SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2022, Vol. 65 152103:1–152103:16
https://doi.org/10.1007/s11432-020-2976-5

# A fully-customized dataflow engine for 3D earthquake simulation with a complex topography

Bingwei CHEN[1,3], Haohuan FU[2,3*], Wayne LUK[4] & Guangwen YANG[1,2,3]

[1]*Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing* 100084*, China;*
[2]*Ministry of Education Key Laboratory for Earth System Modeling, Department of Earth System Science, Tsinghua University, Beijing* 100084*, China;*
[3]*National Supercomputer Center in Wuxi, Wuxi* 214072*, China;*
[4]*Department of Computing, Imperial College London, London* SW7 2AZ*, UK*

Received 21 March 2020/Revised 5 June 2020/Accepted 2 July 2020/Published online 29 November 2021

**Abstract** With HPC (high performance computing) evolving into the exascale era, improvements in computing performance and power efficiency have become increasingly more important. Based on our previous work on enabling earthquake simulations on a large scale on Sunway TaihuLight, we further explore other possibilities to improve the application through a fully-customized hardware design on reconfigurable FPGA (field programmable gate array) devices. We investigate the feasibility and the potential benefits of a complete fixed-point design. We first perform a coarse-resolution-based simulation to analyze the representation range and precision needed to capture both the total energy and the energy distribution of variables over space and time. We then derive a complete fixed-point design that identifies the suitable bitwidth for major categories of variables and dynamically represents the range through a dynamic scaling scheme. Finally, we use the optimized fixed-point design to run a case of the Wenchuan earthquake to demonstrate the potential of supporting large-scale scientific simulations on FPGA devices. The results demonstrate that an 18-bit fixed-point design already provides an almost identical description of the seismic events in the Wenchuan scenario down to a single-precision floating-point version and provides sustainable performance equivalent to 13.1 Intel Xeon Gold 6154 18-core CPUs or 2.10 Sunway 260-core processors, with performance per watt (power efficiency) improved by 15.3 and 3.72 times compared with the Intel Xeon Gold 6154 18-core CPUs and the Sunway 260-core processors, respectively.

**Keywords** reconfigurable computing, earthquake simulation, FPGA, hardware-software co-design, architecture exploration

**Citation** Chen B W, Fu H H, Luk W, et al. A fully-customized dataflow engine for 3D earthquake simulation with a complex topography. Sci China Inf Sci, 2022, 65(5): 152103, https://doi.org/10.1007/s11432-020-2976-5

## 1 Introduction

Since the invention of the first supercomputer in 1964 [1], scientists have found that theorems and experiments have not met the increasing complexity required when solving scientific problems. Therefore, numerical computing began to be applied in significant scientific domains [2], such as earthquake simulations [3], seismic inversions [4], climate simulations [5], sky simulations [6,7], and phase-field simulations [8]. Till now, numerical computing has supported much of cutting-edge research activities in the corresponding fields.

The development of HPC (high performance computing) platforms and the development of computer-related science domains have been driving progress in each field in the last few decades. Better computing power has led to more detailed simulations of various physics, chemistry, or even molecular processes that scientists attempt to investigate. Improved simulations resulting in improved understanding of the processes has led to more complex models and higher demands for computer power.

In the last decade, increasing heat dissipation and power consumption issues have pushed the architecture of computing nodes of the leading supercomputers from being homogeneous to heterogeneous.

* Corresponding author (email: haohuan@tsinghua.edu.cn)

 

The heterogeneous supercomputers consist of both general-purpose processors (traditional CPUs (central processing units)) and many-core accelerators that provide intensive computing capabilities, such as GPUs (graphics processing units) [9], Intel Xeon Phi [10], and Shenwei 26010 processors [11].

With the exascale era on the horizon, improvements in both computer performance and power efficiency have become increasingly important and challenging. HPC researchers have put tremendous efforts into exploring the next generation of the system architecture. Potential changes for the traditional HPC design methodology have been widely researched [12]. Since 2011, there have been many projects that promote hardware-software co-design for exascale design space exploration [12–14], which identifies leading high-impact scientific applications and creates a collaborative design process of both hardware and algorithms using a reconfigurable FPGA (field programmable gate array) platform as an alternative platform for design verification and emulation.

In this work, we choose large-scale earthquake simulations as our target application and explore the performance potential of fully-customized hardware design on reconfigurable FPGA devices. There are two major reasons that we chose the earthquake simulation application.

First, earthquake simulation programs are of great scientific and societal importance. Earthquakes are one of the most severe natural disasters that humanity faces. With a direct detection of only 12 km of the subsurface, seismic simulation becomes a vital tool for the research activities that aim at improving the understanding of the subsurface structures and the mechanism of earthquake formations. Regarding the reduction of earthquake hazards, the simulation programs can also play important roles that range from producing earthquake drill scenarios to providing quantitative evaluation of potential risks for determining the earthquake insurance costs [15].

Second, earthquake simulations are one of the traditional supercomputing applications that demonstrate the most demanding level of both computational and memory challenges. With a limited understanding of the subsurface and to achieve an acceptable level of accuracy, a typical earthquake simulation program involves several different processes, such as a velocity update, stress update, and attenuation. To capture the elastic features of seismic waves, the program could involve up to 50 three-dimensional (3D) arrays. As a result, large-scale earthquake simulations are only possible in some of the leading supercomputer facilities, such as the Earth simulator in Japan [16], Jaguar [17] and Titan [18] in the United States, and Sunway TaihuLight [19] in China.

This paper presents a fully-customized hardware design for our earthquake simulation application SWST on Sunway TaihuLight [20], which can capture the complex surface topography and provide a sustained performance of 9.07 Pflops when using over 10 million cores of the system. Based on previous work that has achieved a well-tuned performance on an existing leading-edge supercomputer, we continue to explore the potential benefits of a fully-customized dataflow design on reconfigurable FPGA devices. We investigate the methods and tools for analyzing the numerical behaviors and for deriving a complete fixed-point design that can achieve both a similar level of accuracy and improved computational efficiency compared with the current floating-point designs. Our major contributions are as follows:

(1) We propose a coarse-resolution simulation-based analysis approach to identify the representation range and precision needed to capture both the total energy (covering over 99% of total energy) and the energy distribution of variables (achieving a coefficient of determination larger than 98%) over space and time (detailed in Section 4).

(2) Based on the analysis results, we derive a complete fixed-point dataflow engine that not only assigns the suitable bitwidth to each variable but also provides a customized scaling scheme that maintains the most informative bits at the front during the time-stepping process (detailed in Subsections 5.1 and 5.2).

(3) Using such a design, we implement a CPU-FPGA hybrid 3D earthquake simulation program that can use both a single FPGA and multiple FPGAs with a curvilinear coordinate transformation and traction image method to deal with complex surface topography (detailed in Subsection 5.3).

Our optimized design could accommodate one deep-pipelined core, which can support 1026 arithmetic operations and provide a sustainable performance that is equivalent to 13.1 Intel Xeon Gold 6154 18-core CPUs, or 2.10 Sunway 260-core CPUs. The simulation results of the Wenchuan model demonstrate that an 18-bit fixed-point design can already provide an almost identical description of the seismic events to a single-precision floating-point version. If we further push toward a 15-bit or even 12-bit design, we can still maintain a correct description on the peaks of strong ground motion with further performance improvement.

## 2 Related work

### 2.1 Precision analysis and optimization

In the early days of FPGA acceleration, due to the limitation of the available reconfigurable resources, precision analysis and optimization began from standalone functions and kernels.

Analytic methods, such as affine arithmetic and the data-range propagation method [21], are proposed to determine the range and the number of bits needed to represent it adequately. In terms of accuracy, studies have discussed the sensitivity of the output to errors in the intermediate variables for fractional bitwidth optimization [22], and the Taylor series has been used to propagate the worst-case truncation error at each node in the dataflow graph to the output. Using such methods, hardware designs with optimized bitwidth for every single variable can be achieved for various accuracy requirements of a specific elementary function [23, 24].

With an increasing amount of programmable resources, FPGAs can gradually accommodate more complex computational workflows. As a result, methods also must be developed for analyzing and optimizing the precision in such scenarios. As scientific computations usually involve tens of thousands of time steps, the above analytic methods, which usually take a fairly conservative approach, propagate both the range and the error. The analysis becomes meaningless after several time-stepping iterations. As a result, a number of existing efforts take an enumerating approach to experiment with several precisions [25, 26] to determine the most suitable configuration of number representations. However, the high computational cost makes it not feasible for extremely complex 3D numerical simulation cases.

