Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A New Translation Algorithm from Ladder Diagrams to Instruction Lists

Yan Yi, Zhang Hang Ping

* *Institute of Intelligence & Software, Hangzhou Dianzi University, Hangzhou, China, (e-mail:yybjyyj@163.com)*

Abstract: A new algorithm for translating ladder diagrams into instruction lists is presented in this paper. To perform this task a new class of digraphs called Fractal Series Parallel (FSP) digraphs is proposed for the first time, which can represent the Boolean functions of a ladder diagram more intuitively and concisely than binary trees. Our analysis is based on the fact that a General Series Parallel (GSP) digraph can be transformed into a FSP digraph that is equivalent to the GSP one in Boolean function.

1. INTRODUCTION

Ladder diagrams (IDs) and instruction lists (ILs) are major programming languages commonly used by Programmable Logic Controllers (PLCs). (Karl-Heinz, 2001) A ladder diagram consists of chart symbols, and expresses the control logic with serial and parallel connections between chart symbols. An instruction list is a set of instructions composed of operation codes and operands, and is similar to assembly.

The interest of the translation algorithm we study is that a ladder diagram cannot be executed directly by PLC, and a feasible way is to translate it into an IL first and then translate the IL to the native code. (Kim et al., 1999)

A translation method of row-by-row is mentioned (Chmiel et al., 2002; Jong-il Kim et al., 1992), which goes horizontally from left to right and begins analysis of next row when vertical connection is reached. In addition, another solving approach, which first translated a LD into a binary tree, and then obtained the IL by traversing the binary tree, is presented (Ge Feng et al., 2006; Kim et al., 1999).

Our main result is a new translation algorithm from a LD to an IL based on Fractal Series Parallel (FSP) digraphs. FSP digraphs provide a concise and intuitive form to describe the structure of ladder diagrams and are beneficial to the logic verification and the code optimization at the level of the IL generation.

The remainder of this paper is divided into four sections. The first one provides the definitions and elementary facts needed to understand the translation procedure. In the second, the definitions of FSP digraphs are outlined and shown correct. The third section presents the detailed steps to transform a LD to an IL using FSP digraphs. The last section presents our conclusion and future work to be conducted.

2. BASIC DEFINITIONS AND RELATIONS

2.1 Graph theoretical definitions

Most of the graph theoretical terms used are standard (Diestel Reinhard, 2003). Thus, the most commonly used terms and those that may produce confusion are not redefined here again.

A graph $G = <V, E>$ consists of a finite set of vertices V denoted by $V(G)$ and a finite set of edges E denoted by $E(G)$. Edges are pairs of distinct vertices, if the edges of a graph are unordered pairs the graph is undirected and if they are ordered the graph is directed. We will abbreviate directed graph as digraph. For the empty graph (ϕ, ϕ) we simply write ϕ.

In-degree of a vertex is the number of edges enters itself, denoted by $deg^+(v)$. Similarly, out-degree of a vertex is the number of edges leaves itself, denoted by $deg^-(v)$. A vertex v of a digraph G is a source denoted by $v_s(G)$ if its in-degree is zero, and is a sink denoted by $v_t(G)$ if its out-degree is zero. A vertex is a split vertex if its in-degree is one or zero and out-degree is greater than one. In contrast, a vertex is a join vertex if its in-degree is greater than one and out-degree is one or zero.

A path is a non-empty graph $P = (V, E)$ with the form $V = \{x_0, x_1, \ldots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$, where the vertex x_i is distinct for all $0 \le i \le k$.

We set $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$ and $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$. If $G_1 \cap G_2 = \phi$, then G_1 and G_2 are disjoint. If $V_1 \subseteq V$ and $E_1 \subseteq E$, then G_1 is a subgraph of G, written as $G_1 \subseteq G$. The function $Sub(G, v_s, v_t)$ is defined to get the maximal subgraph with the source v_s and the sink v_t from G.

