ELSEVIER

CrossMark

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 1726 – 1734

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017,
27-30 June 2017, Modena, Italy

Evolutionary Algorithms for Programming Pneumatic Sequential Circuit Controllers

Sajaysurya Ganesh[a], Saravana Kumar Gurunathan[b,*]

[a]*Former Intern, Indian Institute of Technology Madras, Chennai 600036, India*
[b]*Associate Professor, Indian Institute of Technology Madras, Chennai 600036, India*

Abstract

Sequential actuation of pneumatic cylinders is a common form of automation in small and medium scale industries. By changing such actuation sequences to suit the different products being processed, flexible automation can be economically realized. However, changing the actuation sequence involves manually reprogramming Programmable Logic Controllers (PLC), which consumes time and hinders the implementation of flexible automation. This paper presents a novel methodology to automatically program PLCs by evolving logic equations using Genetic Algorithm and Genetic Programming for the desired actuation sequence. Case studies have been presented to demonstrate the possibility of using the proposed methodology to reliably implement flexible automation.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Flexible Automation; Genetic Algorithms; Genetic Programming: Programmable Logic Controller; Pneumatics

---

1. Introduction

Pneumatics is a cheap source of power that is extensively used to operate fixed automation systems in small and medium scale industries. Such automation systems generally actuate pneumatic cylinders repeatedly in a fixed

---

* Corresponding author. Tel.: +91-44-2257-4736
E-mail address: gsaravana@iitm.ac.in

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.299

sequence and thereby perform the same operation on successive products in a process line. In early 1970s, techniques were developed to derive sequential logic equations [1] that can be used to construct sequential pneumatic cylinder actuation circuits. Subsequently, these techniques were refined [2, 3] and are still being used in the industries, particularly for manually generating logic equations for programming PLCs. Availability of PLCs can be leveraged to convert existing fixed automation systems into flexible automation systems as the PLC programs can be dynamically changed to suit changes in product and process requirements. However, the current manually performed techniques for deriving logic equations must be automated to quickly reprogram PLCs without much downtime.

Previous attempts to automate generation of logic equations include object oriented approaches [4] using C++, expert systems like PNEUMAES [5], Web based collaborative environments for simulating pneumatic circuits [6]. Nevertheless, most industries still rely on manual techniques [7] for designing circuits and programming PLCs.

Existing logic synthesizers use procedures that depend upon the pattern recognition capability of humans which becomes inefficient when adapted to a computer. These methods are either based on the Karnaugh Maps [8] or on Quine McCluskey Algorithm [9] which is functionally similar to Karnaugh mapping. The Quine McCluskey algorithm has its practical limits as it is NP-Complete; in other words, the runtime of the Quine-McCluskey algorithm grows exponentially with the input size [10]. Further, these methods are limited to Boolean operators like AND, OR, NOT for generating logic equations and cannot use operators like XOR, which are supported by PLCs.

This paper presents an alternative logic synthesizer that uses Evolutionary Algorithms for deriving logic equations for sequential actuation of pneumatic cylinders. Evolutionary computing methods like Genetic Algorithm (GA) have been successfully applied to various combinatorial optimization problems and have succeeded in obtaining good solutions for several such problems [11, 12]. They have been used to solve high level logic synthesis for VLSI design [13, 14] and reversible logic circuits [15]. Recently, GA has also been used for optimizing the number of logic gates in synchronous sequential circuits to reduce the circuit complexity [16].

The authors of this paper have previously published a method to automate pneumatic sequential circuit design using Genetic Algorithm [17]. This paper develops upon that and employs Genetic Programming for logic synthesis. The subsequent sections detail the complete methodology by which the logic equations and the corresponding PLC programs can be derived from a given actuation sequence of pneumatic cylinders.

Nomenclature

A, B, C, …	Pneumatic cylinders – Actuators
…, X, Y, Z	Auxiliary memory variables (AMV) of a PLC; relays in electro pneumatics
…, X1, Y1, Z1	Set signal sent to a (AMV/Relay)
…, X0, Y0, Z0	Reset signal sent to a (AMV/Relay)
A1, B1, C1, …	Extend actuation signal sent to cylinder with the name that precedes 1.
A0, B0, C0, …	Retract actuation signal sent to cylinder with the name that precedes 0.
a0, b0, c0, …	Signal from position sensors sensing the retraction of cylinder with the name that precedes 0. Their value is 1 when the cylinder remains in the completely retracted position and is 0 otherwise.
a1, b1, c1, …	Signal from position sensors sensing the extension of cylinder with the name that precedes 1. Their value is 1 when the cylinder remains in the completely extended position and is 0 otherwise.
…, $\bar{x}, \bar{y}, \bar{z}$	The outputs of normally closed contacts of relays. Each of their value is 1 when the corresponding relay is in RESET condition and is 0 when the corresponding relay is in SET condition.
…, x, y, z	The outputs of normally open contacts of relays. Each of their value is 1 when the corresponding relay is in SET condition and is 0 when the corresponding relay is in RESET condition.

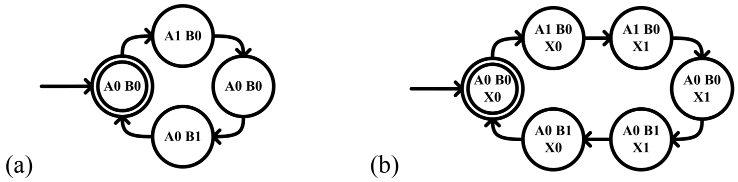

Fig. 1. (a) A1 A0 B1 B0 automaton; (b) A1 X1 A0 B1 X0 B0 automaton