**TELKOMNIKA**, Vol. 11, No. 12, December 2013, pp. 7655~7663
e-ISSN: 2087-278X

# Research on FPGA-based Programmable Logic Controllers' Technology

**Zhu Huabing*, Liang Benlei, Dong Bolin, Feng Xiao**
School of Mechanical and Automotive Engineering of Hefei University of Technology, China
*Corresponding to anthor, e-mail: hfuthbzhu@163.com

***Abstract***

*This paper introduces a scheme which implements the programmable logic controllers' (PLCs) function based on FPGA. This scheme follows the IEC61131-3 standard, selects Ladder Diagram (LD) to write the PLC programs and selects VHDL as target language. Based on VS2005 platform, this scheme implements the construction of Ladder Diagram, compilation, simulation and other functions. This paper focuses on researching the construction method of Ladder Diagram, converting Ladder Diagram into Boolean equation and generating VHDL program by Boolean equivalence. The construction method of Ladder Diagram based on parallel-series hierarchical nested list and the implementing method of Boolean equivalence based on double-layer lists are proposed. Finally the correctness of the scheme is verified through an example.*

***Keywords:*** *IEC61131-3, FPGA, programmable logic controllers, parallel-series hierarchical lists*

## 1. Introduction

Programmable logic controller (PLCs) has become an indispensable control unit in the industrial control field. But the performance of traditional PLC will be restricted by the length of ladder diagram and the operation speed of the microprocessor. It is difficult to adapt to the requirements of high-speed control in modern industry. Therefore, figuring out a way to realize high-speed control becomes more and more important [1].

Field programmable gate array (FPGA) has characteristics of supporting high-speed parallel execution and hardware configuration [2-3]. Realizing the function of PLC by it can greatly improve the execution speed of control logic and this is an important way to solve the current problems of PLC.

The core of implementing PLC based on FPGA is how to convert IEC61131-3 standard language into hardware description language. Ikeshita converted SFC to Verilog language [4]; Ichikawa implemented the research and conversion to the framework of instruction table translation into HDL [5]. But [4, 5] documents don't implement the conversion of the LD that is widely used, and also have indirect and hidden problems.

Combining John T's research on implementing the ladder diagram logic by FPGA [6] and Jae Ick Lee's optimizational method on the VHDL program [7], this paper presents the structure algorithm of LD based on hierarchical nested lists structure and conversion algorithm of Boolean equivalence based on double-layer lists. Finally the conversion from the LD to VHDL is realized.

## 2. Design of PLC based on FPGA

### 2.1. Hardware framework.

The hardware framework of FPGA-based PLC is shown in Figure 1. This framework includes FPGA chip, I/O unit, A/D module, reset circuit, power supply circuit, program memory chip EPCS4, JTAC download port etc. ISA interface is used for communication with host computer, I/O unit and A/D module are used for acquisition of digital and analog signals.

*Received June 30, 2013; Revised August 20, 2013; Accepted September 4, 2013*

### 2.1 Software Framework

VHDL consists of two modules: One is ISA communication decoding module, which packages address decoding and latches data into the interface package. The other is generation module of VHDL program that has equivalent function with the ladder diagram. The difficult and key technology is how to convert the ladder diagram into VHDL.

