PLC Implementation of a DES Supervisor for a Manufacturing Testbed

R. J. Leduc and W. M. Wonham

Systems Control Group
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ont., M5S 1A4
email: leduc@control.toronto.edu, wonham@control.toronto.edu

Abstract - We describe an implementation of Ramadge-Wonham (RW) discrete-event supervisors for a manufacturing testbed. Testbed modeling and supervisor design are discussed. Modular control architecture is adopted. Component supervisors are represented as equivalent clocked Moore synchronous state machines (CMSSM), which finally are implemented on PLCs.

1 Introduction

In this paper we address the conversion of a theoretical discrete event system (DES) supervisor, designed using RW supervisory control theory ([8], [12]), to a physical implementation on programmable logic controllers (PLCs:[6], [9]). This has been shown to be feasible by Brandin ([2], [3]) for specific applications; here we propose a general approach. A PLC based testbed that physically simulates a manufacturing workcell was built for illustration.

We describe the modeling of the testbed, and the design of its controllers. An implementation method for DES supervisors using clocked Moore synchronous state machines (CMSSM:[11]) is proposed. A formal model for CMSSM is defined and it is shown that a CMSSM contains equivalent control information. Finally, implementation of a CMSSM on a PLC is discussed.

2 Description of Testbed

The testbed is designed to simulate a manufacturing workcell, in particular, problems of routing and collision. The configuration of the testbed (Fig. 1) is similar to a portion of the product routing design of the Motorola Fusion Factory [10].

The testbed is composed primarily of model railroad components. The tracks are laid out to resemble a set of three interacting work units. The two trains simulate Automated Guided Vehicles (AGVs) that convey material to and from the manufacturing units. Each unit has a small electric crane to simulate robots loading and unloading the AGVs.

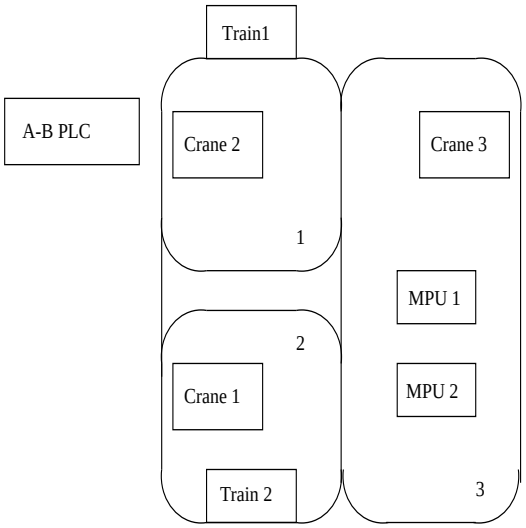

Figure 1: Layout of Testbed

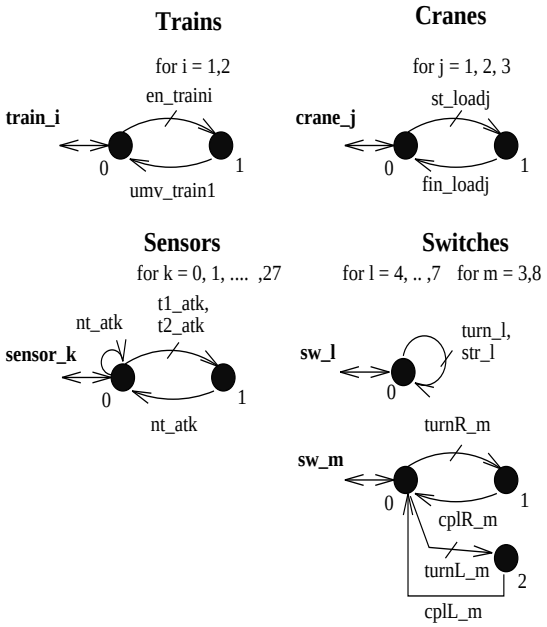

Figure 2: Fundamental Plant Models

The system contains six remote track switches to control the paths of the trains, plus 26 sensors that detect the presence and identity of each train.

The testbed is controlled by two embedded MC68332 processors and an Allen-Bradley PLC. The microprocessors control the trains and cranes directly, and also have a discrete interface for communication with the PLC. The PLC is responsible for control of the overall system.

