Interactive Ultrasonic Field Simulation For Non Destructive Testing

Jason Lambert[a][b], Gilles Rougeron[a], Sylvain Chatillon[a] and Lionel Lacassagne[b]

[a]CEA LIST, CEA Saclay Digiteo Labs, PC120, 91191 Gif-sur-Yvette CEDEX, France;
[b]Laboratoire de Recherche en Informatique, Univ. Paris-Sud, F-91405 Orsay, France

ABSTRACT

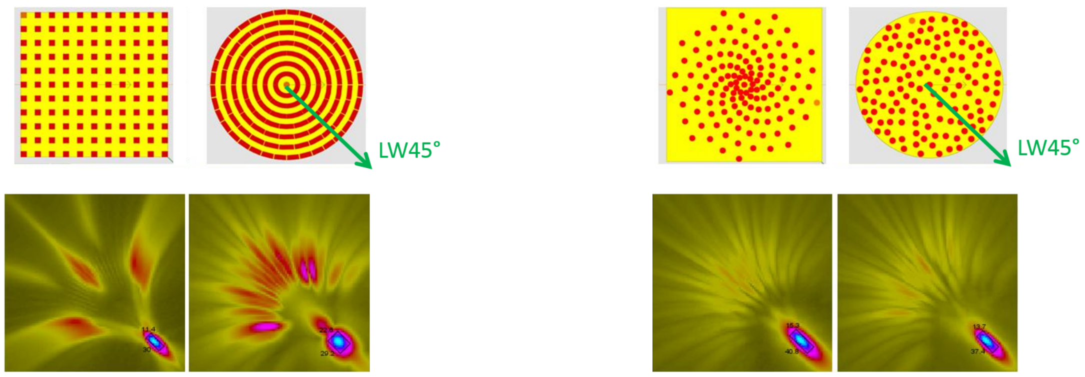
This paper presents an ultrasonic field simulation software, dedicated to Non Destructive Testing, aiming at interactivity. This work relies on Civa Software semi-analytical model. By restricting its scope to homogeneous isotropic specimens with simple geometry and half-skip modes, an almost completely regular algorithm, well suited to modern hardware, can be derived. The perfomance of three implementations on multicore SIMD general purpose processors (GPP), manycore accelerators (MIC) and graphical processing units (GPU) over a set of 18 realistic configurations (a standard one plus 17 variations) are presented and analysed. For GPP and the GPU, interactive performances with almost 30 fps have been reached on the standard configuration. This is, to our knowledge, the very fist time for a NDT ultrasonic field simulation software.

Keywords: Ultrasonic Inspection Simulation, Performance Optimisation, Multicore SIMD processor, GPU

1. INTRODUCTION

Inspection simulation is used in many non-destructive testing (NDT) applications: from designing new inspection methods and probes to qualifying methods and demonstrating capabilities through virtual testing while developing inspection methods. Thus, simulations are carried out to determine the characteristics of the ultrasonic beam radiated in the specimen, such as the focal spot or the local direction. A common application is to help minimizing grating lobes by selecting the right probe[1], as illustrated in figure 1. It shows the simulated ultrasonic field focused at 100-mm deep with a deviation angle of 45 degrees in both the saggital and transverse plane in a ferritic steel plate for different phased array probes with varying distributions of elements. With a regular one as shown by figure 1a grating lobes are important. Whereas with distributions lacking periodicity (fig. 1b), the amplitude of grating lobes is much smaller. In addition, they are more evenly distributed throughout space and do not show strong energy for some directions.

The design of a probe and control configuration may result in a tedious and relatively long phase, often through trial and error. The aim is to speed up this process. By running at interactive pace, a simulation tool



(a) Matrix and annular phased array probe (b) Spiral and random distributed array probes

Figure 1: 2D array designs (top row) and the associated beam field calculations at 45°in both the saggital and transverse planes (bottom) at 100-mm deep in a plate made of ferritic steel[1]

showing really quickly the impact of a change of any parameters value would bring to NDT users both a better mastery and a more intuitive comprehension of the complex physical phenomenons involved in a control.

