AUTOMATYKA/ AUTOMATICS • 2013 • Vol. 17 • No. 2

http://dx.doi.org/10.7494/automat.2013.17.2.187

Grzegorz Karpiel*, Konrad Gac*, Maciej Petko*

FPGA Based Hardware Accelerator for Parallel Robot Kinematic Calculations

1. Introduction

Quality improvement of a manufactured product is a natural process accompanying the development of civilization. Nowadays, designed machines are much more accurate and efficient. In addition, the flexibility of production needs to increase functionality built machines. Among all machines assigned for manufacturing purposes, milling machines play a significant role. Common three-axis milling machines characterized by simple kinematics are being increasingly replaced by universal machines with multiple degrees-of-freedom, as their versatility allows implementation of a variety of tasks, such as turning or milling in different planes. While the machines accuracy is determined by its mechanics parameters and applied control system, the flexibility and performance are often determined by the limitations of computational power of the milling machines control system. When choosing a parallel structure as a support structure for the milling machine, kinematics calculations, necessary for the correct tool movement along the desired trajectory path, are a major concern. Those calculations should be performed in real time, at a frequency level of the operating driver system. In this paper the increase of computation power, when determining the kinematics of the milling machine based on a parallel robot, with the usage of a FPGA system equipped with a processor with additional dedicated instructions, is presented.

1.1. Parallel robots

A parallel robot is a mechanism, which arms consisting of any number of kinematics chains are connected by a joint or a platform [1]. Closed kinematics gives the construction much more rigidity and enables it to work more precisely and faster than constructions with open kinematics. A disadvantage of this structure is a smaller workspace. The workspace is limited by an arm movement range and it is a common part of sets of points possible to achieve by each arm.

* AGH University of Science and Technology, Krakow, Poland
** The project was funded from The National Centre for Research and Development

1.2. Kinematics

To obtain the full advantages of the robots' movement capabilities, the forward and inverse kinematics must be determined. The solution of the forward kinematics gives a clear rule defining the position of the tools tip in Cartesian space, depending on the position of each arm. Much more useful information is provided by their reverse relationship, being the solution of the inverse kinematics, which for a given point in space defined by the Cartesian space gives the position of the robots joints in configuration which the desired point is being reached. For the parallel robots, it is much easier to find the solution of the inverse kinematics, than the forward one [2]. By knowing the solution of the inverse kinematics, the robot can be treated as a Cartesian structure and with tools such as NC code generators for milling machines can be therefore used directly. The only drawback of an approach such as this is the calculation of the inverse kinematics, converting coordinates from the Cartesian space directly in to the joint lengths. It should be noted that the calculations must be carried out at any point of the trajectory, in real time, with the operating frequency of the trajectory generator.

2. The parallel robot for milling

The construction of a parallel robot for milling (Fig. 1) was developed in Department of Robotics and Mechatronics AGH [3]. The robot consists of three arms connected together with a moving platform. Inside the platform there is a spindle rotating a milling bit at 40 000 rpm. This can be classified as high-speed machining (HSM). HSM means using spindle speeds that are significantly higher than those used in conventional machining operations. Typical HSM spindle velocities range between 8 000 and 35 000 rpm, although some spindles nowadays are designed to rotate at over 100 000 rpm.

Fig. 1. Parallel robot with three degrees of freedom for milling

2.1. Kinematic structure

Figure 2 presents the kinematic structure of the robot. It consists of three arms I, II, III attached to the base on the vertices of an equilateral triangle inscribed into a circle with radius R. Each arm can protrude by changing lengths l_1, l_2, l_3. All rotary joints are based on universal joints.

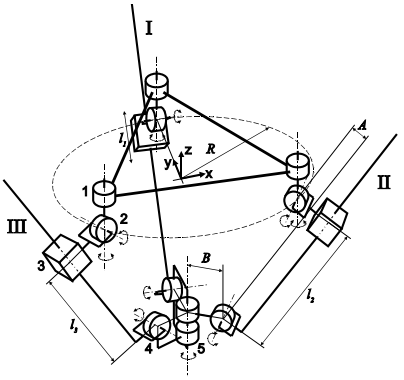

Fig. 2. The kinematic structure of the parallel robot

2.2. The solution of the inverse kinematics problem

For such a kinematic structure, the solution of the inverse kinematics problem was solved [4]. Distances A and B can be simplified [5]. The solution of the inverse kinematics problem takes form of the following equation sets: (1):

$$\begin{aligned} l_1 &= \sqrt{x^2 + (y-R)^2 + z^2} \\ l_2 &= \sqrt{\left(x - \frac{\sqrt{3}}{2}R\right)^2 + \left(y + \frac{1}{2}R\right)^2 + z^2} \\ l_2 &= \sqrt{\left(x + \frac{\sqrt{3}}{2}R\right)^2 + \left(y + \frac{1}{2}R\right)^2 + z^2} \end{aligned} \tag{1}$$

where:

x, y, z – the position of the tool in the Cartesian space,
R – radius of the circle on which robot arms are placed,
l_1, l_2, l_3 – arms lengths.

