would subsequently be wrapped in a main module for a total representation of the ladder program. However, although such an approach would be considered modular it is also an unnecessarily bulky approach. It may have cost effects on time and the execution resources used as indicated in [22].

4.3. SMV Specification and Verification

Verification is performed by defining the main module which will consist of the specification(s) written in Computation Tree Logic (CTL) or temporal logic to be verified. The result of SMV verification is a message stating whether the CTL specification is true or false. If it is not true, a counter example is generated indicating a sequence of state transitions that leads to a violation of the CTL specification (see example in [16]). The CTL is a reachability tree for the finite state machine defined by the SMV model. CTL statements consist of a temporal logic operator along with a logical expression. The temporal logic operators are E, A, X, F, G and U where:

- E represents the existential path quantifier
- A represents the universal path quantifier
- X represents the next time
- F represents the future
- G represents globally
- U represents until

Therefore with an expression *q*, a CTL formula or specification could be written as *Fq* meaning that q holds some time in the future, *Xq* meaning that *q* holds for the next state and so on. If there is more than one SPEC declaration the specification is the conjunction of all the SPEC declarations.
Each of the formulas would be evaluated and the results reported separately in the order of the SPEC declaration in the program text. Considering the example of section 3, a specification can be written that ensures that each rung will be able to “open” or “close” the particular vent being monitored. Hence the derived SMV main module representation of the PLC program would be:

```
MODULEmain
V AR
in1 : boolean;
in2 : boolean;
in3 : boolean;
r1 : rung1(in1; in2; in3);
r2 : rung2(r1:c1; r1:c2; r1:c3);
r3 : rung3(r2:c1; r2:c2; r2:c3);
SPEC
AG(EF(r1:output)&EF!r1:output&
EF(r2:output)&EF!r2:output&
EF(r3:output)&EF!r3:output)
```

Variables in1, in2, and in3 are declared to be of type Boolean in this program but are not assigned values. This leaves the SMV system free values for this these variables, giving them the characteristics of being unconstrained inputs to the system. Instances r1, r2 and r3 represent rung 1, rung 2 and rung 3 respectively which monitors the three different vents. Inputs to rung2 are driven by the inputs to the instance of rung 1. Likewise inputs to rung3 are driven by the inputs to the instance of rung 2. The specification that we are verifying states that the behavior of the system is to allow the vents to be turned “on” and “off”.
The result of this SMV verification was true and was done with the Cadence FormalCheck SMV tool [8]. The resources used for this model are minimal: user time - 0.015625 s, system time - 0.03125 s, BDD [33] nodes allocated - 94, and data segment size - 0.
As presented in [16], it is important to note that model checking only checks the model of the system. For example, when SMV declares a claim as ’true’ or ’false’, this is with respect to the system model whether or not it accurately represents the system.

4.4. System Architecture

The major components of a system and the communication
Between these components identify its structural framework or its architectural design. The final system architecture is presented in Figure 6. The main implemented module is the “*PLCSelectionEngine*” and it contains other sub-modules. The actualization of the “PLCSelectionEngine”, formal verification of a ladder diagram, and the PLC database actualization were presented previously. The java code for the full implementation of the “*PLCSelectionEngine*” and the generation of the SMV given [9]. The next section presents a study of a test case used with the Resource Allocator and the results obtained.

5. Case Study

The aim of this case study is to demonstrate, that given an XML model of a control process, the Resource Allocator tool can be used to select the appropriate PLCs or EBCs. The user is then able to generate a report for each PLC returned in the results of the query to the database. This may

include additional modules, supporting materials, accessories, and diagrams from the database that are linked to the particular PLC or microcontroller. Likewise report listings of the contents of the database can be generated on demand. Figure 9 shows the snapshot of the prototype execution.

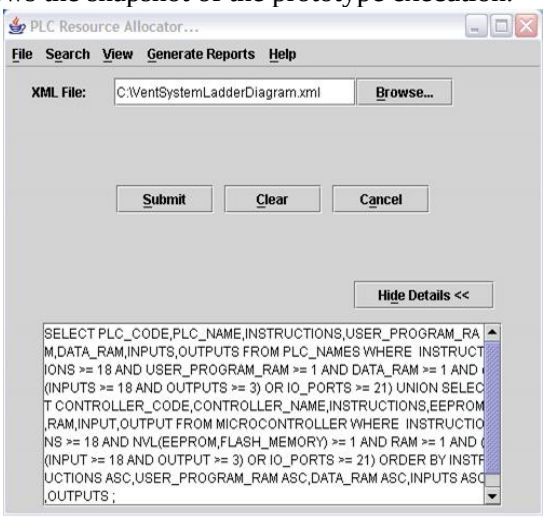

Fig. 9. Resource Allocator Prototype Execution

After submitting the XML file (i.e: VentSystemLadderDiagram.xml [9]), the Resource Allocator automatically generates, the list of PLCs/EBCs in a database that matches the generated parameters. In order to view the parameters that were used to generate the results, we use the View menu option and select the "Generated Parameters" sub-menu item as shown in Figure 10.

