Industrial Control Of Discrete Event Systems Using Coloured Petrinet

Mark Marvellous, Arokoyu Olaoluwa

Abstract— As DES systems become more complex, the need for a means of handling this complexity becomes more and more imminent and to this effect several tools have been created. One of these tools is known as Petri Nets. Petri Nets allow an easily understandable graphical representation of the system using a flow chart, Grafcet and Finite State Machines (FSM) as its language for designing a system. As the system gets bigger, so does the number of states so it might be more complex to handle manually for very large systems.

This report documents the use of Petri Net supervisors to provide control for a Programmable Logic Controller (PLC) for the control of a manufacturing process involving complex assemblies. PLCs use Ladder Logic to operate and as such, a Petri Net to Ladder Logic conversion is necessary and this will be achieved using the Token Passing Logic (TPL) methodology. The complete Ladder Logic Program is then going to be implemented using Supervisory Control And Data Acquisition (SCADA) so that the activities and controls of the PLC on the Industrial Control Trainer (ICT) can be seen and controlled remotely.

Index Terms— GRAFCET, Laddar Logic, Petrinets, Programmeable Logic Controllers (PLCs), Token, Transition.

1 INTRODUCTION

Several solutions have been imbibed into science and technology over the years to proffer solutions to a myriad of both simple and complex technological tasks of which PLCs stand out. Early PLCs were designed to replace relay logic systems and the program used for the PLCs was/ is known as ladder logic which looked very much like the schematic diagram of relay logic.(Jack, 2010) Up to the mid 1990s, PLCs were programmed using special purpose programming terminals which had dedicated function keys representing the various logical elements of PLC programs. In modern times, PLCs can be programmed using application softwares on personal computers and the logic represented in graphic form instead of character symbols. The programming software allows for entry and editing of the ladder logic.
It can be connected to many inputs and outputs depending on the configuration of the inputs and its internal states, and these extensive arrangements can as well be connected
to sensors and actuators ranging from pneumatically actuated to electrically driven devices (*Bender,2008*).
Most modern PLCs can communicate over a network to other systems using Supervisory Communication and Data Acquisition (SCADA) systems which is run on a computer.
PLC programs are usually executed repeatedly for as long as the system keeps running. The program runs from the first rung to the last rung evaluating each rung and marking its inputs and outputs as several conditions are met with and then it updates the I/O image table with the outputs of each rung.
This paper seeks to model a system for a single chute ICT; which is a mimic of a typical industrial manufacturing process line in order to produce a set of possible combinations at the assembly stage using coloured Petri Nets.

- *Marvellous is currently an automation engineer in flour mills of Nigeria PLC Lagos. Obtained an MSc. In advanced control systems from University of Salford UK.. PH-+2348090995532. E-mail:markmp01@gmail.com*
- *Co-Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789. E-mail: author_name@mail.com*
(*This information is optional; change it according to your need.)*

2. Petri-Net

A PN is defined as a bipartite graph which consists of two types of nodes: 'places' and 'transitions' connected by directed arcs.

Place: Denotes a condition within the system being modelled and it is represented by a circle.
Transition: This is an event occurring in the system that may cause a change in the condition of the system and it is often represented by a bar or box.
An arc: Connects a place to a transition or a transition to a place.

A simple PN is shown below.

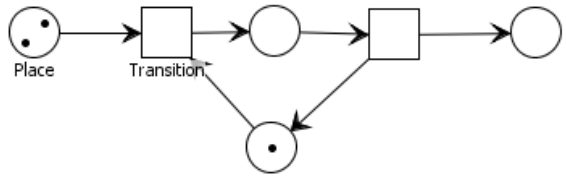

Fig. 2.0.1; A simple petrinet

A transition is enabled/fireable when each of its input places has at least one token. A transition needs to be enabled before it can be fired. A transition fires by removing a token from each of its input places and placing a token in each of its output places; thus allowing tokens to flow through the chart. The PN execution is controlled by the number of tokens in the net and their distribution and this makes the tokens to flow from place to place in the net.

A PN is defined as a bipartite graph which consists of two types of nodes: 'places' and 'transitions' connected by directed arcs.

- Σ is a set of colour sets defined within CPN model. This set contains all possible colours, operations and functions used within CPN.
- C is a colour function. It maps places in P into colours in Σ.
- N is a node function. It maps A into $P \times T \cup T \times P$.
- E is an arc expression function. It maps each arc $a \in A$ into the expression e. The input and output types of the arc expressions must correspond to type of nodes the arc connected to.
- G is a guard function. It maps each transition $t \in T$ into guard expression g. The output of the guard expression should evaluate to Boolean value true or false.
- I is an initialization function. It maps each place p into an initialization expression 'i'. The initialization expression must match a multi-set of tokens with a colour corresponding to the colour of the place C(p).

2.1 UNFOLDING OF COLOURED PNs

Unfolding *(Bonet, 2008)* is a method for reachability analysis which exploits and preserves concurrency information in the Petri net. It is a partially ordered structure of events that represents all possible firing sequences of the net from the initial marking. Unfolding is very important in CPNs because it enables a good visualization of the possible events that could occur when various colours of tokens operate in a net and their concurrency needs to be ensured. Unfolding describes the flow of a token colour through the possible places of its transitions for all the given token colours and interrelates each of them with each other using inhibiting arcs. Fig.2.0.2 illustrates a typical queue with one token colour in it.

