

The Invention of CMOS Amplifiers using Genetic Programming and Current-Flow Analysis

Thanwa Sripramong¹, Christofer Toumazou²

¹Department of Computer Engineering, Mahanakorn University of Technology, Bangkok, 10530, Thailand

²Department of Biological and Medical Systems, Imperial College of Science Technology and Medicine, London, SW7 2BT, UK

ABSTRACT

This paper introduces an automated circuit design system for the evolution and subsequent invention of CMOS amplifiers. The proposed system relies on a mix of genetic programming and a new topology-independent design optimisation method referred to as current-flow analysis. Genetic programming evolves new circuit topologies from the collection of primitive devices and basic building blocks. Current-flow analysis screens and corrects circuits using topology-independent design rules. Experimental results show a promising improvement on the design of operational amplifiers that making the automated analogue design environment using genetic programming a lot more practical.

I. INTRODUCTION

The field of automated analogue circuit design has been one of the most appealing research topics for decades and a large number of systems have been developed. General reviews of these systems can be found in [1,2,3,4,5].

Several automated analogue circuit design systems have concentrated on optimisation techniques using circuit design parameters. Examples are DELIGHT.SPICE[6], ECSTASY[7] and ADOPT[8]. Other systems, such as ISAAC/OPTIMAN [9,10], ASTRX/OBLX [11], ANACONDA[12] and GPCAD[13] have introduced global optimisation methods. Finally, systems such as Ampdes[14], BLADES[15], CAMP[16] and ISAID[17,18,19] apply a knowledge base to synthesise circuit topologies as well as guide the optimisation methods.

Recently, genetic algorithms(GAs) and genetic programming(GP) were introduced to the field. Genetic algorithms were firstly introduced as optimisation tools for parameter searches. Both algorithms were then introduced for topology selection and synthesis. DARWIN[20] uses 24 circuit topologies stored in the library as the starting circuits for the evolution of operational amplifiers. The facility of topology synthesis was recently enhanced by two particular systems, the automated WYWIWYG (What-You-Want-Is-What-You-Get) design process[21], and the system introduced by Lohn et al.[22]. Both systems evolve circuit topologies from a collection of primitive devices and at the same time, optimise circuit parameters. This design concept brings automated analogue design closer to a situation where new topologies can be invented without intervention from the users. However, both the WYWIWYG system and the system by Lohn et al. face a new problem in which invented topologies are not optimised or practical because evolving circuits are evaluated by their performance, not by the design.

In this paper we introduce a new automated circuit design system based on genetic programming with the extension of a knowledge base. The proposed system is not only capable of topology invention and component parameter searches but also provides the necessary facility for *design optimisation*. A new topology-independent design optimisation method which we call *current-flow analysis* corrects circuit topologies. A new circuit

representation technique is also introduced to provide more complex component connections, which is essential for high performance amplifier design.

In section II, we investigate the concepts and techniques used for topology synthesis and parameter searches. The proposed system is described in detail in section III. Experimental results showing the evolution of CMOS amplifiers are given in section IV with conclusions in section V.

II. THE TECHNIQUES FOR TOPOLOGY SYNTHESIS AND PARAMETER OPTIMISATION

A. Canonical concepts for topology synthesis and parameter optimisation

Since the early days of automated circuit synthesis, several systems have been designed. Most systems comprise two separated parts. The first part is the *circuit generator*, which synthesises the first draft of a circuit. The second part is the *circuit optimiser*, which optimises circuit parameters to meet the target specifications. We may classify automated circuit analogue design systems by their methodology as follows:

