Converting PLC instruction sequence into logic circuit: implementation and evaluation

Graduate Adviser: Shuichi Ichikawa 033701 Masanori Akinaka

1 Introduction

PLC (Programmable Logic Controller) is a kind of computer, which has been widely adopted for sequence control of industrial machinery. Though PLC is flexible and well-established, the performance of PLC does not always satisfy the requirements in large and highly responsive systems. Another problem of PLC is that a PLC program is easy to duplicate and to analyze. This often results in the leakage of valuable trade secrets and the rise of clone products.

Recently, it has been proposed to implement a PLC program with hard-wired logic using FPGA (Field Programmable Gate Array) [1]. Since FPGA is a reconfigurable logic LSI, the response time can be improved together with flexibility of PLC. It should be also noted that FPGA is more secure than PLC in protecting intellectual properties, because it is more difficult to analyze an FPGA design than to analyze a PLC program, and some recent FPGA devices provide design security features.

This study presents a converter, which translates PLC instruction sequence into logic description in VHDL. The evaluation results of the converter are also presented with sample PLC programs of actual products. The preliminary version of this study was presented in ISIE 2006 [2].

2 Translation of PLC program to logic circuit

The ladder diagram has been widely accepted to describe PLC programs. A ladder diagram consists of one or more rungs, each of which consists of a condition part and a process part (Fig.1). Either the condition part or the process part can be an output (a) or an instruction (b). Rungs are ordered, and interpreted in due order. When the end is uncounted, the ladder is executed all over again from the first rung. The time repeated once is called scan time. By making the scan time shorter, the system becomes more responsive.

To generate a logic circuit that literally simulates a whole ladder program, it is straightforward to design a sequential circuit, which activates one rung for each cycle in due order. This design is called Sequential Design (SD).

It is possible to execute two or more rungs in parallel, if dependences among rungs are properly maintained. Figure2 shows an example of data dependence, where the output of the upper rung is referred by the lower rung. In such cases, it is essential to execute the upper rung before the lower rung to derive the same result as in the original ladder diagram. By dispatching each rung to the earliest cycle possible (while keeping all dependence), the clock cycles required for each scan can be reduced. This design is called Levelized Design (LD).

It is not necessary to allocate one cycle to each level, but it is possible to implement the ladder program by a combinatorial logic circuit, by removing the internal memory element between the levels of LD. This design is called Flat Design (FD). Each scan is implemented by a single state in this design.

When external device IO instructions are included in the PLC program, inputs and outputs to external device have to be performed immediately with BFM (buffer memory). Thus, an additional state for inputs and outputs is necessary in FPGA.

In the abovementioned designs, the arithmetic units are generated as many as the arithmetical instructions in PLC program. In SD and LD, however, all arithmetic units are not used at simultaneously. The logic scales are hence reducible by sharing the arithmetic unit.

3 Implement and evaluation

The conversion techniques described in Section 2 were implemented, and evaluated with sample programs for Mitsubishi FX2N PLC. In this study, the target device was set to Altera StratixII FPGA (EP2S60F672C5ES with 48,352 ALUTs). Logic scales and clock frequencies were estimated by Altera Quartus II 6.0SP1, where the optimization options were set to default (balanced). The design that generates one arithmetic unit for each arithmetical instruction is shown as "dedicated" in Tables1 and 2. The design that limits the number of arithmetic units of each kind (adder/subtracter, multiplier, and divider) to 1, is shown as "shared". The scan time of FPGA is estimated by the estimated maximum operational frequency and the number of states of the circuit. The scan time of PLC was estimated on the assumption that all condition parts are executed.

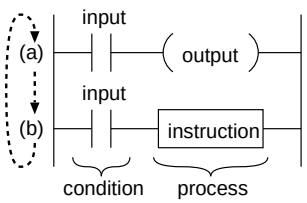

Fig. 1: Overview of ladder diagram

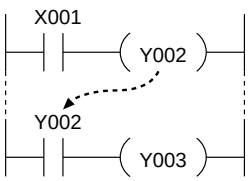

Fig. 2: An example of data dependence

Tab. 1: Evaluation results of sample ladder program A

Device	Design	Arithmetic unit	Num. of states	Max. freq. [MHz]	Logic scale [ALUTs]	Scan time [sec.]
PLC	—	—	—	—	—	1.61×10^{-3}
FPGA	SD	dedicated	74	8.21	5,848	9.01×10^{-6}
		shared	74	6.10	2,863	1.21×10^{-5}
	LD	dedicated	12	7.57	5,686	1.59×10^{-6}
		shared	17	6.88	2,716	2.47×10^{-6}
	FD	dedicated	1	4.67	4,625	2.14×10^{-7}

Tab. 2: Evaluation results of sample ladder program Y

Device	Design	Arithmetic unit	Num. of states	Max. freq. [MHz]	Logic scale [ALUTs]	Scan time [sec.]
PLC	—	—	—	—	—	4.64×10^{-2}
FPGA	SD	dedicated	415	16.08	3,554	2.58×10^{-5}
		shared	415	8.04	4,148	5.16×10^{-5}
	LD	dedicated	76	16.02	2,864	4.74×10^{-6}
		shared	76	7.81	3,445	9.73×10^{-6}
	FD	dedicated	63	13.64	2,643	4.62×10^{-6}

Table1 summarizes the evaluation results of a PLC program, which includes 165 instructions, including 6 add/subtract instructions, 12 multiply instructions, and 9 divide instructions. LD (dedicated) is 5.7 times faster than SD (dedicated), and FD is 42.1 times faster than SD (dedicated). This program is highly concurrent to control many outputs. Consequently, the number of state decreased to 16% by parallelization. When the arithmetic units were shared, the logic scale decreased to 49% in SD, and to 48% in LD.

Table2 lists the evaluation results of a sample PLC program, which is the control program of a product under development. This PLC program includes 1303 instructions, which include 5 add/subtract instructions, 11 multiply instructions, and 2 divide instructions. LD (dedicated) is 5.4 times faster than SD (dedicated). Although FD is 5.6 times faster than SD (dedicated), it is hardly different from LD (dedicated). Since many external device IO instructions are included in this sample, there are many additional states for inputs and outputs. In FD, the number of states is almost the same as LD's. When the arithmetic units were shared, the logic scale increased to 117% in SD, and to 120% in LD. This was caused by the multiplexer and the general-purpose arithmetic unit additionally generated for sharing. When the program includes a small number of arithmetic instructions, the logic scale can be smaller with "dedicated" design.

4 Conclusion

Among the designs proposed in this study, FD was the fastest. However, when many external device IO instructions are included, the merits of FD are restricted. Although logic scales are reduced by sharing the arithmetic unit, the result depends on the property of the target program.

The following items are left for future works: (1) the evaluation of the proposed techniques with actual products, and (2) the enhancements of our converter to support more functions.

References

[1] I. Miyazawa, T. Nagao, M. Fukagawa, Y. Itoh, T. Mizuya, and T. Sekiguchi: "Implementation of ladder diagram for programmable controller using FPGA," in *Proc. 7th IEEE Int'l Conf. Emerging Technologies and Factory Automation (ETFA '99)*, vol. 2, 1999, pp. 1381–1385.

[2] S. Ichikawa, M. Akinaka, R. Ikeda, and H. Yamamoto: "Converting PLC instruction sequence into logic circuit: A preliminary study," in *Proc. IEEE Int'l Symp. Industrial Electronics (ISIE '06)*, 2006, pp. 2930–2935.