IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received February 7, 2017, accepted April 18, 2017, date of publication May 24, 2017, date of current version June 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2705434

Evaluation of FPGA Hardware as a New Approach for Accelerating the Numerical Solution of CFD Problems

ABBAS EBRAHIMI[1] AND MOHAMMAD ZANDSALIMY[2]
[1]Department of Aerospace Engineering, Sharif University of Technology, Tehran 11155-1639, Iran
[2]Sharif University of Technology, Tehran 11365/8639, Iran

Corresponding author: Abbas Ebrahimi (ebrahimi_a@sharif.ir)

ABSTRACT The main purpose of this paper is to investigate the feasibility of using field programmable gate arrays (FPGAs) chips as alternatives for the conventional CPUs to accelerate the numerical solution of the fluid dynamics differential equations. FPGA is an integrated circuit that contains an array of logic blocks, and its architecture can be reprogrammed and reconfigured after manufacturing. Complex circuits for various applications can be designed and implemented using FPGA hardware. The reconfigurable hardware used in this paper is a system on a chip FPGA type that integrates both microprocessor and FPGA architectures into a single device. In this paper, typical computational fluid dynamics problems, such as the Laplace and 1-D Euler equations, are implemented and solved numerically on both reconfigurable hardware and CPU. The precision of results and speedups of the calculations is compared together. In some cases, the computational process on FPGA is up to 20 times faster than a conventional CPU, with the same data precision. Several numerical and analytical solutions are used to validate the results.

INDEX TERMS FPGA, CFD, reconfigurable hardware, numerical solutions, hardware definition language, accelerating numerical solutions.

I. INTRODUCTION

There are three approaches for predicting and analyzing fluid flow problems in various applications; analytical methods, experimental tests and computational fluid dynamics (CFD) [1]. Due to the nonlinearity of Navier-Stokes equations and also very complex configurations in aerospace industry, the analytical methods have never been widely implemented by aerodynamicists. Traditionally design process and optimization in the different fluid dynamics applications are performed via experimental tests which typically are expensive and time-consuming. In recent decades, by the emergence of high-speed processors, CFD has become an auxiliary tool to the experimental tests, by providing a far detailed investigation on the flow field and decreasing the risk of design failure [2]. CFD perfectly complements the empirical methods in a way that has made great contributions in the development of recent commercial transporters such as Boeing 787 and Airbus 380 [3], [4]; however, it is still far from the point to be employed as the unique tool of aerodynamic assessment tool in the foreseeable future [5]. CFD has been known as a technique with massive floating point operations [6], which is due to the fact that most of the aeronautical problems require a fine computational mesh with a good distribution of grid points as well as a sufficient number of iterations to yield an accurate solution. As a result, even by employing the highly advanced Cray computers, the solution time of a typical aerodynamic problem will still be significantly high [7].

Several techniques and ideas have been suggested and implemented for reducing the solution time of differential equations that governs the fluid flow. In general, these methods are categorized into software and hardware methods. Optimization of the computer program and the use of new numerical algorithms are examples of software methods [8], While the use of more powerful CPUs, or employing alternative hardware such as GPUs or HPC (High-Performance Computing) systems are examples of hardware methods [9], [10].

Numerous studies have been performed within the last decade which demonstrates the promising future of using FPGA (Field Programmable Gate Array) for speeding up the CFD computations [6], [11]–[22]. Also, it was shown by [17] that by coupling FPGA logic with high bandwidth external memory achieving a computational performance of several GFLOPs is possible. Furthermore, typical numerical methods implemented to solve the governing equations of the flow field, usually include simple arithmetic operations that in most cases, the calculations should be repeated over and over again for each one of computational nodes of the numerical mesh in order to reach the desired accuracy. This makes FPGA-based flow solvers ideal for computational purposes. This is due to the high efficiency of FPGA-based computers in parallelizing at the hardware level for simple arithmetic or iterative numerical solutions, see [15]–[17]. More important, since parallel processing can be conducted at two levels i.e., system level (using multiple hardware units) and hardware level (configuring a single hardware unit architecture); a device like FPGA logic capable of employing both levels would be of great benefit.

The primary goal of manufacturing FPGAs was for prototyping hardware or being used as a connection bridge between separate hardware units [23]. There are several studies in the literature reporting on different digital applications of FPGAs such as signal and image processing [24]–[28]. Further, the effect of data precision on the numerical solution obtained by using FPGA [29]–[31], the impact of data throughput of this chip via I/O pins with the outside world [32], [33] and FPGA logic area usage for different applications have been studied [34].

Although FPGA has been widely used for digital applications, it has been far less employed in computational sciences. However, by the significant increment of clock frequency as well as logic block density, FPGAs can now be implemented as highly flexible standalone computational processors [21], [35], [36]. In the following, some of the most relevant researches on the later applications of FPGAs are mentioned.

In order to construct a CFD accelerator, Smith and Schnore [11] implemented a reconfigurable hardware for three of the most computationally expensive functions including Euler, Viscous, and Smoothing algorithms and showed that dramatic improvement in sustained computational speed can be achieved through reconfigurable computing. In 2007, Nunez *et al.* [13] proposed and fully discussed three levels of parallelism that can be applied to reconfigurable hardware systems including fine-grained, coarse-grained, and algorithm-level parallelism.

In 2008, Dongarra *et al.* studied the feasibility of integrating both FPGA and CPU logics in a single device as a hybrid architecture to accelerate CFD solutions and achieved 20 times faster calculations than a Pentium4 CPU implementing dense and sparse linear algebra computational kernels [15]. Andrés *et al.* [19] presented a brief study on the feasibility of using FPGAs to accelerate CFD simulations. In 2011, Sanchez-Roman *et al.* [20] exhibited an FPGA-based accelerator to implement a cell-vertex finite volume algorithm for solving the Euler equations. Liu *et al.* [22] published a framework based on reconfigurable logic to implement a 1-D CFD model of a diesel fuel system.

In the present research, the hardware structure and the configuration methods of an FPGA are represented at first. Then, typical CFD cases such as Laplace equation and 1-D Euler equation are solved numerically via implementing different mesh sizes and numerical schemes. The results were compared with corresponding data obtained from a CPU in terms of run time and precision. By this study, the computational advantages of employing FPGA over a conventional CPU were studied and represented.

II. RECONFIGURABLE HARDWARE

Generally, reconfigurable hardware is built of an array of reprogrammable logic blocks that are linked to each other with communication wires. The function of each logic block, as well as the connections between them are reconfigurable after being manufactured. The concept of reconfigurable computing emerged in the 1960s when Estrin proposed the idea of designing and building a computer consisting of a standard processor and an array of reconfigurable hardware blocks [37]. In this architecture, the main processor is supposed to act as the main unit to control the performance of the reconfigurable hardware. Besides, the reprogrammable logic could be programmed to perform a user defined task. Being reconfigurable allows the hardware to be readjusted to perform new tasks that were not desired at first. Such configuration yielded in a hybrid computer structure that combines the flexibility of reconfigurable hardware and the speed of traditional CPUs. Later, Casselman presented a field programmable logic device architecture in 1987 which was aimed to create a computer chip that was able to be completely programmed using software [23].

