Microprocessors and Microsystems 50 (2017) 14–25

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

An FPGA-based low-cost VLIW floating-point processor for CNC applications

Jingchuan Dong[a,b,*], Taiyong Wang[b], Bo Li[a], Zhe Liu[a], Zhiqiang Yu[a]

[a] School of Mechanical Engineering, Tianjin University, Tianjin 300354, PR China
[b] Key Laboratory of Mechanism Theory and Equipment Design of Ministry of Education, Tianjin University, Tianjin 300354, PR China

ARTICLE INFO

Article history:
Received 5 May 2016
Revised 21 September 2016
Accepted 6 February 2017
Available online 13 February 2017

Keywords:
FPGA
Embedded processor
CNC
Motion planning
Asynchronous execution

ABSTRACT

In the high-speed free-form surface machining, the real-time motion planning and interpolation is a challenging task. This paper presents the design and implementation of a dedicated processor for the interpolation task in computerized numerical control (CNC) machine tools. The jerk-limited look-ahead motion planning and interpolation algorithm has been integrated in the interpolation processor to achieve smooth motion in the high-speed machining. The processor features a compactly designed floating-point parallel computing architecture, which employs a 3-stage pipelined reduced instruction set computer (RISC) core and a very long instruction word (VLIW) floating-point arithmetic unit. A new asynchronous execution mechanism has been employed in the processor to allow multi-cycle instructions to be performed in parallel. The proposed processor has been verified on a low-cost field programmable gate array (FPGA) chip in a prototype controller. Experimental result has demonstrated the significant improvement of the computing performance with the interpolation processor in the free-form surface machining.

1. Introduction

Motion profile generation is important in the high speed CNC machining, because the motion profile affects machining time, the smoothness of the motion, the impact on servo system, the structural vibration, contour error, and the energy efficiency [1–3]. To guaranty the smoothness of the motion, the motion profile must be carefully planned to meet the machine tool constraints, such as velocity, acceleration and jerk [4,5]. The motion profile is usually defined by a piecewise high-order polynomial function [6]. The motion profile also needs to be optimized during motion planning. For complex tool paths, the look-ahead algorithm is introduced to solve the optimization problem by analyzing the tool path in advance [7,8]. The geometric properties of the path, such as the distance, the curvature and the sharp corners, should be considered in the real-time motion planning [9-11]. Therefore, the motion planning algorithms for high speed machining are usually complex and computationally intensive. Moreover, to maintain the accuracy in high speed motion, the interpolation interval should be short and the calculation accuracy should be sufficient.

Most controllers implement the motion profile generation task with other tasks (e.g. PLC, interpreter, motor control tasks) on an embedded processor. For low-cost designs, the microcontroller based hardware platforms are often used. A microcontroller is equipped with a processor core, memory and various integrated peripherals. However, the computational ability is limited in a low-cost microcontroller, which makes it difficult to execute motion planning and interpolation algorithms with high frequency and high accuracy. Therefore, in high speed and high accuracy applications, a high-end platform (such as an industrial PC [12], or a high speed floating-point DSP) is often employed. Some researchers also tried to design hardware accelerators to achieve high performance computing. Since the field-programmable gate array (FPGA) chips are commonly used in modern CNC designs to deploy custom logics, the resources in the FPGA chips can be utilized to implement hardware accelerators with low cost. The application of FPGA based hardware accelerators in motion control systems are widely studied in recent years [13]. Chiang et al. [14] employed an FPGA board in the real-time tracking control for linear induction motor drives. Ghaffari et al. [15] implemented an FPGA-based contour controller for a two-axis table using a dynamic contour error estimate algorithm. Kung et al. [16] developed an X-Y table motion control IC based on an FPGA chip using the NIOS II soft-core processor. The point-to-point and circular movement with trapezoidal velocity profile were applied in the processor. Cho et al. [17] designed a multi-axis motion control chip based on Xilinx

* Corresponding author.
E-mail addresses: new_lightning@aliyun.com (J. Dong), tywang@139.com (T. Wang), specterwaxz1314@163.com (B. Li), tyssqlz@sina.com (Z. Liu), yuxiaoqiang@tju.edu.cn (Z. Yu).

http://dx.doi.org/10.1016/j.micpro.2017.02.001
0141-9331/

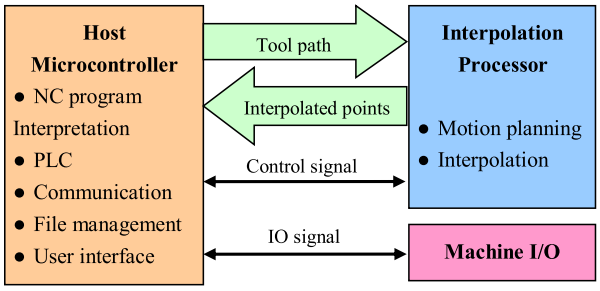

Fig. 1. CNC system based on the interpolation processor.

Virtex-2 FPGA chip to control a robot with 2-DOF movement. The velocity profile generation, interpolation calculation, inverse kinematics calculation, and PID control were implemented by hardware logic using fixed-point algorithm. The velocity profile generation module can generate a motion profile based on pre-defined acceleration (deceleration) coefficient tables. Yau et al. [18] proposed a real-time non-uniform rational B-spline (NURBS) interpolator based on parallel computation and implemented the algorithm on an FPGA. An FPGA-based 3-D dynamic interpolation for process and system dynamics compensation is presented by Oldknow and Yellowley [19]. Osornio-Rios et al. [20] showed an FPGA implementation of high-degree polynomial-based acceleration profile with limited jerk. Ponce et al. [21] developed an FPGA based artificial neural network robust controller with quantitative feedback theory for controlling the CNC micro-machine tool. Amornwongpeeti et al. [22] used time-division multiplexing scheme to control multi-unit PMSM motors. Previous works have implemented many motion control functions on FPGA chips. However, the high speed motion planning algorithm with look-ahead ability is still not realized in the FPGA based hardware accelerators.

The main obstacle in realizing a hardware accelerator for high speed motion planning is the complexity of the algorithm. The real-time algorithm involves iterations, nonlinear equations and complex control flows. If the algorithm is directly mapped into hardware in a fully parallel manner, the resultant design will be too large to be fitted in a low-cost FPGA. An alternative method is to design a dedicated processor for the computationally intensive tasks. In recent years, the FPGA-based soft-core processors are studied in many research fields [23–28]. The FPGA-based soft-core processor provides several advantages, such as flexibility, platform independence, reconfigurability, security, and low cost. The soft-core processor can be deeply customized to accelerate the application specific algorithms, and become popular in modern industrial system designs [29].

This paper presents a new FPGA-based floating-point processor aimed for accelerating the interpolation task in the free-form CNC machining. Compared with the fixed-point number, the floating-point number provides a larger dynamic data range, which is preferred in the high accuracy nonlinear applications [30–32]. The proposed processor features a pipeline designed reduced instruction set computer (RISC) core with very long instruction word (VLIW) support [33], a floating-point unit, and a novel asynchronous execution mechanism. Although the processor has many advanced features, the processor is compactly designed to allow being implemented on a low-cost FPGA. In the proposed CNC system architecture, as shown in Fig. 1, the motion planning and interpolation are performed in the FPGA-based processor, which offloads the computational intensive tasks from the host processor.

The rest of this paper is organized as follows. In Section 2, the smooth motion planning and interpolation algorithm for high speed free-form surface machining is briefly introduced. The algorithm deals with the piecewise linear tool path, which is the most common type in the free-form surface machining. The proposed interpolation processor is described in Section 3. The design objectives, the instruction set and the hardware implementations are shown. In Section 4, the experimental validation of a prototype 3-axis CNC controller based on the interpolation processor is presented. The performance of the design is illustrated in the benchmark test and processor load test. Section 5 gives the conclusion.

2. Motion planning and interpolation

In the free-form surface machining program, the tool path generated by the computer-aided manufacturing (CAM) software is usually represented by a series of short lines. The motion profile should be smooth in order to meet the capacity of the machine tools and avoid exciting the natural modes of the structure. This section focuses on the feedrate planning and interpolation problem in the short line tool path.

