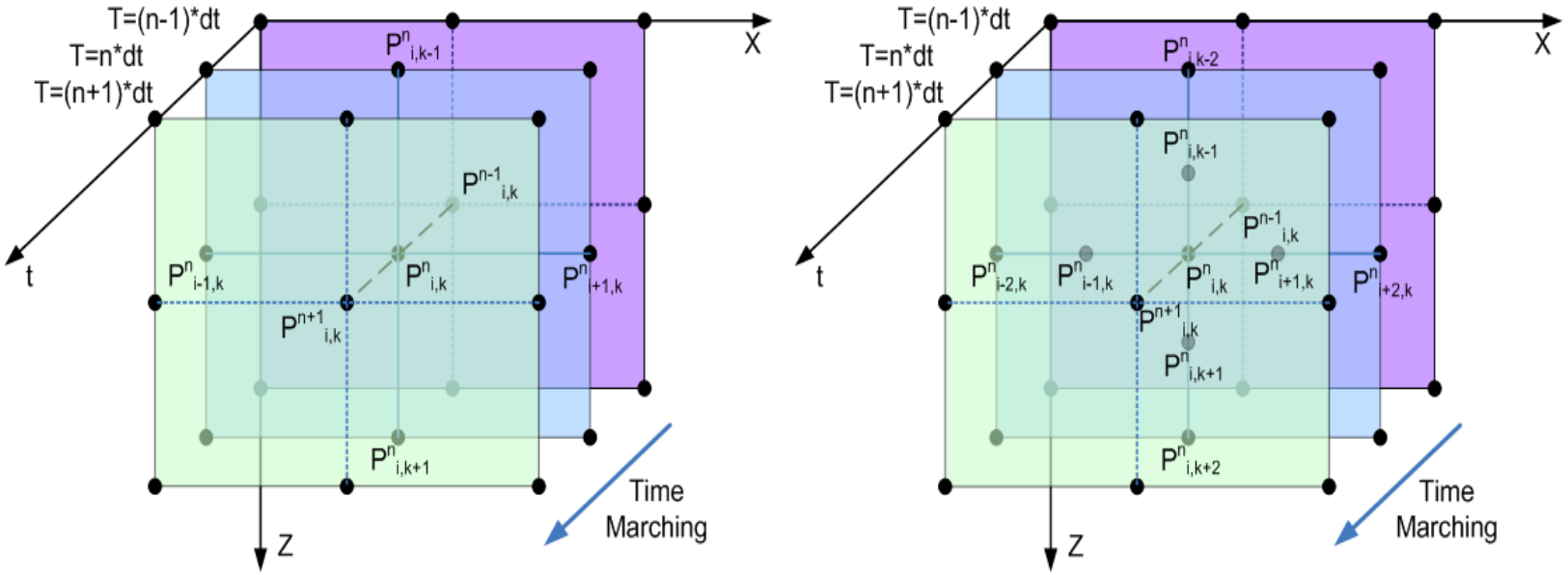
where

$$\alpha_0^m = -2 \cdot \sum_{r=1}^{m} (-1)^{r-1} \frac{2m!^2}{r!(m-r)!(m+r)!} \qquad \alpha_r^m = (-1)^{r-1} \frac{2m!^2}{r!(m-r)!(m+r)!}$$

- **More operands and more computations**
- **Much smaller numerical errors.**
- **Simulate Large area with moderate spatial resolution.**
- **Enlarge spatial sampling interval to reduce the total number of discrete points**

