Petri net representation of relay ladder logic for programmable controllers

by

Douglas K. Hyde

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Iowa State University
Ames, Iowa
1989

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Programmable logic controllers (PLC) and programmable logic devices (PLD) have progressed rapidly since their inception in the 1960s and represent a first step towards automation for many manufacturing facilities. Their initial programming systems involved several methods of implementation or coding including relay ladder logic (RLL), digital logic symbols, boolean logic symbols, and high level language coding. A new concept for control programming is beginning to appear in manufacturing. Petri nets (PN) have been demonstrated as effective tools in the design and control of PLC/PLD systems and also have been used to model manufacturing systems at levels of control from production flow to individual machine control. The use of Petri nets as a language for logic representation at all levels of manufacturing and production planning would help to speed the planning and implementation times for a manufacturing system by reducing the number of different languages used by the implementation team.

In the development of a production system many computer languages are used to develop, test, and implement the system software components. Each language has a distinct programming style, syntax, and structure. A lack of continuity between these many systems makes communication, documentation and validation difficult. In a manufacturing setting one might model the production line with GPSS, implement conveyor control with RLL, program the robotics and the flexible manufacturing system (FMS) with other languages, and document the systems interaction with graphics and word processing, then tie elements of the system together inside another programming environment. As the project proceeds from one task to another, the shift of responsibility for implementation can result in a loss of information, inadvertent change in the focus of the project, or complications in the documentation and implementation of the system. These diverse systems of logic need a common base for design and implementation.

Petri nets provide a common base of communications. Alanche et al. [2] stated:

> "...(if) ...graphical tools such as petri nets are more and more used for the design, the evaluation and the implementation of the control system, a better understanding within the project team will result."

The ability to use one approach to logic and programming through all of these design and implementation phases simplifies and speeds the process of testing, debugging, and documentation yielding a reduction in start up time [9]. Due to the diverse nature of the manufacturing environment, this will not be entirely successful.

Petri nets can be used as a tool for representation, analysis and synthesis. Some applications include operating systems, compilers, communications, speed independent hardware, propositional logic, mathematics, computer hardware and production systems [1, 38]. Petri nets have been applied to manufacturing environments at the level of whole systems, FMSs, and controllers [4, 29, 33].

Ladder logic as a programming tool is successful because the person wiring the wall mounted systems could program the new PLC systems using electrical wiring symbols. Remembering and understanding the relay ladder logic following a long absence from the logic is difficult [9]. The programming of PLC/PLD systems is coming under the control of programmers and systems people and not the "electrician". The present literature indicates that PN based controllers may eventually replace RLL systems. Realistically, RLL will endure for some time in islands of automation and small manufacturing installations [41].

The ability to test, validate and document a system before implementation has not progressed as fast as the control capability of PLC/PLD systems. Validation using simulation sometimes fails to identify potential states and evaluate the system in their presence. The programmer writes the simulation for what is expected rather than the real potential of event occurrence which exists. Failure to check all of the states or potential markings of a system for these "unforeseen" conditions could result in PLC logic which operates incorrectly causing costly implementation delays and perhaps damage or injury. Petri nets can be evaluated for all potential system states using a reachability tree. PN can be used to test and document a system before implementation.

Manufacturing automation's shift from the factory floor to systems design groups, the desire to reduce the number of languages used in developing the automation, the continued use and growth of RLL based PLC/PLD devices in industry, and the need to document and test RLL before implementation combine with Petri nets growth as a language in manufacturing planning to point out a need for generating PN logic from RLL to improve the communications, testing, debugging, and implementation of processes.

CHAPTER 2. LITERATURE REVIEW

Petri Nets

A summary of work on Petri nets and modifications for manufacturing follows.

History

Dr. Carl Adam Petri's dissertation in 1962 [39] formulated the basis for a theory of communication between synchronous components of a computer system. "He was particularly concerned with the description of the causal relationships between events" [38]. This work began the development of Petri nets into the large body of research and development existing today.[1] The basis of Petri nets is to model graphically and test analytically the discrete events of concurrent operations within a system. One goal of the research is to achieve the ability, using Petri nets, to visualize, analyze, and validate a discrete system of any size.

Desirable traits and advantages of Petri nets

Kamath and Viswanadham [19] listed five positive aspects of Petri nets.

[1] Peterson [38, pp. 3 and 4] gives a developmental path.

1. "They describe the modelled system graphically and hence enable an easy visualization of complex systems,
2. Petri nets can model a system hierarchically; systems can be represented in a top-down fashion at various levels of abstraction and detail,
3. A systematic and complete qualitative analysis of the system is possible by well-developed Petri net analysis techniques,
4. The existence of well formulated schemes for Petri net synthesis facilitates system design and synthesis, and
5. Performance evaluation of systems is possible using timed Petri nets."

Peterson [37] elaborated further:

> "Petri nets have been receiving increased attention as a model of parallel computation. In large part this is due to the simplicity of the Petri net model coupled with a careful balance of *modeling power* and *decision power*. The modeling power of Petri nets is quite good, as witnessed by the wide variety of systems which can be modelled by Petri nets. The decision power is also good, since the reachability problem is decidable, and most problems can be converted into reachability problems."

Agerwala [1] states that the advantages of Petri nets include the ability to model at every level of the system which most other design languages cannot do.

Disadvantages of Petri nets

The complexity of analysis grows as the model grows. Model size or complexity must be balanced with desired decision power. Peterson [37] summarized some of the disadvantages:

> "... concurrency of operation has become more and more common. This has generally improved utilization and throughput, but at a consequent increase in complexity."

A balance between modeling and decision power must be struck. Peterson concludes with:

> "Subclasses of Petri nets increase the decision power, but at a cost of being unable to model a large number of systems. Extended Petri net models increase the modeling power, but in all known cases at the expense of decision power, since most analysis questions become undecidable."

There are still some areas and events which Petri nets can not model [38, pp. 190 through 195].

Graphic representation and dynamics of Petri nets

Graphic Representation Petri nets use three basic components: places, transitions, and arcs.

Places represent states of the system components; transitions are events; and arcs are either inputs or outputs of transitions. Graphically, Petri nets are represented by circles (places), bars (transitions) and unidirectional vectors (arcs). An example of

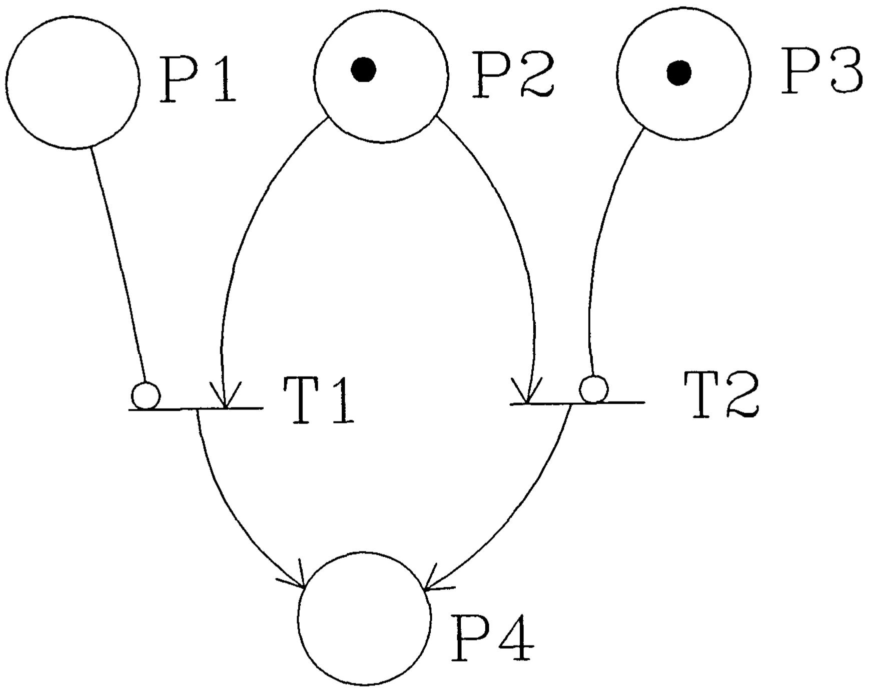


Figure 2.1: Places, transitions, arcs, and tokens of a Petri net

a Petri net in Figure 2.1 demonstrates these three components, shows tokens marking places, and inhibitor arcs used for enabling a transition requiring an absence of markings in a place.

Transitions are the events changing the state of the system. When a transition fires, the system changes state (i.e., from idle to active, available to unavailable). In some models the bars are replaced by boxes representing transition delays or processing time [4].

Places are connected to transitions (and vice versa) by directed arcs [37]. They control the unidirectional flow of tokens between places and transitions. Some Net models are subsystems of larger models so may have an entry place without a feeder arc or a completion place without a departure arc.

> "An extension of Petri nets replaces the arrow head by a small circle creating an inhibitor arc to change the transition firing rule. A transition is enabled when tokens are in all of its normal inputs and *no* tokens are in its inhibitor inputs. This notation is borrowed from switching theory where the small circle means 'not'" [38].

Tokens are dots contained within the places (circles). Tokens are used to define the execution of a Petri net [38, page 16]. There can be one or more tokens in a place representing multiple available resources at that state or the units having reached that state in the system; the ability to generate only a finite number of tokens at any place in the modeled system denotes a bounded Petri net.

Dynamic Behavior The dynamic behavior of a petri net is described by Kamath and Viswanadham [19] as follows:

> "The dynamic behavior of a system is modeled as follows. The occurrence of an event is represented by the *firing* of the corresponding transition. The movement of tokens in the net resulting from the firing of one or more transitions represents a change in the system state. The following are the firing rules for marked Petri nets.
>
> 1. A transition is *enabled* when each of its input places contains at least one token.
> 2. A transition can fire only if it is enabled.
> 3. When a transition fires:
>   - a token is removed from each of its input places, and

- a token is deposited into each of its output places".

A more complete discussion of the dynamics of Petri nets can be found in Peterson [38] or Reisig [40].

Testing of Petri nets

A significant advantage to using Petri nets versus other modeling systems is the ability to test and validate a system. Liveness, boundedness, safeness, and reachability are measures of effectiveness for the Petri net.

Deadlock occurs when a transition cannot fire and no sequence of transition firings will take the net to a marking which allows the transition to fire. A Petri net is live if there is no deadlock [37, 38].

A reachability tree is generated from an initial marking by firing enabled transitions. Reisig [40] and Peterson [37, 38] discuss this in detail. Essentially, if the reachability tree shows no infinite markings (places containing or having the potential to contain an infinite number of tokens) then the tree is bounded and safe. The reachability tree is a finite representation of the usually infinite reachability set from an initial marking of the Petri net.

The reachability problem deals with the ability to reach a marking from an initial marking. Peterson [37, page 147] states:

> "Recent results seem to indicate that the reachability problem is solvable, but it is extremely hard. Thus although the problem can be solved, it may take much too much time and money to be worthwhile, in general. Other problems, such as the equality of the reachability sets of two Petri

nets (useful for considering equivalence and optimization) are known to be unsolvable."

Extensions to Petri nets

There have been many advancements and modifications to the basic Petri net theory. Not all authors and researchers have agreed on the direction of the "improvements" so several branches of Petri net theory have developed. Some have added features by extending basic Petri net concepts. Examples include timed transitions, levels of net operation to reduce repetition, delayed transitions to simulate processing, delayed arcs, and multiple token place representations. The following sections discuss some of the more prevalent extensions to Petri nets and their applications.

Extensions to Petri nets have enhanced or altered the power of these models. Some of the changes are colored Petri nets, timed Petri nets, and stochastic Petri nets.

Colored Petri nets Colored Petri nets (CPN) allow the modeler of systems with repetitive processes to view a smaller network in which tokens have changed colors to: indicate process steps, assign attributes, or differentiate between tokens. The primary function of CPN is data management. The structure of the PN systems are not affected nor are the reachability trees or analysis questions. The color of the tokens is just another data item carried in the markings. The colors represent levels of activity or number of times the part has moved through the process. This model concept is also useful when several parts must be processed through the same system. An example would be in electronic chip manufacture where wafers being fabricated pass through five basic processes many time adding layers of new material onto existing layers. The processes are represented by states, the pass number is represented by the color of the token, and the transition represents the movement between processes.

Timed Petri nets Timed Petri nets introduce timing to a network. The manufacturing area of interest in Petri net analysis would be limited if the timing of a network were not possible. By adding time to the place or transition (two schools of thought diverge here) the system changes from instantaneous firings to one of delayed firing. This can affect the network from several standpoints including availability of resources, timing of future events, and overall completion time for activities.

Control nets Control nets (C nets) are a modified form of Petri net (P,T,I,O) where I and O represent input and output arcs by Murata, Komoda, and Matsumoto [33]. They called the 4-tuple a single element C, included the marking matrix M and 5 new elements representing various process functions (δ, φ, η) (command, response, and gate functions respectively) and system status functions (U,V) (execution status, and transition status respectively) to produce a 7-tuple (C, δ, φ, η,U,V,M) to control a factory automation controller. The process functions are used to allow an operator direct control of token movement in the system. The system status functions allows supervision of the execution status and management of the transition statuses. This is an example of modelling enhancements quickly limiting the decision and analysis attributes of Petri nets.

Applications of Petri Nets

Applications of Petri nets in the manufacturing setting are varied. The usefulness of PN logic in systems design and documentation is demonstrated quite often. There are efforts in simulation, testing, analysis, design, and control using PN systems. Implementations at every level of control has been demonstrated. Simulation systems using Petri nets at the levels of factory, production line, cell, and controller exist. Design of systems using Petri nets have been automated. Systems have been demonstrated which develop RLL from PN graphics and test PN designs before implementation. Three PN based controllers have been developed and demonstrated. PN as a basis of documentation and control is becoming more prevalent and will continue to demonstrate its applicability.

The dissertation in 1962 by Petri [39] came to the attention of Project MAC at MIT which has published numerous reports and dissertations. One of these reports, by M. Hack [15] in 1972, brought together the many facets of the production environment and provided a broad approach to manufacturing systems overview and design in light of Petri nets. Most importantly, the work describes the use of free choice Petri nets and their definitions. This work puts together the nuts and bolts of production schemata and is the seminal work relating Petri nets directly to a manufacturing environment. Since that time work has branched into several different areas of study. Primarily, there are 3 levels of production control:

- Manufacturing Systems
- Flexible Manufacturing Systems or Cells

- Programmable Controllers or Controller based Sequencers

The study in all three of these areas falls into 4 categories:

- Design and Documentation
- Simulation and Modeling
- Analysis and Testing
- Control

Design and documentation

Houldsworth and Brearly [18] state that PLC functions can be enhanced by using a Petri net like programming system and a structured method of programming. They maintain that good methods and a graphical approach to programming are used effectively with a personal computer to provide good documentation of the system. The system they promoted is GRAPH5 by Siemens. Brand and Kopainsky [6] and Bruno and Marchetto [7] cite Petri nets as the basis for process control design. Gentina and Corbeel [14] propose a modified net system for the synthesis of FMS control. Devanathan et al. [13] promote computer aided design of RLL using state diagrams. Krogh and Beck [23] investigate the use of nets for simulation of manufacturing systems and Krogh, Willson and Pathak [24] propose using Petri nets and microcomputers to generate programming for PLCs keeping a data base of used and available data points. Their work allows the retention of RLL but keeps the programming level at a higher level language. Kruempel and Day [25] developed a CAD package for designing RLL for PLCs as well as a simulation system they test against a Modicon controller. Lloyd [27] discusses GRAFCET, a program which uses Petri nets as a basis for PLC programming. Like the GRAPH5 program of Siemens, they promote a function block approach to states and transitions. The elemental functionality of PNs are not considered or demonstrated, only their higher order modeling is used. The function blocks and transitions are evaluated or fired based on RLL written at a lower level. Only the PN logic is displayed. Currently Siemens (GRAPH5), Telemecanique (GRAFCET), and Allen-Bradley (unknown) offer PN-based graphic programming modes in addition to ladder logic and high level languages [41]. Martinez and Silva [30, 31] develop a language for describing concurrent systems such as FMSs and a package for computer design of concurrent logic systems; with Alla [28] they discuss using Petri nets for specification of FMS. Matsuzaki et al. [32] use a GRAFCET-like Petri structured programming system similar to BASIC and report increases in software productivity of 50% to 100% over Relay Ladder Logic, as did Murata, Komoda, and Matsumoto [33] who reported a 50% man hour reduction in software development time from RLL using their PN oriented language.

Manufacturing simulation and modeling using Petri nets

Han [16] not only proposes the specification of the system using Petri nets, he also demonstrates the simulation, modeling and queuing analysis capabilities of the nets. Silva and Velilla [42] investigate the various computer resource consumptions using PLC vs Petri Net based controllers. Alanche et al. [2] presents a Petri Net based Simulator (PSI) to evaluate flexible manufacturing systems. Alla et al. [3] develops some of the mathematics of analysis of FMS using CPN (see section 2).

Narahari and Viswanadham [35] use Petri Nets to analyze FMS, and conclude that simple PN will not suffice for larger systems. Viswanadham works with Kamath [19] to develop the Colored Petri net analysis of an FMS. Beck and Krogh [5] citing Narahari and Viswanadham present modified Petri nets for simulation and control of an FMS. Bruno and Morisio [8] develop PN based simulation of manufacturing cells using some earlier work on PN for the design of FMS. Martinez, Muro, and Silva [29] expand previous work to demonstrate the modeling and validation capabilities of PN in production systems modeling. Peng and Shin [36] who did not deal directly with manufacturing systems develop some interesting concepts of modeling distributed systems for real time control.

Analysis and testing

Hack's [15] seminal work includes a basis for analysis and testing of PN systems. Heimerdinger [17] uses PN systems to analyze fault tolerance at a system level. LeMer [26] develops a software package for the analysis of PN systems which verifies and validates Petri nets. Narahari and Viswanadham's [35] work on PN systems includes the analysis needs of the system; as does Martinez et al. [29].

Control

Chocron and Cerny [10] provide the first example of implementation of a PN based controller (called a sequencer in their work). This example is paralleled by the work of Courvoisier et al. [11, 43] in their development of a PN based controller on a Z80 based computer. The implementation is slow due to the CPU speed and memory size. Advances in semiconductors is expected to improve the processing time. Murata et al. [33, 22, 34] present work on a PN based controller implementation on a larger CPU controlling a FMS cell. Their findings indicate good control and quick comprehension of the graphics used in the control monitor allowing rapid fault detection and repair with increases in software productivity of 50% over the RLL systems. Crockett et al. [12] expands upon all of the previous work to implement a PN based controller at a higher level of control generating software control commands used to interact with hardware signals.

CHAPTER 3. PROPOSAL

The need for documentation and testing capabilities on Relay Ladder Logic used in programmable controllers has importance in the future as islands of automation are incorporated into larger automated systems. The initial control logic was developed for use by electronic technicians, floor foreman, plant maintenance personnel, or electricians who wired the equipment. This level of control has yet to be integrated with larger automation systems. The RLL serves well as a communication system of logic for earlier programmers, but systems people need to consider integration of subsystems using a common control environment.

Replacement of Relay Ladder Logic at its lowest levels of use will be slow. Petri net based controllers have been considered, and some high level petri net systems are offered but their acceptance is far down the road. RLL is an acceptable and proven method of logic development. Its roots in the electrical magnetic relay and switch system make comprehension easy for maintenance and repair teams who work around the equipment. The difficulty of communications between floor personnel who are not well versed in the logic design languages and systems designers who are not well versed in RLL or electrical symbols will cause an increase in the cost and time to implement a system.

Petri nets have been used for designing and developing logic for manufacturing and flexible manufacturing control. Their future use as a documentation and logic development tool for systems has a high probability. They have the potential to become a base language of manufacturing systems design in the future.

Computer Integrated Manufacturing (CIM) and system design is becoming more important at every manufacturing site. Integrating and reducing the number of languages which must be worked with in a CIM project is important. Logic and systems planning using one common language is the ideal solution. Although the ideal usually is not reality, Petri nets have the potential to be that common integration language used in CIM.

Since relay ladder logic is entrenched in the current manufacturing systems and will remain so for some time in the future, there is a need to develop a methodology to provide a Petri net view of the RLL logic for analysis and documentation. In light of this need, this thesis proposal is threefold.

- To develop RLL test models demonstrating basic control logic.
- To determine the generic steps required to convert RLL models to Petri net models.
- To demonstrate that the Petri net and RLL models perform exactly alike.

This work will provide a conceptual model and working algorithm for converting RLL to Petri nets providing future systems designers with a common language for design and documentation, and allowing future designs in RLL to be converted to Petri nets for testing, validation, and debug before implementation on the factory floor.

CHAPTER 4. LOGIC MODELS

Components found in most RLL systems as shown in Figure 4.1 include:

- Normally open switch,
- Normally closed switch, and
- Coil or Relay,

The proposed architecture consists of two models, namely, the processing model and RLL model. PLCs process RLL diagrams according to a predefined algorithm embedded in the PLC system. This processing is the basis of the processing model and must be integrated into the PN models for a valid model of RLL execution. Two RLL models investigated in this research are a serial and a parallel ladder using switch and relay elements. These models are representative of simple RLL systems.

PLC Processing of RLL

The PLC processing model gathers outside inputs for the relay ladder into *current* registers. Each rung is evaluated sequentially based on the current register values. The results of the evaluation are placed in *future* registers used to update the *current* output registers after the ladder scan is completed. During a ladder scan no state

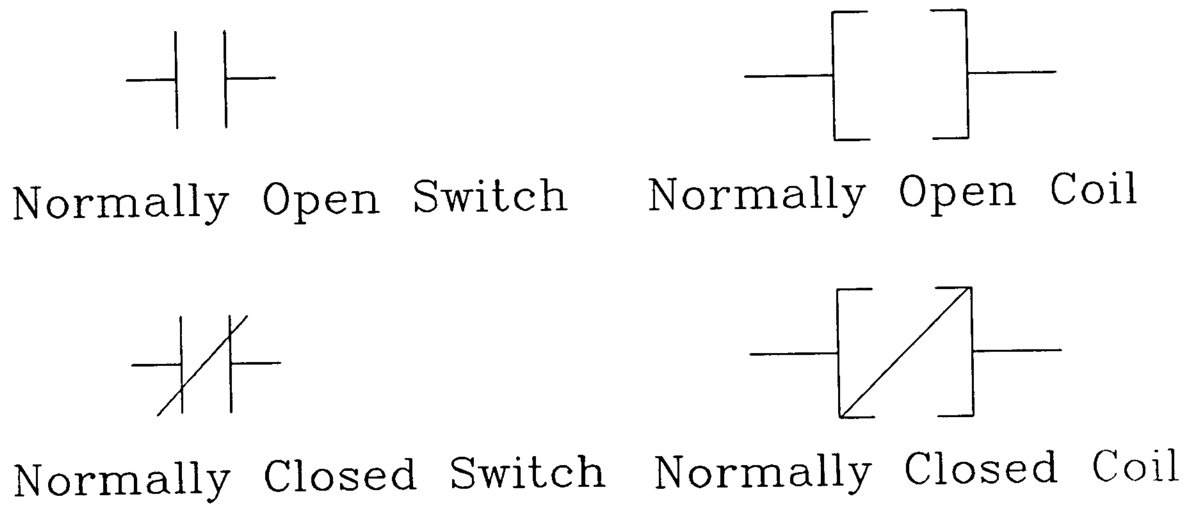


Figure 4.1: Common components of relay ladder logic diagrams

change in earlier rungs output affects the rungs following. The ladder scan speeds make the processes appear to be continuous but RLL based PLC's are all sequential evaluation systems.