For floating-point applications on FPGAs, optimization methods based on integer linear programming [27] and mixed-integer geometric programming [28] are proposed to compute the optimal reduced precision and the optimal resource allocation for the resulting FPGA design. However, these methods only work for floating-point functions with a clear error requirement. For numerical simulations that involve hundreds of thousands of time steps, the accuracy required for each step to maintain the numerical stability and accuracy is unclear. For a precision study on the simulation of 2D shallow water equations, the authors propose the use of the numerical differences caused by different compilation options as an acceptable range of error in long-term simulations. However, the correct precision is still determined through an enumeration of a number of different mantissa bitwidths, which, as mentioned above, would not be a feasible solution for complex 3D cases.

With a similar goal to investigate the tradeoff between accuracy and performance, our analysis starts with a coarse-resolution simulation run. We analyze the numerical behaviors in space and time to derive the appropriate computational scheme to capture the physics phenomena sufficiently and accurately.

### 2.2 Seismic modeling

As one of the major traditional HPC applications, seismic modeling is also a typical target for FPGA researchers.

In 2004, He et al. [29] achieved approximately 15.6 times acceleration in 2D prestack Kirchhoff time migration on the SPACE platform at a conservative speed of 50 MHz compared with 2.4 GHz Pentium 4 workstations, which can calculate the Kirchhoff summations for 50 million points per second.

In 2007, Pell et al. [30] achieved a 40 times speedup for subsurface offset gathers on a MAX-1 accelerator card with a Xilinx Virtex-4 FPGA.

In 2008, Fu et al. [25] optimized seismic computation with an FPGA. Under the constraint of the PCI Express X8 bandwidth between CPU and FPGA, they could support two concurrent computation cores and achieve 13.7-times speedup on a Maxeler Technologies MAX-1 Card with a Xilinx Virtex-4 FX100 FPGA compared with a software implementation on Intel Xeon CPU clocked at 1.86 GHz.

In 2013, Medeiros et al. [31] implemented 2D RTM on an Altera Stratix 260E, with 160 PEs, which had achieved a 29 times speedup in comparison with a CPU implementation, and it was only 13% slower than the GPGPU (general-purpose graphics processing unit) implementation. As for power consumption, the CPU-FPGA design was 1.7 times more efficient than the GPGPU system.

In 2019, Bittencourt et al. [32] designed a hardware/software co-design system on the FPGA platform, which could provide a scalable model for 2D RTM on HPC environments. The team's goal was to provide information about the system architecture considering the computational challenges in seismic imaging for oil exploration. The team implemented the system on an Intel Arria 10 GX FPGA Development Kit board. For performance, they achieved an approximately 112-times speedup over a single core of Intel

Xeon Gold 6148. When compared with a Titan XP GPU, the energy consumption of the system was reduced to approximately 55%.

Most existing efforts on seismic modeling focus on the processing of small-scale artificial seismic events used for oil and gas exploration and are usually constrained to 2D cases. Our work addresses a full workflow of natural earthquake simulations on the FPGA devices. With the capability to capture complex surface topography, our design contains 12 velocity arrays, 24 stress arrays and three elastic parameter arrays, as well as nine grid transformation coefficient arrays. The goal is to effectively evaluate the possibility of supporting extreme-scale high-performance computing applications on FPGA platforms.

## 3 Our target application: 3D elastic earthquake simulation with accurate surface topography

### 3.1 The numerical method

We use seismic wave equations to describe the propagation of seismic wave. To depict the effects caused by complex surface topography in most active seismic regions, we use a finite difference method with curvilinear grid transformation as our major numerical method [20].

In 3D inhomogeneous isotropic elastic media, the first order velocity-strain equations, including the momentum equation and the generalized stress-strain relationship, which control the rule of propagation of elastic waves, can be written as i.e.,

$$\rho\frac{\partial v_i}{\partial t} = \sigma_{ij,j} + f_i, \tag{1}$$

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda v_{k,k}\delta_{ij} + \mu(v_{i,j} + v_{j,i}), \tag{2}$$

where $\lambda$ and $\mu$ are the Lamé parameters in isotropic media, $i, j \in (x, y, z)$ are the directions in coordinate system, $v_i$ are velocity components, $\sigma_{ij}$ are stress components, $f_i$ are seismic sources.

For our convenience, we use $(x, y, z)$ to denote our physical space and write Eqs. (1) and (2) in a matrix form

$$\frac{\partial \boldsymbol{W}}{\partial t} = \boldsymbol{A}\frac{\partial \boldsymbol{W}}{\partial x} + \boldsymbol{B}\frac{\partial \boldsymbol{W}}{\partial y} + \boldsymbol{C}\frac{\partial \boldsymbol{W}}{\partial z}, \tag{3}$$

where $\boldsymbol{W}$ includes unknows needed to be solved, including both velocity components and stress components: $\boldsymbol{W} = [v_x, v_y, v_z, \sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}]^{\mathrm{T}}$.

To deal with the complex surface topography, we use curvilinear transformation to transform the irregular physical coordinates $(x, y, z)$ to computational space $(\xi, \eta, \zeta)$:

$$x = x(\xi, \eta, \zeta), \quad y = y(\xi, \eta, \zeta), \quad z = z(\xi, \eta, \zeta). \tag{4}$$

By performing such transformations, our earthquake simulation program can then process the scenarios with significant changes on surface topography. Most big earthquakes happened because of faulty plates [33,34], which usually results in complex surface topography. A typical example is the hilly region of the Wenchuan earthquake, as shown in [20, Figure 1]. In such an area, the complex surface topography would cause severe effects, such as amplification and basin effects near the edges, influencing the accuracy of the earthquake simulation to a large extent. Adequately dealing with the side effects plays an important role in describing the details of the seismic wave. More discussions about the case study of the Wenchuan earthquake simulation are provided in Section 6.

We compute derivatives in a regular computational space while solving the problem in a physical space. In finite difference method, to reduce the dispersion and dissipation error of one-sided differences, we apply dispersion relation preserving (DRP) in MacCormack-type schemes, which is called DRP/opt MacCormack schemes [35,36]. We use Runge-Kutta time marching schemes [37] to refine the results and split the difference operator into a forward operator and a backward operator, which are used in Runge-Kutta time marching schemes. The fourth-order Runge-Kutta scheme which calculates the wavefields from $nt$ to $n(t+1)$ is shown as follows:

$$\boldsymbol{W}^{(1)} = \boldsymbol{W}^{n}, \tag{5}$$

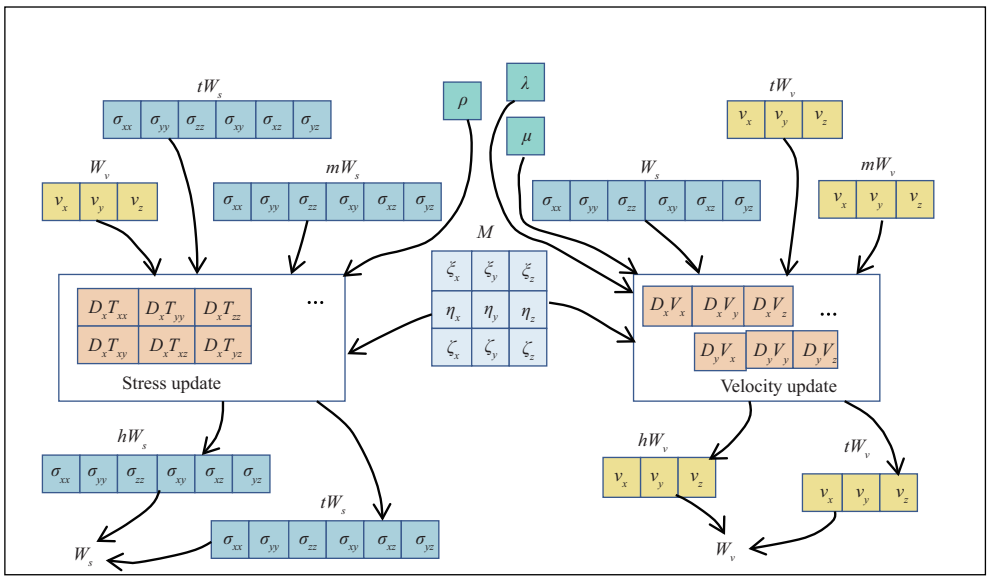


**Figure 1** (Color online) The computational workflow of our target earthquake simulation program. $W_v$, $mW_v$, $hW_v$, and $tW_v$ are the four Runge-Kutta velocity states, each of which has three components on the three axes. $W_s$, $mW_s$, $hW_s$, and $tW_s$ are the four Runge-Kutta stress states, each of which has six components. $\rho$, $\mu$, and $\lambda$ are the three elastic parameters. $M$ is the array of grid transformation coefficients, with 9 components for the transformations along different axes.

