Formal Verification of Intelligent Mechatronic Systems with Decentralized Control Logic

Sandeep Patil
The University of Auckland,
Auckland, New Zealand.
spat251@aucklanduni.ac.nz

Valeriy Vyatkin
The University of Auckland,
Auckland, New Zealand.
v.vyatkin@auckland.ac.nz

Majid Sorouri
The University of Auckland,
Auckland, New Zealand.
msor021@aucklanduni.ac.nz

***Abstract* — This paper introduces an approach to automatic verification of mechatronic systems designed as plug-and-play of Intelligent Mechatronic Components (IMC). The control logic of the system is composed from autonomous controllers of the IMCs and is automatically verified using model-checking. Net Condition Event Systems formalism (a modular extension of Petri net) is used to model the decentralized control logic and discrete-state dynamics of the plant. A re-configurable pick and place robot is used as an illustrative example. At first a three cylinder pick and place robot is used to design our new master-slave architecture for controller design and then the NCES models are re-used without much modification in a new 6 cylinder pick and place robot. The control model is then subjected to model checking using the ViVe/SESA model checker. A multi closed loop model of Plant and Controller is used and controller is extensively verified for safety, liveliness and functional properties of the robot. Computational Tree Logic (CTL) is used to specify these properties.**

Keywords — NCES, ViVe, SESA, Formal Verification, Closed-Loop Modeling.

I. INTRODUCTION

There has been substantial amount of growth in industrial automation systems industry in the last decade with the growing trend towards de-centralization that brings agility, scalability, re-configurability and fault tolerance as compared with the centralized systems [1]. This change has drastically shifted the traditional centralized control design approach to the modular reconfigurable engineering architecture. This has resulted in need for design of distributed control systems. International Electrotechnical Commission (IEC) has come with IEC 61499 [2] standard for design of such distributed control systems [3-5]. The distributed systems should be modular, reusable, flexible, extendible and reconfigurable. The idea of using one model for each of the mechatronic components of an automation system has been explored in detail over the last few years [6, 7]. This results in these models being re-used in another automation system where the same or similar mechatronic components were used.

With the increase in agility requirements there is a need for better testing and verification frameworks of these distributed systems. Simulation of these systems is one widely used approach to testing. In fact IEC 61499 automation tools such as NxtSTUDIO®1.5 [8], ISaGRAF [9] and Function Block Development Kit (FBDK) [10] support visual simulation for the reason of testing. Simulation alone does not guarantee 100% validation of the automation systems. To address this problem, formal verification [11] has been adopted to formally model and verify the target systems. Discrete state model-checking [12] is one such formal verification approaches.

Model checking provides an unsupervised automatic verification process which identifies the model's design pitfalls via counterexamples. The first step of model checking is the formal modeling of the target system in certain formalism. The Net Condition/Event Systems (NCES) [13] is one such formalism designed for modeling distributed control systems. After modeling, the NCES models can be formally verified against the system properties specified in computational tree logic (CTL) by the model checking tools such Vive and SESA [14]. Modeling and verification using NCES modules has been studied before and applied in modeling of IEC 61499 function blocks [15-19]. NCES modules are interconnected by event and condition arcs to form bigger modules just like the event and data connections in IEC 61499 function blocks. Event propagation is modelled directly by event arcs and the runtime scheduling is assumed to be concurrent and instantaneous.

The rest of the paper is organized as follows; Section II presents in brief about using closed loop modeling of the automation systems and present the case study example. Section III presents the proposed master-slave controller design with a visualization model implemented in NxtSTUDIO®1.5 [8] (IEC 61499 function blocks), Section IV presents a brief about NCES and the master-slave controller in NCES, Section V presents the re-usability and scalability features of our new controller design, Section VI presents the CTL verification results and will end with conclusions and future works.

II. MULTI CLOSED-LOOP MODELING

It is often seen that in some verification frameworks e.g. [20, 21] controller is verified as a standalone component, even though such verification has limited capabilities, e.g. it cannot verify liveliness of the system. Modelling distributed system as a multi closed-loop model that also incorporates a model of the plant can achieve a lot more powerful verification framework, because it is the manufacturing plant that specifies the safety constraints and the desired production processes [22-25]. Multi closed-loop modeling allows for thorough verification of the control logic, and reduces the complexity of model checking as compared to only controller verification under an arbitrary inputs assumption. It also allows checking of specifications formulated in terms of the plant variables rather than in

terms of controller inputs/outputs. This paper therefore, describes the modeling of such multi closed-loop behavior as well as presents new controller design architecture.

A. Case Study Example

To demonstrate benefits of our approach, we will use a pick and place object shown in Fig 1(a), which is composed of several mechatronic units as follows:

- There are two horizontal cylinders and a vertical cylinder that extract and retract. The left horizontal cylinder is half the size of the right cylinder. The vertical cylinder picks up the work pieces using the suction unit attached to its end.
- Both horizontal cylinders have two control signals (CGO: Cylinder Go Out: extending, CGI: Cylinder Go In: Retracting). The vertical cylinder has only one control signal (VCGD: Vertical Cylinder Goes Done). When this signal is not active, the cylinder moves up (pulled by the internal spring).
- Each of the cylinders has their own sensors that indicate the cylinder's home and end positions. There are also sensors in each of the three input trays (pp1, pp2 and pp3) and one in the slider (pp0) to indicate the presence of a work piece. The suction unit has a built-in sensor, vacuum indicating that a work piece is sucked.
- Fig 1(b) shows the desired behaviour, it specifies the state of each cylinder and the vacuum unit when picking and dropping each of the work pieces.

