Dependability Analysis of Logic Controller Based on Formal Verification Procedures

Saifulza Alwi and Nurrafidah Jaafar
Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Melaka, Malaysia.
saifulza@utem.edu.my

***Abstract*—Dependability of an automation system requires engineers to implement formal verification procedures in order to eliminate the causes of hazardous conditions. These conditions may vary from case to case and will jeopardize the dependability of manufacturing lines and operators. Therefore, dependability analysis of the control systems to check the possibility of state transitions from safe to unsafe states, for instance, is essential. Formal verification by using model checking procedure is proven as an effective method and widely used in practice for automatic verification of correctness properties against a finite model of a system. Therefore, in the present paper, we introduce a novel method of model checking for logic control design. A Binary Decision Diagram (BDD) based model checking method is used to analyze and design a dependable controller that meets the requirement of certain properties, defined by specified predetermined functions.**

***Index Terms*—Binary Decision Diagram (BDD); Computation Tree Logic (CTL); Programmable Logic Controllers (PLCs).**

I. INTRODUCTION

Design phase in the development of an automation system could be very critical. The fact is that the design engineers who involve in this phase must have technical knowledge and experiences, equipped with strong engineering capabilities to cope with the design requirements. Although beneficial for the engineers, the increasing number of embedded functions and features in the design of PLC-based manufacturing processes introduce more potential safety risks [1]. Thus, their effects on system reliability are much more unpredictable. Furthermore, these also lead to design errors that affect the functional behavior of the systems. Therefore, a successful system design depends on how verification is conducted to reduce the possibility of the design errors while at the same time eliminate the causes of hazardous conditions[2][3].

These issues of risks have brought us to the approach of formal verification by using model checking. Model checking is a formal technique to verify that a mathematical model of a system fulfills a formal specification that describes the property to be checked. The method is proven to be efficient and widely used in practice for automatic verification of correctness properties against a finite model of a system. Among the popular model checking technique is Binary Decision Diagrams (BDDs), that can represent set of states symbolically.

Our present work proposes in this paper is a new technique of model checking for logic control design. A Computation Tree Logic (CTL) temporal logic is used to analyze and design a dependable controller that meets the requirement of certain properties, defined by a predetermined function. The safety function, for instance, is obtained from hazardous conditions which are the property that may put the system into unsafe states. In addition, we verify the effectiveness of the proposed method by experiments on a pick-and-place arm system, controlled using a PLC. We describe the designed controller in the ladder diagram (LD) format. The control system and specified property functions are then transformed into Boolean formulas before analysis on the dependability of the system are carried out.

II. RELATED WORKS

The concept of formal methods has inspired design engineers to implement it for logic control design and synthesis. This is to ensure that no unusual conditions and hazardous behaviors occur that may lead to design errors that will cause malfunction of control systems. In other words, the aim is to improve the dependability of logic controllers. Starting from the implementation to digital functions for logical analysis [4] in the late 70's, BDDs have been widely applied in numerous research fields because of its 'powerful representation' of various kind of control systems. Bryant's approach of BDD manipulation algorithms to logic design verification [5][6] is a cornerstone of the new formalism of model checking technique. Burch et. al. introduced a BDD-based algorithm for symbolic CTL (Computation Tree Logic) model checking [7] with several techniques to improve the efficiency of verification methods based on reachability analysis. Later, instead of BDDs, an alternative approach which is SAT-based procedures for propositional satisfiability problems has been proposed to cope with the state space explosion when using BDDs [8]. This method is familiarly known as Bounded Model Checking (BMC). The performances of these techniques on diversified hardware benchmarks have also been investigated[9]. In addition to CTL model checking in formal verification of a system, analysis of linear time-based LTL (Linear Temporal Logic) specifications to verify safety properties also have been carried out thoroughly [10][11], giving the definition of safety in different manners.

On the other hand, one of the methods used [12] to synthesize control laws for the logic controllers is by solving a Boolean equation that represents all the requirements. Our proposed method introduced is basically followed this interpretation but uses different notions in the algebraic approach. A familiar concept of symbolic and algebraic method framework in discrete event system applications was

introduced by Gunnarsson [13][14]. He proposed a method of control law syntheses by using polynomials over a finite field and Gr¨obner bases as the computation tools. Formally introduced in 1965 [15], the application of Gr¨obner bases currently seems to be borderless. This powerful tool is originally developed for algorithmic solutions of polynomial ideal theory before it becomes an effective and practical method used to solve engineering and mathematical problems. Its applications are broad, for example in reliability improvement of mathematical structure [16], and also time-optimal control by solving problems of finding solutions for algebraic equations [17].