$$\boldsymbol{W}^{(2)} = \boldsymbol{W}^n + \alpha_2 \Delta t \hat{L}^{\mathrm{FFF}}(\boldsymbol{W}^{(1)}), \tag{6}$$

$$\boldsymbol{W}^{(3)} = \boldsymbol{W}^n + \alpha_3 \Delta t \hat{L}^{\mathrm{BBB}}(\boldsymbol{W}^{(2)}), \tag{7}$$

$$\boldsymbol{W}^{(4)} = \boldsymbol{W}^n + \alpha_4 \Delta t \hat{L}^{\mathrm{FFF}}(\boldsymbol{W}^{(3)}), \tag{8}$$

$$\begin{aligned}\boldsymbol{W}^{n+1} = \boldsymbol{W}^n + \Delta t \big[&\beta_1 \hat{L}^{\mathrm{FFF}}(\boldsymbol{W}^{(1)}) + \beta_2 \hat{L}^{\mathrm{BBB}}(\boldsymbol{W}^{(2)}) \\ &+ \beta_3 \hat{L}^{\mathrm{FFF}}(\boldsymbol{W}^{(3)}) + \beta_4 \hat{L}^{\mathrm{BBB}}(\boldsymbol{W}^{(4)})\big],\end{aligned} \tag{9}$$

where $\hat{L}^{\mathrm{FFF}}$ and $\hat{L}^{\mathrm{BBB}}$ are the linear combination of one-sided difference operator, the superscript F means forward direction and B means backward direction.

Free surface boundary condition is of great importance in seismic wave simulation. The free boundary condition we used is called traction image method [35].

### 3.2 The computational workflow

Figure 1 provide a more clear picture about what need to be computed and stored in each time step. We have in total 48 3D arrays, which describe the four Runge-Kutta states of three velocity components, the four Runge-Kutta states of six stress components, the three elastic parameters, and the nine grid transformation coefficients respectively. If including the intermediate arrays within kernels, there would be about 100 variables evolved in each grid.

In each Runge-Kutta step, we need to (1) update the stress states $hW_s$, $tW_s$ using the velocity state $W_v$ and the stress states $tW_s$, $mW_s$; (2) update the velocity states $hW_v$, $tW_v$ using the velocity states $tW_v$, $mW_v$ and the stress state $W_s$; (3) use the updated $hW_s$, and $tW_s$ to compute the next $W_s$; (4) use the updated $hW_v$, $tW_v$ to compute the next $W_v$. In total, around 1016 arithmetic operations need to be performed for the processing of one single grid in the 3D space for one Runge-Kutta step. We need four Runge-Kutta steps to accomplish one complete iteration of the time step.

The complexities in both the number of variables and computational patterns correspond to the advanced features that our target earthquake simulation program provides. The four-step Runge-Kutta method brings an improved numerical accuracy of the scheme. The introduced grid transformation coefficients help to convert the possible irregular surface topography into a regular aligned 3D array, so that the capability to describe irregular shapes and the high computational efficiency can be maintained at the same time. The elastic parameters are added to include elastic behaviors into the simulation. Therefore, some of the most advanced simulation features are achieved by paying the price of increased complexity in both computation and memory. Implementing such a workflow using the generally assumed

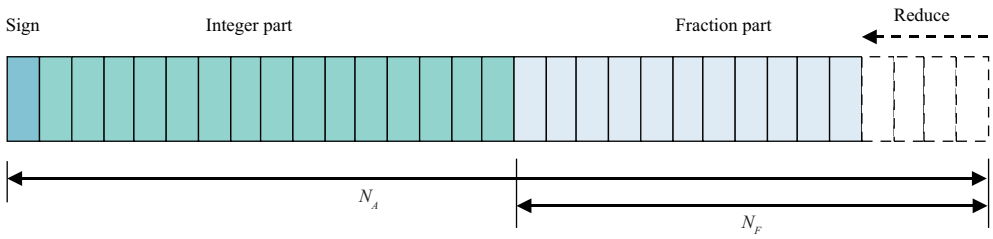


**Figure 2** (Color online) Fixed point format.

single-precision floating-point on an FPGA chip is an overwhelming design goal to achieve. Deriving a complete fixed-point design through careful numerical analysis and systematic optimization becomes the only viable option.

## 4 A customized fixed-point number representation

### 4.1 Algorithmic formulation

A fixed point number can be described by an integer $I$ and a scaling factor $S$. And the value would be expressed by $V = I \times S$, where $V$ is the value of the number. To express tons of grids in fixed-point formats, it is less efficient to store both $I$ and $S$ for every grid. Moreover, due to the complexity of operations between fixed point numbers with different scaling factors, we usually share the scaling factor $S$ among grids of the same variable, which usually cover continuous range in physics space. The fixed point format for one variable can be described as the number of total bits $N_A$ and the number of fraction bits $N_F$, which is shown as Figure 2, where scaling factor $S = 2^{N_F}$.

In general, constructing a computing unit for fixed point with less bits costs less resources, which saves more space for instruction parallelism. Thus, reducing bits of fixed-point format for variables would be one of the effective keys to optimize the hardware and software co-design. By reducing the bits of a kind of fixed-point format, the total bits $N_A$ and fraction bits $N_F$ both decrease, and scaling factor $S$ also changes. However, using less bits would lose the precision of the corresponding bits, how to reach the balance of the precision and performance would be a challenge.

### 4.2 The general methodology

For most physics-oriented numerical simulation programs, the key events to capture are the injection of energy into the system, the distribution and reallocation of the energy over the simulation domain, and the dissipation of energy in the system. For example, in an earthquake simulation, the earthquake source is injected into the simulation region, the wave propagation module then simulates the propagation of the seismic waves to different parts of the region and the gradual attenuation of the waves. For weather and climate modeling, we have a more complicated scenario, with injected solar energy, redistribution of energy and water, consumption of fossil energy, and many other activities happening continuously.

In our earthquake simulation target application, we perform the analysis also from the perspective of capturing the major energy distribution of the system. Our range and precision analysis of the entire simulation process consists of the following major parts:

(1) **A coarse-resolution run.** Our approach is a dynamic approach that makes the analysis and optimization based on actual runs of the simulation. To minimize the computational costs (especially for large-scale earthquake simulations, which can occupy entire supercomputers for days or even weeks), we do the simulation and analysis using a coarse-resolution grid. In general, we set the coarse resolution to be 1/2 or 1/4 of the target one, which corresponds to roughly 6% or 1% of the original computational cost.

(2) **Range analysis.** We first analyze the energy sum of each time step. We consider the energy as the sum of squares of the variables (velocity, stress, or other parameters) over the 3D domain, i.e., $\Sigma_{i,j,k}{W_v}^2$ or $\Sigma_{i,j,k}{W_s}^2$. We perform range analysis to ensure that at least 99% of the total energy is properly described. Then we perform the analysis for all the different time steps, to study the change of the energy over time, and to derive a customized scaling scheme that would help to keep the most significant bits in the optimized design.

(3) **Precision analysis.** While the previous part ensures that the dynamic range of the simulation is well captured, in this part, we need to quantify the additional precision bits needed for each variable, so that the distribution of energy is also described in the correct way. We evaluate the accuracy of the computed distribution using the coefficient of determination $R^2$, which reflects how well the reduced-precision result represent the full-precision result. For this metric, we can set the target to different ratios (98%, 90%, etc.) for different accuracy requirements.

(4) **Verification of the design.** Based on the range and precision analysis, we achieve the optimized bitwdith with a customized scaling scheme using the coarse-resolution run. We then apply the design to perform the target-resolution simulation, and verify whether both the energy sum and the energy distribution is kept within the required regions.

### 4.3 Tools and cases

We use Maxeler dataflow computing platforms [38] to implement our designs. Maxeler dataflow programming model can enable us to describe complex hardware designs in the form of Java code, and explore various fixed-point and floating-point representations. The Maxeler programming tool also provides a bit-accurate simulation feature, which we use to generate the simulation results of various precision configurations.

In this study, we run our experiments on two major cases. One is an ideal case that is more general than various realistic cases, and is more conducive to our focus on theoretical analysis. The effects of complex earthquake sources, 3D media and undulating terrain are not considered in the ideal case. In the ideal case, we use a simple point source with Ricker wavelet, which is widely used in seismic wave simulations. A homogeneous half-space with the compressional wave speed Vp = 6000 km/s, the shear wave speed Vs = 3464 km/s and the density $\rho = 2670$ kg/m$^3$ is used. We use a spatial resolution of 100 m and a time step of 0.009 s to discretize the test model, which consists of $100 \times 100 \times 50$ 3D grid points.