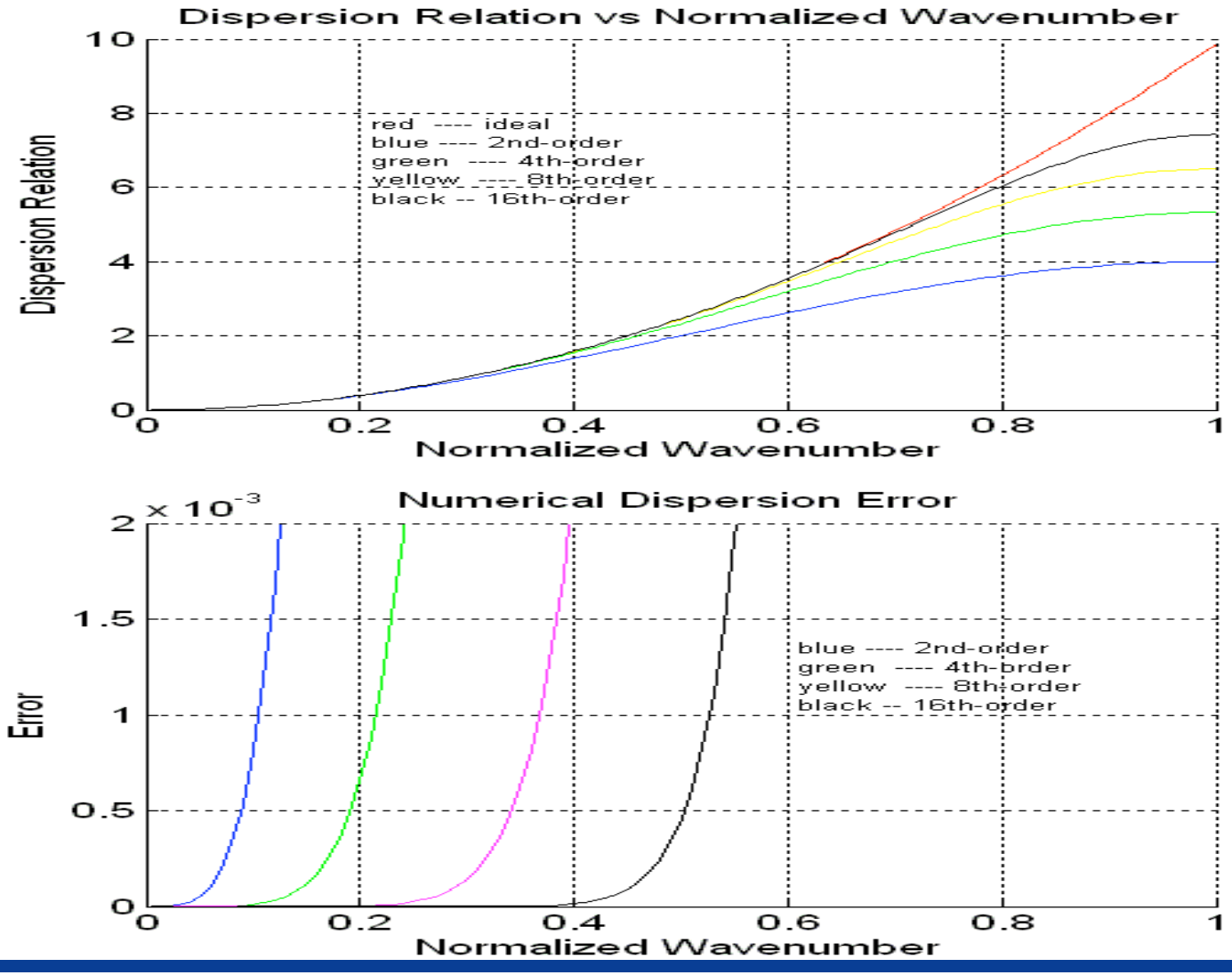
LACSI

Stencils of high-order FD schemes in 2D space



FD Schemes	(2, 2)	(2, 4)	(2, 8)	(2, 16)
Total FP Operations	10	20	32	56
Total Operands	7	11	19	35

Dispersion relation of different FD schemes



LACSI

Shortages of High-order FD approximations

- More computations per grid.
- More operands involved.
- Hard to construct effective high-order ABC or PML.
- Not suitable for abrupt discontinuous media.
- People tend to be conservative in enlarging sampling interval

Results:

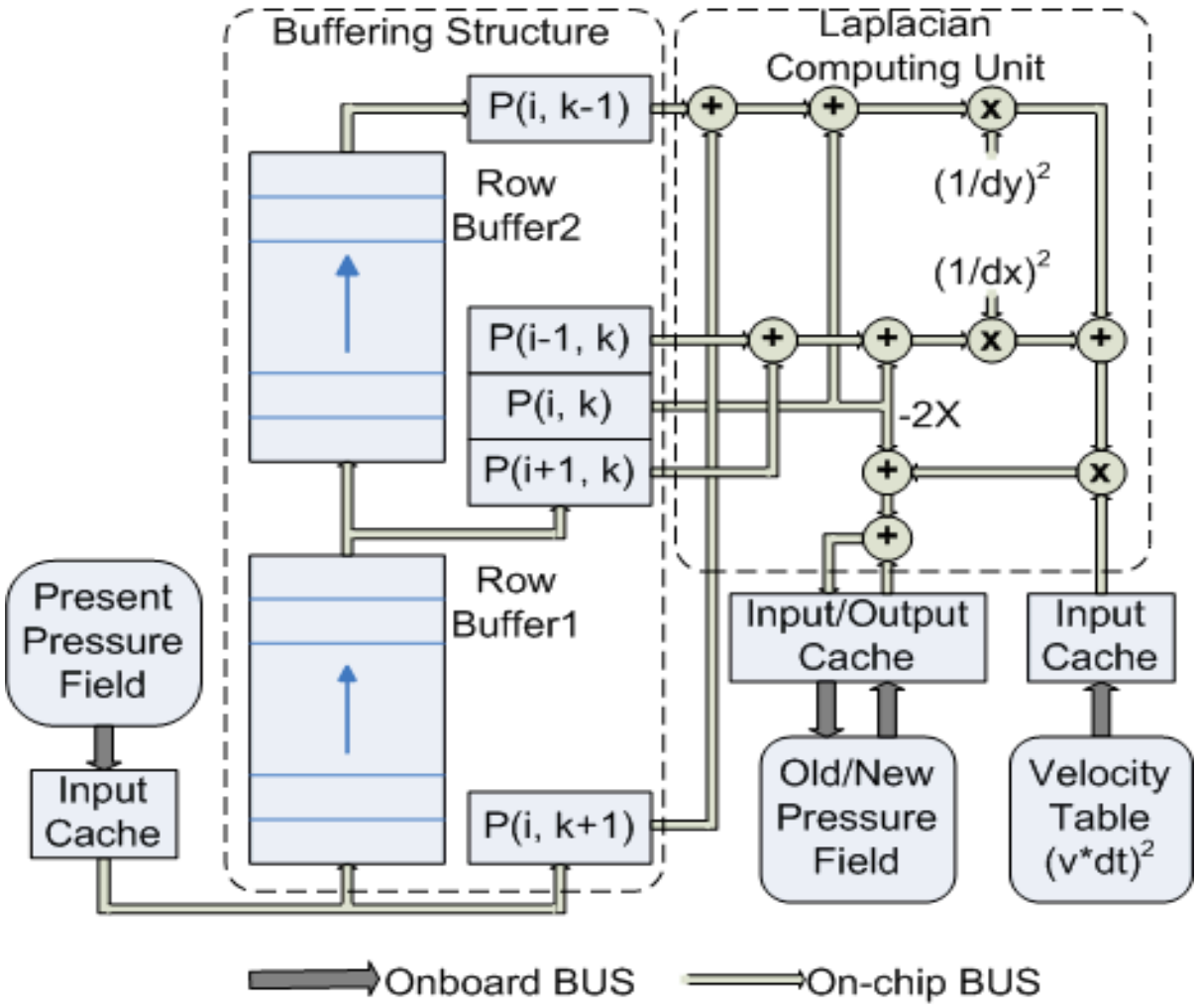
- The decrement of spatial points achieved by high-order FD approximations seems not enough to compensate extra computations they introduced.
- Are commonly slower than low-order schemes and not widely used in reality.