Serial Ladder Model

A three rung serial ladder is illustrated in Figure 4.2. Rung 1 of this model is a start/stop rung. Outputs are represented by coils and inputs by switches. The $Y1$ input is driven by the $Y1$ coil and represents both internal and external control by the $Y1$ coil. The relay $Y1$ is activated by the start switch $X1$ closing and the stop switch $X2$ remaining closed (i.e., no outside actions). Once the relay $Y1$ is activated, the start switch is released and the contact controlled switch $Y1$ maintains the circuit. This rung should not be confused with the hard wire stop switch connected to the PLC to stop processing when the CPU fails. This rung appears in our parallel model also and serves as an example of a Boolean OR decision module. The outcome of

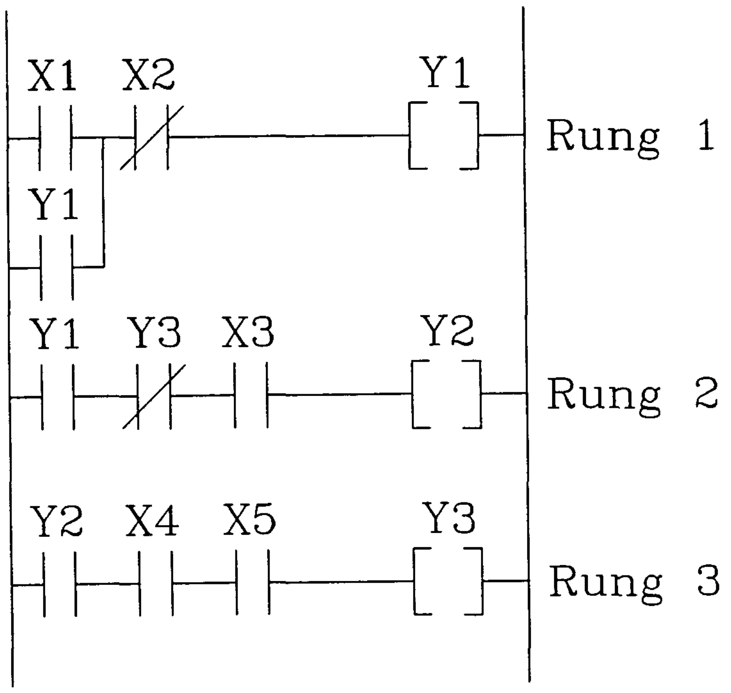


Figure 4.2: Serial relay ladder logic model

this rung may be implemented as a switch in all succeeding rungs to stop all related processing. In our serial model, this dependence has been removed to gain simplicity in our models. Several start/stop rungs may appear in a complex PLC to control separate manufacturing lines or processes.

Rung 2 represents a sequence rung using rung 1 output to activate output $Y2$. Another function employed in the model is the use of "exclusive or" switching. Rung 3 output, $Y3$, controls the actions of rung 2 and vice versa. An example of the use of such a rung pair would be a robot release command. The release command cannot be given until the robot is activated in rung 2 and has moved to a position determined by switches $X4$ and $X5$. Once the robot has reached this position, the output $Y3$ triggers the part release mechanism turns off the $Y2$ output and releases the robot for other tasks. Output $Y3$ may initiate further activity on the released part.

Inputs from outside sensors such as contact closures and photodetectors are represented by the X values. Output to the outside world is represented by the Y notation above the square bracket coil elements, and internal control of rungs is represented by the Y notation above switch elements. This representation is not meant to follow any one PLC system, but to represent the logic of basic PLC RLL notation.

The RLL rungs are processed from top to bottom. The activation of the next rung depends on the previous rung's status. Ladder scanning systems working on this type of dependency require at least n passes through the RLL to activate/deactivate each rung of the ladder where n is the number of rungs in the RLL model. More than n passes are necessary when there is increased dependency upon previous rungs in the model. A ladder scan does not pass the activation of rung 1 to rung 2 until the output of rung 1 is updated. This updating occurs at the end of a ladder scan, so the activation of rung 1 in the first scan can not be reported to the input of rung 2 or successive dependent rungs until the update phase following the scanning of all the rungs. A discussion of PLC rung scanning systems in [20, pages 124-126] reveals that output evaluation occurs after ladder scan completion regardless of the method used to scan the ladder rungs. The rung scanning process used in this paper is rung by rung.

PLC ladder processing is serial in nature, automatically evaluating each rung before testing the next rung. Although current PLCs process each rung in a sequential manner, the current processor speeds give an appearance of simultaneous occurrence of events.

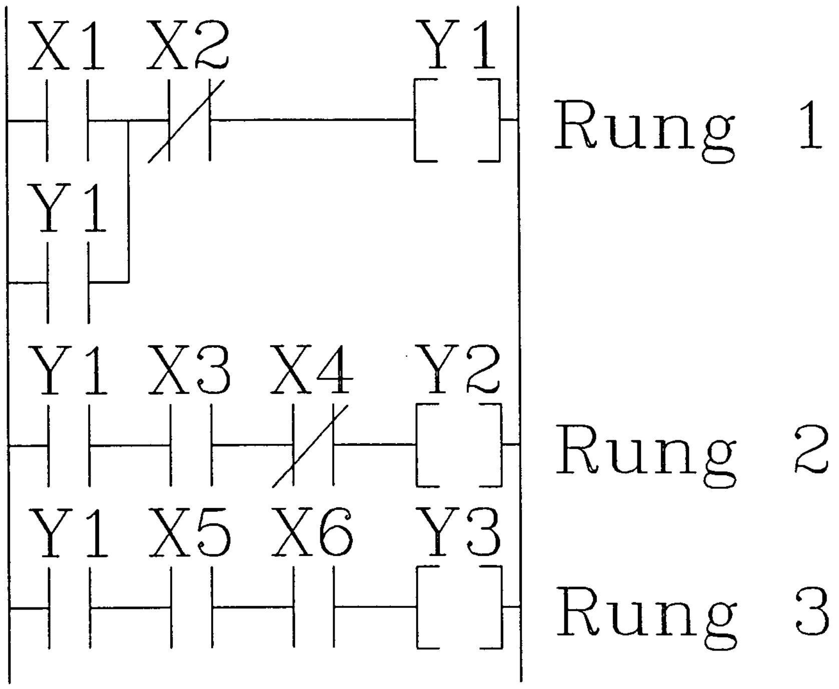


Figure 4.3: Parallel relay ladder logic model

Parallel Ladder Model

The model in Figure 4.3 also begins with the start/stop rung. The purpose of this model is to represent in RLL what can be processed simultaneously. In the wall mounted relay technology, these processes are performed at the same time and are not dependent upon each other. The serial scanning of ladder rungs causes the RLL based PLCs to be sequential in nature and not parallel as the older wall mounted systems are. Only by rapid processing does the parallel ladder model appear to be processing simultaneously. Large RLL diagrams slow down the processing, displaying the sequential nature of the PLC.

As explained in an earlier section, the updating of output is performed at the end of a ladder scan, so any change to output in rung 1 cannot affect rung 2 until the output update at the end of the ladder scan. Petri nets are used routinely to model parallel processes. Our parallel PN model demonstrates two rungs evaluated not in parallel but sequentially due to the higher logic present in the CPU of the PLC. Development of the PN model for a true parallel evaluation is left for future investigative work. The processing used by parallel models is less complex to design, perhaps more complex to implement on a CPU.

CHAPTER 5. ALGORITHMS FOR PETRI NET MODELS

PN models are built to represent the logic process RLL uses to process inputs and produce outputs. Higher level processing of RLL, built into the CPU of the PLC, must be incorporated into the model. There are two separate levels of modeling taking place; the higher level is called Process Nets (PrNs) and the lower level is labeled Rung Evaluation Nets (RENs). The PrN represents the PLC's CPU control system, and the REN is a rung of the relay ladder diagram. The RENs are inside the PrN and the PrN controls the processing of the REN input and output evaluations.

The RLL models are kept small for purposes of clarity, ease of demonstration, and testing. The algorithms used can be applied to any size RLL system. A thorough understanding on smaller RLL systems is suggested before any large ladders are evaluated.

Process Nets

Figure 5.1 shows the PN representation of the PLC processing and Table 5.1 lists the state or action of each place or transition. There is one token in the PrN during evaluation. Another PrN model using parallel output evaluation requires as many tokens as there are output evaluation nets. The PrN evaluates each rung of the model and controls the updating of the future to current memory addresses for each

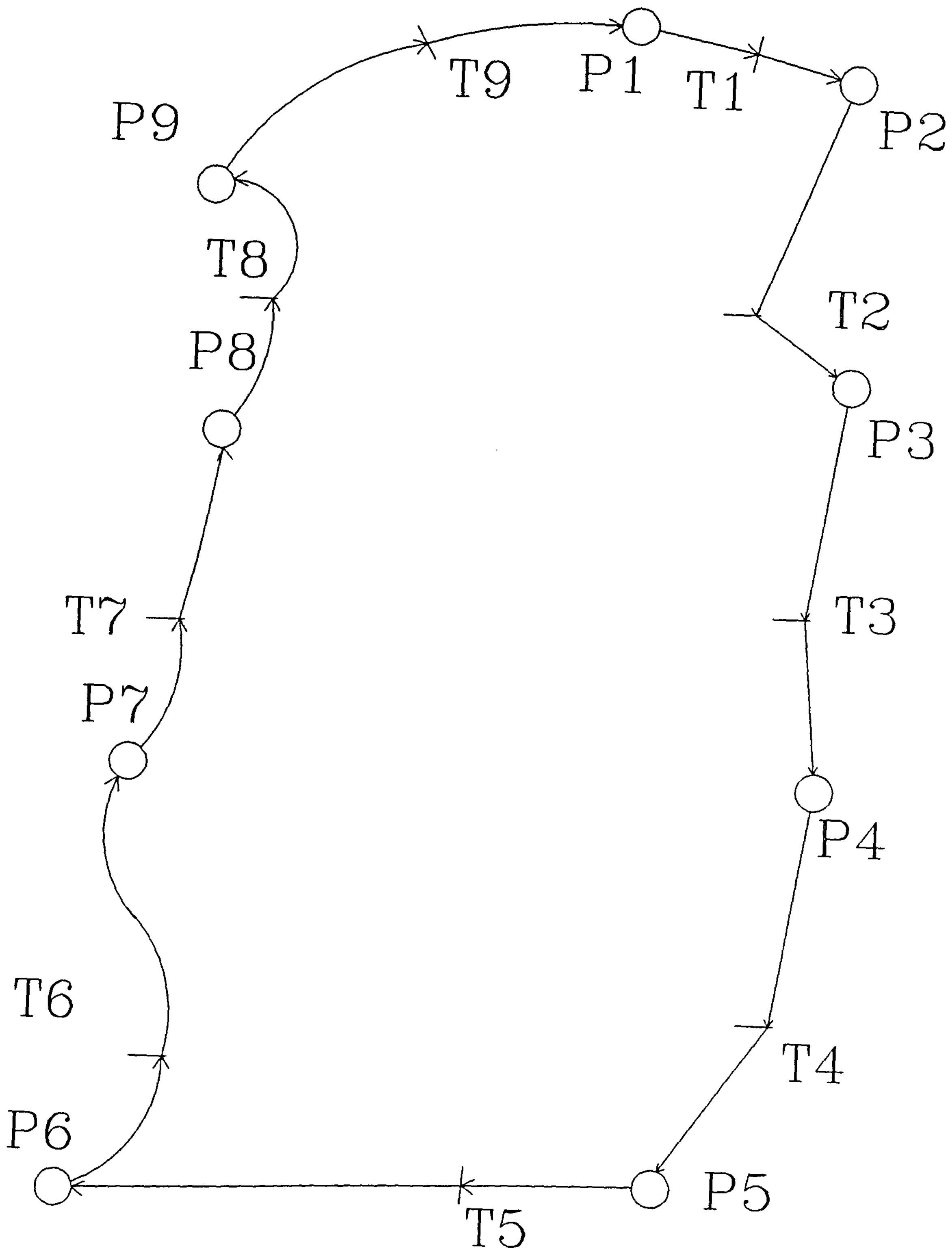


Figure 5.1: PrN for serial RLL

Table 5.1: Transition and place representations for serial PrN

Place	State represented
P1	Ladder scan ready
P2	Rung 1 evaluation ready
P3	Rung 2 evaluation ready
P4	Rung 3 evaluation ready
P5	Ladder scan complete; Output evaluation ready
P6	Rung 3 output scan ready
P7	Rung 2 output scan ready
P8	Rung 1 output scan ready
P9	Output scan completed
Transition	**Event(s) represented**
T1	Start ladder scan, Gather inputs
T2	Evaluate rung 1
T3	Evaluate rung 2
T4	Evaluate rung 3
T5	Mark all output scan places
T6	Evaluate output rung 3
T7	Evaluate output rung 2
T8	Evaluate output rung 1
T9	Report current output statuses; Set ladder scan ready

scan of the ladder. The PrN is composed of:

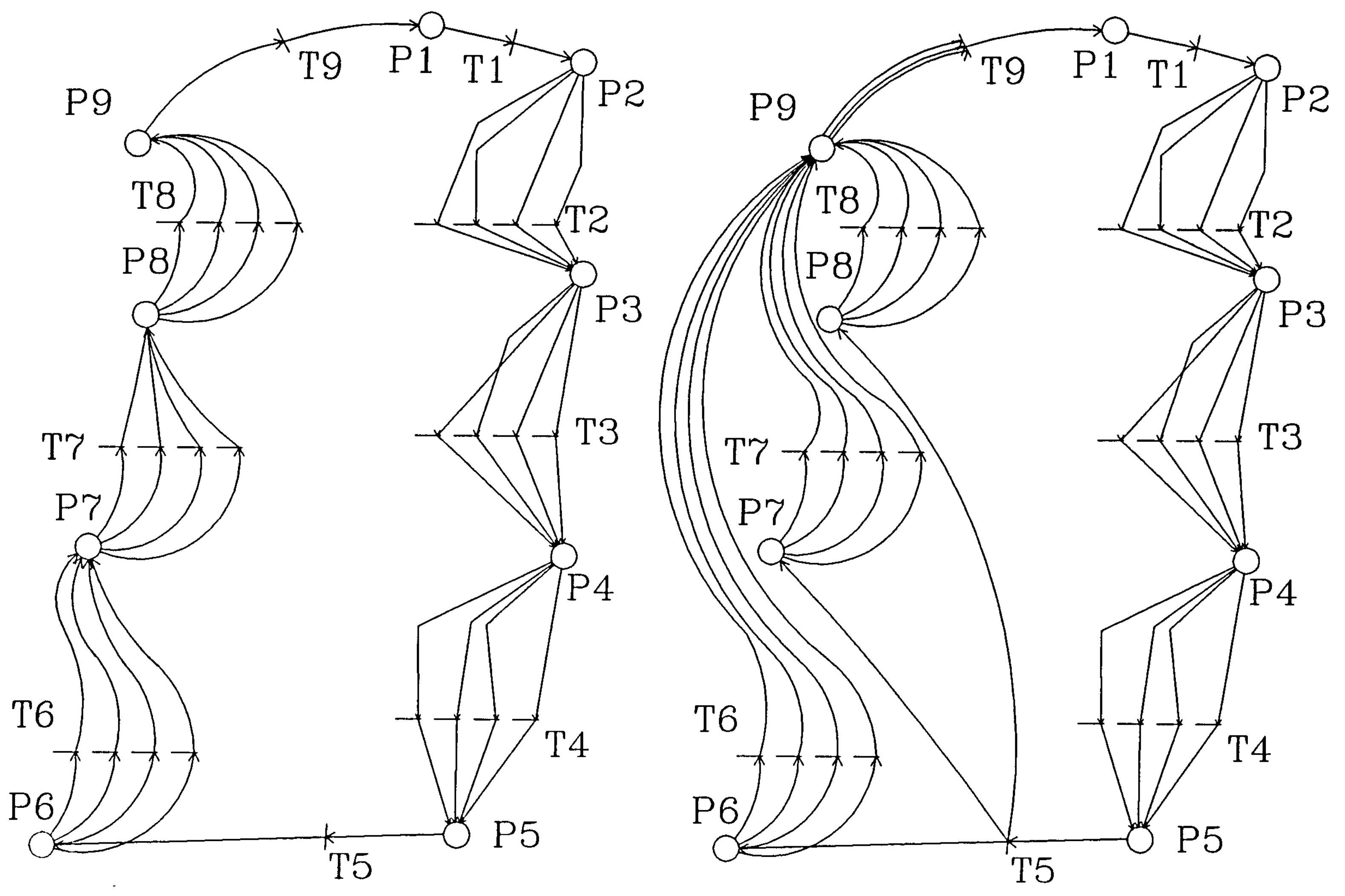
- places and transitions for every REN;
- a place to activate the output evaluation segment of the PrN, and (in the parallel model) a transition to put tokens into every output evaluation place; otherwise, a transition to start output evaluation;
- places for each output evaluation,
- a place to collect the output evaluation token (tokens if the parallel model is used);

- a transition to end the ladder scan and report status of outputs (if the parallel model is used this transition consumes all tokens generated by output evaluation and produces one token to keep the net safe);
- a place to represent completion of ladder scan (the beginning marking) and a transition to activate the ladder scan.

The serial model discussed above and the changes to produce a parallel output evaluation model are shown in Figure 5.2. Note the differences from T5 through T9 of the two models.

The RENs are independent PNs representing one rung of the RLL tied together via the PrN. The PrN model in Figure 5.1 evaluates ladder rungs at transitions T2, T3, and T4. The models in Figure 5.2 are the same model as Figure 5.1. The transitions T2, T3, T4, T6, T7, and T8 have more than one transition detailed in Figure 5.1 while Figure 5.2 shows only one transition at these locations; the PrN transition T2 represents multiple REN transitions. Since only one transition of the REN can be enabled (they are mutually exclusive events) the single token is preserved and no conflict occurs during REN evaluation and the larger PrN model can be shown with only one transition to represent the higher level process.

After REN input transition evaluations, the PrN for parallel output modeling fires transition T5 putting a token in each REN output evaluation place. There is an output evaluation net for each REN. Place P9 gathers these tokens after output evaluation and transition T9 enables when all tokens created at T5 are collected in P9. An alternative to parallel processing the output evaluation is sequential processing.



Left) Serial PrN output evaluation Right) Parallel PrN

Figure 5.2: Two output evaluation schemes for the PrN update analysis

Figure 5.2 shows the two alternatives from T5 through T9. The serial PrN uses a single token while the parallel PrN splits and rejoins the tokens. Either process preserves safeness. The parallel processing allows all outputs to update simultaneously. However, the PN tree that it generates can cause evaluation and documentation problems. The output evaluations do not affect the input places for REN evaluation because the output evaluations are performed after T5 (the end of Ladder rung scan phase) and before T9 (the transition to Ready to begin Ladder scan state). This PrN location isolates the output update from the current input data allowing us to update the outputs using serial rather than parallel processing without affecting the REN outcomes. The PrN models used in this study are serial like the left hand model of Figure 5.2.

Generic REN Modelling Steps

One goal of this research is to arrive at a system of RLL evaluation applicable for any combination of basic rung elements shown in Figure 4.1. In addition, PN modeling requires that we move a token through each net; so all possible outcomes must be modelled rather than the outcomes activating the network. This is not true in RLL due to the built in instruction set of the PLC's CPU.

The RLL in Figure 4.2 demonstrates that the Boolean expression $(Y1 + X1) \cdot \overline{X2}$ can model the activation of $Y1$. This expression will activate $Y1$ for the three combinations of $Y1$, $X1$, and $X2$ (assuming a current and future value for $Y1$) but relies on the instruction set imbedded in the PLC to bypass any rung showing no change. PNs use tokens to activate states; thus the additional consideration of testing the REN when there is no change in state (i.e., 1 is still 1) is necessary to move tokens to a new state by enabling and firing a transition.

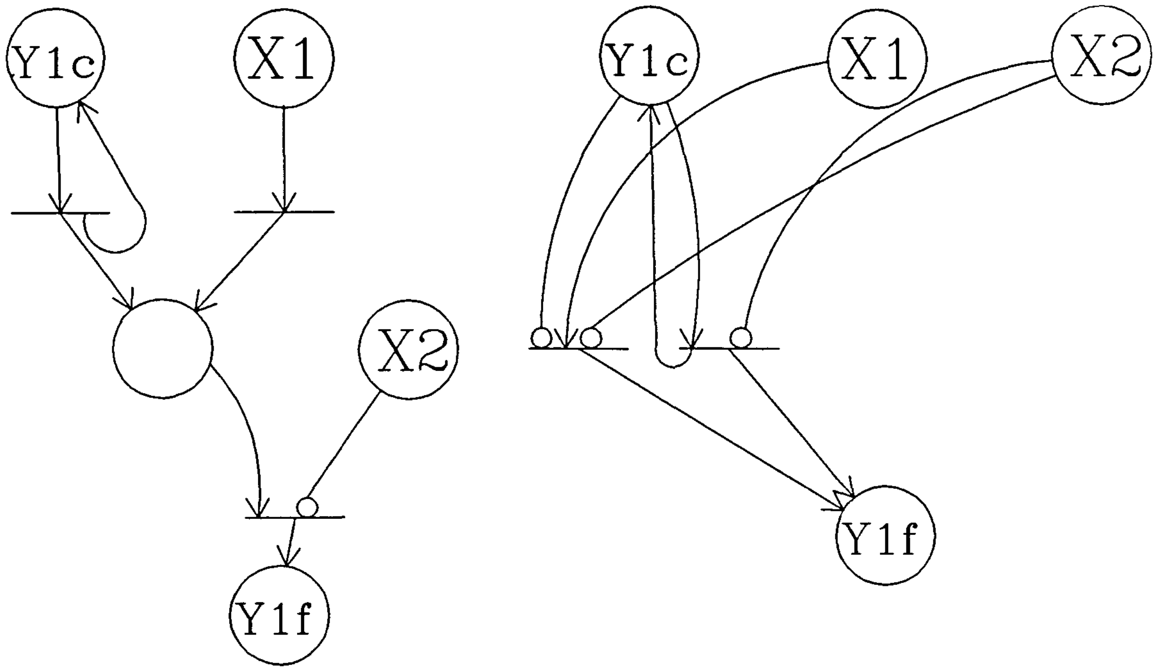
Conflict resolution in PN evaluation requires explicit processing rules. This is undesirable for PN systems used as documentation tools because the explicit rules must be recorded and remembered separately. The PN drawing for these "ruled" systems no longer completely demonstrate the logic of the system. One of the purposes of converting RLL to PN representations is to provide a documentation tool; if the tool does not carry all of the information on the processing of the RLL and relies on outside rules for conflict resolution it become a less useful tool. PN used as an analysis tool to produce reachability trees will also be hampered by the use of rules not built into the model.

In consideration of the previous three paragraphs, the PN based RLL modeling system being developed in this paper must consider:

- the transitions activating the future output,
- the transitions reversing the future to current output, and
- the transitions maintaining the status quo (i.e., 0 is still 0),

in order to move tokens through the system. In addition, the model must be able to produce conflict free transition sets for the RENs so no explicit rule set is needed.

The modeling of 2^n transitions, where n is the number of inputs to the rung, quickly becomes bulky; it is desirable to minimize the number of transitions used by reducing the Boolean expression to a minimum number of phrases. This desire to minimize must be modified by the need to eliminate transition conflict. Developing a



Left) Mixed AND/OR logic Right) Mutually exclusive logic

Figure 5.3: Comparison of Boolean expressions using PN representations

mutually exclusive set of transitions eliminates conflict with a slight (if any) increase in the number of transitions required to evaluate the system.

Another consideration for users of this modeling system is keeping the visual interpretation simple. If an REN has multiple levels of places and transitions associated with the logic to evaluate one rung of RLL, then the simplicity is destroyed and ease of comprehension is reduced. An example of this in Figure 5.3 is the logic for serial rung 1 displayed in PN form on the left and in the REN form of this paper on the right. The lower case c represents the current state of the output variable, and the lower case f the future state. For simplicity, the two PNs only address the activation of the future output in this example. Activation transitions are transitions with arcs from the transition to the future output place; deactivation transitions have no arcs from the transition to the future output place. The $Y1f$ place in both drawings and the unnamed place receiving tokens from $Y1c$ and $X1$ in the left drawing OR the transitions feeding them. The transitions AND the places entering the transition. The complexity of the activation network is increased by mixing AND and OR notation in the expression for the PN representation (the left hand model); the REN transition set reduces the complexity and increases comprehension by giving the reader constant information at each transition for all input variables and allowing users to readily view all optional paths for a token. Each transition is a mutually exclusive transition so only one transition is enabled during any scan of the system. The system on the left of Figure 5.3 has the potential for multiple token generation or for conflict with certain initial markings. The returning arc of $Y1c$ is necessary for any variable affecting information elsewhere in the model. All of the current output variables have a returning arc wherever a token is required to enable a transition because the current output status is needed for the output evaluations. Any X input variables appearing more than once in the same RLL model also need returning arcs. Although the left PN of Figure 5.3 appears to be able to generate an infinite number of tokens from $Y1$ to the unnamed interim place, this is not the case. The controlling factor is the PrN input arc not shown in the drawing to reduce the complexity. This arc controls the flow of the PrN token through only one of the two transitions available. The presence of this arc prevents the multiple token potential mentioned earlier but produces the potential for conflict between the two transitions when each has a token available for delivery to the interim place.

By keeping the transition sets between RENs similar, the RENs become easier to decipher and understand. Each transition in the system represents a Boolean AND. The REN transitions are a set of ANDed conditions ORed together creating an "exclusive or" set of transitions meaning only one transition can be enabled any time the system is evaluated. The "exclusive or" concept is better described by the term mutually exclusive. Rung 1 of the RLL demonstrated in Figure 4.2 has a Boolean representation of $(Y1c + X1) \cdot \overline{X2}$ (where $Y1c$ denotes the current value of $Y1$ and $Y1f$ is the future value) and a PN representation in the left plate of Figure 5.3. The PN can produce two tokens when both $Y1C$ and $X1$ have the value of 1 if no other controlling network exists and all enabled transitions are allowed to fire. The PN is not safe if a transition is capable of generating an undeterminable number of tokens. This network could create an infinite number of tokens if enabled transitions fired multiple tokens into the interim place and only removed one token during each scan of the system. The controlling network (PrN) arcs not shown in the drawing limit this firing yet still allow conflict between two transitions when certain initial markings are used.