3 DES Models and Supervisors

In this section we discuss the models used for the testbed, the desired control specifications, and the supervisors that enforce these specifications.

3.1 Plant Model

The plant model for the testbed consists of the following basic elements: trains, cranes, sensors and switches. They are shown in Fig. 2. These models are augmented by additional specifications that define how the elements interact with one another. In total, there are 108 modular specifications.

The interaction models show the interdependencies of the basic models. The first series of models (Fig. 3) shows how the sensors depend on train movement. If all trains are stationary, then every sensor is disabled. The next series of models represents the sensors' interdependencies. With respect to the starting position of a particular train, sensors can only be reached in a particular order, dictated by their physical location on the testbed.

The following series of models detail the sensors' dependency on the switches. Depending on the current location of the trains, certain sensors are inaccessible if the switch prevents the train's approach. An example of this model is given in Fig. 4.

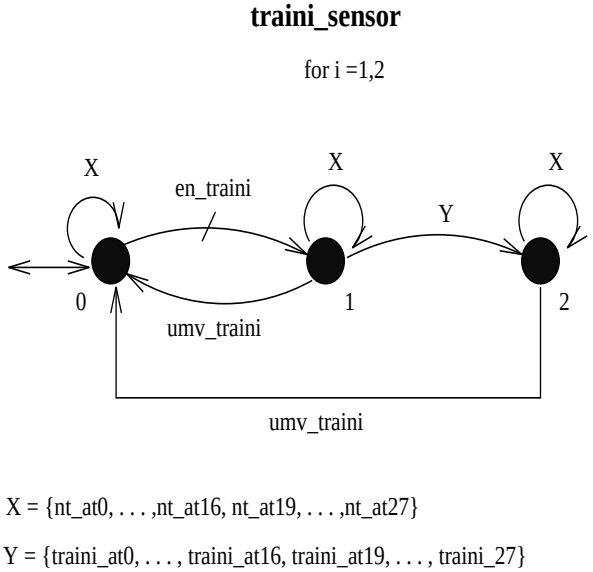

Figure 3: Sensor Dependencies on Trains

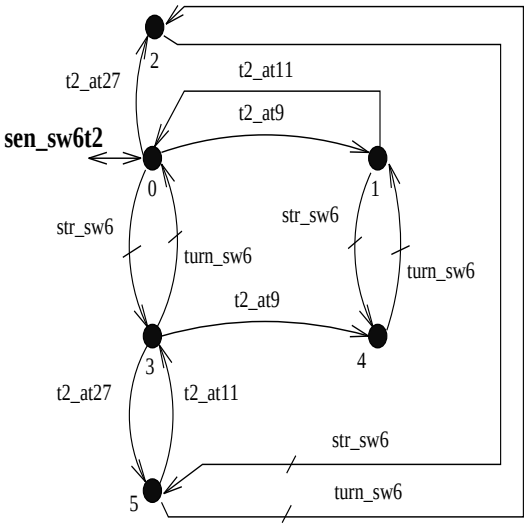

Figure 4: Sensor Dependency for Switch 6

The final series of models is slightly different from the previous series. This series, the switch request handlers, manages how the controllers interact with the switches. The handlers disable the switches' activation events until they are specifically requested.

3.2 Control Specifications

The testbed's safety and operation specifications are as follows: prevent trains from colliding, ensure switches are set correctly while a train is traversing them, enforce specified routes, and synchronize trains to permit loading by cranes.

3.3 DES Supervisors

After the plant model was developed, 29 modular supervisors were designed to enforce the given control specifications. The controllers work as follows. The routing controllers track the progress of each train and activate track switches when necessary.

A crane controller monitors its immediate sensor for a train. When a train arrives, it is stopped, loaded and then released. The crane controller for crane 1 is shown in Fig. 5.

The final class of supervisors are the collision protection controllers. Each track section has its own controller that permits only one train to occupy that portion of the track at any time. Fig. 6 shows an example controller. Note that Fig. 5 and Fig. 6 are complete definitions of the indicated supervisors except for the omission of unimportant self-looped events.

