Hybrid Intelligent Systems for Pneumatic Sequential Circuit Design

Hybridizing Genetic Algorithms and Rule-based Logic Programming

G Sajaysurya
Department of Production Engineering
PSG College of Technology
Coimbatore, India
sajay.edu@gmail.com

G Saravana Kumar
Department of Engineering Design
Indian Institute of Technology Madras
Chennai, India
gsaravana@iitm.ac.in

***Abstract*— Logic synthesis is an important step in process design and automation. Electronics witnessed tremendous improvements in the reliability of automatically designed circuits with reduced lead time with the application of logic synthesis in circuit design. Industrial process automation systems using hydraulic and pneumatic circuits need logic synthesis to obtain proper sequence of process operations. Deterministic approaches have been used for decades to aid in logic synthesis. These approaches are limited by the volume of data that could be handled because of phenomena like exponential explosion. This paper presents an approach based on hybrid intelligence for logic synthesis for industrial automation systems using pneumatic sequential circuits. The proposed logic synthesis system starts with the sequence of operations as its input and generates the truth table required for the pneumatic circuit for performing the given set of operations. The proposed system hybridizes rule based logic programming and genetic algorithms and is capable of solving considerably difficult problems through an evolutionary strategy. Case studies are presented to demonstrate the capability of the proposed approach for logic design for pneumatics based industrial automation.**

Keywords- logic synthesis; sequential pneumatic circuits; asynchronous circuits; genetic algorithm; rule based programming

I. Introduction

Pneumatics is the application of pressurized air for the purpose of mechanical actuation. In industrial automation, pneumatics is preferred over the electrical counterparts because of its reliability, safety and low cost of operation. Sequencing of operations is one of the extensively used subfields of automation in semi-automated industries. Pneumatic actuators operating in a sequence controlled by programmable electronic controllers is common in modern industries. To analyze such systems one can build a mathematical model based on finite state automaton. With the implications drawn from the automata theory and discrete mathematics, numerous works have been carried out to develop fixed set of rules for the design of pneumatic sequential circuits. The most primitive application of Boolean logic for pneumatic circuit synthesis was done in the early 1970s [1]. These techniques were adopted and are still followed by the industries [2,3]. Meanwhile, numerous attempts have been made to automate the design phase of the development of pneumatics sequential systems using expert systems. Object oriented approaches were initially done in [4] using C++. Full-fledged expert systems using inductive logic programming languages like PROLOG were used and the expert system PNEUMAES was also made as described in [5]. The primary problem with these works is that they mimic procedures that are based on the pattern recognition capabilities of the human which becomes inefficient when applied to a computational algorithm. On the other hand the advent of the evolutionary computing methods likes genetic algorithms that mimic the nature to successfully solve many engineering problems particularly involving uncertainties. The applications of such evolutionary methods for the problem of logic synthesis particularly for industrial automation using pneumatics has not been reported in literature and the same has been taken as the objective for the present work.

II. Nomenclature

Consider an industrial automation using pneumatic system. The components of such a system will constitute of the actuators (pneumatic cylinders), position sensors, control valves and memory states and additionally, relays in a programmable logic controller (PLC). The nomenclature reported in table 1 is followed throughout the paper.

TABLE I. NOMENCLATURE

Symbol	Explanation
A, B, C, D	Pneumatic cylinders
W, X, Y, Z	Auxiliary memory variables (AMV) in a PLC or relays (in e-pneumatics) or group control valves (GCV) in pneumatics system)
W1, X1, Y1, Z1	Set signal to a (GCV/AMV/Relay)
W0, X0, Y0, Z0	Reset signal to a (GCV/AMV/Relay)
A1, B1, C1, D1	Extend signal to a pilot controlled memory valve that controls the respective cylinder with the name preceded by 1.
A0, B0, C0, D0	Retract signal to a pilot controlled memory valve that controls the respective cylinder with the name preceded by 0.
a0, b0, c0, d0	Position sensors sensing the end of retraction of cylinders A, B, C, D respectively. Their value is 1 only when the cylinder remains in the completely retracted position and is 0 in all other conditions.