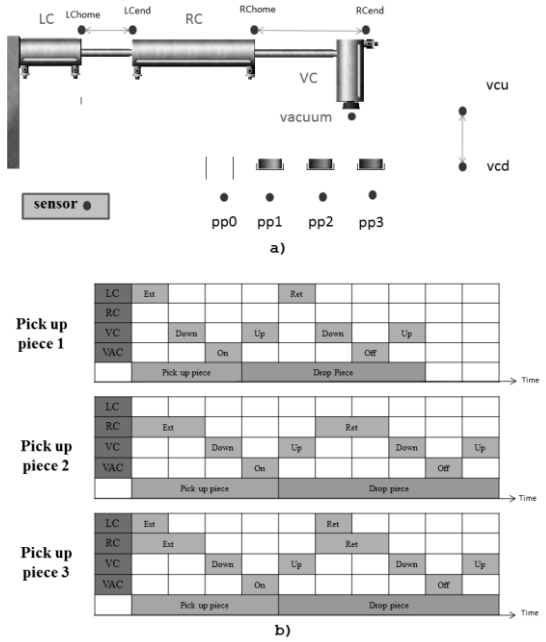

Fig 1: (a) Reference object: pick and place robot. (b) Activity diagram of the pick and place robot

III. MASTER-SLAVE CONTROL ARCHITECURE

Master-Slave is one possible distributed control architecture out of several options considered in [26, 27]. As the name suggests one controller (master) will take control of one or more controllers (slaves) by sending control commands (e.g. wp Manager depicted in Fig 2 and Fig 3 is a master). The slave function blocks solely acts based on the commands being sent to it by the master. Apart from master and slave function blocks, there is master-slave function block that acts a slave to one master function block and master to a slave function block [28]. This is the case for the vertical cylinder, as it acts as slave to work piece manager and master to vacuum unit (Fig 3 and Fig 5).

This example (Fig 1) illustrates the design scenario, where the designer creates different configurations of mechatronic systems from available mechatronic components. This accelerates the development process. Obviously, the process of software design of the robot needs to be accelerated as well and the proposed modular master-slave control architecture promises such acceleration: the controller for each new mechatronic configuration can be assembled by retrieving the corresponding function blocks and connecting them as illustrated in Fig 2.

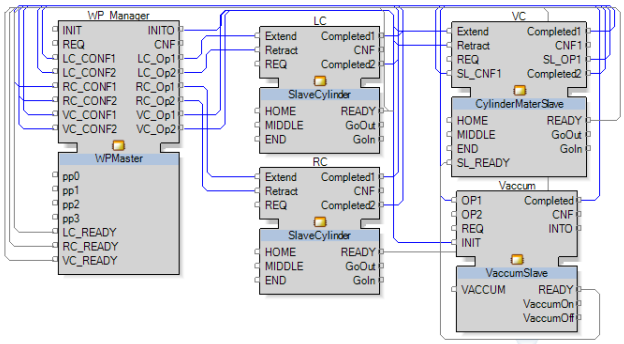

Fig 2: Distributed controller of the robot with Master –Slave architecture implemented in IEC 61499.

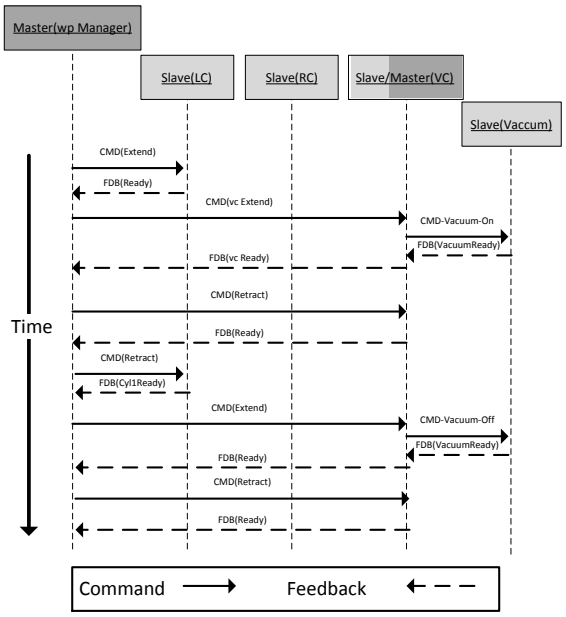

Fig 3: An example of event exchange among master-slave controllers to pick up a work piece from tray one

The master-slave controller uses hand shaking, i.e. whenever master issues a command to the slave, it cannot issue one more command to the same slave until it has received a feedback from the slave saying it has completed

its previous task as indicated by the command and feedback arrows in Fig 3.