3.4 Discussion of Design Results

After the testbed was modeled, it became apparent that centralized control would be infeasible due to the large size of the composite model (on the order of 10^{16} states). By use of Theorem 2 in [4], we were able to verify controllability for the testbed's 29 modular supervisors. The cited theorem identifies a reduced plant model that can be

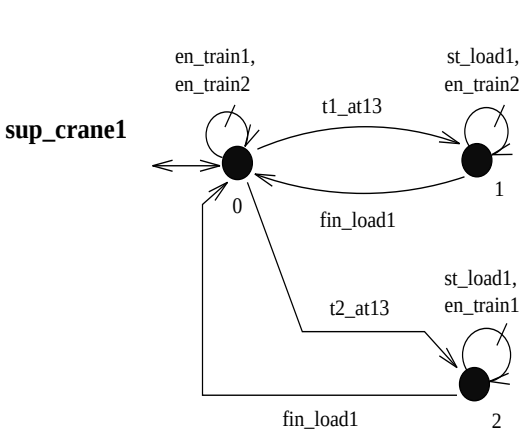

Figure 5: Supervisor for Crane 1

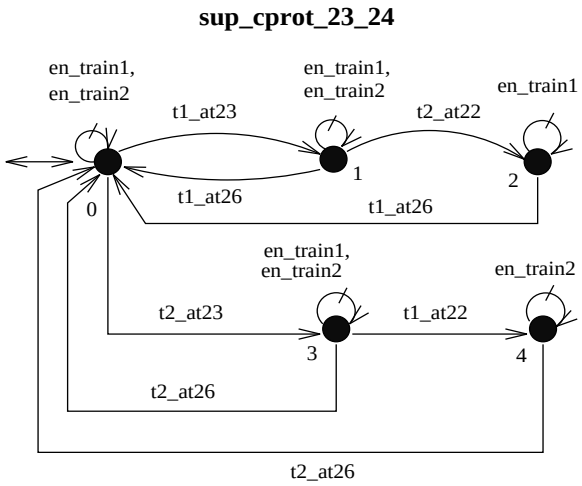

Figure 6: Collision Protection Supervisor

used to verify controllability for a given supervisor. Nonblocking of the testbed has not yet been verified computationally owing to the strong interaction among its component sub-plants.

4 PLC Implementation

Our proposed implementation technique is to translate supervisors into CMSSM, and then to express the boolean logic defining a CMSSM in Relay Ladder Logic for execution on the PLC.

4.1 Synchronous State Machines

A clocked synchronous state machine is a standard way to implement control functions in digital hardware. The state machine only changes states on the rising (falling) edge of a clock pulse. A Moore machine is such that the output is a function only of the present state.

In a CMSSM, all variables represent a binary digit, and the next state vector is determined by a boolean logic equation in the input variables and the current state variables.

4.2 Formal Model

To prove that a CMSSM is an equivalent control representation, we first present a formal model.

Let the CMSSM be represented by the 7-tuple

$$H = (I, Z, Q, \Omega, \Phi, Q_{Res}, T)$$

Here I represents inputs to the state machine; Z represents outputs from the state machine; Q represents the state vector; Ω is the next state function; Φ is the state to output map; Q_{Res} represents the initial or reset state, and T is the period of the clock pulse that drives the state machine.

In detail, let the states of the CMSSM be represented by:

$$Q = [q_0, q_1, \ldots, q_{l-1}], \;\; q_j \in \{0,1\}, \;\; j = 0, 1, \ldots, l-1;$$

the input by:

$$I = [i_0, i_1, \ldots, i_{v-1}], \;\; i_j \in \{0,1\}, \;\; j = 0, 1, \ldots, v-1;$$

and the output by:

$$Z = [z_0, z_1, \ldots, z_{r-1}], \;\; z_j \in \{0,1\}, \;\; j = 0, 1, \ldots, r-1$$

Since inputs, outputs, and states only change on the rising (falling) edge of the clock pulse, they each form a discrete series. Let k represent the index for the series, where $k = 0$ corresponds to when the state machine starts, k = 1 to the state machine $1 \cdot T$ seconds later, and so on. Thus, the state variables form a series:

$$Q(k), \;\; k = 0, 1, 2, 3, \ldots \quad \text{where } Q(k) = [q_0(k), q_1(k), \ldots, q_{l-1}(k)].$$

with $Q(0) = Q_{Res}$. The input and output vectors form similar series.

