Parallel Computer Systems Based on Numerical Integrations

Michal Kraus*
Department of Intelligent Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 1/2, 612 66 Brno, Czech Republic
kraus@fit.vutbr.cz

Abstract

This paper deals with continuous system simulation. The systems can be described by system of differential equations or block diagram. Differential equations are usually solved by numerical methods that are integrated into simulation software such as Matlab, Maple or TKSL.

Taylor series method has been used for numerical solutions of differential equations. The presented method has been proved to be both very accurate and fast and also procesed in parallel systems. The aim of the thesis is to design, implement and compare a few versions of the parallel system.

Categories and Subject Descriptors

C.1 [**Computer Systems Organization**]: Processor Architectures; G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations—*Initial value problems, One-step (single step) methods*

Keywords

numerical integration, Taylor series, parallel system, interconnection networks, integrator

1. Introduction

An important part of contemporary characteristic problems in science and technology is based on simulating and analysing complex systems such as economical models, weather forecast models or technology process control.

In many modern applications in industry simulation has become a powerful tool for engineers in order to predict real system behaviour or validate closed loop controller performance without conducting cost-intensive test cycles on appropriate test facilities. Often the specific problem setup requires the application of real-time algorithms running on external hardware units, as for example hardware-in-the-loop systems on automotive test stands. Simulation speed as well as stability of the performing algorithms are critical in such applications.

Solving most of those problems leads to complex set of linear and non-linear differential and partial differential equations with time changing parameters and large sets of algebraic and transcendental equations. For solving such problems with great emphasis on high accuracy or high speed we need high-performance computer platforms.

Today's most common mathematical simulation software packages (e.g. Matlab/Simulink) provide various types of numerical integration methods [10]. These methods differ primarily in the way the solution at the next time step is calculated, knowing the time derivative at the current time step. Variable-step solvers are able to adapt the interval step size dynamically during simulation, depending on the current rate of change of the solution.Fixed-step solvers which are used in real-time systems have a defined fixed step size because they need to calculate the simulation output deterministically for each time step. Increasing the step size, fixed-step algorithms typically become unstable at a certain step size limit. More information about numerical methods and stability can be found in [12] and [2]. Finding a numerical integration method which is accurate, fast and also robust regarding stability,would therefore increase the quality of such real-time algorithms drastically.

The processor's clock frequency is slowly reaching it's physical limits. The increase of performance is now possible only by including multiple parallel computing units. This approach is supported by form of some problems. It appears that contemporary sequential methods of solving some problems is not natural.

A very promising approach for such problems is the Modern Taylor Series Method [8] - a special parallel system which has been developed at Brno University of Technology. This parallel system can be used in a special hardware unit for the acceleration of numeric integration.

*Recommended by thesis supervisor: Assoc. Prof. Jiří Kunovský. Defended at Faculty of Information Technology, Brno University of Technology in Brno on October 10, 2013.

Kraus, M. Parallel Computer Systems Based on Numerical Integrations. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 5, No. 4 (2013) 1-7

The main component of the parallel system is a numerical integrator carrying out numerical integration based on the Taylor series. A first description of this system was published in [7].

The paper is structured as follows: A basic idea of parallel computations and analogue principles are presented in 2. Section 3 contains a detailed description of the Modern Taylor Series Method and its comparison to Runge-Kutta methods. A transformation of initial systems into systems with polynomials on the right-hand sides of the equations is also presented. In such a case the Taylor series terms can be easily calculated. Section 4 is devoted to a description of a parallel architecture developed for solving systems of differential equations. The basic part of the parallel system is a fixed-point arithmetic logic unit (ALU) designed for Taylor series numerical integration algorithm. Three versions of the ALU will be introduced. Section 5 illustrates the implementation of the parallel system on field programmable gate arrays (FPGAs) using VHDL (hardware description language) and provides tests and comparisons of the presented parallel systems based on three versions of itnegrators. Platform FITkit has been used for the implementation.

2. The Idea of Parallel Computations

Even though the idea of parallel computing and parallel connection of high amount of microprocessors is attractive, it is not easy to reach big increase in performance compared to single processor approach. The potential of parallel data processing has already been studied. It was found, that even a small percentage of sequential steps may lead to high reduction of performance of the entire system. For example if a certain program requires only 5% of instructions to be done sequentially on 64 processor machine, then the actual performance is equivalent to 15 processor machine working perfectly in parallel. This is the consequence of the fact, that most algorithms were not developed for heavy parallel systems.

