High-Performance Hardware Accelerators for Solving Ordinary Differential Equations

***Abstract*—Ordinary Differential Equations (ODEs) are widely used in many high-performance computing applications. However, contemporary processors have limited throughput in these kinds of calculations. A high-performance hardware accelerator has been developed for speeding-up the solution of ODEs. The hardware accelerator has been developed both for single and double floating-point precision types and a design-space exploration has been performed in terms of performance and hardware resources. The hardware accelerator has been mapped to an FPGA board and connected through PCIe to a typical processor. The performance evaluation shows that the proposed scheme can achieve up to 12x speedup compared to a reference single core CPU solution.**

I. Introduction

Systems of ordinary differential equations (ODEs) play an important role in modelling dynamically changing phenomena and evolutionary processes mathematically. Many ODEs of practical interest are nonlinear and, hence, they lack a closed-form solution so that numerical methods are required to compute approximate solutions [1].

High-performance computing (HPC) applications need powerful infrastructures to solve millions of ODEs. Contemporary processors have limited performance in solving these kinds of equations. To reduce the execution, specialized accelerators are required that can speedup the execution time. However, most of the hardware accelerators for ODE solvers are fixed and required significant effort to be customized to specific applications.

In [2] a custom FPGA processor (DEPS) has been presented that is used for the efficient solution of ODEs on FPGAs. A single DEPE on a Xilinx Virtex6 FPGA executes several physiological models faster than real-time while requiring only a few hundred FPGA look-up tables (LUTs). The presented architecture show that a single DEPE is 550 slower than HLS based ODE solvers circuits however the DEPE is 10200 smaller in terms of hardware resources. However, the FPGA processor cannot meet the high throughput requirements that is required by HPC applications.

In this paper, we present a family of high-performance and configurable hardware accelerators that can reduce significantly the execution time of the ODE solvers and can be customized to meet the requirements of several ODE solvers. The accelerators for the ODE solvers have been implemented in reconfigurable logic (FPGA) and have been mapped to a PCIe FPGA board connected with an contemporary Intel processor. Performance evaluation shows that the hardware accelerator can achieve up to 11x speedup compared to a typical desktop processors.

II. ODE Solvers

In this paper, we will discuss an implementation of the Predator-Prey system. This systems models the population sizes of two species [3]. Let u and v denote the population sizes relative to each other. The dynamic interplay of both species is then given, for a specific model, by the following 2x2 ODE system:

$$\frac{du}{dt} = f(u, v, t) \tag{1}$$

$$\frac{dv}{dt} = g(u, v, t) \tag{2}$$

Here, the f and g functions are given by:

$$f(u, v, t) = 0.1 \cdot u - 0.2 \cdot u \cdot v \tag{3}$$

$$g(u, v, t) = -0.2 \cdot v + 0.4 \cdot u \cdot v \tag{4}$$

Other 2x2 ODE systems with variations on this Predator-Prey systems, change of constants or a simple model problem that describes the dynamic interaction of two species of bacteria that compete for the same supply of food [4], are expected to have similar performance results.

For the numerical solution of the above model problem we adopt four widely-used ODE solvers, namely, forward Euler (FwdEuler), modified Euler (ModEuler), and strong stability preserving Runge-Kutta schemes of order two (SSP-RK2) and three (SSP-RK3) [5], respectively. All four methods are fully explicit and thus, lend themselves to parallel implementations. The computational complexity increases from the single-step forward Euler method to the two- and three-step Runge-Kutta methods, so that the speedup of the FPGA implementation over the software solution is expected to be highest for FwdEuler but still considerable for ModEuler, SSP-RK2 and SSP-RK3. The benefit of the latter is, however, their higher temporal accuracy. Furthermore, the selected ODE solvers are used as time-integrators for partial differential equations (PDEs), especially hyperbolic conservation laws, and therefore, the development of efficient FPGA-accelerated solution procedures has the potential to significantly speed-up the numerical simulations of PDE problems at a later stage.