a1, b1, c1, d1	Position sensors sensing the end of extension of cylinders A, B, C, D respectively. Their value is 1 only when the cylinder remains in the completely extended position and is 0 in all other conditions.
$\bar{w}, \bar{x}, \bar{y}, \bar{z}$	The outputs of normally closed contacts of a relay. Each of their value is 1 when the corresponding W, X, Y, Z is in RESET condition and is 0 when the corresponding W, X, Y, Z is in SET condition.
w, x, y, z	The outputs of normally open contacts of a relay. Each of their value is 1 when the corresponding W, X, Y, Z is in SET condition and is 0 when the corresponding W, X, Y, Z is in RESET condition.

III. METHODOLOGY

The proposed method for logic synthesis for pneumatic circuit is a hybrid approach utilizing the best of rule based logic programming and genetic algorithms (GA) an evolutionary method. The schematic of the approach is shown in fig. 1. The methodology involving GA and rule based logic programming is outlined with the help of a simple illustration. The problem illustrated is a semi-automated drilling machine. Let pneumatic cylinder A control the drill vice and pneumatic cylinder B control the movement of a live (rotating) drill bit. The requirement is that, when a start signal is provided, the drill vice must clamp the work piece followed by the immediate reciprocating motion of the drill bit. Once the drill bit is retracted, the drill vice unclamps the work piece and is ready for the next drill cycle. Thus the sequence could be expressed as "A1 B1 B0 A0".

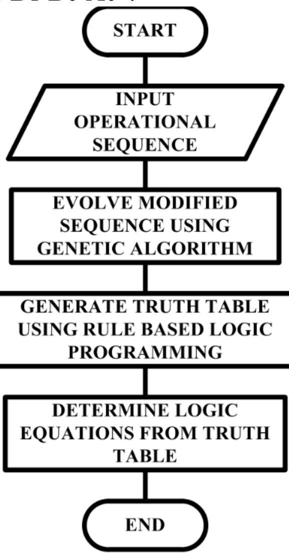
START
INPUT OPERATIONAL SEQUENCE
EVOLVE MODIFIED SEQUENCE USING GENETIC ALGORITHM
GENERATE TRUTH TABLE USING RULE BASED LOGIC PROGRAMMING
DETERMINE LOGIC EQUATIONS FROM TRUTH TABLE
END

Figure 1. Pnematic circuit logic synthesis

It is also important that the completion of all the cylinder movements must be confirmed before actuating the next step in the sequence. In such a case the position sensors confirming the establishment of a particular state can be used to actuate the next step in the sequence. Thus the automaton modeling this system could be represented as shown in fig. 2, where the finite state of the cylinders and the transitions are described. The automaton is invalid because of the ambiguity introduced by exactly similar states (A1B0) actuating separate dissimilar states (A1B1, A0B0). Such a problem is generally tackled by introducing intermediate states either using electric relays, pneumatic GCVs or AMVs in case of a PLC. In this example the sequence can be modified as "A1 B1 X1 B0 A0 X0". The modified automaton is shown in fig. 3. Such a sequence is usually arrived by using fixed rules that are converted into a logic program that arrives at this result deterministically. In this example the introduction of state X (relay / memory variable) differentiates the state transition. For large circuits involving many state changes, the deterministic approaches are not effective. In the present approach a GA based search for the optimal modified sequence has been proposed that can handle large volume of state transition data.