Recently developed reconfigurable hardware fabric are consist of a number of various electronic components such as memory cells and connection switch blocks. Memory cells are used as lookup tables to implement the universal gates and also to control the configuration of the switches in the interconnection network. A configuration is a software program that defines the function of each logical gate and the switch state. The main difference between reconfigurable hardware and conventional microprocessors is the ability to change data paths and having control over data transmission process. The most common type of reconfigurable hardware device is an FPGA which its architecture can be reconfigured through Hardware Description Languages (HDLs) such as "VHDL" and "Verilog".

A. FPGA-CPU CONNECTION

Reconfigurable logic has shown to be inefficient at conducting some specific operations like logical loops or branch control [38]. In order to get the fastest solution of a reconfigurable computing system, these operations need to be executed on a host microprocessor. On the other hand, reconfigurable logic can be used to perform the processes with a high density of computation load, rapidly and smoothly. Different types of FPGA-CPU connection architectures have been developed to be used in reconfigurable computing applications. One of

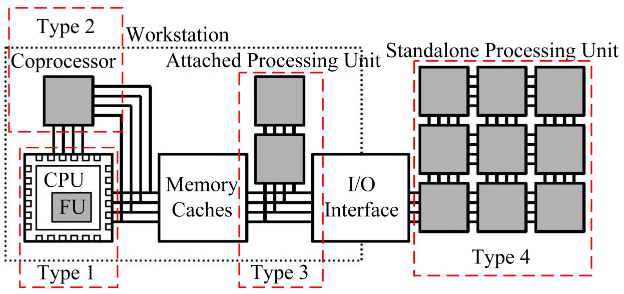

FIGURE 1. Different levels of coupling in a reconfigurable system (Reconfigurable logic is shaded) [38].

the fundamental parameters that characterize a connection architecture is the level of coupling (assuming any) with a host microprocessor. For a hybrid system that integrates both microprocessor and reconfigurable logic, there are several ways to couple these hardware components. The main concept is illustrated in Fig. 1 schematically.

1) Reconfigurable logic can be incorporated inside a host processor. This type provides a customary programming environment with the added reconfigurable hardware that can be modified to execute some custom operations. In this case, the reconfigurable unit acts as an auxiliary subsystem of the primary processor and cannot take action independently.
2) A reconfigurable unit might be employed as a coprocessor. In this concept, the coprocessor which is typically larger than a functional unit of Type 1, can perform calculations independent from the non-stop supervision of the host processor. Even now, the processor triggers the reconfigurable unit and sends the initial data to commence a function on the reconfigurable hardware or gives data about where this information may be found in memory.
3) A joint reconfigurable processing unit acts as an extra processor in a multiprocessor framework. There is, accordingly, a higher delay in correspondence between the host processor and the reconfigurable hardware. For example, when setup information or input and output data are transformed. This type of communication architecture provides a high bandwidth connection between CPU and FPGA.
4) A standalone reconfigurable hardware unit can be coupled to a host processor via their peripheral connections. In this sort of connection architecture, the standalone FPGA-based unit will occasionally have interactions with CPU. This model acts analogous to the workstations of a network in which data processing takes most of the time with the least need to have continuous communication with CPU.

All of the discussed architectures have their own advantages and disadvantages. Another parameter that can be influential in an architecture is the distance between FPGA and CPU. Shorter distance means less communication overhead and accordingly less time required for interactions. On the other hand, a more independent FPGA-based processing unit enables a broader variety of tools for hardware parallelism in program execution, though it may have a higher communication overhead as mentioned before. Communication latency can be the most deteriorating disadvantage for applications with a high number of transaction, where in special circumstances, it may reduce or completely disappear the acceleration benefits that is desired to be achieved through using this type of reconfigurable hardware. The present study considers an FPGA and a CPU as two separate units that are connected to each other with very high-speed lines (second integration type).

III. THE HARDWARE IN USE

In the present study, an SoC FPGA is used as the computing hardware for numerical calculations. An SoC FPGA is made of both programmable logic (FPGA hardware) and processing system (microprocessor hardware) on the same chip. Herein, a Zynq-7020 SoC FPGA from Zynq-7000 family chips by Xilinx Co. is employed. The Zynq-7000 family products integrate a feature-rich dual-core or single-core ARM Cortex-A9 based processing system (PS) and 28 nm Xilinx programmable logic (PL) in a single device. Zynq-7020 chip is optimized for massive computations with low power consumption. The SoC FPGA chip employed in the present study will be usable only if the I/O pins being connected to the standard external connections on an electronic board. Moreover, the implemented board is a z-turn board manufactured by MYiR Co. There are various external connections on this board, such as USB, LAN, JTAG and micro SD as is shown in Fig. 2.

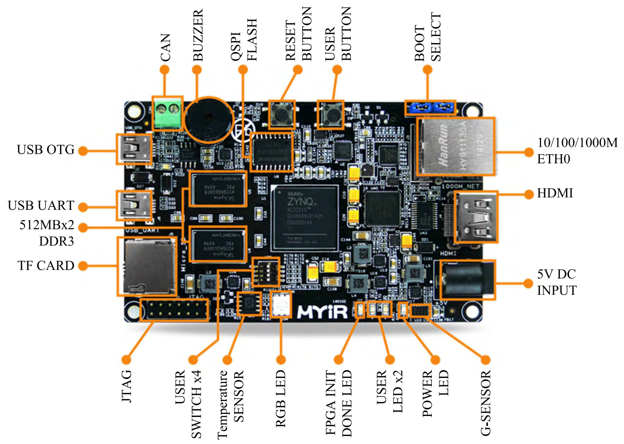

FIGURE 2. z-turn board by MYiR Co.

An intel Core i7 Q-740 processor was used to compare the solution time of various tests with the results obtained from FPGA. This CPU owns 4 physical processing cores each capable of two threads at the same time with a maximum frequency performance of 1.73 GHz per core. In other words, it can perform 13.84 billion floating point operations per second (13.84 GFLOPS of calculation power). This is the maximum nominal calculation performance of this chip that cannot be achieved in a practical application.

IV. THE CONFIGURATION METHOD

In spite of several performance benefits of reconfigurable hardware in program execution, it may be ignored by programmers if incorporating a reconfigurable hardware into a system cannot be done easily. This requires a user-friendly software design environment that helps in creating the hardware configurations. This environment can be extended anywhere between a software assist for manual circuit design, to a complete automated circuit configuration system. Manual circuit description (using HDLs) is a powerful method for designing high-quality circuit configurations [39]. However, this process takes a lot of time and also requires a full knowledge about the particular reconfigurable hardware employed. On the other hand, an automatic design procedure provides a quick and easy way to program reconfigurable systems and therefore makes using this kind of hardware easier for general application programmers. One of these automated design tools is implementing high-level hardware description languages (HLLs) which are based on codes written in C/C++ programming language. Though using HLLs reduces the time spent on developing hardware architecture, they need a manual optimization to operate at the best performance. New series of FPGAs make using floating point numbers and mathematical operations much easier, so they look very promising for applications with floating point calculations [40].

Digital circuit description is the process of describing the user specified functions that are intended to be implemented in the reconfigurable hardware. Performing this procedure can be as complex as specifying the inputs, outputs, and operation signals of each basic function block in the reconfigurable system (manual technique). Besides, it can be as simple as writing a code in C programming language that represents the functionality of the entire algorithm to be implemented in hardware (automatic technique). This process can also be somewhere in the middle of these two techniques; such as the method of specification of the circuit by using prebuilt operational components (e.g. adders and multipliers) which will be mapped to the actual hardware later in the design procedure.