1) *Optimisation tools for parameter searches*: In this group, the key function of such systems is to solve circuit parameters using some kind of optimisation methods. Early automated circuit design systems perform optimisation methods directly on derived analytic functions of circuits. For example, DELIGHT.SPICE[6], ECSTASY[7], and ADOPT[8]. ECSTASY is a system that uses the same design methodology as DELIGHT.SPICE but it improves the search algorithm by introducing the *controlled random search (CRS)* algorithm. In recent years, a concept of using a strategy in which the optimisation methods are shielded from topologies was introduced. In this concept, circuit topologies are selected from the library using heuristic design rules or the topologies are manually selected by the users and are then processed by a *symbolic simulator* or a *circuit compiler* where circuit characteristics are transformed into *cost functions*. Cost functions are then optimised using *simulated annealing*. Such a process shields the optimisation module from user specifications and circuit topologies and thus reduces the amount of work when introducing new designs. The examples are ISAAC/OPTIMAN [9,10] with ISAAC as the symbolic simulator, and ASTRX/OBLX [11] with ASTRX as the circuit compiler. ANACONDA[12] is based on the ASTRX/OBLX system but it applies a new technique referred to as *stochastic pattern search* as an optimiser. Another design approach is transforming analytic equations of circuits into a new domain where circuits can be optimised with guaranteed success. Such a technique was used by GPCAD[13]. GPCAD applies analytic equations of circuits using a new domain called *geometric programming* where transformed analytic functions can be optimised using convex optimisation methods.

2) *Knowledge base for topology synthesis and parameter searches*: In the second group, systems are built using a new concept where various specifications requiring different topologies can be selected or synthesised using design rules. Circuit parameters are initialised and then refined using some kind of optimisation method under guidance by rules. The refining process may also change circuit topologies if necessary. OPASYN[23] uses decision rules for topology selection. Ampdes[14] and CAMPS [16] uses design rules to synthesis circuits and then uses analytic equations to calculate for component sizing. BLADES[15] uses heuristic rules to synthesise circuits from predefined subcircuits. IDAC[24] selects subcircuits from the library and synthesises circuits using design rules. Circuit parameters are initialised using simplified analytic equations. They are then refined by local optimisation algorithms. OASYS[25] and ISAID[17,18,19] use hierarchical structures to represent circuits. User specifications are used by multi-level design plans to synthesise circuits from a collection of subcircuits and primitive devices.

Circuit parameters are then optimised using local optimisation methods under the guidance of derived design plans. ISAID has an advantage over OASYS since it introduces *qualitative reasoning* to generate possible correction strategies and then select the most benefit route for the correction. The correction procedure involves the change of component sizing and circuit topologies.

B. The evolution of circuit topologies and component parameters

A goal in automated circuit design is to design a circuit that the system has no knowledge of. For human designers, the general way to invent a new design is to try-and-test the circuit under required specifications and general knowledge in circuit design. This requires experience and insight of design. The majority of automated circuit design systems are based upon this concept. Although these systems have designed various types of circuits with a level of success, their design is controlled by available rules and subcircuits. While the methodology can guarantee that resulting circuits are practical and optimised if relevant information exists in the systems, it will not guarantee a result when user specifications require topologies that are beyond existing rules and building blocks.

A breakthrough came from the application of evolutionary algorithms. The algorithms are based on the mechanics of natural selection and reproduction of real lives. Circuits can be synthesised without the knowledge of required topologies. Two algorithms that have been widely used in the field are genetic algorithms and genetic programming.

Genetic algorithms (GAs) were introduced by John Holland to abstract the adaptive process of natural system. The result is an artificial software system that retains the important mechanism of a natural system[26].

Genetic programming (GP) was introduced by John R. Koza [27,28,29] as an extension to GAs. Instead of representing a problem function as a chromosome string, GP represents it as a tree structure or as a computer program. GP and GAs process several problem functions or *individuals* at a time. The first generation is generated from a start-up structure using genetic operations. Individuals are then sent for the *fitness measurement*, where they are evaluated by their performance under a desired environment. The evaluation results are called *fitness values*. The fitness values are then used for the *natural selection* for the evolution of the following generation. Genetic operations commonly used by GAs and GP for the evolution of each generation are *crossover*, *mutation* and *reproduction*. Detail of GAs can be found in [26] and GP in [27,28,29,30].

GAs and GP have been applied for evolving analogue circuits in various approaches, as follows:

- *Parameter refinement:* In this approach, genetic algorithms are used to search for component parameters in a fixed topology. The system by Wo'jcikowski et al. [31] used genetic algorithms to search for transistor length, bias voltage and current load of operational amplifiers. A similar system by Zebulum et al. [32] applied genetic algorithms to search for transistor length, compensating capacitance and bias current. GAs were also used for the evolution of other types of circuits. Arslan and Horrocks applied GAs to evolve digital and analogue filters[33]. Horrocks and Kalifa[34] used GAs to optimise capacitor and inductors in LC circuits.
- *Topology search:* Genetic algorithms are used to invent a network. The system by Grimbly [35] constructed a network comprising resistors and capacitors from a start-up network built around an amplifier to perform as active filter. The parameters of transistors and capacitors were sought by numerical optimisation. Thompson et al. [36] applied GAs to evolve various types of circuits on FPGA chips and commercial analogue cross-point switch arrays.