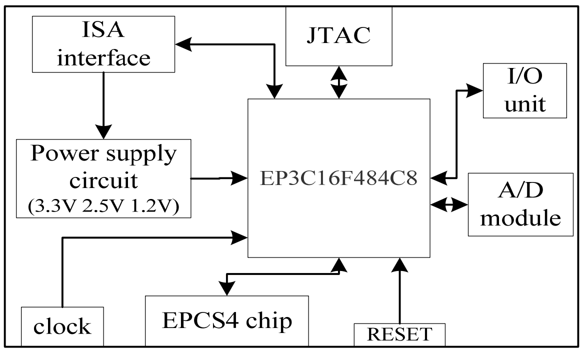


Figure 1. FPGA-based PLC Hardware Framework

### 2.3. Generation of Control Program

The process is as follows: The first step is to convert the LD into VHDL program. (1) Determine the basic elements of constructing the LD; (2) Implement the construction of LD and logical preservation; (3) Convert the LD into Boolean equivalence; (4) Distribute trigger signal and generate VHDL program. The second step is to compost and implement VHDL program in FPGA devices. (1) The VHDL program is taken as the input of design for functional simulation to complete the preliminary function detection; (2) Build logic net lists and take post simulation to check whether the results are consistent with the design; (3) Use the placement and routing function to configure the logic net lists and the bottom unit to the FPGA chip, and to correctly connect the components. (4) Feedback the delay information of placement and routing to the lists, and test whether there are timing violations by the timing simulation. (5) Generate data files, then the files are downloaded and tested in the FPGA.

## 3. Construction of Ladder Diagram

A lot of researches, on the construction of the LD, have been done at home and abroad. Hasegawa M. presented that the ladder diagram could be described by Petri net [8-9]. Tan Jinjie proposed to store the ladder diagram with double linked lists [10]. But all of them can't express the topological structure of the LD briefly and effectively for a lot of redundant information to deal with. Lv Junbai proposed a method, based on crossed lists, to convert a LD into sentence table [11]. Isolated points were easy to cause by this method, which was based on directed graph and classified according to the icon and connector. Ge Fen put converting the LD into instruction list based on the AOVand binary tree [12]. Huang Liuwen directly converted LD into structure text (ST) based on the AOV [13]. However, this application was prone to produce miscarriages for virtual vertices and multiple parallel. This paper introduces parallel-series hierarchical lists, which can realize the construction of LD easily.

### 3.1. Data Structure of the Ladder Diagram

The basic structure of constructing ladder diagram, shown in Figure 2, contains CLDProject class, CLDStep class and other three classes (CLDSeries class, CLDParallel class, CLDUsual class), which were based on CObject class.

CLDUsual class implements the definition of usual properties, such as elements' display position, remark, etc. Shown in Figure 3(a). The subclass of the basic elements were generated from CLDUsual class, which realized operation on basic components [14]. The most important parts in LD structure are the logical relationship and how to realize parallel-series, shown in Figure 3 (b), (c).

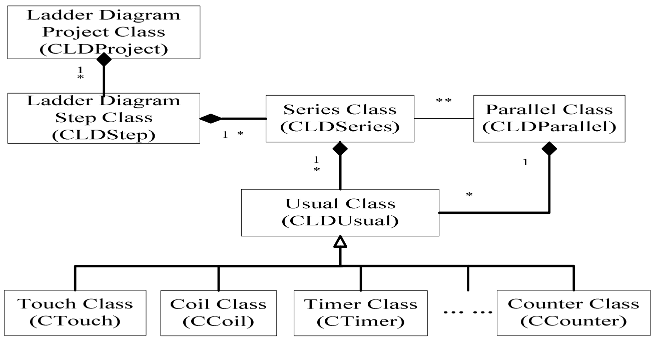


Figure 2. The Basic Structure of Constructs LD

(a)

```
class CLDUsual: public Cobject
//Element universal attribute classes
{ int row;                    //Element row
  int line;                   //Element line
  CString comment;            //Element remark
  CString name;               //Element name
  static BOOL InsertCLDStep();
                       //Insert the step action
  static BOOL
  InsertLDUsual(CLDUsual*new, int type);
         //Insert element action
 void OnLButtonDown(int type, Cpoint
         point);   //Mouse action response
  static BOOL InsertLDBasic (int old, void
*any, CLDUsual*new, int type);
        //Concrete insert operation
......
};
```

(b)

```
class CLDSeries: public Cobject  //Series relationship class
{......
     struct{
           union {......
                      CLDParallel*  PLCParallel;  //Parallel elements
                      CLDUsual*   PLCUsual;       //Basic element
                  }basic;
           int nature                  //Number of Series elements
           }size[MAX_NUM];
};
```

