Tian, W., Sevilla, T., Zuo, W. “A Systematic Evaluation of Accelerating Indoor Airflow Simulations Using Cross Platform Parallel Computing,” *Journal of Building Performance Simulation*, 10(3), pp. 243-255, 2017.

A Systematic Evaluation of Accelerating Indoor Airflow Simulations Using Cross Platform Parallel Computing

Wei Tian[1], Thomas Alonso Sevilla[1], Wangda Zuo[1,*]

[1]*Department of Civil, Architectural and Environmental Engineering, University of Miami, Coral Gables, FL 33146, USA*

Correspondence: Wangda Zuo, 1251 Memorial Drive, #EB320, Coral Gables, FL 33146, w.zuo@miami.edu

A Systematic Evaluation of Accelerating Indoor Airflow Simulations Using Cross Platform Parallel Computing

With new advances in computer hardware and software, users now have widespread accessibility to multicore devices inside personal computers making it feasible for fast indoor airflow simulations. Some exciting preliminary results of a cross-platform parallel computing framework OpenCL using specific hardware were reported. However, those results are largely based on two hypotheses: 1. OpenCL code on all devices will generate the same results; 2. On the same device, running in parallel with multiple processors will be faster than running in sequential with a single processor. This study attempted to evaluate these two hypotheses by systematically studying the accuracy and computing speed of OpenCL for indoor airflow simulations. A Fast Fluid Dynamics (FFD) code was selected as an exemplar indoor airflow simulation program. To compare the cross-platform ability of OpenCL, the evaluation was performed using four Graphics Processing Units (GPUs) and five Central Processing Units (CPUs) from three manufacturers, with different degrees of computing capability and mounted on two operating systems. The test subjects were evaluated using four case studies consisting of various indoor airflows. A sequential FFD code programmed in C and a Computational Fluid Dynamics (CFD) program were first used to perform the case studies and generate numerical benchmarks. The comparison of the numerical simulation results with experimental data showed that CFD and FFD can predict the studied flows with averaged relative errors of 9.99% and 11.30%, respectively. Afterwards, the accuracy and speedup of the OpenCL code was compared with numerical benchmarks. Although the OpenCL code on the CPUs generated identical numerical results, the OpenCL results from the GPUs were slightly dissimilar. This is likely due to varying interpretations, by the manufactures, of an IEEE standard. Depending on the hardware, the speedups of the OpenCL code varied from 0.7-4.2 times on the CPUs and 5.1-129.3 times on the GPUs. The slowdown of computing speed happened when running OpenCL on a two-core CPU in a Windows Operating System using the Boot Camp on a Mac computer. Finally, a separate study on the relationship between speedup and global work size showed that a speedup of 1139 can be achieved when using an AMD FirePro W8100 GPU.

Keywords: FFD, OpenCL, Parallel Computing, Indoor Airflow Simulation

Nomenclature

OpenCL: Open Computing Language

CPU: Central Processing Unit

GPU: Graphics Processing Unit

NVIDIA: A GPU manufacturer

AMD: A CPU and GPU manufacturer

Intel: A semiconductor chip marker

CUDA: Parallel computing platform created by *NVIDA*

OpenCL_FFD: The FFD model implemented in the *OpenCL*

C_FFD: The FFD model implemented in the C language

C_Reference: The results computed by the *C_FFD*

U_i: Velocity vectors, *m/s*

$\boldsymbol{\phi}$: Scalar variable, such as temperature and species concentration;

P : Pressure, *pa*

ρ : Density, *kg/m*^{3}

x_i : Spatial directions, *m*

t : Time, *s*

$\frac{1}{\rho}f_i$: External force vector, *m/s*^{2}

S : Source term for scalar variables

ν : Kinematic viscosity of air, *m*^{2}*/s*

ζ: Transport coefficient, *m*^{2}*/s*

1. **Introduction**

Indoor air quality in buildings relies heavily on ventilation design (Godish and Spengler 1996). Using computer simulation models one can evaluate the performance of the ventilation system by looking into some key parameters such as air velocity, temperature, and contaminant concentration. As reviewed by Chen (2009), there are primarily three computer simulation models for indoor ventilation prediction: Multi-zone (Axley 2007), Zonal (Megri and Haghighat 2007) and Computational Fluid Dynamics (CFD) (Ladeinde and Nearon 1997) models. Among them, CFD has become the most popular due to its detailed prediction of key parameters. However, one challenge of applying the CFD model for indoor ventilation predictions is the long computing time (Zhai et al. 2002).

In order to reduce the computing time, one can perform the CFD simulation using supercomputers or cloud computing services (Gropp et al. 2001). However, this method is usually costly and may not be readily available. An alternate solution is to use multicore devices widely available in modern personal computers (Zuo and Chen 2009a, 2010a; Corrigan et al. 2011; Gorobets et al. 2013a; Gorobets et al. 2013b; Wang et al. 2011). These devices include GPUs, multi-core CPUs, Digital Signal Processors (DSPs), and other microprocessors. For instance, Zuo and Chen (2009b) sped up the CFD simulation 10-30 times by running it on a NVIDIA GeForce 8800 GTX GPU using CUDA (NVIDIA 2007).

However, CUDA only supports NVIDIA GPUs. A more appealing option is OpenCL, which supports GPUs, CPUs, Digital Signal Processors (DSPs), and other microprocessors from different manufacturers (Khronos 2012). Our literature review showed that there is only one indoor airflow simulation study using OpenCL (Wang et al. 2011). In their study, Wang et al. evaluated one Intel CPU and three NVIDIA GPUs on a Windows operating system using one case study. Despite providing speedup data, they did not provide sufficient validation of the OpenCL code in terms of result accuracy.

Although we would assume consistency in the numerical results across different hardware and operating systems using OpenCL, it remains critical to validate this assumption before adopting OpenCL for the indoor airflow simulation. Likewise, it is also interesting to observe how much speedup one can expect on different hardware. As a result, this study attempted to systematically evaluate the accuracy and speedup of cross-platform computing using OpenCL for indoor airflow simulations. In the investigation we selected five CPUs and four GPUs differing in types, ages, and manufacturers.

After implementing a Fast Fluid Dynamics (FFD) model using the OpenCL framework, we validated and evaluated both the FFD model and the OpenCL codes using four different cases that cover basic indoor airflows. These cases are largely based on the experiments done by Wang and Chen (2009), which include forced convection and mixed convection in enclosed spaces. We did not additionally study the natural convection since the modelling of mixed convection is comparably difficult to the modelling of natural convection. The FFD simulation of natural convection flow has been reported in previous studies (Zuo and Chen 2009c). By gradually increasing the complexity of the studied flow, we want to study if and how the flow complexity is going to affect the result consistency as well as the speedups. CFD simulations results were also presented as a comparison. Finally, after analysing the result consistency, speedup, and overall portability, we provided suggestions on using OpenCL for indoor airflow simulations.