If $deg^{-}(v_s(G)) = n$, then G can be represented by $B_1 \cup B_2 \cup ... \cup B_n$ where the Bi for all $1 \le i \le n$ is a subgraphs of G called branch components, and are obtained by (i) getting all different paths $P_1, P_2, \ldots, P_n$ by traversing from the ith incident edge of the source to the sink, then (ii) applying union operation on these paths, namely $P_1 \cup P_2 \cup \ldots \cup P_n$.

As shown in figure 1, the out-degree of G is three, so the digraph G can be represented by $B_1 \cup B_2 \cup B_3$ where $V(B_1) = \{1, 2, 6\}$ and $E(B_1) = \{(1, 2), (2, 6)\}$, $V(B_2) = \{1, 3, 5, 6\}$ and $E(B_2) = \{(1, 3), (3, 5), (5, 6)\}$, $V(B_3) = \{1, 4, 5, 6\}$ and $E(B_3) = \{(1, 4), (4, 5), (5, 6)\}$.

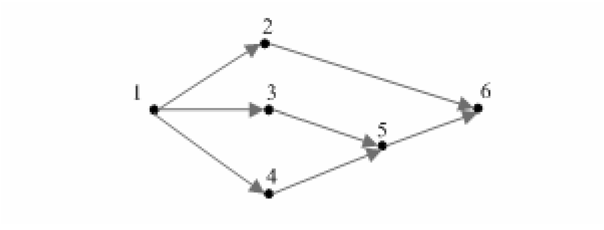


Fig. 1. A digraph G.

2.2 General Series Parallel digraphs

General Series Parallel digraphs have been extensively studied because of its relationship with the networks constructed by connections in series or in parallel of electrical components (resistors, capacitors, etc.).

A ladder diagram can be represented by a GSP digraph in a natural way. Each rung in the ladder diagram is represented as a single GSP digraph, and the ladder logic symbols constitute the vertices of the graph while the connections between symbols are implemented as the edges of the graph (Ngalamou et al., 2004). Figure 2 shows a ladder diagram and its equivalent GSP digraph representation.

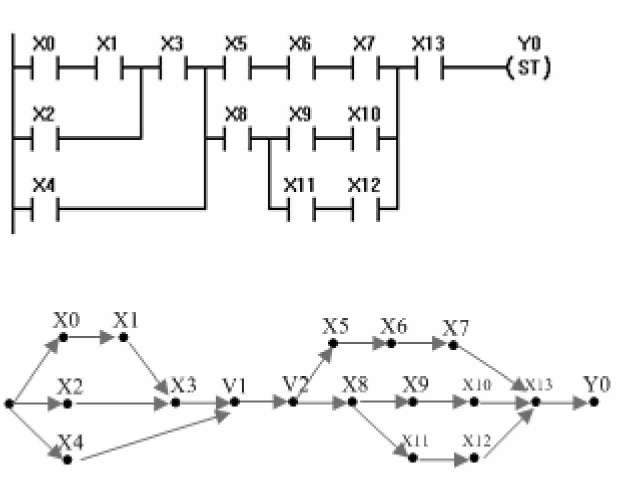


Fig. 2. Ladder Diagram and its equivalent GSP digraph representation .Please note that S, V1, and V2 are the virtual vertices with logic value TRUE.

A GSP digraph (Jacobo et al., 1979; Shih-Yih Wang et al., 1992) is defined recursively as follows:

Definition 1: General Series Parallel digraphs

1) A digraph G is a GSP digraph if it consists of two vertices v_1, v_2 joined by a single edge, and is called edge GSP digraph. The Boolean function of G is written as $F(G) = F(v_1) \cap F(v_2)$.

2) Let G_1 and G_2 be GSP digraphs.

   (i) If $v_t(G_1) = v_s(G_2)$, then a digraph G is a GSP digraph, which is obtained from G_1 and G_2 by identifying the sink of G_1 with the source of G_2. Such a connection is called a series connection. The Boolean function of G is written as $F(G) = F(G_1) \cap F(G_2)$.

   (ii) If $v_s(G_1) = v_s(G_2)$ and $v_t(G_1) = v_t(G_2)$, a digraph G is a GSP digraph, which is obtained from G_1 and G_2 by identifying the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2. Such a connection is called a parallel connection, and the Boolean function of G is written as $F(G) = F(G_1) \cup F(G_2)$. The source of G is called a corresponding split vertex of the sink; similarly, the sink is called a corresponding join vertex of the source.

