A HIGH PERFORMANCE ARCHITECTURE DESIGN OF PLC DEDICATED PROCESSOR

Shuting-zeng
Industrial informatics laboratory
Shenyang institute of automation， Graduate School of the Chinese Academy of Sciences
Shenyang, China
e-mail: zengshuting@sia.cn

Zhijia-yang
Industrial informatics laboratory
Shenyang institute of automation ,CAS
Shenyang, China
e-mail:yang@sia.cn

In order to improve the speed of executing PLC instructions, the high performance PLC processor is researched. The proposed high performance PLC dedicated processor consists of the general processor and the PLC application specific instruction set processor (ASIP), and regards the PLC ASIP as the core. In the PLC ASIP, four kinds of instruction formats and five kinds of instructions are designed. The architecture is designed to accelerate the instruction execution. The PLC ASIP can improve the speed of executing bool instructions, load and store instructions and function block instructions, which occupy [illegible] frequency of PLC instructions. The general processor is used for compiling the PLC concurrent program, controlling the peripheral equipments and executing arithmetic instructions. The general processor and the PLC ASIP can execute concurrently when executing instructions, the two processors are independent with each other. The proposed design helps to improve real-time performance, comparing to the traditional sequential execution of PLC program. To validate the advantage of the proposed design, three ladder programs are recompiled to the instructions of diverse processors. Compared the number of compiled instructions of diverse processors, the number of compiled instructions of the PLC dedicated processor are smallest.

[illegible]

I. INTRODUCTION

Programable logic controller (PLC) is a new servomechanism, which combines the technique of automation and communication, and regards the microprocessor as the core. Because of the high dependability , better anti-jamming ability, hard real-time, volatile control program, PLC is widely applied in the field of industry, such as steel, oil, chemical industry, electric power, building materials, machine manufacture, auto, traffic and so on.

In China, most industrial PLC microprocessor may adopt general microprocessor or consist of bool cooperating processor and general microprocessor. The general processor mostly deal with the data format like byte or word, but mostly PLC instructions are bool instructions, which occupy 69% frequency of PLC instructions.[1] So the general processor hardly satisfy PLC's practical application. Combining bool cooperating processor with general microprocessor may offer the module to execute bool instructions, but hasn't accelerator for executing bool instructions, load and store instructions and function block instructions.

Not only the PLC dedicated processor accelerates the speed of executing most instructions, but also offers the concurrent execution for instructions by dual core architecture to enhance the harder real-time.

II. PLC DEDICATED PROCESSOR

A. *The architecture of PLC dedicated processor*

The high performance PLC dedicated processor （see fig.1）includes a PLC application specific instruction set processor (ASIP) and a general processor. The PLC ASIP is the core of executing PLC instructions. The instruction set of PLC ASIP includes most instructions of PLC, except arithmetic instructions, which is executed in the general processor. Because most data type is bit in the PLC ASIP, the bit addressing mode is mainly used for data accessing. The PLC ASIP adopts direct accessing mode, which is suited for PLC's frequent data accessing.

The PLC ASIP and the general processor can execute concurrently. The PLC ASIP can interrupt the general processor for arithmetic instruction execution, and the next instruction can be executed in the PLC ASIP, if the two instructions haven't dependence relationship with each other.

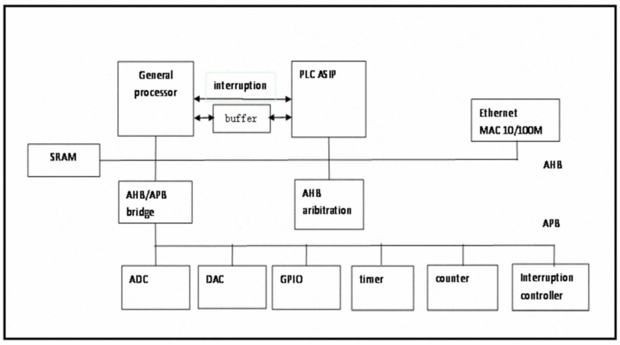

Figure 1. Architecture of PLC dedicated processor

III. PLC ASIP

The PLC ASIP can improve the speed of executing most PLC instructions by its instruction set and architecture.