2 Parallelization of FFD in OpenCL

In this section we introduce the procedure of parallelizing the FFD in OpenCL. FFD solves the same set of governing equations as CFD, however, with different numerical algorithms. FFD can provide a 30-times speedup compared to CFD with some compromises in accuracy (Zuo and Chen 2009c). To reduce the computing time, FFD used simplified numeric algorithm and used the numeric viscosity to replace the turbulent viscosity (Zuo et al. 2012). Thus, FFD is not as accurate as CFD and cannot catch detailed turbulent flow characteristic near the wall (Zuo and Chen 2009c) . Recently, FFD has been used for fast and informative indoor environment modelling. Some examples include natural ventilation in buildings (Jin et al. 2013), wind loading optimization for a tree-form surface (Chronis et al. 2011), ventilation in an office space (Jin and Chen 2015), coupled simulation of indoor environment and HVAC system (Zuo et al. 2014a, 2014b), and smoke dissipation in buildings (Zuo and Chen 2010b).

FFD solves the following governing equations: continuity, momentum, and balance equations for energy and species using a time-splitting method:

$$\frac{\partial U_i}{\partial x_i} = 0, \tag{1}$$

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial}{\partial x_j} U_i = \nu \frac{\partial^2}{\partial {x_j}^2} U_i - \frac{1}{\rho} \frac{\partial}{\partial x_i} P + \frac{1}{\rho} f_i, \tag{2}$$

$$\frac{\partial \phi}{\partial t} + U_j \frac{\partial}{\partial x_j} \phi = \zeta \frac{\partial^2}{\partial {x_j}^2} \phi + S, \tag{3}$$

where U_i and U_j are velocity vectors; $\boldsymbol{\phi}$ is scalar variable, such as temperature and species concentration; P is pressure; ρ is density; x_i is spatial directions and t is time; f_i

is an external force vector; S is source term; ν is kinematic viscosity of air; ζ is the transport coefficient. For the detailed procedure of FFD refer to the previous work (Zuo and Chen 2009c, 2010c).

The OpenCL implementation of FFD is shown in Figure 1. The program can be divided into a *host program* and its *kernels*. As stated in the OpenCL specification (Khronos 2012), the *host program* runs sequentially on the *Host* hardware (e.g. CPU) while the *kernels* run in parallel on the *device* hardware (e.g. GPU or other processors of the CPU). The entire implementation is a hybrid of C and OpenCL code. The C code is responsible for the main program structure while the OpenCL code is used to execute the *kernels.* The kernels are created based on the discretization of the governing equations introduced previously. These codes are then compiled in Mac OS X using Xcode 7.0 (Xcode 2012) and in Windows using Microsoft Visual Studio 2013 professional (Microsoft 2013) together with the AMD APP SDK (AMD 2013).

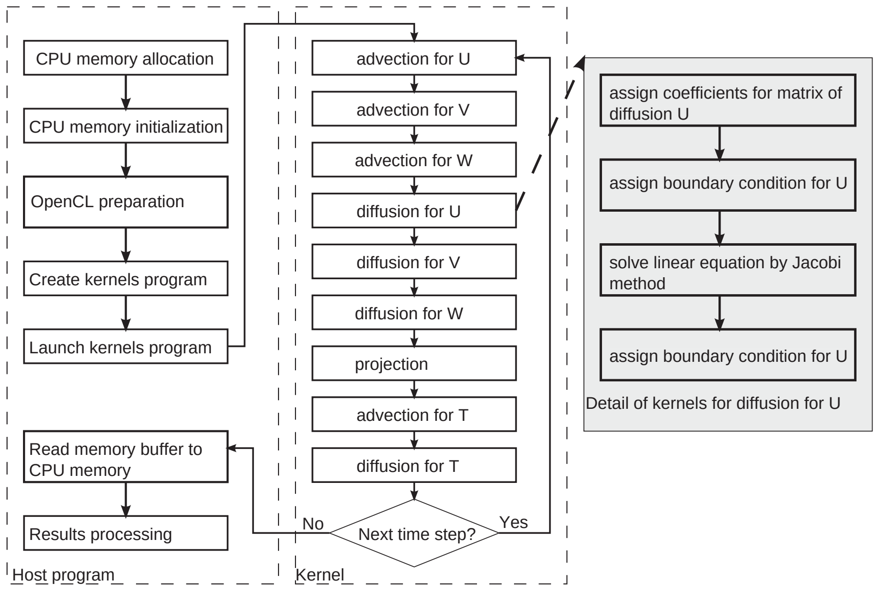


Figure 1 Structure of parallelized FFD using OpenCL

3. Numerical Experiment Settings

3.1. Hardware Device

As summarized in Table 1, four GPUs and five CPUs were selected. All five CPUs are manufactured by Intel while the GPUs are from AMD, NVIDIA, and Intel. Note that *CPU 1* and *CPU 5* are the same *Core i7 2620M* CPU installed on a MacBook Pro laptop. We named it as *CPU 1* under Windows 7 (running using Boot Camp) and *CPU 5* under Mac OS X. The peak performances for the hardware can be found online (NVIDIA 2012; AMD 2011, 2014; Intel 2012a, 2012b, 2012d, 2012c). In general, using double-precision floating point (DPFP) can reduce the round-up errors so that the calculations can be more accurate than those in single-precision floating point (SPFP). However, some devices in this study, e.g. AMD GPUs, did not support the DPFP in OpenCL environment. To carry out fair comparison, all simulations in this study were performed using SPFP.

Table 1 Technique details of devices used in this study

Device	Manufacturer	Model	Year	Base Frequency (MHz)	Peak Performance in SPFP (GFLOPS)	Peak Performance in DPSP (GFLOPS)	Memory Bandwidth (GB/s)	# of Cores	Operating System
CPU 1	Intel	Core i7 2620M	2011	2,700	N/A	43	21.3	2	Win 7
CPU 2	Intel	Xeon E5 1603	2012	2,800	N/A	90	31.4	4	Win 7
CPU 3	Intel	Core i7 4790	2014	3,600	N/A	230	25.6	4	Win 7
CPU 4	Intel	Core i5 3210M	2012	2,500	N/A	40	25.6	2	Mac OS X
CPU 5	Intel	Core i7 2620M	2011	2,700	N/A	43	21.3	2	Mac OS X
CPU 6	Intel	Core i7 3720 QM	2012	2,600	N/A	83	25.6	4	Mac OS X
GPU 1	Intel	HD Graphic 4000	2011	1,350	346*	N/A	25.6	16	Mac OS X
GPU 2	NVIDIA	GT 650M	2012	850	653	N/A	28.8	384	Mac OS X

GPU 3	AMD	FirePro V4900	2012	800	768	N/A	64	480	Win 7
GPU 4	AMD	FirePro W8100	2014	824	4,200	2,100	320	2,560	Win 7

*Estimated due to lack of official information

3.2. Case Description

To evaluate the performance of the parallelized FFD in OpenCL, we selected four different cases which cover the basic indoor airflow types. The benchmark data is available for all four cases.

3.2.1. Flow in a Lid-Driven Cavity

The flow in a lid-driven cavity is shown in Figure 2. The dimension is $1\ m \times 0.0233\ m \times 1\ m$. The top of the cavity is moving at a speed of *1 m/s*. The Reynolds number is set to be 400, based on the lid velocity, length of the cavity in the *X* direction, and kinematic viscosity. The benchmark data is available (Ghia et al. 1982). A non-uniform grid of $129 \times 3 \times 129$ was used for the simulation.