3. THE THEORETICAL DEFINITIONS OF FSP DIGRAPHS

Although GSP digraphs can represent ladder diagrams in a natural way, it is difficult to obtain Boolean functions from them. Given a GSP digraph G composed of other GSP digraphs, if $deg^{-}(v_s(G)) = n$, the Boolean function of G can be written as

$$F(G) = F(v_s(G)) \cap F(v_t(G)) \cap \bigcup_{k=1}^{n} F(\mathrm{B_k}^*) \qquad (1)$$

where B_k^* is obtained from branch component B_k by eliminating its source and sink. There are two problems that make us obtain the Boolean function of B_k^* difficult. First, we cannot guarantee that B_1^*, $B_2^*\ldots$ and B_n^* are all disjoint, namely may be $B_1^* \cap B_2^* \cap \cdots \cap B_n^* \neq \phi$. Second, B_1^*, B_2^*, …, and B_n^* may not be all the class of GSP digraphs. Consequently, it is impossible to divide a GSP digraph into two or more sub GSP digraphs that are simple enough to be transformed into an IL and then combine the results of these sub digraphs to obtain a complete solution to the original one.

As shown in figure 1, the Boolean function of G can be written as $F(1) \cap F(6) \cap (F(B_1^*) \cup F(B_2^*) \cup F(B_3^*))$ where $V(B_1^*) = \{2\}$, $V(B_2^*) = \{3, 5\}$ and $E(B_2^*) = \{(3, 5)\}$, $V(B_3^*) = \{4, 5\}$ and $E(B_3) = \{(4, 5)\}$, and the graphs B_2^* and B_3^* are not disjoin.

To overcome this problem, FSP digraphs are proposed. Before we give the definitions of FSP digraphs two functions about split and join vertices are outlined as follows:

The *split vertex* function : $S_G : V \rightarrow \{0, 1, 2, \ldots, n - 1\}$. $S_G(v) = \phi$ if v is not a join vertex in G, otherwise the corresponding split vertex of v.

The *join vertex* function : $J_G : V \rightarrow \{0, 1, 2, \ldots, n - 1\}$. $J_G(v) = \phi$ if v is not a split vertex in G, otherwise the corresponding join vertex of v.

A FSP digraph is defined recursively as follows:

Definition 2: Fractal Series Parallel digraphs

1) A digraph G of a single vertex is a FSP diagraph.
2) A digraph G consisting of two vertices joined by a single edge is a FSP digraph.
3) Let G_1 and G_2 be FSP digraphs. If $v_t(G_1) = v_s(G_2)$, a digraph G obtained from G_1 and G_2 by identifying vertex $v_t(G_1)$ with vertex $v_s(G_2)$ is a FSP digraph.
4) Let G_1 and G_2 be FSP digraphs; let v_{s1} and v_{t1} are distinct vertices of G1; let v_{s2} and v_{t2} are distinct vertices of G2. If $v_{s1} = v_{s2}$ and $v_{t1} = v_{t2}$, and one of the following conditions (a), (b), (c), (d) is satisfied, then a digraph G obtained from G_1 and G_2 is a FSP graph by identifying v_{s1} with v_{s2} and v_{t1} with v_{t2}.

(a)
$$S_{G_1}(v_{t1}) = v_{s1} \;\; J_{G_1}(v_{s1}) = v_{t1}$$
$$S_{G_2}(v_{t2}) = v_{s2} \;\; J_{G_2}(v_{s2}) = v_{t2}$$

(b)
$$S_{G_1}(v_{t1}) = v_{s1} \;\; J_{G_1}(v_{s1}) = v_{t1}$$
$$S_{G_2}(v_{t2}) = \phi \;\; J_{G_2}(v_{s2}) = \phi$$

(c)
$$S_{G_1}(v_{t1}) = \phi \;\; J_{G_1}(v_{s1}) = \phi$$
$$S_{G_2}(v_{t2}) = v_{s2} \;\; J_{G_2}(v_{s2}) = v_{t2}$$