The implemented method for circuit design in the present study is an automated design process for Zynq-7000 FPGA family recommended by Xilinx Co. Three software are used in this method, all of them optimized for the task, including; Vivado, Vivado HLS, and Xilinx SDK. Using Vivado HLS and starting with a code in C++, some IP (Intellectual Property) blocks were produced and then packaged. An IP block is a logical hardware description layout that includes a number of inputs and outputs. These IP blocks are then transferred to Vivado and connected together to form a fully functional electronic circuit. An example of hardware design in Viavdo for solving the Laplace equation and the used IPs are demonstrated in Fig. 3. Each one of the blocks has a special role and function in the main architecture.

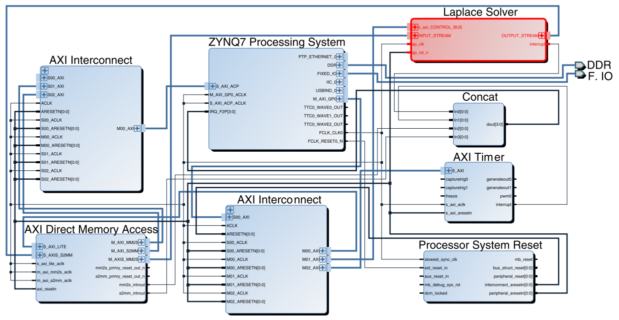

FIGURE 3. Block diagram of hardware designed to solve the Laplace equation on Zynq-7020.

Laplace Solver IP core is the solver of the Laplace equation that has been developed via Vivado HLS and from a program written in C++, particularly for the present study. This IP receives the initial conditions for the Laplace equation and then after completing the solution, sends out the results through the output port. The AXI Timer block is an IP provided by Xilinx Co. to calculate the exact clock rate of the chip during the running time of a program. The main purpose of this block is to calculate the correct solution time of each problem. Also, ZYNQ7 Processing System block can initiate and control the ARM processors available in the Zynq-7020 chip. By means of this IP, it is possible to communicate with the Laplace Solver block through the high bandwidth AXI Interconnect connection block and control the data flow into and from the reconfigurable unit. For the interested readers, a full discussion about IP blocks can be found in [41]. The final circuit design is then applied to the actual hardware and debugged using Xilinx SDK.

The configuration method of the hardware for each problem in question can vary in details (e.g. utilizing various directives to lower the latency, changing the memory addresses for more rapid connections with the memory and so on). Despite these minimal differences, the overall design procedure can be divided into distinguished steps. A simple representation of the configuration method is proposed in Algorithm 1. The procedure starts off with the IP core design inside Vivado HLS (steps 1 through 5) and continues with assembling these IP blocks inside Vivado (steps 6 through 8). After performance analysis and verification, the final design is applied to the actual hardware using Xilinx SDK (steps 9 through 11).

V. RESULTS AND DISCUSSION

The main purpose of this study is to reduce the solution time required for solving CFD problems by using FPGA. In this regard, various CFD problems that are solved by implementing different numerical methods are chosen. Each problem has its own computational load and therefore a more general study can be made. However, every problem with a

```
Algorithm 1 A General Method for High Level Hardware
Configuration
   Input  : A code in C/C++ programming language
 1 C code verification
 2 Design synthesis
 3 Design analysis and optimization
 4 RTL verification
 5 IP core packageing and export
 6 Assembling IP cores
 7 Design synthesis and verification
 8 Hardware packaging and export
 9 Design a software to run the configuration
10 Construct Boot file
11 Apply the bitstream to the actual hardware
   Output: The results of the solution
```

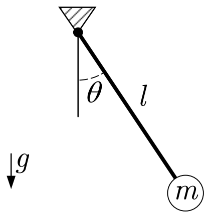

FIGURE 4. Diagram of a simple pendulum.

high computational load may not be fully implemented on a specific FPGA because the number of logic blocks and its components are limited. To tackle this issue, such problems would be partially implemented on both reconfigurable hardware and CPU, and then the results will be compared. Three differential equations have been studied in the present research which are:

1) An ordinary 2nd order differential equation
2) The Laplace equation
3) The quasi-one-dimensional inviscid flow governing equation

A. ORDINARY 2ND ORDER DIFFERENTIAL EQUATION

In physics and engineering, the use of Newton's second law of motion leads to a system of second-order differential equations that are implemented for modeling some of the most important physical phenomena of nature. High-order equations can be studied either directly or through equivalent systems of first-order equations [42]. An example of an ordinary second order differential equation is the equation governing the motion of a simple pendulum. An illustration of this pendulum is shown in Fig. 4. Neglecting the friction forces, the equation of motion about its center of rotation can be written as:

$$\frac{d^2\theta(t)}{dt^2} = -\frac{g}{l}\sin(\theta(t)) \tag{1}$$

Where θ is the angle in radians and l is the pendulum length in meters. By defining following parameters:

$$Y_1(t) = \theta(t), \quad Y_2(t) = \theta'(t) \tag{2}$$

Where the $\theta'(t)$ is the time dericative of $\theta(t)$. Equation 1 can be rewritten to get the system of first order differential equations 3 and initial conditions 4.

$$\begin{cases} Y_1'(t) &= Y_2(t) \\ Y_2'(t) &= -\frac{g}{l}\sin(Y_1(t)) \end{cases} \tag{3}$$

$$Y_1(0) = \theta(0), \quad Y_2(0) = \theta'(0) \tag{4}$$

Using Euler discretization method [42], one can change the equations 3 into the algebraic system of equations 5 that are solvable using a simple computer program. Herein, h is the step size and $y_{1,n}$ and $y_{2,n}$ are the discrete values of Y_1 and Y_2 at the time step n, respectively.

$$\begin{cases} y_{1,n+1} &= hy_{2,n} + y_{1,n} \\ y_{2,n+1} &= -\frac{gh}{l}\sin(y_{1,n}) + y_{2,n} \end{cases} \tag{5}$$

Initial conditions 6 are used in the solution of these equations.

$$\theta(0) = \frac{\pi}{20}, \quad \theta'(0) = 0 \tag{6}$$

The results of the solution of Eq. 5 from [43] for a pendulum of different lengths are given in Table 1. In this table, the relative error of the numerical solution of Eq. 5 for using CPU and FPGA hardware in single precision is also reported. The hardware configuration is designed so that the results of FPGA and CPU become identical with any data precision.

The solution time results of pendulum equation for single and double precision floating points, at different times with $h = 10^{-4}$ are given in Table 2. In Fig. 5, the graph of solution time (the left vertical axis) by using CPU and FPGA versus different times with $h = 10^{-4}$ is plotted. The amount of solution speed increment is shown for different data precisions on the right vertical axis. The solution speed of Eq. 5 for employing FPGA is up to 3.8 times faster than the solution of CPU.