The main contributions of this paper are the followings:

- A high performance accelerator for solving ODEs that can achieve up to 12x speedup
- A design space exploration of single and double floating point architectures

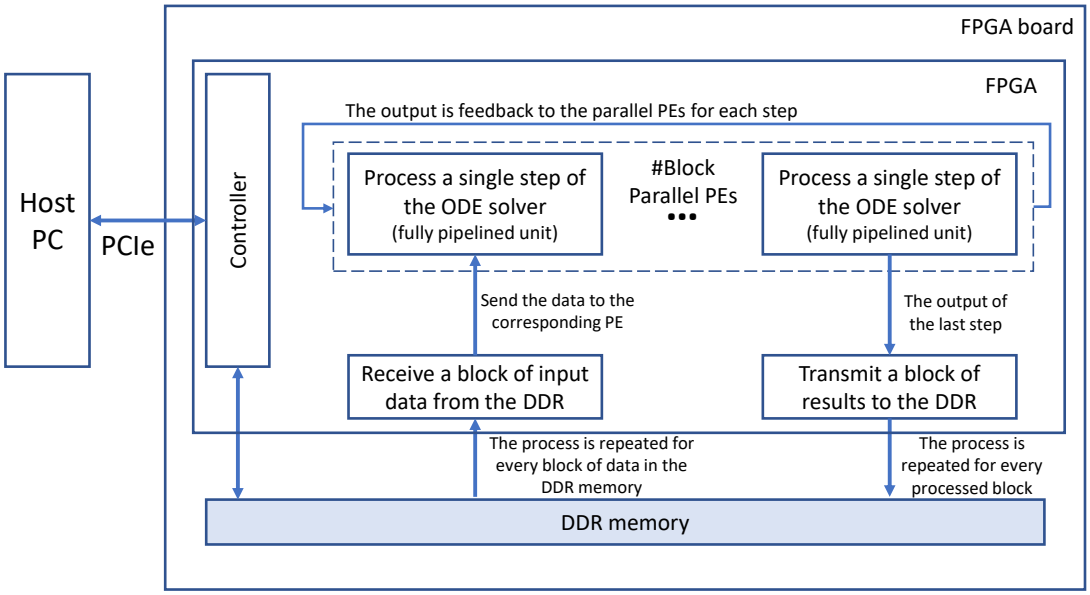

Fig. 1. Architecture of the ODE solvers hosted in an FPGA board

- A performance evaluation of 4 different solvers in terms of throughput, speedup and hardware resources.

III. Implementation

A. Kernel

The input variables of the kernel on the FPGA are: dt, number of steps, the initial values for each u and v and the coefficients of the f and g functions. The output variables are values of each u and v after the number of steps. Per block the elements are store on the DDR RAM on the FPGA board. This leads to an implementation which can be executed for different scenarios.

The selected ODE solvers were implemented using HLS in single- and double-precision floating-point arithmetic. In the kernel on the FPGA single units are created which calculate the values of the next step for a pair of u and v, with the use of the f and g functions. To increase the performance multiple units will process the data in parallel for multiple pairs of u's and v's. The pairs of u's and v's which can be executed in parallel will be referred to as a block. Because of the limited amount of resources on the FPGA it is possible that not all variables are able to fit into a single block. If this is the case first the final values of the block are calculated after which the next block is retrieved from the DDR RAM and then this block is processed. This continues until the final number of steps is reached. If not all the elements are able to fit in the memory the memory will replace the blocks in the DDR Ram. This whole process will repeat until all elements are processed.

For each ODE solver the maximum number of processing units is determined, which is constrained by limited resources on the FPGA. For each of the presented experiments we choose the optimal amount of blocks that fits the amount of elements of the simulation. Additionally, the communication overhead is minimized by transferring as much blocks at once as possible from the host CPU to the FPGA.

Figure 1 presents the architecture of the implemented system. Between the four different ODE solvers only the parallel processing units differ. The Host PC continues to send sets of blocks to the DDR memory until the entire amount of the element is processed. The directive used in the Xilinx Vivado High-Level Synthesis (HLS) development platform to achieve the described architecture is `#pragma HLS PIPELINE` in each sub-function except the main procedure, where we use `#pragma HLS UNROLL` for the nested loop, `# pragma HLS PIPELINE` for the loop of the steps and `#pragma HLS ARRAY PARTITION dim=1` for the memory that stores the input and output data. Also, directives are used for the interface of the top function to be implemented with the Xilinx AXI protocol.