Fig. 2 shows the data flow of the motion planning and interpolation task. The tool path data is firstly passed to the geometrical analysis module, where the geometrical properties of the tool path are being calculated and then pass to the look-ahead module. Based on the tool path geometrical, several successive short lines can be assigned to a planning unit (basic element in motion profile generation). The short line and planning unit data will be appended to the Short Line Queue (SLQ) and Planning Unit Queue (PUQ). The look-ahead module scans the PUQ to generate optimized motion parameters for the current planning unit to be interpolated by applying “backward” and “forward” planning. Then, the jerk limited profile generation module is activated to sampling the distance along the planning unit according to the planned motion parameters. Finally, the tool path interpolation module calculates the position command for the servo motors.

2.1. Tool path geometrical analysis

Suppose the sequence of the end points of the short line segments are $P_i (i = 0, 1, 2...)$, as shown in Fig. 3. If the coordinate of P_i is (x_i, y_i, z_i), the following information can be obtained:

1. The length of each short lines, S_i
2. The unit vector of feed direction, $\vec{u_i}$
3. The change in the feed direction at the corner, $\Delta\vec{u_i}$
4. The estimated curvature radius, $\hat{\rho}_i$

$$\hat{\rho}_i = \frac{S_i}{2} / \cos\frac{\varphi_i}{2} = \frac{S_i}{\left\|\Delta\vec{u_i}\right\|} \tag{1}$$

2.2. Jerk limited acceleration

To achieve a smooth feed motion, the jerk-limited motion profile is applied to constrain the maximum jerk and acceleration, as shown in Fig. 4. A complete jerk-limited acceleration profile consists of 7 phases [4]. The motion equation for the ith phase $(i = 0, 1, ..., 7)$ can be written as

$$\begin{cases} j(t) = j(t_i) \\ a(t) = a(t_i) + j(t_i) \times (t - t_i) \\ v(t) = v(t_i) + a(t_i) \times (t - t_i) + \frac{1}{2} j(t_i) \times (t - t_i)^2 \\ s(t) = s(t_i) + v(t_i) \times (t - t_i) + \frac{1}{2} a(t_i) \times (t - t_i)^2 + \frac{1}{6} j(t_i) \times (t - t_i)^3 \\ t \in [t_i, t_{i+1}) \end{cases} \tag{2}$$

where t is the motion time, t_i is the start time of each phase, $j(t)$, $a(t)$, $v(t)$, $s(t)$ are the jerk, acceleration, velocity and distance functions. The maximum jerk, J, and the maximum acceleration, A, are the planning constraints determined by the dynamic limits of the machine tool.

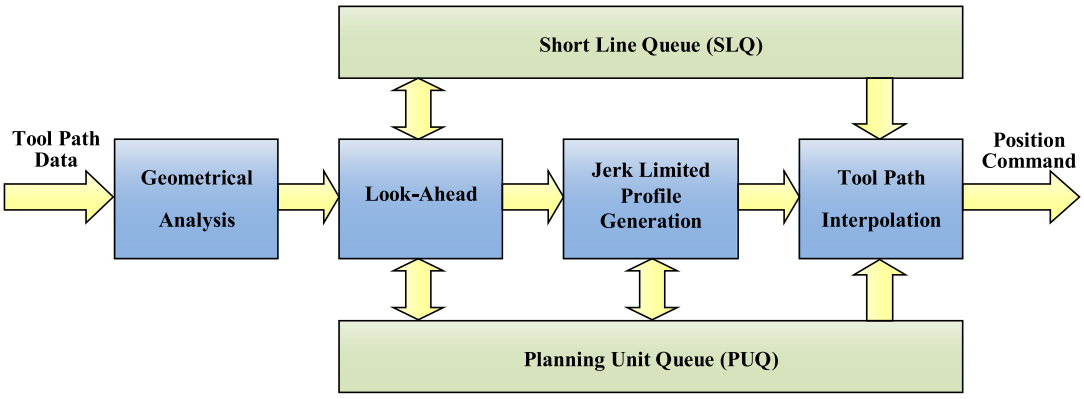

Fig. 2. Data flow of the motion planning and interpolation.

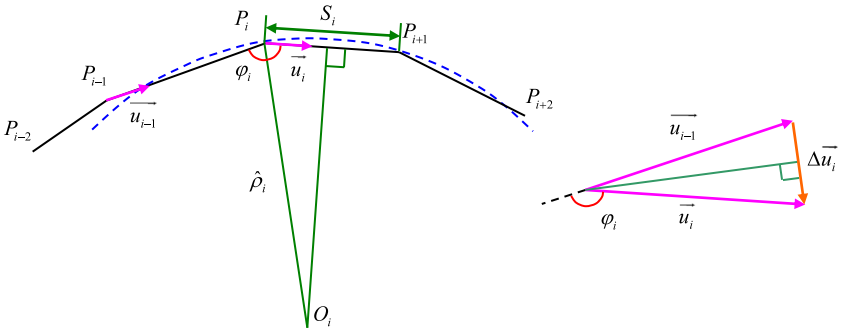

Fig. 3. Geometry analysis of the short line tool path.

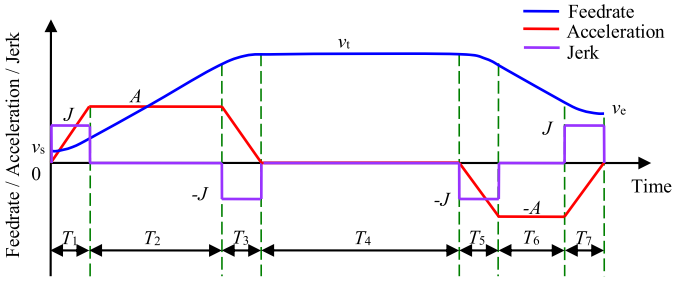

Fig. 4. Complete motion profile of jerk-limited acceleration.

The basic element for the motion planning is called a planning unit, which may contain several successive short lines [9]. In the motion planning, the length of the planning unit, S, the start velocity, v_s, the target velocity, v_t, and the maximum end velocity, v_{et}, are given, the motion time, $T_1 \sim T_7$, should be solved for the planning unit. If the planning unit is too short or the velocity change is too small, there would be less than 7 phases in the profile. Therefore, the motion profiles can be classified into different types. The type of the motion profile can be determined by comparison to the critical distances and the critical velocities.

After the motion type is determined, the distance and velocity equations can be established. These equations can be transformed to polynomial equations of T_i

$$P(T_i) = 0 \tag{3}$$

where P is a high order polynomial. The look-ahead feedrate planning module must solve these non-linear high order polynomial equations in real-time. Since the analytic solutions are difficult to obtain, the iterative methods (such as Newton–Raphson method) are employed in the algorithm [11].

2.3. Look-ahead feedrate planning

In the motion planning, several successive short lines can be read in advance (known as "look-ahead") to generate an optimized feedrate profile. The look-ahead buffer is a First In, First Out (FIFO) queue. There are two look-ahead queues in the proposed design: the Short Line Queue (SLQ) and the Planning Unit Queue (PUQ). Suppose there are n short lines in the SLQ, the look-ahead procedure is described as follows:

1. If the SLQ and PUQ are not full, and there are new short lines available, a short line can be added to the SLQ, and it will become the $(n+1)$th items in the SLQ. Otherwise, go to step 6.
2. If the programmed feedrate is v_p, the maximum centripetal acceleration of the machine tool is A_{cm} and the estimated curvature radius for the short line is $\hat{\rho}(n+1)$, the target velocity, $v_t(n+1)$, can be calculated as:

$$v_t(n+1) = \min\left\{\sqrt{A_{cm}\hat{\rho}(n+1)}, v_p\right\} \tag{4}$$

3. Suppose the maximum allowable velocity step for the X, Y and Z axes are Δv_{mx}, Δv_{my} and Δv_{mz} respectively, and the change in the feed direction at the corner is $\Delta\vec{u}(n) = [\Delta u_x(n), \Delta u_y(n), \Delta u_z(n)]$, the velocity limit at the corner between the nth and the $(n+1)$th short lines, $v_{cl}(n)$, will be

$$v_{cl}(n) = \min\left\{\frac{\Delta v_{mx}}{\Delta u_x(n)}, \frac{\Delta v_{my}}{\Delta u_y(n)}, \frac{\Delta v_{mz}}{\Delta u_z(n)}\right\} \tag{5}$$

4. To improve the continuity in the motion velocity, the target feedrate can be adjusted. Given the relative feedrate tolerance α_- and α_+ $(0 \le \alpha_-, \alpha_+ < 1)$, if

$$(1-\alpha_-)v_t(n) \le v_t(n+1) \le (1+\alpha_+)v_t(n) \tag{6}$$

satisfies, change $v_t(n+1)$ to $v_t(n)$.