(c)

```
class CLDParallel: public Cobject     //Parallel relationship class
{......
      struct{
             union {......
                        CLDSeries*   PLCSeries;      //Series elements
                        CLDUsual*  PLCUsual;         //Basic element
                    }basic;
             int nature;                 //Number of parallel elements
           }size[MAX_NUM];
};
```

Figure 3. Data Structure of LD Core Class

### 3.2. Constructive Algorithms of the Ladder Diagram

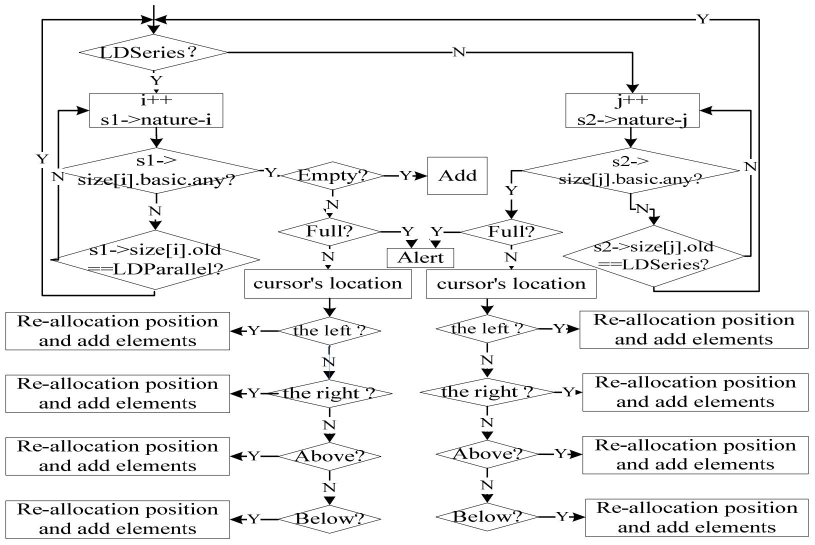


Figure 4. Flow Chart of the Element Inserts Action

The insert action can be implemented through InsertLDUsual() function. (1) The program finds out the series pointer and its cursor's position by recursive algorithm. (2)Adding new elements and doing after-processing. InsertLDUsual() function calls InsertLDBasic()

function to achieve the specific implementation of each step, such as refresh location assignment process etc. The flowchart of InsertLDBasic() function is shown in Figure 4. The parallel-series hierarchical structure, ensures the logical relationship of basic elements, realize the control of internal logic by ladder diagram.

## 4. Generation of Boolean Equivalence

### 4.1. Data Structure of Boolean Equivalence

Converting the LD into Boolean equivalence is a key step for VHDL is compatible with Boolean equivalence. The equivalent form does not support the timer, so its functions need be extended, e.g. Illustrating by example is shown in Figure 5(a),(b), and respectively correspond to Y1<=!X1&X2&(X3&X4); C2(K10)<=X2, Y2<=C2.

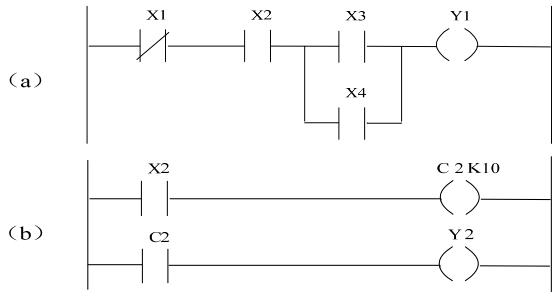


Figure 5. Example of LD-equivalence

The data structure of Boolean equivalence after expanding its function is shown in Figure 6.