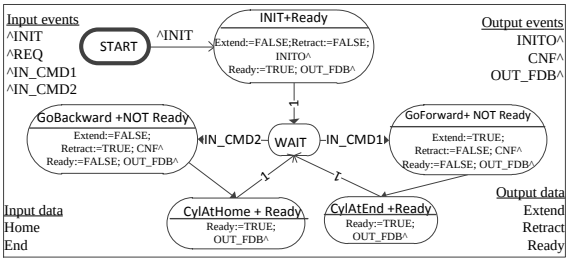

Fig 4: Controller of the horizontal cylinders being implemented as slaves to work piece manager

Fig 4 shows a slave controller, here IN_CMD1 and IN_CMD2 identify the two commands that slave can handle, which is nothing but commands to either extend the cylinder or retract the cylinder. OUT_FDB is the feedback signal to the master. Fig 5 shows the vertical cylinder controller that acts as master and slave, the extra input event IN_FDB is the feedback from the slave and extra output event OUT_CMD is the command to the slave.

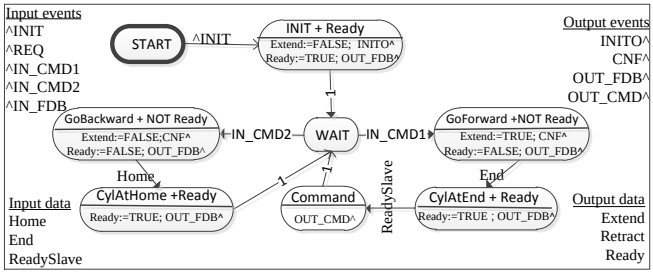

Fig 5: Controller of the vertical cylinder implemented as Master for vacuum unit and slave for work piece manager

IV. NET CONDITION-EVENT SYSTEMS

Net Condition/Event Systems (NCES) [13, 23, 29] can be viewed as a modular extension to Petri Nets [30]. The general idea of NCES is modelling a system as a set of modules with a particular dynamic behaviour and their interconnection via signals. An illustrative example of the graphical notation of a module is provided in Fig 6(a).

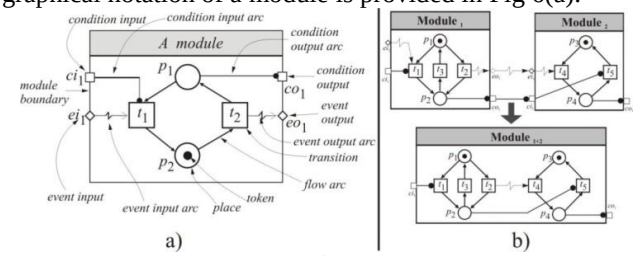

Fig 6: (a) Graphical notation of an NCES module; (b) Modular Composition of NCES modules results in a "flat" NCES.

Once designed, the modules can be re-used over and over. Each module has inputs and outputs of two types:

1) Condition inputs/outputs carrying information on marking of places in other modules.
2) Event inputs/outputs carrying information on firing transitions in other modules.

Condition and event inputs can be connected with some transitions inside the module by condition and event arcs. Places of the module can be connected to the condition outputs by condition arcs, and transitions can be connected to the event outputs by event arcs. This concept provides a basis for a compositional approach to building larger models from smaller components. The "composition" is performed by "gluing" inputs of one module with outputs of another module as shown in Fig 6(b).

A. *Why NCES?Benefits of NCES over other formalism*

There are couple of reasons to prefer place-transition formalisms to many others formalisms, e.g. finite automata. The first is their non-interleaving semantics (i.e. possibility of firing several transitions simultaneously), which better fits to modelling of distributed processes and of their interaction. It results in more compact reachability space, explained as follows.

Modelling of complex distributed systems with automata usually ends up in many concurrent automata models communicating via common variables, as illustrated in Fig 7 (a), where two state machines A and B are combined under "asynchronous parallel operator". Thus, the overall system model is a cross-product of the component automata, and to do model analysis it is necessary to build the cross-product consisting in this case of 9 states, as one sees in Fig 7 (a). Alternatively, in NCES a state of a model is determined by the marking of model places, so any global state of a distributed system is just one state of the model. This is shown in Fig 7 (b), where the same model is implemented in NCES with places (p1-p6) corresponding to states of the automata A or B (in the obvious manner). In the given initial state the reachability space of the model consists of only 4 states. The same behaviour obviously will be shown by the automata model in Fig 7 (a) (the outlined path A1B2→ A1B2→ A2B2→ A2B3→ A3B1→ A3B2), but to get it the whole cross-product automata needs to be built.

The other benefit of NCES is the intuitive modelling approach. Consider an NCES equivalent pick and place model conceptually presented in Fig 8. The NCES model of a moving object, such as a cylinder, is composed in a modular way following the pattern proposed in [29]. The NCES modules are connected with explicit event and data flow as compared to sharing variables in state machines. As noted in state machines model, two sequencers were used to order evolution of the automata. This is a considerable restriction of real life evolutions. In NCES, the controller output is directly connected to the plant input and vice versa, and the models make evolutions asynchronously.