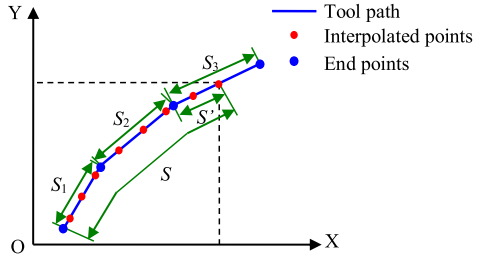

Fig. 5. Interpolation of a planning unit.

5. Check whether the new short line can be appended to the last planning unit. If $n > 0$, and

$$\begin{cases} v_t(n+1) = v_t(n) \\ v_{cl}(n) \le v_t(n) \end{cases} \tag{7}$$

the new short line will be appended to the last planning unit in the PUQ; otherwise, a new planning unit will be created and added to the PUQ.
6. Applying "backward planning" algorithm [11] iteratively on each planning units in the PUQ to determine the maximum end velocity, v_{et}, for the current planning unit (the first unit in the PUQ).
7. Determine the motion profile type and calculate the motion time of each motion phase for the current planning unit by "forward planning".

The look-ahead feedrate planning is performed periodically in the control and the motion profile is optimized in real-time.

2.4. Interpolation calculation

The interpolation is essentially the position sampling on the tool path, as shown in Fig. 5. Once the motion profile is determined, the distance along the planning unit, S, can be obtained by (2). If the kth short line of the planning unit is being interpolated ($i, k = 1, 2, 3, \cdots$), the distance along the short line, S', will be

$$S' = \begin{cases} S & k = 1 \\ S - \sum_{i=1}^{k-1} S_i & k > 1 \end{cases} \tag{8}$$

where S_i is the length of the ith short line in the planning unit .The interpolated point, $\vec{P}$, can be found as

$$\overrightarrow{P} = \vec{P}_k + S'\vec{u}_k \tag{9}$$

where $\vec{P}_k$ is the starting point of the short line and $\vec{u}_k$ is the unit vector of feed direction obtained in the geometrical analysis.

2.5. Summary of the algorithm

For the purpose of designing an efficient custom hardware architecture, the properties of the motion planning and interpolation algorithm should be considered. From the viewpoint of computing, the properties of the above mentioned algorithm can be summarized as following:

1. There are many high-order polynomial calculations in the motion planning algorithm, such as in (2) and solving (3) by the numerical method.
2. Vector calculations are intensively involved, since the tool path is presented in a 3D space.
3. The basic operations in the algorithm include addition, subtraction, multiplication, division, and square root.

Table 1
Polynomial evaluation example.

Clock	×	+
1	x^2	
2	Cx	
3	Ax	
6	ax^2	
7		$Cx + D$
8	Ax^3	
11		$ax^2 + c$
13		$Ax^3 + Cx + D$
18		Result is available

4. The algorithm, including the iterative steps in the look-ahead, must be executed in real-time to ensure a continuous motion.

3. Processor design

3.1. Design objectives

In the high speed free-form surface machining, the CNC control may process hundreds of thousands of short lines in a minute. As we can see, the feedrate planning and interpolation algorithm for high speed machining is nonlinear, iterative and computational intensive. Therefore, executing the algorithm with high performance on a low-cost hardware platform is challenging. Since many low-cost controls are equipped with FPGA chips to implement custom logics, the available resources in the FPGA can be exploited to accelerate the algorithm. In this research, a dedicated soft-core processor for motion planning and interpolation is proposed. According to the characteristics of the feedrate planning and interpolation algorithm, the following design objectives were identified:

1. Including a hardware floating-point unit.
   High-order polynomials, division and square root operations are strongly nonlinear, which can lead to a large dynamic range. Using the floating-point number to handle the large dynamic range and ensure the accuracy in the nonlinear calculations.
2. Applying parallel computing techniques to improve the performance.
   Obviously, the vector calculations will benefit from the parallel computing, since the components in vectors are independent. High-order polynomial evaluation can also be accelerated by parallel computing. For example, to evaluate $f(x) = Ax^3 + Cx + D$and its derivative $f'(x) = ax^2 + c$in the Newton–Raphson iteration on a 5-stage addition pipeline and a 5-stage multiplication pipeline in parallel, a possible computing order in 17 clocks is listed in Table 1. If the polynomials are not calculated in parallel, it will take 40 clocks (3 additions and 5 multiplications). Furthermore, the time slots in the multi-cycle floating-point instructions can be utilized to execute non-conflicting operations in parallel.
3. Supporting branch, stack and subroutine instructions in order to facilitate modular programming.
4. The processor, including data and instruction memories, should be able to be fitted into a low-cost FPGA chip.

3.2. Instruction set

The Instruction Set Architecture (ISA) is the interface between hardware and software. Today, RISC and VLIW are the most popular ISAs in the microprocessor. The RISC architecture mostly execute one simple operation in an instruction (usually 32 bits), thus

Table 2
Integer instructions.

Syntax		Description	Cycles
NOP		No operation	1
AND	Rd, Rx, Ry	Logical AND	1
OR	Rd, Rx, Ry	Logical OR	1
XOR	Rd, Rx, Ry	Logical XOR	1
ADD	Rd, Rx, Ry	Add	1
SUB	Rd, Rx, Ry	Subtract	1
ANDI	Rd, Rx, #imm	AND immediate	1
ORI	Rd, Rx, #imm	OR immediate	1
XORI	Rd, Rx, #imm	XOR immediate	1
ADDI	Rd, Rx, #imm	Add immediate	1
SUBI	Rd, Rx, #imm	Subtract immediate	1
JMP	Label	Absolute jump	2
JZ<.sel>	Label	Jump if zero	1/2
JNZ<.sel>	Label	Jump if non zero	1/2
JNEG<.sel>	Label	Jump if negative	1/2
JPOS<.sel>	Label	Jump if positive	1/2
CALL	Label	Call a subroutine	2
LDPC		Return from a subroutine	2
ST	[Rx], Ry	Store integer register	1
LD	Rd, [Rx]	Load integer register	1
LUI	Rd, #imm	Load upper immediate	1
SVFL	Rd	Save result flag	1
PUSH	Rx	Push register onto stack	1
POP	Rd	Pop register from stack	1

Table 3
Floating-point instructions.

Syntax		Description
FNOP		No operation
TRNS.GET	FRd, {Rx+ 1, Rx}	Transfer from integer registers
TRNS.SEND	{Rd+ 1, Rd}, FRx	Transfer to integer registers
FMOV	FRd, FRx	Move
FADD.SEND	FRx, FRy	Send to add pipeline
FSUB.SEND	FRx, FRy	Send to subtraction pipeline
FMUL.SEND	FRx, FRy	Send to multiplication pipeline
FDIV.SEND	FRx, FRy	Send to division pipeline
FSQRT.SEND	FRx	Send to square root pipeline
FADD.GET	FRd	Get addition result
FSUB.GET	FRd	Get subtraction result
FMUL.GET	FRd	Get multiplication result
FDIV.GET	FRd	Get division result
FSQRT.GET	FRd	Get square root result
FABS	FRd, FRx	Get absolute value
FOPP	FRd, FRx	Get opposite value
FSVFL	Rd	Save floating-point result flag
FMEM.SEND.L	[Rx], FRx	Store register lower word
FMEM.SEND.H	[Rx], FRx	Store register higher word
FMEM.GET.L	FRd, [Rx]	Load register lower word
FMEM.GET.H	FRd, [Rx]	Load register higher word
FSTK.SEND.L	FRx	Push register lower word
FSTK.SEND.H	FRx	Push register higher word
FSTK.GET.L	FRd	Pop register lower word
FSTK.GET.H	FRd	Pop register higher word

the design of RISC processor is compact and can achieve good pipeline performance. On the other hand, the VLIW architecture issues multiple independent operations within a long instruction word (usually more than 64 bits) to exploit instruction level parallelism. For the embedded application, the RISC architecture are typically general purpose processors (such as ARM and MIPS processor), while the VLIW are usually employed in the computing intensive applications (such as DSP systems). The RISC and VLIW are often very different designs and their binaries are usually incompatible. To take the advantages of these two different ISAs, a new compact design that combines the RISC and VLIW instructions is introduced in this research.

