Petri-Net Controller Synthesis using Matlab/Statflow Tools

Ahmed A. Ibraheem* M. Said Abdel Moteleb* Ahmed G. Bahgat**

**Electric Power and Machines Dept., Faculty of Eng., Cairo University, Giza, Egypt,
*Electronics Research Institute, National Research Center, Dokki, Cairo, Egypt

Abstract- **In many automated manufacturing systems, several devices work together with the characteristics of synchronization, concurrency, resource sharing, and cyclic sequences. The logic controller is a discrete event supervisory system which controls parallel and synchronized sequences of elementary operations of each device to achieve the goal of the automated manufacturing system. Even though logic controller is very important in manufacturing system, there is not yet a standard integrated tool, which is sufficiently powerful, versatile and simple to use, can be implemented on the real time platform, and with which is possible to carry out formal analysis of correctness besides the traditional approach of validation through simulation. In this paper a logic controller is presented using the Petri net, which is a graphical and mathematical modeling tool for describing and studying discrete event systems. Programs in Matlab and Statflow Toolbox are one of the most popular software in the area of applied mathematics, is used for representation, analysis and simulations of the designed logic controllers.**

Keywords: Discrete Event, Petri net, Logic controllers

I. INTRODUCTION

Design methods for sequential logic controllers play very important roles in advanced automated systems. The increasing complexity and varying needs of modern discrete manufacturing systems have challenged the traditional design methods and software tools such as the used of the relay ladder logic or the relay ladder logic diagrams (LLD's) for programmable logic controllers PLC. When the controlled system becomes large it is so complex to locating the cause of the problem or the fault when it's detected. Also when the control system specifications changed to meet the dynamically changed requirement in the market, the control sequence need to be changed also to meet this requirements, which is a very important in the context of agile manufacturing systems, which required a flexible, reusable and maintainable controller. The usage of traditional methods for design the control logic algorithm (software) is limited only to control the system but not to analyze and evaluate the qualitative and performance characteristics of the controlled system. So it's very important to fine a stander automated control tools that can be used in all controller design stages starting from design, modeling, simulation, and implementations.

Most of the software computer tools that are listed on the publications perform analysis and simulation of various Petri net classes [1]. Very often offer convenient graphical environment and sometimes they tend to be to complex. On the other hand these tools have very limited possibility of extensions to problems specifically needed for given application. Also some of this tools used a graphical editor for representation and modeling the Petri net model and then exported this model in the text form to anther tool for carrying out the analysis and simulation and then generate a executable code for the propose of implementations. In this paper only one software computer tool is used starting from modeling, analysis, simulation and toward to real time implementations.

II. PETRI NET THEORY

A. Introduction

Petri Net are a graphical and mathematical modeling tools for describing and studying discrete event systems, such as automated manufacturing systems, information processing systems. A Petri Net is represented by a bipartite oriented graph, where one type of nodes represents a system condition (called places) which is represented by a small circuit and the other type of nodes represent a system events (called transitions) which is represented by small bar.
Places are used to represent conditions (true/false), resource availability, or process status, e.g., a machine processing a part. Transitions are used for events, the start and stop of activities. They are interconnected by oriented arcs. The dynamics of the Petri Net is introduced by allowing a place to hold either none or a positive number of tokens (marking) depicted by small solid dots, as seen in the Figure (1). The number of tokens in a place marks the current system state. The occurrence of an event correspondent to the execution of firing of transitions; it changes the marking of places according to weights of graph marking of places, and the tokens are added or removed from places according to the weight and the direction of the related arcs.

A *marked* Petri net is defined by the quintuple $PN = (P, T, I, O, M)$ where $M : P \rightarrow N$ is a mapping from the set of places P to the nonnegative integers N. A marked PN contains *tokens* to represent the marking in a place. The initial marking is denoted by M_o.

B. Petri Net properties. Analysis, and Definitions

For Petri net models for logic controllers, three important properties are typically considered: liveness, safness, and

reversibility. The physical meaning of these properties for a logic controller can be summarized as follows [2]:

Liveness as considered in this paper means the absence of deadlocks. This property guarantees that all transitions are friable, and all the states of a logic controller which are modeled by places can occur. *Safeness* guarantees that there is no attempt to request execution of an ongoing process. *Reversibility* implies that the system will have a cyclic behavior and will perform its function repeatedly. It also characterizes the recoverability of the initial state from any stat of the system.