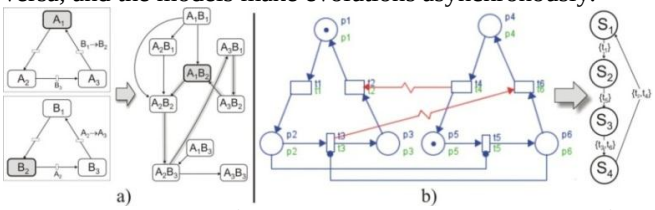

Fig 7: (a) Modelling of two communicating processes by means of concurrent state machines and their cross-product automaton; (b) The same model in NCES and its reachability graph

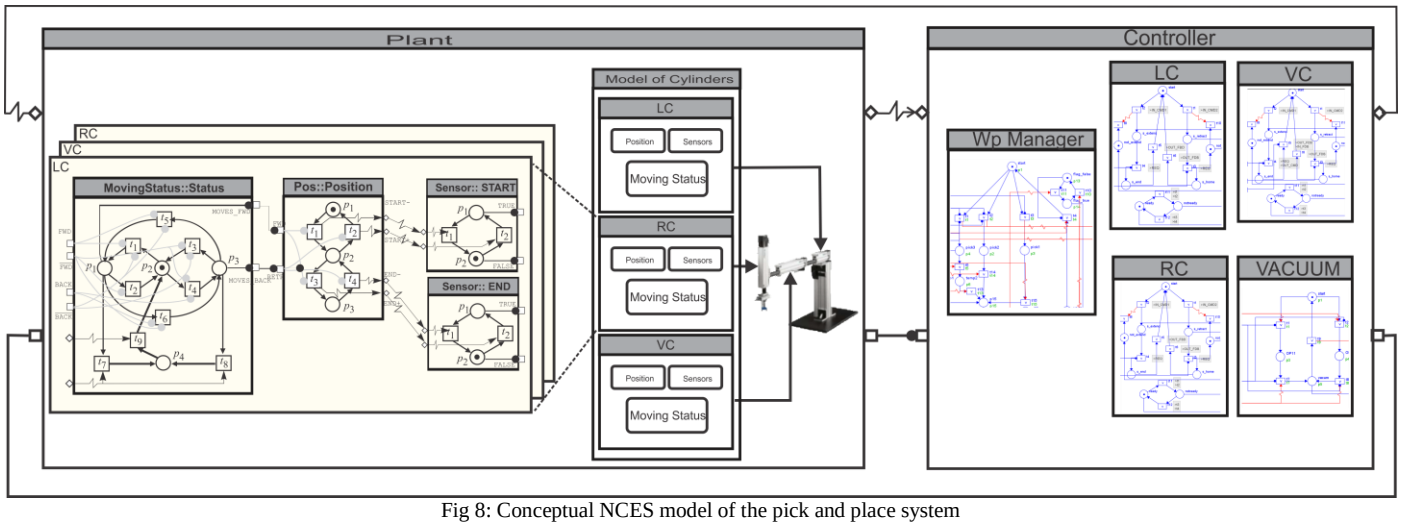

Fig 8: Conceptual NCES model of the pick and place system

B. *Master-Slave model in NCES*

Fig 9 shows NCES implementation of the slave cylinder control logic from Fig 4. Some explicit event connections are not shown in the illustrations to avoid confusing crossover connections; instead the grey boxes next to the net elements specify what they are connected to. Fig 10 shows the NCES model of the master-slave cylinder controller described in Fig 5.

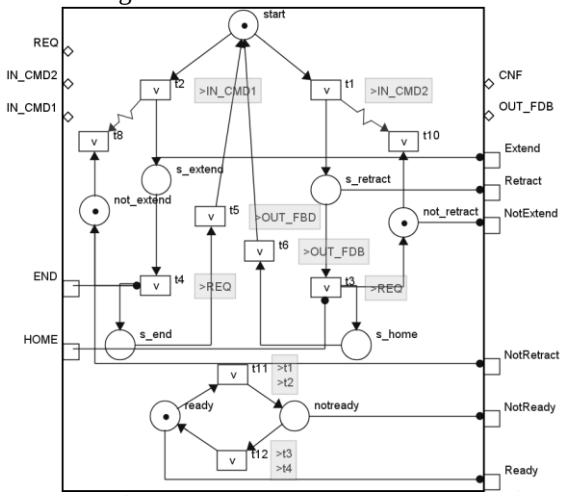

Fig 9: Controller of the horizontal cylinders being implemented as slaves to work piece manager in NCES

The start state in a NCES model is determined by the initial marking of the token, in Fig 9 the initial marking is in "start" place, from this place, on receiving either IN_CMD1 (extend) or IN_CMD2 (retract) event from the master (work piece manager) the token moves from start to either "s_extend" or s_retract" state, once the plant tells the controller that the cylinder is in either of the end positions (home or end), transition "t3" or "t4" is fired accordingly and the token returns to the start state with either "t5" or "t6" firing and emitting OUT_FDB to the work piece manager and thus completing the command-feedback handshake. The only difference between slave and master-slave is transitions "t4" and "t5", once the vertical cylinder reaches its end position, it commands its slave by sending the OUT_CMD event and waits for the IN_FDB event from the slave, once received "t5" is fired and token returns to the start position. Fig 11 shows the NCES model of the entire controller of the robot composed in a modular way similar to the FBDK model in Fig 2.