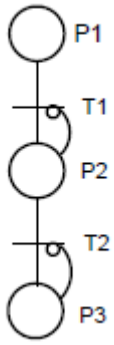

Fig 2.0.2; single queue petrinet design.

The figure above shows a single queue with only one colour of token(s) in it. The tokens will move from P1 to P2 and finally to P3 as long as the succeeding place has no token in it.

Let us say the same queue had four different colours of tokens in it, each colour would need to be represented by a single queue of its own (Note: The different queues only represent the different colours. In reality, it is still one queue with different coloured tokens in it.). Let the colours of the tokens be Red, Blue, Yellow and Green respectively, then the resultant unfolded PN is as shown below

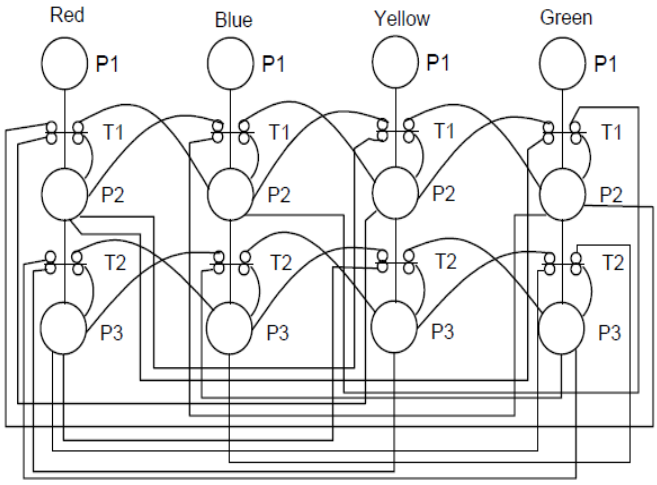

Fig. 2.1.1; unfolded form of single queue.

Inhibiting arcs connected to all transitions T1 ensure that if a single colour -token (say red) goes in first, no other token is allowed into the queue for as long as the red token remains in P2. A token can only enter the queue when T2 fires the red token into P3. Complexity is a major issue in coloured petrinets as an increase in the number of token colours increases the number of columns and makes the system very complex

2.2 SAFE COLOURED PNs

To help reduce the complexity of handling unfolded CPNs, another method has been employed and this method is known as the safe CPNs method. The idea is to handle CPNs without having to battle with the horizontal complexness of the system. To help keep the system simple irrespective of the number of colours within the net, a method to reduce the entire system into two (2) columns has been developed such that one column represents the token(s) and the other represents their corresponding colours.

To determine exactly what is going on in a particular place, the two places opposite each other are interpreted collectively. The token column shows the number of tokens in a particular place (which will always be equal to one in a queue) while the colour column shows what colour of token is occupying that place. Numbers are allocated to colours such that all the colours in the system can be identified by knowing the number associated with that colour.

To further explain this concept, the illustration used in 3.5 will be re-implemented using the safe CPNs method. Let the four token colours and their associated numbers be;

Red = 1

Blue = 2
Yellow = 3
Green = 4

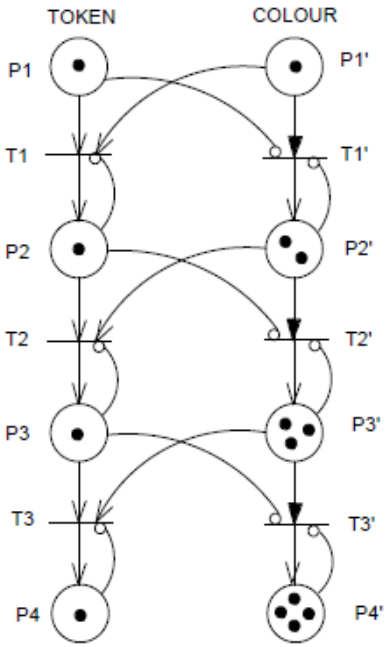

Fig 2.2.1; Safe petrinet of a single queue.

Since this is a single queue, it means the maximum number of tokens allowed in a particular place at a particular time on the 'token' column is 1. P1 and P1' indicate that there is one red token in P1. P2 and P2' indicate that there is one blue token in P2. P3 and P3' indicate that there is one Yellow token in P3. P4 and P4' indicate that there is one green token in P4.
An inhibiting arc connects a place on the 'token' column to the succeeding transition on the 'colour' column and this is done to ensure that the colour associated with a token remains in place with that token.
An enabling arc connects a place on the 'colour' column to the succeeding transition on the 'token' column and this is done to ensure that the transition can be fired when the corresponding colour attached to a token is in place. It further helps to check that the token and its associated colour flow together for ease of identification.
Emptying arcs employed after every place on the 'colour' column ensure that all the tokens in that place are removed and passed on to the succeeding place so that the colour of the associated token remains unchanged as it moves from place to place.
The design maintains the FIFO principle which ensures the order of tokens is kept constant.
As the number of token colours increase, the system does not increase horizontally.