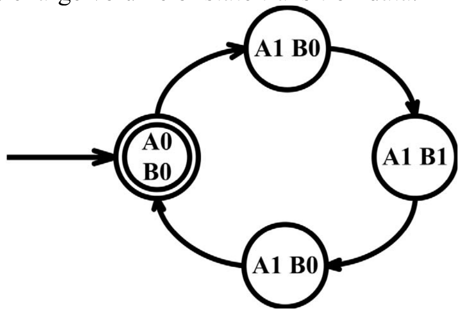

Figure 2. A1 B1 B0 A0 automaton

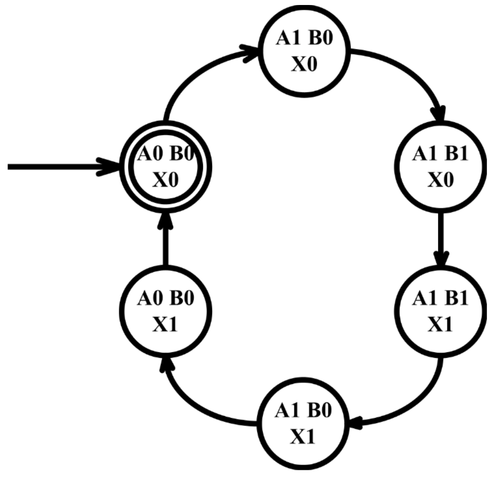

Figure 3. A1 B1 X1 B0 A0 X0 automaton

For simplicity in representation and for further processing a truth table representation is followed. In the truth table representation, the particular combination of position sensors representing the state of the system is marked with "1" in the field named by the step to be followed. Thus a sequential logic is expressed in the form of a combinational logic. The truth table for A1 B1 B0 A0 is shown in table 2. In the truth table *a* and *b* are the position sensors for the cylinder A and B respectively. Here only a single variable is taken instead of *a0*, *a1* and *b0*, *b1*. This is valid provided *a0* and *a1* are mutually exclusive. Here, when *a* = 0, then *a0* = 1 and *a1* = 0. When *a* = 1 then *a0* = 0 and *a1* = 1. Thus *a0* and *a1* can never have the same value. But physically, *a0* =

a1 = 1 is impossible whereas *a0* = *a1* = 0 is possible when the cylinder is in a transition state of either extension or retraction. Nevertheless, such an assumption is made to reduce the size of the truth table and hence to improve the convergence capabilities of computing algorithm. The problem associated with this assumption can be removed by making the position sensors mutually exclusive either by using memory valves or by relay latching in case of electro pneumatics. In the truth table (table. 2) the state of position sensors *a* = 1, *b* = 0 portrays the ambiguity by simultaneous actuation of A0 and B1 which is not indented in the process sequence. In order to avoid such ambiguity a modified sequence is evolved by enforcing the following requirements:

- ***Cyclicity*** **–** The starting state and the ending state of the operational sequence must be the same
- ***Field monopoly*** **–** A particular state must actuate only one step and hence there must be only one "1" in a field for each row.

TABLE II. INITIAL TRUTH TABLE FOR SEQUENCE A1 B1 B0 A0

a	*b*	A1	A0	B1	B0
0	0	1			
0	1				
1	0		1	1	
1	1				1

In the present methodology this requirement of the modified sequence is converted into an algebraic fitness function and is utilized for genetic evolution. The GA adopted follows the evolutionary strategies approach instead of the approach developed by Holland. In this genetic algorithm, starting from a given population of μ parents, λ offspring are generated by crossing and mutating random parents. The parents and the offspring are allowed to coexist and compete in fitness. The best μ out of the μ+λ are selected deterministically which act as parents for the next generation. This process continues till required degree of convergence is achieved. The following GA parameters have been used in the present implementation:

- $\mu = \lambda$
- Mutation rate = 0.25
- Crossover rate = 0.75
- Initial population size = 500
- Maximum number of generations = 500

The mutation which brings in variation and crossover which preserves building blocks have to be balanced for good convergence of the algorithm. The size of the initial population too serves the introduction of variation in the population. The larger the number of generations better will be the results at the expense of time and computer resources. Since this is a novel approach, the GA parameters were chosen by empirical observation. Though they were not optimal, they have adequately served to prove the concept.