(a)

```
typedef struct BooleanExTag{ // Head of Boolean equivalence
   int xuhao;                   //Step serial number
   Leaf*singleleaf;             //Single element unique nature
   Single* arclink;             //The head pointer of step
   BooleanEx* nextRung;         //The head of next expression
}BooleanEx;
```

(b)

```
typedef struct SingleTag{  //The structure of single element
  #define AND    1        //Type number
  #define OR     2
  #define Contacts  3
  ……
  int type;               //Number of element Type
  int relation;           //Number of series and parallel
  Leaf*  singleleaf;  //The unique properties of single element
  Single* singlelist;  //Pointing to  the list of order elements
  Single* singlelink;  //Pointing to the list of same layer element
}Single;
```

(c)

```
typedef struct LeafTag{
                    //A community of single-element
  union{
    Contacts con;
    Coil    coi;
    Timer   tim;
    ……}l;
}Leaf;
```

(d)

```
typedef struct ContactTag
               //Generally enter the structure  of leaf
 { char name[MAX];  //Enter the name of  leaves
 Bool InOut;             //Normally open / closed
 }Contact;
typedef struct TimerTag      //Timer structure
 { char name[MAX];          //Timer name
 int  delay;                  //Timer delay time
 }Timer;
```

Figure 6. Data Structure of Boolean Equivalence

### 4.2. Algorithm of Generating Boolean Equivalence

The core of the algorithm is to obtain a Boolean equivalence which corresponds to a single step; Figure 7 is a flowchart of single step.

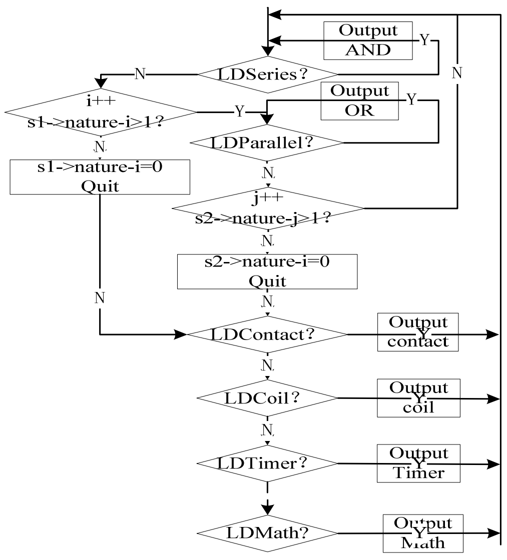


Figure 7. The Basic Flowchart of Boolean Equivalence Generated

The process of converting LD into Boolean equivalence is as follows: (1) Open nodes for output of the first step and rank them. (2) Record the number of series pointer which links to head node. If the number is more than 2, a AND node will be distributed. Or, the next step will be carried out. (3) Scan every element pointed by series pointers circularly from the step head, as well the internal of element. If the scan series are encountered in parallel attribute, an OR node will be assigned and linked. Or, a AND node is assigned and linked. Repeating step (3) until all the basic elements of the step has linked. (4) The latter step links to the former step and does the same processing.

A set of Boolean equivalent form is generated in the conversion process, and its structure is a group of double linked lists. The head node is output of each step, and each head node links to the step inside logical in order to map the step's order and internal parallel-series logic relationship into linked lists that has Boolean equivalence structure.

## 5. Generation of VHDL program

### 5.1. Running Mode of the Ladder Diagram

In order to obtain PLC Running mode on the FPGA, not only did LD's logic need be achieved by VHDL program, but also the control method of PLC will be converted to VHDL program in this paper. In the same process, characteristic of VHDL statement is concurrency and PLC is sequential, shown in Figure 8.

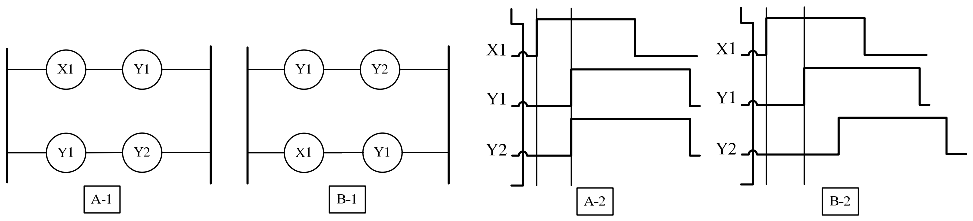


Figure 8. LD Instance and the Timing Differences between the LD

From the graph, we observe that these VHDL programs corresponding to the two groups' LD are same. The program is shown in Figure 9(a). Therefore, the running mode of LD need be considered.

```
Entity p1 is port(  X1:    in std_logic;
                    Y1:  inout std_logic;
                    Y2:  out std_logic;);
end p1;
architecture BooEx of p1 is
begin
  Y1:=X1;
  Y2:=Y1;
end BooEx;
```