The implementation of the safe CPNs method show a much more neater, easy to understand and less complex way of handling several colours in a PN than that of the unfolded CPNs method. A mere glace at the diagram shows that it is not just simpler but also as equally sufficient for handling a wide variety of colours in a net. To this effect, the safe CPNs method will be used for the purpose of this project to further illustrate its efficiency and practicability for designing real life systems.

3.0 Design and Automation of a single chute Industrial control trainer

The Industrial Control Trainer is a sorting, assembly and inspection process that is a similar representation in operation to many full sized systems such as many manufacturing operations as well as packaging, labelling and bottling plants. The ICT Comprises of two conveyors arranged so that components may be transferred from the first (upper) to the second (lower) by means of linear and rotary solenoids and a pair of chutes. A variety of sensors mounted on the conveyors allow the positions of components of different types to be identified. The unit may be controlled by a PLC, Personal Computer (PC) or microprocessor trainer board.

The ICT has two conveyors that allow the transportation of pegs and rings across the system. On the upper conveyor, a random selection of plastic rings, metal rings, plastic pegs and metal pegs are identified and separated by three sensors; object sensor, peg sensor and metal sensor. Upon detection of a ring (plastic or metal) an actuator (sort solenoid) knocks the ring down the ring chute as long as the number of rings in the chute is less than 6 i.e. 0 - 5. Pegs will be detected but left unattended to so that they can slide down the peg chute and automatically go on to the lower conveyor belt (LCB).

At the assembly area, pegs and rings are coupled together. An assembly area sensor is used to check if there is a ring in the assembly area and if/when it is empty, a ring is released by the rotary solenoid into the assembly area where it sits until a passing peg picks it up to form a complete assembly.

As components keep moving on the lower conveyor belt, a complete assembly sensor which detects only complete assemblies senses a passing component if it is a complete assembly and turns on a timer to deactivate the reject solenoid for a period short enough to allow the complete assembly to go through the reject area and into the complete assembly area. If the complete assembly sensor goes not detect a passing component, then the reject sensor triggers the reject solenoid to reject any component it detects into the scrap area.

To conserve power, the lower conveyor motor comes on only when a peg is detected at the upper sort area.

Sensor No	**Sensor type, location and function**	**PLC Input**
1.	Downward-looking reflective IR sensor at the upper sort area. Detects the presence of a peg near to and in front of the solenoid at the top of the ring chute.	I:0/4
2.	Sideways-looking reflective IR sensor at the upper sort area. Detects a component in front of the solenoid at the top of the ring chute.	I:0/1
3.	Reflective IR sensor at the assembly area. Detects the presence of a component at the very bottom of the ring chute beyond the rotary solenoid.	I:0/0
4.	Black pushbutton. Used to commence assembling, say.	I:0/18
5.	Red pushbutton. Used to terminate assembling, say.	I:0/19
6.	Capacitive sensor, near the lower sort area. Detects the presence of passing complete assemblies near the reject solenoid at the upper end of the lower conveyor.	I:0/6
7.	Reflective IR sensor at the capacitive sensor near the lower sort area.	I:0/3
8.	Reflective IR sensor at the lower sort area. Detects the presence of components and assemblies in front of the reject solenoid at the motor end of the lower conveyor.	I:0/2
9.	Through-beam IR sensor, just after the assembly area. Detects components on the lower conveyor leaving the assembly area.	I:0/5
10.	Metal sensor. Detects the presence of a metal component at the top of the ring chute.	I:0/7

Actuator No.	**Actuator type, location and desired function.**	**PLC Output**
1.	Upper conveyor motor. Drives the upper toothed chain conveyor	O:0/3
2.	Lower conveyor motor. Drives the lower plain belt conveyor.	O:0/4
3.	Solenoid at upper sort area. Knocks into the ring chute.	O:0/0
4.	Rotary solenoid at the bottom of ring chute before the assembly area. Releases ring into the	O:0/1

	assembly area.	
5.	Solenoid at the reject area. Rejects unassembled components before the complete assembly collection tray.	O:0/2

Table 3.1 A Brief Description of the Sensors and Actuators

used in The ICT.

3.1 Design Requirement/constraints

A set of requirements necessary for the ICT to produce optimally with very few rejects are as follows;

- The black push button initializes the upper conveyor motor.
- The red push button deactivates both conveyor motors.
- No more than five rings in the ring chute before the rotary solenoid, hence a ring chute counter was put in place.
- No more than two rings trying to get into the assembly area at the same time, hence a timer was attached to the rotary solenoid to allow the solenoid to be activated for just long enough to allow only one ring through.
- For the timing of the system to remain efficient, the pegs, rings and chutes must be buffered regularly.
- Complete assembly should be collected in the complete assembly tray while incomplete components (pegs) at the lower sort area should be rejected.
- Surplus rings should be collected in the surplus rings tray.

The constraints incorporated into the ICT program design include;

- When the number of rings in the ring chute is greater than five (counter C5.1 > 5), the actuator should not kick any ring into the ring chute.
- No pegs should be knocked down the ring chute.
- The rotary solenoid cannot allow a ring into the assembly area whenever the assembly area is not free.
- The reject solenoid should not knock off a complete assembly.