The other one is the Wenchuan earthquake case, which evaluates the performance of our complete fixed-point design in realistic scenarios, with a computational region of 400 km × 400 km × 80 km. We use the elevation data from SRTM_PLUS [39] to generate the surface topography of the computational region. We use the seismic source modeled from dynamic rupture simulation by Zhang et al. [40], which is consistent with other kinematic inversion results. The realistic model of the Wenchuan earthquake can be treated as a more rigorous test to evaluate our complete fixed-point design.

### 4.4 Range analysis

As mentioned above, the key information at a given time step is the velocity and stress values that describe the current distribution of energy over the space. While one can always increase the number of bits to achieve a more accurate representation of the distribution, a minimum number of bits that capture the major information, e.g., over 99% of the total energy, would be a nice candidate to achieve the balance between accuracy and efficiency.

In range analysis, we firstly investigate the value of variables at each single time step. Figure 3 demonstrates the cumulative percentage of energy for exponent values ranging from 29 to $-5$ for the intermediate variable $D_xT_{xx}$ and from 3 to $-31$ for the intermediate variable $D_xV_y$ at the 120th time step. We pick the 120th time step, as it is a time point that the seismic wave already propagates to almost the entire region of the 3D domain. The curve of cumulative percentage for $D_xT_{xx}$ demonstrates that the values with an exponent larger than 14, 13, and 11 already contribute to over 99%, 99.9%, and 99.99% of the total energy simulated at the exact moment, while for $D_xV_y$ the corresponding exponents are $-10$, $-12$, and $-13$, which suggests a reduced-precision design might only need to consider values with a magnitude larger than $2^{14}$ for $D_xT_{xx}$ and $2^{-10}$ for $D_xV_y$ (the corresponding scaling factors are $S = 2^{14}$ for $D_xT_{xx}$ and $S = 2^{-10}$ for $D_xV_y$ respectively).

If we assume that we only keep the values that contribute to the first 99% of the total energy, we can quickly derive the number of bits $N_A$ and $N_F$ needed for representing the range in a fixed-point representation, and the corresponding scaling factor $S$ can be decided by $N_F$. For most velocity, stress, and intermediate variables, we need 12 to 15 bits to represent the dynamic range. For grid transformation coefficient variables, which simply records the relationship between the physical grid and the computational grid, we only need 5 bits to represent the dynamic range.

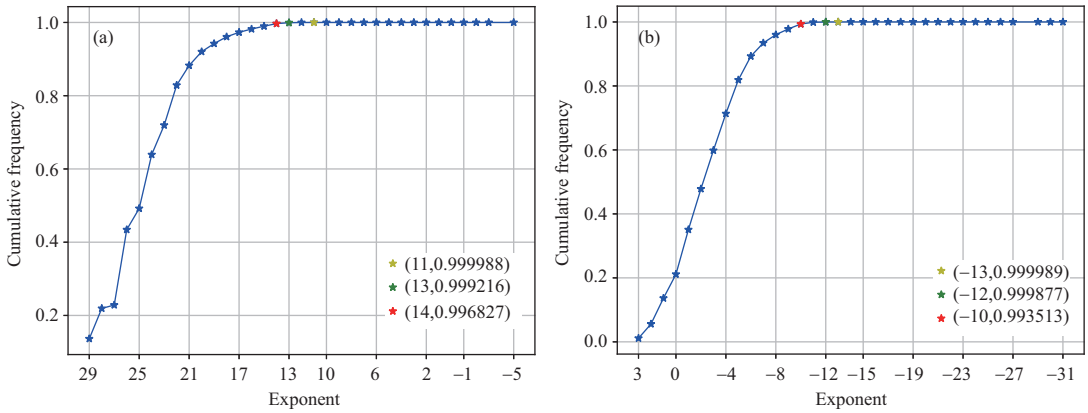


**Figure 3** (Color online) The cumulative percentage of energy for different exponent values for the intermediate variables $D_x T_{xx}$ (a) and $D_x V_y$ (b) at the 120th time step. The red, green, and yellow stars correspond to the point where the cumulative percentage surpasses 99%, 99.9%, and 99.99%.

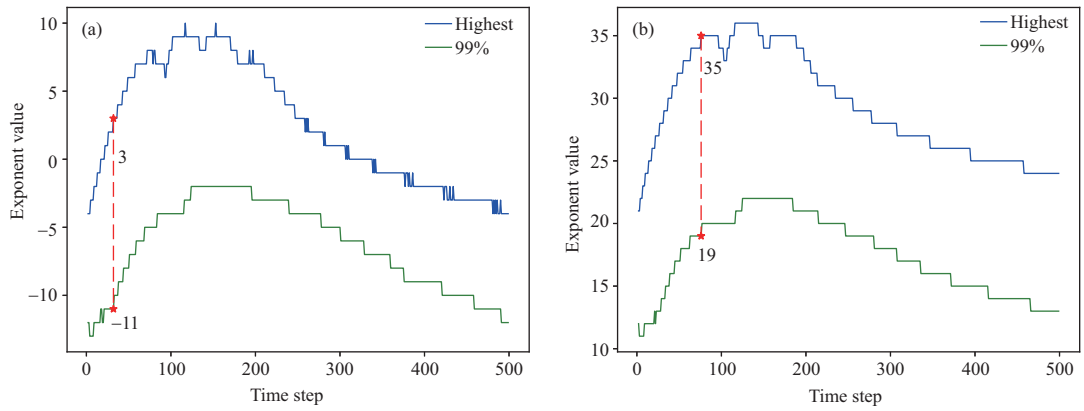


**Figure 4** (Color online) The change of exponents of the largest value of variables $W_v$ (a) and $W_s$ (b), and the value that serves as the dividing point of a cumulative contribution above 99% of the total energy, for the first 500 time steps. The red dots denote the largest gap between the exponents of the two values.

Based on our analysis of the total energy at each given time step, we can then perform the analysis over multiple time steps. Figure 4 demonstrates the results for the $W_v$ and $W_s$ variables during the first 500 time steps of the simulation. The gap between the two lines corresponds to the region of exponent values that can represent over 99% of the signal, similar to the results in Figure 3, but presented as a long time series. The red dots picks the largest gap over time. For the case of the $W_v$ variable, the largest range of exponent values needed for the coverage of 99% is from $-11$ to 3 ($N_A = 14$ bits to represent the range).

Besides the gap between the largest and smallest values for the 99% representation, the trend of the lines also correspond well to the energy injection, propagation, and dissipation processes of the seismic event that we try to capture. The lines would serve as a guide for us to derive the customized scaling scheme for the fixed-point design, as detailed in Section 5.

### 4.5 Precision analysis

After identifying the number of bits needed to represent the range, we identify the number of extra precision bits needed to represent each variable with sufficient accuracy.

As noted in previous studies on precision analysis [24], identifying the optimal word length configuration for a design under the resource cost constraint is an NP-hard problem that requires huge amount of time to achieve a close-to-optimal result. To achieve a feasible fixed-point design for a practical and complex scientific computing application, we take a more engineering-oriented approach.

The first strategy is divide and conquer. Based on the characteristics of different variables, we group them into six main categories: velocity components, stress components, derivatives of velocity, derivatives of stress, curvilinear grid transformation coefficients, and elastic parameters, as shown in Figure 5. For

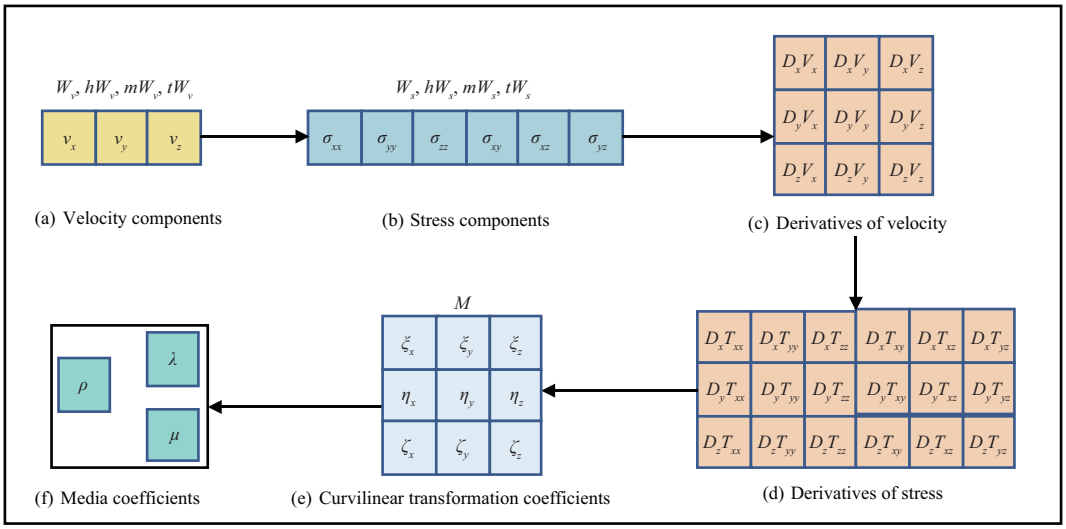


**Figure 5** (Color online) The main categories of variables that are put into different groups. Each group of variables employ the same type of representation precision. We follow the sequence of the arrows to replace each group of variable from full precision (32-bit floating-point) to the optimized fixed-point precision one by one.

each group of variables, we employ the same type of representation format, i.e., the same kind of precision.