The Boolean expression $(\overline{Y1c} \cdot X1 \cdot \overline{X2}) + (Y1c \cdot \overline{X2})$ represents the same logic employing a mutually exclusive set of phrases and is displayed on the right plate of Figure 5.3 in a more comprehensible manner using transitions having a logical variable order. The expression preserves safeness by producing only one token while avoiding conflict between transitions. The work of this thesis is to create a PN modeling algorithm that:

- is repeatable for every RLL rung containing the basic components of RLL;
- identifies the activation/deactivation of the future output, or recognizes status quo events;
- eliminates conflict by ensuring that each combination of inputs enables one and only one transition of the REN each time a token passes through the REN; and
- simplifies reading of the transition set visually and logically for each REN.

As the work for this thesis progressed, two paths to solution emerged. The first path is the mathematical approach to the solution. In this approach, the RLL information is coded into algebraic expressions using Boolean logic, then mathematically reduced to mutually exclusive sets of phrases. This approach is valid and usable for any size system although it becomes increasingly difficult and time consuming work beyond three input variables. The second path is based in the Boolean algebraic solution, but is a more algorithmic solution lending itself to automation. The tabular algorithm involves sorting rows in tables and choosing variable combinations defining mutually exclusive REN transition sets. With the tabular algorithm a set of manipulation rules can be written allowing automation of the solution in the future. The conversions to RENs in this paper will use the tabular method. The algebraic solution is shown to provide a basis for the tabular solution and demonstrate the mathematics behind the row sorting and variable selection. Each algorithm produces the same result.

The mathematic algorithm for translating RLL rungs into RENs is:

1. Build a binary input combinations table.
2. Generate the desired binary output results for the table.
3. Build, reduce, and choose the optimal Boolean output activation expression.
4. Convert each phrase of the Boolean activation expression into a REN activation transition.
5. Compliment and reduce the Boolean activation expression; choose the optimal Boolean deactivation expression.
6. Convert each phrase of the Boolean deactivation expression into a REN deactivation transition.

A detailed explanation of each of these steps follows in the section on the mathematical algorithm.

The tabular algorithm for translating RLL rungs into PN based RENs is:

1. Build a binary input combinations table.

2. Generate the desired binary output results for the table.

3. Sort the table by the output value grouping activation and deactivation rows.

4. Solve for the activation expression by using rows with output values equal to 1 and finding the input variable combinations defining the most rows using the least number of variables without defining any deactivation rows or any previously defined activation rows. Repeat this step until all rows with output values of 1 have been defined by input variable combinations.

5. For each phrase of the activation expression generated in the previous step build the activation transition.

6. Solve for the deactivation expression by using rows with output values equal to 0 and finding the input variable combinations defining the most rows using the least number of variables without defining any activation rows or any previously defined deactivation rows. Repeat this step until all rows with output values of 0 have been defined by input variable combinations.

7. For each phrase of the deactivation expression generated in the previous step build the deactivation transition.

A complete discussion and examples of this algorithm appear in the section on the tabular algorithm beginning on page 48.

The mathematical algorithm

Build a binary input combinations table The table is a binary combinations table with columns for each rung input and output. It is a standard digital

Table 5.2: Truth table for the start/stop rung

$Y1c$	$X1$	$X2$	$Y1f$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

logic table representing every input combination possible. This table is the complete set of combinations not just those that the system designers consider. This allows for the testing for every condition in the system. This list will have 2^n rows where n is the number of input variables. The table technique is common to digital logic. Essentially it is binary counting from 0 to $2^n - 1$ using an n bit register with each row being incremented by 1. The right hand column (output column) remains blank. It is filled in by performing the next step of the algorithm.

Generate the desired binary output results for the table Mark the output column with 0 or 1 using the RLL's Boolean logic to evaluate inputs. Use a 1 in the future output columns for any combination of inputs activating the output and a 0 in the future output column for any combination rendering the output inactive. Though some of these markings cannot occur in the real world system, they must be considered in any logic system. The first two steps build tables like Table 5.2. Table 5.2 is a binary counting table with future output generated from the Boolean analysis of a RLL rung for the given row's input variable values.

Build, reduce, and choose the optimal Boolean output activation expression By looking for all of the active output rows (output column values of 1), representing the rows as ANDed Boolean combinations, and combining the representation phrases into an ORed statement, a Boolean expression is generated. Table 5.2 has active output Boolean representations of

$$\overline{Y1c} \cdot X1 \cdot \overline{X2}$$

$$Y1c \cdot \overline{X1} \cdot \overline{X2}$$

and

$$Y1c \cdot X1 \cdot \overline{X2}$$

for rows 3, 5 and 7 respectively. OR the three phrase representations of rows into one expression to get

$$(\overline{Y1c} \cdot X1 \cdot \overline{X2}) + (Y1c \cdot \overline{X1} \cdot \overline{X2}) + (Y1c \cdot X1 \cdot \overline{X2}). \tag{5.1}$$

Reducing the expression to the lowest level of mutually exclusive set of ORed combinations (using Boolean algebra) is the next step. Boolean algebra is evaluated based on priority rules. The order of Boolean algebra evaluation is bracketed phrases, parenthetic phrases, ANDed values (the $\cdot$ sign indicates an AND operation), and then ORed values (indicated by a $+$ sign).

Reduction of expression (5.1) begins by combining the second and third phrases noting that $X1$ and $\overline{X1}$ are the only differing variables. The ORing of compliments produces a Boolean value of 1 in every case. This elimination of the $X1$ variable produces the expression

$$(\overline{Y1c} \cdot X1 \cdot \overline{X2}) + (Y1c \cdot \overline{X2}). \tag{5.2}$$

Boolean algebra further reduces the expression to

$$(\overline{Y1c} \cdot X1 + Y1c) \cdot \overline{X2}$$

and finally to

$$(Y1c + X1) \cdot \overline{X2}. \tag{5.3}$$

Expression (5.3) is not mutually exclusive because it does not provide for the "exclusive or" between $Y1c$ and $X1$. Expression (5.2) is mutually exclusive since the $Y1c$ variable has its compliment appearing in the other phrase. Multiple rows of the combination table satisfy expression (5.3) thus conflict would exist in the PN representation. PrN safeness would be violated if a PN from expression (5.3) was inserted in the PrN.

No conflict is one goal of the algorithm, and a Boolean expression containing a set of mutually exclusive phrases is another goal; the Boolean expression (5.2) is an activation expression meeting both goals. This represents the least number of transitions needed to model the activation of the $Y1$ output without conflict. Essentially the process incorporates the compliment of the chosen variable into the other phrases of the expression. Based on this process an alternative solution exists.

By taking expression (5.1) and combining the first and third terms the expression

$$(X1 \cdot \overline{X2}) + (Y1c \cdot \overline{X1} \cdot \overline{X2}) \tag{5.4}$$

is created. This expression and expression (5.2) are both mutually exclusive and represent the activation expression. Having two equivalent transitions sets requires additional consideration for selection.

All other considerations being equal and the expressions meeting the criteria of mutual exclusivity the simplification of the REN drawing becomes important. By keeping the number of transitions, arcs, and places to a minimum, the drawing becomes more readable and easier to produce graphically. One consideration for simplification of the drawing is the number of arcs used to generate the REN.

Optimization of the number of arcs considers the PN representation of the RLL system. Each time a variable is included in a phrase an arc is drawn to represent it. If the variable must be renewed for use by other RENs or by the output evaluation, then a returning arc to the enabling place must be provided from each transition enabled by that variable. By selecting the Boolean expression to minimize the number of arcs, we reduce the drawing complexity. Each input variable used elsewhere in the PrN or RENs must have the token returned to the enabling place by the enabled transition thus requiring two arcs. Any transition enabling a future output place has an additional arc besides the input arcs. Any variable not used elsewhere in the PrN or RENs requires only one arc to represent it and does not require the token to be returned to the enabling place. Any transition enabled by the absence of a token in a place requires only one inhibitor arc and no returning arc.

Since the input variables are not used in succeeding rungs, the only multiple arc consideration for the start/stop REN is $Y1c$ (used in rung 2). Expressions (5.2) and (5.4) each require six arcs and two transitions to model the activation expression. The choice in this case is moot, and expression (5.2) will be used. Additional examples will be discussed later to clarify this optimization step.

Convert each phrase of the Boolean activation expression into a REN activation transition Based on the reduced Boolean expression of the previous section we need to build the transition(s) to represent state combinations causing active output for the rung. All of our transition models will have a similar structure. The first two transitions of Figure 5.4 and the left two transitions of T2 in Figure 5.3 (left plate) are the same transitions. The right side of each transition in Figure 5.4 has no arc from the PrN. The location where the PrN arc enters and leaves the transition is open in Figure 5.4 and each REN drawing will have an opening on the right side of each transition for the PrN arc to be included later. The transition constructions are based on arc location upon the transition. There are 4 positions on the transitions of Figure 5.4 representing 3 input variables and 1 PrN arc. The first transition has all three input places filled; the 2 inhibitor arcs represent the $\overline{Y1c}$ and $\overline{X2}$ components of the phrase in the reduced Boolean expression of the last section, and the arrowhead represents the $X1$ component. The second activation transition (transition 2 of Figure 5.4) uses just 2 components to define the transition; note that the physical position for $X1$ is left open to indicate that variable is not used. This notation will be followed throughout the thesis to maintain comprehension and ease of reading PN drawings.

The last item to note in Figure 5.4 is the presence of arcs from the first two

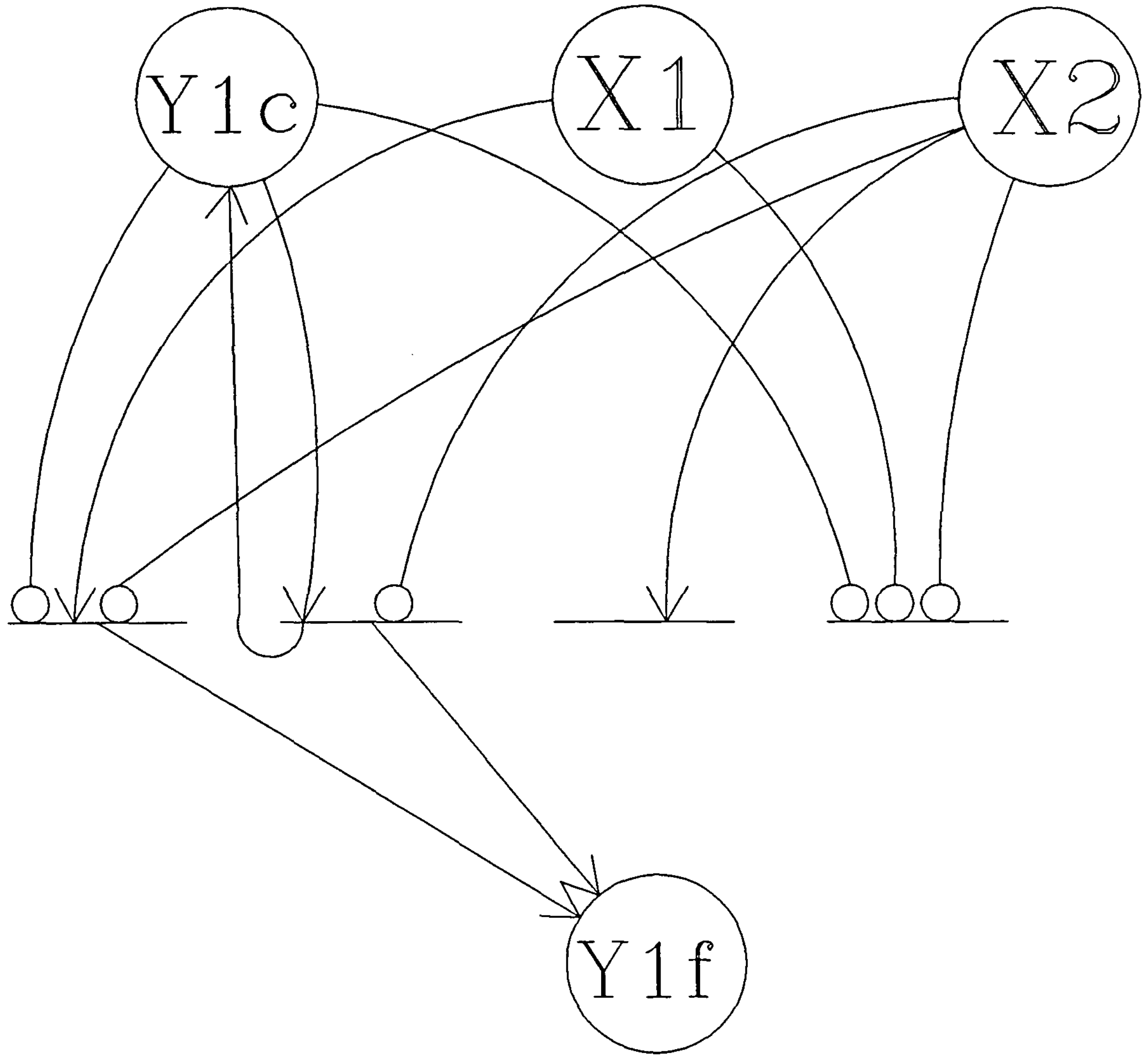


Figure 5.4: PN model of serial rung one

transitions to the $Y1f$ place in the REN model. These arcs represent the future activation of $Y1$ and the $Y1f$ place holds the activation token until the output update phase is reached. It is the presence of these arcs from the transition to the future output place that denotes an "activation" transition. Conversely, the absence of the arc from the transition to the future output denotes a "deactivation" transition which will be developed next. The use of $Y1f$ and $Y1c$ notation prevents the $Y1$ change of state from affecting other rungs during the present ladder scan. After the output evaluation takes place, the change in state of $Y1$ will exercise its effect on any rung using $Y1c$ as input during the next ladder scan.

Compliment and reduce the Boolean activation expression; choose the optimal Boolean deactivation expression The set of transitions deactivating the output must be developed. Boolean logic lends itself to this task handily. When you have a set of true conditions, the compliment (reverse) of that set of conditions yields a set of opposing conditions. Time being an important factor there is a limit to what should be processed by hand manipulation of Boolean expressions. The manipulation rapidly becomes time consuming after 4 input variables. Tabular methods become more expedient.

Complimenting the expression; reducing to the simplest set of OR combinations; and solving for an optimal mutually exclusive representation begins with the complimenting of expression (5.2) giving

$$\overline{(\overline{Y1c} \cdot X1 \cdot \overline{X2}) + (Y1c \cdot \overline{X2})}. \tag{5.5}$$

This expression must now be reduced. Boolean algebra states that any compliment crossing ANDed conditions yields complimented Ored conditions. The same is true when a compliment crosses an ORed conditions; complimented AND conditions are the result. Imbedding the compliment of expression (5.5) produces

$$\overline{(\overline{Y1c}\cdot X1\cdot \overline{X2})}\cdot\overline{(Y1c\cdot \overline{X2})}.$$

Further reduction using the same imbedding technique across each complimented AND phrase results in

$$(Y1c+\overline{X1}+X2)\cdot(\overline{Y1c}+X2).$$

Combining variables across the ANDed condition and reordering them results in

$$X2+(Y1c\cdot X2)+(\overline{Y1c}\cdot X2)+(\overline{X1}\cdot X2)+(\overline{Y1c}\cdot\overline{X1}). \qquad (5.6)$$

The variable $X2$ appears in the first three phrases and is implicit in the third phrase ORed with its compliment. By expanding the fifth phrase we get

$$X2+(Y1c\cdot X2)+(\overline{Y1c}\cdot X2)+(\overline{X1}\cdot X2)+(\overline{Y1c}\cdot\overline{X1}\cdot X2)+(\overline{Y1c}\cdot\overline{X1}\cdot\overline{X2}).$$

Recombining using $X2$ as a common variable the expression

$$X2\cdot(1+Y1c+\overline{Y1c}+\overline{X1}+\overline{Y1c}\cdot\overline{X1})+\overline{Y1c}\cdot\overline{X1}\cdot\overline{X2} \qquad (5.7)$$

is generated. The 1 inside the parenthesis denotes the presence of $X2$ as a single variable in the expression. Boolean algebra provides that 1 ORed with any other variable(s) equals 1; 1 ANDed with any other variable equals that variable. These two facts reduce expression (5.7) to

$$X2+\overline{Y1c}\cdot\overline{X1}\cdot\overline{X2}. \qquad (5.8)$$

Expression (5.6) could be reduced to

$$X2 + \overline{Y1c} \cdot \overline{X1}; \tag{5.9}$$

but mutual exclusivity is a necessary condition satisfied by expression (5.8) not by expression (5.9). Expression (5.8) is the reduced mutually exclusive representation desired in this manipulation.

The last step of this selection process is to find the optimal solution by looking for the least number of arcs for the deactivation expression. This example generated a final solution before optimization was necessary. However, using the complimented expression from serial RLL rung 3 as an example yields six mutually exclusive but not necessarily optimal solutions.

Rung 3 of the serial ladder has one activation phrase:

$$Y2c \cdot X4 \cdot X5.$$

The imbedded compliment of that phrase is the expression

$$\overline{Y2c} + \overline{X4} + \overline{X5}.$$

This is not the mutually exclusive set because any two of the variables equalling 0 causes conflict between the ORed conditions. To resolve conflict, a mutually exclusive set of transitions must be generated. The solution of this expression involves selecting one variable to define 4 rows. The remaining two phrases must have the compliment of the selected variable ANDed to them. One of the remaining pair of phrases is selected to define two rows and the remaining two variable phrase has the compliment of the last variable ANDed to it defining the single remaining row. This procedure produces 6 solutions listed below with arc requirements.

1. $\overline{Y2c} + Y2c \cdot \overline{X4} + Y2c \cdot X4 \cdot \overline{X5}$ requiring 8 arcs,

2. $\overline{Y2c} + Y2c \cdot \overline{X5} + Y2c \cdot \overline{X4} \cdot X5$ requiring 8 arcs,

3. $\overline{X4} + \overline{Y2c} \cdot X4 + Y2c \cdot X4 \cdot \overline{X5}$ requiring 7 arcs,

4. $\overline{X4} + X4 \cdot \overline{X5} + \overline{Y2c} \cdot X4 \cdot X5$ requiring 6 arcs,

5. $\overline{X5} + \overline{X4} \cdot X5 + \overline{Y2c} \cdot X4 \cdot X5$ requiring 6 arcs, and

6. $\overline{X5} + \overline{Y2c} \cdot X5 + Y2c \cdot \overline{X4} \cdot X5$ requiring 7 arcs.

Selection of expression 4 or 5 is equal; so, the first one generated is used in Figure 5.5 to generate the last 3 transitions of that REN.

Convert each phrase of the Boolean deactivation expression into a REN deactivation transition The same rules for building activation transitions apply to building deactivation (i.e., $Y1f = 0$) transitions. The transition structure will leave room for every input variable and the PrN arc. The last two transitions of Figure 5.4 are the representation of the phrases in the expression

$$X2 + \overline{Y1c} \cdot \overline{X1} \cdot \overline{X2}.$$

The first phrase of the expression has only the arrowhead input arc from $X2$. The second phrase uses the inhibitor arc for the variable $X2 = 0$ (a "not" condition). Note that these two arcs appear in the same physical location on their respective transition. If a variable is not used in the enabling of the transition, its space remains to improve the comprehension of the transition. The first of these two transition does not require input from the variable $Y1c$ but the location where that variable would enter the transition is left open to represent the absence of the variable. There are no arcs from the last two transitions of Figure 5.4 to the $Y1f$ place because they are deactivation transitions.

The future places have no tokens in them upon starting a ladder scan. The tokens are removed during the output evaluation phase of the PrN. The transitions in each REN are a mutually exclusive set to avoid conflict and maintain one token to and from the REN, preserving safeness.

The expression chosen for the deactivation of serial rung 3,

$$\overline{Y2c} \cdot X4 \cdot X5 + \overline{X4} + X4 \cdot \overline{X5},$$

contains three phrases representing the last three transitions of Figure 5.5. The structure of the deactivation transitions is the same as the previous activation transitions. Open spaces are left on the transition for the PrN arc and any variable not used by the transition.

The tabular algorithm

As the number of input variables increases, the mathematical algorithm becomes more time consuming to perform. The tabular method is based on the Boolean evaluation method, but is more convenient because of the reduced time to produce the same results. The tabular algorithm will be used in the analysis of the models. The previous work in the mathematical algorithm helps to explain the origin of the tables and the relationships of the rows to their Boolean expression components. A definition for 'defining a row' is given to help understand the process being used.

For n input variables a binary combinations table with 2^n rows will be generated.

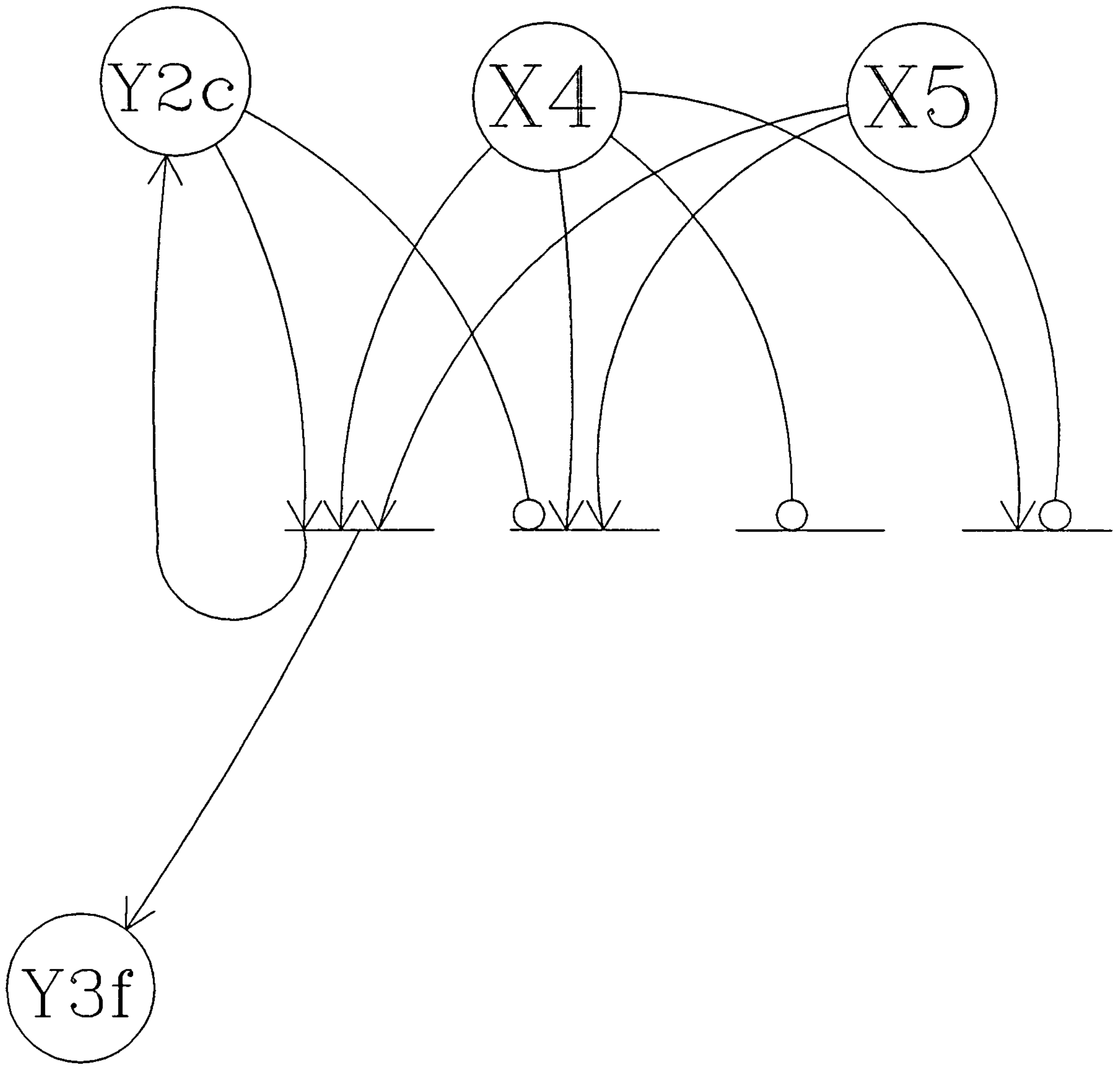


Figure 5.5: PN model of serial rung three

Definition 5.1 *A set of variables define a set of rows in the table if it is the minimum set of input variables necessary and sufficient to specify the maximum set of desired outcomes.*

For table having n input variables the maximum number of rows a Boolean phrase can define is 2^{n-x} where x is the number of variables used in the Boolean phrase. The most rows a single variable can define is 2^{n-1}; two variables define 2^{n-2} rows; three variables define 2^{n-3} rows; etc. This information coupled with the knowledge of how many rows we have to define reveal the minimum number of variables for the first Boolean phrase to define the desired rows. For example, using a 3 input variable table with eight rows; if the deactivation rows (i.e., output values $= 0$) number six, the first Boolean phrase can be a single variable phrase defining 4 rows. If the deactivation rows numbered three, then the first Boolean phrase would use two variables defining 2 rows.