A. PLC specific instruction set

Because bool instructions occupy the 69% frequency of PLC instructions, the bool instruction set is setted in the PLC specific instruction set.

In order to accelerate the speed of executing bool instructions, a skippable bool instruction set is setted. Analyzing a ladder program (see fig.2) ,in the first step, if the value of RE }is zero, and the next instruction is "AND(" , then result of the "AND(" instruction is "0" and the instructions "NOT Rj} ORN X2" need not be executed .[2]If the PLC ASIP comes across the PLC's IL instructions , such as "AND（"、"ANDN（"、"OR（"、"ORN（"，it can accelerate the executing process by the skippable bool instruction set.

Another character of PLC ladder program is step's sequence execution and dependency relationship between steps. The arithmetic instructions and function block instructions and boo : instructions can execute concurrently, when they are in different steps without dependency relationship. See the figure2, the step with "TON" function block and its next step can execute concurrently. The function block instruction set is setted in the PLC specific instruction set for the concurrent execution.

Because most of the data type is bit in the PLC ASIP, the register file and data RAM adopt bit addressing mode for data accessing. And the PLC's data accessing is frequent, so the load and store instruction set adopts direct accessing mode.

According to the characters of PLC program, the instruction set can be divided to be five kinds: bool instruction set; skippable bool instruction set; jump or call instruction set; load and store instruction set; function block instruction set. Some instructions are the part of the PLC ASIP's instruction set in the table 1. The part c in fig.2 is the translated assembler, using the instruction set of the PLC ASIP.

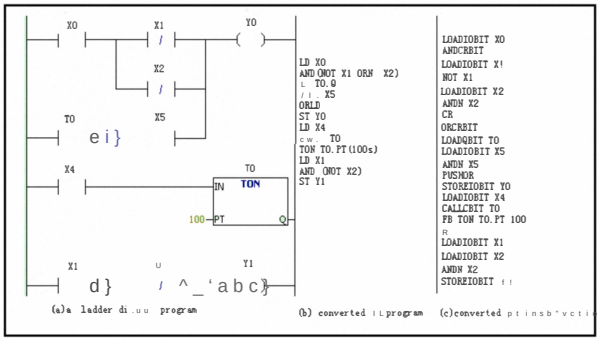

Figure 2. (a)A ladder program(b)Translated IL program、(c)Translated assembler program of PLC ASIP

Table ;.}Part of the PLC ASIP instruction set

FpXnZ\}	Fp\rUkZt}	6\tXrgpwgnk}
ANDBIT	CR, RS, CR	And bit
ORBIT	CR, RS, CR	OR bit
ANDCRBIT	CR	Skippable and bit
CR		End of Skippable and bit
ORCRBIT	CR	Skippable or bit
PUSHOR		End of Skippable or bit
JMPBIT	state	Jump directly
JMPCBIT	state	Jump when meet the condition
CALBIT	Function name or function block name	Call function or function block
CALCBIT	Function name or function block name	Call function or function block when meet the condition
RETBIT		Return from interruption
RETCBIT		Return from interruption when meet the condition
LOADQBIT	Q，CR	Load data from the output of function block register
STOREQEBIT	Q，[mem]/Rm	Store data to the output of function block register
LOADMBIT	[mem],Rs1/CR	Load data from data RAM
STOREMBIT	Rm/CR,[mem]	Store data to data RAM
LOADIOBIT	[I/O],CR	Load data from I/O RAM
STOREIOBIT	Rm/CR, [I/O]	Store data to I/O RAM
LOADRMBIT	Rm，CR	Load data from register file
STORERMBIT	CR , Rm	Store data to register file
FB SR	Function block name.S	Function block of setting data firstly
FB RS	Function block name.R	Function block of resetting data firstly
FB CTU	Function block name.PV	Function block of add counter
FB TON	Function block name.PT	Function block of timer which delay after turning on

1 } *PLC* LH72<9<Ø}LNKO2NS@GDK}CN}

The PLC specific instruction format (see fig.3) has four kinds: bool instruction format, skippable bool instruction format, JUMP\CALL\LOAD\STORE instruction format and function block instruction format.