Nowadays, a major development axis is to develop adapted algorithms to benefit from the parallelism available on modern hardware (multicore general purpose processors, graphic processing units). In the NDT field, multiples works addressed the speed up of analysis algorithms by relying on those architectures. Real time reconstruction have been achieved on GPU for example through the *Synthetic Aperture Focalisation Technique* (SAFT)[2] and Total Focusing Method (TFM)[3,4,5]. On the simulation side, works to speed up computation time have been realized mainly on finite elements and finite difference models, some dedicated to a specific implementation[6,7,8] others taking the form of an open source framework to develop scientific simulation like POGO.[9] However, even with the use of GPU, according to the performances reported by their authors none of these simulations presented fast enough computations to reach interactive pace.

CIVA software, developed by CEA-LIST, implements semi-analytical models based on asymptotic developments. It is a multi-technics platform used to both analyse acquisitions and run simulations. It is validated against international NDT benchmarks[10]. These models are generic and faster than finite elements and finite differences models. However, for configurations with complex geometry or materials, they take seconds to minutes to run a simulation. Significant ongoing efforts are made to reduce computation times by working both on models and computational aspects.

In this context, this paper presents a new parallel algorithm of ultrasonic beam simulation relying on CIVA semi-analytical model, and implemented on three different architectures, multicore general purpose processors (GPP), Intel Many Integrated Core (MIC) accelerators and graphic processing units (GPU). A more restrictive scope (e.g. geometries, materials, modes. . .) is chosen to allow optimizations in order to reach an interactive experience for the user, typically 25 frames per second (fps). A benchmark shows that these architectures provides sizeable speedup and are effectively able to run simulations at interactive speeds. This is a major advance in the NDT field. Those developments will be integrated into the CIVA software for a wide range of applications, from training NDT experts to the calibration of probes on wedges.

2. PARALLEL ULTRASONIC BEAM MODELLING

Ultrasonic beam modelling consists in computing the Rayleigh-Sommerfield's equation diffraction formula. In CIVA model, it relies on a numerical computation through the pencil method[11]. It is based on the propagation through the different materials composing an inspected specimen of an axial ray and its paraxial rays forming a pencil as illustrated by figure 2. Here, the pencil is approximated by computing the axial ray path and deducing its deformation by taking into account the local deformation of each elementary segment of the ray path (propagation through a medium or interface crossing). Pencils are computed between a field point in the region of interest to be simulated in the specimen, and a surface source points of the ultrasonic transducer. The probe can be composed of a single or multiple elements, and can be immersed or in contact with the specimen.

In the general case, there is no simple solution to determine the ray path. However, in isotropic and homogeneous structures, analytical methods can be used. In this simulation, the scope is restricted to homogeneous isotropic specimens whose interfaces consist of canonical planar surfaces. Wave propagation can occur in a

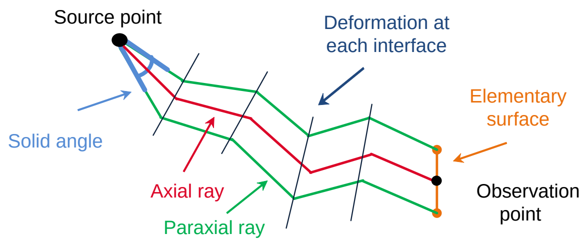


Figure 2: Pencil deformation through interfaces, along the optical ray path

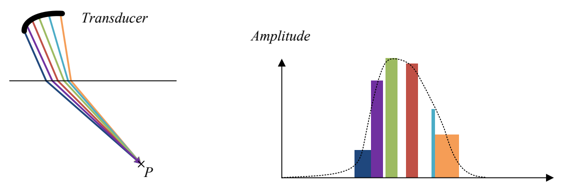


Figure 3: Impulse response formation: sum of pencils contribution according to delays

longitudinal or a transverse mode, on a direct or half-skip (with one back-wall reflexion) path without mode conversion.