Build a binary input combinations table This step is the same as the mathematical algorithm.

Generate the desired binary output results for the table By applying Boolean logic to the RLL rung 1 of Figure 4.2 the output values of Table 5.2 are generated.

Sort the table by the output value to group activation and deactivation rows Sorting by the output variable groups the activation rows (i.e., $Y1f = 1$) and deactivation rows (i.e., $Y1f = 0$) to produce Table 5.3.

Table 5.3: Sorted table for serial rung one

$Y1c$	$X1$	$X2$	$Y1f$
1	0	0	1
1	1	0	1
0	1	0	1
0	0	0	0
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0

Solve for activation expression The activation rows of Table 5.3 (i.e., $Y1f = 1$) can be sorted further to produce Boolean phrases defining the most activation rows using the least number of variables in a phrase. In our example, a single variable set to a value (i.e., $X2 = 0$), can define four rows. In this table, only three rows are activation rows, so another variable will have to be added to the phrase. Using our three input table and a single variable phrase, 2^{3-1} or 4 rows are defined. If two variables are used in the phrase, 2^{3-2} or 2 rows are defined. The goal of this step in the algorithm is to define the most activation rows with a Boolean phrase containing the fewest variables.

The variable $X2$ is equal to 0 for all activation rows. Given this fact, it should be included as one of the two variables in the Boolean phrase to define the most rows. The second variable can be $Y1c$ or $X1$; this leads to two possible phrases. By choosing $X1 = 1$ and $X2 = 0$ the Boolean phrase $X1 \cdot \overline{X2}$ defines two rows in Table 5.4 leaving a third row undefined. This third row requires three variables in the Boolean phrase to be defined uniquely ($2^{3-3} = 1$).

Table 5.4: Second sorting of start/stop truth table

$Y1c$	$X1$	$X2$	$Y1f$
1	1	0	1
0	1	0	1
1	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0
0	0	1	0
0	0	0	0

The tabular solution generates the Boolean phrases required for conversion to transitions in the RENs. Sorting activation rows from Table 5.3 by the input variable values $X2 = 0$ and $X1 = 1$ results in Table 5.4. A second sort using variable values $X2 = 0$ and $Y1c = 1$ generates Table 5.5. A combination of the sorted tables is shown in Table A.1 contrasting the difference in the sorts.

Activation expressions from Table 5.4 and Table 5.5 both satisfy the mutual exclusivity requirements; the expression from Table 5.4,

$$X1 \cdot \overline{X2} + Y1c \cdot \overline{X1} \cdot \overline{X2},$$

is generated by ANDing the two variables defining the double row then ORing it with the three variables ANDed to define the remaining row. The Boolean activation expression from Table 5.4 corresponds to expression (5.4) of the mathematical algorithm. The expression from Table 5.5 is generated in the same manner but uses the $Y1c$ variable instead of the $X1$ variable to define the double row when ANDed with $\overline{X2}$ to define a different pair of rows. The remaining single row is defined with a

Table 5.5: Third sorting of start/stop truth table

$Y1c$	$X1$	$X2$	$Y1f$
1	0	0	1
1	1	0	1
0	1	0	1
0	1	1	0
1	0	1	0
1	1	1	0
0	0	1	0
0	0	0	0

three variable Boolean phrase. The two and three variable phrases are ORed together to form a Boolean expression for activation of serial rung 1. The activation expression from Table 5.5 corresponds to expression (5.2) of the mathematical algorithm. Since both of these Boolean activation expressions are mutually exclusive an optimal solution must be tested for.

The next step is to find an optimal solution with respect to the number of arcs required to build RENs from each expression. This activity is the same as that performed in the mathematical algorithm. Each expression requires 6 arcs for implementation, so the choice is left to the builder. The second expression is chosen for transition construction in this example as it was the first to appear in Table A.1.

For each phrase of the activation expression generated in the previous step build the activation transition Construction of transitions remains the same as that presented in the mathematical algorithm. The structure, arc locations and meanings hold true also. The left pair of transitions in Figure 5.4 detail the construction of the activation Boolean expression generated from Table 5.5 (Table A.1 A).

Solve for the deactivation expression This step is where the most time savings is realized over the mathematical algorithm. As in the section solving for activation phrases, we group common variable values. If any input variable had a common value for all activation rows, the first sort of the deactivation rows should be for this variable's compliment. Sorting for the compliment of a common activation variable value will yield the most rows using a single variable in the Boolean phrase. In the start/stop rung set variable $X2$ has a value of 0 for all activation rows; the variables $Y1c$ and $X1$ have values of 0 and 1 in the activation rows. This means that the deactivation rows contain all of the rows with the variable $X2$ equal to 1. Sorting deactivation rows for $X2 = 1$ yields four rows definable with a Boolean phrase containing one variable. One row remains to be defined and three variables are required to define a single row.

The Boolean expression for deactivation of the future output is generated by ORing the single variable Boolean phrase with the triple variable Boolean phrase. The activation expression is

$$X2 + \overline{Y1c} \cdot \overline{X1} \cdot \overline{X2}.$$

There is only one expression meeting the maximum row definition criteria, so the step of choosing an optimal solution based on arcs required to build the transitions is eliminated. Serial rung three used in the mathematical algorithm is a good example of multiple mutually exclusive Boolean phrases requiring optimal analysis.

Serial rung three is a case where one row activates the output and seven rows deactivate the output. With seven rows to sort, four can be eliminated with a Boolean expression using a single variable. Because the activation has only one row, there are three variables to compliment and sort with on the first pass. After defining four rows out of seven, the remaining three rows can be sorted two ways. This compounding results in six possible tables. Table A.3 in Appendix A shows the six alternative tables. The Boolean expressions generated from these alternatives correspond to the six Boolean expressions generated by the mathematical algorithm as shown on page 46. The sort chosen to represent the REN deactivation transitions for serial rung three is Table A.3 part C.

For each phrase of the deactivation expression generated in the previous step build the deactivation transition Once again, construction of the transitions is the same as that performed in the mathematical algorithm. Each transition has a designated space for the input variable arcs and the PrN arc. If an input variable arc is not used the blank space remains on the transition. If the input variable requires the absence of a token (i.e., $X1 = 0$) an inhibited input arc is used. Any arc from a place used elsewhere in the PrN model must be refreshed by using a return arc from the transition removing the token to the place holding the token before its removal. All of the transitions will be placed side by side and, because the transition set is mutually exclusive, only one transition can be enabled on any PrN pass through the REN. For the serial REN of rung 1 the right two transitions of Figure 5.4 represent deactivation. The PN representation of serial rung 3 deactivation is the right three transitions of Figure 5.5.

Justification of Selected Algorithm

The mathematical and tabular algorithms are equivalent and produce the same selection sets and Boolean expressions as demonstrated by the manipulation of serial rung one and serial rung three to convert RLL rungs to RENs. The algebraic method is time consuming for large sets of input variables. The tabular method allows larger input sets to be easily manipulated, reducing the computation time. The key to the choice of methods is in the automation of the solution for activation and deactivation expressions. Performing Boolean algebraic manipulations and displaying the results on an output screen is difficult if not impossible. Building and sorting tables can be implemented using standard computation algorithms [21].

CHAPTER 6. PETRI NET MODEL DEVELOPMENT

Serial Ladder Petri Net Model

Serial ladder rung one

This first rung has been given in the example material.

Serial ladder rung two

Rung two has two current output variables used in input, so there will be more arcs to be counted if these two variables appear in the expression.

The combinations Table 6.1 is a binary counting to seven as before.

The output combination of

$$Y1c \cdot \overline{Y3c} \cdot X3$$

produces an active output for row six of Table 6.1. Having only one row in the activation table we will need three variables to define it exclusively.

One input combination and seven output combinations produce three choices for sorting to define the most rows and two choices following each major sort. A total of six expressions are considered.

Table 6.1: Combinations generated by rung two input and output states

$Y1c$	$Y3c$	$X3$	$Y2f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

The single row requires three variables to define it exclusively. The phrase

$$Y1c \cdot \overline{Y3c} \cdot X3$$

defines the row. A total of five arcs will be required to define this activation transition: two for $Y1c$, one for $Y3c$, one for $X2$, and one for the activation arc to $Y1f$.

Figure 6.1 has one activation transition. Note the returning arc for $Y1c$ and no returning arc for $Y3c$. The $Y3c$ variable uses an inhibitor arc to enable the transition so no token is removed from the $Y3c$ place and no token need be returned. The activation arc to the future output place completes the transition.

Having seven deactivation rows to define, the largest quantity defined is four rows with one variable. This leaves three rows to be defined by two more phrases; one two variable and one three variable phrase. Because there are three variables in the single activation phrase there are three compliment variables to sort the maximum number of rows by. For each major sort a choice of two sorts using two variables creates six tables to consider. Table A.2 in Appendix A shows all six possible sorts with the number of arcs required to represent each expression in REN form. The table pairs A–B, C–D, and E–F each sort a different primary variable for the deactivation. The pair differ only in the secondary sort selection of variable pairs. The number of arcs required for the activation transition has been included, but will be a constant for all combinations considered having been fixed in the previous steps of the tabular algorithm.

Two combination tables produce sets requiring 12 arcs (including the activation arcs) to represent the REN. The choice is left to the user, and Table A.2 part C is chosen (the first one to appear). The expression

$$Y3c + \overline{Y3c} \cdot \overline{X3} + \overline{Y1c} \cdot \overline{Y3c} \cdot X3$$

contains three phrases to be converted to transitions.

Using the constructions previously discussed the three transitions on the right of Figure 6.1 are developed. The lack of arcs from the transitions to the future output indicates they are deactivation transitions.

Serial ladder rung three

This is another rung with a single row defining the activation of a transition. The procedure is similar to serial rung two. The deactivation transitions were defined as an example for the mathematical and tabular algorithms.

Only row eight of Table 6.2 (all inputs active) has a value of 1 for the output $Y3f$.

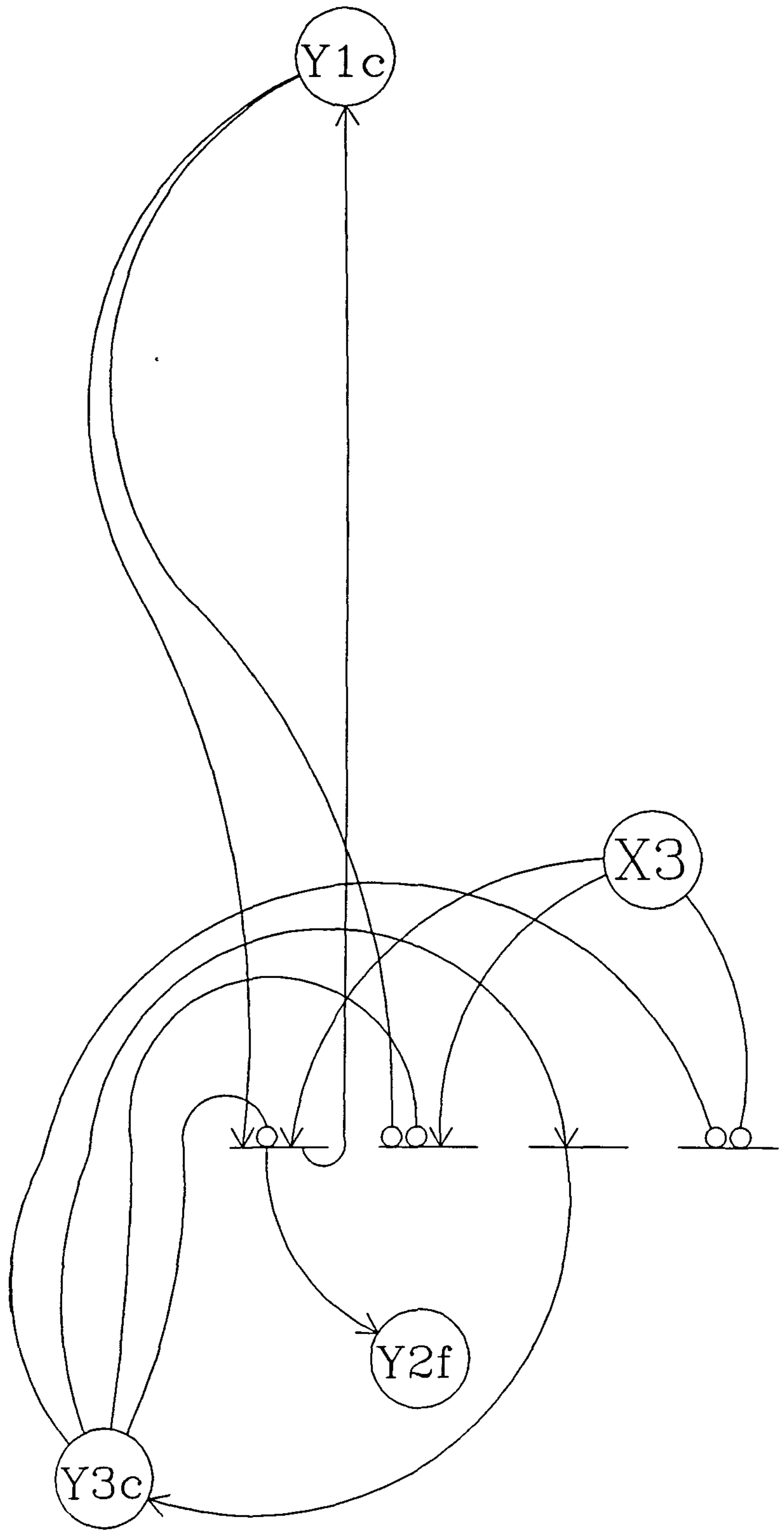


Figure 6.1: PN model of serial rung two

Table 6.2: Table for serial ladder rung three input values showing generated output values

$Y2c$	$X4$	$X5$	$Y3f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Sorting the rows of Table 6.2 moves the activation row to the top and provides three variables for the primary sort of the deactivation rows.

The activation phrase

$$Y2c \cdot X4 \cdot X5$$

defines the single activation row.

The left transition of Figure 5.5 is the activation transition for the above phrase. A total of five arcs are required to generate the transition: two for $Y2c$, one for $X4$, one for $X5$, and one for the activation arc to place $Y1f$.

The six tables generated by the deactivation portion of the tabular algorithm are shown in Table A.3. The arc requirements are listed to the right of the combinations generated. A minimum of eleven arcs are required to construct the transitions, and two combinations of output deactivation variables (Table A.3 C and F) each generate

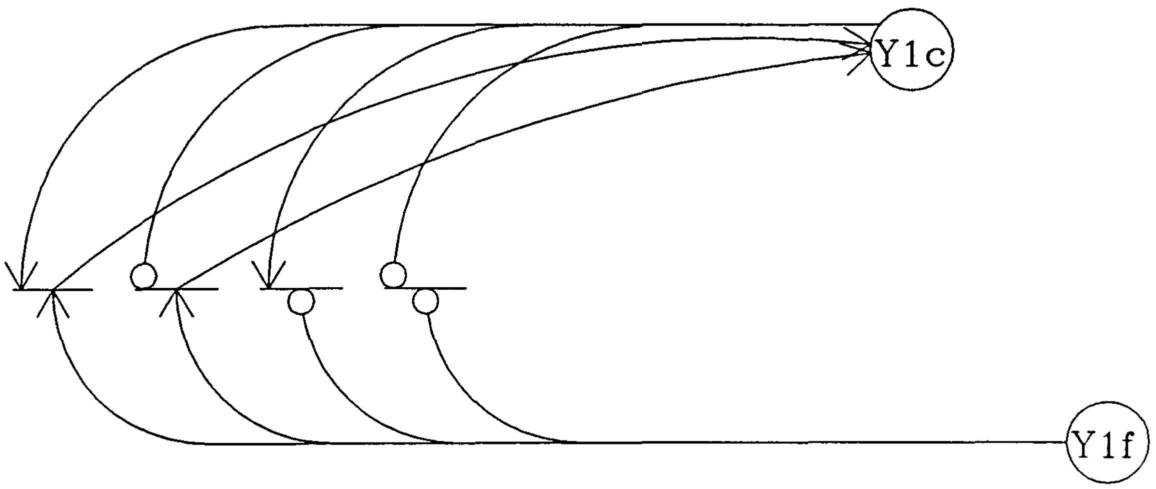


Figure 6.2: Output evaluation network

eleven arcs. Choice is left to the user and Table A.3 C (the first one generated) is chosen.

The three transitions representing the expression

$$\overline{X4} + X4 \cdot \overline{X5} + \overline{Y2c} \cdot X4 \cdot X5$$

from Table A.3 C are shown in Figure 5.5 without arcs to the future output place.

Output evaluation

Figure 6.3 includes a PN not yet discussed. The output evaluation net shown in Figure 6.2 tests for the 4 possible combinations of markings current and future output places and adds or consumes the token as required. This output evaluation network is the same for each rung. For future work connecting the outside world inputs to the internal representations, this same sort of comparison can be made to determine the absence or presence of a token in an input prior to the next pass through the REN ladder scan. Each pass through the output evaluation for a REN will drain the

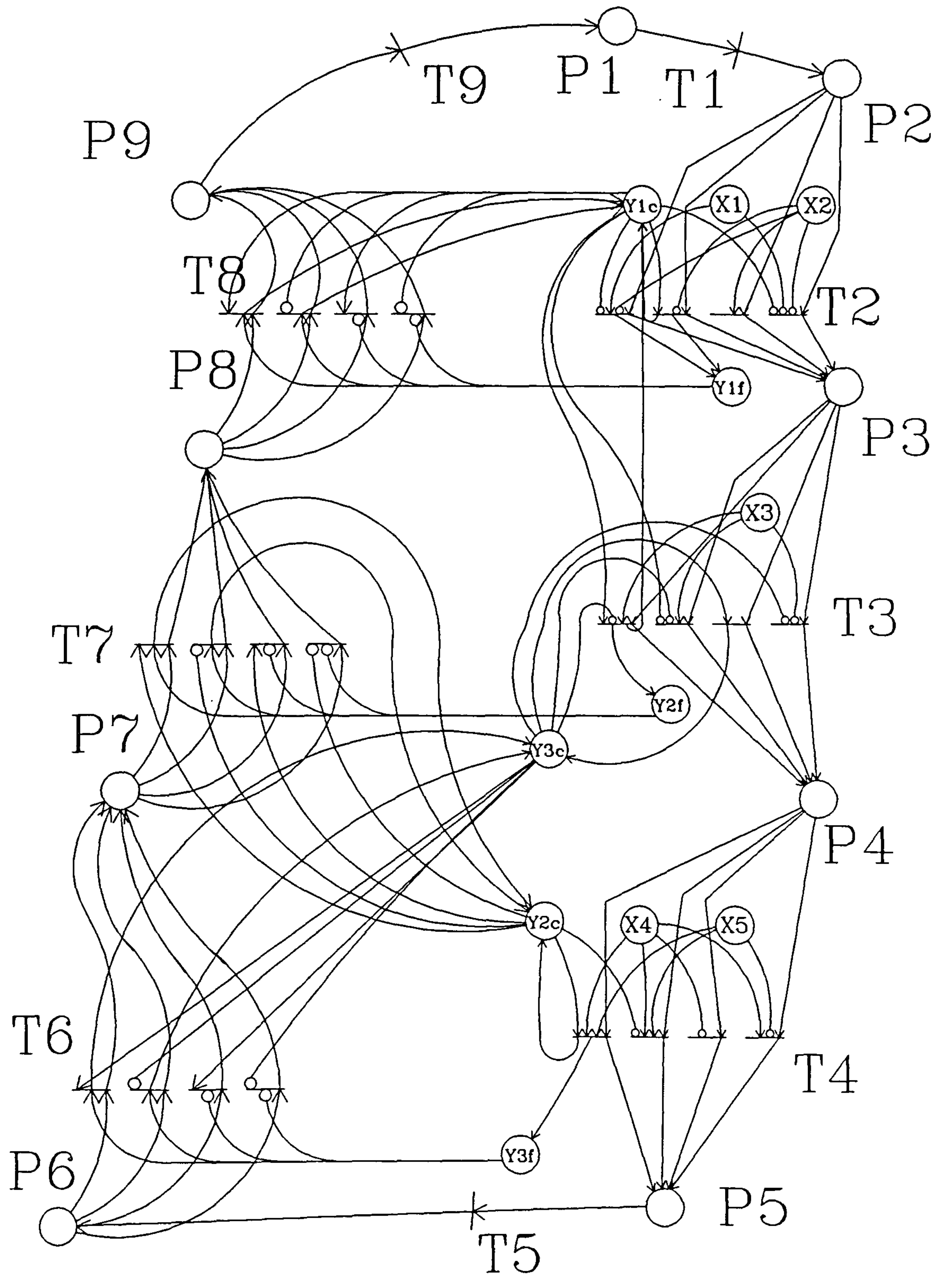


Figure 6.3: Complete PN model of serial relay ladder logic

future register just as each pass through the RENs drains all X registers not used elsewhere in the PrN or RENs. It should be stated here that the $Y1f$ status controls the updated status of $Y1c$. Whatever the token content of $Y1f$ is at the time of output analysis this will be the token count of $Y1c$ after output analysis. This should indicate a simpler route to update testing. However, just as we must pass tokens for the REN even though there is no change of state, we must also pass tokens through the output evaluation net for no change of state. In addition, to preserve safeness, the tokens existing in $Y1c$ must be removed before the new token can be placed from $Y1f$. So, the need for 4 transitions to test the two places is inherent.

Parallel Ladder Petri Net Model

Since the PrN for parallel and serial RLL models have the same place and transition values, the evaluation of the two models will be the same. The differences in the drawing shape only serve to display the new model in a more visual final form. The states and events for the serial and parallel PrNs are exactly the same. Figure 6.4 is the PrN for the parallel RLL model, and Table 6.3 shows the places and transitions and their respective state or event representations.

Unlike the wall mounted predecessor to RLL in PLCs, the parallel operation of RLL is actually a sequential evaluation. Large RLL diagrams on a slow CPU would demonstrate the serial nature of parallel ladders. The serial nature of parallel ladder processing is visible in the PrN for parallel RLL logic shown in Figure 6.4 demonstrating the evaluation of each rung then the sequential updating of outputs. True parallel process development from parallel RLL and for multiple functions programmed to the

Table 6.3: Transition and place representations for parallel PrN

Place	State represented
P1	Ladder scan ready
P2	Rung 1 evaluation ready
P3	Rung 2 evaluation ready
P4	Rung 3 evaluation ready
P5	Ladder scan complete; Output evaluation ready
P6	Rung 3 output scan ready
P7	Rung 2 output scan ready
P8	Rung 1 output scan ready
P9	Output scan completed
Transition	**Event(s) represented**
T1	Start ladder scan, Gather inputs
T2	Evaluate rung 1
T3	Evaluate rung 2
T4	Evaluate rung 3
T5	Mark all output scan places
T6	Evaluate output rung 3
T7	Evaluate output rung 2
T8	Evaluate output rung 1
T9	Report current output statuses; Set ladder scan ready

same PLC and running contiguously are left to future work.

Parallel rung one

The parallel start/stop rung is the same as the serial start/stop rung. The REN drawing appears to be different, but the expression and the four transitions used to activate and deactivate $Y1f$ are the same.

