Int J Adv Manuf Technol (2005) 25: 754–760
DOI 10.1007/s00170-003-1902-9

ORIGINAL ARTICLE

Jin-Shyan Lee · Pau-Lo Hsu

A systematic approach for the sequence controller design in manufacturing systems

Received: 15 March 2003 / Accepted: 27 May 2003 / Published online: 24 November 2004

Abstract This paper presents a systematic approach for the design and implementation of the sequence controller in manufacturing systems. By employing the IDEF0, we construct the simplified Petri net controller (SPNC) through the material flow diagram and the information flow diagram. Then, the ladder logic diagram (LLD) can be transformed from the SPNC through the token passing logic (TPL). The proposed approach, including the IDEF0, SPNC, and TPL tools, leads to the standard IEC1131-3 LLD for PLC implementation. Finally, an application of a stamping process is provided to illustrate the design procedure of the developed approach.

Keywords IDEF0 · Ladder logic diagrams · Manufacturing systems · Petri nets · Programmable logic controllers · Sequence controllers

1 Introduction

Recently, automated manufacturing systems have become more complex. In the control of manufacturing systems, the sequence control for the discrete events plays an important role [1]. Basically, the ladder logic diagram (LLD) has been used to conduct the control software's sequence of operations and it is usually implemented with a programmable logic controller (PLC). The PLC has the advantages of reliability, robustness, and direct programmability. The I/O procedures of the PLC are specified by the LLD and industrial machines, and thus perform repetitive operations in sequence.

For simple systems, it is easy to program the LLD by heuristic methods. However, as systems become more complex, the controller design becomes more difficult and the LLD implementation becomes more complicated. Moreover, because the PLC is usually programmed only to control the process, qualitative analysis and the performance characteristics of the implemented PLC controllers are seldom discussed in practice. In addition, as design specifications change, the LLD program usually needs to be modified significantly [2–4]. Hence, researchers are pursuing a systematic and efficient approach to the design and implementation of the sequence controller.

In the past years, Petri nets (PN) became popular tools for designing sequence controllers in manufacturing systems [5–10]. Uzam and Jones [11] proposed an extended PN method to analyze a target system and then implemented it via the LLD. Feldmann, et al. [12, 13] used colored PN to form the structured text (ST) for PLC implementation. In fact, most industrial PLC users still prefer to program in LLDs. Although some researchers have attempted to convert Petri nets into LLDs [11, 14–17], the resulting LLDs are usually more complex compared to those programmed directly by engineers. Moreover, it is not straightforward or easy for manufacturing engineers to construct the PN models.

In this paper, a systematic approach is proposed for the design and implementation of the sequence controller in manufacturing systems. By introducing the sensor state into the PN to form a simplified Petri net controller (SPNC), we obtain a more compact LLD structure through the token passing logic (TPL). Typically, the sensor state is used to trigger sequences in manufacturing. We show that the integration definition language 0 (IDEF0) [18] can be used to obtain the SPNC model through the material flow diagrams and information flow diagrams in sequence. The proposed IDEF0/SPNC/TPL/LLD approach, including the IDEF0, SPNC, and TPL tools, then leads to the industrial standard IEC1131-3 [19] LLD for PLC implementation.

The remainder of this paper is organized as follows. First, the SPNC is defined in Sect. 2. Then, Sect. 3 introduces the proposed IDEF0/SPNC/TPL/LLD approach. In Sect. 4, an application example of a stamping process is provided to illustrate the proposed approach. Conclusions and recommendations for further research are provided in Sect. 5.

J.-S. Lee · P.-L. Hsu (✉)
Department of Electrical and Control Engineering,
National Chiao-Tung University,
1001 Ta-Hsueh Road, Hsinchu, Taiwan
E-mail: plhsu@cc.nctu.edu.tw

2 Simplified Petri net controller

In this section, we propose a simplified Petri net controller (SPNC) by introducing sensor states into the ordinary PN. The SPNC is applied to simulate the manufacturing system and to lead the IDEF0 to LLD in the proposed IDEF0/SPNC/TPL/LLD approach.