The second strategy is a systematic refinement approach that replace floating-point arithmetic with fixed-point arithmetic. Following a basic sequence from input variables to intermediate variables (also shown as the arrows in Figure 5), we gradually switch each group of variables from 32-bit floating-point numbers to fixed-point numbers. We gradually increase the number of extra precision bits from zero (the number of range bits needed is decided in the range analysis part), and evaluate the accuracy of the specific precision using the computed coefficient of determination ($R^2$), which is computed as the sum of square of the errors of the variable when comparing between a reduced-precision (the current configuration of precision bits) simulation and a full-precision (32-bit floating-point) simulation. The number of precision bits increases until the $R^2$ values of the first 200 time steps are all above the designated threshold (98% or 90%, depending on the accuracy expected in the corresponding application).

Through a number of simulations following both the 'divide and conquer' and 'systematic refinement' strategies, we can then derive the acceptable precision bit configurations for all the different groups of variables.

## 5 A complete fixed-point dataflow engine design

### 5.1 The dataflow engine design

The traditional control flow processors, like CPU, GPU, and Intel Xeon Phi, have similar patterns to execute an application: a series of instructions loaded from memory to processors and then after fetch-decode cycle processors to perform computing or reading and writing data from or to memory. To increase the executing efficiency, an HPC processor often has multiple cores, each of which equipped with multi-level caches and vectorization computing units. For example, the SW26010 processor [11] consists of 256 computing cores, each of which runs at a working frequency of 1.45 GHz and has an on-chip 64 KB scratch-pad memory and a 256-bit vectorization computing unit, which indicate that its performance is based on the high working frequency and data parallelism.

In a reconfigurable dataflow engine, we reconfigure a circuit with thousands of computing units, which support various computing operations and data choreography. The input data stream moves from memory into the dataflow engine and then moves from one computing unit to another computing unit in a cycle, and after a number of cycles, the output data stream contains the results of our algorithm. In such a processing pipeline, the dataflow engine can achieve parallel performance through significant concurrent data parallelism powered by computing units with customized number representations.

Figure 6 illustrates the architecture of our optimized fixed-point design for earthquake simulations. To achieve an enhanced performance through the efficient utilization of memory bandwidth, we load all the

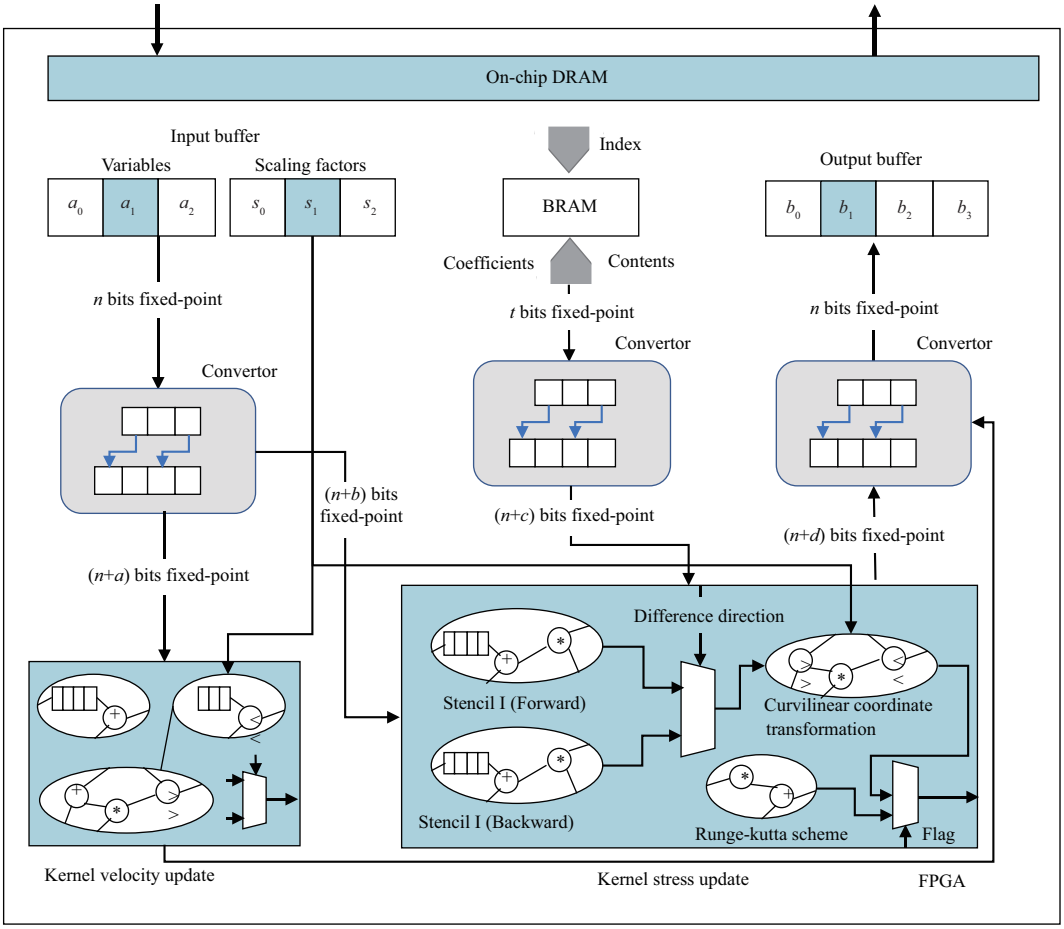


**Figure 6** (Color online) The general architecture of our fixed-point design for earthquake simulation.

arrays into the onboard memory through the peripheral component interconnect express (PCIe) interface at the initialization stage. We then stream the data from onboard dynamic random access memory (DRAM) to drive the computation. The main hardware resources are mapped into two major modules of the design: the velocity update module and the stress update module. To enable dynamic scaling of the range and the format bridging between different modules, we have a number of scaling units and conversion units at both the input/output ports and the intermediate connecting ports. According to the range analysis results, as shown in Figure 4, we derive a predefined scaling scheme and load the scaling factors for each time step into the onboard mapped registers. The streaming circuit can then read the scaling factor that corresponds to the current time step and scale the variables accordingly.

Convertors in Figure 6 are for fixed-point variables with different bitwidths and different dynamic scalers. These convertors can be divided into two categories: (1) in the case that the dynamic scaler of the output variable is larger than the input variable, the input variable and dynamic scalers are first expanded as wide-bitwidth variables. The reason for this is that the dynamic scalers may sometimes be very large. Then, the input variable is multiplied by the quotient of dynamic scalers of the output and input variables. Finally, the result is cast to the output variable, where the higher bits of the result are assigned to the output variable. (2) In the case that the dynamic scaler of the output variable is smaller than the input variable, it is similar, although the input variable should be divided by the quotient of dynamic scalers of the input and output variables. In our design, the change of the data range of variables during iterations can result in a reversed size relationship between dynamic scalers in some cases, and both kinds of convertor should be used at the same time, and finally, a multiplexer is used to choose the correct one. To simplify the operations of multiplication and division, the quotient of dynamic scalers can be precomputed, and the left-shift and right-shift operations can optimize multiplication and division.

**Table 1** Bitwidth configuration of major variables in the fixed-point design

| Variable | $W_v$ | $W_s$ | $M$ | $\rho$ | $D_xV_x$ | $D_xT_{xx}$ |
|---|---|---|---|---|---|---|
| Bitwidth (ideal case) | 21 | 22 | 18 | 15 | 20 | 20 |
| Bitwidth (Wenchuan case) | 18 | 19 | 18 | 15 | 17 | 17 |

### 5.2 Resource-oriented optimization

In a reconfigurable platform, resources on board are usually limited, which makes our 3D elastic earthquake simulation algorithm with complex computing logic become a resource-constraint problem. Thus, reducing resource utilization on board would gain some benefits in our design: (1) Due to features of dataflow in reconfigurable platforms, reducing some of the hardware resources can increase parallelism and derive a more deep-pipelined design. (2) After optimizing the resource utilization, the objective is more likely to be achieved in placement and routing stages with a higher required stream frequency. (3) The reduced-bitwidth variables can relax the requirement for memory bandwidth in multiple aspects, such as bandwidth between computing modules and onboard DRAM between FPGA and CPU, as well as between CPUs.