Parallel rung two

The eight binary combinations produce the first three columns of Table 6.4. The output for Table 6.4 is the fourth column. Only row seven activates future

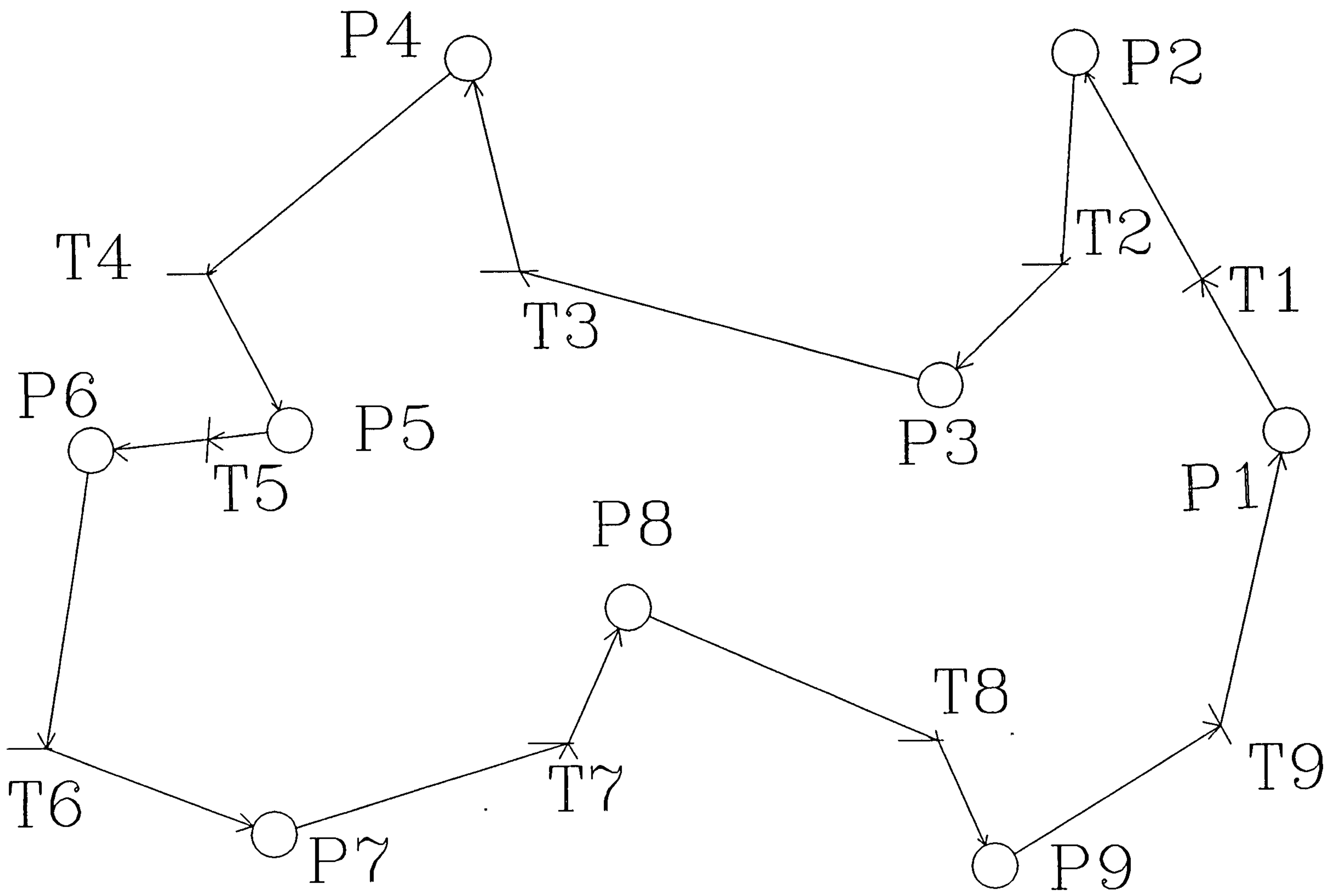


Figure 6.4: PrN model for the parallel RLL model

Table 6.4: Parallel rung two unsorted combinations table

$Y1c$	$X3$	$X4$	$Y2f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

output when $Y1c$ and $X3$ are active and $X4$ is inactive.

Having only one activation row requires three variables to define the row uniquely. The phrase for activation of output is

$$Y1c \cdot X3 \cdot \overline{X4}.$$

For the above phrase one transition will be needed. The transition will have three arrowhead input arcs: two from input variable places and one from the PrN; one inhibitor arc from the input variable place $X4$; and three output arcs: one to the future output place $Y2f$, one returning to the current output place $Y1c$, and one to the PrN. All transitions will have the same location relationship. If a variable is not needed in the enabling of a transition, a blank space remains where the input arc would have been. These are the same rules for constructing transitions used throughout the paper.

Three variables have only one value in the input rows leaving three choices of complimented variable values to sort output columns. These three choices can be

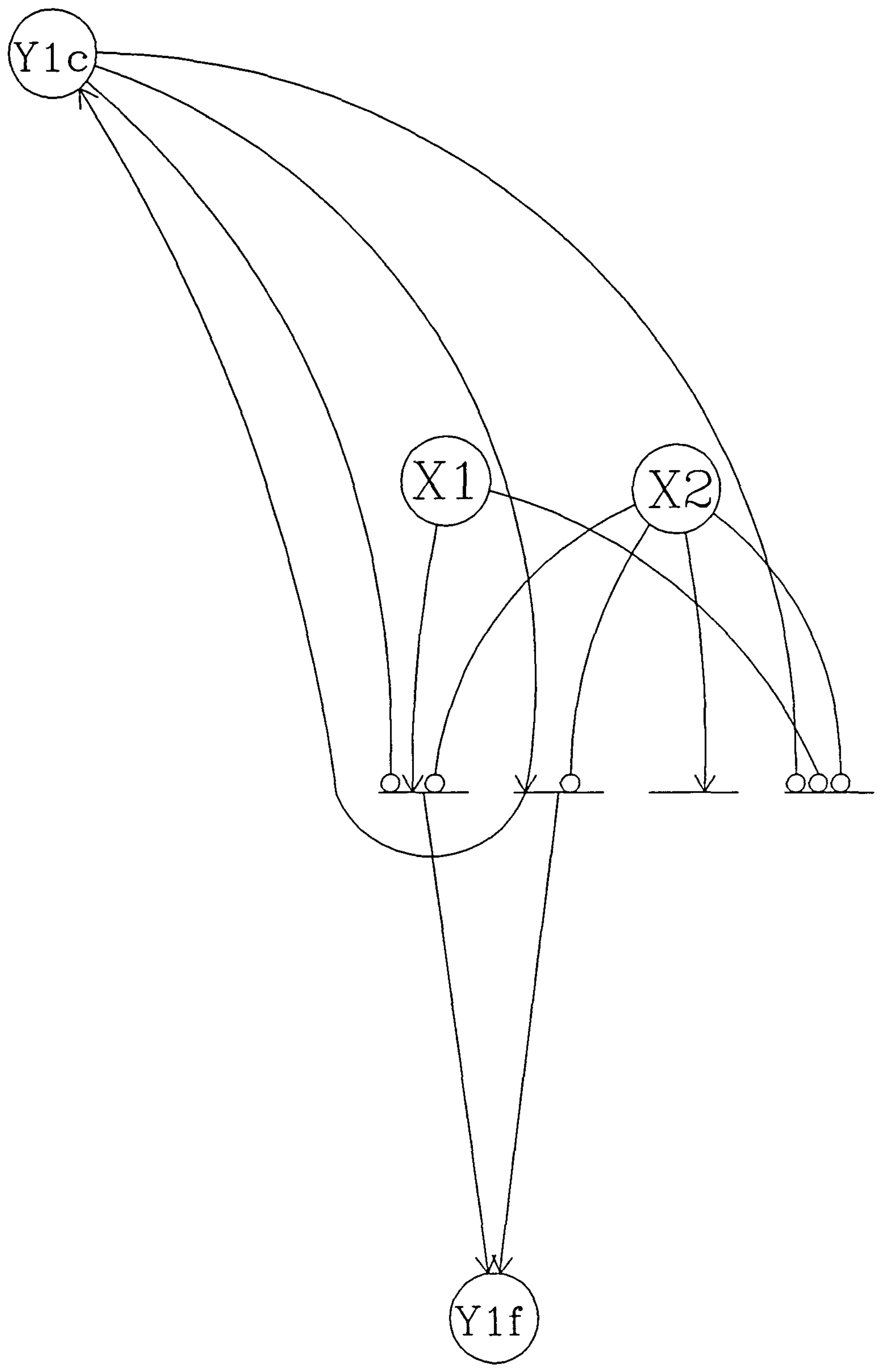


Figure 6.5: PN model of parallel rung one

sorted again by a pair of variables giving two more combinations to choose from. The total of six tables sorted for all combinations of input arcs and optimized for the least number of arcs required is presented in Table A.4 of the Appendix. The minimum number of arcs required to produce the output transitions is eleven. There are two expressions (Table A.4 D and F) satisfying the optimization of arc quantity. The selection is left to the user and Table A.4 D is used for the parallel rung two representation of output transitions. The expression

$$X4 + \overline{X3} \cdot \overline{X4} + \overline{Y1c} \cdot X3 \cdot \overline{X4}$$

as defined by Table A.4 D in the Appendix must now be converted to transitions.

Three transitions are built from the expression. The three transitions on the right side of Figure 6.6 are for deactivation of output. The input arrowhead arcs from the PrN are not shown in this drawing but the space for them on the far right of each transition is.

Parallel rung three

The similarity between serial rung 3 and parallel rung 3 is very noticeable. After all of the tables are built and sorted, the resemblance becomes more apparent. Except for a change of variable names, the tables are exactly the same. This gives rise to future work investigating the development of tables for any number of variables in advance so the computational work of automation becomes one of look up rather than plug and grind.

The activation row for rung three is row 8 of Table 6.5. This value of 1 is gained by evaluating rung three of the parallel RLL model for each of the input combinations

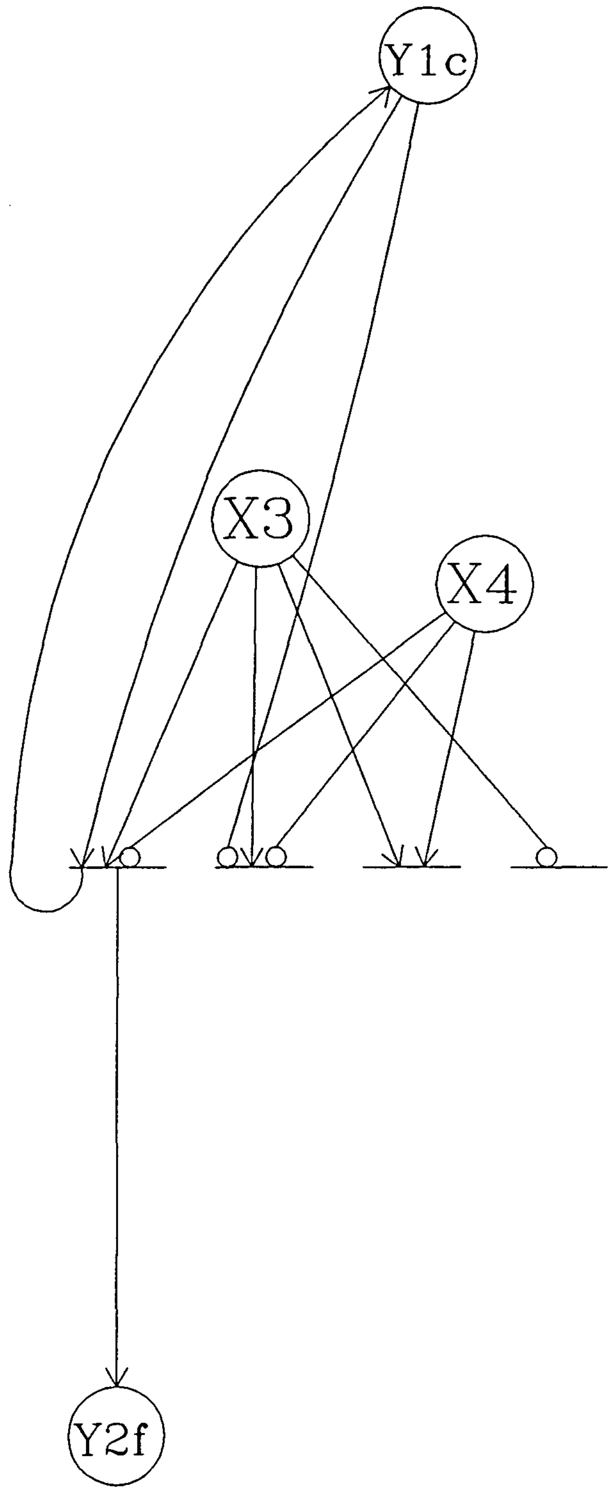


Figure 6.6: PN model of parallel rung two

Table 6.5: Parallel rung three unsorted combinations table

$Y1c$	$X5$	$X6$	$Y3f$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 6.6: Sorted parallel rung three combinations table

$Y1c$	$X5$	$X6$	$Y3f$
1	1	1	1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0

using Boolean logic to produce the output value.

The sorted Table 6.6 raises the activation output value to the top of the table and filters all of the deactivation rows to the bottom.

The only row with an output value equal to one requires three variables to define the row. The phrase

$$Y1c \cdot X5 \cdot X6$$

defines the requirements of the row. Five arcs will be needed to build the transition representing the phrase.

The first transition of Figure 6.7 is the activation transition for future output. The blank space on the right side of the transition is for the PrN input arc to be added later.

As in the evaluation of serial rung three, six combinations are required to represent all possible sorting combinations. The combinations satisfy the requirements of representing the most rows with the least number of variables. The six combinations are shown in Table A.5. Each expression has 3 phrases thus 3 transitions are required; the least number of arcs required to represent the activation and deactivation phrases (excluding PrN arcs) is eleven. Two sorted tables (Table A.5 C and F) represent this rung equally, and the choice is left up to the user. The first expression generated:

$$\overline{X5} + X5 \cdot \overline{X6} + \overline{Y1c} \cdot X5 \cdot X6$$

from Table A.5 C will be converted to transitions.

The three transitions, representing each ORed phrase of the above expression, are shown in Figure 6.7 on the right side of the drawing. They are deactivation transitions and have no arc from the transition to the future output place $Y3f$. The open spaces on the first two places of the second transitions are for the unused arcs from $Y1c$ and $X5$.

The three developed RENs are combined into the parallel PrN to produce a complete PN representation of parallel RLL. Although the serial and parallel drawings look very different, the two system's PrNs are exactly alike in operation. Figure 6.8 shows the parallel model including output evaluation nets.

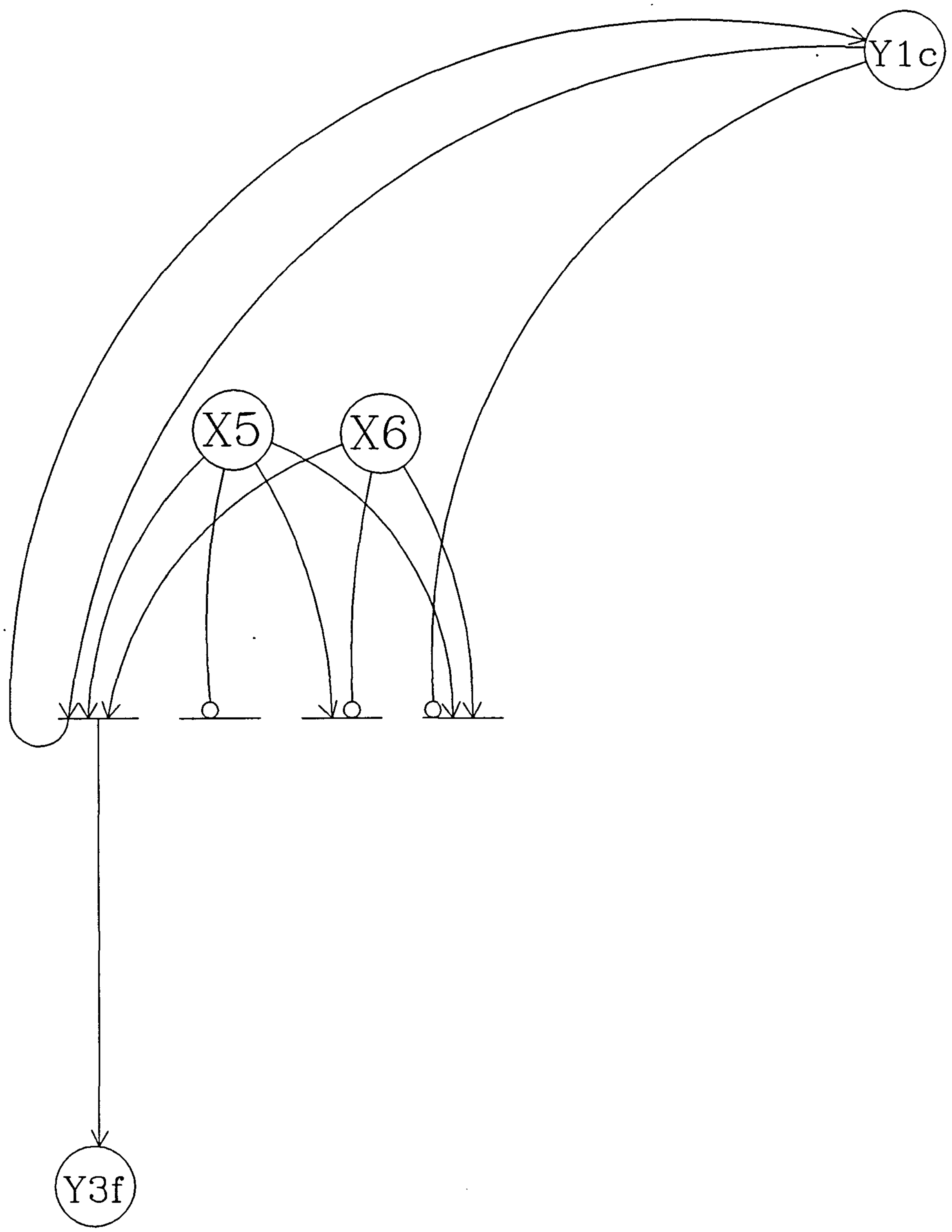


Figure 6.7: PN model of parallel rung three

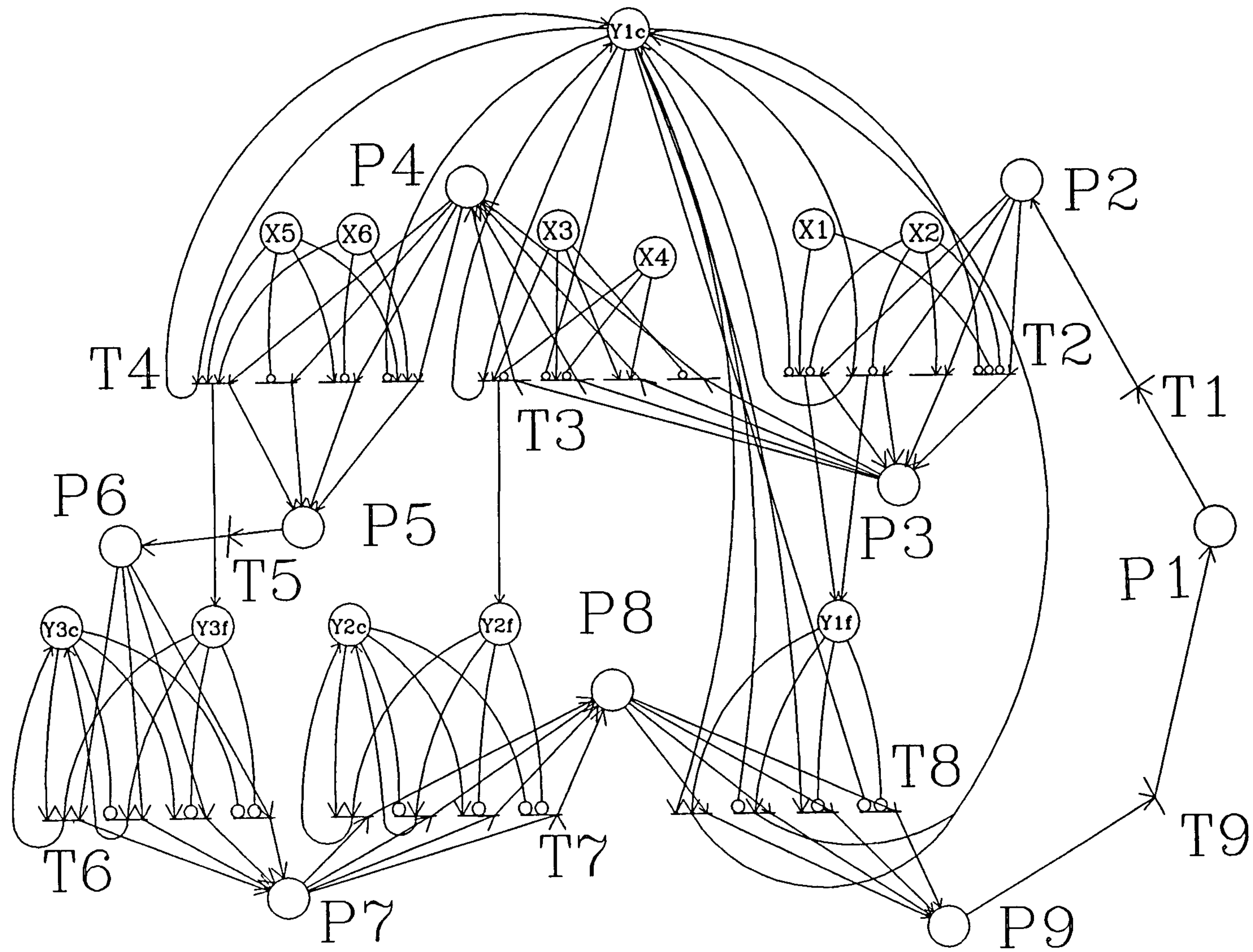


Figure 6.8: Complete parallel RLL PN model representation including PrN, RENs, and output evaluation nets

CHAPTER 7. TESTING AND DEMONSTRATION OF MODELS

The PN model must be demonstrated as functioning the same as the RLL system. There are several approaches to this demonstration. The most obvious and most tedious is to test the models using exhaustive enumeration for every switching condition. With the serial model using 5 inputs and 3 outputs there are 2^8 combinations (256) to test. An alternative would be to test only the conditions which will activate the RLL model. This borders on the realm of programming for known conditions as mentioned in the introduction. The parallel model has 6 inputs and 3 outputs resulting in 512 conditions to be tested. Appendix B contains the extensive tables developed from the evaluation of the models.

RLL outcomes are determined for each combination of input and output variables. The Petri net models are tested after the token reaches place P1 of Figures 5.1 and 6.4 (Ladder scan ready) because this is where outputs have been reported to the outside world via transition T9. The model evaluation tables in Appendix B show the values of the X and Y variables from the external perspective. The process followed to test the PrN and REN systems tracks these internally. The Y variable reported is the updated $Y1c$ variable maintained internally. Multiple PrN passes are shown as additional Y combinations to the right of the column labeled scan one in the tables of Appendix B if necessary. Internal X values at P9 are negligible because they will be refreshed at T1 on the next scan of the rungs. The system never really reaches steady state because outputs will trigger activity externally and alter the X inputs for the next scan.

As stated in the output evaluation explanation at the end of the serial PN model construction, the X input places are compared with outside world inputs at transition T1 in much the same fashion as the output future and current places are compared. Any tokens remaining in input places as T1 fires are refreshed or removed prior to beginning the ladder scan. It is the nature of the output evaluation nets to drain the future output places, so they will always have a 0 marking upon beginning a ladder scan.

The RLL and PN models were tested using the enumeration method described above. After each pass through the pertinent model, a snapshot of the Y results was taken at P1 and compared with the output of the RLL model.

One example of the process of testing PN logic for each model will be included to demonstrate the process to the reader, the bulk of the processing is left to the author and reported in Appendix B.

Serial Model

The output results and number of passes to reach steady state for both the serial RLL and PN models are the same. The PN testing matched the RLL outcomes for every initial state tested.

RLL model results

Using a combination of 5 inputs represented by the X values, a potential 64 combinations result. This applied to the three output combinations for initial conditions results in 2^{5+3} combinations. Using the three output initial combinations as a basis of comparison seeking a final combination, the 64 possible input initial settings are evaluated. The result is eight possible tables of final values. This is not surprising as the eight tables relate to the three rungs possible outcomes of on or off thus 2^3 or eight tables.

Taking each of the input combinations and looking at the output results after the first scan through the table shows if another scan is necessary. If a pattern or constant value has not resulted, another scan is necessary. If more than one scan was required to develop the steady state, the changes in the combinations are shown in the following scan. If no change occurred in succeeding scans, then the area was left blank. As many as four scans are required to reach steady state. The left column of the lower half of each table represents all of the possible input combinations. It also serves as the look up column for each succeeding scan. The complete table of test results can be see in Appendix B Table B.1.

One initial condition creates a repeating pattern. This condition is 10111 for the X values. This pattern would be called the process in operation being controlled by the RLL. No matter what state the output variables started in, they generate the pattern of operation 100, 110, 111, 101, 100. Twenty-four initial X combinations reach a steady state of 000; four reach a steady state of 100, three initial condition reach a steady state of 110, and one pattern repeats as stated above. These thirtytwo initial X conditions are tested over the eight initial conditions of the Y output statuses.

PN model results

The repeating pattern for the initial X values of 10111 is a good system to demonstrate the path through the PN model. The complete path for four scans is shown in Appendix B as Table B.3. In addition to the internal representations of the X, $Y1c$, and $Y1f$ values, a column on the left side of the table demonstrates the transition that fired to produce the internal values. At T9 the outputs are reported to the outside world or the right side of the table.