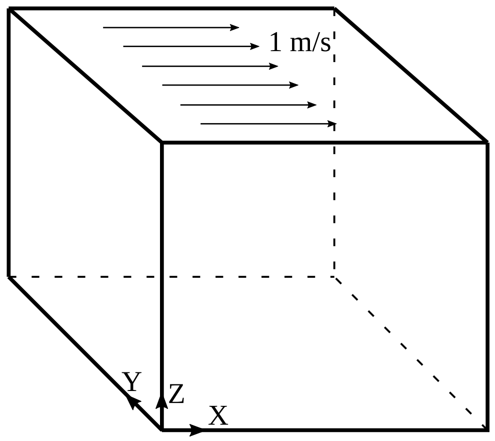


Figure 2 Sketch of Lid-Driven Cavity case

3.2.2. Forced Convection in an Empty Room

This case simulates an isothermal flow in an empty room (Wang and Chen 2009). The room size is $2.44\ m \times 2.44\ m \times 2.44\ m$ with other critical dimensions listed in Figure 3. The grid resolution is *40 × 40 × 40* and the inlet velocity is *1.36 m/s*. The experimental data (Wang and Chen 2009) is available at 10 different locations in Figure 4.

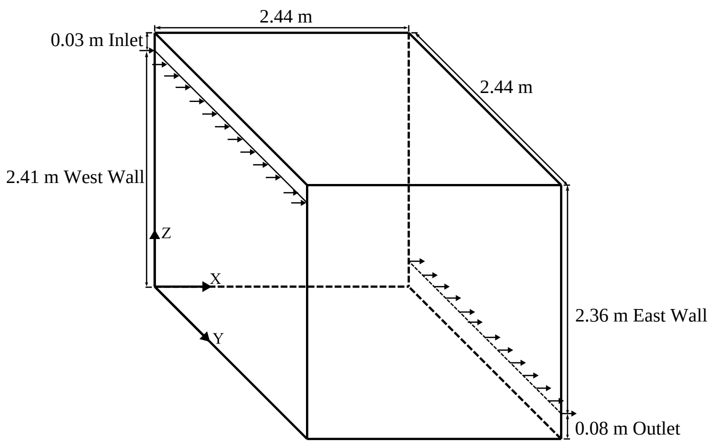


Figure 3 Schematic of the forced convection in an empty room

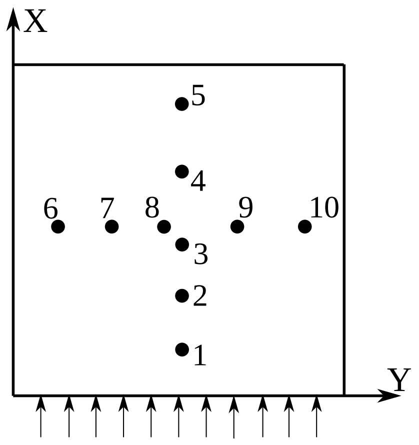


Figure 4 the distribution of ten locations with experimental data available

3.2.3. Forced Convection in a Room with a Box at Centre

Based on the previous case, this case further increases the flow complexity by adding an obstacle (*1.22 m × 1.22 m × 1.22 m*) in the middle of the room (Figure 5).

Again, detailed measurements at the locations described in Figure 4 are available (Wang and Chen 2009).

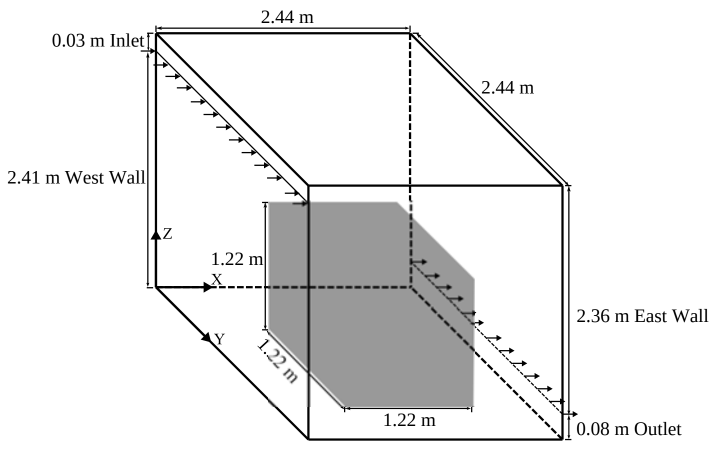


Figure 5 Schematic of the forced convection in an empty room with a box.

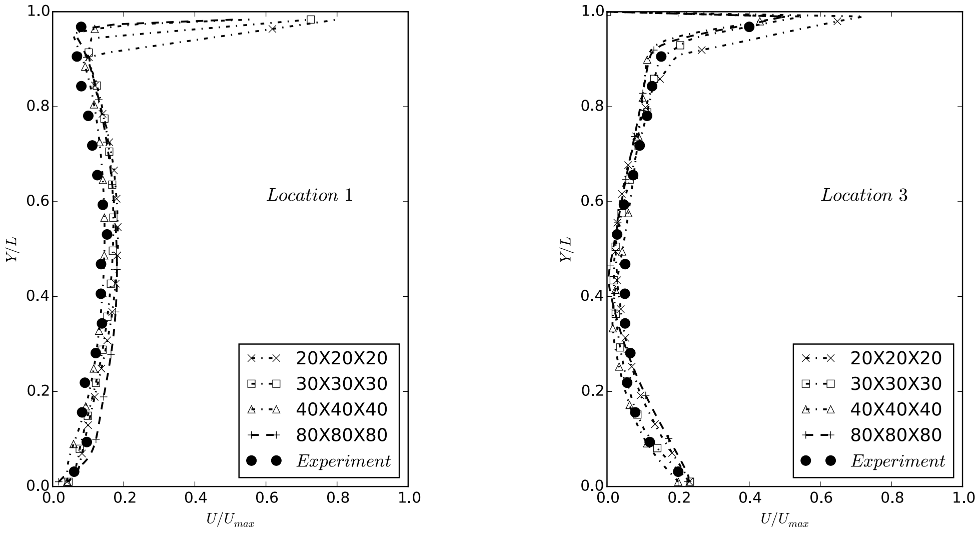
3.2.4. Mixed Convection in a room with a Box at the Centre

Based on the case described in 3.2.3, this case analyses the mixed convection in a space, such as an aircraft cabin, by adding a heat source to the obstacle and controlling the temperature of the walls (Wang and Chen 2009). In the experiment, the temperature of the box surface, inlet flow, ceiling, floor, and other walls is 36.7 *ºC*, 22.2 *ºC*. 25.8 *ºC*, 26.9 *ºC*, and 27.4 *ºC*, respectively. The experimental data is also available for locations in Figure 4 (Wang and Chen 2009).