Through those limitations, in immersion, it is possible to determine a fourth degree polynomial with real coefficients that models the path according to Snell-Descartes' Law[5]. The axial ray path can be obtained by computing its roots through Newton's method. Moreover, as the simulated scope is well known, analytical computations of the beam deformation (divergence factor) and of its Fresnel reflection/refraction coefficients have been determined. The beam is thus fully obtained through analytical formula and consists in a *time of a flight* (TOF) and a *displacement contribution.*

For each field point, once pencils are computed on the whole transducer, their elementary contribution are summed up (according to the delay law applied to the transducer) to obtain the impulse response of the elastodynamics displacement as show in figure 3. To account for the emitted signal, the impulse response is convoluted with the reference signal (in the frequency domain) to obtain the displacement signal for each point (3 signals, on for each coordinate). The maximum amplitude (A_{max}) and associated TOF (T_{max}) images are obtained by extracting the maximum of the module of the displacement signal. To account for phase shift, those maximums are obtained through the envelope of the displacement signal.

3. REFERENCE IMPLEMENTATION

A reference implementation has been carried out on GPP for analysis and validation purposes of the model. In addition to the previous basic algorithms, the following details are be provided. To avoid a too high computation time because of a too fine sampling of the sensors, pencils do not contribute via Diracs but take into account a small surface, resulting in a temporal staggering on the sensor. To avoid memory deallocation and reallocation, a step has been added to the previous algorithm to evaluate, preemptively, the resulting size of signal of displacement. Thus signals are allocated once before the summation occurs. To find the maximum amplitude, the envelope of the resulting signal is computed in the frequency domain. Algorithm 1 presents the reference algorithm for the computations.

This analytical model has been tested against CIVA 11.0, using NDT business rules (amplitude comparison between simulation and acquisitions in the range of 1 dB is considered as acceptable). With a sampling of $\lambda/2$ (λ the wavelength) over the sensor, the accuracy is good in NDT terms ($< 0.3\ dB$). By increasing this sampling, higher accuracy can be reached, but this quality is exceeds requirements for interactive simulations. No disparity between maximum time of flight has been detected. This validates the new model and its implementation in single precision floating point (32 bits). To accelerate signal processing algorithms (during STEP 3.2) Intel Math Kernel Library (MKL) was used for FFT. It is used both for reference and accelerated implementations on GPP.

4. EXTENDED CASE STUDY

This paper goes into extended details over a standard configuration which is both a good estimate of a typical simulation and provides a realistic value for its parameters. This configuration simulates an image of 101×101 pixel representing a 40×40 mm^2 region of interest, in which the longitudinal wave beam from a 32-elements immersed sensor (320 probe points) is propagated in direct mode and focused electronically. The simulated specimen is a wedge of homogeneous steel with a planar entry surface.

The presented work in this paper is based on an extensive case study over 18 ultrasonic field configurations representative of real life studies by increasing the values of the significant parameters of the standard configuration. However the simulated phenomenons are of the same size (same region of interest, same kind of probe or an inspected specimen of the same dimensions): only the definition increases (i.e. more pixels, a more complex description of the wedge). They are both meaningful in NDT terms and relevant toward this parallel algorithm, as they offer variations on the input parameters which are :

- the number of field points (increases the relative definition of the simulation) $\in$ [101×101:401×401]
- the number of probe points (relative to the number of active elements of a probe) $\in$ [320:1280] for [32:128] elements probes
- the temporal sampling of the signals (impacts simulation accuracy, dependent on the probe signal definition) $\in$ [90:360] Mhz