The structure of each and every chromosome is similar and is directly dependent on the sequence taken as input and the maximum number of auxiliary memory variables that are wished to be used. The decision made on the number of auxiliary variables to be used is strategic as too low a number renders the system incapable of handling difficult sequences and at the same time too high a number leads to inefficiency in convergence. For example, the individuals used for the example sequence resemble as shown in fig 4. The components of the base sequence occupy the gene index (loci) at intervals determined by the number of auxiliary memory variables. This is so because, the maximum number of auxiliary memory variable designations that can occur between 2 steps of the base sequence is equal to the maximum number of auxiliary memory variables the system is equipped with. The current illustration utilizes four auxiliary memory variables (W, X, Y and Z). The initial population consists of randomly generated individuals as shown in the fig. 5. Generally, introducing the origin as an individual in the initial population has proved to improve the converging capabilities of minimization problem when GA is used for optimization [6]. In the present implementation the origin (individual shown in fig 4) is included in the initial population. The genes that comprise the base sequence are kept silent. This technique was derived from the gene silencing operator introduced in [7]. The silent genes will never undergo mutation. When a crossover happens it chooses similar points in both individuals so that the base sequence is not disturbed. Such a crossover also conserves the length of the chromosomes. This crossover is classified as homologous as explained in [8] and can be compared with the biological crossover that takes place in higher animals that reproduce sexually.

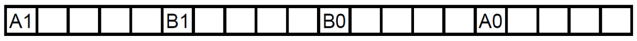

Figure 4. An individual in GA population

The traditional single point crossover is used in the present implementation. Of the two offspring only one survives and both have equal probability of survival. Mutation results in one among the possible alleles (say "W1" and "W0" for variable W). It also includes an empty space. This empty space is not counted while determining the length of the modified sequence. The mechanism of mutation and crossover is portrayed in fig. 5 and fig. 6. During evaluation each (individual) modified sequence is converted into a truth table as described previously and the fitness is analyzed. The fitness is computed as

$$f(x) = \max(\text{No. of occupants in a field}) + (\text{size of modified sequence} / \max(\text{size of an individual})) \quad (1)$$

If the sequence is cyclic, then the fitness is reduced by a value of one. The algorithm tries to minimize the fitness thus finding the valley in the search space. The first priority is given to the number of occupants in a row for which the least value is 1. When the sequence is cyclic the fitness is reduced by 1. Once the fitness is between 0 and 1 the sequence is completely correct. Now the algorithm concentrates to reduce the fractional part of the fitness thus reducing the size of the modified sequence so that least number of memory variables is utilized. For the current illustration, the algorithm converged into the modified sequence "A1 B1 X1 B0 A0 X0". It has to be noted that though four memory variables were included, the GA evolved a sequence with minimum memory variables (in this case, only one i.e. X). The truth table for the above sequence filled in the order of occurrence of states is provided in table 3.

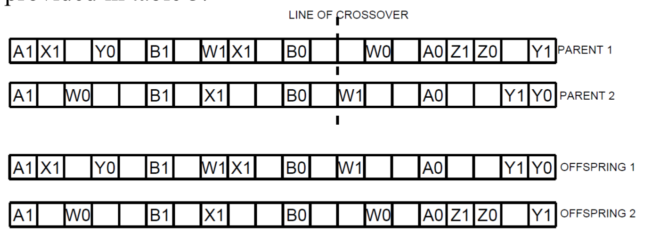
LINE OF CROSSOVER
A1 X1 Y0 B1 W1 X1 B0 W0 A0 Z1 Z0 Y1 PARENT 1
A1 W0 B1 X1 B0 W1 A0 Y1 Y0 PARENT 2
A1 X1 Y0 B1 W1 X1 B0 W1 A0 Y1 Y0 OFFSPRING 1
A1 W0 B1 X1 B0 W0 A0 Z1 Z0 Y1 OFFSPRING 2

Figure 5. Mechanism of crossover

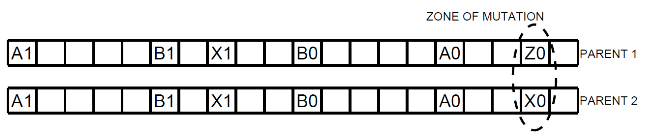

Figure 6. Mechanism of mutation

The obtained order-of-fill truth table has to be validated for signal clash and safety before logic synthesizers could act on the same. The following are the requirements for obtaining a truth table that avoids signal clash and ensures safe operation:

- The required sequence of operation takes place
- There are as many don't-care conditions as possible
- Unsafe hazards are eliminated

The rule base developed for fulfilling the above requirements in the truth table are:

- The rows at the bottom that are never visited by the sequence are filled with "x", a mark of don't-care condition.
- The fields with "1" are located and their opposing signal is marked as "0" to ensure actuation in that step. For example, if A1 is 1 in a row, in the same row A0 is marked 0 to ensure safe operation of A1.
- In a particular column first identify a "0". Fill all the forth coming fields with 0 until a "1" is reached. In the same column spot a "1" and fill all forth coming fields with "x" until a "0" is reached. Do this for all output columns. By doing so, two opposing signals are never actuated simultaneously thus avoiding signal clash, unsafe hazards and inadvertent actions. Further this prevents the occurrence of a change in state of an actuator from SET to RESET or vice versa until demanded.

The truth table completed using these rule bases for the example sequence is shown in table 4.

TABLE III. ORDER-OF-FILL TRUTH TABLE FOR SEQUENCE A1 B1 X1 B0 A0 X0

a	*b*	*x*	A1	A0	B1	B0	X1	X0
0	0	0	1					
1	0	0			1			
1	1	0					1	
1	1	1				1		
1	0	1		1				
0	0	1						1
0	1	0						
0	1	1						

TABLE IV. VALIDATED TRUTH TABLE FOR SEQUENCE A1 B1 X1 B0 A0 X0

a	*b*	*x*	A1	A0	B1	B0	X1	X0
0	0	0	1	0	0	x	0	x
1	0	0	x	0	1	0	0	x
1	1	0	x	0	x	0	1	0
1	1	1	x	0	0	1	x	0
1	0	1	0	1	0	x	x	0
0	0	1	0	x	0	x	0	1
0	1	0	x	x	x	x	x	x
0	1	1	x	x	x	x	x	x

The proposed rules are programmed as a rule based logic program. The described method which hybridizes the genetic based modified sequence evolution with minimum auxiliary variables and rule based validation can be used for logic synthesis to develop Boolean equations for circuit construction. In the present study the same has been accomplished using Quine McCluskey algorithm [9]. Quine McCluskey is a deterministic algorithm that is functionally similar to Karnaugh mapping. It generates the minimized logic equation that satisfies a given truth table. It is actually a computer friendly way of Karnaugh mapping that could handle a very high volume of data. For the truth table (in table 4) the Boolean equations generated using this algorithm are shown in table 5. In order to make use of them in pneumatic circuit building (using position sensors) they can be converted as shown in table 5 second column.

TABLE V. BOOLEAN EQUATIONS FOR SEQUENCE A1 B1 X1 B0 A0 X0

	Modified using position sensors
$A0 = \bar{b} \wedge x$ $A1 = \bar{x}$ $B0 = x$ $B1 = \bar{x} \wedge a$ $X0 = \bar{a}$ $X1 = b$	$A0 = b0 \wedge x$ $A1 = \bar{x}$ $B0 = x$ $B1 = \bar{x} \wedge a1$ $X0 = a0$ $X1 = b1$

IV. CASE STUDY

The proposed hybrid intelligent system for logic synthesis has been tested with several case studies and two such studies are presented here for demonstration and validation. In the first case study a PLC controlled pneumatic sheet metal punching machine is programmed as proposed. The punching press is pneumatically actuated and consists of 2 pneumatic actuators and a non-conventional fluidic muscle actuator to utilize its quick response for the current application. The punching press is configured as shown in fig. 7 with;