B. THE LAPLACE EQUATION

One of the important cases that has been frequently studied in the context of accelerating numerical computations is solving the Laplace equation by taking advantage of FPGAs [13], [21]. The governing equation of an incompressible, inviscid flow (potential flow) and also the governing equation of a simple steady state heat transfer problem without any source terms, is the Laplace equation. In a potential flow by solving the Laplace equation for the stream function (ψ) in a two-dimensional flow and then by calculating its derivatives, velocity field can be yielded. But in the case of a simple steady state heat transfer problem, after solving the Laplace equation the temperature contour is achieved directly. Consider the Laplace equation in 2D Cartesian form (Eq. 7) with the boundary conditions of Fig. 6. In this figure, ψ_x denotes

TABLE 1. **The results of the solution of Eq. 5 from [43] and the difference with solutions on FPGA and CPU.**

l	h	t	$\theta(t)\times 10$	$\theta'(t)\times 10$	$\theta(t)$ Rel. Error	$\theta'(t)$ Rel. Error
0.1	1×10^{-4}	0.1	0.863598	-12.978329	0.07%	$< 10^{-2}\%$
		0.2	-0.622164	-14.268626	0.11%	$< 10^{-2}\%$
		0.3	-1.546879	-2.699039	$< 10^{-2}\%$	$< 10^{-2}\%$
		0.4	-1.078647	11.292661	0.05%	$< 10^{-2}\%$
		0.5	0.361755	15.122973	0.21%	$< 10^{-2}\%$
	5×10^{-5}	0.1	0.863598	-12.978329	0.03%	$< 10^{-2}\%$
		0.2	-0.622164	-14.268626	0.05%	$< 10^{-2}\%$
		0.3	-1.546879	-2.699039	$< 10^{-2}\%$	$< 10^{-2}\%$
		0.4	-1.078647	11.292661	0.02%	$< 10^{-2}\%$
		0.5	0.361755	15.122973	0.11%	$< 10^{-2}\%$
0.2	1×10^{-4}	0.1	1.202398	-7.067398	0.02%	$< 10^{-2}\%$
		0.2	0.269355	-10.82676	0.2%	$< 10^{-2}\%$
		0.3	-0.790372	-9.494859	0.05%	$< 10^{-2}\%$
		0.4	-1.478598	-3.706318	0.01%	$< 10^{-2}\%$
		0.5	-1.473049	3.812674	0.01%	$< 10^{-2}\%$
	5×10^{-5}	0.1	1.202398	-7.067398	0.01%	$< 10^{-2}\%$
		0.2	0.269355	-10.82676	0.1%	$< 10^{-2}\%$
		0.3	-0.790372	-9.494859	0.02%	$< 10^{-2}\%$
		0.4	-1.478598	-3.706318	$< 10^{-2}\%$	$< 10^{-2}\%$
		0.5	-1.473049	3.812674	$< 10^{-2}\%$	$< 10^{-2}\%$

TABLE 2. **The solution time results of Eq. 5 using FPGA and CPU.**

Precision	t	CPU Time [μs]	FPGA Time [μs]	Solution Time Reduction
single	0.1	45	12.347	72.56%
	0.2	89	24.684	72.26%
	0.3	135	37.021	72.57%
	0.4	181	49.358	72.72%
	0.5	230	61.695	73.17%
double	0.1	47.5	12.607	73.46%
	0.2	95	25.195	73.47%
	0.3	138	50.380	72.61%
	0.4	186	75.565	72.91%
	0.5	232	100.75	72.85%

the partial derivative of ψ over x i.e $\frac{\partial \psi}{\partial x}$. According to [44], using the method of separation of variables, its analytical solution is expressed as Eq. 8. For the numerical solution of this problem, the Point Jacobi method has been implemented. According to [45], the discretization of Eq. 7 with explicit Jacobi method yields the Eq. 9. In this equation, β is the ratio of mesh size in x and y directions, i.e. $\beta = \Delta x/\Delta y$.

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = 0 \tag{7}$$

$$\psi(x,y) = \frac{1}{2}(\pi - y) + \sum_{n=1}^{\infty} \frac{2\,((-1)^n - 1)}{\pi n^2}\,\frac{\sinh(n(\pi - y))}{\sinh(n\pi)}\cos(nx) \tag{8}$$

$$\psi_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)}\left(\psi_{i-1,j}^{k} + \psi_{i+1,j}^{k} + \beta^2(\psi_{i,j-1}^{k} + \psi_{i,j+1}^{k})\right) \tag{9}$$

The numerical solution of the Laplace equation with applying the boundary conditions of Fig. 6 and for two numerical grids of 51 × 51 and 101 × 101 was performed. Stream function contours obtained from the numerical solution by employing FPGA and CPU for double precision accuracy and for the grid size of $\Delta x = \Delta y = \pi/100$, are shown in Fig. 7 for 10,000 iterations. Furthermore, in Figure 8 the stream function contours of the analytical solution are sketched. This analytical solution was calculated from Eq. 8 with $n = 100$. In Table 3, the L^2-norm of error between the data of numerical

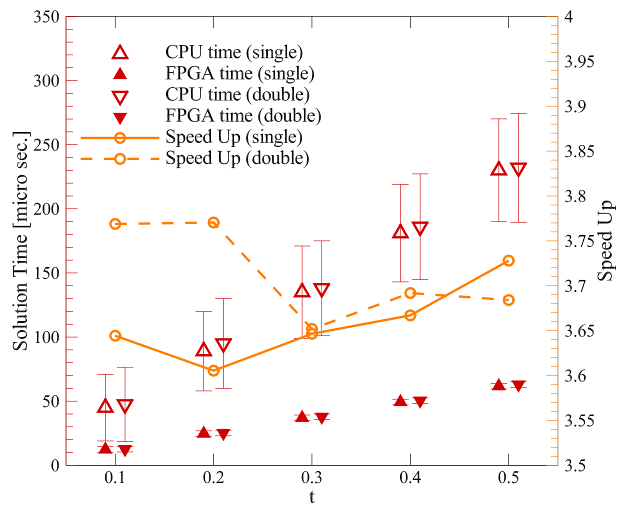

FIGURE 5. **Solution time of Eq. 5 using FPGA and CPU and the speed up.**

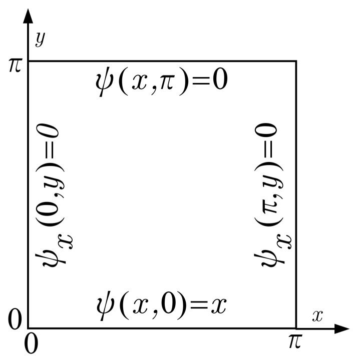

FIGURE 6. **Initial conditions of the Laplace equation.**

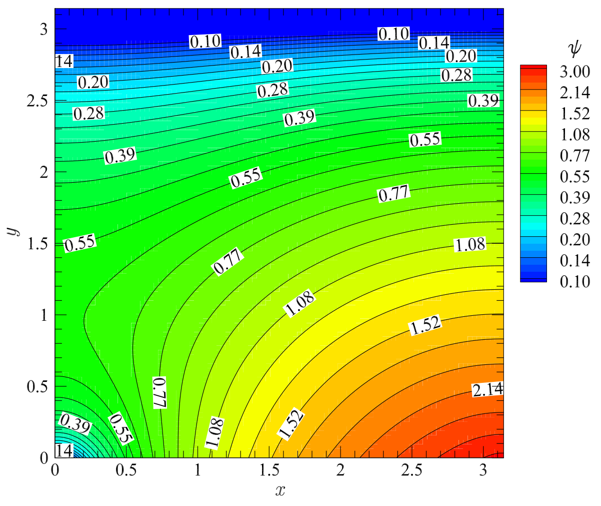

FIGURE 7. Stream function contours obtained from the numerical solution by using FPGA and CPU.