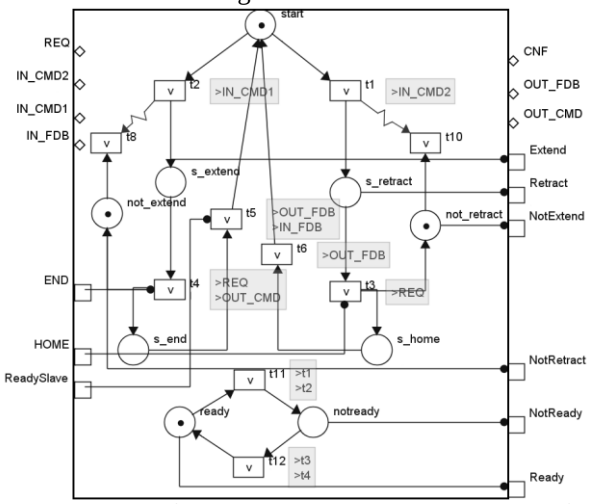

Fig 10: Controller of the vertical cylinder implemented as Master for vacuum unit and slave for work piece manager in NCES

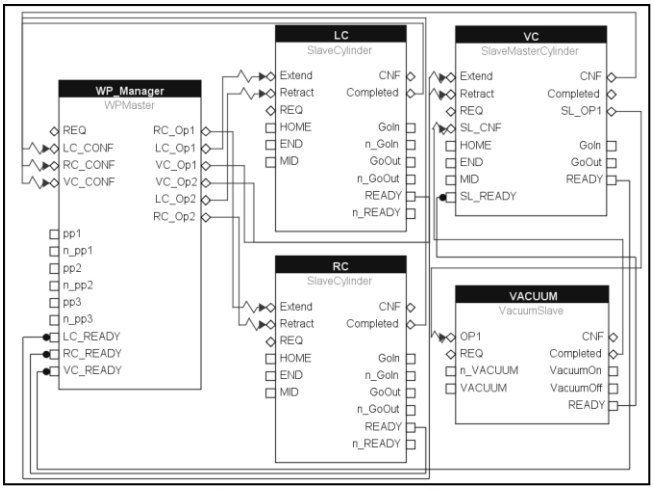

Fig 11: NCES model of distributed controller of the robot implemented with Master –Slave architecture

V. REUSABILITY AND SCALABITY OF MASTER-SLAVE ARCHITECTURE

In order to test the scalability and reusability of the proposed master-slave control pattern, we considered several configurations of the pick and place robot composed from various numbers of mechatronic modules. For example, the configuration in Fig 12 is composed from six cylinders. Also, more freedom was allowed in the location of input trays of work pieces at different levels. Unlike the reconfiguration method described in [31] which is for pre-determined different run time configurations, the method here can be used for any type and any number of re-configurations.

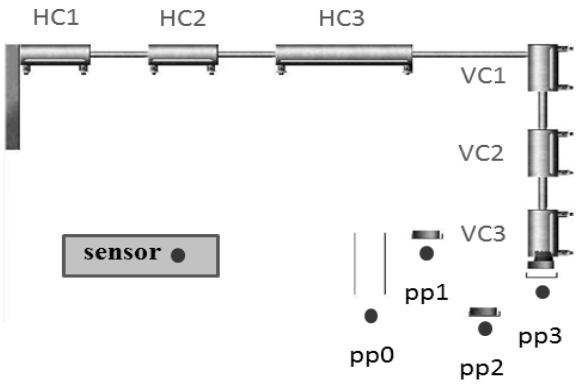

Fig 12: Re-Configured pick and place robot with 6 cylinders

The safety and functional requirements for all those configurations are same as of the 3 cylinder robot. Safety requirements include requirements such as "horizontal and vertical cylinders do not move at the same time". The main functionality is to pick work pieces from the input trays (pp1, pp2 and pp3) and drop them in the output slider pp0.

Formal verification of the NCES model corresponding to a particular mechatronic (and control) configuration is seen as enabler of agile modular mechatronic and software design.

The NCES model for this consisted of 5 slave control models (all three horizontal cylinders and top two vertical cylinders) re-used as is from the 3 cylinder model and 1 master-slave control for the bottom vertical cylinder that has the vacuum unit attached to it which again is just a slave, just like in the 3 cylinder robot. The only difference between the two models is the master control module, the work piece manager. The two work piece mangers are shown in Fig 13. All the other modules are re-used and only this master control model needs to be modelled, which will be automated as well. Auto generation of this model is possible: given a matrix of the pick and place robot as shown in Fig 1(b), the work piece manager can be automatically generated.