To reduce the resources used to build our design, we perform some optimization techniques. First, there are some finite difference coefficients, which are frequently used in our algorithm. To reduce the repeating data transfer from DRAM, we can store the coefficients in the ROM. Second, as the division operations will use many resources, we can pre-calculate the inversion of some numbers in CPU and then transform the division operations to multiplication operations in the dataflow engine. Third, to reduce the number of computing units, we can exchange the computational order of our algorithm to reuse some duplicate computational results but with the acceptance error.

To reduce the bitwidth and complexity of computing units, we apply our complete fixed-point scheme. To achieve over 99% representation of the dynamic range over time and over 98% representation of the distribution, as shown in Table 1, for the ideal case, we need roughly 21 bits for velocity variables, 22 bits for stress variables, 18 bits for grid transformation coefficients, 15 bits for the elastic parameters, and 20 bits for most intermediate variables.

Similar to the ideal case, for the real 3D velocity model of the Wenchuan earthquake region, we perform the same kind of range and precision analysis to determine the most suitable configuration of the bitwidth. Interestingly, the result showed that the actual Wenchuan case, with a more complex surface topography and velocity model, requires a smaller number of bits to represent the different variables. As shown in Table 1, to achieve over 99% representation of the dynamic range over time and over 98% representation of the distribution, we require 18 bits for velocity variables, 19 bits for stress variables, 18 bits for grid transformation coefficients, 15 bits for the elastic parameters, and 17 bits for most intermediate variables.

We suppose that the main reason for a lower precision requirement in the Wenchuan case is a result of the different types of seismic source we used in the simulations. In the ideal case, we use a point source for simplicity. In the case of the Wenchuan earthquake, a finite fault model that consists of many point sources at different locations is used for the simulation. In the process of balancing the accuracy of the maximum and minimum values of the wavefield at each time step, the point source becomes more difficult and demands more bits.

Although the major variables in fixed-point formats can reduce the bitwidth of the registers and computing units, the intermediate variables during computing may still need extra bitwidth due to the multiplication or division operations between variables and scaling factors. Therefore, we must further analyze operations involved with the multiplication of variables and scaling factors. For example, for the stress update module, there are updating operations involving variables $W_v$, $W_s$, $M$, $\lambda$, $\mu$, and their corresponding four scaling factors. In the naive design, the intermediate variables require approximately 56 bits to accommodate all the informative bits. After analyzing the range change between variables, we can reduce the number of intermediate bits to 38 by adjusting the computational order of the variables and scaling factors, which maintains a precision similar with that of the naive design.

Considering all these optimizations, we can refine the resource utilization compared with the original full-precision floating-point design, which is shown in Table 2. For the full-precision floating-point design, the percentage of logic utilization is 47.32%. However, it consists of 32-bit complicated floating-point computing units, and the design cannot meet timing even in a lower stream frequency of 100 MHz. In a higher frequency, more severe timing issues occur, and the bandwidth becomes a challenge. For the optimized complete fixed-point design, with 23.35% of the resource utilization and simper reduced-bitwidth

**Table 2** Resources comparison of the full-precision floating-point design and the optimized fully fixed-point design

| Design | Logic utilization (%) | DSP blocks (%) | BRAM (%) |
|---|---|---|---|
| Original full-precision floating-point design | 47.32 | 19.49 | 23.63 |
| Optimized complete fixed-point design | 23.25 | 18.64 | 24.86 |

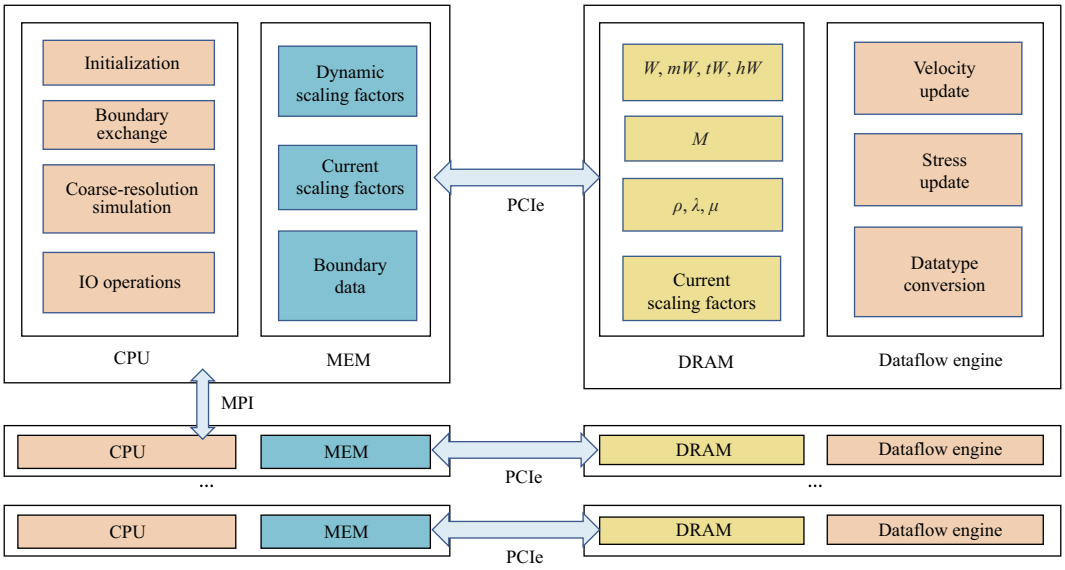


**Figure 7** (Color online) The task mapping for CPU + FPGA hybrid system.

fixed-point computing units, we can increase the stream frequency to 300 MHz, which is equivalent to three computing cores running on a stream frequency of 100 MHz.

### 5.3 CPU-FPGA hybrid system

Using such a design, we implement a CPU-FPGA hybrid system for a 3D earthquake simulation program that can use both a single FPGA and multiple FPGAs with a curvilinear coordinate transformation and traction image method to respond to complex surface topography. The task mapping for CPU and FPGA is shown in Figure 7. We map the initialization, boundary exchange, coarse-resolution simulation, and IO operations to CPU, which can manage and process some simple and tedious tasks. We perform range and precision analysis toward the results from a coarse-resolution run and store the dynamic scaling factors and the updated boundary in the memory. To fully utilize the memory bandwidth, we store the 3D arrays, such as $W$, $mW$, $tW$, $hW$, $M$, $\rho$, $\lambda$, $\mu$ and the current scaling factors in the onboard DRAM. Notably, the current scaling factors and boundary data can be transferred between the memory and the DRAM by PCIe. In the dataflow engine, we map the most density computing tasks, such as velocity update, stress update, and datatype conversion.

By adopting the optimized fixed-point representation, we can use one Xilinx Ultrascale+ VU9P FPGA to implement the deep-pipelined core, achieving parallel performance from thousands of concurrent instruction parallelism. Moreover, with the bandwidth support of onboard DRAM running on a memory frequency of 1066 MHz, we can boost the stream frequency of our design from 100 to 300 MHz, which is equivalent to three computing cores running on 100 MHz.

To target our implementation for large-scale runs in the future, we also implement a partitioning scheme to support multiple FPGAs. We partition the entire space into blocks and assign different blocks to different FPGAs. The boundary regions are communicated among different FPGAs through the MPI interfaces, which are shown in Figure 7. Each FPGA must send the boundary data to the CPU through PCI-e interfaces and then communicate to the other CPUs through the MPI send and receive function calls. In the Wenchuan earthquake simulation case, the time for computing on FPGA is 2.52 s, and the time for MPI communication between CPUs is 0.65 s. The time for boundary transfer from CPU to FPGA is 0.28 s, and the time for boundary transfer from FPGA to CPU is 0.17 s. Thus, the time for computing on FPGA requires about 81.8%, while the communication between FPGAs requires about 19.2%, which demonstrates that the FPGA is primarily devoted to effective computing.