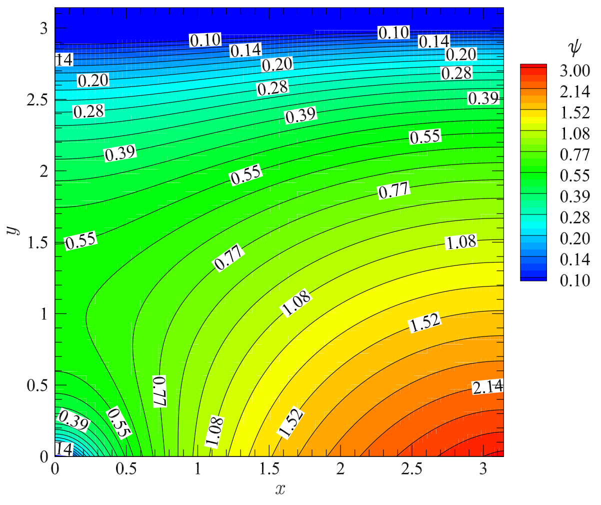

FIGURE 8. Stream function contours of the analytical solution of the Laplace equation (Eq. 8).

solution (Fig. 7) and the analytical solution (Fig. 8) are given for different mesh sizes and time steps. The maximum norm of the difference between these two solutions is 0.0073 that demonstrates the accuracy of the numerical solution. Consider two dummy matrices A and B with the same size of $m \times n$. The L^2-norm of error between these two matrices is calculated using the Eq. 10. In the numerical solution of this problem, the hardware was designed such that the solution accuracy of both FPGA and CPU being equal to each other for any data precision used.

$$L^2\text{-norm} = \left(\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(A_{i,j} - B_{i,j}\right)^2 \right)^{0.5} \tag{10}$$

The results of solution time of the Laplace equation for single and double precision floating point, different mesh sizes, and for one iteration is reported in Table 4. Moreover,

TABLE 3. The L^2-Norm of error between numerical and analytical solution of the laplace equation.

Grid Size	Iteration	L^2-norm
$\pi/50$	1000	0.0980
	5000	0.0073
	10000	0.0073
$\pi/100$	1000	0.4365
	5000	0.0601
	10000	0.0062

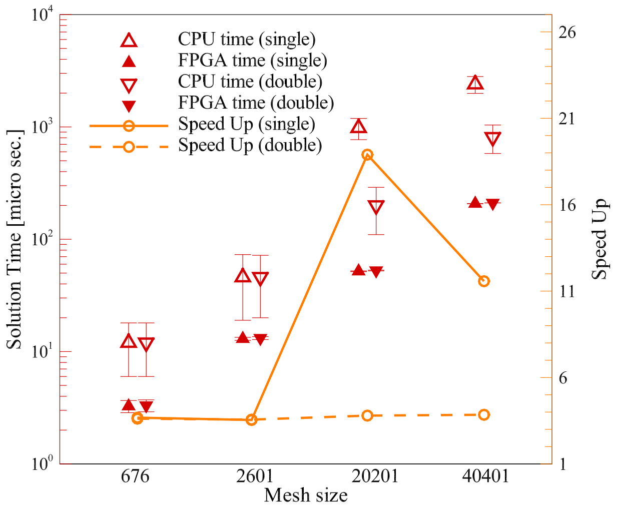

FIGURE 9. Solution time of the Laplace equation vs. grid size using FPGA and CPU and the speed up.

the amount of time solution decrease for using of FPGA is given in this table. The time solution decrease varies between 72% to 95% for different cases.

Fig. 9 illustrates the graph of solution time of the problem (left vertical axis), using CPU and FPGA with different data precisions. Also in this figure, the amount of solution speed increment is shown on the right vertical axis. The plot of numerical solution time of Laplace problem by using FPGA is shown in Fig. 10 against the number of grid points. This figure demonstrates a linear relation between solution time and the grid size. This conclusion is achieved due to the linearity of Laplace equation.

In [13] and [21], the solution of one node of the numerical grid for Laplace equation has been conducted via implementing both CPU and FPGA. The solution time results of each node of the numerical grid with a single precision data format for one iteration of Laplace solution, in the present study and those of [13] and [21] are given in Table 5. Since the hardware and their maximum processing frequency used by [13] and [21] are different from the one exploited in this work (first and second columns of Table 5), therefore solution time cannot be an appropriate criterion to compare the processing powers. For this reason, the nondimensional parameter of Clock cycles has been chosen to make such comparisons. Clock cycles can determine the computational power of a processing hardware irrespective of the employed

TABLE 4. Solution time of the laplace equation using FPGA and CPU.

Precision	Grid Size	CPU Time [μs]	FPGA Time [μs]	Solution Time Reduction
single	$\pi/25$	12	3.264	72.80%
	$\pi/50$	46	12.984	71.77%
	$\pi/100$	980	51.864	94.70%
	$\pi/200$	2400	207.384	91.35%
double	$\pi/25$	12	3.316	72.36%
	$\pi/50$	47	13.192	71.93%
	$\pi/100$	200	52.696	73.65%
	$\pi/200$	810	210.712	73.98%

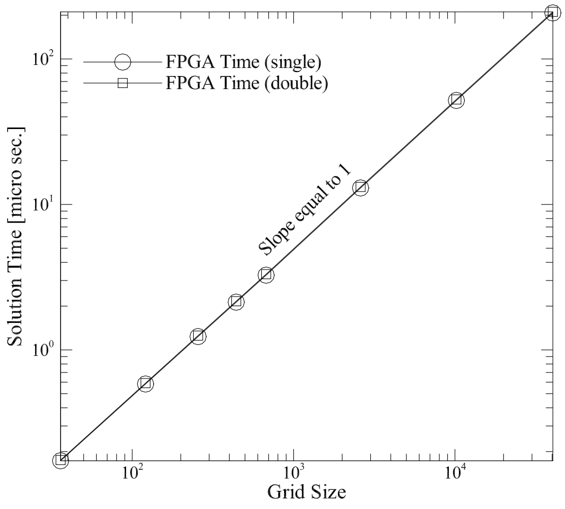

FIGURE 10. Solution time of the Laplace equation using FPGA vs. grid size.

hardware and in this regard, it takes the advantage over other dimensional variables such as the required time for the solution to finish. It is seen that the required Clock cycles for completing a solution by using FPGA are nearly equal for the three studies, while this parameter is different for a solution employing CPU.