(d)
$$S_{G_1}(v_{t1}) = \phi \;\; J_{G_1}(v_{s1}) = \phi$$
$$S_{G_2}(v_{t2}) = \phi \;\; J_{G_2}(v_{s2}) = \phi$$

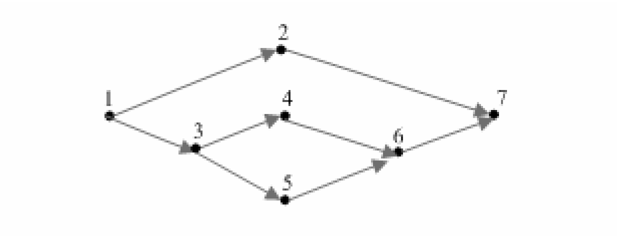


Fig. 3. A FSP digraph G.

Our transformation algorithm from a FSP digraph to an IL is based on the following facts:

Lemma 1: There is one-to-one relationship between a split vertex and its corresponding join vertex in a FSP digraph, namely a split vertex has only one corresponding join vertex and a join vertex has only one split vertex too.

Proof: From the def. 2, we can see that parallel composition only happens when (i) the source and sink all are not split and join vertices, (ii) or source and sink are corresponding split and join vertices of each other. This proves our lemma is correct.

Lemma 2: The out-degree of a split vertex is as many as the in-degree of its corresponding join vertex.

Proof: The increase of out-degree and in-degree is arisen by parallel composition operation. Due to the one-to-one relationship of split and join vertices, the out-degree of a split vertex and the in-degree of its corresponding join vertex increase together.

Lemma 3: Given a FSP digraph G composed of some other FSP digraphs in parallel, if the out-degree of the source is n, then B_1^*, B_2^*, …, and B_n^* are disjoint with each other.

Proof: If $V(B_i^*) \cap V(B_j^*) \neq \phi$, then there exist a corresponding join vertex of the source in the set of $V(B_i^*) \cap V(B_j^*)$. Since the sink is also the corresponding join vertex of the source, this proposition violates the lemma 1. Therefore B_1^*, B_2^*, …, and B_n^* are disjoint with each other.

Lemma 4: Given a FSP digraph G composed of some other FSP digraphs in parallel, if the out-degree of the source is n, then B_1*, B_2*, …, and B_n* are all the class of FSP digraphs (See equation 1).

Proof: Since B_1*, B_2*, …, and B_n* are disjoint with each other, then G is composed of B_1, B_2, …, and B_n in parallel, and B_1, B_2, …, and B_n are all the class of FSP digraphs. B_i^* must connect the source and sink in series, otherwise it will violate the lemma 1. Therefore B_i is composed of the source, B_i^*, and the sink in series. According to the def. 2, we can sure B_i^* is a FSP digraph.

Figure 3 shows a FSP digraph G, and B_1* = <（2）, ϕ >, and B_2* = <{3, 4, 5, 6}, {(3, 4), (4, 6), (3, 5), (5, 6)> are all the FSP digraphs.

4. THE ALGORITHM FOR TRANSLATING A LADDER DIAGRAM TO AN INSTRUCTION LIST

Now we have finally collected enough facts to be able to outline our procedure to translate a LD to an IL.

Algorithm 1: <Translation procedure for a LD to an IL>

Input: a ladder diagram

Output: the Boolean function of the ladder diagram

Step1: Represent the ladder diagram with a GSP digraph G.

Step2: Transform G into a FSP diagram G_T

Step3: Transform G_T into Boolean function

For describing the algorithm concisely and conveniently, we use Boolean function as the result of the translation process. The IL will be obtained by replacing '$\cup$' and '$\cap$' with 'OR' and 'AND' respectively in the process of translation.

4.1 The algorithm for transforming GSP to FSP digraphs

The Boolean function of a FSP digraph can be depicted concisely using equation 1, but FSP digraphs are unable to represent all ladder diagrams directly.