This paper concentrates on large systems of parallel microprocessors. The idea of this approach comes from analogue methods of computations [9]. Analogue methods are basically parallel methods and their analysing shows that independent parallel cooperation of multiple processors may be implemented by applying differential calculus. Using analogue methods, independent parallel cooperation of microprocessors is going to be effective if each microprocessor is going to be numerically integrating.

Generally, initial problems described by autonomous systems of differential equations are in the form:

$$\begin{aligned}
w_1' &= f_1(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,), \quad w_1(0) = w_1^0,\\
w_2' &= f_2(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,), \quad w_2(0) = w_2^0,\\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots\\
w_n' &= f_n(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,), \quad w_n(0) = w_n^0,\\
x_1 &= g_1(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,),\\
x_2 &= g_2(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,),\\
&\vdots\\
x_n &= g_n(w_1, w_2, \ldots, w_n, x_1, x_2, \ldots, x_n,).
\end{aligned} \tag{1}$$

and the corresponding block diagram is in Figure 1. It is clear that integrators can work in parallel. The question now is how to calculate also functions $f_1, f_2, \ldots, f_n$ in parallel. Many test examples have been completed to confirm that, if the functions on the right-hand sides of (1) are of a particular type frequently encountered in engineering applications, a sequence of substitutions can be found that transforms the original system into a new system with polynomials on the right-hand sides. More detailed information is presented in subsection 3.1.

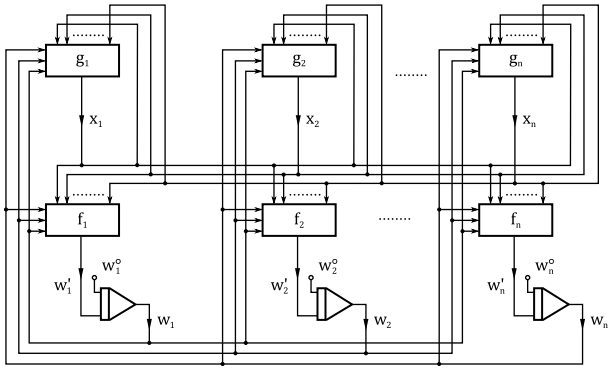


Figure 1: Antonomous system of differential equations - Block diagram

The analogue diagram solving the system of equations (1) is well-known in the theory of analogue and hybrid computers.

If we require higher precision, we have to impose more complex and discrete systems for perform individual mathematical operations. All we shall assume here is that components exist in order to carry out the requisite mathematical operations, as listed in Table 1, where the conventional symbol for each component is indicated.

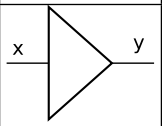
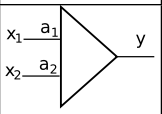
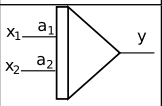
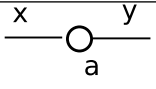
Component	Symbol	Mathematical relation
Inverting amplifier		$y = x$
Summing amplifier		$y = (a_1{\cdot}x_1 + a_2{\cdot}x_2)$
Integrating amplifier		$y = \int(a_1{\cdot}x_1 + a_2{\cdot}x_2)\,\mathrm{d}t$
Linear potentiometer		$y = a{\cdot}x$

Table 1: Analogue-principle symbols

From a hardware point of view it is not a problem to design inverters, adders and coefficients. The question is, of course, a design of digital integrators.

We have applied chosen formulas of numerical integration to solving of sets of homogeneous and non-homogeneous

linear differential equations with constant coefficients, to solving of sets of differential equations with time changing coefficients and for solving of sets of non-linear differential equations.

The algorithms of parallel cooperation of microprocessors arises from the form of numerical solution of individual differential equations (precisely from one step of a numerical solutions). This parallel cooperation of independent microprocessors may be designed using arbitrary chosen numerical integration formula. It is a very special goal to design numerical integrator with respect to numerical integration based on the Taylor series method.

3. Modern Taylor Series Method

The Taylor series method is one of the earliest analytic-numeric algorithms for the approximate solution of initial value problems for ordinary differential equations. Even though this method is not much preferred in literature, experimental calculations have shown and theoretical analyses have verified that the accuracy and stability of the Taylor series method exceeds the currently used algorithms for numerically solving differential equations.