3.2.5. CFD Simulation Setup

To validate the capability of FFD model, steady CFD simulations were performed for the above four cases using Fluent 16.1.0 (Fluent 2015) on cloud. A laminar flow model was applied for the lid-driven cavity flow. A RNG k-ε turbulence model (Yakhot and Orszag 1986) with the standard wall function was utilized for other cases as suggested by Chen (1995). The SIMPLE (Patankar and Spalding 1972) scheme was used to resolve pressure and velocity coupling. This study used standard scheme for pressure discretization and second-order upwind scheme for other equations discretization. The CFD simulation applied the same amount of grid as the FFD simulation although the grid distributions were different due to the wall function applied in the CFD simulation. The convergence criterion is set to have residual within 10^{-3} for velocity and 10^{-6} for temperature calculation. According to Wang and Chen (2009) the mesh grid used in the forced convection and mixed convection cases was fine enough to achieve grid independent results.

We conducted a grid dependency study using four sets of grids (*20 × 20 × 20*, *30 × 30 × 30*, *40 × 40 × 40*, and *80 × 80 × 80*) for the forced convection in an empty room in case 3.2.2*.* As shown the Figure 6, the result from grid of *80 × 80 × 80* is almost the same as that from *40 × 40 × 40*. Thus, a grid resolution of *40 × 40 × 40* is sufficiently fine to conduct the remaining studies. The typical resulted y^+ is around 30 with this grid resolution. Wang and Chen (2009) also drew similar conclusion in their study.



(a) (b)

Figure 6 Grid dependency study using forced convection in an empty room (case 3.2.2)

4. Analysis of Results

To provide clarity, we labelled the parallelized FFD in OpenCL as *OpenCL_FFD* and the sequential FFD in C as *C_FFD.* We employed two benchmarks including one from the experimental data and the other from the *C_FFD* on a single core of *CPU 2* (labelled as *C_Reference*).

4.2. Accuracy Evaluation

Figures 6-10 shows the comparison between *OpenCL_FFD* on the CPUs and GPUs with *C_Reference,* the CFD results, and the experimental data. Beginning with lid-driven cavity case which has a laminar flow, FFD (presented as *C_Reference*) is slightly worse than the CFD results which are perfectly aligned with the experimental data (Figure 7). By increasing the complexity of flows from laminar to turbulent, one can find in the forced convection the CFD still surpassed FFD in accuracy (Figure 8). FFD without the turbulence model was found to be deficient in capturing the flow features near the boundaries, especially the lower part of the profile. By further increasing the airflow features with an obstacle, one can see that even CFD with the turbulence model, cannot accurately predict the airflow near the boundaries (Figure 9). However, CFD still outperformed the FFD, which under-predicted the velocity magnitude, due to the omission of turbulent effect. Finally, in the mixed convection case (Figures 9 and 10) which was deemed the most complicated, interestingly the CFD predictions are closer with the experimental data than those in the forced convection (Figure 9). Again, FFD, due to lack of the turbulence models, predicted the trend with relatively poor accuracy.

To quantify the relative error between simulated and measured data, we employed an Euclidean norm estimator (Celebi et al. 2011):

$$\varepsilon = \frac{\sqrt{\sum_{1}^{N}\frac{(\hat{x}_i - x_i)^2}{{x_i}^2}}}{N}, \tag{4}$$

where $\hat{x}_i$ and x_i are the simulated and measured value at *i* point, respectively; *N* is the total number of points selected for comparison. This estimator has been used to calculate the overall discrepancies at certain points between the results (Wang and Zhai 2012; Wang et al. 2010).

Table 2 summarizes the relative errors of FFD and CFD simulations, CFD simulations have relative errors within the 20% for all locations and cases. The averaged relative error for all locations and cases is 9.99%. As an intermediate method, FFD is less accurate than CFD in most locations. Surprisingly, the averaged relative error of FFD for the studied cases is 11.30%, which only slightly larger than that of CFD. At Z=2.3622 m near the ceiling of the box at *Location1,* for example in case 3.2.2, CFD predicts a flow velocity of 0.235 m/s while the experimental data is 0.079 m/s. This point contributes approximately 40% to the total value of the estimator for the CFD. That’s why visual differences of the profiles in Figure 8 and Figure 9 are largely different from those shown in the estimator in Table 2.

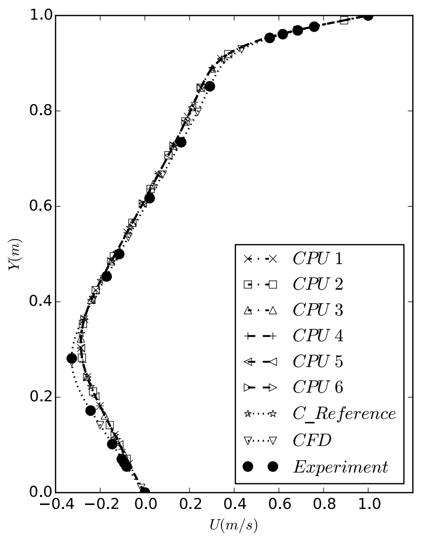
It is worth to mention that the CFD with the RNG *k*-*ε* turbulence model generated better results than that without the turbulence model, as was presented in previous research (Jin et al. 2012). This is consistent with the conclusion from Wang and Chen (2009) that RNG *k*-*ε* can generate overall good performance for cases 2-4.

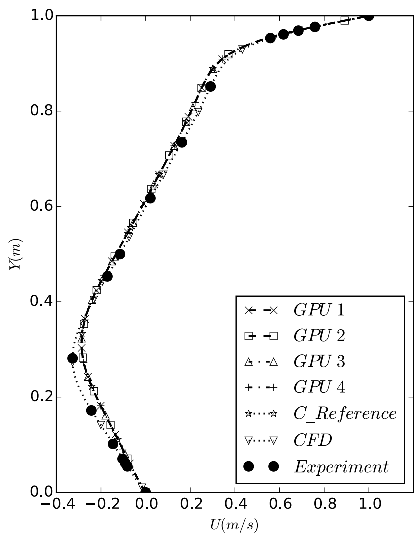
Table 2 Relative Difference of Velocity Profiles Predicted by CFD and FFD

Case	**Program**	**ε (%)**		
		Location1*	***Location 3****	***Location 5***
Case 3.2.1: Flow in a Lid-Driven Cavity	CFD	0.18	2.88	N/A
	FFD	5.57	8.11	N/A
Case 3.2.2: Forced Convection in an Empty Room	CFD	6.82	9.25	12.27
	FFD	7.85	10.6	9.97
Case 3.2.3: Forced Convection in a Room with a Box at Center	CFD	17.09	19.24	14.36
	FFD	17.99	22.19	14.66
Case 3.2.4: Mixed Convection in a Room with a Box at Center	CFD	6.9	15.54	5.39
	FFD	10.4	11.00	5.98

*For case 3.2.1, Location 1 and Location 3 are the line at X=0.5m and the line at Z=0.5m, in the XZ plan which was sliced at Y=0.01165m, respectively.

When running *OpenCL_FFD* on the CPUs, the results are the same as *C_Reference* for all the case studies (Figure 7a, 7a, 8a, 9a, and 10a). Surprisingly, results from *OpenCL_FFD* on the GPUs are not always consistent with *C_Reference.* When the flow is simple, e.g. lid-driven cavity case, all the GPUs generated identical results as *C_Reference* (Figure 7b). However, when the flow gets complex, the GPU results diverge slightly from *C_Reference*, as well as from each other (Figure 8b, 8b, 9b and 10b). This shows that the accuracy of the OpenCL_FFD depends on the executing GPU, which is contradictory to the hypothesis that all OpenCL devices should generate the same results.