C. QUASI-ONE-DIMENSIONAL INVISCID COMPRESSIBLE FLOW

The purpose of this section is to compute the flow properties such as velocity, density, temperature, and pressure in a quasi-one-dimensional inviscid compressible flow through a Shubin nozzle [46]. In this problem, only the velocity component in the longitudinal direction of the nozzle is considered and other directions are neglected. The cross section area, A(x), of the Shubin nozzle varies according to the relation 11, in which nozzle length is considered to be 10 ($x_{max} = 10$). Accordingly, the shape of the nozzle is as shown in Fig. 11.

$$A(x)=1.398+0.347\tanh(0.8x-4), \quad 0<x<x_{max} \tag{11}$$

The governing equation of an inviscid compressible flow are Euler equations that have been taken from [47] and are represented in matrix and nondimensional form as relation 12. In this equation, ρ, u, P and γ are the gas density, longitudinal component of the velocity vector (in the nozzle),

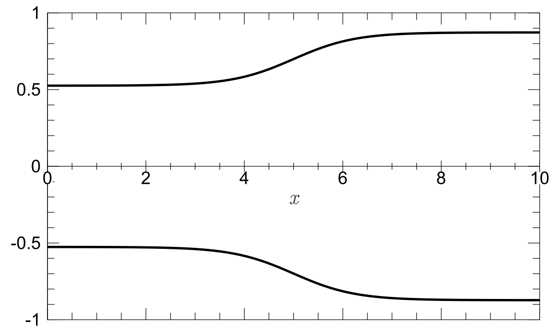

FIGURE 11. The Geometry of the Shubin nozzle.

thermodynamic pressure and heat capacity ratio, respectively.

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}=H,$$

$$U=A\begin{bmatrix}\rho\\ \rho u\\ \dfrac{P}{\gamma-1}+\dfrac{1}{2}\rho u^2\end{bmatrix},\quad F=A\begin{bmatrix}\rho u\\ \rho u^2+P\\ \dfrac{\gamma P}{\gamma-1}+\dfrac{1}{2}\rho u^2\end{bmatrix},$$

$$H=\begin{bmatrix}0\\ P\dfrac{dA}{dx}\\ 0\end{bmatrix} \tag{12}$$

Flux vector splitting method with a first-order explicit formulation has been implemented for the numerical solution. In this method, flux vector F is split into two flux vectors; right-running vector (F^+) and left-running vector (F^-). Then, the governing equations (the nondimensional Eq. 12) are discretized with a backward difference in time and first order one-sided difference in space. This process consists of the following steps:

$$\begin{aligned}&\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}=H\\ &\Longrightarrow \frac{\partial U}{\partial t}+\frac{\partial F^+}{\partial x}+\frac{\partial F^-}{\partial x}=H\\ &\Longrightarrow \frac{U_i^{n+1}-U_i^n}{\Delta t}+\frac{(F_i^+)^n-(F_{i-1}^+)^n}{\Delta x}+\frac{(F_{i+1}^-)^n-(F_i^+)^n}{\Delta x}\\ &= H_i^n\\ &\Longrightarrow U_i^{n+1}=U_i^n-\Delta t\frac{(F_i^+)^n-(F_{i-1}^+)^n}{\Delta x}\\ &\qquad -\Delta t\frac{(F_{i+1}^-)^n-(F_i^+)^n}{\Delta x}+\Delta t H_i^n\end{aligned} \tag{13}$$

TABLE 5. Solution time and clock duration of one iteration of one grid point.

Research	Hardware	Max. Frequency [MHz]	Solution Time [ns]	Clock Duration
Ref. [13]	Intel Xeon Woodcrest CPU	3000	43	130
	Xilinx Virtex 5 FPGA	822.37	1.22	1
Ref. [21]	Intel Core i7 Q-940 CPU	2930	89.11	261.1
	Altera Stratix III EP3SL150 FPGA	133	7.5	1
Present Study	Intel Core i7 Q-740 CPU	1730	18.77	32.472
	Xilinx Zynq-7020 FPGA	250	4.83	1.2071

According to [48], right-running and left-running flux vectors are written as below:

$$F^{+} = A\frac{\rho}{2\gamma}\begin{bmatrix} 2\gamma u + c - u \\ 2(\gamma-1)u^2 + (u+c)^2 \\ (\gamma-1)u^3\frac{(u+c)^3}{2} + \frac{(3-\gamma)(u+c)c^2}{2(\gamma-1)} \end{bmatrix} \tag{14}$$

$$F^{-} = A\frac{\rho}{2\gamma}\begin{bmatrix} u - c \\ (u-c)^2 \\ \frac{(u-c)^3}{2} + \frac{(3-\gamma)(u-c)c^2}{2(\gamma-1)} \end{bmatrix} \tag{15}$$

Where c is the speed of sound. The numerical solution of the nozzle flow has been performed for two different series of boundary conditions as is explained in the following:

1) Supersonic inlet and outlet flow (an isentropic flow without shock wave)
2) Supersonic inlet flow and subsonic outlet (shock wave inside the nozzle)

Inlet conditions are considered as below for both cases:

$$\rho_{in} = 0.5008261, \quad P_{in} = 0.27129$$
$$u_{in} = 1.099184, \quad \text{Mach Number} = 1.262214 \tag{16}$$

The supersonic outlet conditions are determined as following:

$$P_{out} = 0.5156 \tag{17}$$

Also, the subsonic outlet conditions are determined as following:

$$\rho_{out} = 0.7511383, \quad u_{out} = 0.4416178 \tag{18}$$

For the case in which the outlet boundary is subsonic, a shock wave appears inside the nozzle. Capturing the discontinuity in the flow correctly would be a challenge for our numerical solution method.

A one-sided scheme has been used to apply the boundary conditions to the flow field. The L^2-norm difference (See Eq. 10) of two consecutive solutions is considered as the convergence criterion. The convergence history of the problem is illustrated in Fig. 12. The comparison between the results of the exact solution and numerical solution with double precision data format is given in Fig. 13 for the grid size of $\Delta x = 0.1$ and time step of $\Delta t = 0.01$ (CFL=0.2). The exact solution of this problem was obtained using isentropic flow and normal shock wave relations from [49]. Acceptable agreement between the two solutions of Fig. 13 indicates the credibility of the numerical solution.

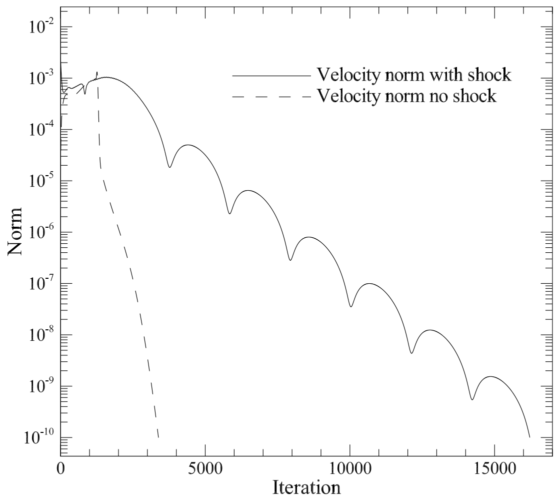

FIGURE 12. Convergence history of velocity magnitude in the solution of Eq. 12.

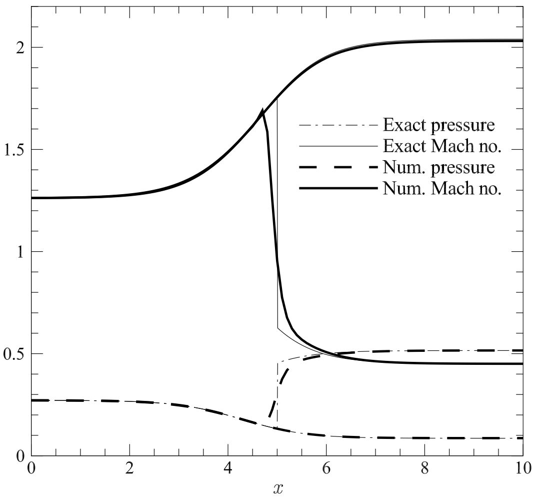

FIGURE 13. The numerical solution of Eq. 12 and the exact solution.