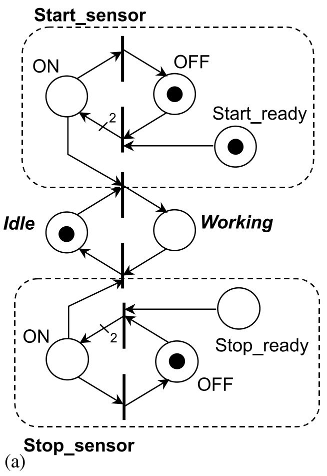
2.1 Formal definition

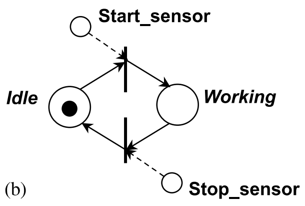
Fig. 1a shows an ordinary PN model for pushing a button to trigger a process. When using the ordinary PN approach in controlling manufacturing processes, dealing with multiple sensor readings makes the net structure more complicated and difficult to analyze. Therefore, by introducing the sensor state into the PN to form an SPNC, implementation of the net structure is simplified. From the control point of view, as shown in Fig. 1b, the sensor state in the SPNC replaces the reading sensor model such as push buttons or limit switches within the ordinary PN. Note that the condition of sensor states may change depending on the practical situation. Thus, as sensors increase in processes, the net structure of the SPNC is greatly simplified, as shown in Fig. 1c. Then, it becomes easy to model and implement the sequence controller through the SPNC defined as:

$$\text{SPNC} = (P, T, A, S, M_0)$$

where,

$P = \{P_1, P_2, \ldots, P_m\}$	is a finite set of places,
$T = \{T_1, T_2, \ldots, T_n\}$	is a finite set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$,
$A \subseteq \{P \times T\} \cup \{T \times P\}$	is the set of arcs between the places and transitions,
$S = \{S_0, S_1, \ldots, S_n\}$	is the set of sensor states, and
$M_0 : P \to 1$	is the initial marking.





Elements	PN	SPNC
Place	8	2
Transition	6	2
Arc	16	6
Sensor state	–	2
(c) Total	30	12

Fig. 1a–c. The comparison between the PN and the SPNC via a simple process. **a** Ordinary PN. **b** SPNC. **c** Comparison results

2.2 Graphical representation

As shown in Fig. 2, the SPNC consists of three kinds of nodes: (1) the place, drawn as a circle, (2) the transition, drawn as a bar, and (3) the sensor state, drawn as a smaller circle with a hidden arrow. The arcs, represented by directed arrows, are either from a place to a transition or from a transition to a place. In modeling, the marking conditions of places represent the status of the system and the transitions represent events. A transition has a set of input and output places, which represent the pre-conditions and post-conditions of the event, respectively. A sensor state, associated with its transitions, represents the sensor readings as a firing condition, which triggers a manufacturing sequence. The sensor state is a Boolean variable that can be 0, in which case the related transition is not fired, or 1, in which case the related transition is fired if it is enabled. The marking of the SPNC is represented by the number of tokens in each place, drawn as black dots. The presence of a token in a given place means that the associated condition is true or that the actions associated with this place are taken.

2.3 Dynamic behavior

The dynamic behavior of a system is simulated by the distribution of tokens in places as the enable transitions fire. The flow of tokens in the SPNC is governed according to the following rules:

1) Enabling rule:
   A transition is said to be enabled if all its input places are marked.
2) Firing rule:
   Furthermore, the enabled transition is fired if all its sensor states are true. When an enabled transition fires, it removes one token from all its input places and deposits one token into all its output places at the same time.

2.4 Comparison with other models

The behavior of the proposed SPNC is similar to the sequential function chart (SFC). However, since SFC is derived from PN with some modifications and simplifications, theoretical results of PN cannot be directly applied to SFC [20]. Since the present SPNC is an extension of the PN by introducing sensor states, SPNC allows formal analysis of various properties, such as the safety, liveness, and reversibility for the process [9, 10]. Moreover, SFC only offers the method for depicting sequences of control system without providing any mechanisms to perform the functional analysis. Note that in the present

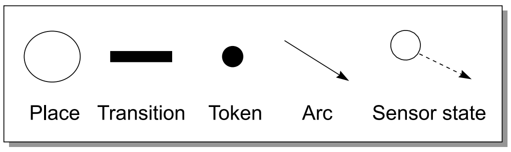


Fig. 2. The icon definition of the SPNC.

IDEF0/SPNC/TPL/LLD approach, by applying the IDEF0 for functional analysis and information flow design, the SPNC model can be transformed from the information flow diagram.