The instruction set of the processor has two parts: the integer instructions and the floating-point instructions, as shown in Tables 2 and 3. Each integer instruction is 32-bit wide and each

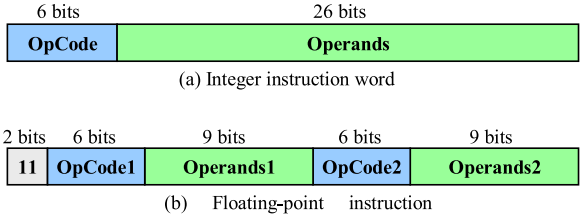

Fig. 6. Instruction word format.

floating-point instruction is 15-bit wide. The instruction word is 32-bit wide, which either encodes one integer instruction or two floating-point instructions, as illustrated in Fig. 6.

The integer part implements classic RISC instructions, including arithmetic, logical, branch, data moving and subroutine operations. The integer register file contains 16 32-bit general purpose registers (R0-R15). Among these registers, R14 also serves as the stack pointer (SP) and R15 can be used as the link register (LR), which holds the return address in a subroutine call.

The floating-point instructions focus on arithmetic and data moving operations. Each floating-point instruction takes only one cycle. The floating-point part is a dual-issue VLIW design, which enables the processor to execute two floating-point instructions in parallel. The floating-point register file contains 16 general purpose floating-point registers (FR0-FR16).

The standard IEEE 754 single precision floating-point format has a 23-bit fraction field, which can provides a position resolution of 1.2 μm for a machine tool with 10 m travel distance. However, modern machine tools has a typical resolution of 1 μm, which is beyond the accuracy of the single precision number. In addition, due to the nonlinearity in the algorithm, the accuracy of the interpolated position value is very sensitive to the round-off errors in the intermediary calculation steps. Therefore, extra guard bits are necessary [34]. Considering the trade-off between the precision and the resource usage, a 36-bit custom floating-point format is employed, including a sign bit, a 27-bit fraction field and an 8-bit exponent field. This format provides a position resolution of 0.075 μm with 10 m travel distance.

3.3. Pipelined RISC core

The processor employs a 3-stage pipelined Harvard RISC core. In the instruction pipeline, instructions can be processed simultaneously in 3 different stages, as shown in Fig. 7. The functions of the stages are summarized as follows:

Instruction Fetch (IF): The processor reads an instruction word from the program memory and sends it to the next stage. The program address is generated by the program counter (PC). The address in the PC is also updated in this stage, which is selected by the instruction decoder.

Instruction Decode (ID): In this stage, the registered instruction word is either decoded by the integer instruction decoder or divided into two sub-words and decoded by the two floating-point decoders. The operands can be obtained from the register files or decoded from the instruction word.

Execute (EXE): The operation is executed in the Arithmetic Logical Unit (ALU) or Floating-point Unit (FPU), and the register files or the data memory can be accessed.

3.4. VLIW floating-point unit with asynchronous execution

The processor implements the dual-issue VLIW architecture to improve the performance in floating-point operations, as shown in Fig. 8. The add/subtraction, multiplication, division and square

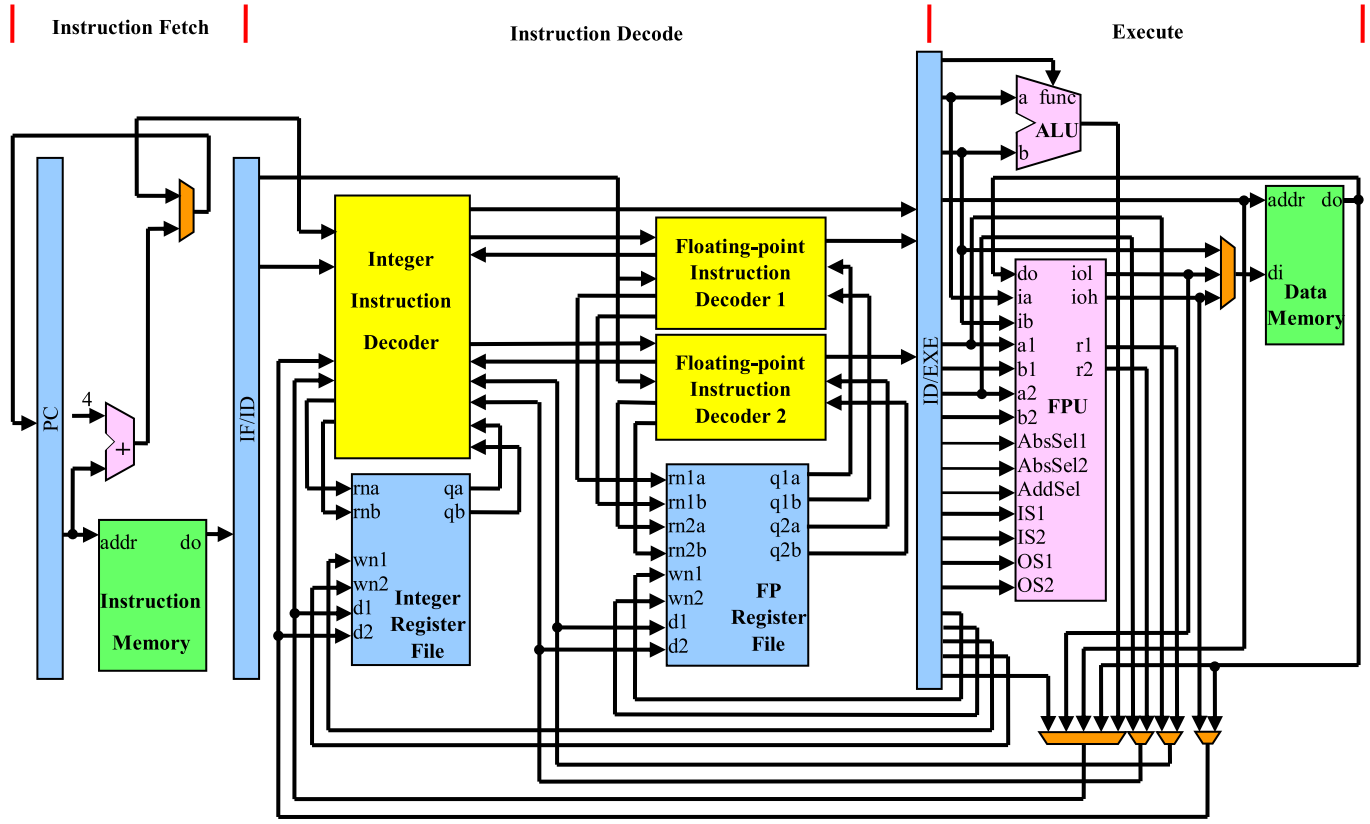

Fig. 7. Block diagram of the instruction pipeline.

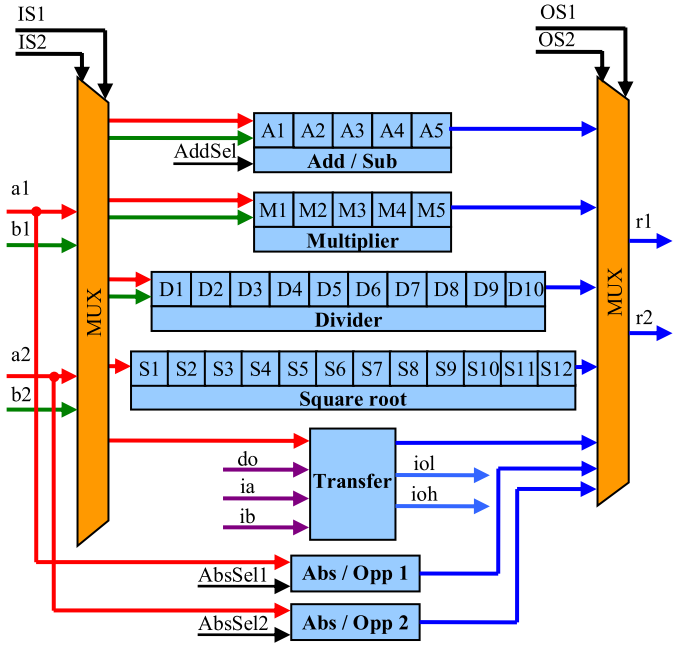

Fig. 8. Block diagram of the FPU.

root operations are pipelined, while the data transfer and absolute/opposite value operations are non-pipelined. Except absolute/opposite value, all the processing resources are shared with the two floating-point channels. The operands and results are selected by the input selection (IS1, IS2) and the output selection (OS1, OS2) signals of individual channel.