In Table 6 the results of solution time for different iterations of the problem, for the case that a shock wave forms inside the nozzle and with different floating point data precisions (single and double) with the grid size of $\Delta x = 0.1$ and time step of $\Delta t = 0.01$ are given. The graph of solution time of the problem (left vertical axis), using CPU and FPGA and for different floating point and data precisions is plotted in Fig. 14 against different iterations. Furthermore, in this figure, the amount of solution speed increment is displayed over the right vertical

TABLE 6. Solution time of the Eq. 12.

Precision	Iteration	CPU Time [ms]	FPGA Time [ms]	Solution Time Reduction
single	5000	246.641	36.722	85.11%
	10000	523.457	73.445	85.96%
	15000	784.469	110.168	85.95%
	20000	995.770	146.891	85.24%
double	5000	354.839	39.050	88.99%
	10000	705.385	78.100	88.92%
	15000	1069.599	117.151	89.04%
	20000	1446.612	156.201	89.20%

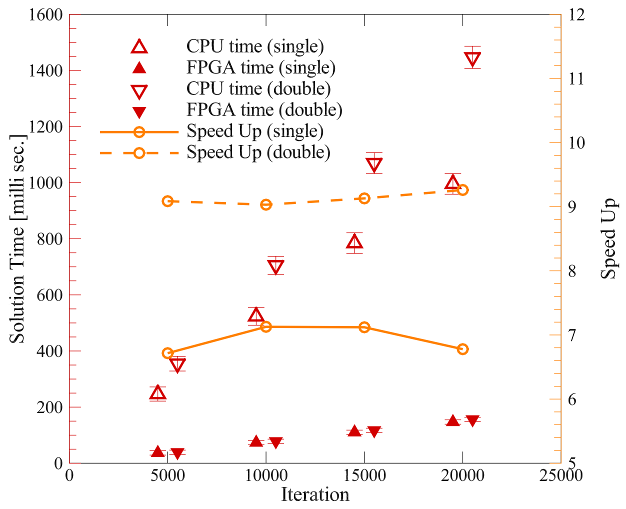

FIGURE 14. Solution time of Eq. 12 and the speed up.

axis which shows that the solution by FPGA has achieved a speed increment up to 9 times than the CPU case.

VI. CONCLUSION

The main purpose of the present study was to improve the solution speed of different problems by employing a configurable FPGA hardware. The hardware used for the numerical calculations in this study was a Zynq-7020 that its reconfigurable unit can operate at a maximum processing frequency of 250 MHz. To construct the FPGA architecture, some IP blocks were built using codes written in C++ programming language and then by arranging these IPs besides each other, the final configuration for performing the calculations were established. This is a high-level hardware design process which is a simpler method in comparison with programming via hardware description languages. Three numerical problems were chosen to be solved using this hardware that includes an ordinary second order differential equation, the Laplace equation and quasi-one-dimensional inviscid compressible flow (one-dimensional Euler equation). For all cases, the results revealed that using FPGA improves the solution speed in comparison with using CPU (up to 20 times faster in the case of the Laplace equation). Moreover, in the case of employing more powerful reconfigurable hardware for conducting the computations, a better improvement of computational speed will be obtained. In the present work, the effect of programming methods and also the use of lower precision data formats on reducing the solution time have not been studied. Moreover, further improvement of numerical solution speed can be achieved by paralleling several FPGAs and exploiting them as one unit to performing computations.

REFERENCES

[1] J. D. Anderson, *Computational Fluid Dynamics*. New York, NY, USA: McGraw-Hill, 1995.

[2] N. Kroll and J. K. Fassbender, Eds., *MEGAFLOW—Numerical Flow Simulation for Aircraft Design* (Notes on Numerical Fluid Mechanics and Multidisciplinary Design). Berlin, Germany: Springer, 2006.

[3] F. T. Johnson, E. N. Tinoco, and N. J. Yu, "Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle," *Comput. Fluids*, vol. 34, no. 10, pp. 1115–1151, 2005.

[4] D. Ball, "Recent applications of CFD to the design of Boeing commercial transports," in *Proc. HPC User Forum*, Roanoke, VA, USA, 2009, pp. 1–27.

[5] E. M. Kraft, "After 40 years why hasn't the computer replaced the wind tunnel?" *ITEA J. Test Eval.*, vol. 31, pp. 329–346, Sep. 2010.

[6] H. Morishita, Y. Osana, N. Fujita, and H. Amano, "Exploiting memory hierarchy for a computational fluid dynamics accelerator on FPGAs," in *Proc. Int. Conf. ICECE Technol. (FPT)*, Dec. 2008, pp. 193–200.

[7] D. Caughey and M. Hafez, *Frontiers of Computational Fluid Dynamics* (Computational Fluid Dynamics). Singapore: World Scientific, 2005.

[8] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[9] A. Corrigan, F. F. Camelli, R. Löhner, and J. Wallin, "Running unstructured grid-based CFD solvers on modern graphics hardware," *Int. J. Numer. Methods Fluids*, vol. 66, no. 2, pp. 221–229, 2011.

[10] S. Dong and G. E. Karniadakis, "Dual-level parallelism for high-order CFD methods," *Parallel Comput.*, vol. 30, no. 1, pp. 1–20, 2004.

[11] W. D. Smith and A. R. Schnore, "Towards an RCC-based accelerator for computational fluid dynamics applications," *J. Supercomput.*, vol. 30, no. 3, pp. 239–261, 2004.

[12] T. Hauser, "A flow solver for a reconfigurable FPGA-based hypercomputer," in *Proc. 43rd AIAA Aeros. Sci. Meeting Exhibit*, Jan. 2005, pp. 1382.

[13] R. C. Núñez, J. G. Gonzalez, and J. A. Camberos, "Large-scale numerical solution of partial differential equations with reconfigurable computing," in *Proc. 18th AIAA Comput. Fluid Dyn. Conf.*, Jun. 2007, p. 4085.

[14] K. Sano, T. Iizuka, and S. Yamamoto, "Systolic architecture for computational fluid dynamics on FPGAs," in *Proc. 15th Annu. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2007, pp. 107–116.

[15] J. Dongarra, G. Peterson, S. Tomov, J. Allred, V. Natoli, and D. Richie, "Exploring new architectures in accelerating CFD for air force applications," in *Proc. DoD HPCMP Users Group Conf. (DOD HPCMP UGC)*, 2008, pp. 472–478.

[16] J. Sun, G. D. Peterson, and O. O. Storaasli, "High-performance mixed-precision linear solver for FPGAs," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1614–1623, Dec. 2008.

[17] E. Andrés, C. Carreras, G. Caffarena, M. del Carmen Molina, O. Nieto-Taladriz, and F. Palacios, "A methodology for CFD acceleration through reconfigurable hardware," in *Proc. 46th AIAA Aerosp. Sci. Meeting Exhibit*, Jan. 2008, pp. 1–20.

[18] K. Inakagata, H. Morishita, Y. Osana, N. Fujita, and H. Amano, "Modularizing flux limiter functions for a computational fluid dynamics accelerator on FPGAs," in *Proc. Int. Conf. Field Program. Logic Appl.*, 2009, pp. 654–657.