For the full realization of the trajectory in the workspace derived equations (1) should be solved. These derivatives allow for the transformation of velocity and acceleration of the tool in the Cartesian space to the speed and acceleration of the drives (2)–(4)

$$v_1 = \frac{1}{2}\frac{2\cdot x\cdot v_x + 2\cdot (y-R)\cdot v_y + 2\cdot z\cdot v_z}{\sqrt{x^2+(y-R)^2+z^2}}$$

$$v_2 = \frac{1}{2}\frac{2\cdot\left(x-\frac{\sqrt{3}}{2}R\right)\cdot v_x + 2\cdot\left(y+\frac{1}{2}R\right)\cdot v_y + 2\cdot z\cdot v_z}{\sqrt{\left(x-\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2}} \tag{2}$$

$$v_3 = \frac{1}{2}\frac{2\cdot\left(x+\frac{\sqrt{3}}{2}R\right)\cdot v_x + 2\cdot\left(y+\frac{1}{2}R\right)\cdot v_y + 2\cdot z\cdot v_z}{\sqrt{\left(x+\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2}}$$

where:

v_x, v_y, v_z – tool velocity in the Cartesian space,
v_1, v_2, v_3 – drives velocity.

$$a_1 = -\frac{1}{4}\frac{\left(2\cdot x\cdot v_x + 2\cdot (y-R)\cdot v_y + 2\cdot z\cdot v_z\right)^2}{\left(x^2+(y-R)^2+z^2\right)^{\frac{3}{2}}} +$$

$$+\frac{1}{2}\frac{2\cdot v_x^2 + 2\cdot x\cdot a_x + 2\cdot v_y^2 + 2\cdot (y-R)\cdot a_y + 2\cdot v_z^2 + 2\cdot z\cdot a_z}{\sqrt{x^2+(y-R)^2+z^2}}$$

$$a_2 = -\frac{1}{4}\frac{\left(2\cdot\left(x-\frac{\sqrt{3}}{2}R\right)\cdot v_x + 2\cdot\left(y+\frac{1}{2}R\right)\cdot v_y + 2\cdot z\cdot v_z\right)^2}{\left(\left(x-\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2\right)^{\frac{3}{2}}} + \tag{3}$$

$$+\frac{1}{2}\frac{2\cdot v_x^2 + 2\cdot\left(x-\frac{\sqrt{3}}{2}R\right)\cdot a_x + 2\cdot v_y^2 + 2\cdot\left(y+\frac{1}{2}R\right)\cdot a_y + 2\cdot v_z^2 + 2\cdot z\cdot a_z}{\sqrt{\left(x-\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2}}$$

$$a_3 = -\frac{1}{4}\frac{\left(2\cdot\left(x+\frac{\sqrt{3}}{2}R\right)\cdot v_x + 2\cdot\left(y+\frac{1}{2}R\right)\cdot v_y + 2\cdot z\cdot v_z\right)^2}{\left(\left(x+\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2\right)^{\frac{3}{2}}} + \frac{1}{2}\frac{2\cdot v_x^2 + 2\cdot\left(x+\frac{\sqrt{3}}{2}R\right)\cdot a_x + 2\cdot v_y^2 + 2\cdot\left(y+\frac{1}{2}R\right)\cdot a_y + 2\cdot v_z^2 + 2\cdot z\cdot a_z}{\sqrt{\left(x+\frac{\sqrt{3}}{2}R\right)^2+\left(y+\frac{1}{2}R\right)^2+z^2}} \tag{4}$$

where:

a_x, a_y, a_z – tool acceleration in the Cartesian space,
a_1, a_2, a_3 – drives acceleration.

3. Accelerator of kinematic calculations

The equations presented in the previous section allow to transform the trajectory generated in the Cartesian space into parallel robot's trajectory in joint space. In the case of implementing equations in actual controller the most important factors are related to accuracy and speed of calculations. In this regard an interesting solution is the implementation of the kinematics equations in a controller based on FPGA [6, 7].