The unique representation of Gröbner bases algorithmic solution has given us the idea for applying it in formal verification procedures of logic control systems [18]. We also have successfully implemented several model checking procedures to verify the safety of the sequential systems by using this powerful method[19]-[22].

III. Logic Control System

Consider a logic control system consisting of discrete signals as shown in Figure 1. Here u is the actuator input and y is the sensor output. The system shows the relationship between the plant and the controller. In this paper, a plant represented in Boolean algebra is considered:

$$x_p(k+1) = f_p(x_p(k), u(k)) \quad (1)$$

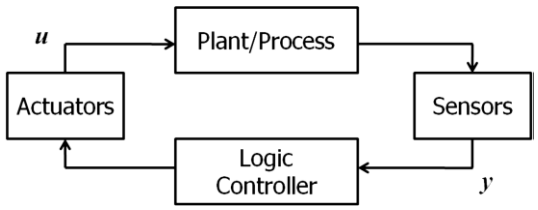

Figure 1: Control system architecture

Assuming that all of the variables, namely, the plant state x and input u, represented by vectors formed by the elements of a binary set, B of **true (1)** and **false (0)**, and having dimensions of n_p and m, then the following can be stated: $x_p(k) \in B_{np}$ is the state vector and $u(k) \in B_m$ is the actuator input vector of the plant. In addition, k is a positive integer that expresses time. Furthermore, the plant itself is a strictly proper system that whose input does not influence the output at the same event time.

Meanwhile, the controller is represented by the following state equation:

$$u(k+1) = h_c(x_c(k), u(k)) \quad (2)$$

where $x_c(k) \in B_{nc}$ represents the state vector of the controller, having a dimension of n_c at the event time k. Furthermore, the state equations of the entire system can also actually be represented by combining both the plant model and the controller.

The state space of the system also includes the input state space, which consists of both x_c and x_p. The entire system state space with $(n_p + n_c)$ order is a set of 2_{np+nc} state combinations. For this kind of system, physically unreachable state space is defined as infeasible space; otherwise, it is feasible space. In addition, when there is normal state space in certain control specification, it is defined as a safe state; otherwise, it is an unsafe state. Figure 2 categorizes the state into four patterns of combinations of feasible/infeasible and safe/unsafe states. A real system is normally controlled within the normal state space; however, in the case where disturbances and faults of the sensor or actuator occur, it is possible for the system to be out of the normal state space. In these situations, the controller must be designed so that it will avoid any state transitions to the unsafe space.

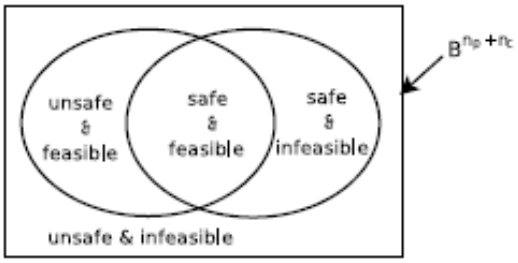

Figure 2: Category of state space

IV. Modeling of a Case Study Example

In this section, we report the experimental setup and modeling Boolean formulas of verifying a logic control system by using a pick-and-place arm workstation. We provide a plant model that describes the system configuration, then we design its control logic based on the regular operation. To verify whether the designed control system meets the safety specification, we take an example of a hazardous state that will be defined as a safety function, described in detail in a later subsection.

The pick-and-place arm workstation uses PLC programming software as its controller and the logic programming tool is based on the IEC 61131-3. Figure 3 illustrates the block diagram of the workstation, with a set of predetermined input state u and plant state x.

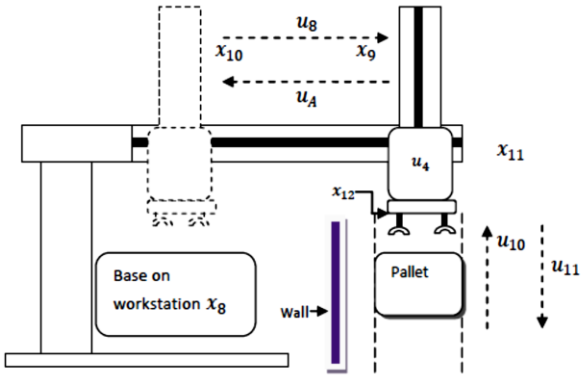

Figure 3: Schematic diagram pick and place arm

The state's definition, where all variable x stands for state inputs and all variable u for actuator outputs are summarized in Table 1.