LACSI

High-order FD schemes on RC platform

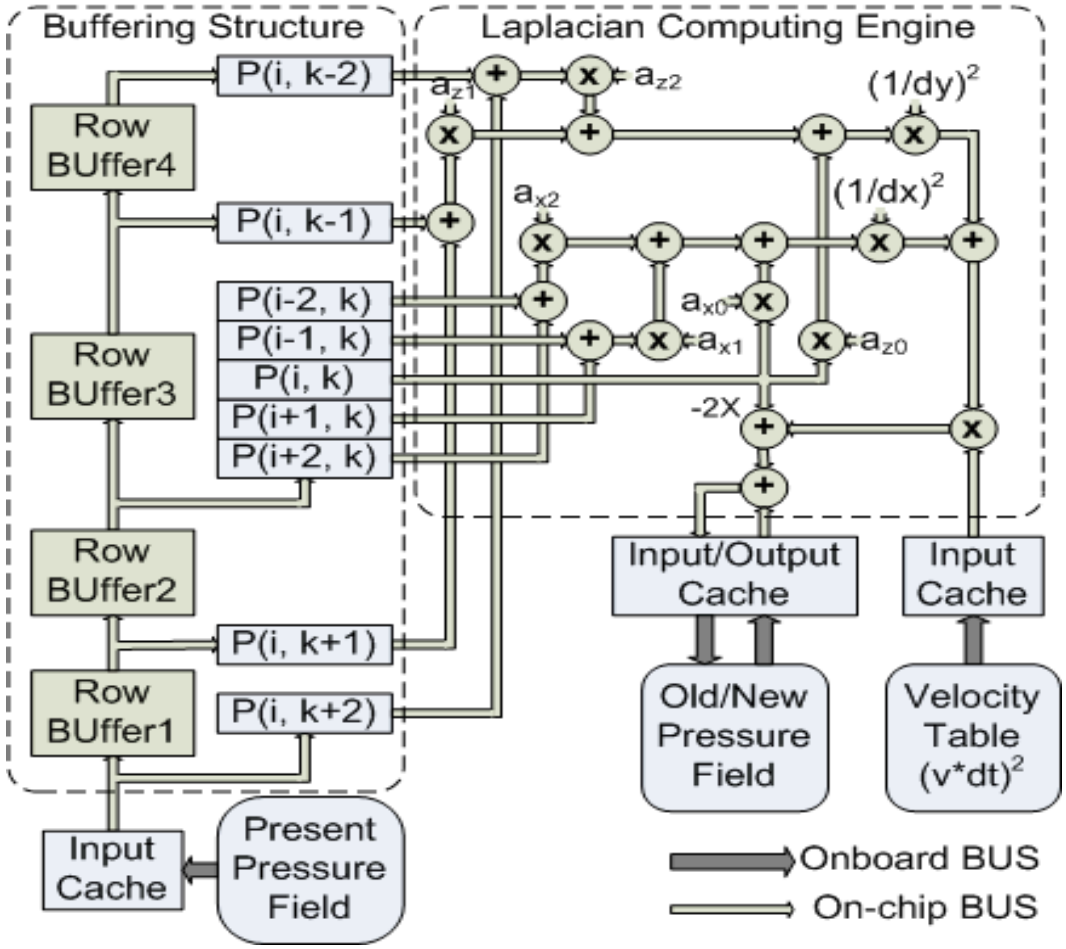
- Customized computing engine with tens to hundreds of FP arithmetic units
- Multiple independent external memory controllers provide wider external memory bandwidth.
- Abundant internal RAM blocks working as data cache or buffer
- Programmable internal routing paths Leads to better caching behavior and data reusability.
- Fine-grain parallelism makes it an effective complement to prevailing PC-cluster systems