B. Host

The access of the accelerator kernels, from the Host CPU, can be achieved by using six functions that were developed using C++. Those functions are responsible for creating the CPU-FPGA connection, programming the FPGA board with the correct kernel, initializing the memories required for the communication, sending the element values, receiving the results, breaking the link between the CPU and the FPGA and freeing the reserved CPU resources. At system startup, the initialization function must be executed once. This function creates the device descriptors used for read/write to the accelerator and also reserves the memory required for the packets that are going to be communicated between the CPU and the FPGA. This process should be avoided to run at each call of

TABLE I
VALUES USED TO CALCULATE REFERENCE SOLUTION USING THE FORWARD EULER SOLVER.

Number of steps	10^7
dt	10^{-5}
u(t = 0)	0.20
v(t = 0)	1.10

the accelerator functions due to the large execution overhead (6-7 seconds).

To process data with one of the accelerators (FwdEuler, ModEuler, SSP-RK2 or SSP-RK3) the corresponding function must be called, that transmits the data to the kernel and receives the results. During the execution of the accelerator functions, a set of element blocks is transmitted to the DDR RAM memories of the FPGA board, then the kernel processes each block sequentially. When all the elements of a block are processed the results are transmitted to the DDR RAM memories and a next block is received for process. When all blocks are processed the results are transmitted back to the Host and a new batch of blocks is received. When the system is about to be terminated or the accelerators are no longer needed, the terminating function releases the buffers and closes the device descriptors, this process takes around 0.06ms.

The SDAccel tool from Xilinx was used for the integration of the kernels to a final system [6]. The SDAccel tool provides a framework for the development of an entire system consisting of the Host running on a CPU and the kernels running on a FPGA. A PCIe connection is utilized for the communication of the Host and the Kernel.

IV. RESULTS & EVALUATION

Each ODE solver gives a different global error depending on the number of steps. The number of steps needed to simulate up until time T with an certain accuracy will be different for the different solvers. To make a fair comparison all the solvers are executed with as less steps as possible while below a certain error. To calculate an reference, which is used as an approximation of the exact solution, the values of *u* and *v* are calculated after 10 seconds with a large number of steps and small *dt*. The number of steps, *dt*, and the initial values for *u* and *v* can be seen in table I. Then the simulation is done with ten thousand steps and a *dt* equal to 1 millisecond, using the forward Euler method. The difference between those results and the results of the reference gives us a divided by two approximation of the average error per element. This error is used as the maximum allowable error while minimizing the number of steps, in steps of 4, of the other ODE solvers. The results can be seen in table II. Those results shows that with the use of higher order solvers a significant reduction in number of steps can be achieved while maintaining the same or better accuracy. This justifies the use of more equations which are needed for higher order solvers in terms of performance.

The Host CPU used is an Intel Core i5-4590 @ 3.30GHz with 4GB RAM and CentOS 7 operating system. The FPGA board is the Alpha Data ADM-PCIE-KU3 board, featuring a Xilinx Kintex Ultrascale (XCKU060 - FFVA1156) FPGA. The FPGA is clocked at a frequency of 200 MHz.

TABLE II
NUMBER OF STEPS, DT, AND AND THE AVERAGE ERROR PER ELEMENT FOR EACH SOLVER. THE AVERAGE ERROR IS CALCULATED IN COMPARISON WITH THE REFERENCE SIMULATION.

Solver	**Number of steps**	**dt**	**Average error**
FwdEuler	10^4	10^{-3}	$2.01 \cdot 10^{-5}$
ModEuler	60	0.17	$1.86 \cdot 10^{-5}$
SSP-RK2	60	0.17	$1.86 \cdot 10^{-5}$
SSP-RK3	12	0.83	$1.92 \cdot 10^{-5}$

For each solver the maximum number of processing units and the maximum number blocks stored in the DDR RAM on the FPGA are determined for both single as double floating point precision. Those results together with the maximum number of use blocks per kernel iteration can be seen in Table III for FwdEuler, Table IV for ModEuler, Table V for SSP-RK2, and Table VI for SSP-RK3. For all the simulations all the processing units are used, together with the used blocks.