(a) (b)

Figure 7 Horizontal velocity profiles in the vertical mid-section (X=0.5m) for the lid-driven cavity flow (case 3.2.1)

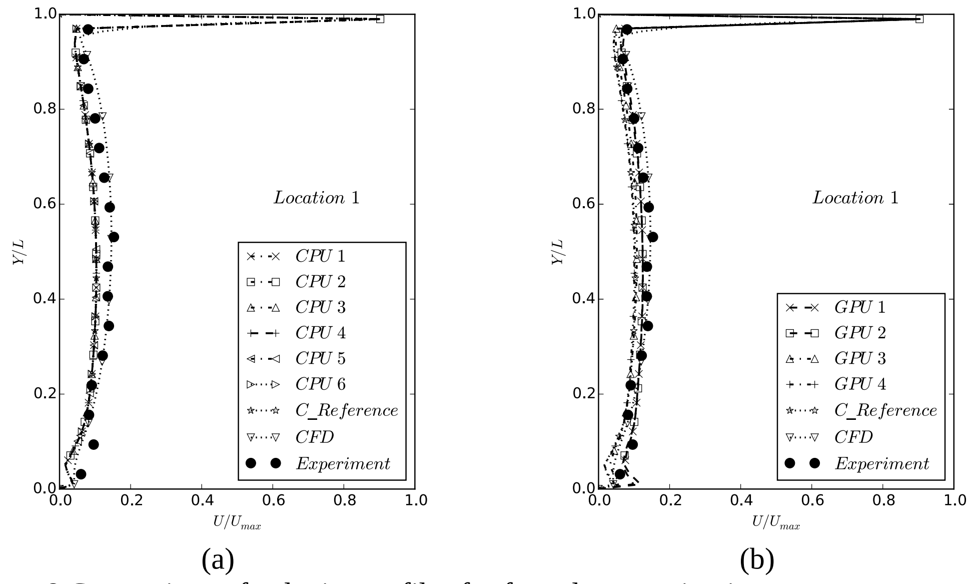


Figure 8 Comparison of velocity profiles for forced convection in an empty room (case 3.2.2)

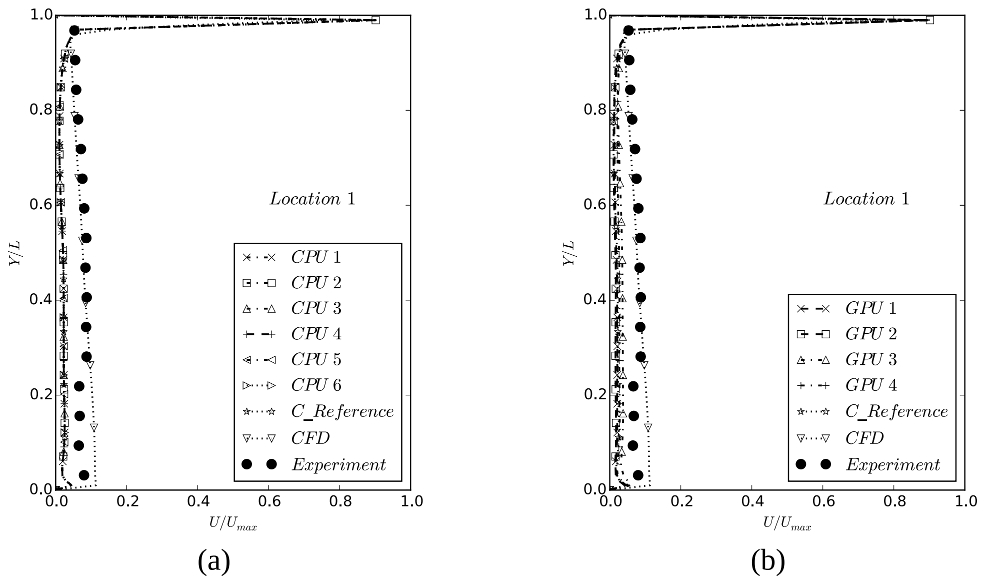
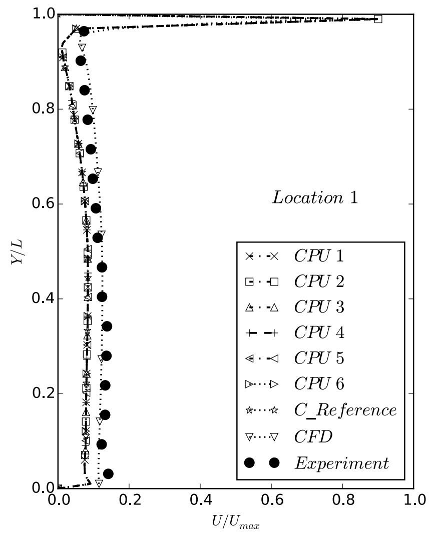
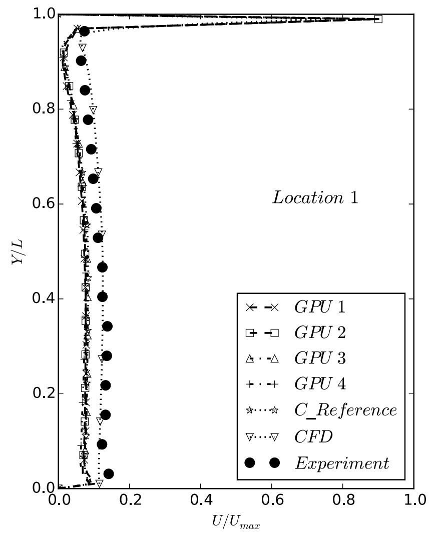


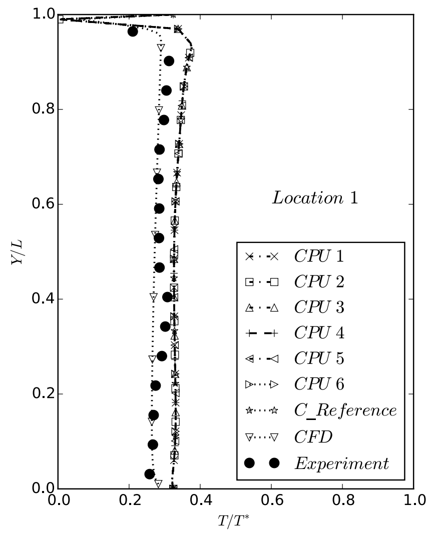
Figure 9 Comparison of velocity profiles for forced convection in a room with a box (case 3.2.3)