The new input, $I(k+1)$ is simply the sampled value of the input at the $(k+1)^{th}$ clock pulse. The CMSSM doesn't model how the input changes, but simply makes decisions based on its last state and the new input.

The next value of Q is determined by the next state function, Ω, defined as follows:

$$\Omega : \; Q \,\mathrm{x}\, I \rightarrow Q; \;\; Q(k+1) = \Omega(Q(k), I(k+1))$$

The current output is determined by the output map, Φ, defined as follows:

$$\Phi : \; Q \rightarrow Z; \;\; Z(k+1) = \Phi(Q(k+1))$$

4.3 Translation Method

In this section, we define a translation method to convert a deterministic DES supervisor into a CMSSM. We present both a centralized and modular implementation. First, we state a few definitions.

Let the plant model be defined as: $G = (Y, \Sigma, \eta, Y_0, Y_m), \;\; \Sigma = \Sigma_u \cup \Sigma_c$
Let S_1, S_2, ..., S_n be n supervisors for G, with

$$S_j = (X_j, \Sigma, \xi_j, X_{0_j}, X_{m_j}), \;\; j = 1, 2, \ldots n$$

Let $S = \text{meet}(S_1, S_2, \ldots, S_n) = (X, \Sigma, \xi, X_0, X_m)$

4.3.1 Centralized Implementation

For the centralized implementation, the CMSSM will be defined with respect to the composite supervisor, S.

Define the state size of Q to satisfy $2^l \geq \mid X \mid$. Define an arbitrary injective map, $\lambda : X \rightarrow Q$. The initial state is defined to be $Q_{Res} = \lambda(X_0)$.

Define the state size of I to be $v = \mid \Sigma \mid$. Define an arbitrary bijective map, $\gamma : \Sigma \rightarrow \{i_0, i_1, \ldots, i_{v-1}\}$. Then define an injective map, $\Gamma : \Sigma \rightarrow I$ where:

$$\Gamma(\sigma) = [0 \cdots 0 \underset{j}{1} 0 \cdots 0] \text{ for } \sigma \in \Sigma \text{ and } \gamma(\sigma) = i_j$$

Define the state size of Z to be $r = \mid \Sigma_c \mid$. Define an arbitrary bijective map, $\delta : \Sigma_c \rightarrow \{z_0, z_1, \ \ldots \ , z_{r-1}\}$.

The interpretation of the input variables (i_0, i_1 etc.) is that the CMSSM senses the occurrence of an event σ when its corresponding input $\gamma(\sigma)$ is true (=1). The interpretation of the output variables (z_0, z_1 etc.) is that a controllable event, $\sigma \in \Sigma_c$, is enabled when its corresponding output $\delta(\sigma)$ is true. Otherwise, the event is disabled.

The next state function, Ω, is defined in accordance with the supervisor. For $x \in X$ and $\sigma \in \Sigma$,

$$\text{if } \ \xi(x,\sigma)! \ \text{ then } \ \ \Omega(\lambda(x), \Gamma(\sigma)) = \lambda(\xi(x,\sigma)) \ \text{ else } \ \Omega(\lambda(x), \Gamma(\sigma)) = \lambda(x)$$

All remaining values of Ω are assigned arbitrarily.

The output map, Φ, is similarly based on the supervisor. For $x \in X$ and $\sigma \in \Sigma_c$,

$$\text{if } \ \xi(x,\sigma)! \ \text{ then } \ \delta(\sigma) = 1 \text{ for the state } \lambda(x). \text{ Thus, } \Phi(\lambda(x)) = [\cdots \underset{j}{1} \cdots] \text{ for } \delta(\sigma) = z_j$$

$$\text{otherwise } \ \delta(\sigma) = 0 \text{ and } \Phi(\lambda(x)) = [\cdots \underset{j}{0} \cdots] \text{ for } \delta(\sigma) = z_j$$

4.3.2 Modular Implementation

For simplicity, we define the modular implementation for the case of two modular DES supervisors, S_1 and S_2. The definition can easily be extended to n supervisors.