The main design flow of the work piece manager is shown in Fig 14. Here, depending on the presence of work pieces, the token follows one of the three (dashed) branches of the model and is returned to start state at the end. The priority considered in the two robot design in this paper is:

- If work piece 1 (pp1) is present, then pp1 is picked. This is achieved in transition 't1' in Fig 14 which is just connected to true condition input of pp1.
- If work piece 2 (pp2) is present and pp1 absent, pp2 is picked. This is achieved in transition 't2' in Fig 14 which is connected to true condition input of pp2 and false condition input of pp1 (n_xxx notation identifies the false condition inputs).
- If work piece 3 (pp3) is present and both pp1 and pp2 are absent, pp3 is picked. This is achieved in transition 't2' in Fig 14 which is connected to true condition input of pp3 and false condition input of pp1 and pp2.

Depending on the input matrix as given in Fig 1(b), the rest of the flow will just be sending command events to slave cylinder and receiving feedback from them.

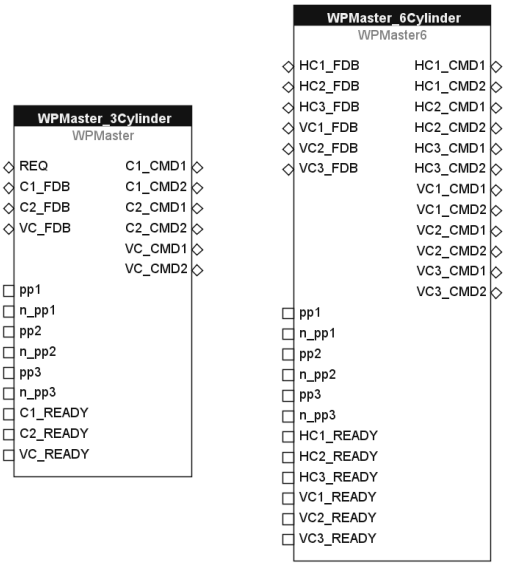

Fig 13: Work piece manager (master) control blocks of 3 cylinder and 6 cylinder robot

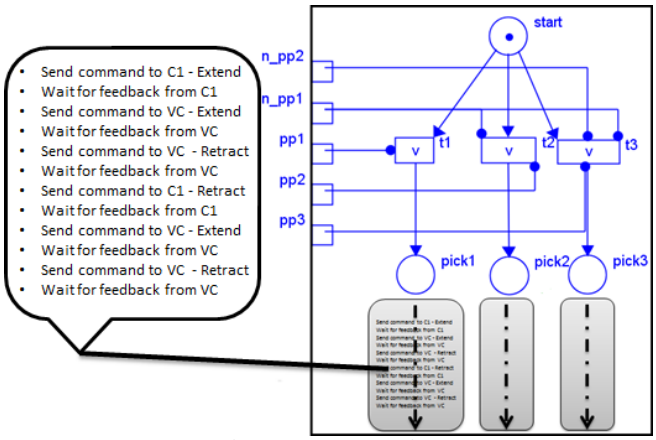

Fig 14: Flow logic for auto generation of work piece manager

A. *Using Meta-Model to generate the work piece manager*

Fig 15 shows the meta-model used to generate the master controller, i.e the work piece manager. It only shows the part that is described in Fig 1(b).

```
<Controller type = "Master-Slave" NumberOfWP = "3" NumberOfHC = "2" NumberOfVC = "1" VacAttach = "VC1">
    <wp1>                          <wp2>                          <wp3>
        <Action value = "pick" >       <Action value = "pick" >       <Action value = "pick" >
            <C1_POS value="Home" />        <C1_POS value="End" />         <C1_POS value="End" />
            <C2_POS value="End" />         <C2_POS value="Home" />        <C2_POS value="End" />
            <VC_POS value="Down" />        <VC_POS value="Down" />        <VC_POS value="Down" />
            <Vac    value="On" />          <Vac    value="On" />          <Vac    value="On" />
        </Action>                      </Action>                      </Action>
        <Action value = "drop" >       <Action value = "drop" >       <Action value = "drop" >
            <C1_POS value="Home" />        <C1_POS value="Home" />        <C1_POS value="Home" />
            <C2_POS value="Home" />        <C2_POS value="Home" />        <C2_POS value="Home" />
            <VC_POS value="Down" />        <VC_POS value="Down" />        <VC_POS value="Down" />
            <Vac    value="Off" />         <Vac    value="Off" />         <Vac    value="Off" />
        </Action>                      </Action>                      </Action>
    </wp1>                         </wp2>                         </wp3>
</Controller>
```

Fig 15: Meta-Model for the matrix in Fig 1(b)

Meta-Model has the following details. Composition of the system (<Controller>) such as type of Controller architecture (attribute "type"), number of work pieces (attribute "NumberOfWP), number of horizontal cylinders (attribute "NumberOfHC), number of vertical cylinders (attribute "NumberOfVC) and to which cylinder is the vacuum unit connected (which module will act as master as well as a slave model i.e attribute "VacAttach). For each work piece (<wp1>, <wp2>, <wp3>), the meta-model specifies position/value of the related sensors for both pick and drop action. Note that the details of each work piece appear column wise in Fig 15, but in actual it is a text file and the details occur one after the other. It is shown this way to save up space.