Parallel Model

RLL model results

Adding one X variable doubles the number of initial conditions to be checked. Still, it only produces five steady state conditions. Because of the nature of the RLL there is no repeating pattern. Forty-eight initial X conditions reach a steady state Y output condition of 000, nine reach 100, three reach 101, three reach 110, and one condition reaches 111. The patterns and number of scans required to reach steady state are shown in Appendix B Table B.2. Each of the sixty-four initial X conditions are tested against all eight Y initial conditions. Additional columns of Y output are scans required beyond one to reach a steady state. Each set of three numbers represents the Y1, Y2, and Y3 outputs at the end of that scan.

PN model results

For the parallel REN model, only three passes are needed for any input initial state to reach a steady state operation. This system operates like the serial PN model because the PrN and output evaluation networks operate the same and have the same structure. All output values are measured at Transition nine after a ladder scan and output evaluation have been made. Table B.4 in Appendix B shows the Transition and output status for the initial input values 101011 (all rungs activated). This table makes three scans to prove steady state and corresponds to the output results of the parallel RLL analysis of Table B.2 in Appendix B for the same initial inputs and initial output state 000. The results for the parallel model external output reporting is identical to Table B.2 so is not included in the interest of saving paper and time.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

Conclusions

The work developed here is only the beginning of a larger investigation. Although the initial investigation is sound, the additional elements found in a complex PLC today will require more development for a complete PN representation of PLCs and the RLL driving them.

Although the RLL diagrams are more pleasing to the eye, the complete logic of the system is not present to the novice reader and easily forgotten by an experienced user of RLL. The PN representations are more communicative in the function of the complete PLC system and operation.

This research develops both mathematical and tabular algorithms for conversion of simple RLL diagrams using a limited set of rung elements. The tabular algorithm is shown to have basis in Boolean logic using the mathematical algorithm presented here also. The tabular algorithm is the preferred method when using large numbers of variables and for automation of the sorting procedures.

Two RLL models of three rungs each are presented to test the algorithm. Each model represented a different type of rung dependence. The serial RLL model acted as a sequencer and each succeeding rung relied upon the earlier results. The parallel

RLL model uses the same first rung but allowed the second and third rung to operate independently of each other relying only on rung one.

Rung evaluation nets (RENs) were generated for each rung of the two models and a PN model of the PLC processing was built. In order to completely model the RLL logic of a PLC, a higher network (PrN) system and output evaluation analysis networks were added to the RENs. After demonstrating the algorithm and generating the PN models of RLL, testing and evaluation demonstrated that the RLL and PN representations were equivalent.

Future Work

The direction of this thesis has been to model RLL using PN logic. The next step would be to take RLL and automate the conversion to PN on a computer by directly reading the RLL graphics and data files. This automation would be developed for each type of equipment being used.

Additional studies on the Parallel networking of systems are needed and their relationships to RLL rungs of PLC. In these system, PN diagraming becomes more useful in providing a clear picture of the functions being controlled and their interdependent or independent relationships.

Connection to the outside world of the variables shown in the PN representation at T1 and T9 would involve testing for the presence/absence of tokens and updating them at the proper transitions in the PrN in a similar manner to the output evaluation nets of the PN models demonstrated.

The use of counters and timers in PLCs is common, and development of these devices would be needed if a PN system was to be useful in system design. Several paths exist for this work, but timed Petri net representations for various timers found in RLL systems seem to be the most logical.

Any automation system would have to include the automation of RLL to REN conversion. In addition to the basic automation of the manipulation of tabular solutions, evaluation of arc optimization, and creation of transition phrases the creation of the PN representation graphics is yet undeveloped. This would involve considerable rule writing for the graphics based systems including the PrN consideration. The automation of any of these components will have to consider the multiple use of input variables in the RLL to allow for arc optimization and graphic design layouts. Complete automation of the RLL graphic to PN graphics opens many doors of investigation.

Analysis of the Petri nets created to determine safeness, boundedness, and reachability trees is an important aspect of creating the PN representations of the RLL systems. Several areas of research are open here. The analysis of the PrN would include reachability trees to determine deadlock and system states reaching infinity. Testing and simulation using the PN representations is also a rich field of study.

BIBLIOGRAPHY

[1] Agerwala, T. "Some Applications of Petri Nets"; *Proceedings of the National Electronics Conference: Volume XXXII*; Tranter, W. H., Ed.; National Electronics Consortium, Inc.: Chicago, Ill.; 1978; Vol. 32, pages 149-154.

[2] Alanche, P.; Benzakour, K.; Dolle, F.; Gillet, P.; Rodrigues, P. and Valette, R. "PSI: A Petri Net Based Simulator for Flexible Manufacturing Systems"; *Lecture Notes in Computer Science: Advances in Petri Nets 1984*; Springer Verlag: Berlin, West Germany; 1985; Vol. 188, pages 1-14.

[3] Alla, H.; Ladet, P.; Martinez, J. and Silva-Suarez, M. "Modelling and Validation of Complex Systems by Coloured Petri Nets: Applications to a Flexible Manufacturing System"; *Lecture Notes in Computer Science: Advances in Petri Nets 1984*; Springer Verlag: Berlin, West Germany; 1985; Vol. 188, pages 15-29.

[4] Balbo, G.; Chiola, G.; Franceschinis, G.; and Roet, G. Molinar. "Generalized Stochastic Petri Nets for the Performance Evaluation of FMS"; *Proceedings of the 1987 Conference on Robotics and Automation*; IEEE Computer Society Press: New York, New York; 1987; Vol. 2, pages 1013-1018.

[5] Beck, C.L. and Krogh, B.H. "Models for Simulation and Discrete Control of Manufacturing Systems"; *Proceedings of Conference on Robotics and Automation*; IEEE Computer Society Press: New York, New York; 1986; Vol. 1, pages 305-310.

[6] Brand, K.P. and Kopainsky, J. "Principles and Engineering of Process Control with Petri Nets"; *IEEE Transactions on Automatic Control*; 1988; AC33, No. 2, 138-149.

[7] Bruno, G. and Marchetto, G. "Process-translatable Petri Nets for the Rapid Prototyping of Process Control Systems"; *IEEE Transactions on Software Engineering*; 1986; SE12, No. 2, 346-357.

[8] Bruno, G. and Morisio, M. "Petri Net Based Simulation of Manufacturing Cells"; *Proceedings of the Conference on Robotics and Automation*; IEEE Computer Society Press: New York, New York; 1987; Vol. 2, pages 1174-1179.

[9] Cherba, D. M. "Reducing Engineering Time for the Development of Ladder Diagrams"; *Control Engineering*; April 1987, 34, No. 4, 125-128.

[10] Chocron, D. and Cerny, E.A. "Petri Net Based Industrial Sequencer"; *IECI Proceedings of Conference on Applications of Mini and Microcomputers*; IEEE Press: New York, New York; 1980; Vol. 1, pages 18-22.

[11] Courvoisier, M.; Valette, R.; Bigou, J. M.; and Esteban, P. "A Programmable Logic Controller Based on a High Level Specification Tool"; *Proceedings of a Conference on Robotics and Automation 1987*; IEEE: New York, New York; 1987; pages 174-179.

[12] Crockett, D.H.; Desrochers, A.A.; DiCesare, F. and Ward, T. "Implementation of a Petri Net Controller for a Machining Workstation"; *Proceedings of the 1983 Conference on Robotics and Automation*; IEEE Society Press: New York, New York; 1987; Vol. 3, pages 1861-1867.

[13] Devanathan, R.; Kuan, F.Y.; Chang, C.J. and Choo, S.A. "Computer Aided Designing of Relay Ladder Logic via State Transition Diagram"; *International Conference on Industrial Electronics, Control and Instrumentation*; IEEE: New York, New York; 1987; Vol. 2, pages 764-772.

[14] Gentina, J.C. and Corbeel, D. "Coloured Adaptive Structured Petri Net: A Tool for the Automatic Synthesis of Hierarchical Control of Flexible Manufacturing Systems"; *International Conference on Robotics and Automation*; IEEE: New York, New York; 1987 Vol. 3, pages 1166-1173.

[15] Hack, M.H.T. "Analysis of Production Schemata by Petri Nets"; *Technical Report 94 Project MAC*; Massachusetts Institute of Technology: Cambridge, Massachusetts; 1972, 110 pages.

[16] Han, Y.W. "Performance of a Digital System Using a Petri Net- like Approach"; *Proceedings of the 1978 National Electronics Conference*; Tranter, W. H., Ed.; National Electronics Consortium, Inc.: Chicago, Ill.; 1978; Vol. 32, pages 166-172.

[17] Heimerdinger, W.L. "A Petri Net Approach to System Level Fault Tolerance Analysis"; *Proceedings of the 1978 National Electronics Conference*; Tranter,

W. H., Ed.; National Electronics Consortium, Inc.: Chicago, Ill.; 1978; Vol. 32, pages 161-165.

[18] Houldsworth, P.A. and Brearly, D. "Programmable Controller Functions are Enhanced by Structured Programming and Graphic Sequence Control Together with Good PC Documentation"; *Proceedings of the Conference on Programmable Controllers 1985*; Peter Lawrenson, Editor; 1985; GAMBICA Programmable Controllers Committee: London, England; pages 129-134.

[19] Kamath, M. and Viswanadham, N. "Applications of Petri Net Based Models in the Modelling and Analysis of Flexible Manufacturing Systems"; *Proceedings of the 1986 International Conference on Robotics and Automation*; IEEE Computer Society Press: New York, New York; 1986; Vol. 1, pages 312-317.

[20] Kissel, T.E. *Understanding and Using Programmable Controllers*; Prentice Hall: Englewood Cliffs, New Jersey; 1986.

[21] Knuth, D.E. *The Art of Computer Programming*; "Volume 3: Sorting and Searching"; Addison-Wesley: Reading, Massachusetts; 1973.

[22] Komoda, N.; Murata, T. and Matsumoto, K. "Petri-Net Based Controller: SCR and its Applications in Factory Automation"; *IEEE International Symposium on Circuits and Systems*; IEEE: New York, New York; 1985; Vol. 2, pages 937-940.

[23] Krogh, B.H. and Beck, C.I. "Synthesis of Place/Transition Nets for Simulation and Control of Manufacturing Systems"; *Fourth Symposium on Large Scale Systems: Theory and Applications*; Zurich, Switzerland, IFAC/IFORS; Pergamon Press: Oxford, England; August 1986; Vol. 2, pages 583-589.

[24] Krogh, B.H.; Willson, R.; and Pathak, D. "Automatic Generation of Control Programs for Discrete Manufacturing Processes"; *The Robotics Institute Annual Research Review*; Carnegie Mellon University; 1987, pages 21-31.

[25] Kruempel, G.E. and Day, A.L. "Personal Computer Aided Design and Simulation of Programmable Controller Programs"; *IEEE Industry Applications Society Annual Meeting 1987*; IEEE: New York, New York; 1987; Vol. 2, pages 1764-1767.

[26] LeMer, E. "OVIDE: A Software Package for Verifying and Validating Petri Nets"; *Third IFAC/IFIP Symposium*; Pergamon Press: Oxford, England; 1982; pages 255-260.

[27] Lloyd, M. "GRAFCET - Graphical Function Chart Program"; *Proceedings of the Conference on Programmable Controllers 1985*; Peter Lawrenson, Editor; GAMBICA Programmable Controllers Committee: London, England; 1985; pages 51-56.

[28] Martinez, J.; Alla, H. and Silva, M. "Petri Nets for the Specification of Flexible Manufacturing Systems"; *Modelling and Design of Flexible Manufacturing Systems*; Kusiak, A. Editor; Elsevier Science Publishers: Amsterdam; 1986; Chapter 8, pages 389-406.

[29] Martinez, J.; Muro, P. and Silva, M. "Modeling, Validation and Software Implementation of Production Systems using High Level Petri Nets"; *Proceedings of the 1987 Conference on Robotics and Automation*; IEEE Computer Society Press: New York, New York; 1987; Vol. 3, pages 1180-1185.

[30] Martinez, J. and Silva, M. "A Package for Computer Design of Concurrent Logic Systems"; *Proceedings of the Third IFAC/IFIP Symposium on Software for Computer Control*; Pergamon Press: Oxford, England; 1982; pages 243-248.

[31] Martinez, J. and Silva, M. "A Language for the Description of Concurrent systems Modelled by Colored Petri Nets: Applications to the Control of Flexible Manufacturing systems"; *Languages for Automation*; Shi–Kuo Chang, editor; Plenum Press: New York, New York; 1985; Chapter 8, pages 369-388.

[32] Matsuzaki, K.; Hata, S.; Junichi, H.; Kurashima, Y. and Torii, M. "Petri-Net Structured Sequence-Control Language with GRAFCET-like Graphical Expression for Programmable Controllers"; *Proceedings of International Conference on Industrial Electronics, Control and Instrumentation*; IEEE Press: New York, New York; 1985; Vol. 1, pages 433-438.

[33] Murata, T.; Komoda, N. and Matsumoto, K. "A Petri Net Based Factory Automation Controller for Flexible and Maintainable Control Specifications"; *Proceedings of Conference on Industrial Electronics, Control and Instrumentation*; IEEE Press: New York, New York; 1984; Vol. 1, pages 362-366.

[34] Murata, T.; Komoda, N.; Matsumoto, K. and Haruna, K. "A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation"; *IEEE Transactions on Industrial Electronics*, February 1986; IE33, No. 1, 1-8.

[35] Narahari, Y. and Viswanadham, N. "A Petri Net Approach to the Modelling and Analysis of Flexible Manufacturing Systems"; *Annals of Operations Research*; Baltzer: Basel, Switzerland; 1985; Vol. 3, pages 449- 472.

[36] Peng, D. and Shin, K.G. "Modeling of Concurrent Task Execution in a Distributed System for Real-Time Control"; *IEEE Transactions on Computers*; April 1987; C36, No. 4, 500-516.

[37] Peterson, J.L. "An Introduction to Petri Nets"; *Proceedings of the National Electronics Conference: Volume XXXII*; Tranter, W. H., Ed.; National Electronics Consortium, Inc.: Chicago, Ill.; 1978; Vol. 32, pages 144-148.

[38] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*; Prentice Hall: Englewood Cliffs, N. J.; 1981.

[39] Petri, C.A. "Kommunikation mit Automaten," Ph.D. dissertation, University of Bonn: Bonn, West Germany, 1962.

[40] Reisig, W. *Petri Nets: an Introduction*; Springer Verlag: Berlin, West Germany; 1985.

[41] Sacks, T. "Is it Time to Step off of the Ladder?"; *Electrical Review (London)*; 9-22 March, 1988; 122; 23-25.

[42] Silva, M. and Velilla, S. "Programmable Logic Controllers and Petri Nets: A Comparative Study"; *IFAC Proceedings of Software for Computer Control SOCOCO '82*; Pergamon Press: Oxford, England; 1982; pages 83-88.

[43] Valette, R.; Courvoisier, M.; Bigou, J.M. and Albukerque, J.A. "Petri Net based Programmable Logic Controller"; *Computer Applications in Production and Engineering (CAPE 83)*; E.A. Warman, Editor; North-Holland Publishing: Amsterdam, The Netherlands; 1983; pages 103-116.

APPENDIX A. COMBINATION TABLES

Table A.1: Table for serial or parallel rung one

A*					B				
$Y1c$	$X1$	$X2$	$Y1f$	Arcs	$Y1c$	$X1$	$X2$	$Y1f$	Arcs
1	0	0	1		1	1	0	1	
1	1	0	1	3	0	1	0	1	2
0	1	0	1	3	1	0	0	1	4
0	1	1	0		0	1	1	0	
1	0	1	0		1	0	1	0	
1	1	1	0		1	1	1	0	
0	0	1	0	1	0	0	1	0	1
0	0	0	0	3	0	0	0	0	3
			Total	10				Total	10

Table A.2: Rung two table reordered tables showing six mutually exclusive alternatives and their arc requirements

A					B				
$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs	$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs
1	0	1	1	5	1	0	1	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
0	1	0	0		0	1	0	0	
0	1	1	0	1	0	1	1	0	1
1	1	0	0		1	1	0	0	
1	1	1	0	4	1	0	0	0	3
1	0	0	0	4	1	1	1	0	5
			Total	14				Total	14
C*					D				
$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs	$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs
1	0	1	1	5	1	0	1	1	5
0	1	0	0		0	1	0	0	
0	1	1	0		0	1	1	0	
1	1	0	0		1	1	0	0	
1	1	1	0	2	1	1	1	0	2
0	0	0	0		0	0	0	0	
1	0	0	0	2	0	0	1	0	2
0	0	1	0	3	1	0	0	0	4
			Total	12				Total	13
E					F				
$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs	$Y1c$	$Y3c$	$X3$	$Y2f$	Arcs
1	0	1	1	5	1	0	1	1	5
0	0	0	0		0	0	0	0	
1	0	0	0		1	0	0	0	
0	1	0	0		0	1	0	0	
1	1	0	0	1	1	1	0	0	1
0	1	1	0		0	1	1	0	
0	0	1	0	2	1	1	1	0	3
1	1	1	0	5	0	0	1	0	3
			Total	13				Total	12

Table A.3: Six tables showing all possible combinations generated for deactivation of the future output of serial ladder rung three

A					B				
$Y2c$	$X4$	$X5$	$Y3f$	Arcs	$Y2c$	$X4$	$X5$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
0	1	0	0		0	1	0	0	
0	1	1	0	1	0	1	1	0	1
1	1	0	0		1	0	1	0	
1	0	0	0	3	1	0	0	0	3
1	0	1	0	4	1	1	0	0	4
			Total	13				Total	13

C*					D				
$Y2c$	$X4$	$X5$	$Y3f$	Arcs	$Y2c$	$X4$	$X5$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
1	0	0	0		1	0	0	0	
1	0	1	0	1	1	0	1	0	1
1	1	0	0		0	1	1	0	
0	1	0	0	2	0	1	0	0	2
0	1	1	0	3	1	1	0	0	4
			Total	11				Total	12

E					F				
$Y2c$	$X4$	$X5$	$Y3f$	Arcs	$Y2c$	$X4$	$X5$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
1	0	0	0		1	0	0	0	
0	1	0	0		0	1	0	0	
1	1	0	0	1	1	1	0	0	1
0	1	1	0		1	0	1	0	
0	0	1	0	2	0	0	1	0	2
1	0	1	0	4	0	1	1	0	3
			Total	12				Total	11

Table A.4: Six sortings for parallel rung two evaluation

A					B				
$Y1c$	$X3$	$X4$	$Y2f$	Arcs	$Y1c$	$X3$	$X4$	$Y2f$	Arcs
1	1	0	1	5	1	1	0	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
0	1	0	0		0	1	0	0	
0	1	1	0	1	0	1	1	0	1
1	0	0	0		1	1	1	0	
1	0	1	0	3	1	0	1	0	3
1	1	1	0	4	1	0	0	0	4
			Total	13				Total	13
C					D*				
$Y1c$	$X3$	$X4$	$Y2f$	Arcs	$Y1c$	$X3$	$X4$	$Y2f$	Arcs
1	1	0	1	5	1	1	0	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
1	0	0	0		1	0	0	0	
1	0	1	0	1	1	0	1	0	1
0	1	0	0		1	1	1	0	
0	1	1	0	2	0	1	1	0	2
1	1	1	0	4	0	1	0	0	3
			Total	12				Total	11
E					F				
$Y1c$	$X3$	$X4$	$Y2f$	Arcs	$Y1c$	$X3$	$X4$	$Y2f$	Arcs
1	1	0	1	5	1	1	0	1	5
0	0	1	0		0	0	1	0	
0	1	1	0		0	1	1	0	
1	0	1	0		1	0	1	0	
1	1	1	0	1	1	1	1	0	1
0	0	0	0		0	0	0	0	
0	1	0	0	2	1	0	0	0	2
1	0	0	0	4	0	1	0	0	3
			Total	12				Total	11

Table A.5: Six combinations generated for parallel rung three

A					B				
$Y1c$	$X5$	$X6$	$Y3f$	Arcs	$Y1c$	$X5$	$X6$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
0	1	0	0		0	1	0	0	
0	1	1	0	1	0	1	1	0	1
1	1	0	0		1	0	1	0	
1	0	0	0	3	1	0	0	0	3
1	0	1	0	4	1	1	0	0	4
			Total	13				Total	13
C*					D				
$Y1c$	$X5$	$X6$	$Y3f$	Arcs	$Y1c$	$X5$	$X6$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
0	0	1	0		0	0	1	0	
1	0	0	0		1	0	0	0	
1	0	1	0	1	1	0	1	0	1
1	1	0	0		0	1	1	0	
0	1	0	0	2	0	1	0	0	2
0	1	1	0	3	1	1	0	0	4
			Total	11				Total	12
E					F				
$Y1c$	$X5$	$X6$	$Y3f$	Arcs	$Y1c$	$X5$	$X6$	$Y3f$	Arcs
1	1	1	1	5	1	1	1	1	5
0	0	0	0		0	0	0	0	
1	0	0	0		1	0	0	0	
0	1	0	0		0	1	0	0	
1	1	0	0	1	1	1	0	0	1
0	1	1	0		1	0	1	0	
0	0	1	0	2	0	0	1	0	2
1	0	1	0	4	0	1	1	0	3
			Total	12				Total	11

APPENDIX B. TEST EVALUATION DATA

Table B.1: Evaluation data for serial RLL

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
0 0 0 0 0	0 0 0	0 0 0			
0 0 0 0 0	0 0 1	0 0 0			
0 0 0 0 0	0 1 0	0 0 0			
0 0 0 0 0	0 1 1	0 0 0			
0 0 0 0 0	1 0 0	0 0 0			
0 0 0 0 0	1 0 1	0 0 0			
0 0 0 0 0	1 1 0	0 0 0			
0 0 0 0 0	1 1 1	0 0 0			
0 0 0 0 1	0 0 0	0 0 0			
0 0 0 0 1	0 0 1	0 0 0			
0 0 0 0 1	0 1 0	0 0 0			
0 0 0 0 1	0 1 1	0 0 0			
0 0 0 0 1	1 0 0	0 0 0			
0 0 0 0 1	1 0 1	0 0 0			
0 0 0 0 1	1 1 0	0 0 0			
0 0 0 0 1	1 1 1	0 0 0			
0 0 0 1 0	0 0 0	0 0 0			
0 0 0 1 0	0 0 1	0 0 0			
0 0 0 1 0	0 1 0	0 0 0			
0 0 0 1 0	0 1 1	0 0 0			
0 0 0 1 0	1 0 0	0 0 0			
0 0 0 1 0	1 0 1	0 0 0			
0 0 0 1 0	1 1 0	0 0 0			
0 0 0 1 0	1 1 1	0 0 0			
0 0 0 1 1	0 0 0	0 0 0	0 0 0		
0 0 0 1 1	0 0 1	0 0 0	0 0 0		
0 0 0 1 1	0 1 0	0 0 1	0 0 0		
0 0 0 1 1	0 1 1	0 0 1	0 0 0		
0 0 0 1 1	1 0 0	0 0 0	0 0 0		
0 0 0 1 1	1 0 1	0 0 0	0 0 0		
0 0 0 1 1	1 1 0	0 0 1	0 0 0		
0 0 0 1 1	1 1 1	0 0 1	0 0 0		

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
0 0 1 0 0	0 0 0	0 0 0	0 0 0		
0 0 1 0 0	0 0 1	0 0 0	0 0 0		
0 0 1 0 0	0 1 0	0 0 0	0 0 0		
0 0 1 0 0	0 1 1	0 0 0	0 0 0		
0 0 1 0 0	1 0 0	0 1 0	0 0 0		
0 0 1 0 0	1 0 1	0 0 0	0 0 0		
0 0 1 0 0	1 1 0	0 1 0	0 0 0		
0 0 1 0 0	1 1 1	0 0 0	0 0 0		
0 0 1 0 1	0 0 0	0 0 0	0 0 0		
0 0 1 0 1	0 0 1	0 0 0	0 0 0		
0 0 1 0 1	0 1 0	0 0 0	0 0 0		
0 0 1 0 1	0 1 1	0 0 0	0 0 0		
0 0 1 0 1	1 0 0	0 1 0	0 0 0		
0 0 1 0 1	1 0 1	0 0 0	0 0 0		
0 0 1 0 1	1 1 0	0 1 0	0 0 0		
0 0 1 0 1	1 1 1	0 0 0	0 0 0		
0 0 1 1 0	0 0 0	0 0 0	0 0 0		
0 0 1 1 0	0 0 1	0 0 0	0 0 0		
0 0 1 1 0	0 1 0	0 0 0	0 0 0		
0 0 1 1 0	0 1 1	0 0 0	0 0 0		
0 0 1 1 0	1 0 0	0 1 0	0 0 0		
0 0 1 1 0	1 0 1	0 0 0	0 0 0		
0 0 1 1 0	1 1 0	0 1 0	0 0 0		
0 0 1 1 0	1 1 1	0 0 0	0 0 0		
0 0 1 1 1	0 0 0	0 0 0	0 0 0	0 0 0	
0 0 1 1 1	0 0 1	0 0 0	0 0 0	0 0 0	
0 0 1 1 1	0 1 0	0 0 1	0 0 0	0 0 0	
0 0 1 1 1	0 1 1	0 0 1	0 0 0	0 0 0	
0 0 1 1 1	1 0 0	0 1 0	0 0 1	0 0 0	
0 0 1 1 1	1 0 1	0 0 0	0 0 0	0 0 0	
0 0 1 1 1	1 1 0	0 1 1	0 0 1	0 0 0	
0 0 1 1 1	1 1 1	0 0 1	0 0 0	0 0 0	