First, the two supervisors are converted to centralized CMSSM as above. But, instead of actually controlling the system's outputs, they generate 'local' outputs. Thus we have the following controllers:

$$H_1 = (I, Z_1, Q_1, \Omega_1, \Phi_1, Q_{Res_1}, T) \quad \text{and} \quad H_2 = (I, Z_2, Q_2, \Omega_2, \Phi_2, Q_{Res_2}, T)$$

The Composite Controller, H_{mod}, is then implemented by taking the conjunction of the two modular controllers' outputs.

We then define the following maps:

i) $\zeta = \zeta_1 \wedge \zeta_2 \quad (X_1 \text{ x } X_2) \ = \ \zeta_1(X_1) \wedge \zeta_2(X_2)$
ii) $\Omega = \Omega_1 \text{ x } \Omega_2 \ (Q_1 \text{ x } Q_2) \ = \ \Omega_1(Q_1) \text{ x } \Omega_2(Q_2)$
iii) $\Phi = \Phi_1 \wedge \Phi_2 \ (Q_1 \text{ x } Q_2) \ = \ \Phi_1(Q_1) \wedge \Phi_2(Q_2)$

Thus, the controller is defined: $H_{mod} = (I, Z_1 \wedge Z_2, Q_1 \text{ x } Q_2, \Omega, \Phi, Q_{Res_1} \text{ x } Q_{Res_2}, T)$

4.4 Control Equivalence

For our purposes, we are only interested in whether a CMSSM will enforce the same control action as the original DES supervisor. The effects of transmission delay and concurrent events are not considered. For a discussion on these subjects, refer to [1], [5] and [7]. To aid in the following discussion, some definitions are useful.

Definition: A DES supervisor and a CMSSM are control equivalent if they both take the same control action based on any sequence of events generated by the plant.

Define $\zeta : X \rightarrow Z$, to map the control action specified in the supervisor at a given state to the same representation used by the CMSSM.

4.4.1 Centralized Equivalence

For centralized equivalence we have the following:

Theorem 1 *If S is controllable for plant G then the centralized implementation, H, is control equivalent to S.*

To prove Theorem 1, it is sufficient to show that Fig. 7 commutes. It is clear from the definitions of λ, Γ, Ω, Φ, and ζ that this is in fact the case.

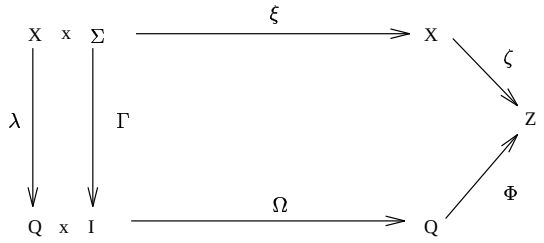

Figure 7: Control Equivalence Diagram

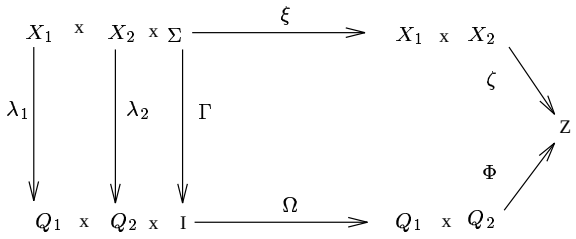

Figure 8: Modular Control Equivalence Diagram

4.4.2 Modular Equivalence

For modular equivalence, we present the following theorem:

Theorem 2 *If S_1 and S_2 are controllable for plant G then the modular implementation, H_{mod}, is control equivalent to $S_1 \wedge S_2$.*

To prove Theorem 2, it is sufficient to show that Fig. 8 commutes, namely whenever the upper path through the diagram is defined, then so is the lower, and yields the same results. This follows directly from the definitions of the relevant maps. This result can easily be extended to n supervisors.

4.5 Implementation Details

In this section we discuss the implementation of the synchronous state machine's mapping functions, and the execution algorithm used to implement the CMSSM.