Algorithm 1: Fast Ultrasonic field simulation Algorithm - Pipeline version

```
1  foreach P, field point do
       // STEP 1: Pencil computation step
2      foreach E, sample from the sensor do
3          foreach Pencil do
               // Pencils reflect both geometry complexity and simulated modes
               // Numerical solving of the analytical polynomial
4              Ray = SolvePolynomialEquation(P,E,Pencil);
5              // Analytical computations
6              ToF = TimeOfFlight(Ray);
7              $\Delta$T = Staggering(Ray);
8              $\overrightarrow{d}$ = Displacement(Ray);

       // STEP 2: Determining signal size
9      Determining signal size(...);
10     $\overrightarrow{displacement}(t) = \vec{0}$; // Memory allocation and initialization to zero
       // STEP 3: Signal processing
       // STEP 3.1: Summation of the pencils to displacement signals
11     foreach E, sample from the sensor do
12         foreach Pencil do
               // Pencils reflect both geometry complexity and simulated modes
               // Temporal sampling of the signal is a user parameter
13             foreach $i \in [\![ToF - 0.5\Delta T; ToF + 0.5\Delta T]\!]$ do
14                 $\overrightarrow{displacement}(i) += \overrightarrow{d}$;

       // STEP 3.2: Maximum extraction through signals manipulations
15     Amplitude(t) := Envelope( Module( $\overrightarrow{displacement}(t) \bigotimes$ impulse(t) ) ); // FFTs required here
16     $\{(A_{max}, T_{max}) | A_{max} = Amplitude(T_{max}), \forall t\ A_{max} \geq Amplitude(t)\}$;
```

- the simulated modes (some defects or configurations require half-skip mode) $\in$ [longitudinal, transversal, direct and half-skip]
- the geometrical complexity of the inspected specimen (requiring multiple entry and back-wall surfaces) $\in$ [1 entry: 3/10 and 10/3 entry/backwall]

To illustrate the geometrical complexities, the number of pencils in each point, computed in step 1, is determined by the following formula: `entry surfaces` $\times$ (`directs modes` $+$ `halfskip modes` $\times$ `backwall surfaces`)

5. GENERAL PURPOSE PROCESSORS

Starting from the reference implementation, the following optimizations and parallelization tasks have been carried out on GPP.

OpeMP is used to multithread the code on multicore architectures. As work is packed by field points, a single point being computed by the same thread. This allows for a better memory management: for a given configuration, memory for signals can be reused from one point to another as signal length do not vary much (to benefit from MKL FFT, the number of samples per signal is increased to the next power of 2, thus signal sizes are often the same). FFT plans are shared between threads (and allocated on demand if a new size of signals is used). This allows for each field point the lowest memory consumption. As steps are no longer computed in a monolithic way, load balancing is easier.

To take advantage of Single Instruction Multiple Data (SIMD) instructions available on GPP, an algorithm should express a high regularity in its design. Two ways are commonly taken toward this goal : *SIMDization* where the programmer expresses the regularity himself and *vectorization* in which the compiler match the code with known pattern to benefits from SIMD instruction. As compilers improve *vectorization* improves and requires less and less help however due to te specifics of the computations SIMDization is retained.

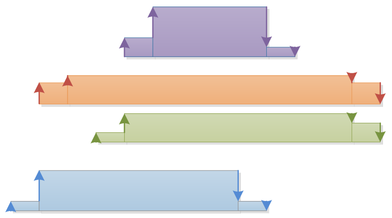


Figure 4: Time slots associated to pencil contributions, to be cumulated to form the impulse response

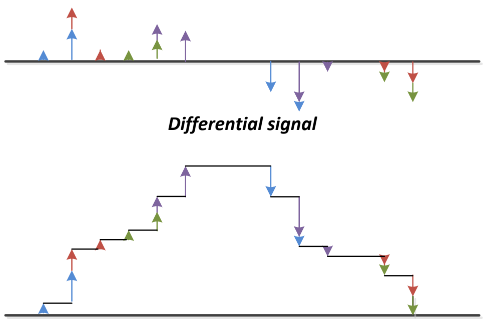


Figure 5: Differential summation of pencil contributions: first differences are summed (top) and then the final signal value is integrated