The numerical solution of an ordinary differential equation (2)

$$y' = f(t,y), \qquad y(t_0) = y_0 \tag{2}$$

is written as the sequence (3)

$$[y(t_0) = y_0], \quad [y(t_1) = y_1], \quad \cdots \quad [y(t_n) = y_n]. \tag{3}$$

The best-known and most accurate method of calculating a new value of a numerical solution of a differential equation is to construct the Taylor series in the form (4).

$$\begin{aligned} y_{n+1} = y_n &+ h{\cdot}f(t_n,y_n) + \frac{h^2}{2!}{\cdot}f^{[1]}(t_n,y_n) + \cdots \\ &+ \frac{h^p}{p!}{\cdot}f^{[p-1]}(t_n,y_n) \end{aligned} \tag{4}$$

where h is the integration step.

The main idea behind the Modern Taylor Series Method is an automatic integration method order setting, i.e. using as many Taylor series terms for computing as needed to achieve the required accuracy. The Modern Taylor Series Method used in the computations increases the method order automatically, i.e. the values of the terms (5) are computed for increasing integer values of p until adding the next term does not improve the accuracy of the solution (last three terms of Taylor series are equal to zero).

$$\frac{h^p}{p!}{\cdot}f^{[p-1]}(t_n,y_n) \tag{5}$$

The main problem connected with using the Taylor series (in the form of (4)) is the need to generate higher derivatives $f^{[1]}, f^{[2]}, \cdots$. If it is possible, however, to obtain the terms with higher derivatives, the accuracy of calculations by the Taylor Series Method is extreme (it is in fact only limited by the type of the arithmetic unit used). A drawback of this method is that $f(t,y)$ has to belong to a special class. Fortunately, this class is large enough to contain the functions that appear in many applications. This is typical, in particular, of the solution of the technical initial problems.

3.1 Technical Initial Problems

Technical initial problems are defined as initial problems where the right-hand side functions of the system are those occurring in technical practice, that is functions generated by adding, multiplying and superposing elementary functions. Such systems can be expanded into systems with polynomials on the right-hand sides of the equations. In such a case the Taylor series terms can be easily calculated.

To demonstrate this, the equation (6) is analyzed.

$$y' = a{\cdot}y{\cdot}\cos t \qquad y(0) = y_0 \tag{6}$$

A simple computation scheme based on equation (2) follows

$$f(t,y) = a{\cdot}y{\cdot}\cos t \tag{7}$$

Let $v = \cos t$ then

$$\begin{aligned} f(t,y) &= a{\cdot}y{\cdot}v \\ f^{[1]}(t,y) &= a(f(t,y){\cdot}v + y{\cdot}v') \\ f^{[2]}(t,y) &= a(f^{[1]}(t,y){\cdot}v + 2{\cdot}f(t,y){\cdot}v' + y{\cdot}v'') \\ &\vdots \\ f^{[p-1]}(t,y) &= a{\cdot}(\sum_{i=0}^{p-2} f^{[p-2-i]}(t,y){\cdot}v^{[i]}\binom{p-1}{i} + \\ &\quad + y{\cdot}v^{[p-1]}) \quad (p \geq 2) \end{aligned} \tag{8}$$

where

$$\begin{aligned} v' &= -\sin t \\ u &= \sin t \\ v^{[p]} &= -v^{[p-2]} \quad (p \geq 2) \\ u^{[p-1]} &= v^{[p-2]} \quad (p \geq 2) \end{aligned} \tag{9}$$

Transformed technical initial problem (6) can be seen in Figure 2.