(a)

```
architecture BooEx of p1 is
begin
  process(CLK) begin
    if(clk'event and clk='1') then
          Y1:=X1;
      Y2:=Y1;
        end if;
  end process;
```

(b)

Figure 9. VHDL Programs of before and after using the Clock Event

VHDL language's clock event has been used to make Boolean equivalence be arranged according to the order of ladder diagram. The process is show in Figure 9(b). Because the Qn+1=D characteristic of the D latch, self-lock of the LD can be converted. The rising and falling edges elements of LD can be equivalent by using D trigger.

### 5.2. Generation and Optimization of VHDL

Although the VHDL program can be generated by adding head and clock to Boolean equivalence list, the more efficient VHDL program should be obtained by further optimization. Using CSG theory as reference, the directed and undirected graphs can be used for further optimization of the VHDL program, as the instance shown in Figure 10(a).

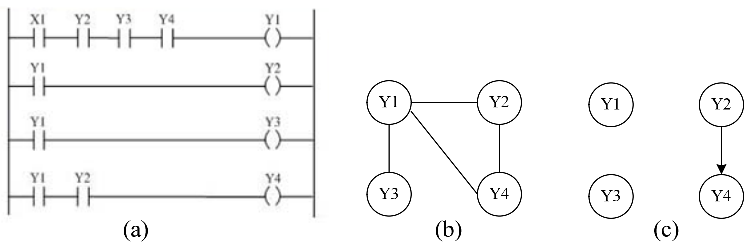


Figure 10. LD Instance and the Corresponding Directed Graph and Undirected Graph

The first step, the directed and undirected graphs of output element can be obtained by traversing the Boolean equivalence, correspond to Figure 10(b) and (c), and their adjacency matrixes correspond to the A1, A2.

$$A1=\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&0&0\\1&1&0&0\end{bmatrix} A2=\begin{bmatrix}0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{bmatrix} A3=\begin{bmatrix}0&1&1\\1&0&0\\1&0&0\end{bmatrix}$$

The second step, traverse undirected graph matrix by the row. These elements that were link to all of other elements are extracted (full-connected elements), and new adjacency matrix of the undirected graph are established by these elements. Y1 element is isolated in the instance and adjacency matrixes correspond to the A3.

The third step, the relation between directed graphs can be obtained by traversing adjacency matrix of the directed graph. The Y2 and Y4 are compressed together to simplify undirected graph matrix and isolate independent elements that are Y24 (Y2 and Y4) and Y3.

Full-connected elements Y1 and independent elements Y3 are first excited, and then the other independent elements are excited according to the order. As shown in Table 1.

Table 1. The Excite Order of the LD

| Excite order | Nodes element | |
|---|---|---|
| 1 | Y3 | Y1 |
| 2 | | Y2 |
| 3 | | Y4 |

Data structure after traversing can be obtained by expanding the structure tree. The format is show in Figure 11.

```
class SimpleBoolean
{CString Name;        // Output element name
   int Father;          // Name of the parent element
   int Child1;          // Left child element names
   int Left;            // Left element name
   int Right;           // Right element name
   int Order;           // Signal distribution order;
   CString exBool;// corresponding Boolean equivalence
   SimpleBoolean*pNext;// Points to the next structure
  };
```

Figure 11. The Format of Data Structure after Traversing

The order of Boolean equivalence has been rearranged through the graphical optimization process, and a process will be assigned after interpreting every Boolean equivalence logic. If just the general logic, it will be directly allocated to the process; if the timer/counter output is contained, the conversion process in the VHDL program will be interpreted as a timer/counter, and they will be distributed the corresponding parameter; The trigger signal is orderly distributed according to the Order sequence, the processes with same Order are distributed the same trigger signal. The complete VHDL program can be obtained by adding the input and output illustrations and the communication package to the entity and the structure.