The algorithm has been SIMDized using Boost.SIMD[12] which allows the same code to target multiple SIMD instructions sets. In **Step 1**, pencils computations are packed together between a single field point and multiple sensor points according SIMD cardinal of the target machine. Rays are computed together for the same wave mode and for the same couple of surfaces. Then pencils TOF and contributions are extracted for the whole register. In **Step 2**, signal length determination operation *cannot* SIMDized due to both it low run time (<2ms on the whole set) and its low arithmetic complexity (min/max determinations, memory bound over temporal data analyses). In **Step 3.1**, summation of pencils contribution *cannot* be SIMDized because it was not suitable due to sparse accesses and inconstant sized pencil as shown in figure 4. However a scalar version of differential summation (DS) has been implemented. For DS, in a first step, the variation between two consecutive samples is accumulated to create a differential signal (fig. 5, upper signal). Then in a second step, these differences are integrated to reconstruct the signals summation (fig. 5, resulting signal). In **Step 3.2**, signal processing operations between FFTs (point-wise multiplication of two functions for convolution, envelope, module) have been vectorized.

To highlight the behaviour of the SIMDization of each part, the table 1 presents the performance of each computation step applied on all the field points at once: step 1 is carried out entirely before computing step 2 and so on. On one hand, the complete ultrasonic field simulation version using this scheme is nicknamed "Step 1+2+3". On the other hand, the one described by the algorithm 1 is a pipelined simulation.

Architecture	SIMD	Step 1 (pencils)	Step 3 (sum + signal proc.)	Step 1+2+3	Pipeline
Westmere	SSE4.2	$\times$3.2 (80%) [$\times$2.8 - $\times$3.3]	$\times$1.3 (33%) [$\times$1.1 - $\times$1.5]	$\times$2.1 (52%) [$\times$1.7 - $\times$2.6]	$\times$2.4 (60%) [$\times$2.2 - $\times$2.7]
Sandy Bridge	AVX	$\times$4.0 (50%) [$\times$3.6 - $\times$4.0]	$\times$1.2 (15%) [$\times$1.1 - $\times$1.5]	$\times$2.4 (30%) [$\times$1.9 - $\times$3.1]	$\times$2.7 (33%) [$\times$2.5 - $\times$3.2]
Ivy Bridge	AVX	$\times$3.8 (48%) [$\times$3.4 - $\times$3.9]	$\times$1.3 (16%) [$\times$1.1 - $\times$1.5]	$\times$2.3 (29%) [$\times$1.8 - $\times$3.0]	$\times$2.6 (32%) [$\times$2.4 - $\times$3.0]
Haswell	AVX2	$\times$3.8 (48%) [$\times$3.3 - $\times$4.2]	$\times$1.3 (16%) [$\times$1.1 - $\times$1.4]	$\times$2.3 (28%) [$\times$1.8 - $\times$3.2]	$\times$2.4 (31%) [$\times$2.4 - $\times$3.2]
MIC	KNC	$\times$5.2 (32%) [$\times$5.1 - $\times$7.8]	$\times$1.1 (7%) [$\times$1.0 - $\times$1.2]	$\times$1.7 (11%) [$\times$1.4 - $\times$2.5]	$\times$3.8 (24%) [$\times$1.1 - $\times$4.2]

Table 1: Optimized performance (SIMDization & differential sum) on both GPP and MIC: speedup attained compared to the reference scalar implementation. Efficiency is computed according to the SIMD cardinal in single precision (SSE 4.2 : 4, AVX/AVX2 : 8, KNC : 16). Those measures were realised with mono-thread execution. Results for the standard configuration and [min - max] of the set.

Step 1 shows an efficient SIMDization on SSE4.2 instructions. On wider vectors (AVX and AVX2), due to the need of high latency and low throughput operations such as transcendental functions (e.g. square roots, cosines...), no performance gains are reached when compared to SSE4.2 hence the relatively low efficiency (50%). **Step 2** *cannot* be accelerated with SIMDization. On **Step 3**, results analysis is more blurry due to the heavy interlacing between scalar summation, SIMDized functions and MKL calls (already using SIMD instructions in the reference implementation).