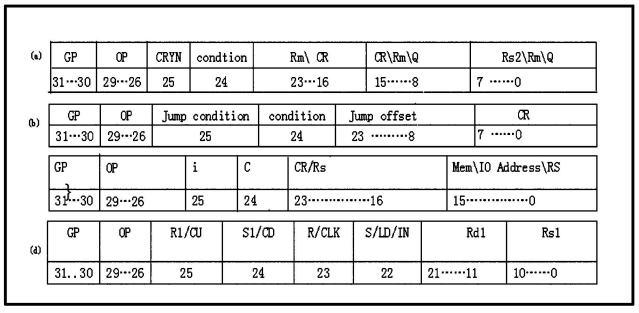

Figure 3. Instruction formats of the special PLC processor .(a)bool instruction format.(b)skippable bool instruction format.(c)jump and call instruction format.(d)special function block instruction format.

- Program counter design(see fig. 6)
  The program counter is used for fetching instructions to the instruction register. There are three kinds of instructions' address to be processed. First, counting the next instruction's address by adding four based on the present instruction's address. Second, when coming across the skippable boo : instruction, the instruction's address is changed according to the processing flow chart (see fig.7) of the skippable bool instruction. [2] Third, when processing the " JUMP " and " CALL " instructions, the next instruction's address is changed by the " JUMP " and " CALL " instructions' offset.

  The Flow chart of skippable bool instructions(see fig.7) describes the processing flow of skippable bool instructions. At the beginning, the bit processor judges whether the instruction is skippable bool instruction or not by group code; at the second step, the bit processor judges whether the operation of instruction is " AND(" operation, " OR(" operation or not through operation code; at the third step, first, if the operation is "AND(" operation, and the value of the top accumulation stack is zero, then the bit processor sets the value of jump condition of skippable bool instruction format to be one, and keeps the original value of the top accumulator stack to be zero, then the program counter jumps to the next instruciton according to the jump offset in the skippable boo : instruction set. Second, if the operation is "ANDN(" operation, and the value of the top accumulation stack is zero, then the bit processor sets the value of jump condition of skippable bool instruction format to be one, and sets the original value of the top accumulator stack to be one, then the program counter jumps to the next instruciton according to the jump offset in the skippable boo : instruction set. Third, if the operation is "OR" operation, and the value of the top accumulation stack is one, then the bit processor sets the value of jump condition of skippable bool instruction format to be one, and keeps the original value of the top accumulator stack to be one, then the program counter jumps to the next instruciton according to the jump offset in the skippable bool instruction set. Fourth, if the operation is "ORN (" operation, and the value of the top accumulation stack is one, then the bit processor sets the value of jump condition of skippable bool instruction format to be one, and keeps the original value of the top accumulator stack to be one, then the program counter jumps to the next instruciton according to the jump offset in the skippable bool instruction set. At the end, if all conditions described in the third step cann't be meet, then the bit processor ends executing the skippable boo : instruction.

Figure 7. Flow chart of skippable bool instructions

- Bit processor design (see fig.8)
  The bit processor[3] mainly executes the bool instructions and the skippalbe bool instructions.The bit processor adopts the accumulator stack to process the serial bool instructions. The operands of bit processor are mainly from accumulator stack, register file or data RAM. The result is directly stored in the data RAM, I/O RAM. condition register and accumulator. The accumulator stack's data are from register file, function block register and I\O RAM. Take the ladder program in the figure 2 for example. When executing the first step of program, the data RE is loaded from I\O RAM to the top of accumulator stack, then the " ANDCRBIT " instruction is executed by the bit processor. If the value of {m} is zero, the bit processor executes the " ANDBIT " instruction and keeps the value of the accumulator stack's data to be zero, then the program counter skips the instruction's address to the address of the instruction " CR " 's next instruction.