[19] E. Andrés, M. Widhalm, and A. Caloto, "Achieving high speed CFD simulations: Optimization, parallelization, and FPGA acceleration for the unstructured DLR TAU code," in *Proc. 47th AIAA Aerosp. Sci. Meeting Including New Horizons Forum Aerosp. Expo.*, Jan. 2009, pp. 1–20.

[20] D. Sanchez-Roman, G. Sutter, S. Lopez-Buedo, I. Gonzalez, F. J. Gomez-Arribas, and J. Aracil, "An Euler solver accelerator in FPGA for computational fluid dynamics applications," in *Proc. 7th Southern Conf. Program. Logic (SPL)*, 2011, pp. 149–154.

[21] K. Sano, Y. Hatsuda, and S. Yamamoto, "Performance evaluation of FPGA-based custom accelerators for iterative linear-equation solvers," in *Proc. 20th AIAA Comput. Fluid Dyn. Conf.*, Jun. 2011, pp. 1–8.

[22] I. Liu, E. A. Lee, M. Viele, G. Wang, and H. Andrade, "A heterogeneous architecture for evaluating real-time one-dimensional computational fluid dynamics on FPGAs," in *Proc. IEEE 20th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr./May 2012, pp. 125–132.

[23] *Field Programmable Gate Array Chips: History*. [Online]. Available: https://web.archive.org/web/20070412183416/ and http://filebox.vt.edu/users/tmagin/history

[24] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug./Sep. 2009, pp. 126–131.

[25] D. Crookes, K. Benkrid, A. Bouridane, K. Alotaibi, and A. Benkrid, "Design and implementation of a high level programming environment for FPGA-based image processing," *IEE Proc.-Vis., Image Signal Process.*, vol. 147, no. 4, pp. 377–384, Aug. 2000.

[26] C. Dick and F. Harris, "FPGA signal processing using sigma-delta modulation," *IEEE Signal Process. Mag.*, vol. 17, no. 1, pp. 20–35, Jan. 2000.

[27] B. Block, P. Virnau, and T. Preis, "Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model," *Comput. Phys. Commun.*, vol. 181, no. 9, pp. 1549–1556, 2010.

[28] M. Weigel, "Performance potential for simulating spin models on GPU," *J. Comput. Phys.*, vol. 231, no. 8, pp. 3064–3082, 2012.

[29] N. Shirazi, A. Walters, and P. Athanas, "Quantitative analysis of floating point arithmetic on FPGA based custom computing machines," in *Proc. IEEE Symp. FPGAs Custom Comput. Mach.*, Apr. 1995, pp. 155–162.

[30] K. R. Nichols, M. A. Moussa, and S. M. Areibi, "Feasibility of floating-point arithmetic in FPGA based artificial neural networks," in *Proc. CAINE*, 2002, pp. 8–13.

[31] M. A. Zidan, A. G. Radwan, and K. N. Salama, "The effect of numerical techniques on differential equation based chaotic generators," in *Proc. ICM*, Dec. 2011, pp. 1–4.

[32] N. Margolus, "An FPGA architecture for DRAM-based systolic computations," in *Proc. 5th Annu. IEEE Symp. Field-Program. Custom Comput. Mach.*, Apr. 1997, pp. 2–11.

[33] W. Chen, P. Kosmas, M. Leeser, and C. Rappaport, "An FPGA implementation of the two-dimensional finite-difference time-domain (FDTD) algorithm," in *Proc. ACM/SIGDA 12th Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2004, pp. 213–222.

[34] W. Sun, M. J. Wirthlin, and S. Neuendorffer, "FPGA pipeline synthesis design exploration using module selection and resource sharing," *IEEE Trans. Comput.-Aided Des. Integr.*, vol. 26, no. 2, pp. 254–265, Feb. 2007.

[35] R. Lysecky and F. Vahid, "A study of the speedups and competitiveness of FPGA soft processor cores using dynamic hardware/software partitioning," in *Proc. Design, Autom. Test Eur.*, vol. 1. Mar. 2005, pp. 18–23.

[36] Y. Lin, F. Wang, X. Zheng, H. Gao, and L. Zhang, "Monte Carlo simulation of the Ising model on FPGA," *J. Comput. Phys.*, vol. 237, pp. 224–234, Mar. 2013.

[37] G. Estrin, "Reconfigurable computer origins: The UCLA fixed-plus-variable (F+V) structure computer," *IEEE Ann. Hist. Comput.*, vol. 24, no. 4, pp. 3–9, Oct. 2002.

[38] M. B. Gokhale and P. S. Graham, *Reconfigurable Computing: Accelerating Computation With Field-Programmable Gate Arrays*. Berlin/Heidelberg, Germany: Springer, 2005.

[39] E. Christen and K. Bakalar, "VHDL-AMS—A hardware description language for analog and mixed-signal applications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 10, pp. 1263–1272, Oct. 1999.

[40] A. Canis *et al.*, "LegUp: High-level synthesis for FPGA-based processor/accelerator systems," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2011, pp. 33–36.

[41] Xilinx.com. *Intellectual Property*, accessed on Jun. 16, 2016. [Online]. Available: https://www.xilinx.com/products/intellectual-property.html

[42] K. Atkinson, W. Han, and D. E. Stewart, *Numerical Solution of Ordinary Differential Equations* (Pure and Applied Mathematics: A Wiley Series of Texts, Monographs, and Tracts). Hoboken, NJ, USA: Wiley, 2011.

[43] M. Y. Kamil, A. A. Al-Zuky, and R. S. Al-Tawil, "Study of experimental simple pendulum approximation based on image processing algorithms," *Appl. Phys. Res.*, vol. 3, no. 1, p. 29, 2011.

[44] T. Myint-U and L. Debnath, *Linear Partial Differential Equations for Scientists and Engineers*. Boston, MA, USA: Birkhäuser, 2007.

[45] K. A. Hoffmann and S. T. Chiang, *Computational Fluid Dynamics for Engineers*, vol. 1, ed. 2. Wichita, KS, USA: Engineering Education System, 1993.

[46] G. R. Shubin, A. B. Stephens, and H. M. Glaz, "Steady shock tracking and Newton's method applied to one-dimensional duct flow," *J. Comput. Phys.*, vol. 39, no. 2, pp. 364–374, 1981.

[47] H. C. Yee, R. M. Beam, and R. F. Warming, "Boundary approximations for implicit schemes for one-dimensional inviscid equations of gasdynamics," *AIAA J.*, vol. 20, no. 9, pp. 1203–1211, Sep. 1982.

[48] J. L. Steger and R. F. Warming, "Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods," *J. Comput. Phys.*, vol. 40, no. 2, pp. 263–293, 1981.

[49] J. D. Anderson, *Fundamentals of Aerodynamics* (Anderson). New York, NY, USA: McGraw-Hill, 2011.

ABBAS EBRAHIMI received the Ph.D. degree in aerodynamic from the Sharif University of Technology, Tehran, Iran. He is currently an Assistant Professor with the Aerospace Engineering Department, Sharif University of Technology. His research interests include CFD, applied aerodynamics, unsteady aerodynamics, and wind tunnel testing.

MOHAMMAD ZANDSALIMY is currently pursuing the Ph.D. degree with the Aerospace Engineering Department, Sharif University of Technology, Tehran, Iran. His research interests include CFD, numerical modeling and simulation, applied aerodynamics, and unsteady aerodynamics.