3.1. Desired trajectory

An example of the trajectory is shown in Figure 3. Was assumed, that trajectory is a horizontal circle with 0.150 m radius situated on the height of 0.3 m.

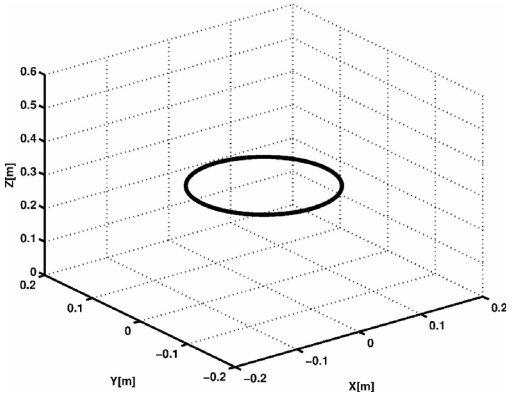

Fig. 3. Tool path in Cartesian space

Figure 4 shows the waveforms corresponding to the trajectory in Cartesian space and the joint space. Each point of the circle in the joint space was described by the nine parameters.: position *px*, *py*, *pz*, velocity *vx*, *vy*, *vz* and acceleration *ax*, *ay*, *az*. For each point calculated lengths *l*1, *l*2, *l*3, also the speed *v*1, *v*2, *v*3 and accelerations *a*1, *a*2, *a*3 were calculated. These points are determined by solving the inverse kinematics, which implemented on a PC with the use of double-precision floating-point variables (double).

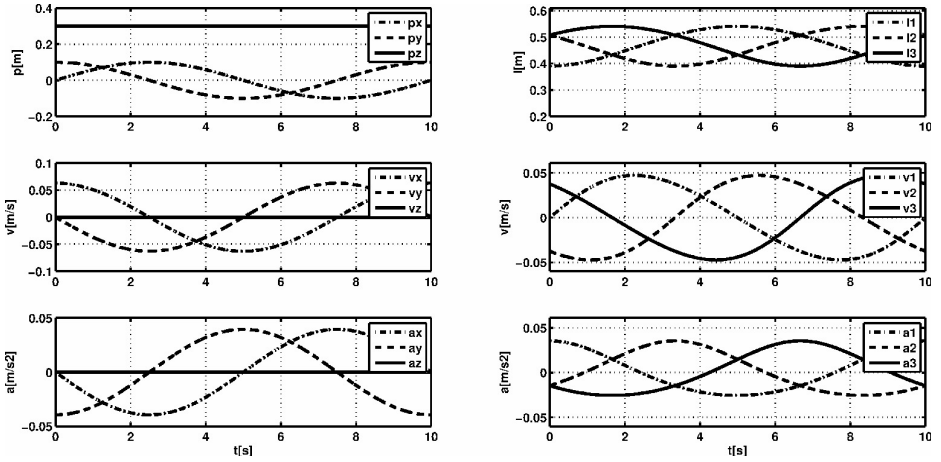

Fig. 4. Trajectory in Cartesian space (left) and in the joint space (on the right)

3.2. Implementation

Equations (1)–(4) were implemented in Altera FPGA chip [8]. Terasic DE2-115 board was chosen as a platform for testing (Fig. 5). Clock frequency of the Cyclone IV was set to 50 MHz.

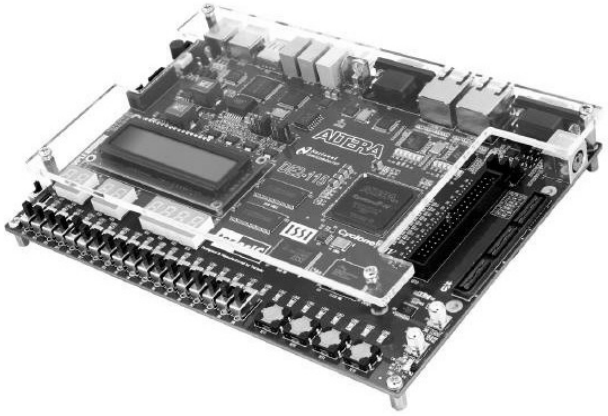

Fig. 5. DE2-115 board – test platform

In the first step by using the tools included in the package QSYS microprocessor system based on the Nios II processor was built (Fig. 6). It contains the memory of "OnChip" and data "sdram" on trajectory. Communication with a PC and the JTAG interface and an RS232 port located on the DE2-115.