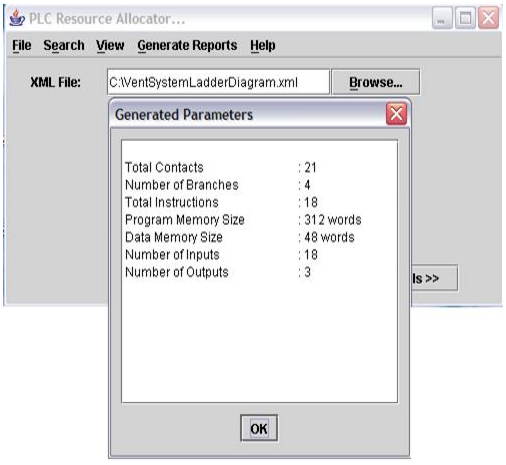

Fig. 10. Parameters generated from XML file

Likewise, to see the verifiable generated SMV code, the user should go to the View sub-menu and select the SMV Code menu item (Figure 11).

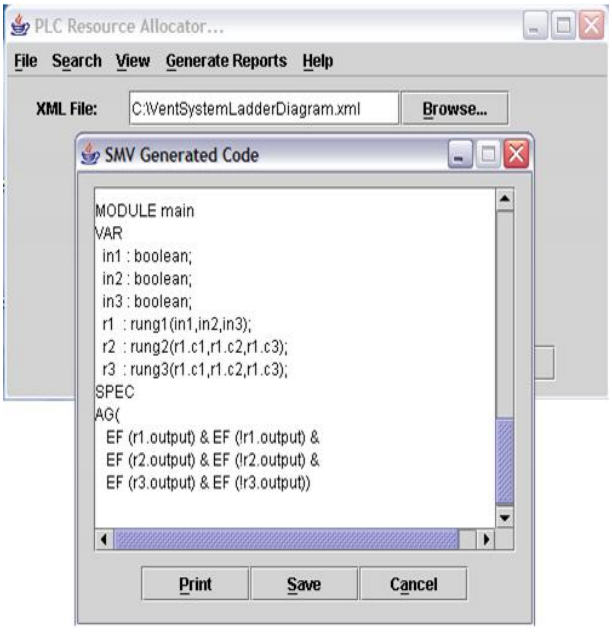

Fig. 11. SMV code generated from XML file

We will present only one case study. The remaining examples are described in [9].
Consider the example of a real-life control process, that consists of a painting system (Figure 12).

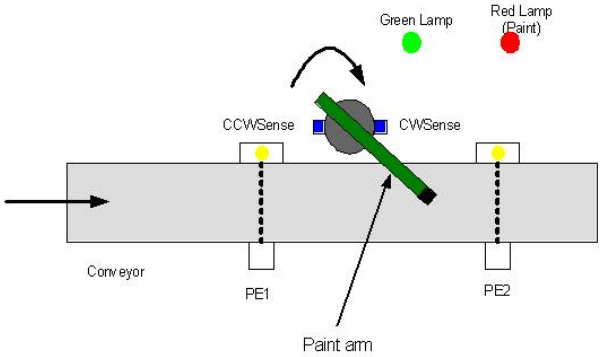

Fig. 12. Diagram of the painting system [11]

A conveyor system that retrieves parts as is needed for a robot to complete paint job. The robot sweeps over the part, before the part can move on. The sensor lamps must be on for the conveyor to work. All actuators and lamps should be off when the switch is off. When the "On" switch is turned, the conveyor should start. It should run until PE1 indicates the presence of a part at the paint station. At this point, the conveyor should automatically turn off. The paint arm, which is assumed to have started in its counter clockwise position, should be moved to the clockwise position (CW), and then back to counter clockwise (CCW) position. While the paint arm is moving, the paint should be spraying (represented by the Red lamp being on). After a complete spray operation, the Red lamp should be off. The green light should turn on and

stay on for two seconds (use of a timer), indicating the process is complete.
The conveyor should then turn on again. The system should then receive another part. Figure 13 gives a snapshot of the ladder diagram of a painting system. The system controller has a set of inputs and outputs (tables 2 and 3).