We will solve the problem with two steps: first, we partition a GSP digraph into series digraphs; this step is equivalent to partition the GSP digraph into edge GSP digraphs and connect them in series. Second, a FSP digraph equivalent to the GSP one in Boolean function will be constructed by connecting these series digraphs in parallel. A topology transformation method is used when the parallel condition of FSP digraphs is not satisfied.

Algorithm 2: <Transformation for a GSP to the FSP>

Input: a GSP digraph G

Output: a FSP digraph G_T and the list L_P composed of pairs of corresponding split and join vertices in G_T

Step1: *Decompose G into series digraphs.*

(i) A series digraph G_i will be gotten by traversing G with the depth-first traversal from a source to a sink, and put the series digraph into the list L_S.

(ii) Remove the edges of series digraph G_i from G.

(iii) Remove the nodes whose degree is zero from G.

(iiii) Repeat the operations (i), (ii), and (iii) until G become empty.

Step2: *Reconnect series digraphs in parallel according to the definitions of FSP digraphs.*

```
Fetch a series digraph $G_T$ from the list $L_S$
while the $L_S$ is not empty do
    fetch a series digraph $Gi$ from $L_S$
    if $v_s(G_i) \subseteq V(G_T)$ and $v_t(G_i) \subseteq V(G_T)$ then
        if GT and $G_i$ fail to satisfy the parallel connection
        conditions of FSP digraphs at vertices $v_s(G_i)$ and $v_t(G_i)$
        then
            Transform the topology of $G_T$ into a new digraph
            denoted by $G_T'$, which can connect with $G_i$ in
            parallel to construct a new FSP digraph
            obtain a new FSP digraph $G'$ from $G_T'$ and $G_i$ with
            parallel connection
        else
            obtain a new FSP digraph $G'$ from $G_T$ and $G_i$ with
            parallel connection
        end if
        Let $G_T = G'$
        Put the pair of the source and sink of $G_i$ into the list $L_P$
    end if
end while
```

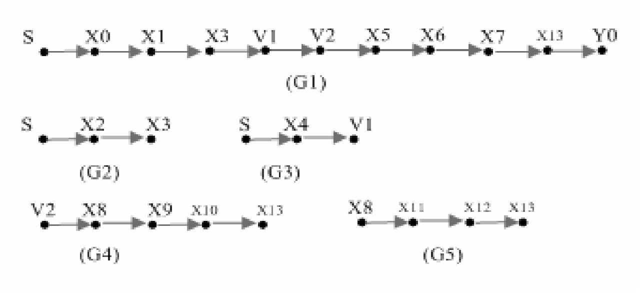


Fig. 4 Series digraphs obtained by decomposing the GSP digraph of fig. 2.

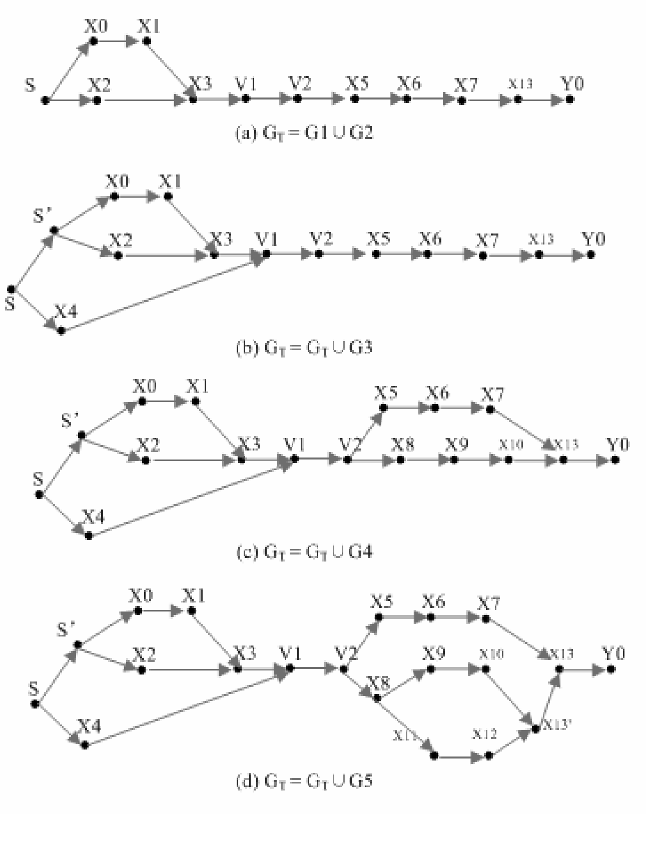


Fig. 5 The process of constructing the new FSP digraph by connecting series digraphs in fig. 4.