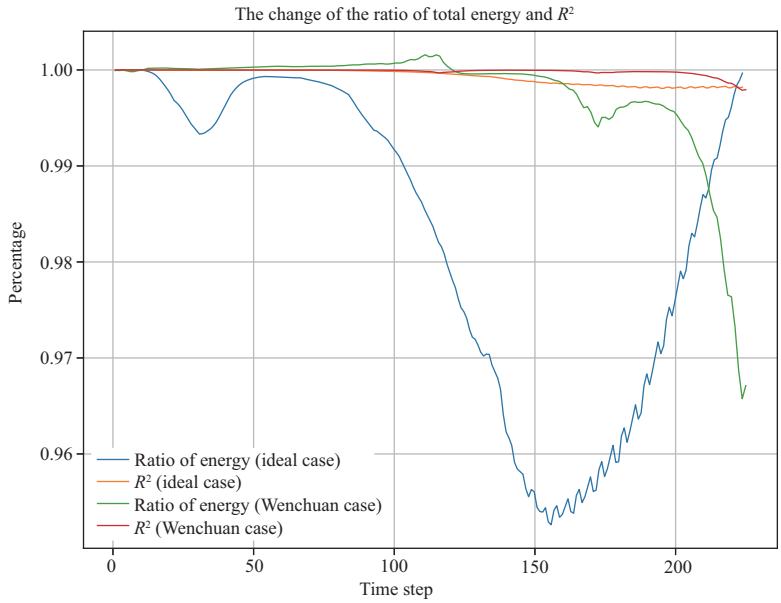


**Figure 8** (Color online) Verification of the total energy and the coefficient of determination after projecting the analysis result of a coarse-resolution simulation to a target-resolution design.

## 6 Results and discussions

### 6.1 Accuracy verification of the design

In our approach, the extensive range and precision analysis occur in the coarse-resolution simulation. We then derived the fixed-point design from handling the fine-resolution target scenario.

Therefore, after achieving the fixed-point design, we must verify whether it still maintains an acceptable level of accuracy for both representing the correct amount of total energy and representing the correct energy distribution.

As shown in Figure 8, we evaluate the representation of the total energy by calculating the ratio between the total energy simulated using fixed-point numbers and the total energy computed using single-precision floating-point numbers. We then evaluate the representation of the energy distribution by computing the coefficient of determination and between the fixed-point and the floating-point representation. In the analysis process, we set the target ratio to be 99% and the target $R^2$ to be 98%. The curves of both the ideal and the Wenchuan cases demonstrate that the ratio of total energy is well maintained above 95% and $R^2$ is well preserved above 99%, which demonstrates that the analysis in the coarse-resolution case can serve as decent guidance for the design in a fine-resolution scenario. There is an interesting fluctuation in the ratio of total energy through the different time steps, especially for the Wenchuan case, which could be due to either the delayed effect of the scaling scheme or the internal physics phenomena.

From another point of view, the formula for the coefficient of determination (marked as $R^2$) is: $R^2 = 1 - \mathrm{SSE}/\mathrm{SST}$, where SSE is often used in error analysis. Moreover, Figure 8 demonstrates the accuracy loss of the system. For the ideal case, we can see up to approximately 5% energy loss and 1% accuracy loss throughout the simulation. For the Wenchuan case, we can observe up to about 4% energy loss and 1% accuracy loss throughout the simulation.

### 6.2 Simulation and performance results

For the Wenchuan case, we compare the results of our complete fixed-point design with the full-precision result, as shown in Figure 9. The dynamic rupture source of the Wenchuan earthquake is used in the seismic wave simulation. There is no visible deviation between the results calculated by our selected fixed-point design (Figure 9, second column) and the full-precision (Figure 9, first column). This indicates that our fixed-point design achieves high-precision seismic wave simulations. We can relax the acceptable accuracy (with a relaxed coefficient of determination threshold of 90% and 70%) to further reduce the number of bits (Figure 9, third and fourth column, velocity bitwidth down to 15 or 12 bits), and the

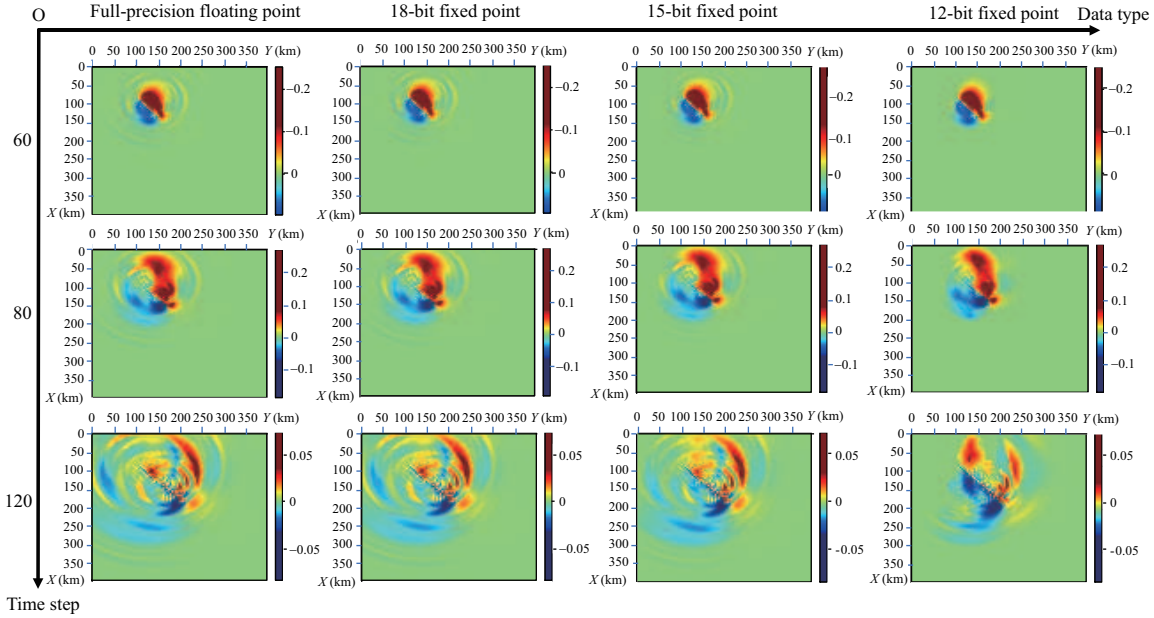


**Figure 9** (Color online) Comparison of seismic wave snapshots (velocity component on the $x$-axis) calculated by the full-precision and our complete fixed-point design for the Wenchuan earthquake simulation. The traditional full-precision arithmetic calculates the results in the first column, and the results in the second, third, and fourth columns are calculated by our complete fixed-point design using 18 (taking velocity as the principal indicator), 15, and 12 bits, respectively. The results of the first, second, and third row depict the snapshots at the 60th, 80th, and 120th time step, respectively.

**Table 3** Overall performance on the CPU and the CPU-FPGA platforms

| Platform | Performance (grid points/s) | Speed-up | Power (W) | Performance per watt (PPW) (grid points/(W·s)) | PPW speed-up |
|---|---|---|---|---|---|
| Intel Xeon E5-2643 6-core CPU | 67.6k | 1x | 127.6 | 529.8 | 1x |
| Intel Xeon Gold 6154 18-core CPU | 175.8k | 2.6x | 205.3 | 856.3 | 1.61x |
| 260-core Sunway TaihuLight node | 1.10M | 16.3x | 312.5 | 3.5k | 6.6x |
| MaxWorkstation | 1.36M | 20.1x | 82.9 | 16.4k | 30.9x |
| MaxNode | 2.31M | 34.2x | 175.8 | 13.1k | 24.7x |

accuracy of the results also decreases. The results using 12 bits deviate significantly from the exact solution at the 120th time step since the error accumulates over time. Notably, even among the worst results (Figure 9, fourth column), the maximum value in the wave field was less different from the exact result. The peaks of strong ground motion are often a focus of seismic disaster analysis, and applications such as earthquake disaster analysis do not require extremely high accuracy in the waveforms. Hence, it is useful to quickly calculate a relatively accurate strong ground motion result.

Table 3 demonstrates the performance results of our FPGA-based design as compared with the Intel CPU version and Sunway many-core processor version (SW26010 260-core processor). A detailed description of the Sunway architecture and its design of elastic earthquake simulation can be found in [20]. A single SW26010 processor provides a peak performance of over 3 TFlops and a maximum memory bandwidth of 136 GB/s.

The performance of a hybrid CPU-FPGA algorithm on MaxWorkstation with one Ultrascale+ VU9P demonstrated a 20.1-times speedup over one Intel Xeon E5-2643 6-core CPU, 7.73 times speedup over an Intel Xeon Gold 6154 18-core CPU, and a 1.24-times speedup over one 260-core Sunway TaihuLight supercomputer node. With MaxNode with two Ultrascale+ VU9P, we can gain a 34.2-times speedup over one Intel Xeon E5-2643 6-core CPU and a 13.1-times speedup over an Intel Xeon Gold 6154 18-core CPU, which is also 2.10 times faster than one 260-core Sunway TaihuLight node.