Furthermore, in contrast to other extended PN applications such as Interpreted PN [21], Automation PN [11], or Signal Interpreted PN [16], which use external events to model sensor readings, the present SPNC simply applies the sensor states to model the firing conditions. Also, the present IDEF0/SPNC/TPL/LLD approach obtains the PLC programs systematically, from the design specifications through the SPNC, and to the final LLD. Since the PN model is inherently concurrent, whereas the LLD is typically scan-based, the sequential specification must be determinate and deterministic [5] in the present approach. Also, the mono marked restrictions design is required in the proposed SPNC to guarantee the safety of the sequence in practice.

3 The IDEF0/SPNC/TPL/LLD approach

In this section, the integrated IDEF0/SPNC/TPL/LLD approach, including the IDEF0, SPNC, and TPL tools, is proposed to systematically obtain the LLD for PLC implementation. The design procedure of the IDEF0/SPNC/TPL/LLD approach, depicted in Fig. 3, consist of five stages and each stage is described as follows.

3.1 Functional analysis stage: IDEF0

With the given specifications, the purpose of functional analysis is to realize the functions and operations of the system and then generate the control signals for the next stage. At this stage, each function of the manufacturing system has to be specified with a top-down hierarchically decomposing process by using the IDEF0 [18]. IDEF0 is an activity-oriented modeling approach and its representation of a manufacturing process consists of an ordered set of boxes representing activities performed by the system. The inputs are those items transformed by the activity and the outputs are the results of the activity, as shown in Fig. 4. The mechanisms, drawn as supporting arrows, represent resources such us machines, computers, and operators, etc. The decomposition process continues until there is sufficient detail on the basic activities to serve the purpose of sequence control. A functional model of the material flow diagram is obtained at this stage.

3.2 Information flow design stage: IDEF0

At this second stage, the information flow is used to control the material flow in a manufacturing system. The information flow diagram is constructed from the material flow diagram with static analysis, again using the IDEF0. In the information flow diagram, the input and output commands are designed to enable the activity and to change the machine status after firing, respectively. Because the mechanisms will be assigned within the I/O ports at the layout stage later, the supporting arrows for mechanisms are omitted here to simplify the information flow design. The sensor readings representing the conditions to fire the activity are drawn as control signal arrows. A controllable model of the information flow diagram is obtained at this stage.

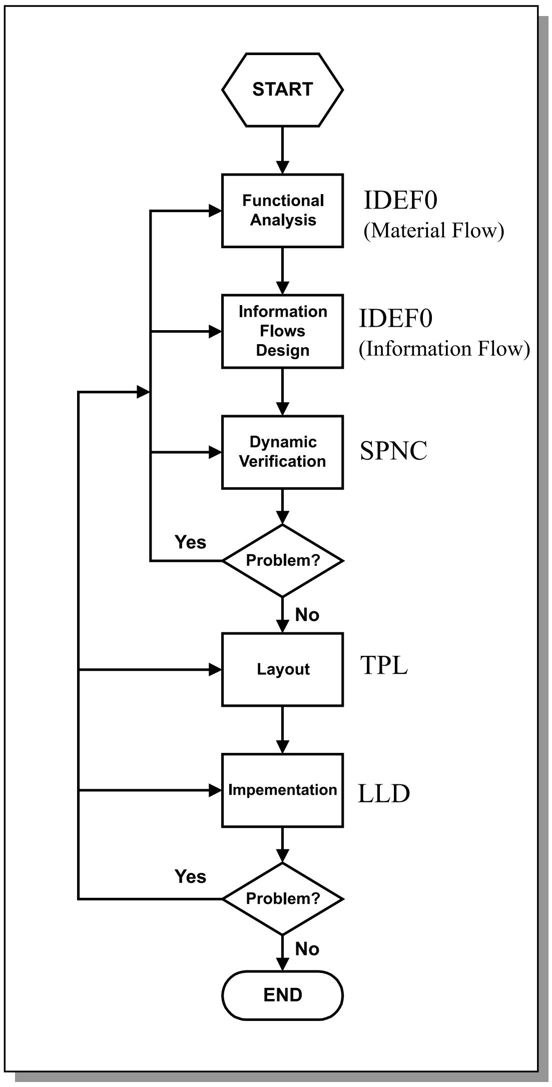


Fig. 3. Design procedure of the IDEF0/SPNC/TPL/LLD approach

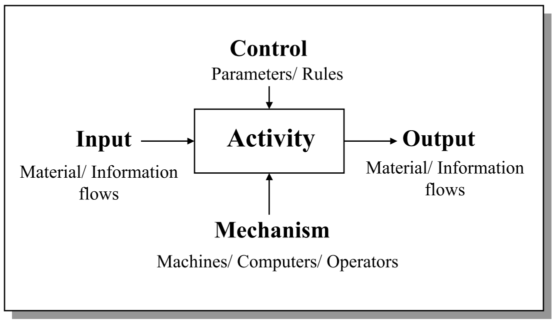

Control
Parameters/ Rules
Input
Material/ Information flows
Activity
Output
Material/ Information flows
Mechanism
Machines/ Computers/ Operators