Figure 4 shows the series digraphs of the GSP digraph in figure 2. Figure 5 shows the process of constructing the FSP digraph G_T by connecting series digraphs in figure 4 in parallel.

As shown in figure 5(b), the parallel composition conditions is not satisfied when G_T and G_3 connect in parallel at vertices S and V_1, since the corresponding join vertex of S is X_3. A new vertex $S^{'}$ that is equivalent to S in Boolean function is created and used as the split vertex of V_1. Similarly, a new vertex $X_{13}^{'}$ identical to vertex X_{13} has been created when connecting G_T and G_5 in parallel as shown in figure 5(d).

When G_T and G_i with the source v_s and the sink v_t connect in parallel, there are three possibilities:

(i) $J_{G_T}(v_s) \neq v_t \; S_{G_T}(v_t) = \phi$,

(ii) $J_{G_T}(v_s) = \phi \; S_{G_T}(v_t) \neq v_s$,

(iii) $J_{G_T}(v_s) \neq v_t \; S_{G_T}(v_t) \neq v_s$.

For the possibility (i), a new vertex identical to v_s should be created in G_T, as shown in figure 5(b). For the possibility (ii), a new vertex identical to v_t should be created in G_T, as shown in figure 5(d). Finally, for the third possibility, then two vertices identical to v_s and v_t respectively should be created in G_T, as shown in figure 6.

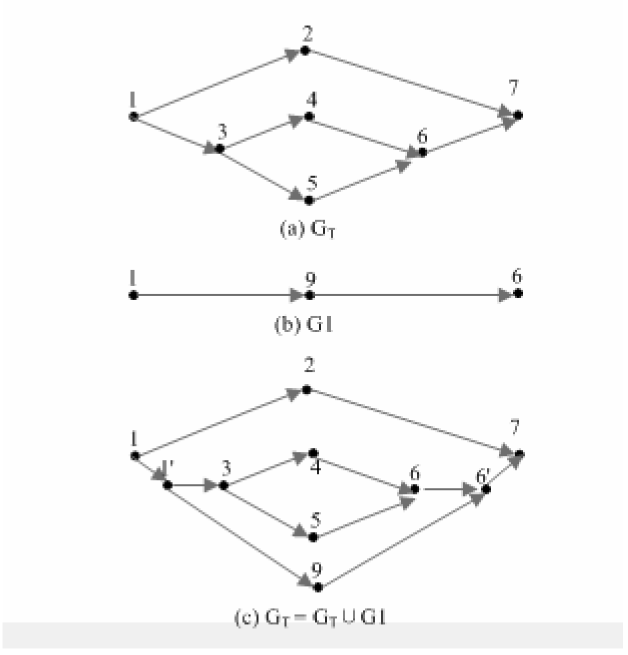


Fig. 6 Two vertices created in transformation process.

4.2 THE ALGORITHM FOR TRANSLATING A FSP DIGRAPH TO AN INSTRUCTION LIST

We now consider the problem of implementing Step 3 of the translation algorithm of a LD into an IL. The algorithm is based on the Lemmas in the third section.

Algorithm 3: <Translation of a FSP digraph into an instruction lists>

Input: a FSP digraph G_T and the list L_P composed of all pairs of corresponding split and join vertices in G_T.