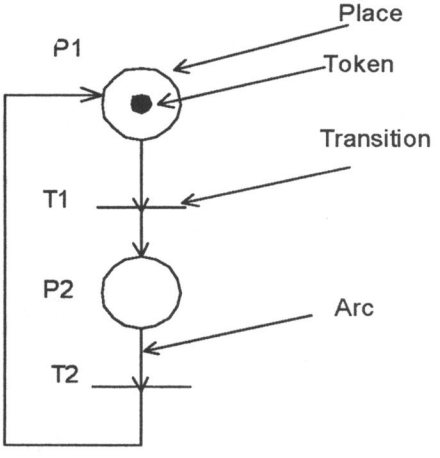

Fig.1. A simple Petri Net

The analysis of the Petri net models of the logic controllers may be used to verify the safeness, liveness, and reversibility properties to guarantee the desired behavior of the controlled system. There are three approaches in the analysis of the qualitative properties of the Petri net models: analysis by enumeration, analysis using linear algebraic techniques and analysis by transformation. Enumeration methods are based on the construction of the reachability graph RG or the converability graph. The state transition equation and invariance concept are used in the linear algebraic approach. The analysis of by transformation is based on simple reduction rules which prevent the properties of the Petri nets. The details of these techniques can be found in [3] [4]

C. *Real Time Based Petri Net Controller*

The stander Petri net does not deal with the inputs/outputs signals. The real time Petri net RTPN-based controller can be developed by assign physical output (actuator signals) function to places and assigning physical input functions (sensor signals) and timing variables to transitions of the Petri net. The state of the system can be represented by marking M of the Petri net, in the logic controller, the sequence of control actions is generated by the dynamic evaluation of the Petri net model.

III. MATLAB / STATE FLOW CONTROL SOFTWARE

Matlab is one of the most popular software tools in the area of applied mathematics and control engineering, it's becomes the standard environment to interchange ideas implemented in algorithms. Stateflow is a graphical design and development tool for control and supervisory logic used in conjunction with Simulink. It provides clear, concise descriptions of complex system behavior using finite state machine theory, flow diagram notations, and state-transition diagrams all in the same Stateflow diagram.

Stateflow uses a variant of the finite state machine notations established by Harel [5]. Using Stateflow, we can create Stateflow diagrams. A Statflow diagrams is a graphical representation of finite state machine where *state* and *transition* from the basic building blocks of the system. Stateflow provides a block that we can include in a Simulink model. In the Stateflow, the Stateflow diagram (called chart) consist of a set of graphical objects (states, transitions, connective junctions, and history junctions) and nongraphical objects (events, data, and target) as show in Figure (2). The *state* describes a mode of an event-driven system, which equivalent in sense to the place in the Petri net. The activity or inactivity of the states dynamically changes based on events and conditions. A *transition* is a graphical object that, in most cases, links one object to anther, one end of the transition is attached to the source and the other end to a destination object. A transition *label* describes the circumstances under which the system moves from one state to anther, this equivalent to the transition in the Petri net [5][6].

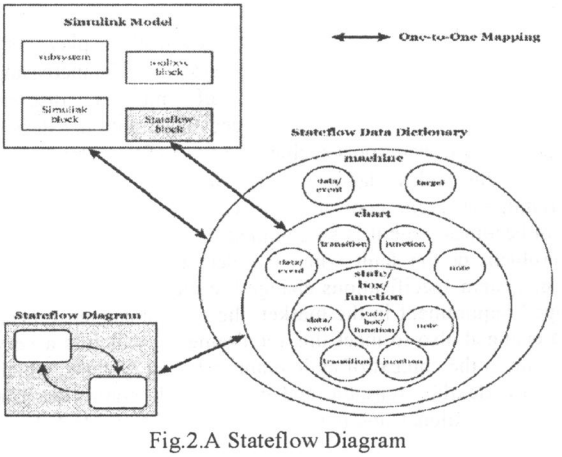

Fig.2.A Stateflow Diagram

IV. CONTROLLER DESIGN STEPS

The Petri net controller synthesis consists of several steps. The first step is constructing the Petri net model of the system under consideration. Then put the system in the form of Statflow representations as different modules connected together, each module consist of an event/action, then check both system performance controller validity by simulations. For the real time implementations, the inputs and outputs

(Sensors/actuators) signals can be assigned to the transitions and places of the modeled controller.