As for the performance per watt (aka power efficiency), our CPU-FPGA hybrid algorithm with two FPGAs is about 24.7, 15.3, and 3.72 times more efficient than the Intel Xeon E5-2643 6-core CPU, the Intel Xeon Gold 6154 18-core CPU, and the 260-core Sunway TaihuLight node, respectively.

## 7 Conclusion

In this paper, we target elastic earthquake simulations, which is one of the most complicated applications of leading supercomputers. We demonstrate our methods of achieving a highly efficient fixed-point design that can provide an acceptable level of accuracy as well as decent improvement in computational efficiency. Through a careful analysis of the numerical behavior, we manage to achieve a fixed-point design that employs 20-bit numbers to achieve an acceptable level of accuracy, while providing equivalent performance to 34.2 Intel Xeon E5-2643 6-core CPUs, or 13.1 Intel Xeon Gold 6154 18-core CPUs or 2.10 Sunway 260-core processors. We believe that such analysis and optimization methods are key to applying FPGA technologies to a wide range of numerical simulations for scientific applications.

**Acknowledgements** This work was supported by National Natural Science Foundation of China (Grant Nos. 51761135015, U1839206), Center for High Performance Computing and System Simulation, Pilot National Laboratory for Marine Science and Technology (Qingdao), Intel, Maxeler, Xilinx, and the United Kingdom EPSRC (Grant Nos. EP/L016796/1, EP/N031768/1, EP/P010040/1, EP/S030069/1). We would also like to thank Dr. Bastiaan Willem Kwaadgras, Dr. Pavel Burovskiy from Maxeler Technologies, Dr. Wenqiang ZHANG from University of Science and Technology of China, and Prof. Wei ZHANG, Prof. Xiaofei CHEN from Southern University of Science and Technology for their support.

**References**

1 Thornton J E. The CDC 6600 project. IEEE Ann Hist Comput, 1980, 2: 338–348
2 Hey T, Tansley S, Tolle K, et al. The Fourth Paradigm: Data-Intensive Scientific Discovery. Redmond: Microsoft Research, 2009
3 Komatitsch D, Tsuboi S, Ji C, et al. A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the earth simulator. In: Proceedings of ACM/IEEE Conference on Supercomputing, 2003
4 Rudi J, Malossi A C I, Isaac T, et al. An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, 2015
5 Shingu S, Takahara H, Fuchigami H, et al. A 26.58 TFlops global atmospheric simulation with the spectral transform method on the earth simulator. In: Proceedings of ACM/IEEE Conference on Supercomputing, 2002
6 Ishiyama T, Nitadori K, Makino J. 4.45 PFlops astrophysical n-body simulation on k computer — the gravitational trillion-body problem. In: Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis, 2012
7 Habib S, Morozov V, Finkel H, et al. The universe at extreme scale: multi-petaflop sky simulation on the BG/Q. In: Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis, 2012
8 Shimokawabe T, Aoki T, Takaki T, et al. Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, 2011
9 Yang X J, Liao X K, Lu K, et al. The TianHe-1A supercomputer: its hardware and software. J Comput Sci Technol, 2011, 26: 344–351
10 Liao X K, Xiao L Q, Yang C Q, et al. MilkyWay-2 supercomputer: system and application. Front Comput Sci, 2014, 8: 345–356
11 Fu H H, Liao J F, Yang J Z, et al. The Sunway TaihuLight supercomputer: system and applications. Sci China Inf Sci, 2016, 59: 072001
12 Shalf J, Quinlan D, Janssen C. Rethinking hardware-software codesign for exascale systems. Computer, 2011, 44: 22–30
13 Dosanjh S, Barrett R, Heroux M, et al. Achieving exascale computing through hardware/software co-design. In: Proceedings of European MPI Users' Group Meeting, 2011. 5–7
14 Dosanjh S S, Barrett R F, Doerfler D W, et al. Exascale design space exploration and co-design. Future Generation Comput Syst, 2014, 30: 46–58
15 Maechling P, Deelman E, Zhao L, et al. SCEC cybershake workflows — automating probabilistic seismic hazard analysis calculations. In: Proceedings of Workflows for e-Science, 2007. 143–163
16 Komatitsch D, Tsuboi S, Ji C, et al. A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the earth simulator. In: Proceedings of ACM/IEEE Conference on Supercomputing, 2003
17 Cui Y, Olsen K B, Jordan T H, et al. Scalable earthquake simulation on petascale supercomputers. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2010
18 Cui Y, Poyraz E, Olsen K B, et al. Physics-based seismic hazard analysis on petascale heterogeneous supercomputers. In: Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis, 2013
19 Fu H H, Yin W W, Yang G W, et al. 18.9-PFlops nonlinear earthquake simulation on Sunway TaihuLight: enabling depiction of 18-HZ and 8-meter scenarios. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, 2017
20 Chen B W, Fu H H, Wei Y W, et al. Simulating the Wenchuan earthquake with accurate surface topography on Sunway TaihuLight. In: Proceedings of International Conference for High Performance Computing, Networking, Storage, and Analysis, 2018
21 Benedetti A, Perona P. Bit-width optimization for configurable DSP's by multi-interval analysis. In: Proceedings of Conference Record of the 34th Asilomar Conference on Signals, Systems and Computers, 2000. 355–359
22 Wadekar S A, Parker A C. Accuracy sensitive word-length selection for algorithm optimization. In: Proceedings of International Conference on Computer Design, 1998. 54–61
23 Lee D U, Gaffar A A, Mencer O, et al. Optimizing hardware function evaluation. IEEE Trans Comput, 2005, 54: 1520–1531
24 Lee D U, Gaffar A A, Cheung R C C, et al. Accuracy-guaranteed bit-width optimization. IEEE Trans Comput-Aided Des Integr Circ Syst, 2006, 25: 1990–2000
25 Fu H H, Osborne W, Clapp R G, et al. Accelerating seismic computations on FPGAs from the perspective of number representations. In: Proceedings of the 70th EAGE Conference and Exhibition Incorporating SPE EUROPEC 2008, 2008

26 Gan L, Fu H H, Luk W, et al. Accelerating solvers for global atmospheric equations through mixed-precision data flow engine. In: Proceedings of the 23rd International Conference on Field Programmable Logic and Applications, 2013

27 Chow G C, Kwok K, Luk W, et al. Mixed precision processing in reconfigurable systems. In: Proceedings of the 19th Annual International Symposium on Field-Programmable Custom Computing Machines, 2011. 17–24

28 Chow G C T, Tse A H T, Jin Q, et al. A mixed precision Monte Carlo methodology for reconfigurable accelerator systems. In: Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2012. 57–66

29 He C, Lu M, Sun C. Accelerating seismic migration using FPGA-based coprocessor platform. In: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2004. 207–216

30 Pell O, Clapp R G. Accelerating subsurface offset gathers for 3D seismic applications using FPGAs. In: Proceedings of SEG Technical Program Expanded Abstracts, 2007. 2383–2387

31 Medeiros V, Barros A, Silva-Filho A, et al. High performance implementation of RTM seismic modeling on FPGAs: architecture, arithmetic and power issues. In: Proceedings of High-Performance Computing Using FPGAs, 2013. 305–334

32 Bittencourt J C, Oliveira W L, Nascimento A, et al. Performance and energy efficiency analysis of reverse time migration on a FPGA platform. In: Proceedings of IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC), 2019. 50–58

33 Ellsworth W L. Earthquake history, 1769-1989. United States Geological Survey, Professional Paper (USA), 1990. http://geologycafe.com/california/pp1515/chapter6.html

34 Washburn Z, Arrowsmith J R, Forman S L, et al. Late Holocene earthquake history of the central Altyn Tagh fault, China. Geology, 2001, 29: 1051–1054

35 Zhang W, Chen X F. Traction image method for irregular free surface boundaries in finite difference seismic wave simulation. Geophys J Int, 2006, 167: 337–353

36 Zhang W, Zhang Z G, Chen X F. Three-dimensional elastic wave numerical modelling in the presence of surface topography by a collocated-grid finite-difference method on curvilinear grids. Geophys J Int, 2012, 190: 358–378

37 Butcher J C, Butcher J. The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods. Hoboken: Wiley, 1987

38 Pell O, Mencer O, Tsoi K H, et al. Maximum performance computing with dataflow engines. In: Proceedings of High-performance Computing Using FPGAs, 2013. 747–774

39 Becker J J, Sandwell D T, Smith W H F, et al. Global bathymetry and elevation data at 30 Arc seconds resolution: SRTM30_PLUS. Mar Geodesy, 2009, 32: 355–371

40 Zhang Z, Zhang W, Chen X. Dynamic rupture simulations of the 2008 Mw 7.9 Wenchuan earthquake by the curved grid finite-difference method. J Geophys Res Solid Earth, 2019, 124: 10565–10582