- *Topology selection/invention and parameter optimisation:* Genetic algorithms and genetic programming were used for evolving circuit topologies and, within the same process, searching the component parameters. Early implementation was for the selection of circuit topologies. DARWIN[20] generates the first generation by randomly selecting from 24 possible topologies in the library. The evolutionary process gradually eliminates topologies which are not suitable for the problem while, at the same time, searching for component values. Later, the idea to represent the components and how to connect them together as a circuit was introduced. Such an idea was used in the system by Lohn et al.[22] and the WYWIWYG system [21].

The automated WYWIWYG design process is based on genetic programming. It has successfully evolved various types of circuits including passive filters [21], 60dB and 96 dB operational amplifiers [37,38]. The WYWIWYG system evolves circuits from an *embryonic circuit*, which can be considered as a start-up circuit comprising only simple wires. These wires are then evolved into complex connections of wires as well as components. Each generation contains a large group of *individuals* representing evolving circuits. Circuits are compiled into circuit netlists, which are simulated for their performance and evaluated as *fitness values*. New generations are then evolved from previous generations using the genetic operations comprising *crossover, mutation, and reproduction* based on *fitness values* of parent individuals.

The system by Lohn and Colombano is similar to the WYWIWYG system, but it is based on genetic algorithms. The system uses a variable sized representation called *cc-bot instructions*. Each construction is a command describing a new component and how to place the component to adjacent components. The system has successfully evolved a number of passive filters as well as operational amplifiers.

C. Shortcomings of the existing systems for topology synthesis

The canonical concepts which use a knowledge base to synthesis circuit topologies and then optimise circuit parameters have been proved to be a success. Several techniques for circuit optimisation have been intensively researched and some of them have been used commercially. The techniques for topology synthesis have been focused on the use of a knowledge base by means of design rules and building blocks representing subcircuits. These techniques are guaranteed success under the condition that relevant rules and building blocks must be given. The concept of evolving circuit topologies from a collection of primitive devices is designed to provide results regardless of relevant knowledge. However, it also brings more shortcomings:

The systems require large computational resources. Because genetic algorithms and genetic programming involve the evolution of a large amount of circuits, and also the simulation for each evolving circuit. This requires long computational time. To reduce the time used by systems, the use of a parallel computer is preferred. The WYWIWYG system evolved 60dB operational amplifiers on a parallel computer with the population size of 640,000[37]. While the system by Lohn et al. evolved circuits using a group of Sun workstations or Intel-based PCs, with various population sizes from 1000 to 18000 depending on circuit types[22].

Resulting circuits from both systems are ill-designed. Both systems perform well when evolving passive filters.

However, the evolution of operational amplifiers is not so good because the results are limited by the

connections that a transistor can make. For the WYWIWYG system, the third terminal of a transistor can be connected to one of two remaining terminals, a global circuit node, or a power supply, while the system by Lohn and Colombano limits transistor connections to be connected to the adjacent components in the circuit representation structure, to the ground, input and output connection. However, the design of CMOS circuits which comprise several transistors, can be a problem because of complexity. Furthermore, both systems lack algorithms to eliminate the redundancy and unconventional connectivity. Although the WYWIWYG system possesses a simplifying algorithm that removes short-circuits or floating components from evolving circuits, such an algorithm does not eliminate unconventional circuit structures but only ensures that the evolving circuits can be simulated without error. Resulting amplifier circuits evolved by the WYWIWYG system are still full of unconventional connectivity of transistors as well as huge redundancy [37,38]. The system by Lohn and Colombano is based upon a desired property that all sets of circuit-constructing primitives result in valid circuit graphs, so the correction procedure is not required. However, several transistors in the resulting circuits still show unconventional connections [22].