Function blocks and data flow for (2, 2) FD scheme

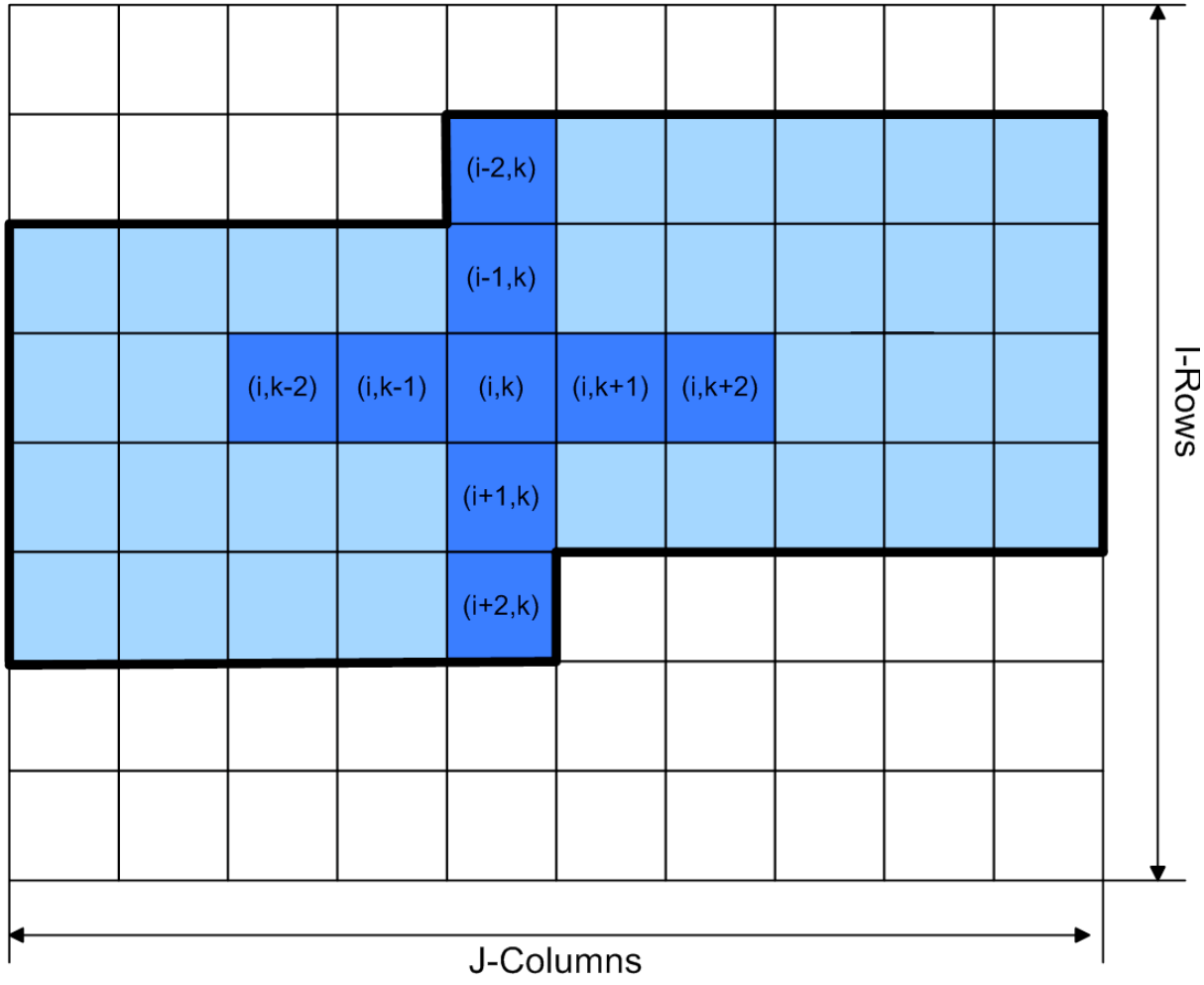


LACSI

Function blocks and data flow for (2, 4) FD scheme



Sliding window for (2, 4) FD scheme in 2D space



LACSI

Comparison of FP operations and memory accesses for different FD schemes

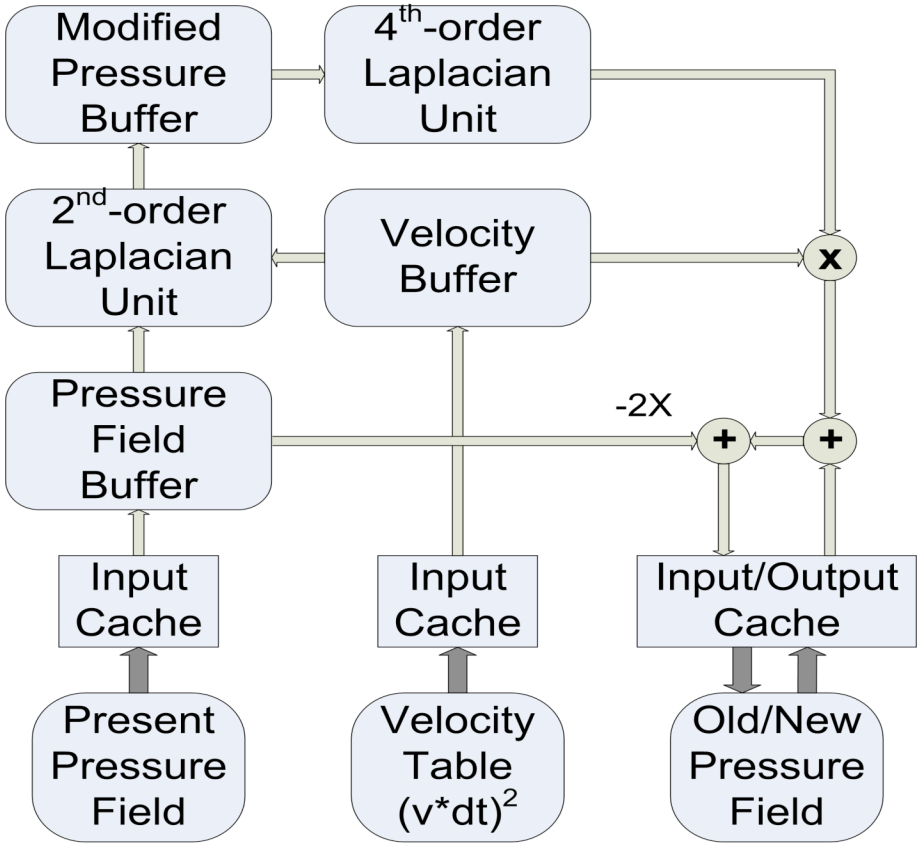
FD Schemes	(2, 2)	(2, 4)	(2, 8)	(2, 16)
Total FP Operations	10	20	32	56
Total Operands	7	11	19	35
Total external memory accesses	4	4	4	4
FP operation to memory access ratio	2.5	5	8	14

* The Theoretical FP performance to memory bandwidth ratio for Intel P4 3.0GHz Processor is 3.75 (6G flops / 800M Words * 2 channels)