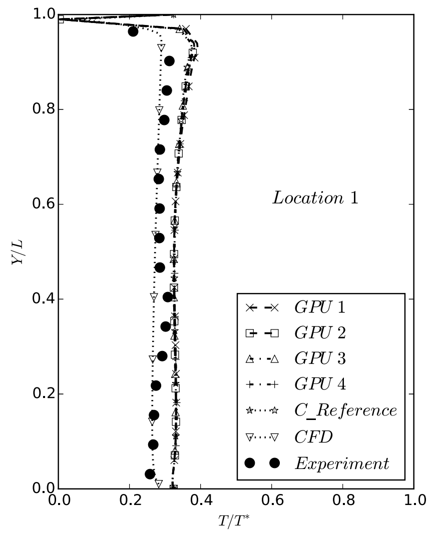




(a) (b)

Figure 10 Comparison of velocity for mixed convection in a room with a box (Case 3.2.4)





(a) (b)

Figure 11 Comparison of temperature for mixed convection in a room with a box (Case 3.2.4)

We can use *Coefficient of Determination* R^2 to quantify the difference between *OpenCL_FFD* and *C_Reference.* The R^2 is defined as follows:

$$\boldsymbol{R^2 = 1 - \frac{\sum_1^n (x_i - \hat{x}_i)^2}{\sum_1^n (x_i - \bar{x})^2}}, \tag{5}$$

where x_i is the value at *i* point in the profile from *C_Reference*; $\bar{x}$ is the mean value of *n* points from *C_Reference*; $\hat{x}_i$ is the value at i point in the profile from *OpenCL_FFD*; *n* is the total number of points in the profile, which in this case is 100. Table 3 summarizes the R^2 for velocity and temperature profiles at different locations in Figures 7-10 for the three cases. For most cases, the value of R^2 is above 0.9 which indicates that the errors are not significant. Intriguingly, the value of R^2 seems independent to the GPU manufacturers, the complexity of the flow, or locations at which profiles were extracted.

Table 3 R^2 of the results from *OpenCL_FFD* on GPUs

	R^2 of velocity profile for the forced convection in an empty room				**R^2 of velocity profile for the forced convection in a room with a box**			
	GPU 1	*GPU 2*	*GPU 3*	*GPU 4*	*GPU 1*	*GPU 2*	*GPU 3*	*GPU 4*
Location 1	0.9106	0.9270	0.9946	0.9994	0.9994	0.9979	0.9825	0.9958
Location 3	0.9931	0.9902	0.9969	0.9993	0.9982	0.9969	0.9956	0.9964
Location 5	0.8907	0.8927	0.9852	0.9769	0.9678	0.9794	0.9794	0.9636
	R^2 of velocity profile for the mixed convection in a room with a box				**R^2 of temperature profile for the Mixed convection in a room with a box**			
	GPU 1	*GPU 2*	*GPU 3*	*GPU 4*	*GPU 1*	*GPU 2*	*GPU 3*	*GPU 4*
Location 1	0.9947	0.9969	0.9995	0.9931	0.9722	0.9939	0.9987	0.9941
Location 3	0.9846	0.9887	0.9966	0.9831	0.9990	0.9994	0.9999	0.9991
Location 5	0.9568	0.9544	0.9891	0.9700	0.9781	0.9706	0.9904	0.9786

To investigate why OpenCL on GPUs generated different results, we performed a numerical experiment to check the output at each step of *OpenCL_FFD* simulation. For example, Table 4 shows the comparison for case 3.2.2 at five control volumes. Two references were created using *C_FFD* on the *CPU 2* and *CPU 4*.

The GPUs computed different values than the *C_Reference* at either the 1st or 100th time step. The difference is less than 2×10^{-6} at the first step. This indicates that the inaccuracy may be a round-off error since SPFP is applied. However, the difference increased up to 5×10^{-6} at the 100th time step, which is likely due to the accumulation of the round-off errors.

A recent study (Gu et al. 2015) provided more insights on the inconsistency of the OpenCL based calculations on the GPUs. It found that due to the lack of clarification in the current OpenCL specification, manufacturers could implement the Fused Multiply and Add (FMA) process in different ways, although they are all compatible with the *IEEE-754 2008 standard* (IEEE 2008). Moreover, current AMD-manufactured GPUs are not *IEEE-754* compatible since they implement a truncation instead of round-off operation during their FMA process. Therefore, the round-off errors observed in Table 4 are likely caused by the varying FMA implementations. As a results, the hypothesis that the OpenCL code will generate the same results on different results is not valid for the current OpenCL version 1.2 (Khronos 2012).

4.3. Computing Speed Evaluation

4.3.1. Case Study Speedup

We define the speedup *N* as

$$\boldsymbol{N = t_{bench}/t_{OpenCL}}, \tag{6}$$

where t_{bench} is the computing time used by the benchmark code (*C_Reference* on a *CPU)* and t_{OpenCL} is the computing time used by *OpenCL_FFD* on different devices.

The speedup of *OpenCL_FFD* for the multi-core CPU was calculated based on the *C_Reference* using a single processor of the same CPU. The implementation of the *OpenCL_FFD* code flattened a two-dimensional array of variables into one-dimension, which reduced data access time when compared to *C_FFD*, which used a two-dimensional array for its variable storage. As a result, the optimization in implementation makes it possible that speedups of *OpenCL_FFD* can be higher than the number processors. For instance, the speedups of CPU 4 are larger than 2 which correlates with its number processors. It is also interesting to see that *OpenCL_FFD* on CPU 1 (which is on a Mac Computer running Windows using Boot Camp) is slower than the *C_Reference*. As a comparison, CPU 5, the same CPU on the Mac computer running on Mac OS X has a much higher performance. Since *OpenCL_FFD* achieved speedups on other CPUs while under a native Windows machine (CPU 2 and CPU 3), it is likely that the slowdown of CPU 1 is caused by the use of Boot Camp on the MacBook Pro laptop to run Windows.

Table 4 Comparison of GPU results at 1st and 100th time steps in 5 control volumes for the mixed convection in a room with a box (Shading of cell indicates that the GPU result is different than the reference computed by CPU)

Device	**OS**	**Value at selected points (i, j, k)**									
		U @ (2, 38, 34)	**U @ (2, 38, 34)**	**U @ (2, 12, 38)**	**U @ (2, 12, 38)**	**T @ (13, 23, 40)**	**T @ (13, 23, 40)**	**V @ (1, 29, 35)**	**V @ (1, 29, 35)**	**U@ (14,23,38)**	**U @(14,23,38)**
		1st time step	**100th time step**	**1st time step**	**100th time step**	**1st time step**	**100th time step**	**1st time step**	**100th time step**	**1st time step**	**100th time step**
Reference 1	Windows	0.031247	0.033471	0.098339	1.359278	22.199999	22.200006	0.000635	0.000155	0.000000	1.351563
Reference 2	Mac OS X	0.031247	0.033471	0.098339	1.359278	22.199999	22.200006	0.000635	0.000155	0.000000	1.351562
GPU 1	Mac OS X	0.031247	0.033469	0.098339	1.359277	22.199999	22.200005	0.000634	0.000155	0.000000	1.351564
GPU 2	Mac OS X	0.031247	0.033469	0.098339	1.359277	22.199999	22.200005	0.000634	0.000155	0.000000	1.351564
GPU 3	Windows	0.031247	0.033469	0.098339	1.359278	22.200001	22.199999	0.000634	0.000155	0.000000	1.351564
GPU 4	Windows	0.031247	0.033471	0.098338	1.359278	22.200001	22.200001	0.000634	0.000155	0.000000	1.351563