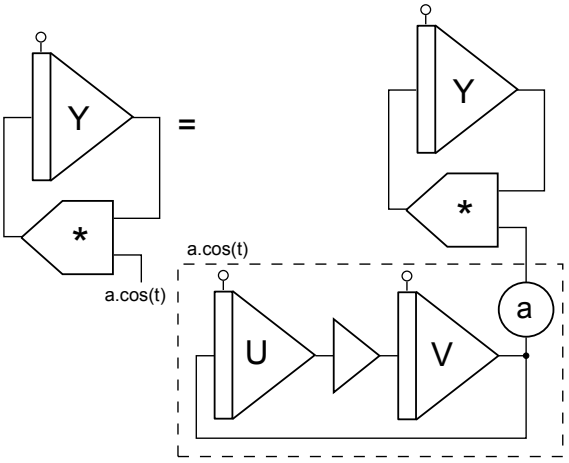


Figure 2: Corresponding block diagram

Similar constructions can be created for all elementary functions, such as $\exp, \sin, \cos, \tan, \coth, \ln, \sinh, \ldots$. The right-hand side of equations that belong to the technical initial problems can be decomposed in a sequence of simple operations: addition, subtraction, multiplication,

division and combination of them. These rules can be applied recursively so that recursive formulas for the derivatives of a function described by combinations of these elementary functions can be obtained.

A comparison of two numerical integration formulas is presented in this article and in [6], including 2nd and 4th order Runge-Kutta method and Taylor series method. A quite low precision output can be obtained by applying Runge-Kutta methods. The accuracy can be increased by increasing the order, but this relatively decreases speed. A test equation has been solved using Runge-Kutta formulas (RK2, RK4) and the Taylor series method (Taylor). The best results have been obtained by the Taylor series method.

Table 2 shows the error of solution at $t = 3s$. Fixed integration step h is a parametr of all computation. Error of computation is calculated as a difference between known analytic solution and corresponding numerical solution. Error is calculated at each step and maximum error during interval $0 - 3s$ is displayed. An accuracy as high as 10^{-17} can be obtained only with Taylor series method.

Table 3 shows corresponding computation time. For example, an accuracy 10^{-10} can be obtained in 1470 ms (using RK2) or in 6.48 ms (using RK4) or in 0.225 ms using Taylor series method of the 8th order.

Method	$h = 1$	$h = 0.1$	$h = 0.01$	$h = 0.001$
RK2	10^{0}	10^{-1}	10^{-4}	10^{-6}
RK4	10^{-2}	10^{-5}	10^{-9}	10^{-13}
Taylor	10^{-17}	10^{-17}	10^{-17}	10^{-17}

Table 2: Error of solution at $t = 3$

Error	10^{-1}	10^{-3}	10^{-5}	10^{-10}	10^{-15}
RK2 [ms]	0.154	0.752	7.47	1470	-
RK4 [ms]	0.029	0.119	0.334	6.48	324
Taylor2 [ms]	0.115	0.576	7.25	1140	-
Taylor4 [ms]	-	0.0824	0.329	4.5	225
Taylor8 [ms]	-	-	0.0444	0.225	0.916

Table 3: Computation time

More detailed information about the Modern Taylor Series Method can be also found in [8]. The Modern Taylor Series Method has been implemented in TKSL software [13].

4. Architecture of the Parallel System

Integrators (represented by **ALU** units) are the basic part of the parallel system that contains a control unit **CU** and an interconnecting network **ICN** (shown in Figure 3). All integrators (arithmetic logic units) are controlled by one control unit - each integrator carries out the same computation. Inputs and outputs of integrators are connected into an interconnecting network. This computation design corresponds to SIMD (Single Instruction Multiple Data) computation system [4].

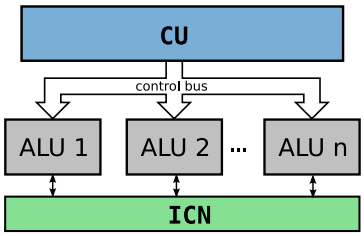


Figure 3: Parallel system

Corresponding mathematical operations have been analysed and it has been found that following operations are required:

- addition - carried out by parallel adder
- subtraction - converted into addition by change of the sign (complement notation)
- multiplication - carried out by either parallel multiplier or serially by Booth's algorithm

Two main operations are required: **multiplication** and **addition**. Addition is typically done in adders, multiplication can be done by either serial or parallel way. As communication can also be serial and parallel we can classify the integrators into following three categories:

- Parallel-parallel integrators - parallel communication and parallel computing
- Serial-parallel integrators - serial communication and parallel computing
- Serial-serial integrators - serial communication and serial computing

4.1 Parallel-parallel integrator

Computation is completed by parallel multiplier (multiplication) and parallel adder (addition). Block diagram is in Figure 4.