## 5.3. A Verified Instance

The tool magazine control platform of CNC system is shown in Figure 12.

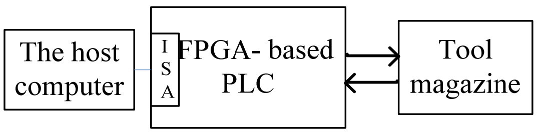
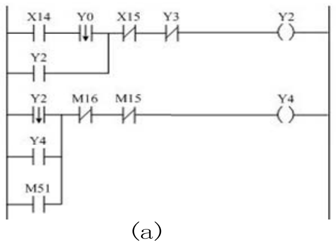


Figure 12. Experimental Platform Framework



```
Method:process (clock,reset)
begin
   if (reset= '1')then
      b_process<=”0000”;
   else  if  (clock'event  and  clock=”1”)
then
     if (b_process=”1111”) then
       b_process<=”0000”;
     else
       b_process<=b_process+”0001”;
……                  (b)
```

Figure 13. The LD Fragment of Tool Magazine and Compilation Process

After DAOKU.vhd acquired the ISA interface package, the instructions data of the host computer sends to the bus have been latched by the address decoding, which can excite the control logic of tool magazine and drive the tool magazine move. The state of the tool magazine can be obtained by scanning feedback signal. Fragment of the tool magazine LD program is shown in Figure 13(a).

```
u1: dff
port map(clk=>clk,cd=>cd,sd=>sd,d=>Y0,q=>Y0_1);
No_4:process (b_process)
begin
      if b_process = “0100” then
Y2<=((Y14 and Y0 and not Y0_1) or Y2) and not X15 and not Y3;
u2:dff
port map(clk=>clk,cd=>cd,sd=>sd,d=>Y2,q=>Y2_1);
No_5:process (b_process)
begin
      if b_process = “0101” then
Y4<=((Y2 and not Y2_1) or Y4 or M51) and not M16 and not M15;
```

Figure 14. VHDL Program of Corresponding to the Ladder Diagram Segment

The complete VHDL procedures can be got after the tool magazine LD was compiled, and the state machine was used to control the excitation sequence. The process is show in Figure 13(b).

The VHDL program corresponding to the ladder diagram segment is shown in Figure 14. dff is a D trigger, which will be call on the rising or falling edges of ladder diagram.

Use Modelsim_Altera to simulate the VHDL program.The simulation results of unload 2 tool and load the 3 tool is shown in Fig.15.After address decoded the Txx instruction which was sent by the host computer, the data [15-0] are latched and data [3-0] signals are passed to the X4-X1 separately to trigger the logic.

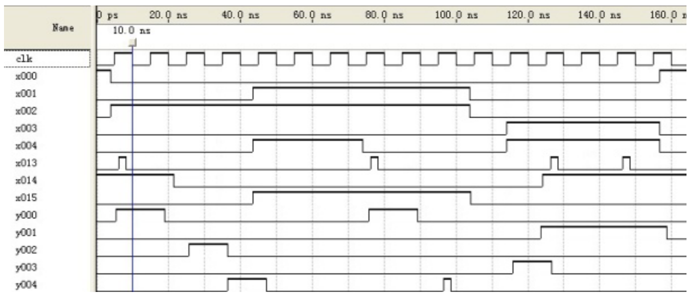


Figure 15. Tool Control Procedures VHDL Timing Diagram

(1) In the first 4 clock, FPGA receives the T2 instruction from the host computer, pulls M2 and trigger Y0. Then the tool magazine rotates and stops untill to No. 2 cutter location; Trigger Y2 and moves tools until it pressures the forward limit switch, Y4 will be triggered to open the air pump in order to unload 2 tool. (2) In the next 3 clock, FPGA receives the T11 instruction and pulls M50 to output Y4 in order to close the air pump. (3) In the next 4 clock, FPGA receives the T3 instruction, and pulls M3 to trigger Y0. Then the tool magazine rotates and stops untill to No. 3 cutter location. Low Y4 to control air pump in order to clamp the 3 tool. (4) In the last 5 clock, FPGA receives the T12 instruction and pulls M51 to trigger Y3 to control the tool back. When the tool pressures the rear limit switch, Y1 is triggered to control the tool reversal untill to the original location.