Output: the Boolean function of G_T

```
Step1: Sorting the list L_P.
Let Q be an empty queue
Traverse G_T with width-first search algorithm, and if a split vertex is reached, then insert it into the top of Q
Reorder the elements of L_P according its position in Q
Step2: Reduce G_T with Boolean function.
while (the list L_P is not empty)
    fetch the top element denoted by ($v_s^i$, $v_j^i$) from the L_P
    for (k= 1; k<= $deg^-$ ($v_s^i$); k++)
        node = next vertex that is adjacent from $v_s^i$
        $F^{'}$ = TRUE
        while (node $\neq$ $v_j^i$)
            $F^{'} = F^{'} \cap F$ (node)
            node = vertex that is adjacent from node
        end while
        $F = F \cup F^{'}$
    end for
    $F = F(v_s^i) \cap F(v_j^i) \cap F$
    create a new vertex with Boolean function F to replace the subgraph $Sub(G_T, v_s^i, v_j^i)$
end while
node = $v_s(G)$
$F$ = TRUE
while(node != $v_t(G)$)
    $F = F \cap F$(node)
    node = vertex that is adjacent from node
end while
return F
```

When we apply the algorithm on the FSP digraph G_T in figure 5(d), the elements of Q are X8, V2, $S^{'}$, and S; the elements of the reordered L_P is {X8, X13'}, {V2, X13}, {S', X3}, and {S, V1}. The reducing process is shown in figure 7, and figure 8 shows the instruction lists of the final Boolean function. It is important to point out here that the instruction lists generated by the algorithm have some redundant and needless instructions such as step 1, 2, 8, 11, 12, 13, 23 and 27 in figure 8, which is created by vertices S , S', V1, V2, and X13'. Yet, the redundant instruction can be removed easily by recognizing these vertices in the translation process.

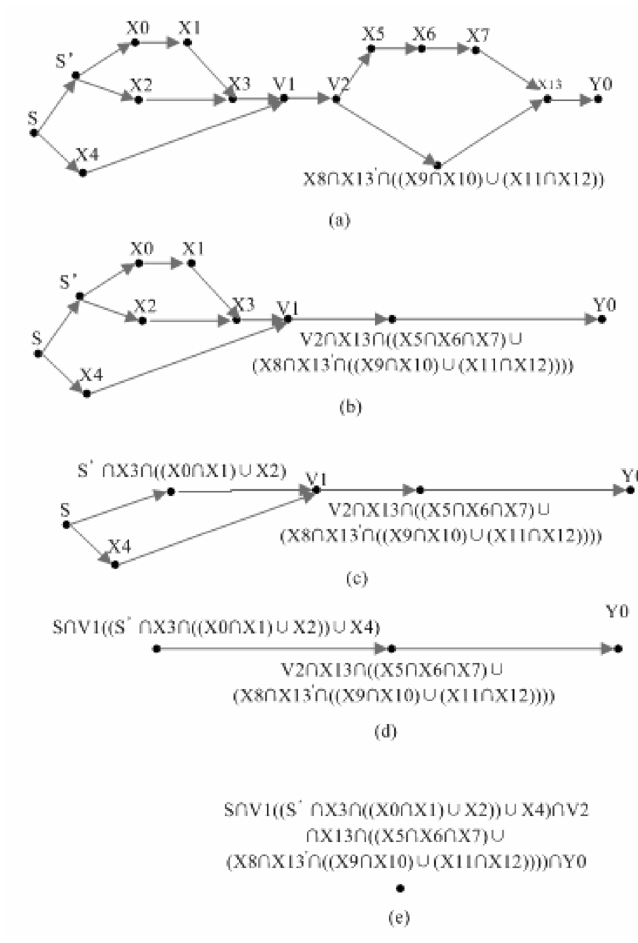
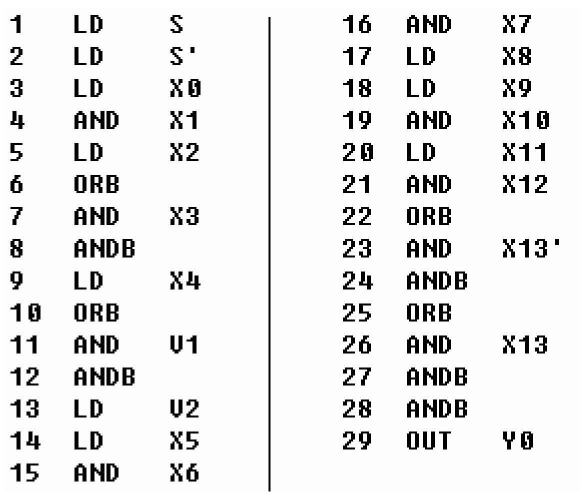