III. THE PROPOSED SYSTEM

To overcome such disadvantages, the proposed system is based upon the genetic programming paradigm, with an extension for design correction. The population size can be expanded or reduced when needed to minimise unnecessary computation. A user-defined library containing building blocks is also available, so the mutation can use alternative choices for the evolution. The redundancy and connectivity are controlled by a novel design correction procedure, which we call *current-flow analysis*. We also introduce a new circuit representation to aid the complexity of the circuit structure. It allows transistors to be connected to most of the circuit nodes within the whole circuit with a single evolving step. Details of the system are shown in figure 1.

To design a circuit, the system requires a number of start-up configurations from the user, which are circuit specifications, technological parameters and embryonic circuits. The user can also adjust the system parameters and insert new building blocks to suit the requirement. Circuits in the first generation are evolved from an embryonic circuit (start-up circuit) using mutation, and then sent through the *current-flow analysis*. Current-flow analysis creates current-flow lists for each circuit, which describe the connection of components based upon the current flowing within the circuit. Correction rules are performed on the current-flow lists to alter the connection of components, remove isolated parts or any parts which do not affect performance. The remaining circuits are then evaluated for fitness values. A number of candidates are then selected from evaluated circuits for refining performance. Improved candidates are then inserted back into the generation, and sorted. If no circuit meets the target specifications, new generations are then evolved from the current generation. The process runs until all necessary specifications are satisfied.

A. The circuit representation technique based on current-flow analysis

The new circuit representation is similar to the one used in the WYWIWYG system [21,37,38], in that the tree structure representing a circuit, which we will call a *GP-tree representation*, is the construction of *evolving functions* which evolve a single wire into a complex structure of components. However, the new circuit representation also allows transistors to be connected to most locations within the whole circuit via *cross-linked connections* which will

be described later. A connection modifying function, where by a wire is evolved which connects two circuit nodes together, e.g. the so-called *VIA function* in the WYWIWYG system, is also presented in the proposed method but with an improvement. The comparison in connection modifying functions used in the WYWIWYG system and in the proposed system is shown in figure 2.

In the figure, the topological evolution starts from an embryonic circuit, which generally contains a wire connecting input to output. The WYWIWYG system may evolve a single wire into a loop using a parallel connection modifying function. The VIA functions can be used to connect any part of circuit to the power supply, allowing active components to work. Transistors may receive current from such a connection, or it may connect its terminal to the power supply directly. The WYWIWYG system defines a small number of global sharing nodes, which can be used by any components.

When complex loops are required, a single wire grows into several loops using several parallel connection modifying functions placed close together, or using two VIA functions that connect to the same circuit node that is predefined and available for all components.

However, when several transistors are presented in a circuit, more global sharing nodes are required by VIA functions and transistor creating functions to represent more complex network structures. In the WYWIWYG system, the global sharing nodes are limited. Furthermore, to connect a transistor terminal to a target node in the circuit via the global sharing node, the WYWIWYG system needs at least two evolutionary steps, firstly connecting transistors to a global node, and secondly, connecting the target location to the sharing global node using a VIA function. Evolving a connection between two nodes from any location in the circuit requires a similar task. In a later publication, the way to perform such a task was suggested, and required several evolutionary steps [39].

The proposed representation technique is not only capable of performing in a similar manner to the WYWIWYG system, but it also automatically defines a new referencing node when evolving every serial connection modifying function. In addition to the circuit connections between components which are placed adjacent to each other in a GP-tree representation, transistors are allowed to connect one of their terminals to the referencing nodes, power supplies, or input node of circuit. With the use of VIA functions, any circuit node can be connected to one of the referencing nodes, power supplies, or input node as well. To identify the connections made to referencing nodes defined by serial connection modifying functions, we will refer to them as *cross-linked connections*. Basic representation of two-terminal components and three-terminal components are shown in figure 3.

From the figure, the two-terminal component creating function comprises only one GP node. These components are wire (Z), floating connection (FLOAT), resistor (R), capacitor (C), inductor (L), voltage independence source (V), and current independent source (I).

The three-terminal component creating functions are more complicated. They comprise a GP node containing the component type and component parameters, accompanied by two optional subtrees. There are two available components in this group, which are N-type and P-type transistors.

The connection modifying functions are evolving functions that evolve new connections, wires, or several components from a single component or a wire. They are parallel (PSS), serial (SER), cross-linked (VIA), and connection swapping (FLIP) functions, shown in figure 4. The SER function also defines a referencing circuit node which will be used as a target connection for VIA functions and transistor creating functions.