Table 1. Inputs required by painting controller

Inputs	**Description**
PE1	Photo Electric sensor signal that indicates the position of the part being painted (begin)
PE2	Photo Electric sensor signal that indicates the position of the part being painted (end)
CCWSense	Sensing position signal of the paint arm and its rotation counter clockwise
CWSense	Sensing position signal of the paint arm and its rotation counter clockwise
OnSwitch	Signal that starts the process
StopSwitch	Signal that stops the process
G_timer	Signal that activates the 2-second timer

Table 2. Outputs of the painting controller

Outputs	**Description**
CCWMotor	Control Signal used for switching on/off the motor that turns the paint arm counter clockwise
CWMotor	Control Signal used for switching on/off the motor that turns the paint arm counter clockwise
ConMotor	Control signal used for starting and stopping the conveyor
GLamp	Green lamp signal
RLamp	Red lamp signal
Spainter	Spray painter control signal

The representative XML representation is saved as a file named PaintingSystemLadderDiagram.xml [9].

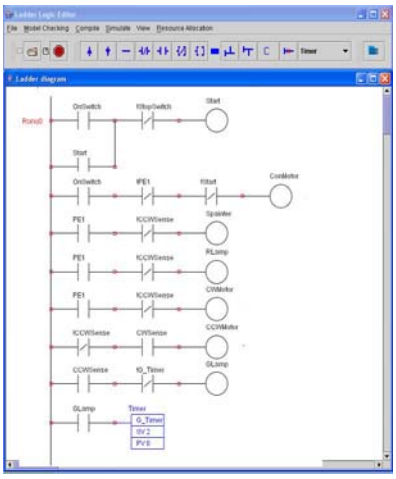

Fig. 13. Snap shot of the ladder diagram editor.

Using this file as input to the PLC/EBC Resource Allocator the following results were observed for the generated parameters, the selected PLCs/EBCs, the generated SMV model and verification. The generated parameters are given below:
Total Contacts : 25
Number of Branches : 1
Total Instructions : 20
Program Memory Size : 360 words
Data Memory Size : 104 words
Number of Inputs : 17
Number of Outputs : 8
Number of Timers : 1

The Allocator generates results that are similar to those of the vent control of figure 10. However it is important to recognize that the results returned are dependent on more than one factor. The collective points in the previous case are reiterated here; That is, the type and number of PLCs/MCUs returned depend on the population of the database [9]. Factors such as the variation in the values of the parameters in the database, a larger quantity of data and marked differences in the complexity of the control processes being studied cause greater variation in the results obtained. For example the *"Find dialog"* command on the Resource Allocator can return more specific PLCs/MCUs by directly specifying the size of the parameters needed. If a search for all MCU from Z-World Inc [32] is made, a sample report similar to the one of Figure 14 should be obtained.

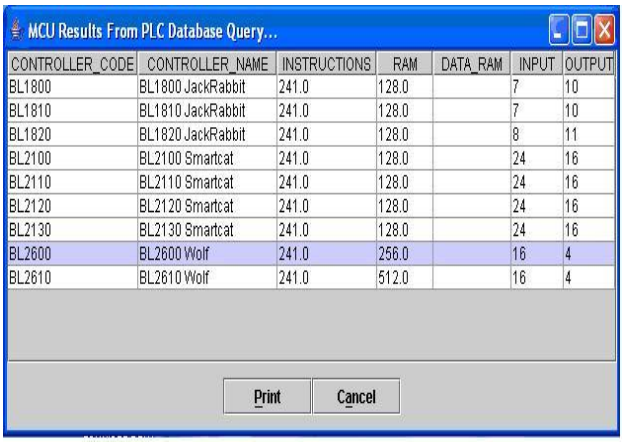

Fig. 14. Z-world MCUs Query

Now, with a larger quantity of data in the database, a variation in the data is shown. Using the "*Find dialog*" command again and specifying MCUs with at least 256 Kbytes of RAM leads to few elements. The variation of parameters are directly related to the complexity of the control process being studied. Hence, since the test cases showed similar complexities, the results from the database agreed likewise (Figure 15). The SMV Model of the Painting system obtained from the XML using our software tool is given in Appendix 2 and the verification results in Appendix 3.