The last two columns of the table 1 present the two full simulations. In the table 1, performance of the global computation, both on the "Step 1+2+3" version and on the Pipeline one, only shows from $\times$1.7 to $\times$3.2 accelerations because of the important part of non SIMDized code (step 3.1 and step 2). Note that MKL FFT is used for both accelerated and basic code resulting in a lower speedup compared to a basic code with "basic" scalar monothread FFT implementation. The ×1 factor obtained on some configurations of the whole set can be explained. It results from a lot of computed but rejected pencils: in step 1, all possible pencils are computed for infinite surfaces and then if the specimen profile is restrictive (small surfaces, occlusions…) the computed pencil are dropped out. Thus on step 3.1, when summation occurs, a lot of pencil data are loaded and then discarded (sometime with a ratio of 100 rejects for 1 effective contribution). The chosen algorithm cannot provide any speedup in such worst case scenarios.

OpenMP is very efficient (tab. 2).

Architecture	Model	# cores N	Multithreaded pointwise version
Westmere	X5960	2×6 cores	×11.1 (92%) [×11.1 - ×11.9]
Sandy Bridge	E3-1290	1×4 cores	×4.0 (99%) [×4.0 - ×4.0]
Ivy Bridge	E5-2697v2	2×12 cores	×18.4 (77%) [×16.5 - ×19.5]
Haswell	E3-1240v3	1×4 cores	×3.9 (98%) [×3.9 - ×4.0]
MIC	Phi 3120	1×57 cores	×55.6 (98% from 4 to 228 threads)

Table 2: Impact of OpenMP : speedup and efficiency on the standard configuration and [min - max] speedups

Table 3 illustrates the global accelerations and the total computation time shown as fps. By the way, the CIVA 11.0 software on the Westmere GPP simulates this standard configuration in about 40 seconds. On the GPP, the best attained performance on the standard configuration is 29.8 fps on the Ivy Bridge, offers a ×38.9 speedup compared with the reference implementation. Those results allow an easier use of ultrasonic field simulation on realistic problems and are very encouraging for non yet interactive configurations.

GPP architecture	OpenMP + SIMD Speedup	fps
Westmere	×26.5 [×24.2 - ×31.7]	13.6 fps [0.9 - 13.6]
SandyBridge	×9.5 [×8.3 - ×12.9]	6.5 fps [0.4 - 6.5]
IvyBridge	×38.9 [×34.0 - ×60.3]	29.8 fps [1.9 - 29.8]
Haswell	×9.6 [×8.0 - ×13.0]	8.0 fps [0.5 - 8.0]
MIC	×144.2 [N/D]	13.8 fps [0.5 - 13.8]

Table 3: OpenMP+SIMD speedups for the standard configuration and [min - max] speedups

6. MANYCORE ACCELERATORS

The MIC benchmarked here is a 57-core manycore platform. It takes the form of an accelerator card plugged into an host computer. It has a peak performance of 2GFlops (simple precision), an internal bandwidth of 240 GB/s and an external bandwidth of 8 GB/s limited by PCI-Express connexion.

In the case of ultrasonic simulations, the cost of transferring data back and forth the accelerators does not impede the interactive capacities as the simulated regions of interest are small (a typical TOF or amplitude image which definition is 100×100 pixels only weighs about 40kB). Intel MIC are based on original Pentium 1 architecture and thus are compatible with x86 instructions. However two differences exist : firstly, due to the composition of the cores, to hide latency 4 threads per core are required. Secondly the only SIMD instruction set available on MIC is KNC which is a subset of the AVX-512. OpenMP and Boost.SIMD are available and used to generate multithreaded KNC code.

Table 2 presents MIC simulations regarding global scaling. On MIC, as on the GPP, the OpenMP parallelization performs well with a global efficiency of 98% (4 threads on 1 core to 4×57 threads on 57 cores). For SIMD computations, the results presented by table 1 follows the same pacing as on the GPP: a notable speedup on the compute heavy step 1 for pencil computations which are sunk into the final implementation resulting in a smaller ×2.4 factor (only partial SIMDization) on the standard configuration. The same penalty is felt here when geometrical complexity becomes too important and when too many pencils are rejected.