The proposed hierarchical circuit representation is based upon current-flow lists. An individual contains several current-flow lists connected together as a single GP-tree. Each subtree represents a current-flow list. Some GP-nodes may contain subtrees and this allows hierarchical structures to be identified.

Figure 5 demonstrates how a single current-flow list comprising four components can be represented as a GP-tree. The example list has three two-terminal components and one three-terminal component (with no optional subtree). They are connected together using three SER connection modifying functions. Figure 6 is an example of an individual, which comprises several subtrees. The main tree is considered as the *root hierarchy*. Other subtrees are connected via connection modifying functions or three-terminal component creating functions and are considered as the *lower hierarchy*. The cross-linked connections made by three-terminal component creating functions and VIA connection modifying functions can freely connect to the global sharing node of any SER connection modifying functions, or global nodes which include VDD, VSS, and Vin.

B. Current-flow analysis

Current-flow analysis is a technique to analyse and model a circuit in the form of *current-flow lists*. It is designed especially for circuits involving transistors. The algorithm to create current-flow lists from a circuit represented in GP-tree individual is shown in figure 7.

The current-flow analysis task firstly translates a GP-tree representing an evolving circuit, into current-flow lists. Each current-flow list may contain a single component or several components connected together in the way that it allows an amount of DC current to flow through all components. A component, which does not allow continuous DC current to flow such as a capacitor, is considered as a separated list.

A circuit may be represented as a set of current-flow lists as following:

$$Circuit(idv) = \{L_1(n_{1a}, n_{1b}), L_2(n_{2a}, n_{2b}), \dots, L_n(n_{na}, n_{nb})\} \quad (1)$$

Whereas a = 0 and b = number of components in each L_i , while $L_i \in \{L_1, \dots, L_n\}$.

idv represents the description of the circuit in a population. Each circuit contains n current-flow lists. $L_i(n_{ia}, n_{ib})$ is a current-flow list, which may comprise single component or several components connected together. A particular L_i , which contains j components, may be described as:

$$L_i(n_{i0}, n_{ij}) = \{C_1(n_{i0}, n_{i1}), C_2(n_{i1}, n_{i2}), \dots, C_j(n_{i,j-1}, n_{ij})\} \quad (2)$$

$L_i(n_{i0}, n_{ij})$ possesses two main circuit nodes, n_{i0} and n_{ij} , at the ends of the list. A current-flow list allows an amount of current to flow through all components in the chain via n_{i0} and n_{ij} . $C_j(n_a, n_b)$ represents a two-terminal component. For P-type and N-type transistors, $C_j(n_a, n_b)$ is substituted by $C_j(n_a, n_b, n_c)$ where n_b notifies the connection of the gate terminal, n_a and n_c are the connections of drain and source terminals.

To demonstrate how the current-flow analysis transforms a circuit into the current-flow lists, a two-stage amplifier shown in figure 8 can be transformed as:

$$Circuit(idv) = \{L_1, L_2, L_3, L_4, L_5\} \quad (3)$$

Where L_1 to L_5 are

$$\begin{aligned} L_1(VDD, VSS) &= \{I_1(VDD, 7), NMOS_{m6}(7, 7, VSS)\} \\ L_2(VDD, VSS) &= \{PMOS_{m5}(VDD, 3, Vout), NMOS_{m8}(Vout, 7, VSS)\} \\ L_3(VDD, 5) &= \{PMOS_{m4}(VDD, 4, 3), NMOS_{m1}(3, Vin+, 5)\} \\ L_4(VDD, 5) &= \{PMOS_{m3}(VDD, 4, 4), NMOS_{m2}(4, Vin-, 5)\} \end{aligned}$$

$$L_5(5, VSS) = \{NMOS_{m7}(5, 7, VSS)\} \quad (4)$$

Once the current-flow lists are created, a number of analyses can be done and a certain action can be performed. These are possible characteristics, which can be obtained from the current-flow lists for the design correction purpose:

1) *Rule 1. Direction of current:* the direction of current could be sought when at least one of the ends of the current-flow list is VDD or VSS. From equation (1) and (2) the direction of current which flows through $L_i(n_{ia}, n_{ib})$ can be found as:

For each list $L_i(n_{ia}, n_{ib}); L_i \in \{L_1, \dots, L_n\}$

if ($n_{ia} = n_{ib}$)

 No current flows between a and b

else if ($(n_{ia} = VDD)$ or ($n_{ib} = VSS$))

 Direction of current is from n_{ia} to n_{ib}

else if ($(n_{ia} = VSS)$ or ($n_{ib} = VDD$))

 Direction of current is from n_{ib} to n_{ia}

else

 Direction of current is unknown

where n_{ia} and n_{ib} are the circuit nodes at the ends of the current-flow list.