A: Spring returned pneumatic muscle

B: Double acting guided cylinder for shearing sheet metal

C: Feed cylinder to feed the continuous sheet metal stock in stages

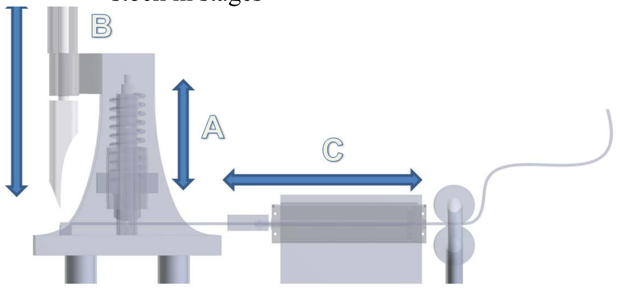

Figure 7. Punching press configuration

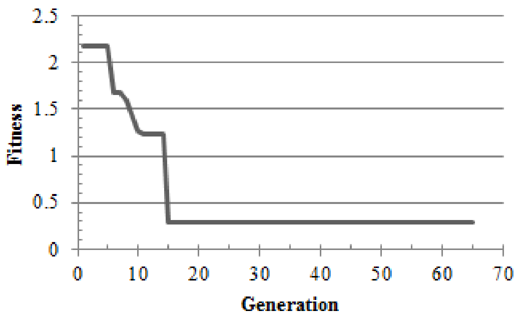

Figure 8. Convergence plot for proposed Genetic Algorithm

The sequence starts with the punching stroke manipulated by the pneumatic muscle A. The pneumatic muscle A is de-energized such that punching occurs because of the energy released by the strain stored in the compression spring. At the end of this stroke, the pneumatic muscle is once again energized to retract the punch. Cylinder C feeds the sheet metal stock with a single to and fro motion such that cylinder extension is immediately followed by cylinder retraction. The punched sheet metal protruding out of the machine is sheared by the blade actuated by cylinder B with a reciprocating motion such that, the component is separated from the continuous stock for further processing and the machine is made ready for the next sequence. This sequence can be represented as "A0 A1 C1 C0 B1 B0". This is taken as the input sequence for the proposed hybrid intelligent logic synthesis system. The initial sequence is modified by the GA part of the system to obtain an unambiguous sequence with the introduction of memory variables as "A0 W1 A1 C1 X1 C0 W0 B1 X0 B0". The convergence characteristic of the GA for evolving the above sequence is shown in fig. 8. Two memory variables (W and X) have been added. This modified sequence is then acted upon by the rule base program of the hybrid intelligent system to generate a truth table resulting in safe operation of the pneumatic system. The generated truth table is reported in table 5.

TABLE VI. COMPLETED TRUTH TABLE FOR A0 W1 A1 C1 X1 C0 W0 B1 X0 B0

a	*b*	*c*	*w*	*x*	A1	A0	B1	B0	C1	C0	W1	W0	X1	X0
1	0	0	0	0	0	1	0	x	0	x	0	x	0	x
0	0	0	0	0	0	x	0	x	0	x	1	0	0	x
0	0	0	1	0	1	0	0	x	0	x	x	0	0	x
1	0	0	1	0	x	0	0	x	1	0	x	0	0	x
1	0	1	1	0	x	0	0	x	x	0	x	0	1	0
1	0	1	1	1	x	0	0	x	0	1	x	0	x	0
1	0	0	1	1	x	0	0	x	0	x	0	1	x	0
1	0	0	0	1	x	0	1	0	0	x	0	x	x	0
1	1	0	0	1	x	0	x	0	0	x	0	x	0	1
1	1	0	0	0	x	0	0	1	0	x	0	x	0	x
0	0	0	0	1	x	x	x	x	x	x	x	x	x	x
0	0	0	1	1	x	x	x	x	x	x	x	x	x	x
0	0	1	0	0	x	x	x	x	x	x	x	x	x	x
0	0	1	0	1	x	x	x	x	x	x	x	x	x	x
0	0	1	1	0	x	x	x	x	x	x	x	x	x	x
0	0	1	1	1	x	x	x	x	x	x	x	x	x	x
0	1	0	0	0	x	x	x	x	x	x	x	x	x	x
0	1	0	0	1	x	x	x	x	x	x	x	x	x	x
0	1	0	1	0	x	x	x	x	x	x	x	x	x	x
0	1	0	1	1	x	x	x	x	x	x	x	x	x	x
0	1	1	0	0	x	x	x	x	x	x	x	x	x	x
0	1	1	0	1	x	x	x	x	x	x	x	x	x	x
0	1	1	1	0	x	x	x	x	x	x	x	x	x	x
0	1	1	1	1	x	x	x	x	x	x	x	x	x	x
1	0	1	0	0	x	x	x	x	x	x	x	x	x	x
1	0	1	0	1	x	x	x	x	x	x	x	x	x	x
1	1	0	1	0	x	x	x	x	x	x	x	x	x	x
1	1	0	1	1	x	x	x	x	x	x	x	x	x	x
1	1	1	0	0	x	x	x	x	x	x	x	x	x	x
1	1	1	0	1	x	x	x	x	x	x	x	x	x	x
1	1	1	1	0	x	x	x	x	x	x	x	x	x	x
1	1	1	1	1	x	x	x	x	x	x	x	x	x	x