By referring to Table 1, we create the pre-post condition table that shows the relationship between input and output of the model. This is the preliminary step in order to get the model of the entire pick-and-place system. Table 2 to Table 6 show the operational models of each arm at different states and conditions.

Table 1
States Definition of Inputs and Outputs

Input (Sensors)		Output (Actuators)	
x_8	Base at workstation	u_8	Moving from workstation to conveyor
x_9	Conveyor sensor	u_A	Moving from conveyor to station
x_{10}	Workstation sensor	u_{10}	Moving upward
x_{11}	Top sensor	u_{11}	Moving downward
x_{13}	Object in suction cup	u_{12}	Suction on

For example, in Table 3, the result of pre-post conditions of arm 2 is expressed by the following statements (case by case from first until the fourth row of the table):

1) When arm is at conveyor position if arm starts to actuate to the workstation, then on the next event time ($k+1$) arm will not be on conveyor nor at workstation position.
2) When the arm is not on the conveyor nor at the workstation position, if arm starts to actuate to the workstation, then on the next event time arm will be at workstation position.
3) When the arm is at workstation position, if arm starts to actuate to the conveyor, then on the next event time arm will not be on conveyor nor at workstation position.
4) When the arm is not on the conveyor nor at the workstation position, if arm starts to actuate to the conveyor, then on the next event time arm will be at conveyor position.

Table 2
Operational Model of Arm 1

Pre-Cond.		Input		Post-Cond.	
x_8	x_{11}	u_{10}	u_{11}	dx_8	dx_{11}
1	1	0	1	1	0
1	0	1	0	0	0
0	0	1	0	0	1

By the pre-post condition obtained in Table 2, we get the Boolean expressions as in Equations (3) and (4):

$$dx_8 = x_8 x_{11} u_{11} \overline{u_{10}} \vee x_8 \overline{\overline{x_{11}} u_{10} \overline{u_{11}}} \tag{3}$$

$$dx_{11} = \overline{\overline{x_8} x_{11} \overline{u_{11}} u_{10}} \vee x_{11} \overline{u_{11} \overline{u_{10}} x_8} \tag{4}$$

The procedure is similar to the rest of operational models, where we can produce the Boolean equations from the pre-postconditions.

Table 3
Operational Model of Arm 2

Pre-Cond.		Input		Post-Cond.	
x_9	x_{10}	u_A	u_8	dx_9	dx_{10}
1	0	1	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	0	0	1	1	0

From Table 3, we obtained the following equations:

$$dx_9 = \overline{x_9 x_{10} u_A} u_8 \vee x_9 \overline{\overline{x_{10}} u_A \overline{u_8}} \tag{5}$$

$$dx_{10} = \overline{x_9 x_{10} u_8} u_A \vee x_{10} \overline{\overline{x_9} u_8 \overline{u_A}} \tag{6}$$