Right timing sequence action can be obtained from the above analysis. Programs, which need 15 instruction cycles in the traditional PLC, can be executed within a few clock cycles in FPGA and achieve the same function as LD. The tool control can be realized after downloading the program to FPGA.

## 6. Conclusion

PLC, based on FPGA, can greatly improve the speed of logic control. In this paper, the conversion from IEC61131-3 standard language to hardware description language was researched and software converting the ladder diagram into VHDL program was developed. The software implemented the generation of ladder diagram structure, Boolean equivalence and VHDL program. The control of tool magazine was realized by PLC based on FPGA. Research shows that PLC, based on FPGA, plays an important role in promoting the development of high-speed PLC control.

## References

[1] B Magnussen. *A parallel control computer structure for complex high speed applications*. First IEEE International Conference on Engineering of complex computer Systems. 1995: 385-388.

[2] Guangyou Yang, Zhijian Ye, Yurong Pan, Zhiyan Ma. The Implementation of S-curve Acceleration and Deceleration Using FPGA. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(1): 279-286.

[3] Sutikno T, Jidin AZ, Jidin A, Idris NRN. Simplified VHDL Coding of Modified Non-Restoring Square Root Calculator. *International Journal of Reconfigurable and Embedded System*. 2011: 1(1): 37-42.

[4] Mako Ikeshita, Yuji Takeda. *An Application of FPGA to high speed Programmable Controller-Development of the conversion program from SFC to Verilog*. Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation. 1999: 1386-1390.

[5] Shuichi Ichikawa, Ryou Ikeda. *A Study on a Hardware Translation Tool for PLC Programs*. Proceedings of the 2006 IEEE International Symposium on Industrial Electronics. 2006: 2930-2935.

[6] John T Welch, Joan Carletta. *A Direct mapping FPGA architecture for Industrial Process Control Applications*. Proceedings of the 2000 IEEE International Conference on Computer Design. 2000: 595-598.

[7] Jae Ick Lee, Sung Wook Chun, Soon Ju Kang. Virtual prototyping of PLC-based embedded system using object model of target and behavior model by converting RLL-to-statechart directly. *Journal of Systems Architecture*. 2002: 17-35

[8] Karl-Heinz John, Michael Tiegelkamp. IEC61131-3 Programming industrial automation system. Germany: Springer-Verlag Company. 2001.

[9] Hasegawa M, Takata M, Temmyo T, et al. *Modelling of Exception Handling in Manufacturing Cell Control and Its Application to PLC Programming*. Proceedings of the IEEE International Conference on Robotics and Automation. 1990: 514-519.

[10] Tan Jinjie, Chen Lianghong. Idea of Description on Ladder Diagram's Data Structure for Embedded PLC. *Computer engineering*. 2004; 30(10): 85-87.

[11] Lv Junbai, Shi Minfang. Visual Editing of PLC Ladder Diagram and the Automatic Generation of Sentence List. *Process Automation Instrumentation*. 2005; 26(3): 28-30.

[12] Ge Fen, Wu Ning. Transformation Algorithm Between Ladder Diagram and Instruction List Based on AOV Diagraph and Binary Tree. *Journal of Nanjing University of Aeronautics & Astronautics*. 2006; 38(6): 754-758.

[13] Huang Liuwen, Liu Wei, Liu Zhanqing. Algorithm and Realization of Transformation from PLC Ladder Diagram to Structured Text. *Journal of Chinese Computer Systems*. 2011; 32(2): 339-341.

[14] Wu Ning, Ge Fen. Design and realization of general development platform for PLC. *Chinese Journal of Scientific Instrument*. 2007; 28(8):1486-1491.