VI. Model Checking with Vive/SESA

Both the 3 cylinder and 6 cylinder models were model checked using the ViVe and SESA tools. CTL was used to represent safety, liveliness and other functional requirements. The advantage with these tools is the ease with properties can be mentioned. The properties are presented in terms of the places in the NCES models.

The ViVe tool flattens the whole model consisting of different sub modules into one (possibly huge) NCES model as described in section IV Fig 6 (b), the ViVe tools tree view of the flat model is shown in Fig 16. The flattened model can be exposed then to SESA model checker.

For example, to check for the property that says "Opposite actuator signals (extend and retract) to the cylinders (C1, C2 in case of 3 cylinder model) should never be emitted at the same time", we write the property as "AG (NOT(p136ANDp137))", where places p136 and p137 correspond to global place number in the flattened NCES model that actually correspond to the places "s_extend" and "s_retract" in Fig 9.

In NCES terminology, the tool checks if at all there is a possibility that a token can be present in both these places at any given time. Table 1 below summarizes all the properties that were checked for our two reconfigurable models. It is even easier to check for the liveliness property of the cylinder, simply check if all the places in the cylinder model if they become false (have no token) in future once they were true (had token). The format of the CTL property will be AG (pXX -> EF (NOT (pXX))), Where "XX" corresponds to every place of the flattened controller model.

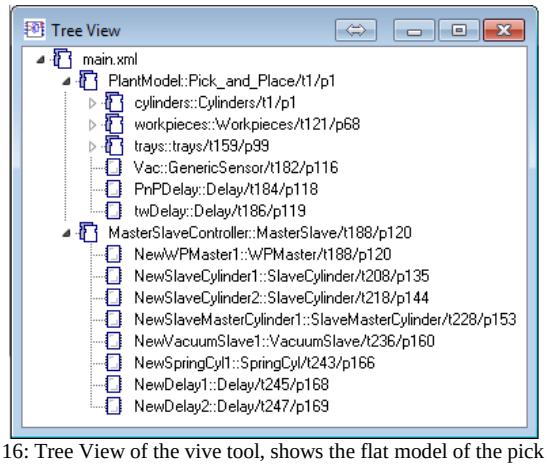

Fig 16: Tree View of the vive tool, shows the flat model of the pick and place robot

The 3 cylinder pick and place robot model checking resulted in a state space of 3406 states and all the properties were verified to be true. The 6 cylinder pick and place robot model checking resulted in a state space of 5853 states and all the properties were verified to be true. 100% verification of all the properties proves that the designed controller for the mechatronic system is reliable, stable and satisfactory.

TABLE 1: LIST OF SAFETY, LIVENESS AND FUNCTIONAL PROPERTIES FOR THE PICK AND PLACE ROBOT.

	Specifications
Safety	Opposite actuator signals to the horizontal cylinders should never be emitted at the same time.
Safety	If the signal to descend the vertical cylinder is emitted, the horizontal cylinder should stand still.
Safety	If there is an emission of a control command corresponding to movements of the horizontal cylinders then the sensor "vcu" of all the vertical cylinders must be true.
Safety	The horizontal cylinders can move only if the value of sensor "vcu" of all vertical cylinders is true.
Liveness	Absence of deadlocks in the (decentralized) control logic.
Functional	If a part is detected by pp1, pp2 or pp3, then in future one of the horizontal cylinders will be extended (Fig 1(b)).
Functional	If a part is detected by pp1, pp2 or pp3, then in the future, the part will be removed from the tray as given in Fig 1(b).
Functional	When the vertical cylinder goes down, both horizontal cylinders are (and remain) in their end positions (home or end).

VII. Conclusion

One of the main issues with formal verification is design and development of formal models. It is time consuming, needs some level of understanding and experience in modeling. In this paper a new reusable and scalable model has been proposed, developed and tried on different reconfigurable systems. With the use of master slave controller architecture, any reconfigured systems of the pick and place robot was automatically generated and model checked for pre-defined properties. Even the writing of the CTL properties can be fully automated using a Meta Model representation of the matrix in Fig 1(b) and also the list of properties in Table 1.

VIII. Future Work

The first work which is already in progress is creating a meta- model using some standards such as CAEX and an application that can read such a CAEX schema and generate the master controller like the work piece manager. Work is also in progress to formulate this idea to prove the advantages of the master-slave architecture. It will also be interesting to integrate the ideas such as introducing selective non determinism in the plant model as expressed in [32] to achieve better verification results.

References

[1] G. Black and V. Vyatkin, "Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499," *Automation Science and Engineering, IEEE Transactions on*, vol. 7, pp. 337-351, 2010.

[2] International Electrotechnical Commission - IEC61499, "Function Blocks for Industrial Process Measurement and Control Systems – Part 1: Architecture," ed. Geneva: Tech. Comm. 65, Working group 6, 2005.

[3] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 277-284 vol.2.

[4] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design* vol. 2: ISA 2007.

[5] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *Industrial Informatics, IEEE Transactions on*, vol. 7, pp. 768-781, 2011.