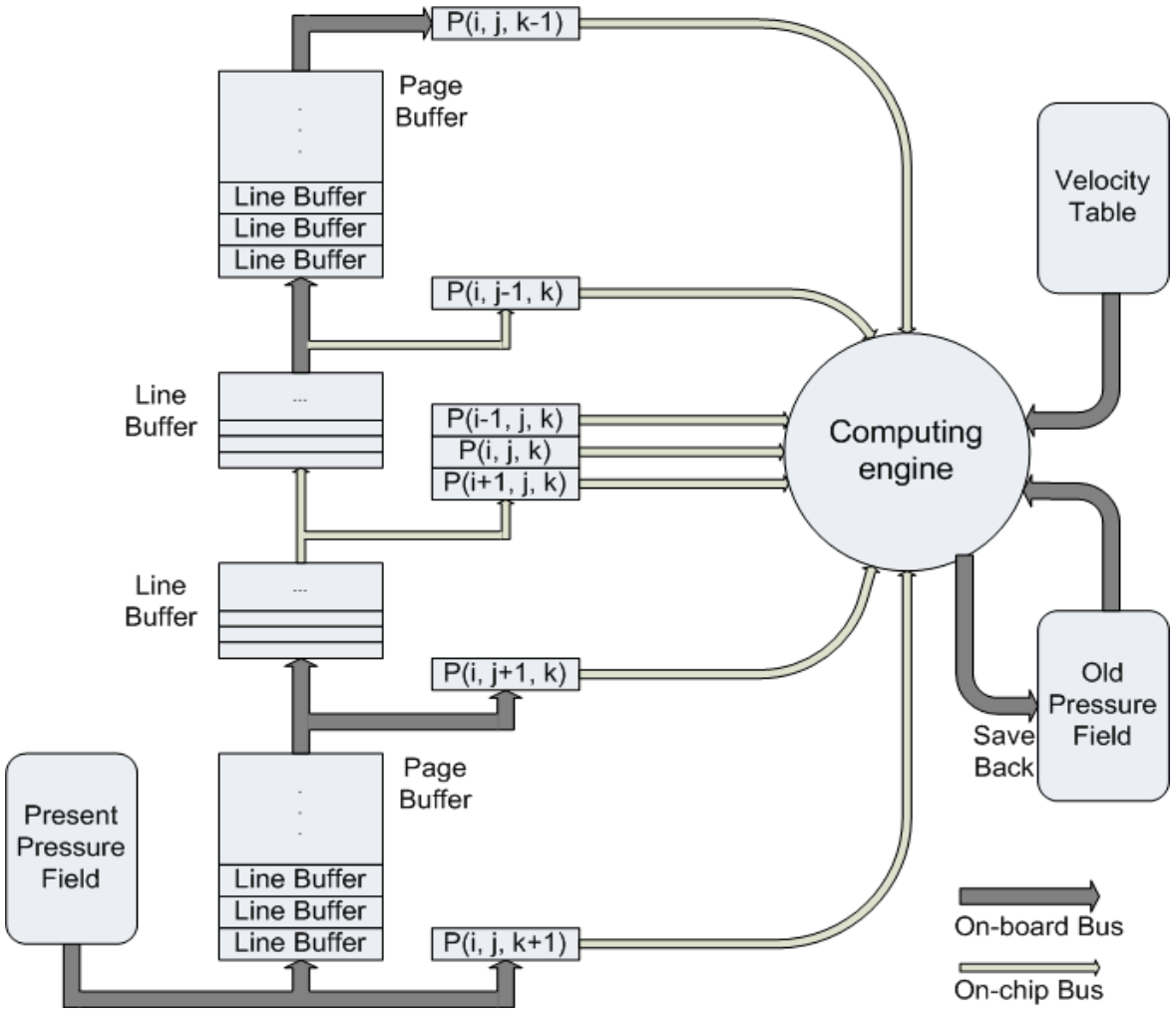
LACSI

Extension to high-order in time: (4, 4) FD scheme

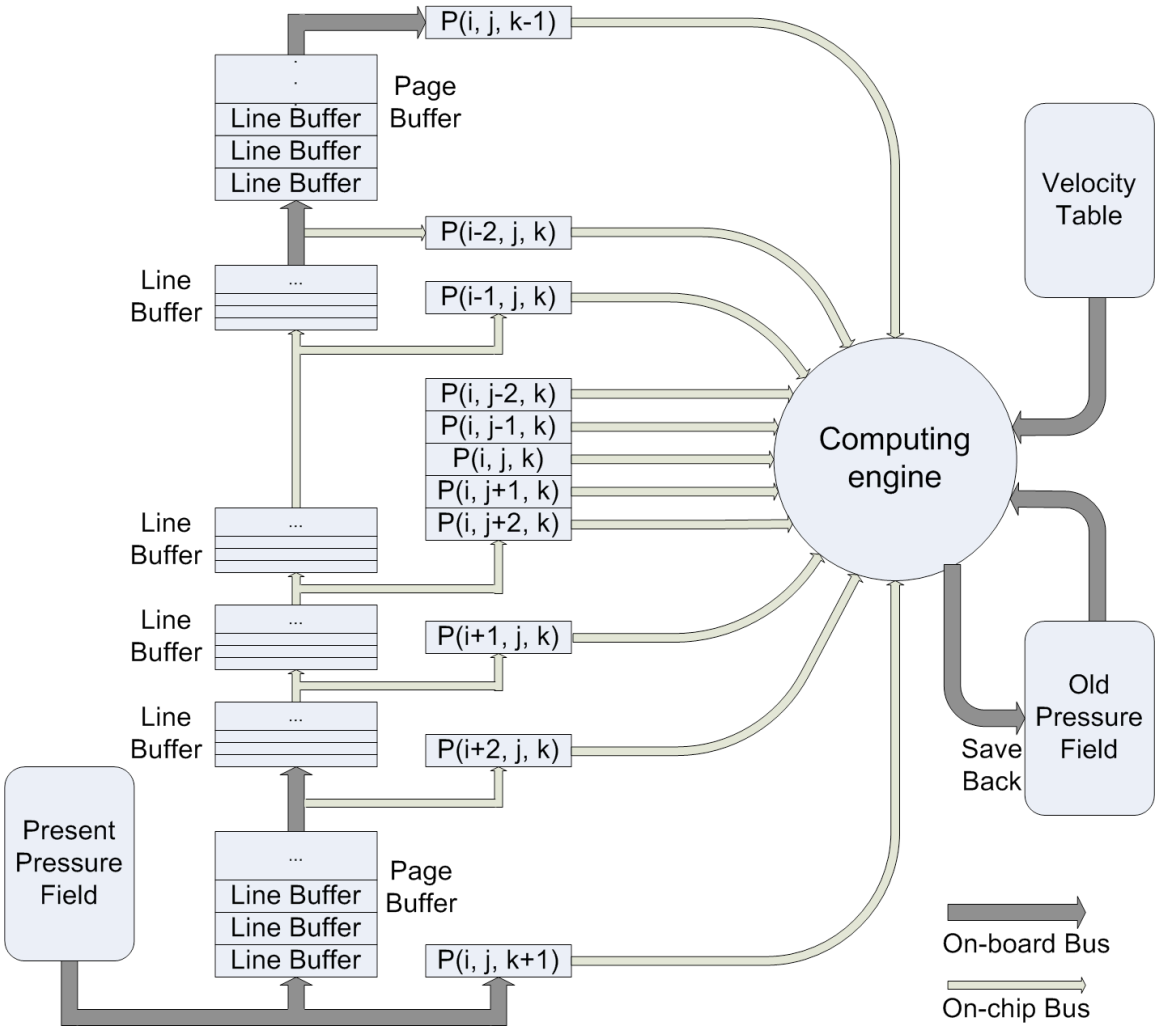
$$\frac{P^{n+1} - 2P^n + P^{n-1}}{(dt)^2} = v^2 \Delta^{(4)} \left(P^n + \frac{(dt)^2}{12} \cdot \left(v^2 \Delta^{(2)} P^n \right) \right)$$



Extend (2, 2) FD scheme to 3D space



Extension to 3D (2, 4-4-2) FD scheme



LACSI

Highlights of the hardware implementation

- High-order FD schemes:
  - Increase the number of FP computations per grid
  - Reduce numerical dispersions
  - Much more accurate simulation
  - Reduce the total number of girds
- Pipelined FP computing engine
  - Sustained computational speed
  - One output at every clock cycle
- Sliding window based data buffering subsystem
  - Efficient cascaded FIFO structure
  - Provide a new set of operands to the computing engine at every clock cycle
  - Input/output cache to hide the overhead of SDRAM

LACSI

The basic idea of our approach

- Keep the number of external memory accesses unchanged using sliding window buffering structure.
- Saturate external memory bandwidth by adjust the clock rate applied to the computing engine.
- Adjust the order of FD approximations to take full advantage of FPGA's computational potential.
- Adopt different FD schemes (high-order in space, high-order in time, hybrid schemes…) for different RC platforms

Results:

- A balancing point can always be reached where the utilization of onboard FPGA resources and external memory bandwidth are all maximized.
- This preferable scalability makes our design accommodated to most commercial RC platforms

Simulation results - Platform

- Entry-level Xilinx ML401 Vertex-4 evaluation board
  - One XC4LX25 integrating 24,192 Logic Cells, 48 DSP Slices, 72 18Kb SRAM Blocks.
  - 64MByte onboard DDR-SDRAM modules with 32-bit interface connecting to FPGA.
  - 9Mb onboard ZBT-SRAM with 32-bit interface.
- Software development environment
  - Xilinx ISE 6.3i
  - ModelSim 6.0se
- Referential platform
  - P4 3.0 GHz Dell workstation with dual-channel 1GB memory
  - INTEL C++ compiler v8.1 for Linux

Wave propagation test in 2D media with constant velocity

- Number of spatial grids: 1000×1000

  Storage requirements: 4 M words

  Total time-march steps; 6000

  Total number of grid computations: 6×10^9

- Keep the number of 2D grid points unchanged to evaluate the speedup attributed purely to our hardware implementations.
- The clock rate applied to onboard DDR-SDRAM modules is 100MHz, to the computing engine is 50 MHz.
- The referential software codes are programmed by ANSI C and compiled using Intel C++ v8.1 on Linux OS.
- Compiler is optimized for speed. (-O3 -tpp7 and -xK) and 20% of the CPU's peak performance are reached.