Additionally the resources used on the FPGA are shown in Table VII, Table VIII, Table IX and Table X for each of the ODE solvers. Those resources are the estimations for the kernels, as produced from the SDAccel tool; the PCIe controller was not considered. From those tables we can see the increased resources requirements between the single and double precision implementations and between the ODE solvers.

As expected, when double precision is used the amount of parallel processing units decreases, due to higher FPGA resources requirements, which will limit the achievable speedup of the accelerator. The same holds for the rest of the ODE accelerators. Because the second-order and third-order strong-stability-preserving Runge-Kutta ODE solvers require more operations in comparison with the Euler solvers, the usage of FPGA resources for the Runge-Kutta ODE solvers is increased. As a result the amount of parallel processing units decreases when more operations are needed for the ODE solver. An further increase on the parallel processing units or the number of blocks that are stored to the DDR memories, for each of the ODE solvers, will cause the operating frequency to drop under 200 MHz, which is the targeted operating frequency.

We measured the execution time of each solver from one thousand up to one hundred million elements. For the measurements, the function *gettimeofday* is used. Those measurements are done with double floating point precision on the host CPU and with both single and double precision on the FPGA. The results can be found in Table XI, Table XII, Table XIII, and Table XIV for respectively the FwdEuler, the ModEuler, the SSP-RK2, and the SSP-RK3 ODE solver. Those results shows that a higher order solver need less time to execute. This is because a higher order solver need less steps to simulate, while keeping below a certain threshold.

TABLE III
FOR THE FWDEULER IMPLEMENTATION THE NUMBER OF PROCESSING UNITS, MAXIMUM NUMBER OF BLOCKS WHICH CAN BE STORED, AND NUMBER OF USED BLOCKS IN THE SIMULATIONS GIVEN FOR DIFFERENT NUMBER OF ELEMENTS. FOR SINGLE FLOATING POINT AND DOUBLE FLOATING POINT PRECISION.

	Number of elements	Float	Double
Processing units	All	120	60
Maximum block storage		240	140
Used blocks	10^3	9	17
	10^4	84	140
	Rest	240	140

TABLE IV
FOR THE MODEULER IMPLEMENTATION THE NUMBER OF PROCESSING UNITS, MAXIMUM NUMBER OF BLOCKS WHICH CAN BE STORED, AND NUMBER OF USED BLOCKS IN THE SIMULATIONS GIVEN FOR DIFFERENT NUMBER OF ELEMENTS. FOR SINGLE FLOATING POINT AND DOUBLE FLOATING POINT PRECISION.

	Number of elements	Float	Double
Processing units	All	120	40
Maximum block storage		170	220
Used blocks	10^3	9	25
	10^4	84	140
	Rest	170	220

TABLE V
FOR THE SSP-RK2 IMPLEMENTATION THE NUMBER OF PROCESSING UNITS, MAXIMUM NUMBER OF BLOCKS WHICH CAN BE STORED, AND NUMBER OF USED BLOCKS IN THE SIMULATIONS GIVEN FOR DIFFERENT NUMBER OF ELEMENTS. FOR SINGLE FLOATING POINT AND DOUBLE FLOATING POINT PRECISION.

	Number of elements	Float	Double
Processing units	All	80	60
Maximum block storage		220	167
Used blocks	10^3	13	17
	10^4	125	167
	Rest	220	167

TABLE VI
FOR THE SSP-RK3 IMPLEMENTATION THE NUMBER OF PROCESSING UNITS, MAXIMUM NUMBER OF BLOCKS WHICH CAN BE STORED, AND NUMBER OF USED BLOCKS IN THE SIMULATIONS GIVEN FOR DIFFERENT NUMBER OF ELEMENTS. FOR SINGLE FLOATING POINT AND DOUBLE FLOATING POINT PRECISION.