[6] V. Vyatkin, H.-M. Hanisch, S. Karras, T. Pfeiffer, and V. Dubinin, "Rapid engineering and re-configuration of automation objects aided by formal modelling and verification," *International Journal of Manufacturing Research*, vol. 1, pp. 382-404, 01-01-2006 2006.

[7] C. Maffezzoni, L. L. Ferrarini, and E. Carpanzano, "Object-oriented models for advanced automation engineering - modular modeling in an object oriented database," *Control Engineering Practice*, vol. 7, pp. 957-968, 1999.

[8] nxtControl. (2012). *NxtSTUDIO*. Available: www.nxtcontrol.com

[9] *ICS Triplex ISaGRAF Workbench for IEC 61499/ 61131, v6*. Available: http://www.icstriplex.com/

[10] *FBDK – Function Block Development Kit*. Available: www.holobloc.com

[11] R. Drechsler, *Advanced Formal Verification*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[12] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge: The MIT Press, 1999.

[13] L. Pinzon, M. A. Jafari, H. M. Hanisch, and Z. Peng, "Modeling admissible behavior using event signals," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, pp. 1435-1448, 2004.

[14] V. Vyatkin, P. Starke, and H.-M. Hanisch. (1999-2002). *ViVe and SESA Model Checkers*. Available: http://www.ece.auckland.ac.nz/~vyatkin/tools/modelchekers.html

[15] D. Missal, M. Hirsch, and H. M. Hanisch, "Hierarchical distributed controllers - design and verification," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, 2007, pp. 657-664.

[16] V. Vyatkin and H. M. Hanisch, "A modeling approach for verification of IEC1499 function blocks using net condition/event systems," in *Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99. 1999 7th IEEE International Conference on*, 1999, pp. 261-270 vol.1.

[17] V. Vyatkin and H.-M. Hanisch, "Verification of distributed control systems in intelligent manufacturing," *Journal of Intelligent Manufacturing*, vol. 14, pp. 123-136, 2003.

[18] H. C. Lapp, C. Gerber, and H. M. Hanisch, "Improving verification and reliability of distributed control systems design according to IEC 61499," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, 2010, pp. 1-8.

[19] V. Vyatkin, Hanisch, H.M., Pfeiffer, T., "Modular typed formalism for systematic modeling of automation systems," in *1st IEEE Conference on Industrial Informatics (INDIN'03)*, Banff, Canada, 2003.

[20] K. G. Larsen, P. Pettersson, and Y. Wang, "Compositional and symbolic model-checking of real-time systems," in *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, 1995, pp. 76-87.

[21] K. G. Larsen, P. Pettersson, and W. Yi, "Model-Checking for Real-Time Systems," in *Proceedings of the 10th International Conference on Fundamentals of Computation Theory*, Dresden, Germany, 1995, pp. 62-88.

[22] H.-M. Hanisch, "Closed-Loop Modeling and Related Problems of Embedded Control Systems in Engineering," in *Abstract State Machines 2004. Advances in Theory and Practice*. vol. 3052, W. Zimmermann and B. Thalheim, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 6-19.

[23] H.-M. Hanisch and A. Lüder, "Modular modeling of closed-loop systems," in *Proc of Colloquium on Petri Net Technologies for Modeling Communication Based Systems*, ed Berlin, Germany, 2000, pp. 103-126.

[24] S. Preusse and H. M. Hanisch, "Specification and verification of technical plant behavior with symbolic timing diagrams," in *Design and Test Workshop, 2008. IDT 2008. 3rd International*, 2008, pp. 313-318.

[25] V. Vyatkin, H.-M. Hanisch, C. Pang, and C.-H. Yang, "Closed-loop modeling in future automation system engineering and validation," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, vol. 39, pp. 17-28, 2009.

[26] V. Vyatkin and H. M. Hanisch, "Design of Controllers aiming at Plug-and-Play Engineering of Automated Systems from Mechatronic Components," presented at the Annual Conference of Italian Automation Society (ANIPLA), Rome, Italy, 2006.

[27] M. Sorouri, S. Patil, and V. Vyatkin, "Plug-and-Play IEC 61499 function blocks for distributed control design of Intelligent Mechatronic Systems," in *10th Annual IEEE International Conference on Industrial Informatics*, Beijing, China (Paper just accepted), 2012.

[28] V. Vyatkin and H.-M. Hanisch, "Design of Controllers for Plug-And-Play Composition of Automated Systems from Smart Mechatronic Components," presented at the Annual Conference of Italian Automation Society (ANIPLA), Rome, Italy, 2006.

[29] R. Sreenivas and B. Krogh, "On condition/event systems with discrete state realizations," *Discrete Event Dynamic Systems*, vol. 1, pp. 209-236, 1991.

[30] Z. Mengchu and V. Kurapati, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Hackensack, NJ: World Scientific Publishing Company, 1999.

[31] M. Khalgui, "NCES-based modelling and CTL-based verification of reconfigurable embedded control systems," *Computers in Industry*, vol. 61, pp. 198-212, 2010.

[32] S. Patil, S. Bhadra, and V. Vyatkin, "Closed-loop formal verification framework with non-determinism, configurable by meta-modelling," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 3770-3775.