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
0 1 0 0 0	0 0 0	0 0 0			
0 1 0 0 0	0 0 1	0 0 0			
0 1 0 0 0	0 1 0	0 0 0			
0 1 0 0 0	0 1 1	0 0 0			
0 1 0 0 0	1 0 0	0 0 0			
0 1 0 0 0	1 0 1	0 0 0			
0 1 0 0 0	1 1 0	0 0 0			
0 1 0 0 0	1 1 1	0 0 0			
0 1 0 0 1	0 0 0	0 0 0			
0 1 0 0 1	0 0 1	0 0 0			
0 1 0 0 1	0 1 0	0 0 0			
0 1 0 0 1	0 1 1	0 0 0			
0 1 0 0 1	1 0 0	0 0 0			
0 1 0 0 1	1 0 1	0 0 0			
0 1 0 0 1	1 1 0	0 0 0			
0 1 0 0 1	1 1 1	0 0 0			
0 1 0 1 0	0 0 0	0 0 0			
0 1 0 1 0	0 0 1	0 0 0			
0 1 0 1 0	0 1 0	0 0 0			
0 1 0 1 0	0 1 1	0 0 0			
0 1 0 1 0	1 0 0	0 0 0			
0 1 0 1 0	1 0 1	0 0 0			
0 1 0 1 0	1 1 0	0 0 0			
0 1 0 1 0	1 1 1	0 0 0			
0 1 0 1 1	0 0 0	0 0 0	0 0 0		
0 1 0 1 1	0 0 1	0 0 0	0 0 0		
0 1 0 1 1	0 1 0	0 0 1	0 0 0		
0 1 0 1 1	0 1 1	0 0 1	0 0 0		
0 1 0 1 1	1 0 0	0 0 0	0 0 0		
0 1 0 1 1	1 0 1	0 0 0	0 0 0		
0 1 0 1 1	1 1 0	0 0 1	0 0 0		
0 1 0 1 1	1 1 1	0 0 1	0 0 0		

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
0 1 1 0 0	0 0 0	0 0 0	0 0 0		
0 1 1 0 0	0 0 1	0 0 0	0 0 0		
0 1 1 0 0	0 1 0	0 0 0	0 0 0		
0 1 1 0 0	0 1 1	0 0 0	0 0 0		
0 1 1 0 0	1 0 0	0 1 0	0 0 0		
0 1 1 0 0	1 0 1	0 0 0	0 0 0		
0 1 1 0 0	1 1 0	0 1 0	0 0 0		
0 1 1 0 0	1 1 1	0 0 0	0 0 0		
0 1 1 0 1	0 0 0	0 0 0	0 0 0		
0 1 1 0 1	0 0 1	0 0 0	0 0 0		
0 1 1 0 1	0 1 0	0 0 0	0 0 0		
0 1 1 0 1	0 1 1	0 0 0	0 0 0		
0 1 1 0 1	1 0 0	0 1 0	0 0 0		
0 1 1 0 1	1 0 1	0 0 0	0 0 0		
0 1 1 0 1	1 1 0	0 1 0	0 0 0		
0 1 1 0 1	1 1 1	0 0 0	0 0 0		
0 1 1 1 0	0 0 0	0 0 0	0 0 0		
0 1 1 1 0	0 0 1	0 0 0	0 0 0		
0 1 1 1 0	0 1 0	0 0 0	0 0 0		
0 1 1 1 0	0 1 1	0 0 0	0 0 0		
0 1 1 1 0	1 0 0	0 1 0	0 0 0		
0 1 1 1 0	1 0 1	0 0 0	0 0 0		
0 1 1 1 0	1 1 0	0 1 0	0 0 0		
0 1 1 1 0	1 1 1	0 0 0	0 0 0		
0 1 1 1 1	0 0 0	0 0 0	0 0 0	0 0 0	
0 1 1 1 1	0 0 1	0 0 0	0 0 0	0 0 0	
0 1 1 1 1	0 1 0	0 0 1	0 0 0	0 0 0	
0 1 1 1 1	0 1 1	0 0 1	0 0 0	0 0 0	
0 1 1 1 1	1 0 0	0 1 0	0 0 1	0 0 0	
0 1 1 1 1	1 0 1	0 0 0	0 0 0	0 0 0	
0 1 1 1 1	1 1 0	0 1 1	0 0 1	0 0 0	
0 1 1 1 1	1 1 1	0 0 1	0 0 0	0 0 0	

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1 0 0 0 0	0 0 0	1 0 0			
1 0 0 0 0	0 0 1	1 0 0			
1 0 0 0 0	0 1 0	1 0 0			
1 0 0 0 0	0 1 1	1 0 0			
1 0 0 0 0	1 0 0	1 0 0			
1 0 0 0 0	1 0 1	1 0 0			
1 0 0 0 0	1 1 0	1 0 0			
1 0 0 0 0	1 1 1	1 0 0			
1 0 0 0 1	0 0 0	1 0 0			
1 0 0 0 1	0 0 1	1 0 0			
1 0 0 0 1	0 1 0	1 0 0			
1 0 0 0 1	0 1 1	1 0 0			
1 0 0 0 1	1 0 0	1 0 0			
1 0 0 0 1	1 0 1	1 0 0			
1 0 0 0 1	1 1 0	1 0 0			
1 0 0 0 1	1 1 1	1 0 0			
1 0 0 1 0	0 0 0	1 0 0			
1 0 0 1 0	0 0 1	1 0 0			
1 0 0 1 0	0 1 0	1 0 0			
1 0 0 1 0	0 1 1	1 0 0			
1 0 0 1 0	1 0 0	1 0 0			
1 0 0 1 0	1 0 1	1 0 0			
1 0 0 1 0	1 1 0	1 0 0			
1 0 0 1 0	1 1 1	1 0 0			
1 0 0 1 1	0 0 0	1 0 0	1 0 0		
1 0 0 1 1	0 0 1	1 0 0	1 0 0		
1 0 0 1 1	0 1 0	1 0 1	1 0 0		
1 0 0 1 1	0 1 1	1 0 1	1 0 0		
1 0 0 1 1	1 0 0	1 0 0	1 0 0		
1 0 0 1 1	1 0 1	1 0 0	1 0 0		
1 0 0 1 1	1 1 0	1 0 1	1 0 0		
1 0 0 1 1	1 1 1	1 0 1	1 0 0		

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1 0 1 0 0	0 0 0	1 0 0	1 1 0		
1 0 1 0 0	0 0 1	1 0 0	1 1 0		
1 0 1 0 0	0 1 0	1 0 0	1 1 0		
1 0 1 0 0	0 1 1	1 0 0	1 1 0		
1 0 1 0 0	1 0 0	1 1 0	1 1 0		
1 0 1 0 0	1 0 1	1 0 0	1 1 0		
1 0 1 0 0	1 1 0	1 1 0	1 1 0		
1 0 1 0 0	1 1 1	1 0 0	1 1 0		
1 0 1 0 1	0 0 0	1 0 0	1 1 0		
1 0 1 0 1	0 0 1	1 0 0	1 1 0		
1 0 1 0 1	0 1 0	1 0 0	1 1 0		
1 0 1 0 1	0 1 1	1 0 0	1 1 0		
1 0 1 0 1	1 0 0	1 1 0	1 1 0		
1 0 1 0 1	1 0 1	1 0 0	1 1 0		
1 0 1 0 1	1 1 0	1 1 0	1 1 0		
1 0 1 0 1	1 1 1	1 0 0	1 1 0		
1 0 1 1 0	0 0 0	1 0 0	1 1 0		
1 0 1 1 0	0 0 1	1 0 0	1 1 0		
1 0 1 1 0	0 1 0	1 0 0	1 1 0		
1 0 1 1 0	0 1 1	1 0 0	1 1 0		
1 0 1 1 0	1 0 0	1 1 0	1 1 0		
1 0 1 1 0	1 0 1	1 0 0	1 1 0		
1 0 1 1 0	1 1 0	1 1 0	1 1 0		
1 0 1 1 0	1 1 1	1 0 0	1 1 0		
1 0 1 1 1	0 0 0	1 0 0	1 1 0	1 1 1	1 0 1
1 0 1 1 1	0 0 1	1 0 0	1 1 0	1 1 1	1 0 1
1 0 1 1 1	0 1 0	1 0 1	1 0 0	1 1 0	1 1 1
1 0 1 1 1	0 1 1	1 0 1	1 0 0	1 1 0	1 1 1
1 0 1 1 1	1 0 0	1 1 0	1 1 1	1 0 1	1 0 0
1 0 1 1 1	1 0 1	1 0 0	1 1 0	1 1 1	1 0 1
1 0 1 1 1	1 1 0	1 1 1	1 0 1	1 0 0	1 1 0
1 0 1 1 1	1 1 1	1 0 1	1 0 0	1 1 0	1 1 1

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1 1 0 0 0	0 0 0	0 0 0			
1 1 0 0 0	0 0 1	0 0 0			
1 1 0 0 0	0 1 0	0 0 0			
1 1 0 0 0	0 1 1	0 0 0			
1 1 0 0 0	1 0 0	0 0 0			
1 1 0 0 0	1 0 1	0 0 0			
1 1 0 0 0	1 1 0	0 0 0			
1 1 0 0 0	1 1 1	0 0 0			
1 1 0 0 1	0 0 0	0 0 0			
1 1 0 0 1	0 0 1	0 0 0			
1 1 0 0 1	0 1 0	0 0 0			
1 1 0 0 1	0 1 1	0 0 0			
1 1 0 0 1	1 0 0	0 0 0			
1 1 0 0 1	1 0 1	0 0 0			
1 1 0 0 1	1 1 0	0 0 0			
1 1 0 0 1	1 1 1	0 0 0			
1 1 0 1 0	0 0 0	0 0 0			
1 1 0 1 0	0 0 1	0 0 0			
1 1 0 1 0	0 1 0	0 0 0			
1 1 0 1 0	0 1 1	0 0 0			
1 1 0 1 0	1 0 0	0 0 0			
1 1 0 1 0	1 0 1	0 0 0			
1 1 0 1 0	1 1 0	0 0 0			
1 1 0 1 0	1 1 1	0 0 0			
1 1 0 1 1	0 0 0	0 0 0	0 0 0		
1 1 0 1 1	0 0 1	0 0 0	0 0 0		
1 1 0 1 1	0 1 0	0 0 1	0 0 0		
1 1 0 1 1	0 1 1	0 0 1	0 0 0		
1 1 0 1 1	1 0 0	0 0 0	0 0 0		
1 1 0 1 1	1 0 1	0 0 0	0 0 0		
1 1 0 1 1	1 1 0	0 0 1	0 0 0		
1 1 0 1 1	1 1 1	0 0 1	0 0 0		

Table B.1 Continued

INPUTS	OUTPUTS				
	INIT.	SCAN 1	SCAN 2	SCAN 3	SCAN 4
X X X X X	Y Y Y	Y Y Y	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1 1 1 0 0	0 0 0	0 0 0	0 0 0		
1 1 1 0 0	0 0 1	0 0 0	0 0 0		
1 1 1 0 0	0 1 0	0 0 0	0 0 0		
1 1 1 0 0	0 1 1	0 0 0	0 0 0		
1 1 1 0 0	1 0 0	0 1 0	0 0 0		
1 1 1 0 0	1 0 1	0 0 0	0 0 0		
1 1 1 0 0	1 1 0	0 1 0	0 0 0		
1 1 1 0 0	1 1 1	0 0 0	0 0 0		
1 1 1 0 1	0 0 0	0 0 0	0 0 0		
1 1 1 0 1	0 0 1	0 0 0	0 0 0		
1 1 1 0 1	0 1 0	0 0 0	0 0 0		
1 1 1 0 1	0 1 1	0 0 0	0 0 0		
1 1 1 0 1	1 0 0	0 1 0	0 0 0		
1 1 1 0 1	1 0 1	0 0 0	0 0 0		
1 1 1 0 1	1 1 0	0 1 0	0 0 0		
1 1 1 0 1	1 1 1	0 0 0	0 0 0		
1 1 1 1 0	0 0 0	0 0 0	0 0 0		
1 1 1 1 0	0 0 1	0 0 0	0 0 0		
1 1 1 1 0	0 1 0	0 0 0	0 0 0		
1 1 1 1 0	0 1 1	0 0 0	0 0 0		
1 1 1 1 0	1 0 0	0 1 0	0 0 0		
1 1 1 1 0	1 0 1	0 0 0	0 0 0		
1 1 1 1 0	1 1 0	0 1 0	0 0 0		
1 1 1 1 0	1 1 1	0 0 0	0 0 0		
1 1 1 1 1	0 0 0	0 0 0	0 0 0	0 0 0	
1 1 1 1 1	0 0 1	0 0 0	0 0 0	0 0 0	
1 1 1 1 1	0 1 0	0 0 1	0 0 0	0 0 0	
1 1 1 1 1	0 1 1	0 0 1	0 0 0	0 0 0	
1 1 1 1 1	1 0 0	0 1 0	0 0 1	0 0 0	
1 1 1 1 1	1 0 1	0 0 0	0 0 0	0 0 0	
1 1 1 1 1	1 1 0	0 1 1	0 0 1	0 0 0	
1 1 1 1 1	1 1 1	0 0 1	0 0 0	0 0 0	

Table B.2: Evaluation data for parallel RLL

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 0 0 0 0 0	0 0 0	0 0 0	
0 0 0 0 0 0	0 0 1	0 0 0	
0 0 0 0 0 0	0 1 0	0 0 0	
0 0 0 0 0 0	0 1 1	0 0 0	
0 0 0 0 0 0	1 0 0	0 0 0	
0 0 0 0 0 0	1 0 1	0 0 0	
0 0 0 0 0 0	1 1 0	0 0 0	
0 0 0 0 0 0	1 1 1	0 0 0	
0 0 0 0 0 1	0 0 0	0 0 0	
0 0 0 0 0 1	0 0 1	0 0 0	
0 0 0 0 0 1	0 1 0	0 0 0	
0 0 0 0 0 1	0 1 1	0 0 0	
0 0 0 0 0 1	1 0 0	0 0 0	
0 0 0 0 0 1	1 0 1	0 0 0	
0 0 0 0 0 1	1 1 0	0 0 0	
0 0 0 0 0 1	1 1 1	0 0 0	
0 0 0 0 1 0	0 0 0	0 0 0	
0 0 0 0 1 0	0 0 1	0 0 0	
0 0 0 0 1 0	0 1 0	0 0 0	
0 0 0 0 1 0	0 1 1	0 0 0	
0 0 0 0 1 0	1 0 0	0 0 0	
0 0 0 0 1 0	1 0 1	0 0 0	
0 0 0 0 1 0	1 1 0	0 0 0	
0 0 0 0 1 0	1 1 1	0 0 0	
0 0 0 0 1 1	0 0 0	0 0 0	0 0 0
0 0 0 0 1 1	0 0 1	0 0 0	0 0 0
0 0 0 0 1 1	0 1 0	0 0 0	0 0 0
0 0 0 0 1 1	0 1 1	0 0 0	0 0 0
0 0 0 0 1 1	1 0 0	0 0 1	0 0 0
0 0 0 0 1 1	1 0 1	0 0 1	0 0 0
0 0 0 0 1 1	1 1 0	0 0 1	0 0 0
0 0 0 0 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 0 0 1 0 0	0 0 0	0 0 0	
0 0 0 1 0 0	0 0 1	0 0 0	
0 0 0 1 0 0	0 1 0	0 0 0	
0 0 0 1 0 0	0 1 1	0 0 0	
0 0 0 1 0 0	1 0 0	0 0 0	
0 0 0 1 0 0	1 0 1	0 0 0	
0 0 0 1 0 0	1 1 0	0 0 0	
0 0 0 1 0 0	1 1 1	0 0 0	
0 0 0 1 0 1	0 0 0	0 0 0	
0 0 0 1 0 1	0 0 1	0 0 0	
0 0 0 1 0 1	0 1 0	0 0 0	
0 0 0 1 0 1	0 1 1	0 0 0	
0 0 0 1 0 1	1 0 0	0 0 0	
0 0 0 1 0 1	1 0 1	0 0 0	
0 0 0 1 0 1	1 1 0	0 0 0	
0 0 0 1 0 1	1 1 1	0 0 0	
0 0 0 1 1 0	0 0 0	0 0 0	
0 0 0 1 1 0	0 0 1	0 0 0	
0 0 0 1 1 0	0 1 0	0 0 0	
0 0 0 1 1 0	0 1 1	0 0 0	
0 0 0 1 1 0	1 0 0	0 0 0	
0 0 0 1 1 0	1 0 1	0 0 0	
0 0 0 1 1 0	1 1 0	0 0 0	
0 0 0 1 1 0	1 1 1	0 0 0	
0 0 0 1 1 1	0 0 0	0 0 0	0 0 0
0 0 0 1 1 1	0 0 1	0 0 0	0 0 0
0 0 0 1 1 1	0 1 0	0 0 0	0 0 0
0 0 0 1 1 1	0 1 1	0 0 0	0 0 0
0 0 0 1 1 1	1 0 0	0 0 1	0 0 0
0 0 0 1 1 1	1 0 1	0 0 1	0 0 0
0 0 0 1 1 1	1 1 0	0 0 1	0 0 0
0 0 0 1 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 0 1 0 0 0	0 0 0	0 0 0	0 0 0
0 0 1 0 0 0	0 0 1	0 0 0	0 0 0
0 0 1 0 0 0	0 1 0	0 0 0	0 0 0
0 0 1 0 0 0	0 1 1	0 0 0	0 0 0
0 0 1 0 0 0	1 0 0	0 1 0	0 0 0
0 0 1 0 0 0	1 0 1	0 1 0	0 0 0
0 0 1 0 0 0	1 1 0	0 1 0	0 0 0
0 0 1 0 0 0	1 1 1	0 1 0	0 0 0
0 0 1 0 0 1	0 0 0	0 0 0	0 0 0
0 0 1 0 0 1	0 0 1	0 0 0	0 0 0
0 0 1 0 0 1	0 1 0	0 0 0	0 0 0
0 0 1 0 0 1	0 1 1	0 0 0	0 0 0
0 0 1 0 0 1	1 0 0	0 1 0	0 0 0
0 0 1 0 0 1	1 0 1	0 1 0	0 0 0
0 0 1 0 0 1	1 1 0	0 1 0	0 0 0
0 0 1 0 0 1	1 1 1	0 1 0	0 0 0
0 0 1 0 1 0	0 0 0	0 0 0	0 0 0
0 0 1 0 1 0	0 0 1	0 0 0	0 0 0
0 0 1 0 1 0	0 1 0	0 0 0	0 0 0
0 0 1 0 1 0	0 1 1	0 0 0	0 0 0
0 0 1 0 1 0	1 0 0	0 1 0	0 0 0
0 0 1 0 1 0	1 0 1	0 1 0	0 0 0
0 0 1 0 1 0	1 1 0	0 1 0	0 0 0
0 0 1 0 1 0	1 1 1	0 1 0	0 0 0
0 0 1 0 1 1	0 0 0	0 0 0	0 0 0
0 0 1 0 1 1	0 0 1	0 0 0	0 0 0
0 0 1 0 1 1	0 1 0	0 0 0	0 0 0
0 0 1 0 1 1	0 1 1	0 0 0	0 0 0
0 0 1 0 1 1	1 0 0	0 1 1	0 0 0
0 0 1 0 1 1	1 0 1	0 1 1	0 0 0
0 0 1 0 1 1	1 1 0	0 1 1	0 0 0
0 0 1 0 1 1	1 1 1	0 1 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 0 1 1 0 0	0 0 0	0 0 0	
0 0 1 1 0 0	0 0 1	0 0 0	
0 0 1 1 0 0	0 1 0	0 0 0	
0 0 1 1 0 0	0 1 1	0 0 0	
0 0 1 1 0 0	1 0 0	0 0 0	
0 0 1 1 0 0	1 0 1	0 0 0	
0 0 1 1 0 0	1 1 0	0 0 0	
0 0 1 1 0 0	1 1 1	0 0 0	
0 0 1 1 0 1	0 0 0	0 0 0	
0 0 1 1 0 1	0 0 1	0 0 0	
0 0 1 1 0 1	0 1 0	0 0 0	
0 0 1 1 0 1	0 1 1	0 0 0	
0 0 1 1 0 1	1 0 0	0 0 0	
0 0 1 1 0 1	1 0 1	0 0 0	
0 0 1 1 0 1	1 1 0	0 0 0	
0 0 1 1 0 1	1 1 1	0 0 0	
0 0 1 1 1 0	0 0 0	0 0 0	
0 0 1 1 1 0	0 0 1	0 0 0	
0 0 1 1 1 0	0 1 0	0 0 0	
0 0 1 1 1 0	0 1 1	0 0 0	
0 0 1 1 1 0	1 0 0	0 0 0	
0 0 1 1 1 0	1 0 1	0 0 0	
0 0 1 1 1 0	1 1 0	0 0 0	
0 0 1 1 1 0	1 1 1	0 0 0	
0 0 1 1 1 1	0 0 0	0 0 0	0 0 0
0 0 1 1 1 1	0 0 1	0 0 0	0 0 0
0 0 1 1 1 1	0 1 0	0 0 0	0 0 0
0 0 1 1 1 1	0 1 1	0 0 0	0 0 0
0 0 1 1 1 1	1 0 0	0 0 1	0 0 0
0 0 1 1 1 1	1 0 1	0 0 1	0 0 0
0 0 1 1 1 1	1 1 0	0 0 1	0 0 0
0 0 1 1 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 1 0 0 0 0	0 0 0	0 0 0	
0 1 0 0 0 0	0 0 1	0 0 0	
0 1 0 0 0 0	0 1 0	0 0 0	
0 1 0 0 0 0	0 1 1	0 0 0	
0 1 0 0 0 0	1 0 0	0 0 0	
0 1 0 0 0 0	1 0 1	0 0 0	
0 1 0 0 0 0	1 1 0	0 0 0	
0 1 0 0 0 0	1 1 1	0 0 0	
0 1 0 0 0 1	0 0 0	0 0 0	
0 1 0 0 0 1	0 0 1	0 0 0	
0 1 0 0 0 1	0 1 0	0 0 0	
0 1 0 0 0 1	0 1 1	0 0 0	
0 1 0 0 0 1	1 0 0	0 0 0	
0 1 0 0 0 1	1 0 1	0 0 0	
0 1 0 0 0 1	1 1 0	0 0 0	
0 1 0 0 0 1	1 1 1	0 0 0	
0 1 0 0 1 0	0 0 0	0 0 0	
0 1 0 0 1 0	0 0 1	0 0 0	
0 1 0 0 1 0	0 1 0	0 0 0	
0 1 0 0 1 0	0 1 1	0 0 0	
0 1 0 0 1 0	1 0 0	0 0 0	
0 1 0 0 1 0	1 0 1	0 0 0	
0 1 0 0 1 0	1 1 0	0 0 0	
0 1 0 0 1 0	1 1 1	0 0 0	
0 1 0 0 1 1	0 0 0	0 0 0	0 0 0
0 1 0 0 1 1	0 0 1	0 0 0	0 0 0
0 1 0 0 1 1	0 1 0	0 0 0	0 0 0
0 1 0 0 1 1	0 1 1	0 0 0	0 0 0
0 1 0 0 1 1	1 0 0	0 0 1	0 0 0
0 1 0 0 1 1	1 0 1	0 0 1	0 0 0
0 1 0 0 1 1	1 1 0	0 0 1	0 0 0
0 1 0 0 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 1 0 1 0 0	0 0 0	0 0 0	
0 1 0 1 0 0	0 0 1	0 0 0	
0 1 0 1 0 0	0 1 0	0 0 0	
0 1 0 1 0 0	0 1 1	0 0 0	
0 1 0 1 0 0	1 0 0	0 0 0	
0 1 0 1 0 0	1 0 1	0 0 0	
0 1 0 1 0 0	1 1 0	0 0 0	
0 1 0 1 0 0	1 1 1	0 0 0	
0 1 0 1 0 1	0 0 0	0 0 0	
0 1 0 1 0 1	0 0 1	0 0 0	
0 1 0 1 0 1	0 1 0	0 0 0	
0 1 0 1 0 1	0 1 1	0 0 0	
0 1 0 1 0 1	1 0 0	0 0 0	
0 1 0 1 0 1	1 0 1	0 0 0	
0 1 0 1 0 1	1 1 0	0 0 0	
0 1 0 1 0 1	1 1 1	0 0 0	
0 1 0 1 1 0	0 0 0	0 0 0	
0 1 0 1 1 0	0 0 1	0 0 0	
0 1 0 1 1 0	0 1 0	0 0 0	
0 1 0 1 1 0	0 1 1	0 0 0	
0 1 0 1 1 0	1 0 0	0 0 0	
0 1 0 1 1 0	1 0 1	0 0 0	
0 1 0 1 1 0	1 1 0	0 0 0	
0 1 0 1 1 0	1 1 1	0 0 0	
0 1 0 1 1 1	0 0 0	0 0 0	0 0 0
0 1 0 1 1 1	0 0 1	0 0 0	0 0 0
0 1 0 1 1 1	0 1 0	0 0 0	0 0 0
0 1 0 1 1 1	0 1 1	0 0 0	0 0 0
0 1 0 1 1 1	1 0 0	0 0 1	0 0 0
0 1 0 1 1 1	1 0 1	0 0 1	0 0 0
0 1 0 1 1 1	1 1 0	0 0 1	0 0 0
0 1 0 1 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 1 1 0 0 0	0 0 0	0 0 0	0 0 0
0 1 1 0 0 0	0 0 1	0 0 0	0 0 0
0 1 1 0 0 0	0 1 0	0 0 0	0 0 0
0 1 1 0 0 0	0 1 1	0 0 0	0 0 0
0 1 1 0 0 0	1 0 0	0 1 0	0 0 0
0 1 1 0 0 0	1 0 1	0 1 0	0 0 0
0 1 1 0 0 0	1 1 0	0 1 0	0 0 0
0 1 1 0 0 0	1 1 1	0 1 0	0 0 0
0 1 1 0 0 1	0 0 0	0 0 0	0 0 0
0 1 1 0 0 1	0 0 1	0 0 0	0 0 0
0 1 1 0 0 1	0 1 0	0 0 0	0 0 0
0 1 1 0 0 1	0 1 1	0 0 0	0 0 0
0 1 1 0 0 1	1 0 0	0 1 0	0 0 0
0 1 1 0 0 1	1 0 1	0 1 0	0 0 0
0 1 1 0 0 1	1 1 0	0 1 0	0 0 0
0 1 1 0 0 1	1 1 1	0 1 0	0 0 0
0 1 1 0 1 0	0 0 0	0 0 0	0 0 0
0 1 1 0 1 0	0 0 1	0 0 0	0 0 0
0 1 1 0 1 0	0 1 0	0 0 0	0 0 0
0 1 1 0 1 0	0 1 1	0 0 0	0 0 0
0 1 1 0 1 0	1 0 0	0 1 0	0 0 0
0 1 1 0 1 0	1 0 1	0 1 0	0 0 0
0 1 1 0 1 0	1 1 0	0 1 0	0 0 0
0 1 1 0 1 0	1 1 1	0 1 0	0 0 0
0 1 1 0 1 1	0 0 0	0 0 0	0 0 0
0 1 1 0 1 1	0 0 1	0 0 0	0 0 0
0 1 1 0 1 1	0 1 0	0 0 0	0 0 0
0 1 1 0 1 1	0 1 1	0 0 0	0 0 0
0 1 1 0 1 1	1 0 0	0 1 1	0 0 0
0 1 1 0 1 1	1 0 1	0 1 1	0 0 0
0 1 1 0 1 1	1 1 0	0 1 1	0 0 0
0 1 1 0 1 1	1 1 1	0 1 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
0 1 1 1 0 0	0 0 0	0 0 0	
0 1 1 1 0 0	0 0 1	0 0 0	
0 1 1 1 0 0	0 1 0	0 0 0	
0 1 1 1 0 0	0 1 1	0 0 0	
0 1 1 1 0 0	1 0 0	0 0 0	
0 1 1 1 0 0	1 0 1	0 0 0	
0 1 1 1 0 0	1 1 0	0 0 0	
0 1 1 1 0 0	1 1 1	0 0 0	
0 1 1 1 0 1	0 0 0	0 0 0	
0 1 1 1 0 1	0 0 1	0 0 0	
0 1 1 1 0 1	0 1 0	0 0 0	
0 1 1 1 0 1	0 1 1	0 0 0	
0 1 1 1 0 1	1 0 0	0 0 0	
0 1 1 1 0 1	1 0 1	0 0 0	
0 1 1 1 0 1	1 1 0	0 0 0	
0 1 1 1 0 1	1 1 1	0 0 0	
0 1 1 1 1 0	0 0 0	0 0 0	
0 1 1 1 1 0	0 0 1	0 0 0	
0 1 1 1 1 0	0 1 0	0 0 0	
0 1 1 1 1 0	0 1 1	0 0 0	
0 1 1 1 1 0	1 0 0	0 0 0	
0 1 1 1 1 0	1 0 1	0 0 0	
0 1 1 1 1 0	1 1 0	0 0 0	
0 1 1 1 1 0	1 1 1	0 0 0	
0 1 1 1 1 1	0 0 0	0 0 0	0 0 0
0 1 1 1 1 1	0 0 1	0 0 0	0 0 0
0 1 1 1 1 1	0 1 0	0 0 0	0 0 0
0 1 1 1 1 1	0 1 1	0 0 0	0 0 0
0 1 1 1 1 1	1 0 0	0 0 1	0 0 0
0 1 1 1 1 1	1 0 1	0 0 1	0 0 0
0 1 1 1 1 1	1 1 0	0 0 1	0 0 0
0 1 1 1 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 0 0 0 0 0	0 0 0	1 0 0	
1 0 0 0 0 0	0 0 1	1 0 0	
1 0 0 0 0 0	0 1 0	1 0 0	
1 0 0 0 0 0	0 1 1	1 0 0	
1 0 0 0 0 0	1 0 0	1 0 0	
1 0 0 0 0 0	1 0 1	1 0 0	
1 0 0 0 0 0	1 1 0	1 0 0	
1 0 0 0 0 0	1 1 1	1 0 0	
1 0 0 0 0 1	0 0 0	1 0 0	
1 0 0 0 0 1	0 0 1	1 0 0	
1 0 0 0 0 1	0 1 0	1 0 0	
1 0 0 0 0 1	0 1 1	1 0 0	
1 0 0 0 0 1	1 0 0	1 0 0	
1 0 0 0 0 1	1 0 1	1 0 0	
1 0 0 0 0 1	1 1 0	1 0 0	
1 0 0 0 0 1	1 1 1	1 0 0	
1 0 0 0 1 0	0 0 0	1 0 0	
1 0 0 0 1 0	0 0 1	1 0 0	
1 0 0 0 1 0	0 1 0	1 0 0	
1 0 0 0 1 0	0 1 1	1 0 0	
1 0 0 0 1 0	1 0 0	1 0 0	
1 0 0 0 1 0	1 0 1	1 0 0	
1 0 0 0 1 0	1 1 0	1 0 0	
1 0 0 0 1 0	1 1 1	1 0 0	
1 0 0 0 1 1	0 0 0	1 0 0	1 0 1
1 0 0 0 1 1	0 0 1	1 0 0	1 0 1
1 0 0 0 1 1	0 1 0	1 0 0	1 0 1
1 0 0 0 1 1	0 1 1	1 0 0	1 0 1
1 0 0 0 1 1	1 0 0	1 0 1	1 0 1
1 0 0 0 1 1	1 0 1	1 0 1	1 0 1
1 0 0 0 1 1	1 1 0	1 0 1	1 0 1
1 0 0 0 1 1	1 1 1	1 0 1	1 0 1