Fig. 4. The IDEF0 scheme

3.3 Dynamic verification stage: SPNC

The information flow diagram only represents system activities and their interrelationships. Since it does not show direct logical and dynamic dependencies between activities, it is transformed into a dynamic SPNC model, which is applied to verify the dynamic behavior of the system. The transformation of information flow diagram into SPNC model is based on the following steps:

1. An activity box in the information flow diagram is transformed into a transition of the SPNC.
2. The input and output commands are transformed into input and output places, respectively.
3. The control signals of the sensor readings are transformed into sensor states.
4. The initial marking of the SPNC is set according to the initial condition of the system.

An example is shown in Fig. 5. The activity of the information flow diagram is transformed into the transition T_1. The input command I_1 and output command I_2 are transformed into the input place P_1 and output place P_2, respectively, and the control signal 'control' is transformed into the sensor state S_1. When the SPNC model is obtained, the correctness of the sequence order can be verified by studying the behaviors via computer simulations. Also, the properties of the PN such as the safety, liveness, and reversibility can be analyzed to identify the dynamic behavior [9].

3.4 Layout stage: TPL

To simplify the conversion of the SPNC into the LLD, the token passing logic (TPL) is employed in this stage [11]. The attractive feature of the TPL is that it facilitates the direct conversion of a SPNC into a generic form of control logic, which may be implemented with low-level languages such as LLD, or with high-level languages such as C. This is achieved by adopting the SPNC concept of using tokens as the main mechanism for controlling the flow of the control logic. At this stage, the SPNC model is transformed into the TPL model to assign the I/O ports for actions and sensor readings. For applications in a variety of industrial PLC hardware, the TPL is modified as follows:

The Modified TPL = $(M, T, A, in, out, time)$,

where,

$M = \{M_1, M_2, \ldots, M_m\}$	is a finite set of memory bits,
$T = \{T_1, T_2, \ldots, T_n\}$	is a finite set of transitions,
$A \subseteq \{M \times T\} \cup \{T \times M\}$	is the set of arcs between the memories and transitions,
$in = \{in_0, in_1, \ldots, in_n\}$	is the set of sensor inputs,
$out = \{out_1, out_2, \ldots, out_m\}$	is the set of actuator outputs, and
$time = \{time_1, time_2, \ldots, time_m\}$	is the set of delay timers.

The transformation from the SPNC model into the TPL form is based on the following steps:

1. The transition of the SPNC is transformed into a transition of the TPL.
2. The place is transformed into a memory bit.
3. The sensor state is transformed into a sensor input.
4. For the action with a place, besides the memory bit, an actuator output is assigned.
5. For the delay time with a place, besides the memory bit, a delay timer is assigned.

An example is shown in Fig. 6. The places P_1 and P_2 are transformed into the memory bits M_1 and M_2, respectively, and the sensor state S_1 is transformed into the sensor input in_1. Assume there is an action with P_2, the actuator output out_1 is assigned. Hence, each place whose capacity is limited to one within the SPNC corresponds to a memory bit in the TPL. The token flow is then simulated by setting and resetting these memory bits. Thus, each place within the SPNC has at least one associated memory bit in the TPL. The sensor state within the SPNC corresponds to a sensor input contact in the TPL. To simulate the firing of a transition, if the memory bits associated with input places are set and the sensor inputs of the transition yield "true", the memory bits at the input places are reset and the