3.2 PNs for the different design stages

PN methodology is used to provide a good model for the single chute ICT implemented on a Bytronic ICT system. Essentially, the design is made to sort, assemble and screen components (red, blue, yellow, green and black) to ensure that only desired system requirements are achieved. These requirements are highlighted thus;

Metal pegs(MP), Metal Rings(MR), Plastic Pegs(PP), and Plastic Rings(PR) are assembled to form the following possible combinations at the assembly area. MP—PR, MP-MR, PP-PR, PP-MR.
By separating the entire system into four sections some level of independency will be achieved.

- initialization
- Sort area (Upper)
- Assembly area
- Reject area (Lower)

3.2.1 Initialization

Initialization is achieved by pushing a black pushbutton which turns on the upper conveyor motor to allow components pass through the upper sort area.

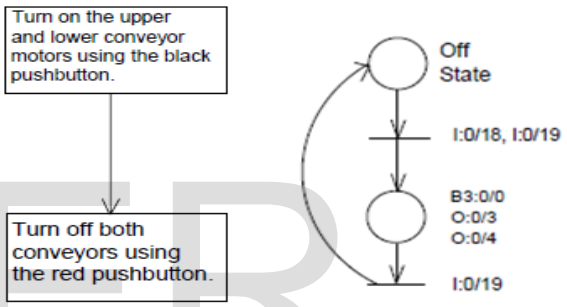

Fig 3.2.1(A) flowchart and (B) PN initialization of the ICT

3.2.2 Sort Area

The upper sort area consists of metal detector (I:0/7), object sensor (I:0/1), peg sensor (I:0/4), sort solenoid (O:0/0) and a ring counter (C5:1, ring chute count < 6)
Components are detected by a combination of different inputs to the sensors such that;

PR - I:0/1 true and I:0/4, I:0/7 false.
MR - I:0/1, I:0/7 true and I:0/4 false.
PP - I:0/1, I:0/4 true and I:0/7 false.
MP- I:0/1, I:0/4 and I:0/7 true.

After the successful identification of the components at the upper sort area, the identified components are then recorded as colours in the design and it is theses colours that the system works with henceforth. The colours representing the various components are shown below;

Metal Peg = Red

Plastic Peg = Blue

Metal Ring = Yellow

Plastic Ring = Green

The upper sort area can only have a token in it at a time and as such it is a mutually exclusive system. The Petri Nets diagram for the identification of token colours is shown below.