In the conventional processor design, the source and destination registers are packed in the same instruction word and pass through the processor pipeline synchronously. Although this method facilitates programming, there are also some drawbacks. First, in order to encode the source and destination registers in a single instruction, the instruction word will be very long. Second, the decoding circuit is complicated since all the register addresses in the instruction should be decoded. Third, the destination address must be passed through the floating-point pipeline, which requires additional registers at each stage. These drawbacks make it difficult to realize VLIW floating-point unit in a compact design.

To overcome these drawbacks, a new asynchronous execution mechanism is introduced in the FPU to improve the performance and resource usage. The word “asynchronous” here indicates that the source and destination registers are assigned in separate instructions (i.e., SEND and GET) that executed at different time. The asynchronous execution splits a pipelined operation into two floating-point instructions, namely SEND and GET, as shown in Table 3. When executing a SEND instruction, the operands in the floating-point registers will be sent to the first stage of the designated pipeline. When executing a GET instruction, the required result is fetched and written to the floating-point register. Between a SEND and GET pair, there are 4 to 11 delay slots available, which can be utilized to perform any non-conflicting instructions. However, the asynchronous execution mechanism requires precise timing in the programming to guarantee the GET instruction can obtain the desired result at the correct time.

Since the destination register is designated in the GET instruction, there is no need to register the destination register address in the pipelines. The split of source/destination register addresses by the SEND/GET pair also enables a shorter floating-point instruction word (only 15 bits in the study). As shown in Fig. 6, each floating-point instruction has a 9-bit operands field, which is suf-

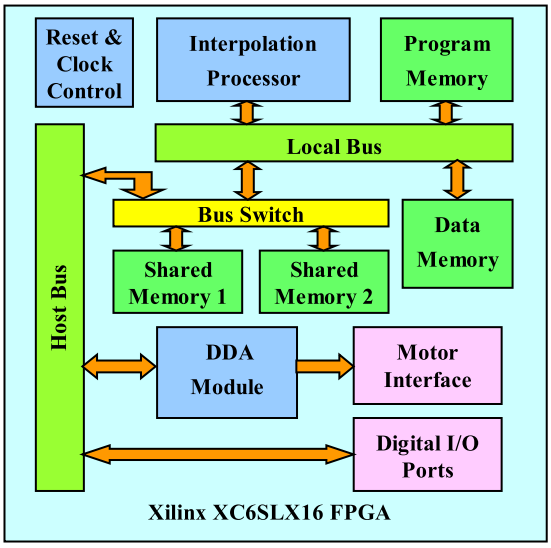

Fig. 9. Structure of the FPGA based motion control chip.

ficient to include two source addresses or one destination address. Furthermore, the same register address decoding circuit can be shared with the SEND and GET instructions. By introducing the asynchronous execution mechanism, the proposed design has successively packed two floating-point operations in the same 32-bit instruction word and saves a lot of routing and register resource. This is the key technique that enables a dual-issue VLIW floating-point unit within a 32-bit RISC processor.

4. Implementation and experimental results

4.1. Controller hardware

A 3-axis CNC controller based on a microcontroller and a low-cost FPGA chip was built to verify the proposed design. The microcontroller was based on the ARM Cortex-M4 core (STM32F405ZG, from STMicroelectronics) and connected to the Xilinx XC6SLX16 FPGA chip by the flexible static memory controller (FSMC) interface.

The custom motion control logic in the FPGA is shown in Fig. 9. The interpolation subsystem consists of the proposed processor, a 16 KB program memory, a 16 KB data memory, a bus switch and two shared memories. The bus switch is controlled by the interpolation processor and the host can monitor the switch status through a status register. If the shared memory 1 is switched to the host bus, the shared memory 2 will be connected to the interpolation processor, and vice versa. The motor interface uses pulse and direction signals to control the servo motors. The Digital Differential Analyzer (DDA) module is used to generate the pulse commands according to the interpolated position. The custom logics, including the processor, were developed using the Verilog Hardware Description Language (HDL). The implementation used 2255 logic slices (98%), 16 block RAM blocks (50%), 4 DSP48A1 slices (12%). It can be seen that the design has exploited the available resources in the FPGA and no additional hardware (such as memory chips) is required to support the processor. When the processor clock is 66.67 MHz, the total power consumption of the FPGA is 137.05 mW (98.22 mW dynamic + 38.83 mW quiescent, reported by the Xilinx XPower Analyzer). Since the CNC machine tools typically has a power consumption of 10∼100 kW, the power consumption of the FPGA is negligible. Nevertheless, the power consumption is reasonable for many embedded applications. For reference, the STM32F405ZG microcontroller has a typical power consumption of 151.8∼306.9 mW at 168 MHz clock [35].

4.2. Firmware design

The firmware implements the feedrate planning and interpolation algorithms introduced in Section 2. The firmware was optimized for the processor to utilize the parallel floating-point computing ability. Fig. 10 shows a piece of the assembly code of the firmware (from the forward planning subroutine in the look-ahead module). It can be seen that in the computing intensive task, the firmware has efficiently implemented by using the dual-issue VLIW floating-point instructions and the delay slots in the asynchronous execution. A simple assembler was developed to translate the source code.

The firmware uses a protocol to communicate with the host microcontroller. At the beginning, the host writes the parameters to the interpolation processor. In the interpolation phase, the host sends the coordinates and programmed feedrate of the short lines. The interpolation processor will append the newly received data to its internal input buffer and execute the interpolation task. The coordinates of the interpolated points will be sent to the host. By employing the two shared memories, the host and the interpolation processor can read/write the communication data at a same time, which allows the host to prepare the short line data for the next communication cycle while the interpolation is running.

4.3. Benchmark test

To evaluate the performance of the proposed interpolation processor, two benchmark test programs were developed. Both benchmark tests have many successive short line blocks, which are commonly used for the rough finishing and fine finishing process in the CNC machining. This type of machining program requires high performance CNC control to processing the large data in real-time in order to guarantee smooth high-speed motion. The first program is a 2D contouring tool path with 1503 short lines, as shown in Fig. 11. The second program simulates the machining process of a complex 3D surface, which has a tool path consist of 44,879 short lines, as shown in Fig. 12. The total execution time of the interpolation subroutines was recorded by a performance timer in the FPGA. The performance timer is a controllable accumulator for the processor clock. The interpolation processor enables the accumulator when entering the interpolation subroutines, and disables it when leaving the interpolation subroutines. The total execution time (the number of processor clock ticks) of the interpolation subroutines can be read by the host after the benchmark test is finished. In the test, the program feedrate was 6000 mm/min, the size of SLQ and PUQ was 100 and 10 respectively, and the interpolation period was 0.0001 s.

To compare the performance of the interpolation processor, the benchmark test programs were ported to some popular low cost processors, including the STM32F405 (ARM Cortex-M4) from STMicroelectronics, AT91SAM9G45 (ARM9) from Atmel and Microblaze soft processor core from Xilinx. The test results are summarized in Tables 4 and 5. In the test, the Microblaze was deployed on the same FPGA chip used to verify the interpolation processor. The result shows that the proposed design can perform much better than the other platforms in the benchmark test. In the 3D surface benchmark test, the proposed design can process 1877.5 short lines per seconds on average, which is sufficient for the high speed complex surface machining. It is also noticed that, by utilizing the specially designed architecture, the interpolation processor can achieve up to 50 times performance increase, compared to the standard Microblaze soft-core processor on the same FPGA chip.

```
FMUL.GET FR6 || FADD.GET FR7                  ; FR6 = tmp[0] = a*a || FR7 = 2*c
FMUL.GET FR8 || FOPP FR7, FR7                 ; FR8 = tmp[1] = b*b || FR7 = tmp[5] = -2*c
FMUL.GET FR9 || FADD.GET FR15                 ; FR9 = - a*d || FR15 = tmp[4] = 2*b
FMUL.GET FR0 || FOPP FR9, FR9                 ; FR0 = tmp[2] = e*e || FR9 = tmp[3] = a*d
FADD.SEND FR7, FR15 || FMUL.SEND FR13, FR13   ; tmp[5]+tmp[4] || c*c
FSUB.SEND FR15, FR11 || FMUL.SEND FR12, FR11 ; tmp[4]-e || b*e
FADD.SEND FR0, FR0                            ; 2*tmp[2]
FMUL.SEND FR5, FR9                            ; S*tmp[3]
FSTK.GET.L FR15
FADD.GET FR1 || FMUL.GET FR13                 ; FR1 = tmp[5]+tmp[4] || FR13 = c*c
FSUB.GET FR2 || FMUL.GET FR3                  ; FR2 = tmp[4]-e || FR3 = b*e
FADD.GET FR12 || FADD.SEND FR3, FR3           ; FR12 = 2*tmp[2] || 2*b*e
```

Fig. 10. Source code in the firmware.