Table 5 Speedups of OpenCL_FFD on CPUs for all case study

Device	**Number of Processors**	**Speedup**			
		Lid driven cavity flow	**Forced convection in an empty room**	**Forced convection in a room with a box**	**Mixed convection in a room with a box**
CPU 1	2	0.8	0.8	0.8	0.7
CPU 2	4	4.2	2.7	2.6	2.6
CPU 3	4	2.1	2.1	2.0	2.1
CPU 4	2	2.5	3.0	2.5	2.6
CPU 5	2	2.0	2.2	1.9	1.9
CPU 6	4	3.8	4.1	3.2	3.3

The speedup of *OpenCL_FFD* for the GPUs was calculated based on *C_Reference* on *CPU 2*. As shown in Table 6, a higher peak performance can lead to a larger speedup. For example, *GPU 4* has the highest peak performance which is about one order of magnitude larger than other studied GPUs. As a result, GPU 4 provided speedups which were one to two orders of magnitude higher than the others. However, the speedup is not perfectly proportional to the peak performance; other factors may also affect the speedup such as the global work size. The global work size is the number of all work items, which is equal to the total number of grids in our case. In next the section we discuss how the global work size affects the speedup.

Table 6 Speedup of OpenCL_FFD on GPUs for all case study

Device	**Peak Performance (FLOPS)**	**Speedup**			
		Lid driven cavity flow	**Forced convection in an empty room**	**Forced convection in a room with a box**	**Mixed convection in a room with a box**
GPU 1	346*	7.6	5.3	5.1	5.1
GPU 2	653	7.9	8.1	7.2	7.2
GPU 3	768	17.5	15.9	13.7	13.8
GPU 4	4,200	129.3	77.2	72.6	73.3

*Estimated due to lack of official information

4.3.2. Impact of Global Work Size

In order to analyse the impact of global work size on the performance of the devices, we measured the speedup of *OpenCL_FFD* using different ranges. The test was performed using a lid-driven cavity case described in section 3.2.1 but with a different dimension of *1m* × *1m* × *1m*. Three devices (*CPU 2*, *GPU 3*, and *GPU 4*) were examined in the test. Since the global work size is equal to the number of grids in our case, we can adjust the global work size by adjusting the number of grids.

As we can see from Figure 8, *OpenCL_FFD* on *CPU 2, GPU 3,* and *GPU 4* can eventually achieve a maximum speedup of 12, 140, and 1139, respectively. The speedup of OpenCL_FFD on *CPU 2* can be larger than its number of processors due to the optimization in the OpenCL implementation for more efficient data access, which is discussed in the section 4.2.1. Note that a threshold exists for each device which dictates if the speedup increases or stalls with the increase of global work sizes. The threshold is about $1.25{\times}10^5$ for *CPU 2* and about $2.16{\times}10^5$ for *GPU 3* and *GPU 4.* When the global work size is below the threshold, the speedup increases with the work size because the computing capacity of the device is not fully utilized. After the work size exceeds the threshold, the speedup stops increasing because all the device's computing capacity is used up.

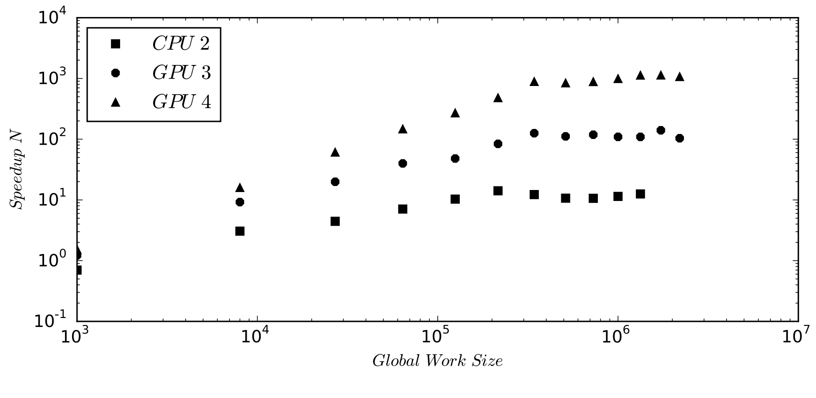


Figure 12 Speedup of OpenCL with different global work size for the lid-driven cavity flow

5. Conclusion

This study evaluated two hypotheses for the cross-platform computing using OpenCL for indoor airflow simulation. The first hypothesis that the OpenCL code will generate the same results on different devices was not valid for the current OpenCL version 1.2. Although running the OpenCL code on the different CPUs produced the identical results, the results generated by GPUs differ with a R^2 larger than 0.9. The dissimilar results by GPUs are likely caused by the divergent FMA implementation from different GPU manufacturers. Although the initial discrepancies are small at level of 10^{-6}, they can accumulate over time during simulations.

The second hypothesis that running in parallel on multiple processors of the same device will speed up the indoor airflow simulation was valid although the speedup is affected by the capacity of the device (e.g. peak performance) and the global work sizes. In addition, optimizing the data access can provide additional speedup. A separate study on the relationship and number of grids showed that a speedup of 1139 times can be achieved using an AMD FirePro W8100 GPU.

In addition, the comparison of FFD and CFD with RNG k-ε model showed that both CFD and FFD can predicted the studied flows with averaged relative errors of 9.99% and 11.30%, respectively.

In conclusion, OpenCL is an appealing method for accelerating the indoor airflow simulations by utilizing parallel computing on local devices. However, it is critical that GPU manufacturers address the irregularity in the FMA implementation to ensure consistent results can be generated by different GPUs. As an intermediate method, the accuracy of FFD is comparable to the CFD.

Acknowledgments

To Be Added

References

AMD. "Firepro V4900." https://www.amd.com/Documents/AMDFirePro_FamilyBrochure.pdf.

AMD. http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/.

AMD. "Firepro W8100." http://www.amd.com/en-us/products/graphics/workstation/firepro-3d/8100.

Axley, J. 2007. Multizone Airflow Modeling in Buildings: History and Theory. *HVAC&R Research,* 13 (6):907-28.

Celebi, M. E., F. Celiker, and H. A. Kingravi. 2011. On Euclidean Norm Approximations. *Pattern Recognition,* 44 (2):278-83.

Chen, Q. 1995. Comparison of Different K-E Models for Indoor Air-Flow Computations. *Numerical Heat Transfer Part B: Fundamentals,* 28 (3):353-69.

Chen, Q. 2009. Ventilation Performance Prediction for Buildings: A Method Overview and Recent Applications. *Building and Environment,* 44 (4):848-58.

Chronis, A., A. Turner, and M. Tsigkari. 2011. Generative Fluid Dynamics: Integration of Fast Fluid Dynamics and Genetic Algorithms for Wind Loading Optimization of a Free Form Surface. *Proceedings of the Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design*.

Corrigan, A., F. F. Camelli, R. Löhner, and J. Wallin. 2011. Running Unstructured Grid ‐ Based CFD Solvers on Modern Graphics Hardware. *International Journal for Numerical Methods in Fluids,* 66 (2):221-9.