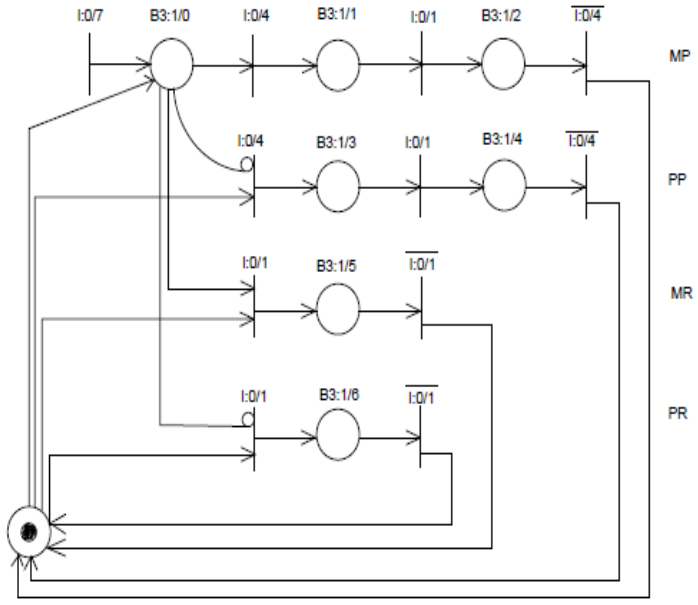

Figure 3.2.2 (A) Mutually Exclusive Sort Area PN

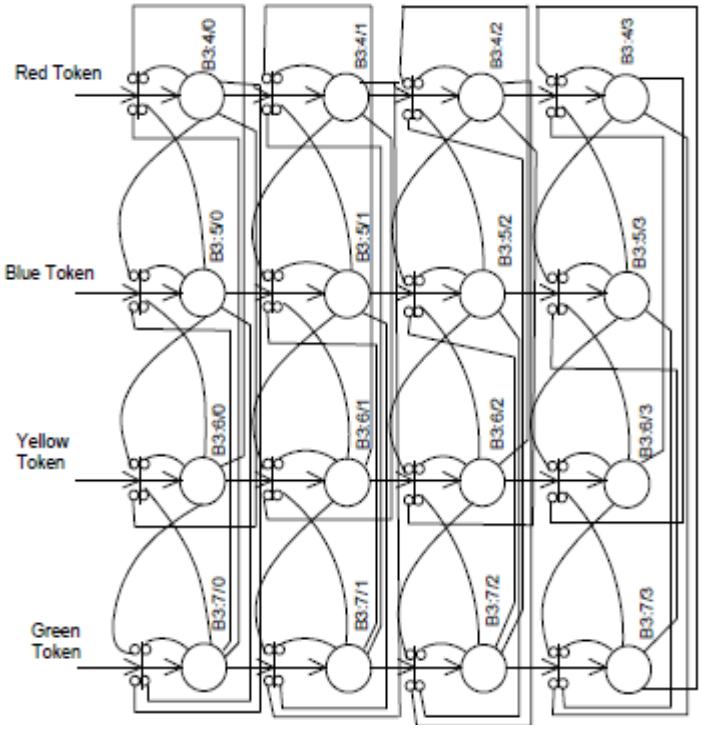

Figure 3.2.2 (B) Mutually Exclusive Sort Area

After the successful identification of the various colours, the next step is to allow the upper sort area solenoid to knock down certain colours into the ring chute. Let the colours be yellow and green. The ring chute has a capacity of five (5) so only 5 rings can be in the ring chute at any given time. When MR or PR becomes identified and the number of rings in the ring chute is less than five, the upper sort area solenoid then knocks that ring into the ring chute. The mutual exclusiveness of the upper sort area makes sure only one token remains in the region so to further ensure the right order is maintained a mutually exclusive queue is put in place to make sure a certain action is only possible if/when all other similar actions are not already occurring. There are four possibilities (red, blue, yellow and green) and as such a transition is only enabled if the succeeding place and all other similar places inhibiting that transition are empty. To further buttress this point, the CPNs design is shown below.

3.2.3 Assembly Area

As rings are knocked into the ring chute, a sensor mounted as the assembly area (assembly area sensor, (I:0/1) activates the rotary solenoid (O:0/1) for 0.7 seconds to allow a ring into the assembly area only when the assembly area sensor does not detect the presence of a ring. As the ring settles into the assembly area, the ring chute counter (C5:1) is decreased by one (1). This process reiterates until there is no ring left in the ring chute i.e. ring chute count is zero (0).

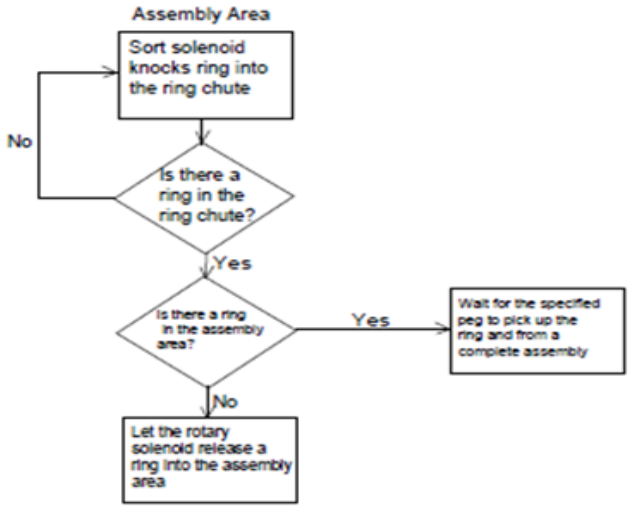

Fig. 3.2.3 (A) Flow Chart for the assembly area

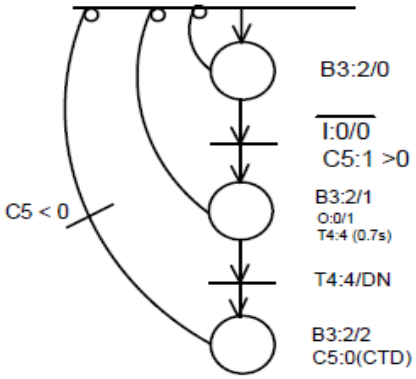

Figure 3.2.3 (B) PN For The Assembly Area

3.2.4 Reject Area

The lower sort area is an important stage of the design because the formation of complete assemblies produces a new component which requires a new token colour to be introduced into the system. Let the new token colour be black such that the colours in the system now become:

Red - Metal peg

Blue - Plastic peg

Yellow - Metal ring

Green - Plastic ring

Black - Complete assemblies

When a yellow or green token is knocked into the ring chute, the previous states associated with those colours are unlatched. However, if a yellow or green token makes its way through the upper sort area without being knocked into the ring chute, the reject area is free to knock the component into the plastic tray as the identified object will not be a red, blue or black token. Red, blue and black coloured tokens should not be found in the plastic tray and yellow and green components should not be found in the metal tray. To achieve this, the through-beam infrared sensor (I:0/5) is made to detect a passing red, blue or black token which latches on a new state (B3:8/0). A reflective infrared sensor (I:0/3) confirms the passing of a component and when (I:0/2) detects a component, a counter (C5:2) counts up by one. However, if the through-beam infrared sensor does not detect any component, the reject area solenoid is free to knock of any object detected by the reject area sensor. The Flow Chart and PN diagrams for the rejection area are shown below.