For the state machine, both the next state function (Ω) and the output map (Φ) are implemented in a modular manner. The next state function is implemented as l boolean logic functions. Thus,

$$\Omega(Q(k), I(k+1)) = [\omega_0(Q(k), I(k+1)), \ \ldots \ , \omega_{l-1}(Q(k), I(k+1))]$$

Similarly, the output map is implemented as r boolean logic functions. Thus,

$$\Phi(Q(k+1)) = [\phi_0(Q(k+1)), \ \ldots \ , \phi_{r-1}(Q(k+1))]$$

The execution algorithm used to implement the CMSSM is as follows:

1. Initialize state machine by setting $Q(0) = Q_{Res}$, $Z(0) = \Phi(Q_{Res})$and $k = 0$.
2. Wait for next clock pulse $(k+1)$. Sample inputs and set $I(k+1)$ to these values.
3. Compute $Q(k+1) = \Omega(Q(k), I(k+1))$ and $Z(k+1) = \Phi(Q(k+1)$.
4. Set $k = k+1$ and then go to step 2.

4.6 PLC Interpretation

A programmable logic controller is similar to a microcomputer, but specialized for control purposes. The PLC was chosen for the physical implementation of the synchronous state machine because of its robustness, and its wide use in industry.

The CMSSM will be implemented on the PLC in the form of a relay ladder logic (RLL) program. A RLL program is made up of several steps called rungs. A rung is composed of a logical evaluation statement and an output assignment statement. If the logic evaluates to true, then the output is energized. The basic components of a rung are shown in Fig. 9.

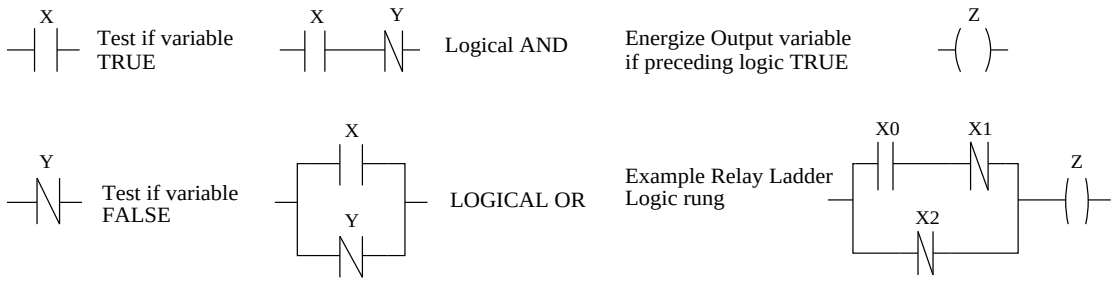

Figure 9: Components of Relay Ladder Logic

The execution algorithm given in Section 4.5 will be translated into a RLL program. The code will have five main parts: initialize variables, evaluate next state logic, assign temporary outputs, set new states to be current states, and combine temporary outputs to generate system outputs.

4.7 PLC Controllers

The 29 DES supervisors were implemented using the modular method outlined in Section 4.3.2. As an example, we illustrate the implementation of supervisor *sup_crane1* (Fig. 5). The CMSSM representation is shown in Fig. 10. The reset state, next state function, and the local output map are defined below. Fig. 11 shows the equivalent Relay Ladder Logic for computing the next state of the state variables. The formulas below only show inputs that are relevant to deciding the next state, and only those outputs that the controller actually disables.

$Qa_{Res} = [0, 0]$

$$\begin{aligned}\omega_1(Qa(k), Ia(k+1)) &= [\neg qa_1(k) \wedge \neg qa_0(k) \wedge \text{t2\_at13(k+1)}] \vee [qa_1(k) \\ &\quad \wedge \neg \text{f\_ld1(k+1)}] \\ \omega_0(Qa(k), Ia(k+1)) &= [\neg qa_1(k) \wedge \neg qa_0(k) \wedge \text{t1\_at13(k+1)}] \vee [qa_0(k) \\ &\quad \wedge \neg \text{f\_ld1(k+1)}]\end{aligned}$$

$$\phi_2(Qa(k+1)) = \neg qa_1(k+1) \qquad \phi_1(Qa(k+1)) = \neg qa_0(k+1)$$
$$\phi_0(Qa(k+1)) = qa_1(k+1) \vee qa_0(k+1)$$