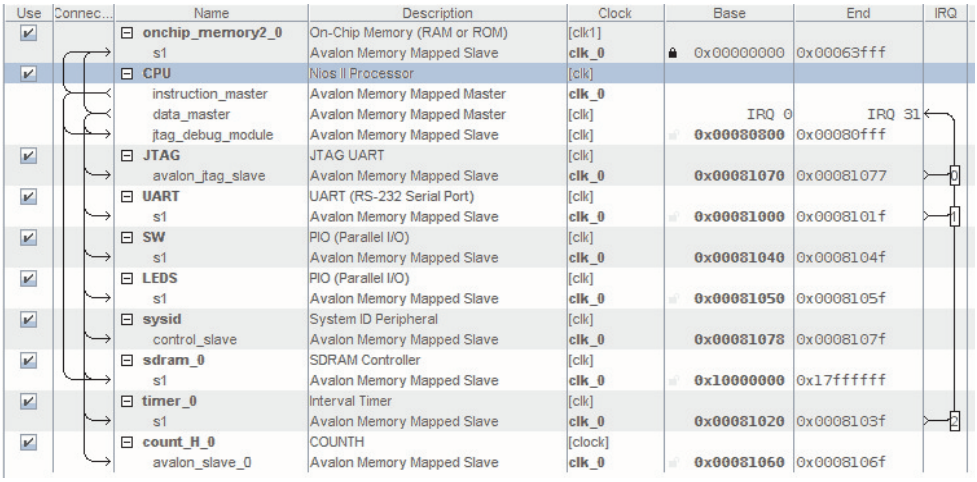

Fig. 6. SoPC structure designer in QSYS application

Three systems were created. The first one is without arithmetic coprocessor, in which floating-point operations are executed in software with single precision using math library provided by a compiler. The second system was equipped with a floating-point coprocessor of single precision provided by Altera. This coprocessor enables to perform operations like hardware summation multiplication, subtraction and division. The last system is also equipped with the arithmetic coprocessor but is expanded with a custom instruction enabling hardware calculation of the square root. Written in Verilog algorithm of floating-point square root [10] is reduced to the calculation of the square root of an integer, which results was determined by a mathematical relation (5):

$$\sqrt{a \cdot 2^b} = \sqrt{a} \cdot 2^{\frac{b}{2}} \tag{5}$$

In the calculations of kinematic equations, implemented in C programming language, the root was replaced by a previously defined macro instruction:

```
#define  Sqrt(A)  __builtin_custom_fnf(0x000000000,(A))
```

The compiler in the process of compiling these instructions translated into a „custom" logic of hardware responsible for calculating the square root.

3.3. The results of the calculations

The calculation correctness obtained from the designed system, with its own square root calculation instruction has been tested by comparing the trajectories generated in the joint space with the previously calculated trajectory on a PC. It should be noted, that the Nios II processor performs all floating point operations with single-precision accuracy (float 32-bits). By comparing all the points of the obtained trajectories, the calculations error characteristic has been achieved along the trajectory (Fig. 7).

Different intensities of gray color show respectively the error for the first, second and third linear drive.

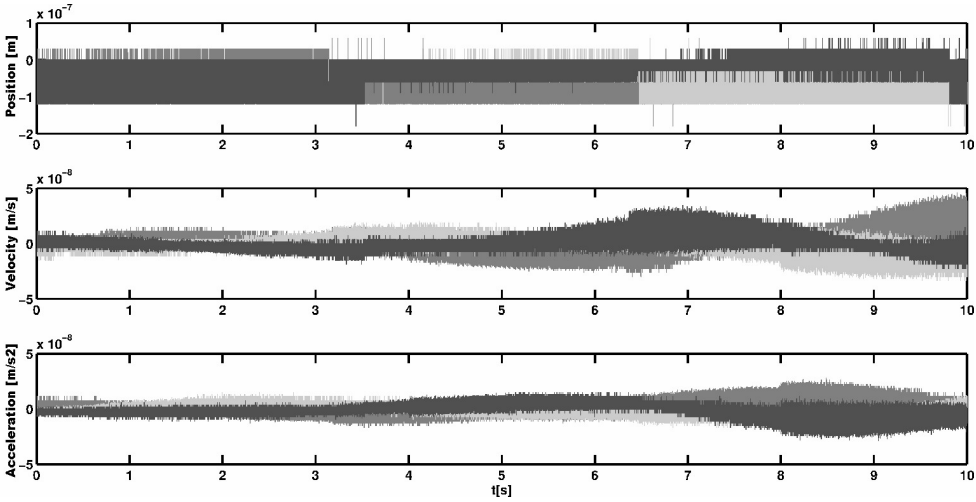

Fig. 7. Calculation error for position, velocity and acceleration

The resulting error of 50 nm for the position measurement is fully acceptable.