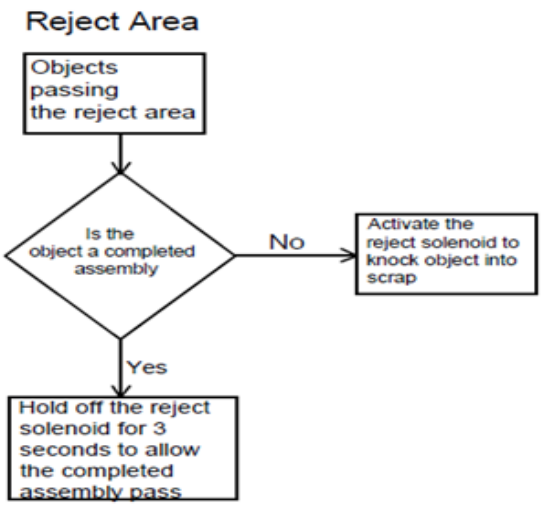

Figure 3.2.4 (A) Flow Chart For The Reject Area

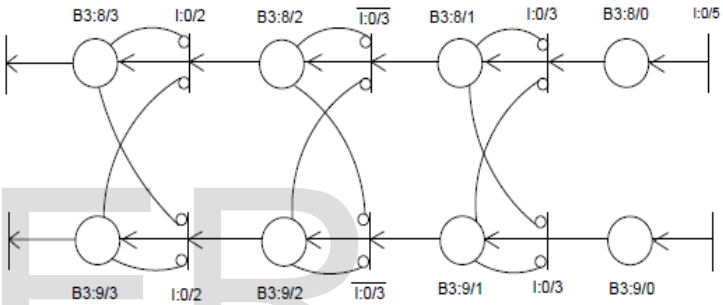

Figure 3.2.4 (B) PN For The Lower Sort Area

The PN design shows the objects (tokens) on the lower conveyor belt after they have gone through the upper sort area. In order to get a clear picture of the design, using the PN approach is the best way as it reduces the complexity of the scenario. Once again inhibiting arcs have to be used to prevent unwanted events from occurring. Every place inhibits the transition before it from being enabled so that a certain desired order is maintained throughout the system as shown in Figure 3.2.4 (B).

3.3 FINAL PN DESIGN FOR THE ICT

3.3.1 UNFOLDED CPNS METHOD

The PN design is to enable the correct selection of specific token colours into their preprogrammed collection trays. Putting all the four sections explained earlier into one coloured Petri Nets diagram enables a good visualization of the entire system. This has been adequately provided for and the overall unfolded CPNs design for the proper functioning of the ICT to achieve this is shown in figure 3.3.1 The unfolded coloured Petri Nets method for this design is explained and shown in this section.

After the successful identification of the four components, red and blue tokens are sent down the peg chute while yellow and green tokens are sent down the ring chute by the upper sort area solenoid. The capacity of the ring chute is five (5), and as such this action is reiterated for as long as the number of tokens in the ring chute remains less or equal to five (5). The yellow and green tokens settle into the ring chute and the rotary solenoid opens up for just long enough to allow one ring into the assembly area only if there is at least one ring in the ring chute and there is currently no token in the assembly area. After the token settles into the assembly area, it simply waits for a peg to come and pick it up and this produces a fifth possible entity on the lower conveyor belt which happens to be a 'complete assembly' and is identified by the system as a black token. No matter what combination of tokens (red, blue and yellow, green) that produce a 'complete assembly', the result is a black token. On the other hand, if a red or blue token goes through the assembly area while there is no token in it, the result is a red or blue token on the lower conveyor belt. Under normal operating conditions, a yellow or green token should not be found on the lower conveyor belt but in any case, if a yellow or green token makes its way to the lower conveyor belt it will still be identified and will be adequately handled at the lower sort area/reject area. This design allows the lower sort area ensure that only red, blue and black tokens make their way into the metal tray and yellow or green tokens go into the plastic tray. The type of token that goes into a particular destination can always be changed by redirecting the destination of the token in the program (Ladder Logic).

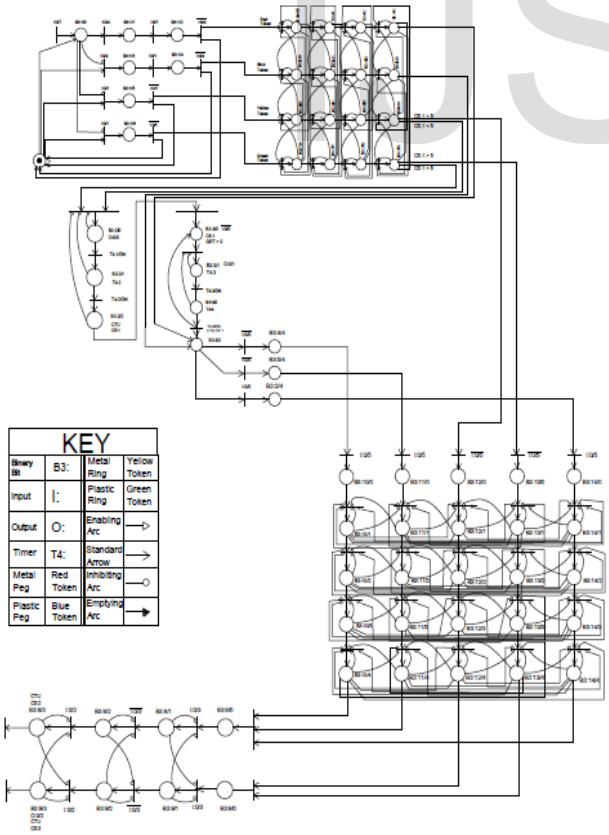

Figure 3.3.1 Final Unfolded CPNs Design

3.3.2 SAFE CPNS METHOD

The same design shown in 3.3.1 will be implemented using the proposed safe coloured Petri Nets methodology. In this case, everything is almost the same except that the token colours are associated with numbers. The number is attached to a token colour using a numerical bit (N7:) and it is with this number that the system determines what happens to the token it is associated with. Due to the mutual exclusiveness of queues, this method is able to thrive as it also obeys the FIFO rule.