	Number of elements	Float	Double
Processing units	All	80	1
Maximum block storage		200	up to 100m
Used blocks	10^3	13	10^3
	10^4	125	10^4
	Rest	200	# of elements

TABLE VII
FPGA RESOURCES FOR THE FWDEULER SOLVER

Elements	**FF**	**LUT**	**DSP**	**BRAM** (36kb)
FPGA Float	53k	36k	182	2
FPGA Double	72k	46k	372	4

TABLE VIII
FPGA RESOURCES FOR THE MODEULER SOLVER

Elements	**FF**	**LUT**	**DSP**	**BRAM** (36kb)
FPGA Float	54k	38k	187	2
FPGA Double	57k	41k	406	4

TABLE IX
FPGA RESOURCES FOR THE SSP-RK2 SOLVER

Elements	**FF**	**LUT**	**DSP**	**BRAM** (36kb)
FPGA Float	49k	36k	271	2
FPGA Double	80k	54k	594	4

TABLE X
FPGA RESOURCES FOR THE SSP-RK3 SOLVER

Elements	**FF**	**LUT**	**DSP**	**BRAM** (36kb)
FPGA Float	57k	41k	370	2
FPGA Double	69k	50k	916	4

TABLE XI
EXECUTION TIME (MS) FOR THE FWDEULER SOLVER

Elements	**1k**	**10k**	**100k**	**1m**	**10m**	**100m**
CPU	$1.51{\cdot}10^2$	$1.51{\cdot}10^3$	$1.51{\cdot}10^4$	$1.51{\cdot}10^5$	$1.51{\cdot}10^6$	$1.51{\cdot}10^7$
FPGA Float	14.33	$1.28{\cdot}10^2$	$1.45{\cdot}10^3$	$1.27{\cdot}10^4$	$1.26{\cdot}10^5$	$1.26{\cdot}10^6$
FPGA Double	41.80	$6.76{\cdot}10^2$	$4.07{\cdot}10^3$	$4.07{\cdot}10^4$	$4.04{\cdot}10^5$	$4.05{\cdot}10^6$

TABLE XII
EXECUTION TIME (MS) FOR THE MODEULER SOLVER

Elements	**1k**	**10k**	**100k**	**1m**	**10m**	**100m**
CPU	2.06	31.85	$1.98{\cdot}10^2$	$1.93{\cdot}10^3$	$1.92{\cdot}10^4$	$1.92{\cdot}10^5$
FPGA Float	0.84	3.00	24.59	$2.43{\cdot}10^2$	$2.37{\cdot}10^3$	$2.39{\cdot}10^4$
FPGA Double	1.57	10.61	98.29	$9.06{\cdot}10^2$	$9.10{\cdot}10^3$	$9.15{\cdot}10^4$

TABLE XIII
EXECUTION TIME (MS) FOR THE SSP-RK2 SOLVER

Elements	**1k**	**10k**	**100k**	**1m**	**10m**	**100m**
CPU	1.79	44.38	173.5	$1.73{\cdot}10^3$	$1.75{\cdot}10^4$	$1.73{\cdot}10^5$
FPGA Float	1.04	3.79	33.7	$3.02\ {\cdot}10^2$	$3.02{\cdot}10^3$	$3.06{\cdot}10^4$
FPGA Double	1.33	6.6 3	59.47	$5.86\ {\cdot}10^2$	$5.85{\cdot}10^3$	$5.91{\cdot}10^4$

Figure 2 presents the speedup achieved from the hardware acceleration of each ODE solver for the single precision floating point implementation. As the complexity of the ODE solver increases the achieved speedup decreases, because less parallel processing units can fit in the targeted FPGA and/or less blocks can be stored to the DDR memory of the board. Figure 3 presents the speedup achieved for the double precision floating point implementation. Here the achieved