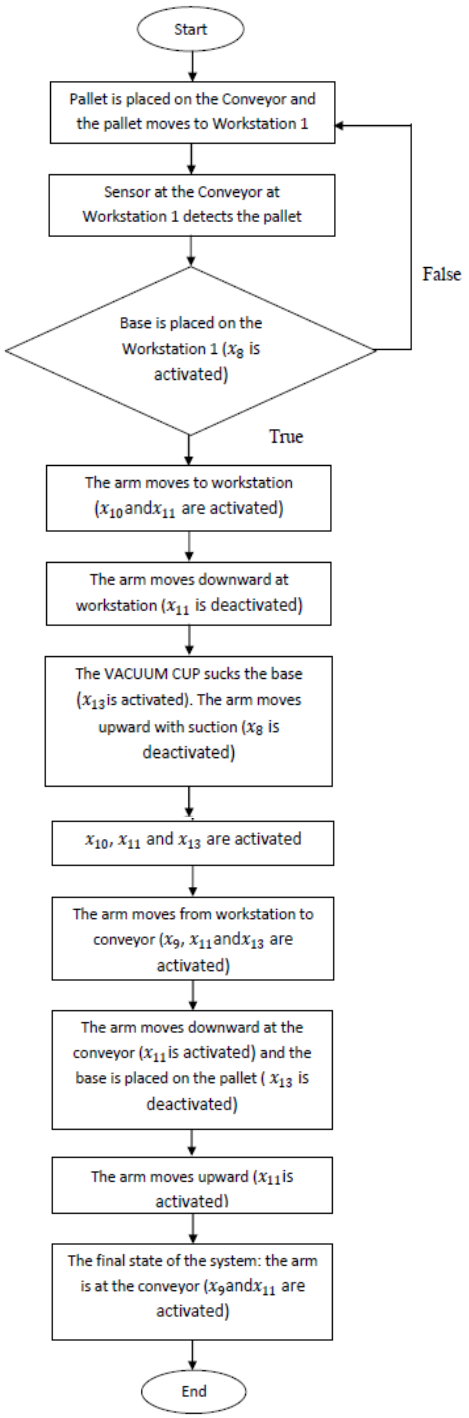

Figure 4: Flowchart of pick-and-place arm operation

Table 4 and Table 5 are the operational models for suction activation, which involves a number of sensors and input states. Therefore, the Boolean expression for this model is stated in Equation (7).

$$dx_{13} = x_8 \overline{x_9} x_{10} \overline{x_{11}} x_{13} u_{10} u_{11} \overline{u_{12}} \vee x_8 \overline{x_9} x_{10} x_{13} \overline{x_{11}} u_{10} u_{12} \overline{u_{11}} \tag{7}$$

From Table 5, the mathematical expression for suction actuation of arm 1 at conveyor can be written as:

$$dx_{13} = \overline{x_8}\,\overline{x_{10}}x_9x_{11}x_{13}\overline{u_{10}}u_{11}\overline{u_{12}} \vee x_{13}u_{10}u_{12}\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}\,\overline{x_{11}}\,\overline{u_{11}} \quad (8)$$

Table 4
Operational Model for Suction Activation of Arm 1 at Workstation

Pre-Cond.					Input			Post-Cond.				
x_8	x_9	x_{10}	x_{11}	x_{13}	u_0	u_1	u_2	dx_8	dx_9	dx_{10}	dx_{11}	dx_{13}
1	0	1	1	0	0	1	0	1	0	1	0	0
1	0	1	0	0	0	1	0	1	0	1	0	1
1	0	1	0	1	1	0	1	1	0	1	1	1

Table 5
Operational Model for Suction Activation of Arm 1 at Conveyor

Pre-Cond.					Input			Post-Cond.				
x_8	x_9	x_{10}	x_{11}	x_{13}	u_0	u_1	u_2	dx_8	dx_9	dx_{10}	dx_{11}	dx_{13}
0	1	0	1	1	0	1	1	0	1	0	0	1
0	1	0	0	1	0	0	1	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	1	0

The last operational model as in Table 6, is obtained for suction activation of arm 2, as in Equation (9).

$$\begin{aligned} dx_{13} = {} & x_8\overline{x_9}x_{10}x_{11}\overline{x_{12}}u_A\overline{u_8}\,\overline{u_{12}} \\ & \vee x_8\overline{x_9}x_{10}x_{11}x_{12}\overline{u_A}\,\overline{u_8}u_{12} \\ & \vee \overline{x_8}\,\overline{x_9}x_{10}x_{11}x_{12}\overline{u_A}u_8u_{12} \\ & \vee \overline{x_8}x_9\overline{x_{10}}x_{11}x_{12}\overline{u_A}\,\overline{u_8}u_{12} \\ & \vee \overline{x_8}x_9\overline{x_{10}}x_{11}x_{12}\overline{u_A}\,\overline{u_8}u_{12} \end{aligned} \quad (9)$$

Table 6
Operational Model for Suction Activation of Arm 2

Pre-Cond.					Input			Post-Cond.				
x_8	x_9	x_{10}	x_{11}	x_{13}	u_A	u_8	u_{12}	dx_8	dx_9	dx_{10}	dx_{11}	dx_{13}
1	1	0	1	0	1	0	0	1	0	0	1	0
1	0	0	1	0	1	0	0	1	0	1	1	0
1	0	1	1	0	1	0	0	1	0	1	1	1
1	0	1	1	1	0	0	1	0	0	1	1	1
0	0	1	1	1	0	1	1	0	1	0	1	1
0	0	0	1	1	0	1	1	0	1	0	1	1

V. Design of Control Logic

We use ladder diagram (LD) as a tool of PLC language in designing the logic controller. The controller is designed to operate and control the plant as the actual operation of the pick and place robotic system. Figure 4 shows the logic controller for the pick and place system, while Table 7 shows the addresses used in the ladder diagram.