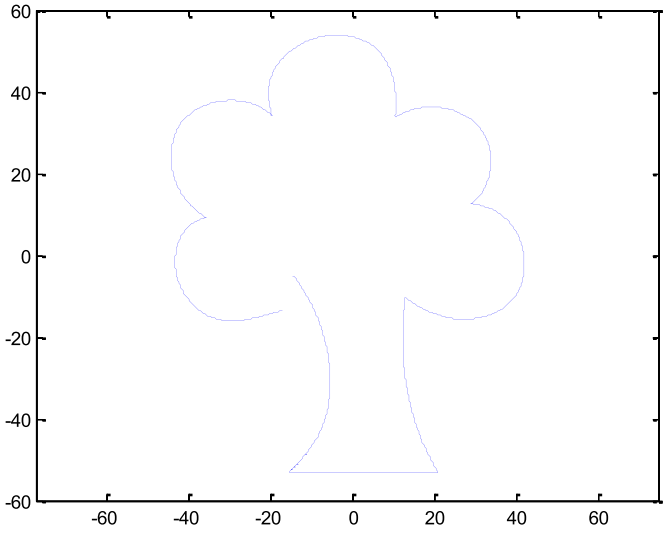

Fig. 11. Tool path of the 2D contouring benchmark test.

It should be mentioned that the firmware of the interpolation processor takes advantage of the delay slots of the VLIW floating-point instructions to obtain high performance. Two instruction counters were deployed in the FPGA to count the total number of executed instructions and the number of NOP instructions respectively. The NOP instructions only occupied 5.58% of the processing time in the 3D surface benchmark test, which implies the time slots were well utilized to achieve parallelism for the algorithm.

4.4. Performance tests with different parameters

The PUQ and SLQ sizes are important design parameters for the look-ahead feedrate planning for the short line tool path. If the PUQ and SLQ sizes are small, the machining time will be too long. On the other hand, when the SLQ and PUQ are large enough, increasing the sizes of PUQ or SLQ will not decrease motion time significantly, since the motion profile has nearly fully optimized by the look-ahead function. Large PUQ and SLQ sizes will also increase the computing burdens and memory usage.

For the guidance of choosing PUQ and SLQ sizes, a motion time test was conducted. In the motion time test, the 3D surface machining tool path (Fig. 12) was employed, and the motion parameters were the same as in the benchmark test. The motion time of the tool path with different combinations of PUQ size and SLQ size was recorded during the test, as shown in Fig. 13. The result shows that, when the (PUQ, SLQ) size pairs were (2, 10), (10, 100) and (20, 200), the motion time were 1639.281, 461.252 and 460.724 s respectively. Considering the efficiency, the PUQ and SLQ sizes were set to 10 and 100 in the benchmark test.

Several performance tests with different parameters were also carried out with the 3D surface machining tool path. The results of these tests are plotted in Fig. 14. Fig. 14(a) shows the interpolation time decreases with the increasing in the interpolation period, since the interpolation calculations are less with larger interpolation period. When the interpolation period is large (more than 0.002 s), the interpolation task only occupied small proportion of computing load, therefore the reduction in the interpolation time became slow. Fig. 14(b) and (c) shows the effect of the programmed feedrate on the motion time and interpolation time. Obviously, the lower programmed feedrate resulted in longer motion time. However, when the programmed feedrate was larger than 8000 mm/min, the motion time was limited by the motion planning constraints (acceleration, jerk, etc). The interpolation time shows a similar trend, since the interpolation load depends on mainly on the motion time in this condition. Fig. 14(d) and (e) illustrates the impact of the SLQ and PUQ sizes on interpolation time. Again, increasing SLQ and PUQ sizes had reduced the interpolation time because the motion time was shorten and the interpolation computing load was reduced. It is not surprised that Fig. 14(f) implies the average throughput has a linear relation with the processor clock, since the instruction execution cycle is totally determined in the interpolation processor.

4.5. Improvement on the host processor load

A simple CNC control program was built on the prototype hardware to verify the feasibility of the proposed system design and evaluate the processor load in the real-time machining. The tool path shown in Fig. 12 was used in the test. The control program implements all the CNC tasks in three preemptive priority levels. The highest priority is dedicated to the motor control task, which is executed in a fixed period of 0.0001 s. The motor control task sends the interpolated position to the DDA module to generate command pulses. The second level priority tasks are invoked every 0.001 s. This priority includes the motion planning, interpolation and discrete I/O control tasks. The lowest priority tasks run in the background, including all the non-real-time services.

The control program has two versions: one version uses both the host and the interpolation processors, while the other version only uses the host processor. In the host interpolation task, the execution time was recorded to calculate the task load on the processor. The test result is summarized in Table 6. The result shows that, in the single processor case, the interpolation task occupied 29.1% of the processing time on average. After enabling the interpolation processor, since the computational intensive motion planning and interpolation algorithms were removed from the host real-time tasks, the average processor load had dramatically dropped

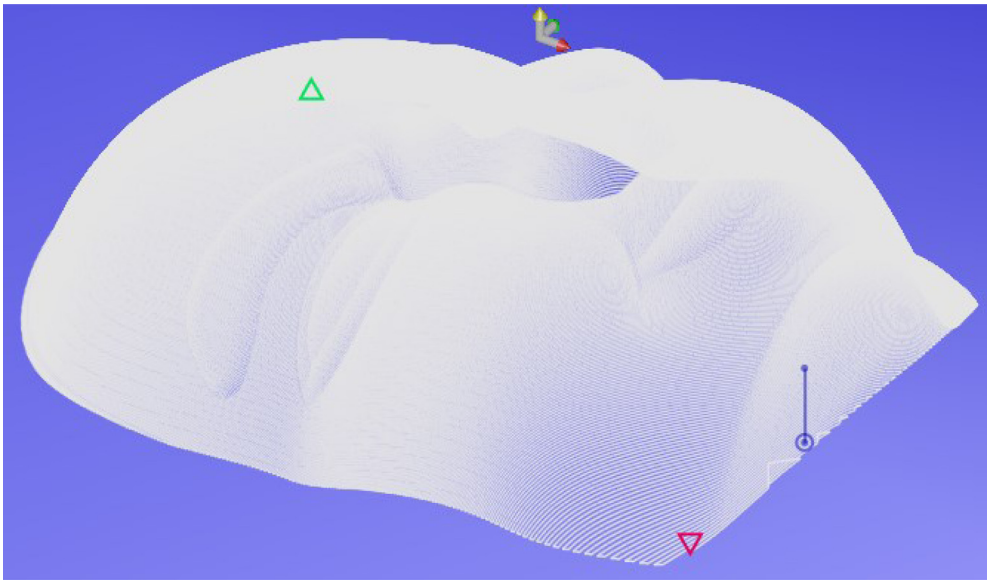

Fig. 12. Tool path of the 3D surface machining benchmark test.

Table 4
2D contouring benchmark test results.

Hardware platform	Frequency (MHz)	Interpolation time (s)	Average throughput (lines/s)
Interpolation processor	66.67	0.325	4624.6
STM32F405	168.00	1.014	1482.2
AT91SAM9G45	400.00	6.111	245.9
Microblaze	100.00	9.328	161.1

Table 5
3D surface machining benchmark test results.

Hardware platform	Frequency (MHz)	Interpolation time (s)	Average throughput (lines/s)
Interpolation processor	66.67	23.904	1877.5
STM32F405	168.00	126.785	354.0
AT91SAM9G45	400.00	681.429	65.9
Microblaze	100.00	1206.644	37.2

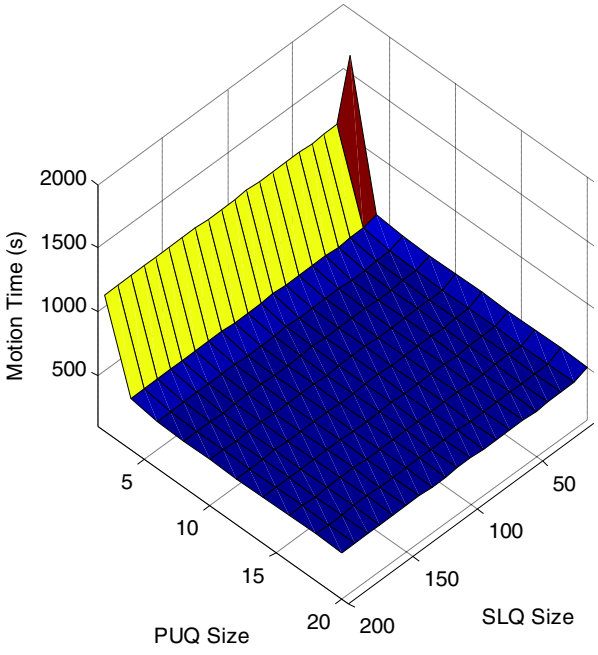

Fig. 13. Motion time test result with different PUQ and SLQ sizes.