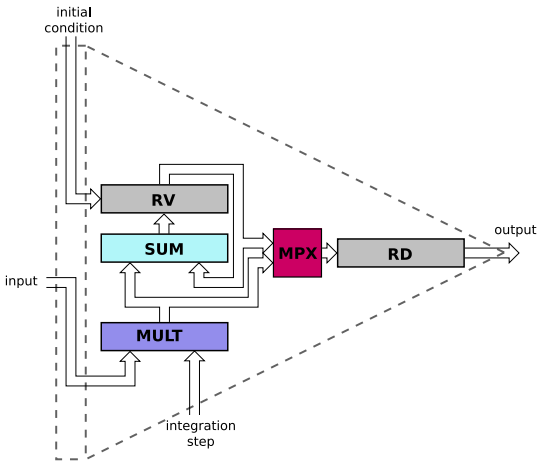


Figure 4: Parallel integrator

The integrator consists of the following blocks:

- **RV** Register of result

- **RD** Register of product
- **MPX** Multiplexer
- **SUM** Parallel adder
- **MULT** Parallel multiplier

This type represents the fastest numerical integrator. Computation time of one term of the Taylor series is equal to (10).

$$t_{PP} = \tau_{mult} + \tau_{add} + \tau_{net} \quad (10)$$

where τ_{mult} is time of product calculation, τ_{add} is time of total calculation and τ_{net} is delay in the communication network. Because parallel multipliers are usually complicated, this integrator has high hardware resource utilization and is not suitable for a wide range of applications - only for special cases where the time calculation is significant. Parallel bus is also heavy on hardware resources.

4.2 Serial-parallel integrator

Multiplication is completed by sequential method (Booth's algorithm of multiplication). Addition is completed by parallel adder. Block diagram is in Figure 5.

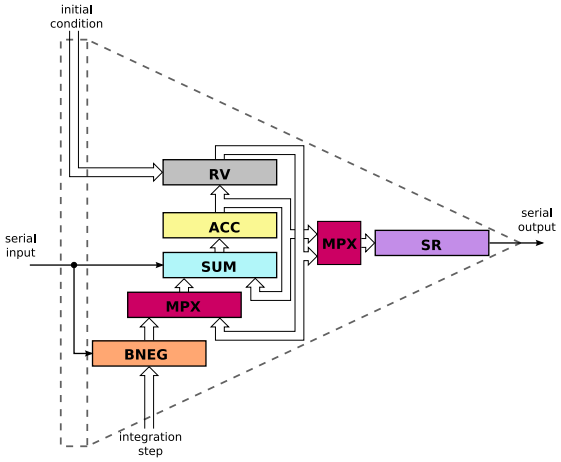


Figure 5: Serial-parallel integrator

The integrator consists of the following blocks:

- **RV** Register of result
- **MPX** Multiplexer
- **SUM** Parallel adder
- **ACC** Accumulator
- **SR** Communication shift register
- **BNEG** Circuits for Booth algorithm multiplication

An advantage of this integrator is a smaller chip occupation because of using a serial bus and missing complicated parallel multiplier. A disadvantage of this integrator is longer computation time because the multiplication is carried out serially in n steps and time of the computation of one Taylor series term is equal to (11).

$$\begin{aligned} t_{SP} &= \tau_{mult} + \tau_{add} + \tau_{net} \\ t_{SP} &= n{\cdot}\tau_{add} + \tau_{add} + \tau_{net} \end{aligned} \quad (11)$$

4.3 Serial-serial integrator

This integrator is basen on the previous one - serial-parallel integrator. Unlike the serial-parallel integrator, addition is also performed serially - by a full one-bit adder. Block diagram is in Figure 6.

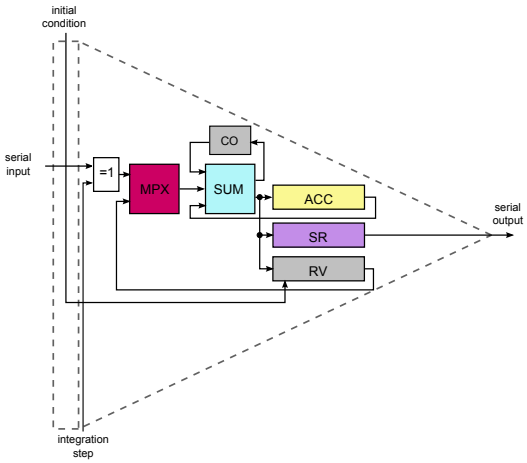


Figure 6: Serial-serial integrator

The integrator consists of the following blocks:

- **RV** Shift register of result
- **MPX** Multiplexer
- **SUM** Full one-bit adder
- **CO** Carry out register
- **ACC** Accumulator
- **SR** Output shift register

Chip slices occupation is a bit smaller than in the previous serial-parallel version. But the computation time of one Taylor series term is equal to an exponencial formula (12).

$$\begin{aligned} t_{SS} &= \tau_{mult} + \tau_{add} + \tau_{net} \\ t_{SS} &= n{\cdot}n{\cdot}\tau_{add} + n{\cdot}\tau_{add} + \tau_{net} \\ t_{SS} &= n^2{\cdot}\tau_{add} + n{\cdot}\tau_{add} + \tau_{net} \end{aligned} \quad (12)$$

In this case, τ_{add} is time of addition of the one-bit adder and n is number of bits used for storing values - the data with size as in the previous version.

5. Implementation of the Parallel System

Parallel system with all three versions of integrators, control unit and interconnecting network have been designed, simulated and finally implemented on field programmable gate arrays (FPGAs). FPGAs provide massive parallel structures and high density logic arithmetic with short design cycles. First integrators and control units have been designed in VHDL (hardware description language) [5] and simulated in a simulation ModelSim [11]. VHDL is commonly used as a design-entry language for field programmable gate arrays. Platform FITkit [3] has been used for implementation.

FITkit is evaluation and development platform, that has been developed at our faculty. FITkit is used for trying out a hardware implementation, not just simulation a

Table 4: Chip slices occupation and computation time of one Taylor series term

Data width [bit]	**16**	**32**	**64**	**16**	**32**	**64**	**128**	**16**	**32**	**64**	**128**
Chip occupation [%]	14	25	154	19	27	44	86	23	30	43	71
Multiplication time [ns]	8	15	28	96	224	576	1664	1024	4096	16384	65536
Addition time [ns]	6	7	9	6	7	9	13	4	4	4	4
Computation time [ns]	14	22	37	102	231	585	1677	1088	4224	16640	66048
Integrator type	**parallel-parallel**			**serial-parallel**				**serial-serial**			

hardware design as was usual later. FITkit platform contains FPGA chip XC3S50 Spartan 3. The VHDL model was synthesized with Xilinx ISE Software.

A parallel system containing each version of integrators, control unit constituted by a Moore machine, a static interconnection network and a SPI controller for the FITkit communication system has been implemented and synthesized.

These parallel systems have been compared so that a chip slices occupation and computation time of one Taylor series term to be shown. Results are presented in Table 4. Computation time has been calculated according to equations (10), (11) and (12). Values τ_{mult} and τ_{add} have been obtained from the synthesis results, value τ_{net} has been disregarded.