MIC runs at the 13.8 fps for the standard configuration that lies a speedup of ×144.2 (tab. 3). The reached performance matches GPP ones.

7. GRAPHICS PROCESSING UNIT

Since almost a decade GPU are extensively used as general purpose computing accelerators (GPGPU). GPU belongs to the manycore accelerator family, with thousands of small compute cores, gathered in streaming multiprocessors controlling instructions flows. As for the MIC, in ultrasonic simulation, data transfers do not hinder interactivity. However the host needs to drive the device computations by specifying the compute queue of successive compute kernels. The following developments have been realized in NVidia CUDA (Compute Unified Device Architecture).

To benefit from the GPU capacities, the algorithm must be refactored into kernels, each realizing a batch of a specific and single task over data in the GPU memory. The space over which the computations take place is divided into a grid and block hierarchy. Blocks are dedicated to a single pixel and pencils computations are packed per ray path mode. CUDA threads from a single block work in a similar fashion to SIMD on GPP, best suited for computational regularity. A specific kernel used for pencil computations is developed for each ray mode. Then a scan takes place over the pencils to determine the signal size. The summation occurs with a specific kernel. And finally signal processing kernels are chained up to obtain the resulting pixel value. Specific signal processing kernels have been developed to complement the FFT capabilities of cuFFT library (envelope, convolution, module).

Table 4 presents the performance of three different GPU on the whole dataset. Due to the absence of monothread code on the GPU, only the results of the algorithms are shown. The GTX Titan outperforms Fermi architecture GPU by a factor of almost 5. It reaches a performance of 28.8 fps on the standard configuration. GTX 580 have less memory but more cores at a higher frequency and so are faster than the C2070.

GPU Architecture	Model	run time	Total FPS
Fermi	C2070	210.4 ms	4.8 fps
448 cores @ 575MHz		[177.1 - 955.2]	[1.0 - 5.6]
Fermi	GTX 580	153.4 ms	6.5 fps
512 cores @ 772MHz		[128.2 - 711.6]	[1.4 - 7.8]
Kepler	GTX Titan	34.8 ms	28.8 fps
2688 cores @ 837MHz		[30.7 - 376.3]	[2.6 - 32.5]

Table 4: GPU performance overall (run times and fps) on standard configuration and min/max between bracket

The performances between Kepler and Fermi GPUs is over the theorical speedup from the increase in peak performance (tab. 4). One boasted improvement about Kepler architecture is the performance of the *atomic* operations. As step 3 summation relies on those operations due to noncoherent concurrent accesses, the impact of *atomic operation* has been evaluated. A variation of this algorithm is presented *without atomics* in which they have been replaced with an equivalent non-atomic memory operations. Simulations are obviously wrong but the number of global memory accesses remains the same. Table 5 presents the performance of three different GPU *with* and *without atomics*. It indicates that on a new hardware, Kepler-class GPU such as the GTX Titan, almost no extra cost occurs for the summation part of the algorithm (step 3.1) whereas performance drops on previous architecture Fermi.

GPU Architecture	GPU Model	*with atomics*	*without atomics*	*atomics* overhead
Fermi	C2070	210.4 ms	88.6 ms	$\times$2.4
Fermi	GTX 580	153.4 ms	58.2 ms	$\times$2.6
Kepler	GTX Titan	34.8 ms	33.4 ms	$\times$**1.05**

Table 5: GPU performance of atomic operations on the standard configuration

8. CONCLUSION AND PERSPECTIVES

This works delved into interactive ultrasonic beam propagation simulation. It presents a first insight into the performance obtained on three architecture classes along with a short characterization. Very good performance on each targeted platforms have been reached on the standard configuration, with even interactive rate for two of them.

- For the GPP, the Ivy Bridge has reached a *realtime* performance of 29.8 fps with the use of both SIMD and multithread parallelization (faster than the targeted 25 fps).
- For manycore, 13.8 fps have been reached.
- For GPU, the GTX Titan is able to deliver a *realtime* performance of 28.8 fps.

On the extended configurations set, performance depends heavily on the parameters.