Fluent. "Fluent 16.1.0." http://www.ansys.com/Products/Fluids/ANSYS-Fluent.

Ghia, U., K. N. Ghia, and C. T. Shin. 1982. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics,* 48 (3):387-411.

Godish, T., and J. D. Spengler. 1996. Relationships between Ventilation and Indoor Air Quality: A Review. *Indoor Air,* 6 (2):135-45.

Gorobets, A., F. Trias, and A. Oliva. 2013a. A Parallel Mpi+ Openmp+ Opencl Algorithm for Hybrid Supercomputations of Incompressible Flows. *Computers & Fluids,* 88:764-72.

Gorobets, A., F. X. Trias, and A. Oliva. 2013b. An Opencl-Based Parallel CFD Code for Simulations on Hybrid Systems with Massively-Parallel Accelerators. *Procedia Engineering,* 61:81-6.

Gropp, W. D., D. K. Kaushik, D. E. Keyes, and B. F. Smith. 2001. High-Performance Parallel Implicit CFD. *Parallel Computing,* 27 (4):337-62.

Gu, Y., T. Wahl, M. Bayati, and M. Leeser. 2015. Behavioral Non-Portability in Scientific Numeric Computing. In *Euro-Par 2015: Parallel Processing*, 558-69. Springer.

IEEE, S. A. 2008. Standard for Floating-Point Arithmetic. *IEEE 754-2008*.

Intel. "E5-1603." http://download.intel.com/support/processors/xeon/sb/xeon_E5-1600.pdf.

Intel. "I5-3210m." http://download.intel.com/support/processors/corei5/sb/core_i5-3200_m.pdf.

Intel. "I7-2620m." http://www.intel.com/content/dam/support/us/en/documents/processors/corei7/sb/core_i7-2600_m.pdf.

Intel. "I7-3720qm." http://download.intel.com/support/processors/corei7/sb/core_i7-3700_m.pdf.

Jin, M., and Q. Chen. 2015. Improvement of Fast Fluid Dynamics with a Conservative Semi-Lagrangian Scheme. *International Journal of Numerical Methods for Heat & Fluid Flow,* 25 (1):2-18.

Jin, M., W. Zuo, and Q. Chen. 2012. Improvements of Fast Fluid Dynamics for Simulating Air Flow in Buildings. *Numerical Heat Transfer, Part B: Fundamentals,* 62 (6):419-38.

Jin, M., W. Zuo, and Q. Chen. 2013. Simulating Natural Ventilation in and around Buildings by Fast Fluid Dynamics. *Numerical Heat Transfer, Part A: Applications,* 64 (4):273-89.

Khronos, G. "The Opencl Specification, Version 1.2." https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf.

Ladeinde, F., and M. D. Nearon. 1997. CFD Applications in the HVAC&R Industry. *ASHRAE Journal,* 39 (1):44-8.

Megri, A. C., and F. Haghighat. 2007. Zonal Modeling for Simulating Indoor Environment of Buildings: Review, Recent Developments, and Applications. *HVAC&R Research,* 13 (6):887-905.

Microsoft. "Microsoft Visual Studio 2013 Professional." https://msdn.microsoft.com/en-us/library/dd831853(v=vs.120).aspx.

NVIDIA. 2007. *Nvidia Cuda Compute Unified Device Architecture-- Programming Guide (Version 1.1)*. Santa Clara, California: NVIDIA Corporation.

NVIDIA. "Gt 650m." https://www.techpowerup.com/gpudb/547/geforce-gt-650m.html.

Patankar, S. V., and D. B. Spalding. 1972. A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows. *International journal of heat and mass transfer,* 15 (10):1787-806.

Wang, B., B. Zhao, and C. Chen. 2010. A Simplified Methodology for the Prediction of Mean Air Velocity and Particle Concentration in Isolation Rooms with Downward Ventilation Systems. *Building and Environment,* 45 (8):1847-53.

Wang, H., and Z. J. Zhai. 2012. Analyzing Grid Independency and Numerical Viscosity of Computational Fluid Dynamics for Indoor Environment Applications. *Building and Environment,* 52:107-18.

Wang, M., and Q. Chen. 2009. Assessment of Various Turbulence Models for Transitional Flows in an Enclosed Environment (Rp-1271). *HVAC&R Research,* 15 (6):1099-119.

Wang, Y., A. Malkawi, Y. Yi, and T. C. Center. 2011. Implementing CFD (Computational Fluid Dynamics) in Opencl for Building Simulation. *Proceedings of The 12th International Building Performance Simulation (Building Simulation 2011).*

Xcode. https://developer.apple.com/xcode/download/.

Yakhot, V., and S. A. Orszag. 1986. Renormalization-Group Analysis of Turbulence. *Physical review letters,* 57 (14):1722.

Zhai, Z., Q. Chen, P. Haves, and J. Klems. 2002. On Approaches to Couple Energy Simulation and Computational Fluid Dynamics Programs. *Building and Environment,* 37 (8-9):857-64.

Zuo, W., and Q. Chen. 2009a. Fast Parallelized Flow Simulations on Graphic Processing Units. *Proceedings of the the 11th International Conference on Air Distribution in Rooms (RoomVent 2009)*, Busan, Korea.

Zuo, W., and Q. Chen. 2009b. High-Performance and Low-Cost Computing for Indoor Airflow. *Proceedings of the Proceedings of the 11th International IBPSA Conference (Building Simulation 2009)*, Glasgow, U.K.

Zuo, W., and Q. Chen. 2009c. Real-Time or Faster-Than-Real-Time Simulation of Airflow in Buildings. *Indoor Air,* 19 (1):33-44.

Zuo, W., and Q. Chen. 2010a. Fast and Informative Flow Simulations in a Building by Using Fast Fluid Dynamics Model on Graphics Processing Unit. *Building and Environment,* 45 (3):747-57.

Zuo, W., and Q. Chen. 2010b. Fast Simulation of Smoke Transport in Buildings. *Proceedings of the the 41st International HVAC&R congress*, Beograd, Serbian, December 1-3.

Zuo, W., and Q. Chen. 2010c. Improvements on the Fast Fluid Dynamics Model for Indoor Airflow Simulation. *Proceedings of the 4th National Conference of IBPSA-USA (SimBuild 2010)*, New York, NY.

Zuo, W., M. Jin, and Q. Chen. 2012. Reduction of Numerical Viscosity in FFD Model. *Engineering Applications of Computational Fluid Mechanics,* 6 (2):234-47.

Zuo, W., M. Wetter, D. Li, M. Jin, W. Tian, and Q. Chen. 2014a. Coupled Simulation of Indoor Environment, HVAC and Control System by Using Fast Fluid Dynamics and Modelica. *Proceedings of the 2014 ASHRAE/IBPSA-USA Building Simulation Conference*, Atlanta, GA, Sep. 10-12.

Zuo, W., M. Wetter, D. Li, M. Jin, W. Tian, and Q. Chen. 2014b. Coupled Simulation of Indoor Environment, HVAC and Control System by Using Fast Fluid Dynamics and the Modelica Buildings Library. *Proceedings of the American Society of Heating, Refrigeration, and Air-Conditioning Engineers (ASHRAE)*.