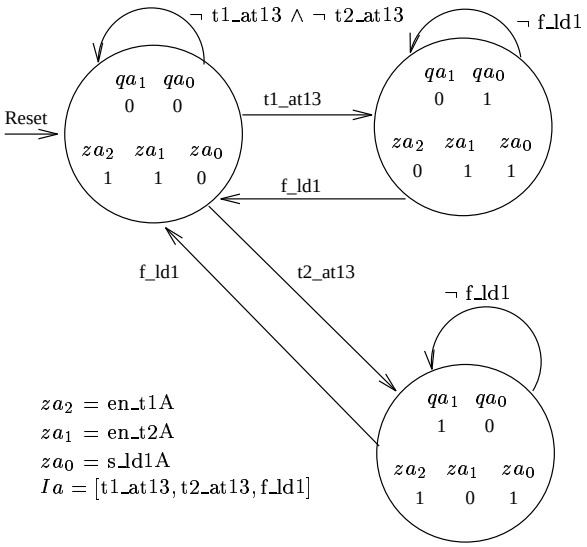

Figure 10: CMSSM for *sup_crane1*

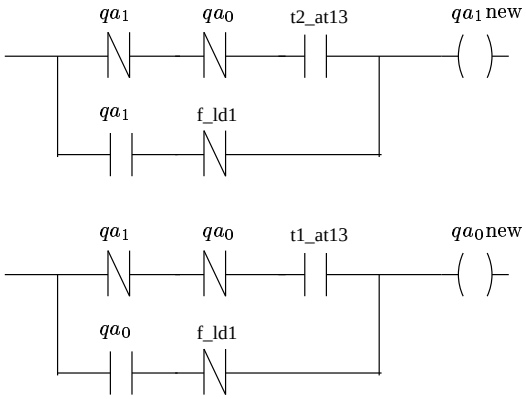

Figure 11: RLL for Next State Equations

4.8 Discussion of Implementation Results

Each DES supervisor has been implemented as a synchronous state machine on the Allen-Bradley PLC. An interrupt-driven, discrete interface has also been implemented on each of the embedded MC68332 processors for communicating with the PLC. All PLC controllers have been tested and are operational.

5 Conclusion

In conclusion, we have designed and built a PLC based testbed for investigating the implementation of DES supervisors. The system was modeled and several RW modular supervisors were designed and verified for controllability.

A method to implement DES supervisors as clocked Moore synchronous state machines has been presented and shown to be an equivalent representation. Also, a method to implement supervisors modularly has been defined and shown to be equivalent to implementing a centralized supervisor.

Finally, all 29 DES supervisors have been translated into CMSSM and implemented on a PLC.

References

[1] S. Balemi. Input/output discrete event processes and communication delays. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):41–85, 1994.

[2] B. A. Brandin. *Real-Time Supervisory Control of Automated Manufacturing Systems.* Ph.D thesis, Department of Electrical Engineering, University of Toronto, 1993.

[3] B.A. Brandin, W.M. Wonham, and B. Benhabib. Discrete-event systems supervisory control applied to the management of manufacturing workcells. In *7th Inter-*

national Conference on Computer Aided Manufacturing Engineering, pp. 527–536, Cookeville, Elsevier, 1991.

[4] R.J. Leduc and W.M. Wonham. Discrete event systems modeling and control of a manufacturing testbed. In *Canadian Conference on Electrical and Computer Engineering*, volume 2, pp. 793–796, Sept 1995.

[5] Yong Li. Supervisory Control of Real-Time Discrete-Event Systems. M.A.Sc thesis, Department of Electrical Engineering, University of Toronto, Toronto, ONT, 1986.

[6] G. Michel. *Programmable Logic Controllers- Architecture and Applications.* Wiley, 1990.

[7] T.- M. Pai. Real-Time Implementation of Discrete-Event Controllers. M.A.Sc thesis, Department of Electrical Engineering, University of Toronto, Toronto, ONT, 1990.

[8] P. Ramadge and W. Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control And Optimization*, 25(1):206–230, 1987.

[9] J. Stenerson. *Fundamentals of Programmable Logic Controllers, Sensors, and Communications.* Prentice Hall, 1993.

[10] R. Strobel and A. Johnson. Pocket pagers in lots of one. *IEEE Spectrum*, pp. 29–32, Sept 1993.

[11] J. Wakerley. *Digital Design Principles.* Prentice-Hall, 1990.

[12] W. Wonham and P. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.