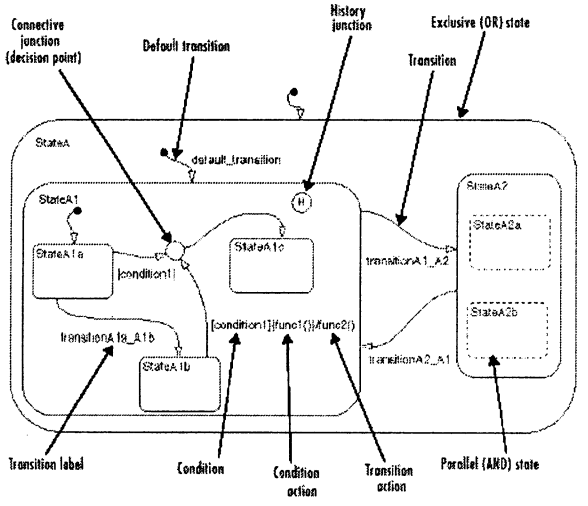

Fig.2.B Stateflow Machine

V. APPLICATION EXAMPLE

The simple automated machining and assembly system to demonstrate the usefulness of the proposed methods. This system only makes one type of product which needs two machining operations and one assembly operation. as shown in Figure (3), the system has one assembly station A1, one robot R1, and two machines, machine M1 and machine M2 respectively. Furthermore, we have the following assumptions for the system:

(1) The supply of row stock S1 and 2 in storage S1 and storage S2 is never exhausted.
(2) Finishing product will be taken away so that there is no overflow problem with storage S3.

The production is depicted as follow:

(1) The conveyor C1 (C2) transfers raw stock type type1 (type2) to M1 (M2).
(2) M1 (M2) start machining
(3) After M1 and M2 finish their operations, the robot R1 takes part1 from M1 to A1, and then takes part 2 from M2 to A1.
(4) R1 begins the assembly on A1 using part1 and part2.
(5) After R1 finishes the assembly on A1, it moves the product to storage S3.

The first step of the controller design is to construct the system modeling as shown in Figure (4) according to the place/transition Table 1, then construct the Statflow representations as a sub-modules (conveyor loading/unloading, robot, and assembly area modules) as shown in Figure (5) then combine the control modules together in a single module and check the designed controller validity by running the Matlab program module, which used for check the conflict in the design controller. Figure (6) shows the graph representing the simulations of job sequence for each device in the automated machining and assembly systems. This simulation is used for checking the designed controller performance and properties (liveness, Safeness, and Reversibility).

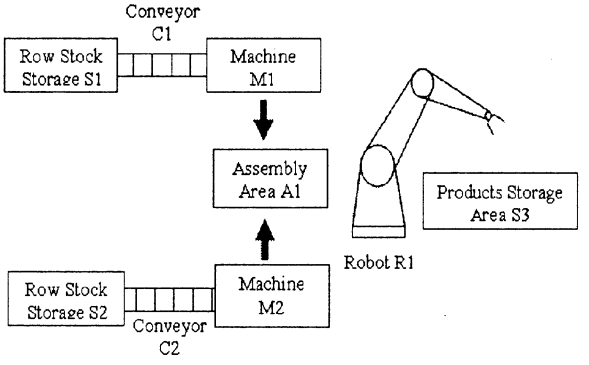

Fig.3. A simple automated and assembly system

TABLE I

PLACES AND TRANSITIONS OF THE PN MODEL

P1	M1 available
P2	Product/Conv. C1 is available
P3	Moving part 1 from S1 to M1
P4	M1 machining
P5	A1 available
P6	Part 1 available
P7	M2 available
P8	Product/Conv. C2 is available
P9	Moving part 2 from S2 to M2
P10	M1 machining
P11	Part 2 available
P12	R1 available
P13	Moving part 1&2 to loading area A
P14	R1 moving part 1&2 to A1
P15	R1 assembly the product 1&2
P16	Moving final product to storage S3
T1	Moving part1 to M1
T2	Machining part1 on M1
T3	M1 finishing
T4	Moving part2 to M2
T5	Machining part1 on M1
T6	M2 finishing
T7	Moving Part 1&2
T8	Part 1&2 on A1
T9	Part1 kept on A1
T10	End of assembly on A1
T11	Moving final product to S3