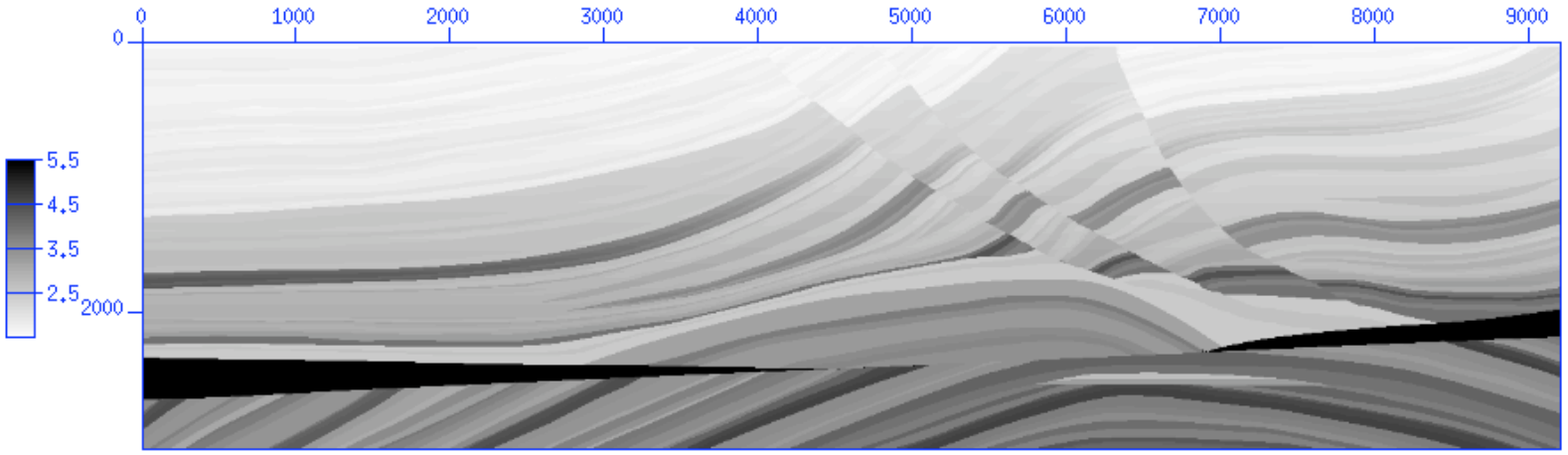
LACSI

Speed Comparison between PC and RC

FD schemes	Software Computational Throughput (Million Grid / second)	Hardware Implementations		
		Computational Throughput (Grid/second)	Speed-up	Resource Utilizations (RAM Blocks /DSP Slices /Logic Slices)
(2, 2)	33.27	49.71	1.49	16/10/4885
(2, 4)	27.53	49.63	1.80	20/22/7212
(2, 8)	19.90	49.50	2.49	28/30/9732
(4, 4)	15.10	48.38	3.20	28/30/9818

Performance for realistic problem: Marmousi model

- A prevailing 2D model with complex underground structure synthesized in 1990s.
- A benchmark in seismic data processing industry for calibrating imaging algorithms.
- 2D grids cover dimensions of 9200m by 3000m with 4m interval (2300 X 750 grids)