Quine McCluskey algorithm is then used to synthesize logic from the truth table. The Boolean equations describing the pneumatic circuit are,

$$A0 = b0.\overline{w}.\overline{x}$$
$$A1 = w$$
$$B0 = \overline{x}$$
$$B1 = x.\overline{w}$$
$$C0 = x$$
$$C1 = a1.w.\overline{x}$$
$$W0 = x.c0$$
$$W1 = a0$$
$$X0 = b1$$
$$X1 = c1$$

The pneumatic circuit and the PLC wiring diagram for this Boolean equations is shown in fig. 9. The end-of-extension position sensors are alone latched with a relay to have a memory. This is done to reduce the number of relays to be used. The synthesized Boolean equations can be converted into an equivalent PLC ladder diagram or instruction list code to run the physical pneumatic circuit.

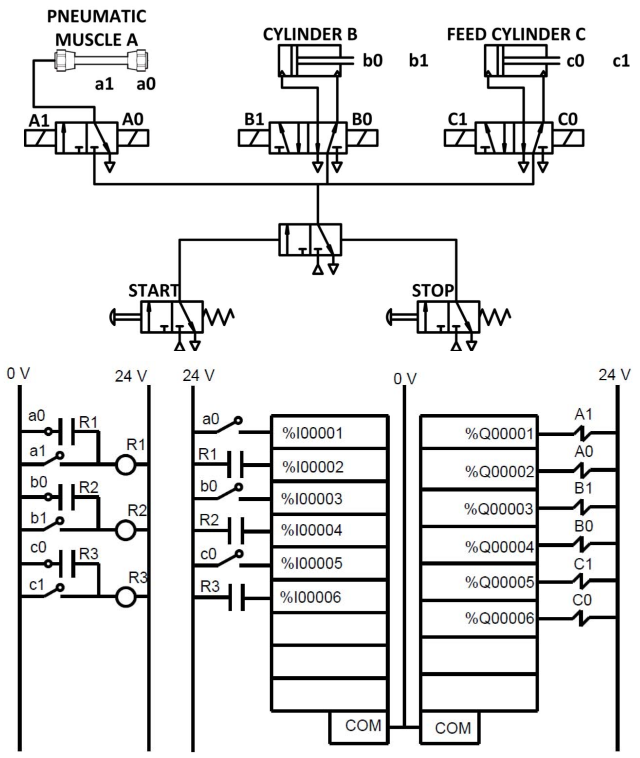

Figure 9. Pneumatic circuit and PLC wiring configuration for the punching press

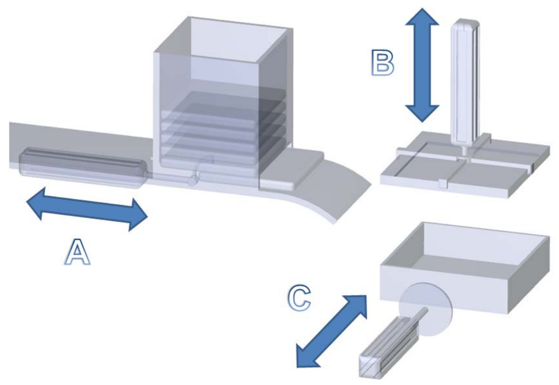

Figure 10. Material handling unit configuration

The proposed system can handle multiple signal clashes which has not been reported earlier in literature [4,5]. A material handling unit to pack a given number of units in a package (fig. 10) is simulated to demonstrate this. Consider the automation of this task with three cylinders. Let cylinder A push the units from a magazine to the package, cylinder B seal the package and is further pushed upon by cylinder C on to a conveyor. For packing three units in a package, the sequence is "A1 A0 A1 A0 A1 A0 B1 B0 C1 C0" and the synthesized modified sequence evolved by GA is "A1 W1 A0 Y1 A1 Z1 A0 W0 A1 X0 A0 B1 Z0 B0 C1 X1 Y0 C0". The Boolean equations generated by using the rule base and Quine McCluskey algorithm there after are;

$$A0 = \bar{x} + \bar{y}.w + z.w$$
$$A1 = c0.x.\bar{w} + c0.x.y.\bar{z}$$
$$B0 = \bar{z}$$
$$B1 = a0.z.\bar{x}$$
$$C0 = \bar{y}$$
$$C1 = b0.\bar{x}.\bar{z}$$
$$X0 = a1.z.\bar{w}$$
$$X1 = c1$$
$$Y0 = c1.x$$
$$Y1 = a0.w$$
$$Z0 = b1$$
$$Z1 = a1.y$$
$$W0 = a0.z$$
$$W1 = a1.\bar{z}$$

V. Conclusion