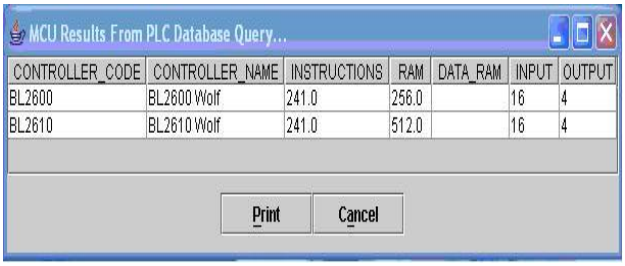

Fig. 15. Finding Results - Z-world MCUs with at least 256 K RAM

6. Conclusion and Future Direction

6.1. Conclusion

This paper presents a preliminary study that combines software methods for effective deployment of programmable logic controllers in control processes. The database is intended to be representative of all PLCs and microcontrollers. This is achieved by capturing the essential characteristics of these components through the analysis of data sheets from various industry practitioners. The intent is that the resultant database should be general enough to represent all the fundamental information necessary for any selection, as well as to provide auxiliary information on the components selected.

The prototype of the Resource Allocator tool has been designed with a few limitations with regard to the structure and size of the Ladder Diagram programs that can be handled. The rung depth has been restricted to 2 and the number of rungs to 20. Additionally, all programs are assumed to be sequential while in reality programs may have jumps or loops. Nevertheless the structural limitations were sufficient for the level of analysis needed for this research. However, the Resource Allocator tool can be expanded to improve these limitations and also to increase the number and type of contacts that are accommodated. This will increase the number of instructions or the size of the PLC program that can be processed.

The digraph-XML model presented requires further testing and analysis with more industrial PLC programs written in Ladder Diagram for added validation and verification of the model. Although the test case was

successfully verified, the model presented can be refined to be more semantically rich thus increasing the capabilities of the Resource Allocator.

The XML model can be represented as a set of edges and as a simplified incidence matrix which can increase the performance of the PLC selection engine. Such a model is purported to be more memory efficient and will increase the speed of parsing. Therefore it is expected to be a core part of the resource allocation system in future implementations. There are also inherent limitations in the generated SMV code. The problem of being able to generate ad hoc CTL specifications in the SPEC section of the SMV main module for each control process is still an unaccomplished task. It is no small feat to specify invariants or other properties such as fairness, safety and liveness that remain constant for all control processes.

The prospect of performing this task appears possible with the concept of a more semantically rich model. However, SMV code generated in this work models the system and model verification checks whether or not it accurately represents the system. This work has the potential to be very useful to practitioners in the PLC industry and is a precursory step in the total automation and formal verification of industrial control processes. It facilitates this process through the implementation of a reference database, a PLC selection engine and a SMV code generator for Ladder Diagram program verification. It also provides a number of perspectives for future research in the field.

6.2. Future Direction

Further The future direction of this work can be gleaned from the answer to the following question: What is expected in the future of resource allocation and model verification? It is anticipated that more efficient methods for automatic selection and verification will be produced based on enhanced or novel models. This should result in the improved performance of the selection engine. It can be noted that even the simple creation of strategic indexes (indices) in the PLC database can significantly improve the database's performance as the size of the database is continually increased. However, the focus was on the implementation and functionality of the model, so no indices were created on the reference database.

It should be the case that verification is done automatically before the selection engine is called to perform automatic resource allocation. That is, getting a result from the Resource Allocator tool should be dependent on the ladder logic being correct to the users' specification. Hence we forecast a tighter integration or a convergence of the Resource Allocator and the formal verification tools to produce more complete automation process. This remains a task for a later version of the system. It has been the nature of formal verification to create an intermediate model of the system prior to translating it into the formal language. For example, [10] used state chart, Thomas and Bryla [27] used transition systems diagrams and for our study digraphs and XML.

To use our model as the basis for formal verification would require an equivalent digraph representation for the ladder logic programs written in any of the five different PLC programming languages. Essentially, this would allow programs written in other languages to be represented in ladder diagram according to our specifications. A tool that represents ladder diagrams internally as digraphs and that generates the corresponding XML model from the ladder diagram programs is created in [4, 18] The generated XML model is then used to generate the formal model in SMV for verification. Alternatively, a common XML model could be found for the programming languages represented in the International Electro-technical Commission (IEC) standard, IEC 61131- 3, which can be used for the basis of all PLC verification. Achieving any of these could be the precursor for the standardization of formal verification of ladder diagrams using SMV.