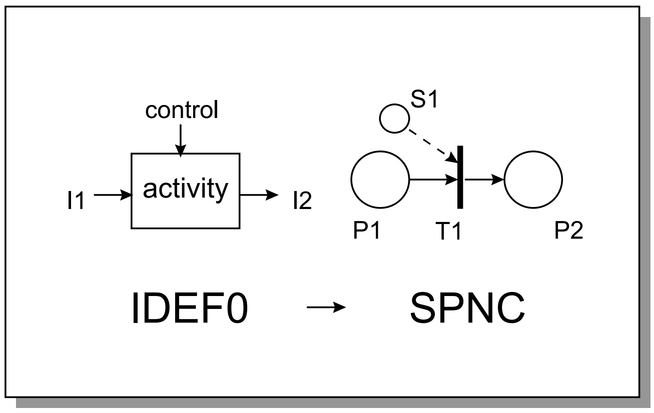


Fig. 5. The transformation from the IDEF0 to the SPNC

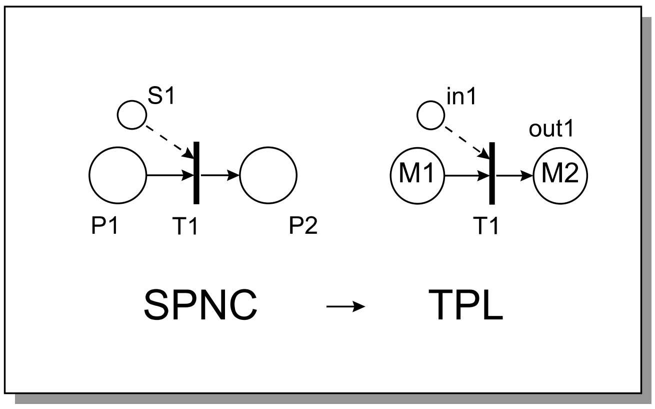


Fig. 6. The transformation from the SPNC to the TPL

memory bits at the output places are set simultaneously. Moreover, the actions and delays within the SPNC are assigned to appropriate memory bits within the TPL by using the actuator outputs and delay timers, respectively. By using the Modified TPL, the I/O ports for the sensor readings and actuator outputs are assigned and the layout for implementation in LLD can be completed. The Modified TPL bridges the gap between SPNC and LLD and provides a simple way of developing PLC controllers.

3.5 Implementation stage: LLD

In order to convert the TPL model into LLD code for real-time implementation, a direct mapping is used from the TPL to the LLD by maintaining the enabling and firing rules at this stage. The transformation from TPL model into LLD format is based on the following steps:

1. *Initial condition setting*: the token in the SPNC is mapped to the corresponding internal relay with the SET command.
2. For each transition, the input memory is mapped to a conditional contact and an internal relay with the RST command, and the output memory is mapped to an internal relay with the SET command.
3. The sensor input is mapped to a conditional contact for the associated transition.
4. The output relay is assigned to send the command to perform the operation.
5. The delay timer is assigned to perform the delay.

An example is shown in Fig. 7. For transition T_1, the input memory M_1 is mapped to a conditional contact and an internal relay M_1 with the RST command, and the output memory M_2 is mapped to a internal relay M_2 with the SET command. The sensor input in_1 is mapped to a conditional contact X_1 and the actuator output out_1 is mapped to the output relay Y_1. By integrating the initial condition and setting all transitions, the LLD for sequence control is thus completed.

In the proposed IDEF0/SPNC/TPL/LLD approach, the material flow diagram and the information flow diagram are obtained by using the IDEF0 technique for functional analysis and information flow design. Then, the information flow diagram is transformed into the SPNC model to verify its dynamic behavior. Subsequently, the SPNC model is converted into a modified

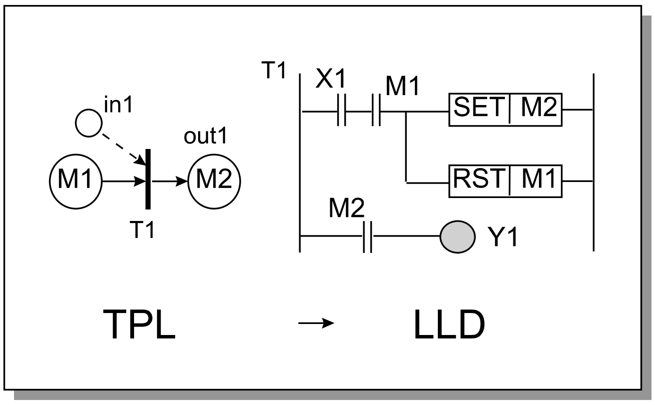


Fig. 7. The transformation from the TPL to the LLD

TPL model for implementation layout. Finally, the IEC1131-3 LLD for implementation on the PLC controller is obtained using a direct mapping from the TPL to the LLD. Figure 8 summarizes the transformations in the proposed IDEF0/SPNC/TPL/LLD approach.