For precautionary steps or safety measures to ensure that the controller is safe, every rung needs to include all the sensors and outputs. To differentiate, each rung has different sensors that need to be activated and deactivated with reference to the sensor involve for the actuation to occur.

From the controller, the Boolean mathematical model is generated in order to verify together with the system model using Symbolic Model Verifier (SMV) software. The following equations are the actuation equations generated from the PLC logic controller for every rung of actuation.

Table 7
Description of the address used in ladder diagram

Address	Description
0.08	Sensor sensing base
0.09	Sensor at conveyor
0.10	Sensor at workstation
0.11	Top sensor
0.13	Suction sensor
100.01	Moving from conveyor to workstation
100.08	Moving from workstation to conveyor
100.10	Moving upward
100.11	Moving downward
100.12	Suction

$$du_A = x_8x_{11}\overline{x_{10}}x_9\overline{x_{13}}u_8u_{10}u_{11}u_{12} \quad (10)$$

$$du_{11} = ((x_8x_{10}\overline{x_9}u_{12}) \vee (x_9x_{13}\overline{x_8}x_{10})) \,\&\, \overline{x_{11}}\,\overline{u_{10}}\,\overline{u_A}\,\overline{u_8} \quad (11)$$

$$du_{10} = x_8((x_{10}\overline{x_9}x_{13}) \vee (x_9\overline{x_{10}}\,\overline{u_{12}})) \,\&\, \overline{x_{11}}\,\overline{u_{10}}\,\overline{u_A}\,\overline{u_8} \quad (12)$$

$$du_8 = x_8((x_{10}\overline{x_9}\,\overline{u_{12}}) \vee (x_9\overline{x_{10}}x_{13})) \,\&\, x_{11}\overline{u_{10}}\,\overline{u_A}\,\overline{u_{11}} \quad (13)$$

$$du_{12} = \overline{x_8}((x_{10}\overline{x_9}) \vee (x_9\overline{x_{10}}))((x_{11}x_{13}u_8) \mid (x_{13}u_{10}) \mid (x_{13}u_{11}))\overline{u_A} \quad (14)$$

where,

du_A : post condition for actuation arm moving from conveyor to the workstation (rung 1)
du_{11} : post condition for actuation arm moving downward (rung 2)
du_{10} : post condition for actuation arm moving upward (rung 3)
du_8 : post condition for actuation arm moving from workstation to conveyor (rung 4)
du_{12} : post condition for actuation for suction (rung 5)

VI. Initial Analysis on Dependability

In this study, dependability is defined by several properties such as safety, resettability and reachability.

A. Property 1: Safety

Safety property is a forbidden state that is not supposed to occur or "nothing bad should happen". Safety property is important in model checking because it determines whether the system is safe to operate or not. If the verification result is FALSE, then the system is unsafe and there might be an error in the designed controller. For the pick and place system, the forbidden state is the arm under all circumstances will not move from conveyor to workspace (right and left movement) when it is at the bottom position.

EG – there exist a path for the specification to hold TRUE globally in the future.