Appendixes

Appendix 1: Sample XML Model

```
</LadderDiagram>
...
- <Graph graphNumber=”3”>
- <Vertices>
- <Vertex number=”0”>
<startX>60.0</startX>
<startY>470.0</startY>
</Vertex>
- <Vertex number=”24”>
<startX>180.0</startX>
<startY>470.0</startY>
</Vertex>
= <Vertex number=”25”>
<startX>300.0</startX>
<startY>470.0</startY>
</Vertex>
- <Vertex number=”26”>
<startX>420.0</startX>
<startY>470.0</startY>
</Vertex>
-<Vertex number=”27”>
<startX>540.0</startX>
<startY>470.0</startY>
</Vertex>
</Vertices>
- <Edges>
-<Edge type=”CloseContactEdge”>
<from>0</from>
<to>24</to>
<address>000.02</address>
<symbol>V1</symbol>
</Edge>
- <Edge type=”CloseContactEdge”>
Exploring an Approach for Effective Deployment of Programmable Logic Controllers (PLCs) 11
<from>24</from>
<to>25</to>
<address>000.01</address>
<symbol>V2</symbol>
</Edge>
- <Edge type=”CloseContactEdge”>
<from>25</from>
<to>26</to>
<address>000.03</address>
<symbol>V3</symbol>
</Edge>
- <Edge type=”OpenOutputEdge”>
<from>26</from>
<to>27</to>
<address>003.00</address>
<symbol>NO VENT</symbol>
</Edge>
```

</Edges>
</Graph>
</LadderDiagram>

Appendix 2: Generated SMV from the XML model of the Painting System Controller

```
MODULE rung1(ONSWITCH,OFFSWITCH,START)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (ONSWITCH — START) &
(!OFFSWITCH);
DEFINE
c1 := ONSWITCH;
c2 := OFFSWITCH;
c3 := START;

MODULE rung2(ONSWITCH,PE1,START)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (ONSWITCH & !PE1 & !START);
DEFINE c1 := ONSWITCH;
c2 := PE1;
c3 := START;

MODULE rung3(PE1,CCWSENSE)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (PE1 & !CCWSENSE);
DEFINE
c1 := PE1;
c2 := CCWSENSE;

MODULE rung4(PE1,CCWSENSE)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (PE1 & !CCWSENSE);
DEFINE
c1 := PE1;
c2 := CCWSENSE;

MODULE rung5(PE1,CCWSENSE)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (PE1 & !CCWSENSE);
DEFINE c1 := PE1;
c2 := CCWSENSE;

MODULE rung6(CCWSENSE,CWSENSE)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (!CCWSENSE & CWSENSE);
DEFINE
c1 := CCWSENSE;
c2 := CWSENSE;

MODULE rung7(CCWSENSE,GTIMER)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (CCWSENSE & !GTIMER);
DEFINE
c1 := CCWSENSE;
c2 := GTIMER;

MODULE rung8(GLAMP)
VAR
output : boolean;
ASSIGN
init(output) := 0;
next(output) := (GLAMP);
DEFINE
c1 := GLAMP;

MODULE main
VAR
in1 : boolean;
in2 : boolean;
in3 : boolean;
in4 : boolean;
in5 : boolean;
in6 : boolean;
r1 : rung1(in1,in2,r1.output);
r2 : rung2(r1.c1,in3,r1.c3);
r3 : rung3(r2.c2,in4);
r4 : rung4(r2.c2,r3.c2);
r5 : rung5(r2.c2,r3.c2);
r6 : rung6(r3.c2,in5);
r7 : rung7(r3.c2,in6);
r8 : rung8(r7.output);
SPEC
AG(
EF (r1.output) & EF (!r1.output) &
EF (r2.output) & EF (!r2.output) &
EF (r3.output) & EF (!r3.output) &
```

EF (r4.output) & EF (!r4.output) &
EF (r5.output) & EF (!r5.output) &
EF (r6.output) & EF (!r6.output) &
EF (r7.output) & EF (!r7.output) &
EF (r8.output) & EF (!r8.output))

Appendix 3: Summary of the Verification Results for
Painting System

Model checking results
==========
(AG ((((((((((((((EF r1.output)&(EF (˜r1.output)))&(EF
r2.output))&(.....true
user time.....0.046875 s
system time.....0.03125 s

Resources used
==========
user time.....0.046875 s
system time.....0.03125 s
BDD nodes allocated.....415
data segment size.....0