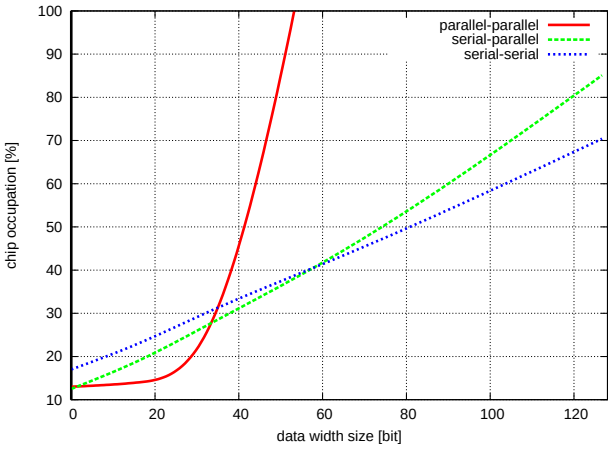


Figure 7: Chip resources occupation

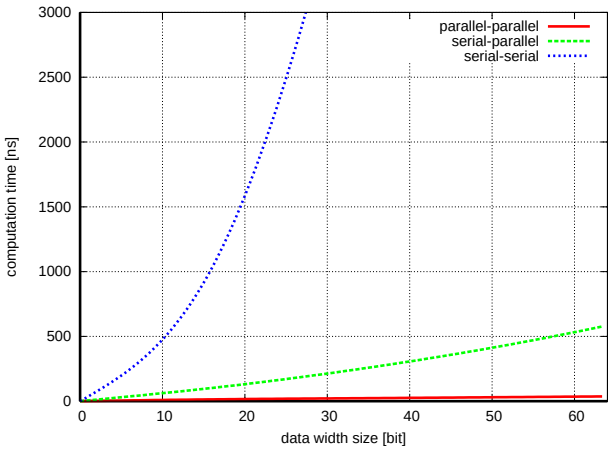


Figure 8: Taylor series term computation time

Figures 7 and 8 display results from Table 4 in a graphic form. As can be seen, the chip occupation of the serial-serial version of integrator is the smallest, but on the other hand the calculation speed is very slow.

The chip occupation of the serial-parallel version of integrator is only a bit higher than the serial-serial version because the only difference is in using a parallel adder instead of a one-bit full adder. Value for money of the serial-parallel integrator seems to be the best. Serial mode structures proved to have advantages for applications that require small area usage with a short clock cycle. They provide a reasonable response time even for relatively large bit-widths on a very small area.

Parallel-parallel integrator is the fastest but is suitable only for a special range of applications where the time calculation is significant. The occupation is the highest but can be decreased by using a chip which already contains embeded dedicated multipliers. In this case, multipliers do not have to be synthesized and chip slices can be used for other block elements such as registers, multiplexers and adders. However, the embedded hardware multipliers are consumed very rapidly for bit-widths larger than 18 and furthermore, the number on a chip is limited.

There are very few studies that consider the design of arithmetic operations in FPGAs. Text [1] compares the results achieved when implementation of basic fixed-point arithmetic units in FPGA.

Table 5 summarizes the FPGA hardware resource utilization (slices occupation) of parallel system with 1, 2, 4, 8 and 16 integrators and 32-bit data with size. We can see that parallel integrator (probably its parallel inputs and outputs) is heavy on number of slices.

Table 5: FPGA chip occupation

Type of integrator	**Number of ALU units**				
	1	**2**	**4**	**8**	**16**
Parallel-parallel	25%	99%	-	-	-
Serial-parallel	27%	37%	56%	95%	-
Serial-serial	30%	33%	38%	50%	72%

6. Conclusions

Systems of homogenous linear differential equations, electronic circuits simulations, control systems, partial differentials equations and systems of algebraic differential equations are typical applications of parallel cooperations of integrators. We can often observe that, if the functions on the right-hand sides are of a particular type frequently encountered in engineering applications, a sequence of substitutions can be found that transforms the original system into a new system with polynomials on

the right-hand sides. This is the main advantage of the modification.

Analogue diagrams represent a very convenient tool for describing parallel tasks. The idea of analogue principles is used in our Taylor series integrators. Actually, the methodology is in fact SIMD parallel architecture. Two main mathematical operations are required in the integrators: multiplication and addition. As required mathematical operations and the communication can be done by either serial or parallel way, the integrators can be classified into following three categories: parallel-parallel, serial-parallel and serial-serial. Serial mode structures proved to have advantages for applications that require small area usage with a short clock cycle. They provide a reasonable response time even for relatively large bit-widths on a very small area.

Acknowledgements. The research has been supported by the project MSM0021630528 (Ministry of Education, Youth and Sports, Czech Republic) "Security-Oriented Research in Information Technology" and FR2681/2010/G1 - "Didactic Tool of Computer Hardware Course Innovation". Also an international scholarship programm Aktion supported me during my study and research stay in Vienna University of Technology in the academic year 2009/2010.

References

[1] M. Bečvář and P. Štukjunger. Fixed-point arithmetic in FPGA. *Acta Polytechnica*, 45(2):67–72, 2005. ISSN 1210-2709.

[2] J. C. Butcher. *Numerical methods for ordinary differential equations*. Wiley, 2 edition, 2008. ISBN 978-0-470-72335-7.

[3] FIT Brno. *FITkit Homepage*, [online]. URL: http:// merlin.fit.vutbr.cz/FITkit/.

[4] J. L. Hennessy and D. A. Patterson. *Computer architecture: A quantitative approach*. Morgan Kaufmann Publisher, 4 edition, 2006. ISBN 0-12-370490-1.