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 0 0 1 0 0	0 0 0	1 0 0	
1 0 0 1 0 0	0 0 1	1 0 0	
1 0 0 1 0 0	0 1 0	1 0 0	
1 0 0 1 0 0	0 1 1	1 0 0	
1 0 0 1 0 0	1 0 0	1 0 0	
1 0 0 1 0 0	1 0 1	1 0 0	
1 0 0 1 0 0	1 1 0	1 0 0	
1 0 0 1 0 0	1 1 1	1 0 0	
1 0 0 1 0 1	0 0 0	1 0 0	
1 0 0 1 0 1	0 0 1	1 0 0	
1 0 0 1 0 1	0 1 0	1 0 0	
1 0 0 1 0 1	0 1 1	1 0 0	
1 0 0 1 0 1	1 0 0	1 0 0	
1 0 0 1 0 1	1 0 1	1 0 0	
1 0 0 1 0 1	1 1 0	1 0 0	
1 0 0 1 0 1	1 1 1	1 0 0	
1 0 0 1 1 0	0 0 0	1 0 0	
1 0 0 1 1 0	0 0 1	1 0 0	
1 0 0 1 1 0	0 1 0	1 0 0	
1 0 0 1 1 0	0 1 1	1 0 0	
1 0 0 1 1 0	1 0 0	1 0 0	
1 0 0 1 1 0	1 0 1	1 0 0	
1 0 0 1 1 0	1 1 0	1 0 0	
1 0 0 1 1 0	1 1 1	1 0 0	
1 0 0 1 1 1	0 0 0	1 0 0	1 0 1
1 0 0 1 1 1	0 0 1	1 0 0	1 0 1
1 0 0 1 1 1	0 1 0	1 0 0	1 0 1
1 0 0 1 1 1	0 1 1	1 0 0	1 0 1
1 0 0 1 1 1	1 0 0	1 0 1	1 0 1
1 0 0 1 1 1	1 0 1	1 0 1	1 0 1
1 0 0 1 1 1	1 1 0	1 0 1	1 0 1
1 0 0 1 1 1	1 1 1	1 0 1	1 0 1

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 0 1 0 0 0	0 0 0	1 0 0	1 1 0
1 0 1 0 0 0	0 0 1	1 0 0	1 1 0
1 0 1 0 0 0	0 1 0	1 0 0	1 1 0
1 0 1 0 0 0	0 1 1	1 0 0	1 1 0
1 0 1 0 0 0	1 0 0	1 1 0	1 1 0
1 0 1 0 0 0	1 0 1	1 1 0	1 1 0
1 0 1 0 0 0	1 1 0	1 1 0	1 1 0
1 0 1 0 0 0	1 1 1	1 1 0	1 1 0
1 0 1 0 0 1	0 0 0	1 0 0	1 1 0
1 0 1 0 0 1	0 0 1	1 0 0	1 1 0
1 0 1 0 0 1	0 1 0	1 0 0	1 1 0
1 0 1 0 0 1	0 1 1	1 0 0	1 1 0
1 0 1 0 0 1	1 0 0	1 1 0	1 1 0
1 0 1 0 0 1	1 0 1	1 1 0	1 1 0
1 0 1 0 0 1	1 1 0	1 1 0	1 1 0
1 0 1 0 0 1	1 1 1	1 1 0	1 1 0
1 0 1 0 1 0	0 0 0	1 0 0	1 1 0
1 0 1 0 1 0	0 0 1	1 0 0	1 1 0
1 0 1 0 1 0	0 1 0	1 0 0	1 1 0
1 0 1 0 1 0	0 1 1	1 0 0	1 1 0
1 0 1 0 1 0	1 0 0	1 1 0	1 1 0
1 0 1 0 1 0	1 0 1	1 1 0	1 1 0
1 0 1 0 1 0	1 1 0	1 1 0	1 1 0
1 0 1 0 1 0	1 1 1	1 1 0	1 1 0
1 0 1 0 1 1	0 0 0	1 0 0	1 1 1
1 0 1 0 1 1	0 0 1	1 0 0	1 1 1
1 0 1 0 1 1	0 1 0	1 0 0	1 1 1
1 0 1 0 1 1	0 1 1	1 0 0	1 1 1
1 0 1 0 1 1	1 0 0	1 1 1	1 1 1
1 0 1 0 1 1	1 0 1	1 1 1	1 1 1
1 0 1 0 1 1	1 1 0	1 1 1	1 1 1
1 0 1 0 1 1	1 1 1	1 1 1	1 1 1

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 0 1 1 0 0	0 0 0	1 0 0	
1 0 1 1 0 0	0 0 1	1 0 0	
1 0 1 1 0 0	0 1 0	1 0 0	
1 0 1 1 0 0	0 1 1	1 0 0	
1 0 1 1 0 0	1 0 0	1 0 0	
1 0 1 1 0 0	1 0 1	1 0 0	
1 0 1 1 0 0	1 1 0	1 0 0	
1 0 1 1 0 0	1 1 1	1 0 0	
1 0 1 1 0 1	0 0 0	1 0 0	
1 0 1 1 0 1	0 0 1	1 0 0	
1 0 1 1 0 1	0 1 0	1 0 0	
1 0 1 1 0 1	0 1 1	1 0 0	
1 0 1 1 0 1	1 0 0	1 0 0	
1 0 1 1 0 1	1 0 1	1 0 0	
1 0 1 1 0 1	1 1 0	1 0 0	
1 0 1 1 0 1	1 1 1	1 0 0	
1 0 1 1 1 0	0 0 0	1 0 0	
1 0 1 1 1 0	0 0 1	1 0 0	
1 0 1 1 1 0	0 1 0	1 0 0	
1 0 1 1 1 0	0 1 1	1 0 0	
1 0 1 1 1 0	1 0 0	1 0 0	
1 0 1 1 1 0	1 0 1	1 0 0	
1 0 1 1 1 0	1 1 0	1 0 0	
1 0 1 1 1 0	1 1 1	1 0 0	
1 0 1 1 1 1	0 0 0	1 0 0	1 0 1
1 0 1 1 1 1	0 0 1	1 0 0	1 0 1
1 0 1 1 1 1	0 1 0	1 0 0	1 0 1
1 0 1 1 1 1	0 1 1	1 0 0	1 0 1
1 0 1 1 1 1	1 0 0	1 0 1	1 0 1
1 0 1 1 1 1	1 0 1	1 0 1	1 0 1
1 0 1 1 1 1	1 1 0	1 0 1	1 0 1
1 0 1 1 1 1	1 1 1	1 0 1	1 0 1

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 1 0 0 0 0	0 0 0	0 0 0	
1 1 0 0 0 0	0 0 1	0 0 0	
1 1 0 0 0 0	0 1 0	0 0 0	
1 1 0 0 0 0	0 1 1	0 0 0	
1 1 0 0 0 0	1 0 0	0 0 0	
1 1 0 0 0 0	1 0 1	0 0 0	
1 1 0 0 0 0	1 1 0	0 0 0	
1 1 0 0 0 0	1 1 1	0 0 0	
1 1 0 0 0 1	0 0 0	0 0 0	
1 1 0 0 0 1	0 0 1	0 0 0	
1 1 0 0 0 1	0 1 0	0 0 0	
1 1 0 0 0 1	0 1 1	0 0 0	
1 1 0 0 0 1	1 0 0	0 0 0	
1 1 0 0 0 1	1 0 1	0 0 0	
1 1 0 0 0 1	1 1 0	0 0 0	
1 1 0 0 0 1	1 1 1	0 0 0	
1 1 0 0 1 0	0 0 0	0 0 0	
1 1 0 0 1 0	0 0 1	0 0 0	
1 1 0 0 1 0	0 1 0	0 0 0	
1 1 0 0 1 0	0 1 1	0 0 0	
1 1 0 0 1 0	1 0 0	0 0 0	
1 1 0 0 1 0	1 0 1	0 0 0	
1 1 0 0 1 0	1 1 0	0 0 0	
1 1 0 0 1 0	1 1 1	0 0 0	
1 1 0 0 1 1	0 0 0	0 0 0	0 0 0
1 1 0 0 1 1	0 0 1	0 0 0	0 0 0
1 1 0 0 1 1	0 1 0	0 0 0	0 0 0
1 1 0 0 1 1	0 1 1	0 0 0	0 0 0
1 1 0 0 1 1	1 0 0	0 0 1	0 0 0
1 1 0 0 1 1	1 0 1	0 0 1	0 0 0
1 1 0 0 1 1	1 1 0	0 0 1	0 0 0
1 1 0 0 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 1 0 1 0 0	0 0 0	0 0 0	
1 1 0 1 0 0	0 0 1	0 0 0	
1 1 0 1 0 0	0 1 0	0 0 0	
1 1 0 1 0 0	0 1 1	0 0 0	
1 1 0 1 0 0	1 0 0	0 0 0	
1 1 0 1 0 0	1 0 1	0 0 0	
1 1 0 1 0 0	1 1 0	0 0 0	
1 1 0 1 0 0	1 1 1	0 0 0	
1 1 0 1 0 1	0 0 0	0 0 0	
1 1 0 1 0 1	0 0 1	0 0 0	
1 1 0 1 0 1	0 1 0	0 0 0	
1 1 0 1 0 1	0 1 1	0 0 0	
1 1 0 1 0 1	1 0 0	0 0 0	
1 1 0 1 0 1	1 0 1	0 0 0	
1 1 0 1 0 1	1 1 0	0 0 0	
1 1 0 1 0 1	1 1 1	0 0 0	
1 1 0 1 1 0	0 0 0	0 0 0	
1 1 0 1 1 0	0 0 1	0 0 0	
1 1 0 1 1 0	0 1 0	0 0 0	
1 1 0 1 1 0	0 1 1	0 0 0	
1 1 0 1 1 0	1 0 0	0 0 0	
1 1 0 1 1 0	1 0 1	0 0 0	
1 1 0 1 1 0	1 1 0	0 0 0	
1 1 0 1 1 0	1 1 1	0 0 0	
1 1 0 1 1 1	0 0 0	0 0 0	0 0 0
1 1 0 1 1 1	0 0 1	0 0 0	0 0 0
1 1 0 1 1 1	0 1 0	0 0 0	0 0 0
1 1 0 1 1 1	0 1 1	0 0 0	0 0 0
1 1 0 1 1 1	1 0 0	0 0 1	0 0 0
1 1 0 1 1 1	1 0 1	0 0 1	0 0 0
1 1 0 1 1 1	1 1 0	0 0 1	0 0 0
1 1 0 1 1 1	1 1 1	0 0 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 1 1 0 0 0	0 0 0	0 0 0	0 0 0
1 1 1 0 0 0	0 0 1	0 0 0	0 0 0
1 1 1 0 0 0	0 1 0	0 0 0	0 0 0
1 1 1 0 0 0	0 1 1	0 0 0	0 0 0
1 1 1 0 0 0	1 0 0	0 1 0	0 0 0
1 1 1 0 0 0	1 0 1	0 1 0	0 0 0
1 1 1 0 0 0	1 1 0	0 1 0	0 0 0
1 1 1 0 0 0	1 1 1	0 1 0	0 0 0
1 1 1 0 0 1	0 0 0	0 0 0	0 0 0
1 1 1 0 0 1	0 0 1	0 0 0	0 0 0
1 1 1 0 0 1	0 1 0	0 0 0	0 0 0
1 1 1 0 0 1	0 1 1	0 0 0	0 0 0
1 1 1 0 0 1	1 0 0	0 1 0	0 0 0
1 1 1 0 0 1	1 0 1	0 1 0	0 0 0
1 1 1 0 0 1	1 1 0	0 1 0	0 0 0
1 1 1 0 0 1	1 1 1	0 1 0	0 0 0
1 1 1 0 1 0	0 0 0	0 0 0	0 0 0
1 1 1 0 1 0	0 0 1	0 0 0	0 0 0
1 1 1 0 1 0	0 1 0	0 0 0	0 0 0
1 1 1 0 1 0	0 1 1	0 0 0	0 0 0
1 1 1 0 1 0	1 0 0	0 1 0	0 0 0
1 1 1 0 1 0	1 0 1	0 1 0	0 0 0
1 1 1 0 1 0	1 1 0	0 1 0	0 0 0
1 1 1 0 1 0	1 1 1	0 1 0	0 0 0
1 1 1 0 1 1	0 0 0	0 0 0	0 0 0
1 1 1 0 1 1	0 0 1	0 0 0	0 0 0
1 1 1 0 1 1	0 1 0	0 0 0	0 0 0
1 1 1 0 1 1	0 1 1	0 0 0	0 0 0
1 1 1 0 1 1	1 0 0	0 1 1	0 0 0
1 1 1 0 1 1	1 0 1	0 1 1	0 0 0
1 1 1 0 1 1	1 1 0	0 1 1	0 0 0
1 1 1 0 1 1	1 1 1	0 1 1	0 0 0

Table B.2 Continued

INPUTS	OUTPUTS		
	INIT.	SCAN 1	SCAN 2
X X X X X X	Y Y Y	Y Y Y	Y Y Y
1 2 3 4 5 6	1 2 3	1 2 3	1 2 3
1 1 1 1 0 0	0 0 0	0 0 0	
1 1 1 1 0 0	0 0 1	0 0 0	
1 1 1 1 0 0	0 1 0	0 0 0	
1 1 1 1 0 0	0 1 1	0 0 0	
1 1 1 1 0 0	1 0 0	0 0 0	
1 1 1 1 0 0	1 0 1	0 0 0	
1 1 1 1 0 0	1 1 0	0 0 0	
1 1 1 1 0 0	1 1 1	0 0 0	
1 1 1 1 0 1	0 0 0	0 0 0	
1 1 1 1 0 1	0 0 1	0 0 0	
1 1 1 1 0 1	0 1 0	0 0 0	
1 1 1 1 0 1	0 1 1	0 0 0	
1 1 1 1 0 1	1 0 0	0 0 0	
1 1 1 1 0 1	1 0 1	0 0 0	
1 1 1 1 0 1	1 1 0	0 0 0	
1 1 1 1 0 1	1 1 1	0 0 0	
1 1 1 1 1 0	0 0 0	0 0 0	
1 1 1 1 1 0	0 0 1	0 0 0	
1 1 1 1 1 0	0 1 0	0 0 0	
1 1 1 1 1 0	0 1 1	0 0 0	
1 1 1 1 1 0	1 0 0	0 0 0	
1 1 1 1 1 0	1 0 1	0 0 0	
1 1 1 1 1 0	1 1 0	0 0 0	
1 1 1 1 1 0	1 1 1	0 0 0	
1 1 1 1 1 1	0 0 0	0 0 0	0 0 0
1 1 1 1 1 1	0 0 1	0 0 0	0 0 0
1 1 1 1 1 1	0 1 0	0 0 0	0 0 0
1 1 1 1 1 1	0 1 1	0 0 0	0 0 0
1 1 1 1 1 1	1 0 0	0 0 1	0 0 0
1 1 1 1 1 1	1 0 1	0 0 1	0 0 0
1 1 1 1 1 1	1 1 0	0 0 1	0 0 0
1 1 1 1 1 1	1 1 1	0 0 1	0 0 0

Table B.3: Evaluation data for serial REN

Transition	Initial	Internal		Reported
Fired	Inputs	Current	Future	Output
T1	10111	000	000	
T2.1	00111	000	100	
T3.2	00011	000	100	
T4.2, T5	00000	000	100	
T6.4	00000	000	100	
T7.4	00000	000	100	
T8.2	00000	100	000	
T9	00000	100		100
Scan 2				
T1	10111	100	000	
T2.2	10111	100	100	
T3.1	10011	100	110	
T4.2, T5	10000	100	110	
T6.4	10000	100	110	
T7.2	10000	110	100	
T8.1	10000	110	000	
T9	10000	110		110
Scan 3				
T1	10111	110	000	
T2.2	10111	110	100	
T3.1	10011	110	110	
T4.1, T5	10000	110	111	
T6.2	10000	111	110	
T7.1	10000	111	100	
T8.1	10000	111	000	
T9	10000	111		111
Scan 4				
T1	10111	111	000	
T2.2	10111	111	100	
T3.3	10011	111	110	
T4.1, T5	10000	111	111	
T6.1	10000	111	110	
T7.3	10000	101	100	
T8.1	10000	101	000	
T9	10000	101		101

Table B.4: Evaluation data for parallel REN

Transition Fired	Initial Inputs	Internal Current	Internal Future	Reported Output
T1	101011	000	000	
T2.1	001011	000	100	
T3.2	000011	000	100	
T4.4, T5	000000	000	100	
T6.4	000000	000	100	
T7.4	000000	000	100	
T8.2	000000	100	000	
T9	000000	100		100
Scan 2				
T1	101011	100	000	
T2.2	101011	100	100	
T3.1	100011	100	110	
T4.1, T5	100000	100	111	
T6.2	100000	101	110	
T7.2	100000	111	100	
T8.1	100000	111	000	
T9	100000	111		111
Scan 3				
T1	101011	111	000	
T2.2	101011	111	100	
T3.1	100011	111	110	
T4.1, T5	100000	111	111	
T6.1	100000	111	110	
T7.1	100000	111	100	
T8.1	100000	111	000	
T9	100000	111		111