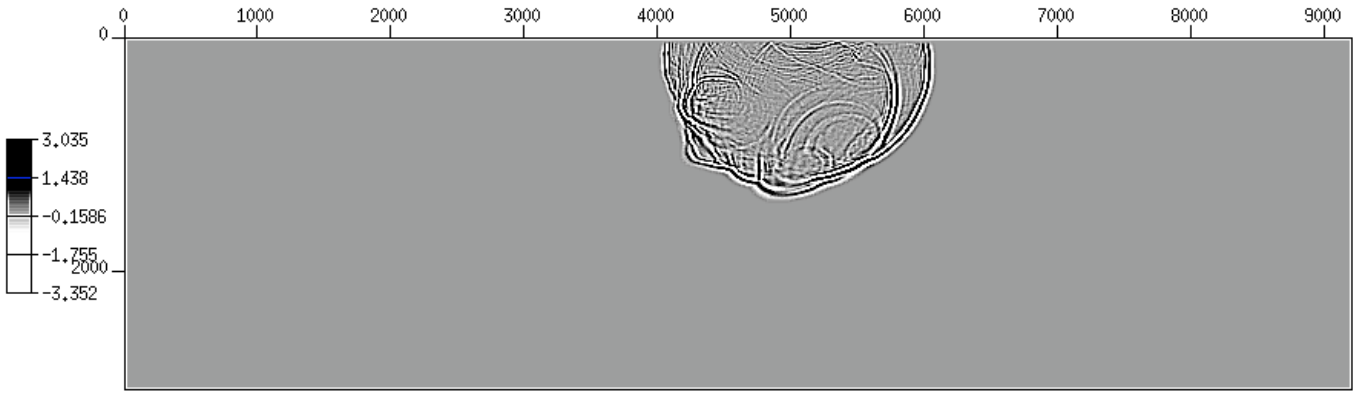
velocity profile

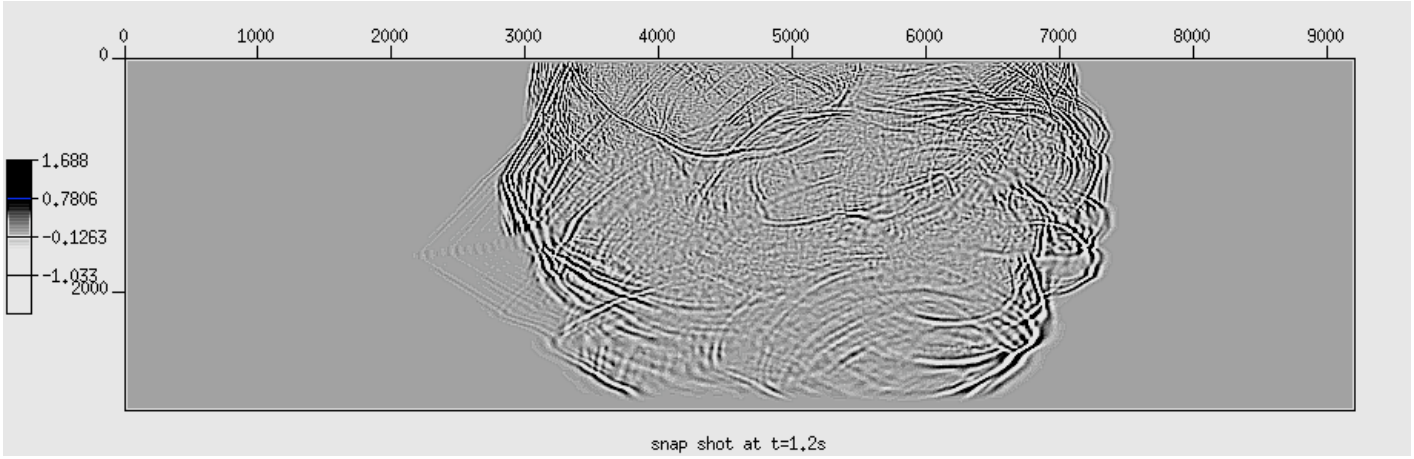
LACSI

Parameters for Marmousi modeling

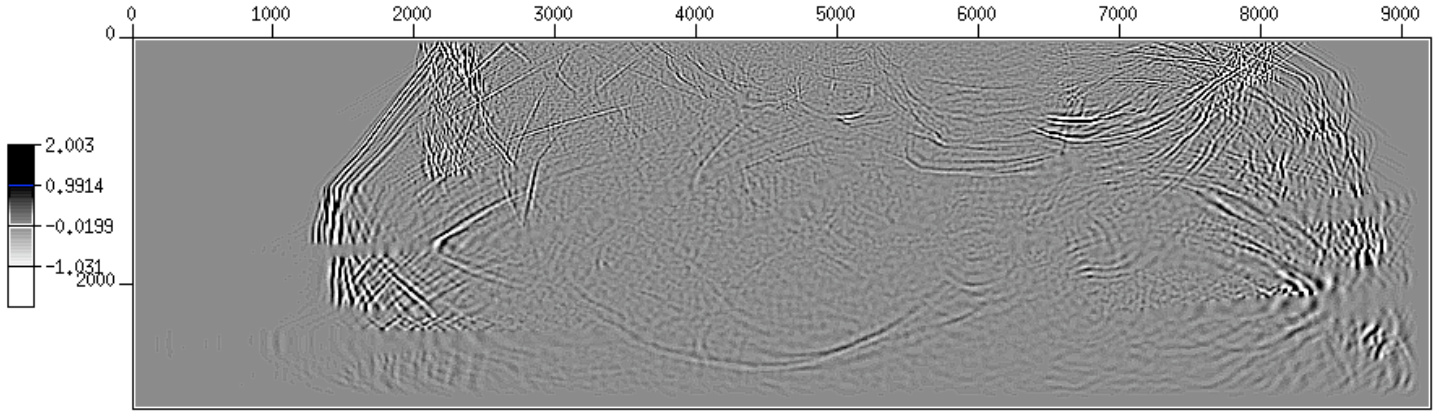
- Free surface boundary condition on the top.
- Damping boundary conditions are applied to 50 absorbing layers on the left, right and bottom.
- Excitation at x=5000m, z=8m. (ricker wavelet)
- Maximum frequency in source wavelet is set to 80Hz.
- (2, 8) FD computing engine.
- Two extra FP multiplier for boundary damping.
- One extra FP adder together with internal SRAM blocks to introduce source wavelet.

Snapshots of Marmousi model produced by RC platform

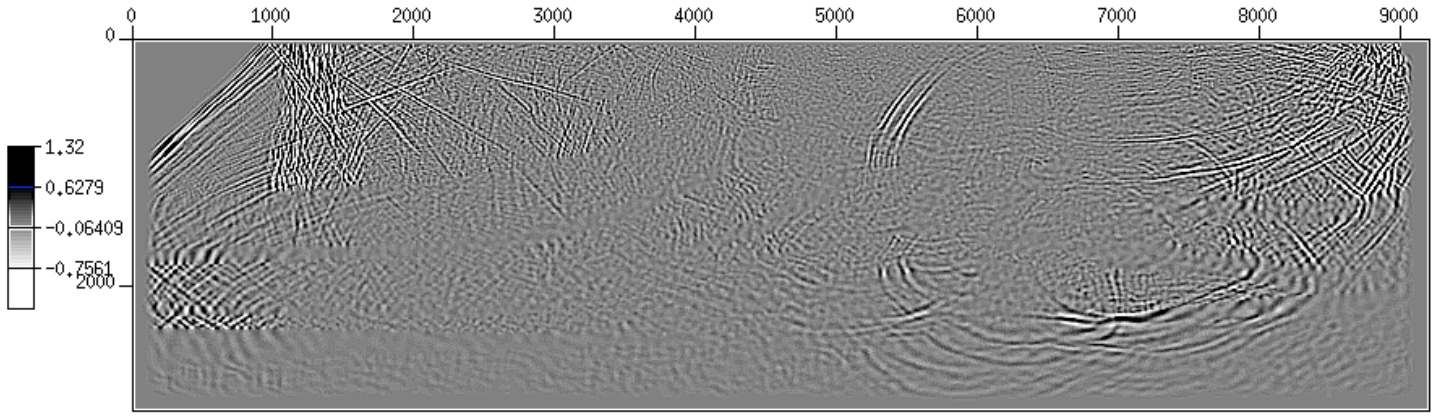




Snapshots of Marmousi model produced by RC platform



snap shot at t=1.8s



snap shot at t=2.4s

LACSI

Common pitfalls people might make in performance comparison between PC- and RC-based solutions

- Software algorithms are commonly migrated to RC platforms with limited modifications.
- FPGA resources and memories are always abundant on application-specified RC platforms.
- Onboard Hardware architecture and interconnection pattern are well-tailored to maximize the computational performance for particular applications.
- External memory bandwidth never imposes a performance bottleneck.
- RC implementations typically choose small but fast onboard SRAM modules or even internal RAM blocks as working space.
- Performance comparisons are made between one PC workstation and a complex RC platform with multiple FPGA chips.

- Naïve software implementations on PC platform are used as references without careful performance tuning