The method basically operated using two columns and a couple of rows limited by the capacity of the queue or the number of colours in the system. The first column contains the number of tokens which is always one (1) in this case because the design allows for only one token in a place at a time. The second column contains the colours of the tokens and these colours are identified by associating a number to a colour such that the number of 'tokens' in any place on this column is indicative of the colour of the token at that place. Two mutually exclusive queues where used in this design. The first one is located at the upper sort area and the colour numbers associated with the identified tokens are as shown below.

Red Token = 1

Blue Token = 2

Yellow Token = 3

Green Token = 4

The second queue is located just before the lower sort area (reject area) and because of the introduction of a new token colour as a result of the formation of 'complete assemblies', the lower sort area queue has five (5) different token colours and the colour numbers associated with the identified tokens are as shown below.

Red Token = 6

Blue Token = 7

Yellow Token = 8

Green Token = 9

Black Token = 10

Implementing the safe coloured Petri Nets Design using ladder logic requires the use of move logic operations. There are two basic types of move functions;

1. MOV (value, destination) - moves a value to a memory location.
2. MVM (value, mask, destination) - moves a value to a memory location, but with a mask to select specific bits.

The MOV operation will move a value from one location in memory and place it in another memory location. When A becomes logically true, the MOV operation moves a floating point number from the source to the destination address. The data in the source address is left unchanged. When B becomes logically true, the floating point number in the source will be converted to an integer and stored in the destination address in integer memory. The floating point number will be rounded up or down to the nearest integer. When C becomes logically true, the integer value of 123 will be placed in the integer file N7:23. The value in the old location remains unchanged so the integer value of '0' is put into the previous location and this is done as a means of resetting the value in that location. On the move logics used in the queues, odd numbers (N7:1/3/5/7) represent token colours while even numbers (N7:0/2/4/6) represent token numbers.

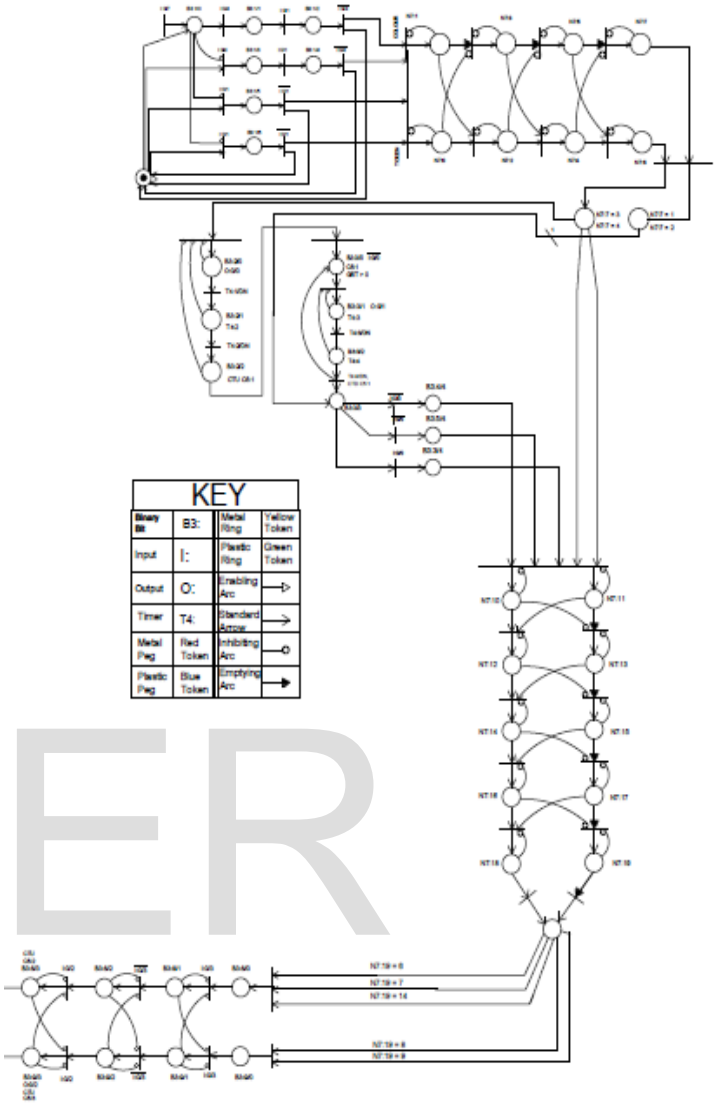

Fig. 3.3.2; Final Safe CPNs Design.

Comparing the two methodologies and their final Petri Nets diagrams shown in figures 3.3.1 and 3.3.2, it can clearly be seen that the exact same design was achieved with a great deal of complexity removed by adapting the safe coloured Petri Nets methodology. The safe coloured Petri Nets methodology is simpler, less complex and not as clumsy as the regular unfolded coloured Petri Nets methodology.