[5] E. O. Hwang. *Digital Logic and Microprocessor Design with VHDL*. Thomson, 1 edition, 2006. ISBN 0-534-46593-5.

[6] M. Kraus, J. Kunovský, and V. Šátek. Fourier analysis and modern taylor series method. In *Proceedings of the 7th International Conference on Applied Mathematics*, page 6, 2007.

[7] M. Kraus, J. Kunovský, V. Šátek, and V. Kaluža. Parallel computations based on analogue principles. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation*, pages 111–116. IEEE Computer Society, 2009.

[8] J. Kunovský. *Modern Taylor Series Method*. FEI-VUT Brno, 1994. Habilitation work.

[9] B. J. MacLennan. A review of analog computing. Technical report, Department of Electrical Engineering & Computer Science University of Tennessee, Knoxville, 2007.

[10] Mathworks. *Web sites,* M*atlab Simulink Solvers*, [online]. URL: http:// www.mathworks.com/ access/helpdesk/help/toolbox/ simulink/ug/f11-69449.html.

[11] Mentor Graphics. *High Performance and Capacity Mixed* HDL *Simulation* - M*odel*S*im*, [online]. URL: http:// www.mentor.com/products/fv/modelsim/.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, September 2007.

[13] TKSL software. *High Performance Computing*, [online]. URL: https:// www.fit.vutbr.cz/ kunovsky/TKSL/index.html.en.

Selected Papers by the Author

J. Kunovský, M. Kraus, V. Šátek, A. Szollos. Parallel Computations Based on Modified Numerical Integration Methods. In *Proceedings of the 10th International Conference of Numerical Analysis and Applied Mathematics*, AIP Conference Proceedings, volume 1479, pages 2217–2220, Kos, GR, 2012. American Institute of Physics.

J. Kunovský, M. Kraus, V. Vopěnka. Runge-Kutta Based Parallel Computations. In *Proceedings of the MATHMOD VIENNA 2012 - 7th Vienna Conference on Mathematical Modelling*, pages 6, Vienna, AT, 2012. ARGE Simulation News.

J. Kunovský, M. Kraus, V. Šátek. New Trends in Taylor Series Based Computations. In *Numerical Analysis and Applied Mathematics*, AIP Conference Proceedings, volume 1168, pages 282–285, Rethymno, Crete, GR, 2009. American Institute of Physics.

J. Kunovský, M. Kraus, V. Šátek, J. Kopřiva. Automatic Method Order Settings. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation (UKSim 2009)*, pages 117–122, Cambridge, GB, 2009. IEEE Computer Society.

J. Kunovský, M. Kraus, V. Šátek, V. Kaluža. Parallel Computations Based on Analogue Principles. In *Proceedings of Eleventh International Conference on Computer Modelling and Simulation (UKSim 2009)*, pages 111–116, Cambridge, GB, 2009. IEEE Computer Society.

J. Kunovský, M. Kraus, V. Šátek, V. Kaluža. Accuracy and Word Width in TKSL. In *Proceedings of Second UKSIM European Symposium on Computer Modeling and Simulation (EMS 2008)*, pages 153–158, Liverpool, GB, 2008. IEEE Computer Society.

J. Kunovský, M. Kraus, V. Šátek. Taylor Series Numerical Integrator. In *Proceedings of Second UKSIM European Symposium on Computer Modeling and Simulation (EMS 2008)*, pages 177–180, Liverpool, GB, 2008. IEEE Computer Society.

J. Kunovský, M. Kraus, V. Šátek. 25th Anniversary of TKSL. In *Numerical Analysis and Applied Mathematics*, AIP Conference Proceedings, volume 1048, pages 343–346, Psalidi, Kos, GR, 2008. American Institute of Physics.

J. Kunovský, M. Kraus, V. Šátek, M. Pindrič. Taylor Series in Control Theory. In *Proceedings UKSim 10th International Conference on Computer Modelling and Simulation (EUROSIM/UKSim)*, pages 378–379, Cambridge, GB, 2008. IEEE Computer Society.

J. Kunovský, M. Kraus, V. Šimek, J. Petřek. GPU Based Accelleration of Telegraph equation. In *Proceedings UKSim 10th International Conference on Computer Modelling and Simulation (EUROSIM/UKSim)*, pages 378–379, Cambridge, GB, 2008. IEEE Computer Society.