References

[1] Aiken, M. Fahndrich, and Zhendong Su. Detecting Races in Relay Ladder Logic Programs. In Proc. 4th Int. Conf. Tools and Algorithms for Construction and Analysis of Systems (TACAS'98), Lisbon, Portugal, March 1998, Volume 1384, Lecture Notes in Computer Science, pp. 184-200, Springer, 1998.
[2] Andrews D. 1996, Formal Methods in Software Engineering Education: Discussion Summary, Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SL: E&P '96), pp. 514-515.
[3] Automatic Direct 2006, Using the Considerations for a PLC Worksheet, http://www.automaticdirect.com.
[4] Buchanan L. 2006, Retargetable Ladder Logic Diagrams Tool, MPhil. Thesis, University of the West Indies, Jamica.
[5] Date C. 2000, An Introduction to Database Systems, Addison- Wesley.
[6] Emerson, E. A. 1990, Temopral and Modal Logic, In J. Van Leeuwen, Editor, Handbook of Theoretical Computer Science, Vol. B, Chapter 16, pp. 995 - 1072. Elseiver Science.
[7] Frey G. and Litz G., Formal Methods in PLC Programming. Proceeedings of the IEEE Conference on Systems Man and Cybernetics SMC 2000, Nashville, pp. 2431 – 2435.
[8] Formal 2006, Cadence Formal Check, http://www.cadence.com/webforms/cblsoftware/index.aspx
[9] Gordon A. 2006, Automaic Resource Allocation and Model Verification in Programmable Logic Controllers, MSc. Computer Science Thesis, University of the West Indies, Jamaica.
[10] Grama, R., Srinivasan, G. R. and Gluch, D. P., 1998, A Study of Practice Issues in Model-Based Verification Using the Symbolic Model Verifier (SMV), Internal CMU/SEI-98-TR-013/ESC-TR-98-013 available at http://www.sei.cmu.edu/pub/documents/98.reports/pdf/98tr013.pdf.
[11] Holloway, 2005 http://www.engr.uky.edu/holloway/MFS605.
[12] ICE (International Electrotechnical Commission), 1993, IEC Standard 61131-3: Programmable Controllers, Part 3.
[13] A. Mader, What is the method in applying formal methods to PLC Applications, 4th Int. Confutomation of Mixed Processes: Hybrid Dynamic Systems (ADPM), S. Engel, S. Kowaleski, and J. Zaytoon (eds), Shaker Verlag, Aachen, Germany, 2000, pp. 165 - 171.
[14] McMillan, K. 1993, Symbolic Model Checking, Kluwer Academic.
[15] McMillan K., Symbolic Model Checking, PhD Thesis, available at http://www.kenmcmil.com/pubs/thesis.pdf.
[16] McMillan, K. 2006, SMV Tutorial, http://www.kenmcmil.com/tutorial.ps.
[17] Moon, I. 1993,Modeling Programmable Logic Controlers for Logic Verification, IEEE Control Systems, 14(2): 53-59, 1993.
[18] Ngalamou L., L. Buchanan, and L. Myers, August 4-6, 2004, Architecture of a Retargetable Ladder Logic Diagrams Tool, in the Proceedings of SICE Annual Conference in Sapporo, pp. 215 - 2519.
[19] Ngalamou L. and L. Myers, Modelling PLC Charcateristics for Resource Allocation, International Journal of Computer Applications in Technology, Inderscience, Vol. 31, Nos. 3/4, 2008, pp. 263 – 274.
[20] Noergaard, T., 2005, Embedded Systems Architecture – A Comprehensive Guide for Engineers and Programmers, Newnes - Elseiver.
[21] Oracle 2006, http://www.oracle.com/index.html.
[22] Rausch, M. and Krogh, B. H. June 1998, Formal Verification of PLC Programs, American Control Conference, Philadelphia, PA, USA.
[23] Rossi, O. and Schnoebelen, P. Sept.2000, Formal modeling of timed function blocks for the

automatic verification of ladder diagram programs, 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems, ADPM'2000, Dortmund (Germany), pp. 177-182.
[24] Sax 2010, http://www.saxproject.org/.
[25] Silberschatz, A. and Korth H. F., 1997, DatabaseSystem Concepts, Third Edition, McGraw Hill.
[26] Smet, Cuffin R., R O., Canet G., J.-J. Lesage J., S chnoebelen P., and Papini H. , October 2000, Safe Programming of PLC using Formal Verification Methods, 4th International PLCopen conference on Industrial Control Programming, ICP'2000, Utrecht (The Netherlands), pp. 73-78.
[27] Thomas B. and Bryla B. 2002, OCA/OCP: Oracle9i DBA Fundamentals I Study Guide, Sybex.