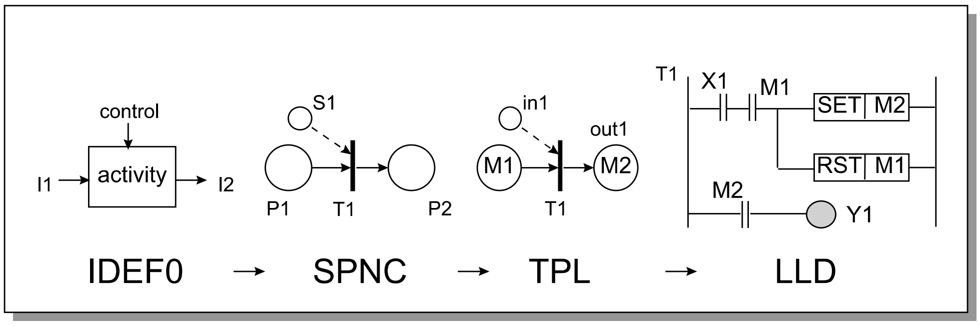
4 An application example

To demonstrate the viability of the developed approach, an application to a stamping process is provided.

4.1 System description

As shown in Fig. 9, a stamping system consists of three cylinders that are operated by four-port and two-way solenoid valves. Each cylinder has two normally open limit switches. For example, when the end of pusher_A contacts limit switch a0, a0 is then closed. This indicates that pusher_A is at the end of its return stoke. The whole system has 7 input sensors corresponding to six limit switches, one push button for starting the system and six output actuators corresponding to six solenoid valves. In the stamping process, pusher_A moves the workpiece from a store onto the worktable. Then the workpiece is stamped by stamper_B and afterwards is ejected by thrower_C. Thus, the sequence of the stamping system is A+, B+, {A−, B − }, C+, C−,

Fig. 8. The transformations of the IDEF0/SPNC/TPL/LLD approach



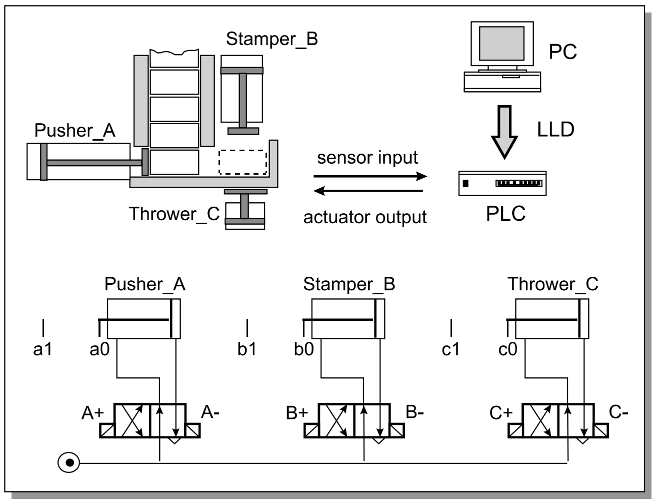


Fig. 9. The stamping system

where the plus and the minus signs mean a piston performing forward strokes and return strokes, respectively. {A−, B − } represents two concurrent actions as the pistons of both pusher_A and stamper_B perform return stokes simultaneously.

Table 1. Notations for the stamping process

SPNC element	TPL element	LLD element	Description
P_1	M_1	M_1	Ready
P_2	M_2, out_1	M_2, Y_1	Holding {A}
P_3	M_3, out_2	M_3, Y_2	Stamping {B}
P_4	M_4, out_3	M_4, Y_3	Releasing {A}
P_5	M_5, out_4	M_5, Y_4	Releasing {B}
P_6	M_6	M_6	–
P_7	M_7	M_7	–
P_8	M_8, out_5	M_8, Y_5	Throwing {C}
P_9	M_9, out_6	M_9, Y_6	Resetting {C}
T_1	T_1	–	Push in and Hold on {A+}
T_2	T_2	–	Stamp down {B+}
T_3	T_3	–	Release workpiece {A−, B −}
T_4	T_4	–	–
T_5	T_5	–	–
T_6	T_6	–	Throw out {C+}
T_7	T_7	–	Reset {C−}
T_8	T_8	–	Repeat {A+}
S_0	in_0	X_0	Push button {ON}
S_1	in_1	X_1	Sensor a1 {ON}
S_2	in_2	X_2	Sensor b1 {ON}
S_3	in_3	X_3	Sensor a0 {ON}
S_4	in_4	X_4	Sensor b0 {ON}
S_5	in_5	X_5	Sensor c1 {ON}
S_6	in_6	X_6	Sensor c0 {ON}