Table 6
Results for the processor load test.

Results	Without interpolation processor	With interpolation processor
Average load	29.1%	2.7%
Peak load	85.6%	5.8%
Standard deviation	17.3%	0.3%

to 2.7%. The peak load in the test had reached to 85.6% in the single processor test. The difference between the peak and average load was attributed to the complexity of the motion planning flow, in which the number of calculation steps differs significantly for different motion types. For the dual-processor case, the peak load of the interpolation task was only 5.8%, and the standard deviation of the load was reduced to 0.3%. As a result, the host can release more processing power for other real-time tasks when employing the interpolation processor.

5. Conclusions

A new floating-point interpolation processor design for low-cost CNC controllers with FPGAs is proposed in this paper. In the system design with the interpolation processor, the computational intensive real-time motion planning and interpolation tasks can be

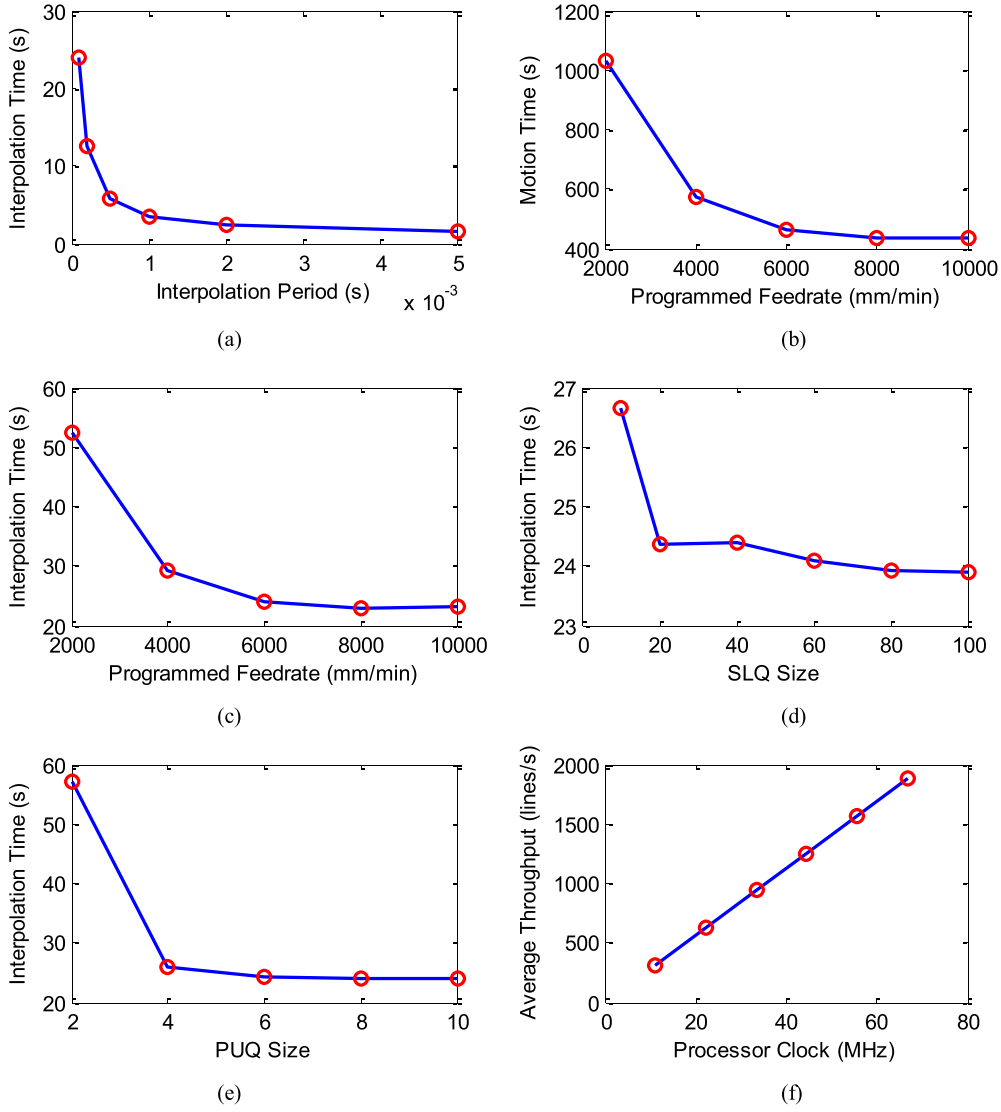

Fig. 14. Experiment results with different parameters.

executed on the FPGA with high performance, and the host processor can perform other tasks in parallel. The jerk-limited look-ahead motion planning and interpolation algorithm for high-speed free-form surface machining with short lines is presented and implemented in the firmware of the interpolation processor to generate a smooth motion profile. To achieve high performance, the hardware design of the processor utilizes several parallel execution methods. A 3-stage pipelined RISC core and a dual-issue VLIW FPU are employed to improve the computational performance. The ISA of the processor allows encoding one RISC operation or two VLIW operations in a single 32-bit instruction word. The novel asynchronous execution mechanism allows the latency in the floating-point pipeline to be utilized for non-conflicting operations while keeping a compact architecture.

In the experiment, a prototype controller was built to verify the proposed design. The interpolation processor was implemented on a low-cost FPGA chip with other custom logics. Benchmark test results shows that the performance of the interpolation processor achieved an average throughput of 1877.5 lines per second in the free-form surface machining, which is absolutely superior to the other low-cost processors compared in the test. The efficiency of the integrated system was also demonstrated in the prototype controller. Compared to the single processor design, the host average load on the interpolation task was reduced from 29.1% to 2.7%, and the peak load was reduced from 85.6% to 5.8%. The result demonstrates that the proposed processor can greatly improve the performance of the low-cost CNC controller in high speed applications. Besides, the proposed design requires no additional hardware cost other than the existing resources in the FPGA chips, which enables the possibilities for using in other low-cost applications. The source code of the processor can be downloaded from: https://github.com/tsxxjq/VLIWFPProcessor.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 51605328 and 51475324, and the Open Research Fund of Jiangsu Key Laboratory of Digital Manufacturing for Industrial Equipment and Control Technology under Grant DM2015001.

References

[1] M.R. Khoshdarregi., S. Tappe, Y. Altintas, Integrated five-axis trajectory shaping and contour error compensation for high-speed CNC machine tools,, IEEE/ASME Trans. Mechatronics 19 (6) (Dec. 2014) 1859–1871.
[2] L. Tang, R.G. Landers, "Predictive contour control with adaptive feed rate,, IEEE/ASME Trans. Mechatronics 17 (4) (Feb. 2012) 669–679.