A hybrid intelligent system for logic synthesis for industrial automation systems using pneumatic sequential circuits has been presented. The system uses a combination of genetic algorithm based evolution and rule base program based modification of the input pneumatic sequence to obtain an unambiguous sequence with safe operation. The method presented can handle higher number of actuators and signal clashes which is a major problem faced by other reported methods [4, 5]. The results are encouraging with promising industrial automation application. The authors are currently working on the proposed system with a plan to include an evolutionary method for Boolean equation synthesis from the truth table of the operation sequence.

References

[1] R. M. H. Cheng and K. Foster, “Systematic method of designing fluidic – pneumatic control circuits,” ARCHIVE: Proceedings of the Institution of Mechanical Engineers 1847-1982, vol. 186, pp. 401-408, 1972.

[2] C. H. Chung, K. Tam and G. Liu, “Pneumatic sequential circuits,” Industrial Engineering Journal, vol 1, 1980

[3] P. Rohner, Fluid Power Logic Circuit Design, London: The Macmillan Press Ltd., 1979.

[4] P. K. Wong, T. P. Leung, C. W. Chuen and W. H. Chan, “Object-oriented CAD for electro-pneumatic sequential circuit design in low cost automation,” IEEE Symposium on Emerging Technologies and Factory Automation, pp. 278-284, 1994, DOI:10.1109/ETFA.1994.402033

[5] S. K. Sim and P. S. K. Chua, “Symbolic pattern manipulation of Karnaugh-Veitch maps for pneumatic circuits,” Artificial Intelligence in Engineering, vol. 10, pp. 71-83, 1996, DOI:10.1016/0954-1810(95)00017-8.

[6] H. Maaranen, K. Miettinen, and M. M. M\&\#228;kel\&\#228;. 2004. “Quasi-random initial population for genetic algorithms,” Computers and Mathematics with Applications, vol. 47, pp. 1885-1895, June 2004, DOI=10.1016/j.camwa.2003.07.011

[7] S. S. Sathya and S. Kuppuswami, “Gene silencing – A genetic operator for constrained optimization,” Applied Soft Computing, DOI: 10.1016/j.asoc.2011.01.042.

[8] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone, Genetic Programming – An Introduction: On the automatic Evolution of Computer Programs and its Applications, Morgan Kaufmann, 1998.

[9] E. J. McCluskey, “Algebraic minimization and the design of two-terminal contact networks” D.Sc. dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, 1956.