IV. EVALUATION

To explain the high performance of PLC dedicated processor, a ladder compiler- "ldmicro" is used, which can compile ladder program to binary code for PIC16 or AVR. To contrast the processors' performance of executing PLC program, I have compiled three ladder programs to binary code of Microchip PIC16F877and Atmel AVR ATmga16 40-PDIP, then Anti-compiled the binary code to assembler with software of "icprog" and "AVR Studio3.53". Then I have translated the three ladder programs to assembler of PLC dedicated processor manually.

The first ladder program with ton function block in the fig.2, all its instructions are in the PLC ASIP's instruction

set. All the instructions of PLC's IL prgram in the part b of fig.2, which is like the assmbler, almost can execute directly on the PLC ASIP. The part c in fig.2 is the translated assembler, using the instruction set of the PLC ASIP.

The second ladder program is traffic led program, which includes the bool instructions, function block instructions executed by the PLC ASIP and compare instructions executed by the general processor. The interruption has been sent to the general processor by the PLC ASIP. The PLC ASIP should wait the result of the compare instruction from the general processor. So the executing location of the PLC ASIP need not be changed, when the interruption has been send.

The third ladder program contains a arithmetic instruction in the general processor, which can execuate concurrently with function block instruction in the PLC ASIP.

Table 2. Contrast the number of compı ed instructions

Processor / Number	PIC16F877	ATmga16	PLC dedicated processor
Ton program	98	256	21
Traffic led program	618	944	193
Arithmetic program	147	281	61

From the table2, we can see the row of "Ton program", the PLC dedicated processor can reduce to about 21.4% and 8.2% based on the compiled instructions of PIC16F877 and ATmga16. The row of "Traffic led program", the PLC dedicated processor can reduce to about 31.2% and 20.4% instruction number based on the compiled instructions of PIC16F877 and ATmga16. The row of "Arithmetic program", the PLC dedicated processor can reduce to about 41.4% and 21.7% instruction number based on the compiled instructions of PIC16F877 and ATmga16.Because the traffic led program and the arithmetic program has the instruction of the general processor, the compiled instruction number will be increased, compared to the ton program. As we all know, fewer number of compiled instructions will lead to lesser executing time. According to the one advantage of the PLC dedicated processor with its own instruction set, one part of high performance can be embodied.

REFERENCES

[1] Gab SeonRho, Kyeonog-hoon Koo,Naehyuc Chang, Jaehyun Park, Yeong-gi Kim and Wook Hyun Kwon. "Implementation of a RISC microprocessor for programmable logic controllers", Elsevier Science B.V, Microprocessors and Microsystems Volume 19 .Number 10 ,December .1995

[2] Takashi Yamauchi, Minamitsuru, "PLC PROCESSOR AND PLC", United States Patent, 5,233,697, Aug. 1993.

[3] XU Mei-hua, RAN Feng, CHEN Zhang-jin, KANG Shu-feng and LI Run-guang, "IP Core Design of PLC Microprocessor with Boolean Module", High Density Microsystem Design and Packaging and Component Failure Analysis, 2005 Conference on Digital Object Identifier: 10.1109/HDP.2005.251460 Publication Year: 2005 , Page(s): 1 – 5

[4] Kyenonghoon Koo, Gab Seon Rho, Wook Hyun Kwon and Jaehyun Park, "Architecutual Design of a RISC Processor for Programmable Logic Controllers", Preprint submitted to Elsevier preprint ,15 january 1996.

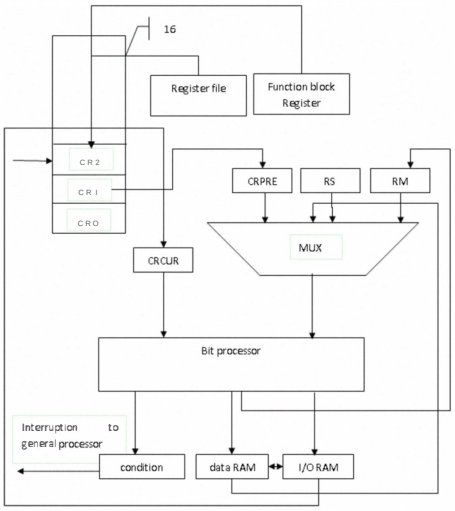

Figure - } Bit processor