3.4. The time of calculations

Table 1 shows the changes in the calculation time and the amount of required FPGA resources depending on the microprocessor system version. For the first variation, the whole system need 4501 logical elements (LE) of FPGA. The calculation time for one point of the trajectory was 2324 μs. It means that without supporting the calculation by the arithmetic coprocessor, the maximum frequency is 430 Hz. In the case of standard hardware Nios II processor instructions for floating-point operations, the time of calculations shortens to 560 μs. It gives the calculation frequency of 1785 Hz. The addition of the arithmetic coprocessor slightly increased the amount of used resources.

The highest calculation efficiency was obtained by adding a custom instruction supporting square root calculation. The amount of used resources increased slightly (by about

1400 LE comparing to version with Altera coprocessor alone) however the time of calculation shortened to 143 μs. This enables the kinematic calculations to be made with frequency of 7 kHz. In typical aluminum machining by high-speed milling, the feed rate is 2 m/min. It means that in one second the tool moves by 33 mm. With a calculation frequency of 7 kHz, the distance between the points of trajectory is 4.7 μm. The distance is fully acceptable.

Table 1
Calculation time for a single point of the trajectory

Nios II	Calculation time [μs]	Total LEs used (114480 available)
No coprocessor	2324	4501 (4%)
With coprocessor	560	5840 (5%)
With coprocessor and custom square root instruction	143	7219 (6%)

4. Conclusion

In thise paper, knowledge of how to use a custom instruction for improving the performance of calculations when solving the inverse kinematics of a parallel robot has been presented. An example of an actually constructed parallel robot designed for milling applications has been shown, together with its geometric structure and solution of the inverse kinematics problem.

The accelerator featured in the paper expands the standard floating-point coprocessor for the Nios II processor with additional square root instruction. The calculation speed was quadrupled whereas the number of used logical elements increased by 1% for the Altera Cyclone IV FPGA chip. The obtained results, based on the developed accelerator, allow to build a driver, working at a frequency of 7 kHz and ensuring an accurate calculation of displacement up to 50 nm.

Further computational acceleration of the inverse kinematics would require the usage of an extended math coprocessor involving much more complex operations. The number of resources of currently available FPGA systems can fully perform the kinematics calculations by hardware. In addition, equations allow for a partial parallelization of individual operations, shortening the computation time. In future research, the authors plan to build a completely hardware accelerator for the kinetic calculations for the presented structure of the robot, based on fixed-point arithmetic.

References

[1] Gogu G., *Structural Synthesis of Parallel Robots. Part 1.* Springer, 2008, ISBN 978-1-4020-5102-9.

[2] Petko M., *Wybrane metody projektowania mechatronicznego*. AGH, Kraków-Radom, 2008, ISBN 978-83-7204-709-0.
[3] Karpiel G., Petko M., Uhl T., *Manipulator równoległy trzyramienny*. PL203631B1(B25J18/00).
[4] Karpiel G., *Zastosowanie podejścia mechatronicznego w projektowaniu robotów równoległych*. Ph.D. Thesis, AGH, 2007.
[5] Prusak D., Petko M., Karpiel G., *Manipulator trzyramienny o prostym modelu geometrycznym*. PL210002(B25J18/04;B25J9/06).
[6] Sanchez D., Munoz D., Llanos C., Motta J., *A Reconfigurable System Approach to the Direct Kinematics of a 5 D.o.f RoboticManipulator*. International Journal of Reconfigurable Computing, 2010, ArticleID 727909.
[7] Karpiel G., Prusak D., *System sterowania robotem równoległym oparty na układzie FPGA.* in: Projektowanie, analiza i implementacja systemów czasu rzeczywistego, Warszawa, Wydawnictwa Komunikacji i Łączności, 2011, ISBN: 878-83-206-1822-8.
[8] *http://www.altera.com/*, 2012-04-25.
[9] *http://www.terasic.com.tw/,* 2012-04-25.
[10] Piromsopa K., Aporntewan C., Chongsatitvatana P., *An FPGA implementation of a fixed-point square root operation.* http://pioneer.netserv.chula.ac.th/~achatcha/Publications/0012.pd, 2012-04-25f.