This work constitutes is a *major breakthrough* in the NDT field to replace simple ray path tools in assisting an operator to predict the outcome of the simulated inspection. As far as we know, this is the first time that a complete field simulation is carried out on classical problems at interactive pace in the NDT field. It shall be integrated in future versions of the CIVA software. The current accelerated implementations both on GPPs, MIC and GPUs should benefit from computation power increase of the next generation that will make more configurations to run in realtime.

Multiple extensions can be applied to obtain a complete integrated solution for interactive ultrasonic field simulations. On one part, this work shall be extended to address more complex configurations like non planar geometries, multi-bounce modes. On the other part, performance is highly dependants of the range of the simulation parameters. Some are inherent to the needs of the user. However the geometrical complexity can be addressed by adding heuristics to avoid the computation of useless pencils. Moreover, the integrated simulation should take care of memory management to enable very large configurations, especially for accelerators. Finally, with CIVA model field simulation is required to simulate defect responses. Interactive UT field simulation is the first building block for interactive echo simulation.

REFERENCES

1. F. Reverdy, G. Ithurralde, and N. Dominguez, "Advanced ultrasonic 2d phased-array probes," in *proceedings of the World Conference of NDT 2012*, 2012.
2. O. Martínez-Graullera, R. T. Higuti, C. J. Martín, L. G. Ullate, D. Romero, and M. Parrilla, "Improving synthetic aperture image by image compounding in beamforming process," *AIP Conference Proceedings* **1335**(1), pp. 728–735, 2011.
3. D. Romero-Laorden, O. Martínez-Graullera, C. J. Martín-Arguedas, M. Pérez-López, and L. Gómez-Ullate, "Paralelización de los procesos de conformación de haz para la implementación del total focusing method," in *12 Congreso Español de END*, 2011.
4. M. Sutcliffe, M. Weston, B. Dutton, P. Charlton, and K. Donne, "Real-time full matrix capture for ultrasonic non-destructive testing with acceleration of post-processing through graphic hardware," *NDT & E International* **51**(6), pp. 16 – 23, 2012.
5. J. Lambert, A. Pedron, G. Gens, F. Bimbard, L. Lacassagne, and E. Iakovleva, "Performance evaluation of total focusing method on GPP and GPU.," in *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing (DASIP), Karlsruhe, Germany, October 23-25, 2012*, pp. 1–8, IEEE, 2012.
6. D. Romero, O. Martinez-Graullera, C. J. Martin, and L. G. Ullate, "GPGPU techniques to accelerate modelling in NDE," *AIP Conference Proceedings* **1211**, pp. 1991–1998, 2010.
7. C. A. Nahas, P. Rajagopal, K. Balasubramaniam, and C. V. Krishnamurthy, "Graphics processing unit based computation for NDE applications," in *American Institute of Physics Conference Series*, *American Institute of Physics Conference Series* **1430**, pp. 1998–2005, May 2012.
8. Y. Sakai, M. Nagano, Y. Hirose, and Y. Ikegami, "An investigation of the optimal simulation condition for nde with the help of high-speed and high-accuracy fem simulator," in *JRC-NDE 2012 - Modelling*, pp. 308–313, 2012.
9. P. Huthwaite, "Accelerated finite element elastodynamic simulations using the GPU," *Journal of Computational Physics* **257, Part A**, pp. 687 – 707, 2014.
10. G. Toullelan, R. Raillon, S. Chatillon, and S. Lonne, "Results of the 2013 UT modeling benchmark obtained with models implemented in CIVA," *AIP Conference Proceedings* **1581**, pp. 2093–2100, 2014.
11. N. Gengembre, "Pencil method for ultrasonic beam computation," in *World Congress on Ultrasonics*, pp. 1533–1536, (Paris, France), 2003.
12. P. Estérie, M. Gaunard, J. Falcou, J.-T. Lapresté, and B. Rozoy, "Boost.SIMD: Generic programming for portable SIMDization," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, *PACT '12*, pp. 431–432, ACM, (New York, NY, USA), 2012.