4.2 Sequence controller design

Through the use of the proposed IDEF0/SPNC/TPL/LLD approach, as shown in Fig. 10, the LLD code for real-time implementation on PLC controllers was systematically generated. First, the material flow diagram and the information flow diagram were obtained by using the IDEF0 technique. Then, to

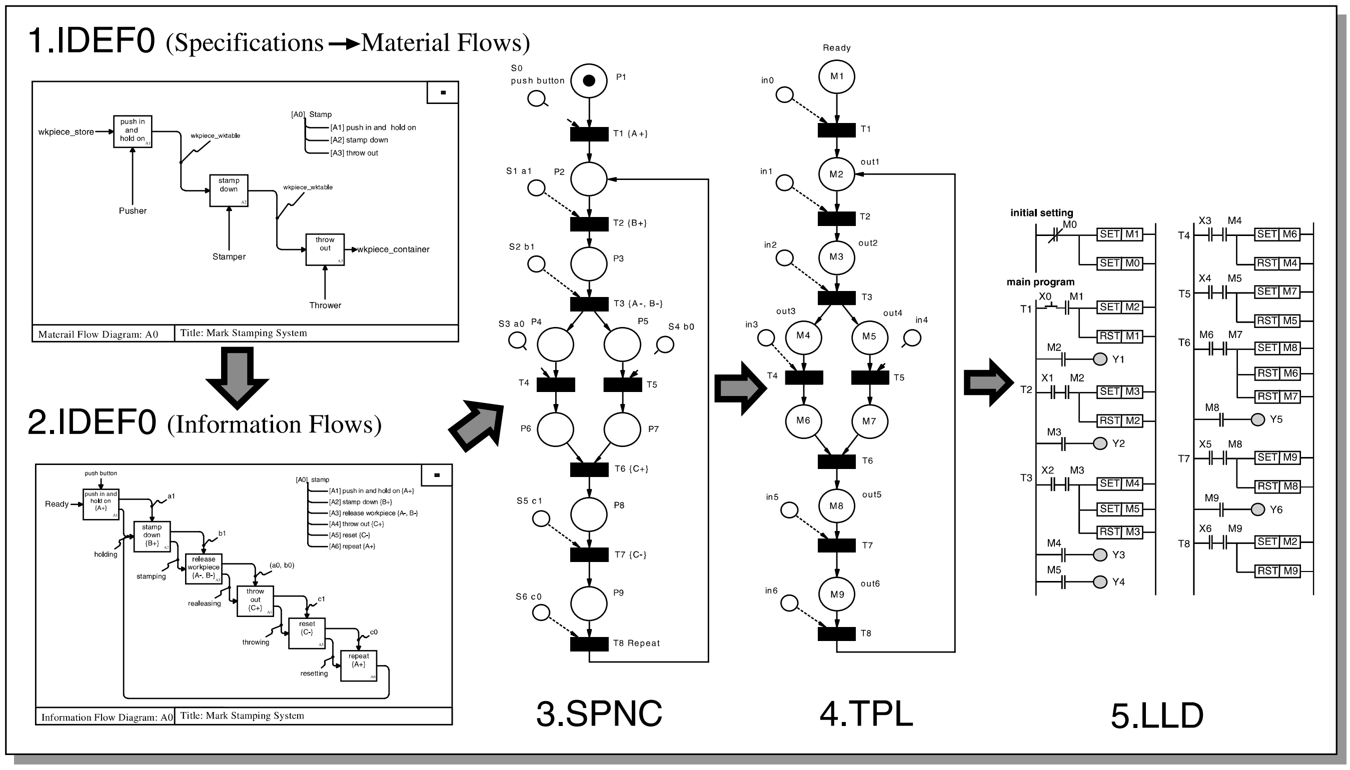


Fig. 10. Design of the sequence controller using IDEF0/SPNC/TPL/LLD approach

verify its dynamic behavior, the information flow diagram was transformed into the SPNC model. Subsequently, the SPNC model was converted into a modified TPL model for layout. Finally, the LLD for implementation with PLC controllers was obtained by a direct mapping from the TPL. This LLD code is written for Mitsubishi FX2 PLCs, which meet IEC1131-3. Table 1 gives the notations used in the IDEF0/SPNC/TPL/LLD together with their descriptions.