X0
X1
S'
X2
X3
V1
V2
X5
X6
X7
X13
Y0
S
X4
X8∩X13'∩((X9∩X10)∪(X11∩X12))
(a)
V2∩X13∩((X5∩X6∩X7)∪
(X8∩X13'∩((X9∩X10)∪(X11∩X12))))
(b)
S' ∩X3∩((X0∩X1)∪X2)
(c)
S∩V1((S' ∩X3∩((X0∩X1)∪X2))∪X4)
(d)
S∩V1((S' ∩X3∩((X0∩X1)∪X2))∪X4)∩V2
∩X13∩((X5∩X6∩X7)∪
(X8∩X13'∩((X9∩X10)∪(X11∩X12))))∩Y0
(e)

Fig. 7. The process of reducing G_T in figure 5(d) with Boolean function.



```
1   LD    S      | 16  AND   X7
2   LD    S'     | 17  LD    X8
3   LD    X0     | 18  LD    X9
4   AND   X1     | 19  AND   X10
5   LD    X2     | 20  LD    X11
6   ORB          | 21  AND   X12
7   AND   X3     | 22  ORB
8   ANDB         | 23  AND   X13'
9   LD    X4     | 24  ANDB
10  ORB          | 25  ORB
11  AND   V1     | 26  AND   X13
12  ANDB         | 27  ANDB
13  LD    V2     | 28  ANDB
14  LD    X5     | 29  OUT   Y0
15  AND   X6     |
```

Fig. 8. The instruction list of the G_T in figure 5(d).

5. CONCLUSIONS AND FUTURE WORK

In this paper, a complete translation algorithm from a LD to an IL is presented, which use FSP digraphs to depict the Boolean function of ladder diagrams.

FSP digraphs are the powerful model to describe the structure of all networks constructed by connection in parallel or in series of elements, and are more intuitive than binary tree. Presently we are studying to optimize the ILs using FSP digraphs.

The next step will extend the definitions of FSP digraphs and make it support Sequential function charts, Logic block and Petri Nets.

REFERENCES

Chmiel M, Hrynkiewicz E, Muszynski M. (2002) The way of ladder diagram analysis for small compact programmable controller. The 6th Russian-Korean International Symposium on Science and Technology. Novosibirsk, Russia: IEEE Electron Devices Society,: 169-173.

Diestel Reinhard.(2003) *Graph Theory*, Springer-Verlag. New York

Ge Feng, Wu Ning. (2006). Transformation Algorithm Between Ladder Diagram and Instruction List Based on AOV Diagraph and Binary Tree. *Journal of Nanjing University of Aeronautics & Astronautics,* **Vol.38 NO.6**.

Jong-il Kim, Jaehyun Park, Wook Hyun Kwon. (1992). Architecture of a ladder solving processor for programmable controllers. *Microprocessors and Microsystems*.

Jacobo Valdes, Robert E.Tat'jan, Eugene L.Lawler. (1979). The recognition of Series Parallel digraphs. *Journal ACM.*

Karl-Heinz J, Tiegelkamp M. (2001). *Programming industrial automation systems. IEC61131-3,* Springer-Verlag Berlin

Kim H S, Kwon W H, Chang N. (1999). A translation method for ladder diagram with application to a manufacturing process. Proceedings of the IEEE International Conference on Robotics and Automation. Detroit, Michigan: Robotics and Automation Society: 793-798.

L.Ngalamou, L.Buchanan, L.Myers, V.Watt. (2004). Architecture of a Retargetable Ladder Logic Diagrams Tool. SICE Annual conference. Sapporo.

Shih-Yih Wang, Lih-Hsing Husu. (1992). Maximum and minimum matchings for series-parallel networks. *IEEE.*