Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 36 (2015) 17 – 22

www.elsevier.com/locate/procedia

25th CIRP Design Conference

Genetic Algorithm Software System for Analog Circuit Design

Miri Weiss Cohen*, Michael Aga, Tomer Weinberg

Software Engineering Department, Braude College of Engineering, Karmiel, Israel

* Corresponding author. Tel.:97249901758; fax: +97249901982. *E-mail address:* miri@braude.ac.il

Abstract

A software system sketcher to facilitate analog circuit design is proposed. The system requires solving two sub-problems. First, the issue of placement of the devices is resolved by using genetic algorithms (GAs), followed by activation of a sub-process that combines routing preferences using a search algorithm, A*. An altruism procedure is implemented over the solution output of the A* search to achieve a better design. The sketcher mimics and implements the combination of experience and intuition required from a human designer.

Peer-review under responsibility of the scientific committee of the CIRP 25th Design Conference Innovative Product Creation
Keywords: analog circuit design; design automation; genetic algorithms

1. Introduction

Analog circuit design is a challenging and complex process that requires fulfilling a variety of goals while conforming to constraints. The engineer's aim is to create a circuit diagram that satisfies specified design goals and complies with the drawing rules globally known as IEC (International Electrotechnical Commission) standards [8]. The design comprises three major stages: topology selection, component sizing and layout generation. Both the selected topology and the sizing must ensure that the resulting circuit satisfies the design objectives. Aaserud and Nielsen [1] noted the following: "In contrast to digital design, most of the analog circuits are still handcrafted by the experts and so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product. Analog circuit design is a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than science." Over the last three decades researchers have extensively investigated the design, control and planning of analog circuit design. Modern automation of this procedure entails heuristics [2,5,9], genetic algorithms [3,6,10,12,13,17], multi objective optimization [14] and swarm intelligence [16].

Genetic algorithms (GAs) are considered to be a part of evolutionary computation (EC) methods. They belong to a class of non-gradient methods that have grown in popularity following the seminal publications by Holland [11] and Goldberg [7], which extended and helped popularize the idea. GAs are stochastic search methods that mimic natural biological evolution. They operate on populations of potential solutions by applying the principle of "survival of the fittest" to produce better and better solutions. A GA uses a population of individuals (solutions) instead of a single solution to perform a parallel search in the problem space. At each generation, a new set of approximations is created by a nature-inspired process. The natural processes commonly mimicked by GAs are selection, breeding, mutation, migration and survival of the fittest.

The basic structure of GAs is iterative, as follows[4]:

1. Population initialization: A GA is constructed randomly to generate an initial population. Each individual chromosome is defined as a combination of circuit units—i.e., components—provided by the user. For each unit *Ui* we select a position (*x, y*) and orientation (0, 90, 180, 270 degrees), as depicted in Fig. 1. Each chromosome represents a possible circuit design.

Peer-review under responsibility of the scientific committee of the CIRP 25th Design Conference Innovative Product Creation
doi:10.1016/j.procir.2015.01.033

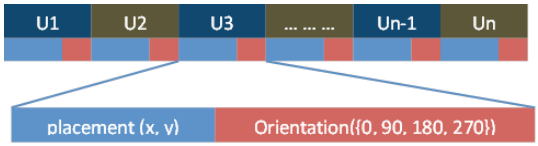


Fig. 1. Unit positioning chromosome structure.

2. Fitness Function: A fitness function is calculated for each individual in the population.
3. Selection: Selection is based on binary tournament selection [4], where two individuals are chosen from the population according to their fitness function.
4. Crossover: Given that population diversity is needed, the GA algorithm provides a crossover procedure that enables reproduction of individuals in the population.
5. Mutation: A schematic procedure is applied to the new chromosome with a prescribed probability.
6. The initial population is replaced by the new population, and Stages 1-5 are repeated recursively until the evaluation function does not develop to a better value.