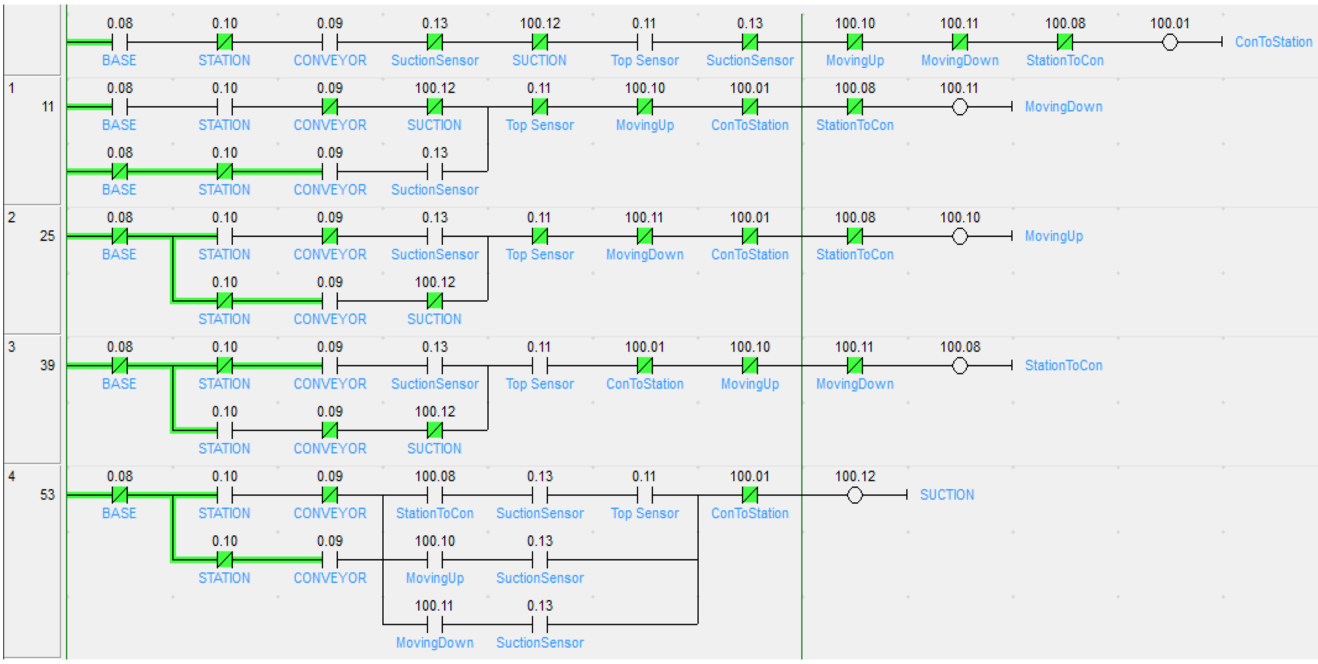

Figure 5: Logic controller for pick-and-place arm

$$\begin{aligned} &x_{11} = 0,\ u_A = 1 \\ &EG((\sim (\sim x_{11} \ \& \ u_A))) \end{aligned} \quad (15)$$

B. *Property 2: Reachability*

Reachability property is a property to check can the system reach to the certain next state from its present state. Given the system in a particular state, could it possibly reach a state which satisfies certain given properties sometimes in the future. For the pick and place system, the reachability specification can be written as Equation (16) which means at the present state where the arm is at the conveyor, top position, can it reach to the next state, which is at the workstation, top position.

AG – for every path the specification holds TRUE globally in the future.

EF – there exist a path for the specification to hold TRUE sometime in the future.

$$\begin{aligned} &x_9 = 1,\ x_{10} = 1,\ x_{11} = 1 \\ &AG((x_{11} \ \& \ x_9 \rightarrow (EF \ \ x_{11} \ \& \ x_{10})) \end{aligned} \quad (16)$$

C. *Property 3: Resettability*

Another property is resettability, which is a property to check whether the system at any present state, will be reset back to its original position. When arm at the bottom position at workstation and suction occurs, eventually it will reset back to its home position. The next equation is the specification written in the Symbolic Model Verifier (SMV) software to verify the robotic pick and place system.

AG – for every path the specification holds true globally in the future.

EF – there exist a path for the specification to hold true sometime in the future.

$$\begin{aligned} &x_{13} = 1,\ x_{10} = 1,\ x_{11} = 0,\ x_9 = 1,\ x_{11} = 1 \\ &AG((x_{13} \ \& \ x_{10} \ \& \ x_{11}) \rightarrow (EF \ \ x_9 \ \& \ x_{11}))) \end{aligned} \quad (17)$$

VII. Conclusion and Future Works

This paper discussed on designing a dependable logic controller for a robotic pick and place system. The context of dependable is defined by verification of the logic controller whether meets the defined temporal properties or not. In other words, the system must be verified before can be operated in for real application. To verify the dependability of the logic controller, properties such as safety requirement, reachability and resettability are the specifications used. In future, The simulation by using model checking software will be used to determine the independable condition of the system that may lead to system errors and malfunction so that precautionary steps can be taken earlier.

Acknowledgement

The authors are pleased to acknowledge the financial and administrative support from the Minister of Higher Education (MOHE), Malaysia and Universiti Teknikal Malaysia Melaka under the FGRS/1/2015/TK04/FKE/02/F00263 research grant project entitled “A Novel Method of Groebner Bases Computation for Distributed Discrete Controllers “.