TABLE XIV
EXECUTION TIME (MS) FOR THE SSP-RK3 SOLVER

Elements	1k	10k	100k	1m	10m	100m
CPU	2.13	5.6 4	70.34	$5.18{\cdot}10^2$	$5.15{\cdot}10^3$	$5.15{\cdot}10^4$
FPGA Float	0.82	2.218	18.28	$1.50{\cdot}10^2$	$1.48{*}{\cdot}10^3$	$1.48{\cdot}10^4$
FPGA Double	9.73	94.79	$9.41{\cdot}10^2$	$9.48{\cdot}10^3$	$9.50{\cdot}10^4$	$9.49{\cdot}10^5$

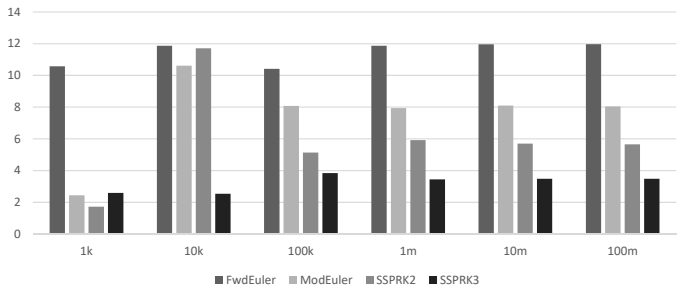

Fig. 2. Achieved speedup of the accelerated ODE solvers (single precision floating point arithmetic)

speedups are significant lower than the speedup of single floating point implementations, with the SSP-RK3 accelerator requiring more execution time than the software implementation. Although the entire set of the elements can be transferred to the DDR RAM for the double precision implementation of the SSP-RK3 solver.

V. CONCLUSIONS

In this work, we have implemented and evaluated the performance of four different ODE solvers which can, by design, be highly parallelized. The hardware accelerators that have been implemented can be tuned to meet the application requirements of several ODE solvers. The performance evaluation shows that the proposed hardware accelerators can achieve up to 12x speedup compared to contemporary processors, thus reducing significantly the execution time of the predator prey application. It can be seen that when using more resources of the FPGA, with increasing ODE-solver complexity or increasing precision, the achieved speedup decreases because less parallel processing units can fit in the targeted FPGA.

VI. FUTURE WORK

Interesting future is to pipeline the process of the blocks, that are stored to the DDR memory of the FPGA, using as intermediate buffers extra registers of the FPGA. This change in the design of the accelerator, leads to a large data dependency distance, which can further increase the performance of the hardware solvers. Additionally other applications with larger ODE systems can be tried to be accelerated with the use of an FPGA.

VII. ACKNOWLEDGMENTS

This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 687628 - VINEYARD: Versatile Integrated Accelerator-based Heterogeneous Data Centers www.vineyard-h2020.eu

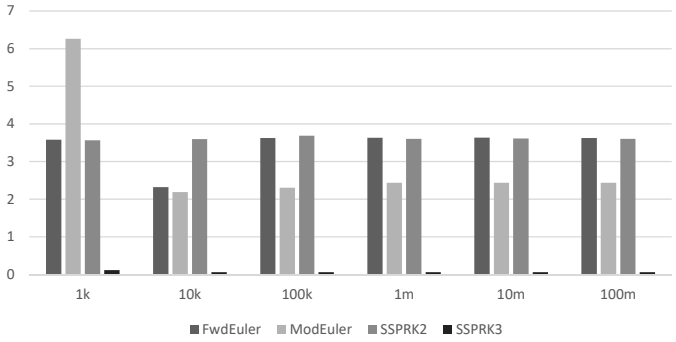

Fig. 3. Achieved speedup of the accelerated ODE solvers (double precision floating point arithmetic)

REFERENCES

[1] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.
[2] C. Huang, F. Vahid, and T. Givargis. A custom fpga processor for physical model ordinary differential equation solving. *IEEE Embedded Systems Letters*, 3(4):113–116, Dec 2011.
[3] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*, volume 40. Springer, 2001.
[4] James Dickson Murray. *Mathematical biology. I. , An introduction*. Interdisciplinary applied mathematics. Springer, New York, 2002.
[5] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev*, 43:89–112, 2001.
[6] The Xilinx SDAccel Development Environment: Bringing The Best Performance/Watt to the Data Center. Technical report, 2015.