5.0 CONCLUSION

Concurrent systems, processes and events can easily be modeled using Petri Nets as they allow for Larger reachable state spaces, a more versatile complexity and they also allow for a more compact representation. Compared to the regular

unfolding of Coloured Petri Nets which poses to end up very complicated as the number of colours in the system increases, a new method which proves to be impeccably accurate and less complicated for handling systems with a wide variety of colours in them has been developed. This method presents a simple, easy-to-use approach for handling the design and implementation of an industrial manufacturing process using Safe Coloured Petri Nets method along with GRAFCET and Ladder Logic and it does not increase horizontally as the number of token colours in the system increases.

In this report, the Petri net concepts have been extended to deal with Petri Net controllers, by including actuators and sensors within the Petri Net controller. The challenge for the work presented was to develop an alternative method of building a model and running a simulation using tools commonly used in the industrial environment. The basic idea was to use safe coloured Petri Nets to control the PLC and as such prove the efficacy of the proposed method.

REFERENCES

1. Bender, D. F. et al 2008. *Ladder Metamodeling and PLC Program Validation through Time Petri Nets.* Model Driven Architecture-Foundations and Applications (ECMDA 2008), Berlin-Germany. pp. 121-136.
2. Bolton, W 2009. *Programmable Logic Controllers*, 5th edition. Newnes. ISBN-13: 978-0-7506-8112-4.
3. Bonet, B. et al 2008. *Directed Unfolding of Petri Nets*. Dortmund.
4. Casandras, C., Lafortune, S. 2007. *Introduction To Discrete Event Systems*. 2nd Edition. Springer.
5. Chirn, J.L & McFarlane D.C. 1999. *Petri nets based design of ladder logic diagram internal report*. Institute for Manufacturing: Cambridge University.
6. da Silva Simao, A., et al 2009. Generating *reduced tests for FSMs with extra states*. Lecture Notes in Computer Science 5826, 129–145.
7. David, R 1996. Grafcet: *A Powerful Tool For Specification of Logic Controllers*. IEEE Transaction on Control Sytems Technology Vol 3 Issue 3: pp. 253-268.
8. Fabre, E 2006. *On the construction of pullbacks for safe Petri nets*. In Proc. ICATPN '06, volume 4024 of Lecture Notes in Computer Science.
9. IEC 61131-3 2003. *Programmable controllers-part 3*: Programming languages 2nd Edition. International Electrotechnical Commision.
10. Jean-Marc Roussel et al 2011. *A Formal Semantics For GRAFCET Specifications*. 7th IEEE Conference on Automation, Science & Engineering (IEEE CASE, 2011), Trieste; Italy.
11. Jean-Marie Proth, 1997. Petri Nets: *A Tool for Design and Management of Manufacturing of Manufacturing Systems.* 1st Edition. Wiley.
12. Jensen, K 2009. *Coloured Petri Nets*. Aarhus University, Denmark.
13. Jorgensen, J.B. et al 2006. *From task descriptions via coloured Petri nets towards an implementation of a new electronic patient record*. In K. Jensen, editor, Proceedings of the 7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, volume PB-579 of Daimi Reports pages 137155, Aarhus, Denmark.
14. Jones, A. H.; Uzam, M 1996. *Design of Sequential Control Systems in Statement Lists Using TPLL.* Part I - Token Passing Statement List.
15. Parr, E. A. 1998. *Industrial Control Handbook. Second Edition*. Industrial Press INC 1999. ISBN 0-8311-3085-7.
16. Petri, C. A. 1962. *Kommunikation mit Automaten*. Ph. D. Thesis. University of Bonn.
17. Provost, J., Roussel, J.M., Faure, J. M. 2010. *SIC-testability of sequential logic controllers*. In: Proceedings of 10th International Workshop on Discrete Event Systems (WODES 2010). pp. 203–208.
18. Ray, C. A 1992. *Robots and Manufacturing Automation*, 2nd Edition. Wiley.
19. Rockwell Automation: MicroLogix1000 user manual: http://literature.rockwellautomation.com/idc/groups/literature/documents/in/1761-in001_-en-p.pdf.(Accessed 2nd August, 2014).
20. Stremersch, G. 2001. *Supervision of Petri Nets.* Kluwer Academic Publishers. Massachusetts.
21. Taholakian, A. and Hales, W.M.M., 1997*: A methodology for designing, simulating and coding PLC based control systems using Petri nets*, International Journal of Product Research, June 1997, Vol. 35, No. 6, pp. 1743-1762.
22. Tretmans, J., 2008. *Model based testing with labelled transition systems*. Lecture Notes in Computer Science 4949, pp. 1–38.
23. Uzam, M., Jones, A., Ajlouni, N. 1996. *"Conversion of Petri nets controllers for manufacturing systems into ladder logic diagrams"*, IEEE Symposium on Emerging Technology and Factory Automation, ETFA, 1996, Vol. 2, pp. 649-655.
24. Winskel, G. 2007. *Event structures with symmetry. Electronic Notes in Theoretical Computer Science*, pp. 172.
25. Zurawski, R. 2004. *The Industrial Information Technology Handbook* . CRC Press, Florida.

.