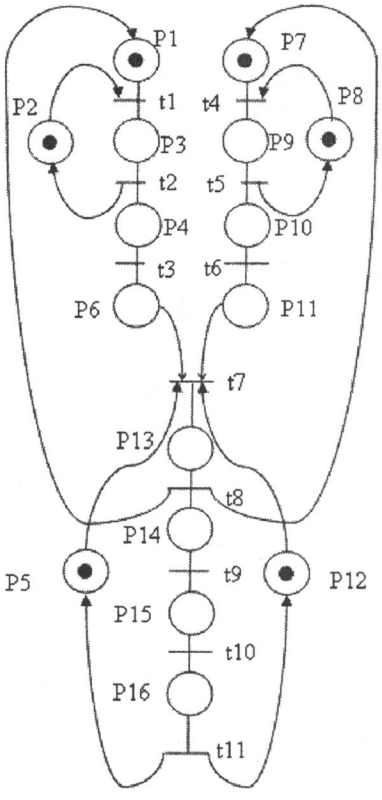

Fig.4. Structure of the Petri net model

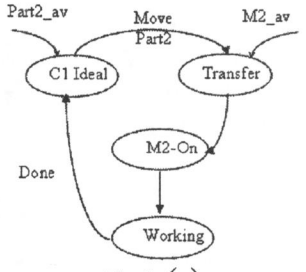

Fig.5. (a)

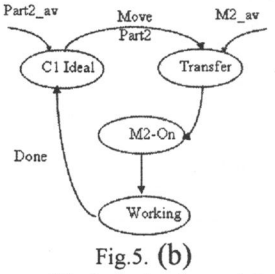

Fig.5. (b)

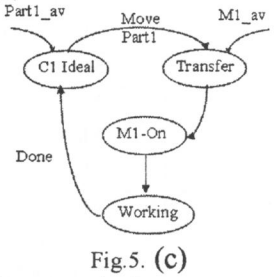

Fig.5. (c)

Fig.5. A simple model of subsystems, (a) Conveyer Loading/Unloading , (b) Robot Loading/Unloading , and Assembly area module.

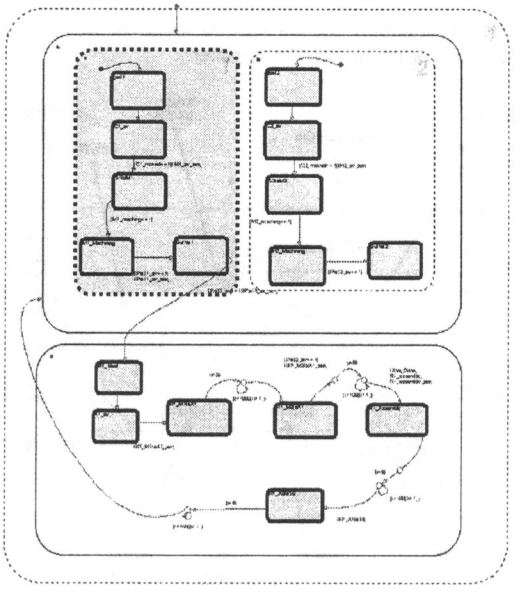

Fig.6. State Flow diagram of the automated and assembly line

CONCLUSION

This paper described a software tool (Stateflow-toolbox) that is used for modeling, simulation and evaluation the discrete event system. Its descried how a modular structure is used to construct a logic controller, and how these modules are verified. This software tool is a user friendly graphical user interface, and can be integrated with other Matlab environmental toolboxes which can be introduced it to real time implementations in a PC platform.

REFRENCES

[1] Zdenek Hanzalek "Matlab Based Petri Net Analysis," Technical Reports, Center for applied cybernetics, DCE FEE, Czech Technical University in Prague, 2003.

[2] T. Murata, (1989). "Petri Nets: Properties, Analysis ana Applications." *Proceedings of the IEEE*, Vol.77,no.4, Aprile1989. pp. 451-480. USA.

[3] M.Zhou, F. Dicesare "Petri Net Synthesis for Discrete Event Control of Manufacturing Systems," Kluwer Academic Publiisher, 1993.

[4] M.Zhou, F. Dicesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri net methods for manufacting systems," IEEE Trans. Robot. Automat., Vol.8, pp.350-361,June 1992.

[5] Harel, David, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming* 8, 1987, pages 231-274.

[6] Stateflow ® and Stateflow Coder ® User's Guide