References

[1] IEC, Functional safety of electrical/electronic/programmable electronic safety-related systems-Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3, *International Standard IEC 61508-6*, International Electrotechnical Commission, 2010.
[2] P. J. G. Ramadge and W. M. Wonham, “The Control of Discrete Event Systems,” *Proc. of the IEEE*, vol.77, pp. 81-97, 1989.
[3] C. G. Cassandras and S. Lafortune, “*Introduction to Discrete Event Systems*,” Springer Science + Business Media, Inc. 1999.
[4] S. B. Akers, “Binary Decision Diagrams,” *IEEE Trans. Comput.*, vol.C-27, pp. 509-516, June 1978.
[5] R. E. Bryant, “Graph-based Algorithms for Boolean Function Manipulation,” *IEEE Trans. Comp.*, vol. C-35, Aug. 1986.

[6] K. S. Brace, R. L. Rudell and R. E. Bryant, "Efficient Implementation of a BDD Package," in *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 759-764, 1990.

[7] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan and D. L. Dill, "Symbolic Model Checking for Sequential Circuit Verification," *IEEE Trans. Computer-Aided Design of Integ. Circuits and Syst.*, vol. 13, no. 4, pp. 401-424, April 1994.

[8] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, Y. Zhu,"Symbolic Model Checking Using SAT Procedures Instead of BDDs," in *Proc. 36th Annual ACM/IEEE Design Automation Conference*, pp. 317-320, 1999.

[9] N. Amla, X. Du, A. Kuehlmann, R. P. Kurshan and K. L. McMillan,"An Analysis of SAT-based Model Checking Techniques in An Industrial Environment," *Lecture Notes in Computer Science*, SpringerLink, 2005.

[10] O. Kupferman and M. Vardi, "Model Checking of Safety Properties," in *Formal Methods in System Design*, 2001.

[11] T. Latvala, "Efficient Model Checking of Safety Properties," in Model Checking Software, *10th Int. SPIN Workshop*, USA.

[12] Y. Hietter, J. M. Roussel and J. J. Lesage, "Algebraic synthesis of dependable logic controllers," in *17th IFAC World Congress*, Seoul,Korea, 2008.

[13] J. Gunnarsson, "*Symbolic Methods and Tools for Discrete Event Dynamic Systems*," Ph.D. Thesis, Linköping University, Sweden, 1997.

[14] J. Gunnarson, "Algebraic Methods for Discrete Event Systems – A Tutorial," *Proc. of IEE WODES'96*, Edinburgh (GB), pp. 18-30, 1996.

[15] B. Buchberger, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory," in *N. K. Bose Ed., Multidimensional Systems Theory*, D. Reidel Publishing, pp. 184-232, 1985.

[16] B. Giglio, Daniel Q. Naiman and H. P. Wynn, "Gröbner Bases, Abstract Tubes, and Inclusion-Exclusion Reliability Bounds," *IEEE Trans. Reliability*, vol. 51, no. 3, pp. 358-366, Sept. 2002.

[17] U. Walther, T. T. Georgiou and A. Tannenbaum, "On the Computation of Switching Surfaces in Optimal Control: A Gr¨obner Bases Approach," in *IEEE Trans. Auto. Control*, vol.46, no.4, pp. 534-540, April 2001.

[18] S. Alwi and Y. Fujimoto, "Formal Verification of Logic Control Systems with Nondeterministic Behaviors", in *IEEJ Journal of Industry Applications* 2(16), pp. 306-314, 2013.

[19] Saifulza bin Alwi, "*Verification and Validation of Logic Control Systems by Model Checking*", Ph.D Thesis, Yokohama National University, 2013.

[20] S. Alwi and Y. Fujimoto, "Safety Property Comparison between Gr¨obner Bases and BDD-based Model Checking method" in *13th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Singapore. 2014.

[21] S. Alwi and Y. Fujimoto, "On A Safety of Sequential Control System based on Gr¨obner Bases Computation" in *2010 Int. Conf. On Control, Automation and Systems (ICCAS2010)*, KINTEX, Korea.

[22] S. Alwi and Y. Fujimoto, "A Gr¨obner Bases approach for Safety Evaluation of Logic Control System", in *Proc. IEEE Industrial Informatics (INDIN)*, Osaka, Japan, 2010.