[3] Y. Wang, Y. Zhao, S.A. Bortoff, K. Ueda, "A real-time energy-optimal trajectory generation method for a servomotor system„ IEEE Trans. Ind. Electron. 62 (2) (2015) 1175–1188.
[4] K. Erkorkmaz, Y. Altintas, "High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation, Int. J. Mach. Tools Manuf. 41 (9) (Jul. 2001) 1323–1345.
[5] X. Beudaert, Lavernhe S., C. Tournier, Feedrate interpolation with axis jerk constraints on 5-axis NURBS and G1 tool path, Int. J. Mach. Tools Manuf. 57 (Jun. 2012) 73–82.
[6] K.-H. Rew, K.-S. Kim, A closed-form solution to asymmetric motion profile allowing acceleration manipulation, IEEE Trans. Ind. Electron. 57 (7,) (Jul. 2010) 2499–2506.
[7] M.-T. Lin, M.-S. Tsai, H.-T. Yau, Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm, Int. J. Mach. Tools Manuf. 47 (15) (Dec. 2007) 2246–2262.
[8] Y.S. Wang, D.S. Yang, R.L. Gai, S.H. Wang, S.J. Sun, Design of trigonometric velocity scheduling algorithm based on pre-interpolation and look-ahead interpolation, Int. J. Mach. Tools Manuf. 96 (Sep. 2015) 94–105.
[9] C. Shi, P. Ye, The look-ahead function-based interpolation algorithm for continuous micro-line trajectories, Int. J. Adv. Manuf. Technol. 54 (5-8) (May. 2011) 649–668.
[10] J.R. Conway, A.L. Darling, C.A. Ernesto, R.T. Farouki, C.A. Palomares, Experimental study of contouring accuracy for CNC machines executing curved paths with constant and curvature-dependent feedrates, Robot. Comput.-Integr. Manuf. 29 (2) (Apr. 2013) 357–369.
[11] J. Dong, T. Wang, B. Li, Y. Ding, Smooth feedrate planning for continuous short line tool path with contour error constraint, Int. J. Mach. Tools Manuf. 76 (Jan. 2014) 1–12.
[12] K. Erwinski, M. Paprocki, L.M. Grzesiak, K. Karwowski, A. Wawrzak, Application of Ethernet Powerlink for communication in a Linux RTAI open CNC system, IEEE Trans. Ind. Electron. 60 (2) (Feb. 2013) 628–636.
[13] R. Dubey, P. Agarwal, M.K. Vasantha, Programmable logic devices for motion control—a review, IEEE Trans. Ind. Electron. 54 (1) (Feb. 2007) 559–566.
[14] H.-H. Chiang, K.-C. Hsu, I-H. Li, Optimized adaptive motion control through an SoPC implementation for linear induction motor drives, IEEE/ASME Trans. Mechatronics 20 (1) (Feb. 2015) 348–360.
[15] A. Ghaffari, A.G. Ulsoy, Dynamic contour error estimation and feedback modification for high-precision contouring, IEEE/ASME Trans. Mechatronics 21 (3) (2016) 1732–1741.
[16] Y.-S. Kung, R.-F. Fung, T.-Y. Tai, Realization of a motion control IC for X-Y table based on novel FPGA technology, IEEE Trans. Ind. Electron. 56 (1) (Jan. 2009) 43–53.
[17] J.U. Cho, Q.N. Le, J.W. Jeon, An FPGA-based multiple-axis motion control chip, IEEE Trans. Ind. Electron. 56 (3) (Mar. 2009) 856–870.
[18] H.-T. Yau, M.-T. Lin, Y.-T. Chan, K.-C. Yuan, Design and implementation of real–time NURBS interpolator using a FPGA-based motion controller, in: Proc. IEEE Int. Conf. Mechatronics, Taipei, Jul. 2005, pp. 56–61.
[19] K.D. Oldknow, I. Yellowley, FPGA-based servo control and three dimensional dynamic interpolation, IEEE/ASME Trans. Mechatronics 10 (1) (Feb. 2005) 98–110.
[20] R.A. Osornio-Rios, R.J. Romero-Troncoso, G. Herrera-Ruiz, R. Castaneda-Miranda, FPGA implementation of higher degree polynomial acceleration profiles for peak jerk reduction in servomotors, Robot. Comput.-Integr. Manuf. 25 (2) (Apr. 2009) 379–392.
[21] P. Ponce, A. Molina, H. Bastida, B. MacCleery, Real-time hardware ANN-QFT robust controller for reconfigurable micro-machine tool, Int. J. Adv. Manuf. Technol. 79 (1-4) (Jul. 2015) 1–20.
[22] S. Amornwongpeeti, M. Ekpanyapong, N. Chayopitak, J.L. Monteiro, J.S. Martins, J.L. Afonso, A single chip FPGA-based solution for controlling of multi--unit PMSM motor with time-division multiplexing scheme, Microprocess. Microsyst. 39 (8) (Nov. 2015) 621–633.
[23] C. Huang, F. Vahid, T. Givargis, A custom FPGA processor for physical model ordinary differential equation solving, IEEE Embedded Syst. Lett. 3 (4) (Dec. 2011) 113–116.
[24] H. Kim, S. Lee, Design and implementation of a private and public key crypto processor and its application to a security system, IEEE Trans. Consum. Electron. 50 (1) (Feb. 2004) 214–224.
[25] D. Stevens, V. Chouliaras, V. Azorin-Peris, J. Zheng, A. Echiadis, S.J. Hu, BioThreads: a novel VLIW-based chip multiprocessor for accelerating biomedical image processing applications, IEEE Trans. Biomed. Circuits Syst. 6 (3) (Jun. 2012) 257–268.
[26] Y. Gu, T. VanCourt, M.C. Herbordt, Explicit design of FPGA-based coprocessors for short-range force computations in molecular dynamics simulations, Parallel Comput. 34 (4-5) (May. 2008) 261–277.
[27] L. Morales-Velazquez, R.J. Romero-Troncoso, R.A. Osornio-Rios, G. Herrera-Ruiz, J.J. Santiago-Perez, Special purpose processor for parameter identification of CNC second order servo systems on a low-cost FPGA platform, Mechatronics 20 (2) (Mar. 2010) 265–272.
[28] Z. Hajduk, B. Trybus, J. Sadolewski, Architecture of FPGA embedded multiprocessor programmable controller, IEEE Trans. Ind. Electron. 62 (5) (May. 2015) 2952–2961.
[29] J.J. Rodriguez-Andina, Advanced features and industrial applications of FPGAs—a review, IEEE Trans. Ind. Inform. 11 (4) (May. 2015) 853–864.
[30] Y. Chen, V. Dinavahi, An Iterative real-time nonlinear electromagnetic transient solver on FPGA, IEEE Trans. Ind. Electron. 58 (6) (Jun. 2011) 2547–2555.
[31] W.T. Wang, Z.X. Shen, V. Dinavahi, Physics-based device-level power electronic circuit hardware emulation on FPGA, IEEE Trans. Ind. Inform. 10 (4) (Nov. 2014) 2166–2179.
[32] E.N. Hartley, J.L. Jerez, A. Suardi, J.M. Maciejowski, E.C. Kerrigan, G.A. Constantinides, Predictive control using an fpga with application to aircraft control, IEEE Trans. Control Syst. Technol. 22 (3) (May. 2014) 1006–1017.
[33] G.A. Malazgirt, A. Yurdakul, S. Niar, Customizing VLIW processors from dynamically profiled execution traces, Microprocess. Microsyst. 39 (8) (Nov. 2015) 656–673.
[34] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, USA, 2002.
[35] STMicroelectronics, DS8626: STM32F405xx STM32F407xx, 2016.

Jingchuan Dong received the B.E. and Ph.D. degree in mechanical engineering from Tianjin University, Tianjin, China, in 2005 and 2010, respectively. He is currently an Electrical and Mechanical Engineer with the School of Mechanical Engineering, Tianjin University. His current research interests include motion control, embedded systems, FPGA-based reconfigurable architecture and CNC machine tools.

Taiyong Wang received the B.E. degree in mechanical manufacturing from Shenyang University of Technology, Shenyang, China, in 1983, and the M.E. and Ph.D. degrees from Tianjin University, Tianjin, China, in 1986 and 1995 respectively. From 1992 to 1995, he was an Associate Professor with the Department of Mechanical Engineering, Tianjin University, Tianjin, where he has been a Professor since 1996 and is currently with the Tianjin Key Laboratory of Equipment Design and Manufacturing Technology. From October 2008 to March 2009, he was a Visiting Scholar with the Department of Mechanical Engineering at the University of Michigan, Ann Arbor, MI, U.S.A. His research interests include machine tools, manufacturing automation and mechanical dynamics.

Bo Li received the B.E. degree in mechanical engineering from Tianjin University, Tianjin, China, in 2010, where he is currently working toward the Ph.D. degree in the School of Mechanical Engineering. His research interests include motion control, intelligent control and embedded computing.

Zhe Liu received the B.E. degree in mechanical engineering from Taiyuan University of Technology, Taiyuan, China, in 2012. He is currently working toward the Ph.D. degree in mechanical engineering in the School of Mechanical Engineering, Tianjin University, Tianjin, China. His research interests include embedded systems, robotics and FPGA-based servo control.

Zhiqiang Yu received the B.E. degree in mechanical engineering from Southwest University, Chongqing, China, in 2013. He is currently working toward the Ph.D. degree in mechanical engineering in the School of Mechanical Engineering, Tianjin University, Tianjin, China. His research interests include embedded hardware design and high-speed CNC machining.