The aim of this study is to provide the designer with an intuitive tool that mimics the intelligent process of analog system design. The system offers interactive options as well as the choice of a rigid user-defined solution or a totally automatic design solution. The main algorithm of our system is depicted in Fig. 2.

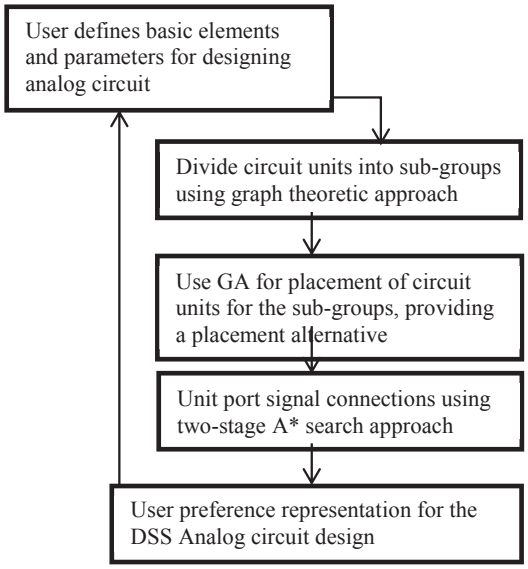


Fig. 2. The flowchart of the DSS of analog circuit design.

2. The Decision Support Sketcher

2.1. Step 1: User defines the basic elements and parameters of the analog circuit design

Each analog circuit design scheme consists of units that will be deployed on the plane to construct a design option for the decision support system (DSS). The problem input is given by the following dataset:

U (designation,$P_u \subseteq P$) – A set of units
P (designation,$S_i \in S$) – A set of pins
S (designation) – A set of signals

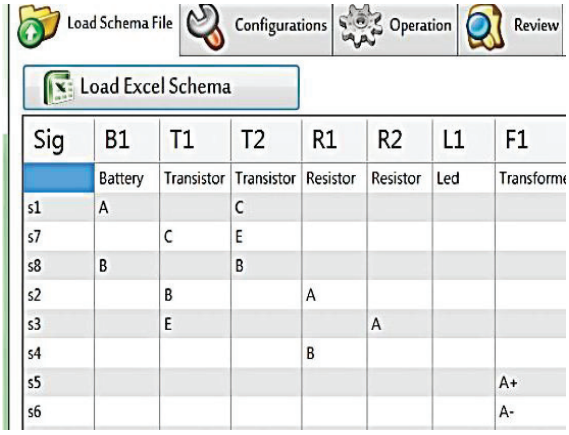
Input for the algorithm will be in the form of the matrix C=[U,S], where each $C_{U,S}$ is assigned a letter and a number representing the pin in the device with which the signal is associated. The process is demonstrated in Fig. 3.

2.2. Step 2: Dividing the circuit units into sub-groups using a graph theory approach

The software implements a partitioning algorithm that tries to obtain groups with the maximum number of connected devices by adding connected devices to the same group. The pseudo code is as follows:

Groups[] (**Devices**[])
Group = new group
  For each **Device** in **Devices**
    For each **Mating_Device** of **Device**
      If **Group** is full
      Group = new group:
      Add **Mating_Device** to **Group**
      Remove **MatingDevice** from **Devices**
    Next **MatingDevice**
  Next **Device**

The algorithm picks a device and puts it in a group. For each device connected to this device, the mating device procedure uses a weighted graph that supplies the edges with weights corresponding to the number of connections to other devices in the group. The groups are formalized according to two criteria: (1) by the number of devices in a group as defined by the user and (2) by the following procedure that chooses the groups according to connectivity weights.



Sig	B1	T1	T2	R1	R2	L1	F1
	Battery	Transistor	Transistor	Resistor	Resistor	Led	Transforme
s1	A		C				
s7		C	E				
s8	B		B				
s2		B		A			
s3		E			A		
s4				B			
s5							A+
s6							A-

Fig. 3. Example signal connection user input (arbitrary)

Fig. 4 shows an example of the influence of user input on group size definition. Both options are in respect to the same input depicted in Figure 3.