5 Conclusions

In this paper, we have proposed a systematic IDEF0/SPNC/TPL/LLD approach for the PLC-based sequence controller design in manufacturing systems. To obtain the LLD for PLC implementation, the SPNC is defined by introducing the sensor states into the ordinary Petri net. This leads to meaningfully simplified process modeling. Moreover, the IDEF0 technique is employed to construct the SPNC model through the material flow diagram and information flow diagram. Starting from the basic sequential specification, the proposed approach includes IDEF0, SPNC, and TPL, and systematically leads to the standard IEC1131-3 LLD for PLC implementation. An application of a stamping process is provided to demonstrate the viability of the developed approach. In the future, we plan to enrich the sequence controller with the capability of fault diagnosis, and to apply the IDEF0/SPNC/TPL/LLD approach to more complicated systems.

Acknowledgement This work was supported by the National Science Council, R.O.C., under Grant NSC 89-2218-E-009-034.

References

1. Tilbury D, Khargonekar P (2001) Challenges and opportunities in logic control for manufacturing systems. IEEE Control Syst Mag 21(1): 105–108
2. Venkatesh K, Zhou MC, Caudill RJ (1994) Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system. IEEE Trans Ind Electron 41(6):611–619
3. Zhou MC, Twiss E (1998) Design of industrial automated systems via relay ladder logic programming and Petri nets. IEEE Trans Syst Man Cybern C 28(1):137–150
4. Lee JS, Hsu PL (2001) A new approach to evaluate ladder logic diagrams and Petri nets via the IF-THEN transformation. In: Proceedings of IEEE International Conference on Systems Man, and Cybernetics, Tucson, USA, pp 2711–2716, October 2001
5. David R, Alla H (1994) Petri nets for modeling of dynamics systems – a survey. Automatica 30(2):175–202
6. Liang GR, Hong HM (1994) Hierarchy transformation method for repetitive manufacturing system specification, design, verification and implementation. Comput Integr Manuf Syst 7(3):191–205
7. Janneck JW, Naedele M (1998) Modeling a die bonder with Petri nets: a case study. IEEE Trans Semicond Manuf 11(3):404–409
8. Santarek K, Buseif IM (1998) Modeling and design of flexible manufacturing systems using SADT and Petri nets tools. J Mater Process Technol 76:212–218
9. Zimmermann A, Hommel G (1999) Modeling and evaluation of manufacturing systems using dedicated Petri nets. Int J Adv Manuf Technol 15(2):132–138
10. Lee JS, Hsu PL (2000) A PLC-based design for the sequence controller in discrete event systems. In: Proceedings of IEEE International Conference on Control Applications, Anchorage, USA, pp 929–934, September 2000
11. Uzam M, Jones AH (1998) Discrete event control system design using automation Petri nets and their ladder diagram implementation. Int J Adv Manuf Technol 14(10):716–728
12. Feldmann K, Colombo AW, Schnur C, Stockel T (1999) Specification, design, and implementation of logic controllers based on colored Petri net models and the standard IEC1131. I. Specification and design. IEEE Trans Control Syst Technol 7(6):657–665
13. Feldmann K, Colombo AW, Schnur C, Stockel T (1999) Specification, design, and implementation of logic controllers based on colored Petri net models and the standard IEC1131. II. Design and implementation. IEEE Trans Control Syst Technol 7(6):666–674
14. Satoh T, Oshima H, Nose K, Kumagai S (1992) Automatic generation system of ladder list program by Petri net. In: Proceedings of IEEE International Workshop on Emerging Technologies and Factory Automation (ETFA), pp 128–133
15. Jafari MA, Boucher TO (1994) A rule-based system for generating a ladder logic control program from a high level system model. J Intell Manuf 5:103–120
16. Frey B (2000) Automatic implementation of Petri net based control algorithm on PLC. In: Proceedings of American Control Conference 4:2819–2823
17. Lai HF, Lee CE (2001) A hybrid specification method for the design of a workcell controller in manufacturing systems. Int J Adv Manuf Technol 17(12):928–938
18. National Institute of Standards and Technology (1993) FIPS 183 Integration Definition for Function Modeling (IDEF0). NIST, USA
19. International Electrotechnical Commission (1993) Programmable Controllers Part 3, Programming Languages. IEC 1131-3, IEC Geneva
20. Miyazawa I, Tanaka H, Sekiguchi T (1997) Verification of the behavior of sequential function chart based on its Petri net model. In: Proceedings of IEEE International Workshop on Emerging Technologies and Factory Automation (ETFA) pp 532–537
21. Moalla M (1985) Réseaux de Petri interprétés et Grafcet. TSI-Technique et Science Informatique 14(1):17–30