This rule identifies two interesting situations. The first one is the situation that an amount of current flows between n_{ia} and n_{ib} . Another situation is the existence of a current-flow list without any current flowing. This means that all circuit nodes in the current pathway from n_{ia} to n_{ib} have the same voltage. Thus, such a current-flow list can be removed. For example, in figure 9 and equation (4), all current-flow lists connect to at least one of the ends of the lists to VDD or VSS. Thus, the direction of current for all current-flow lists can be obtained and it is from the left hand side of the lists, to the right hand side of the lists.

2) *Rule 2. Floating list:* The floating list is the current-flow list that contains a floating component. No current flows through the current-flow list. In these circumstances, the current-flow list can be removed. The floating lists can be determined as follows:

For each list $L_i(n_{ia}, n_{ib}); L_i \in \{L_1, \dots, L_n\}$

if ($n_{ia} \notin \{VDD, VSS, n_{0a}, n_{0b}, \dots, n_{(i-1)a}, n_{(i-1)b}, n_{(i+1)a}, n_{(i+1)b}, \dots, n_{na}, n_{nb}\}$)

or ($n_{ib} \notin \{VDD, VSS, n_{0a}, n_{0b}, \dots, n_{(i-1)a}, n_{(i-1)b}, n_{(i+1)a}, n_{(i+1)b}, \dots, n_{na}, n_{nb}\}$))

L_i is a floating list.

3) *Rule 3. Isolated list:* The isolated list contains components, which can be removed without any change in DC characteristics. The checking task starts from the current-flow list that contains the output circuit node. Then other current-flow lists are checked to see if they share any circuit connection to the current-flow list being considered, except the connection being made by gate terminal of transistors in other current-flow lists, and the connection to the power supplies. The checking task performs repeatedly until no more current-flow lists are selected. The current-

flow lists that are left out from the process can be considered as isolated lists, and all components within the lists could be removed without any effect on the whole circuit.

4) *Rule 4. The transistor region:* Once the direction of current is known for the current-flow list that hosts the particular transistor, the direction of current that flows via the transistor can also be identified and its region can be obtained as follows:

For each transistor $C_k(n_{i(k-1)}, ng_k, n_{ik})$ in $L_i(n_{ia}, n_{ib})$ with known current direction $n_{i(k-1)} \rightarrow n_{ik}$ or $n_{ik} \rightarrow n_{i(k-1)}$

if(($C_k=PMOS$) and ($n_{i(k-1)} \rightarrow n_{ik}$)) or (($C_k=NMOS$) and ($n_{ik} \rightarrow n_{i(k-1)}$))

if ($ng_k \in \{n_{i0}, \dots, n_{i(k-1)}\}$)

C_k is in cut-off region

if($ng_k = n_{ik}$)

C_k is in saturation region

if ($ng_k \in \{n_{i(k+1)}, \dots, n_{ib}\}$)

C_k is in saturation or triode region

if(($C_k=NMOS$) and ($n_{i(k-1)} \rightarrow n_{ik}$)) or (($C_k=PMOS$) and ($n_{ik} \rightarrow n_{i(k-1)}$))

if ($ng_k \in \{n_{ik}, \dots, n_{ib}\}$)

C_k is in cut-off region

if($ng_k = n_{i(k-1)}$)

C_k is in saturation region

if ($ng_k \in \{n_{ia}, \dots, n_{i(k-2)}\}$)

C_k is in saturation or triode region

in other cases that $ng_k \notin \{n_{ia}, \dots, n_{ib}\}$

transistor region of C_k is unknown

If L_i shares circuit node at the end of the list with another current-flow list and both lists connect the circuit node at the other end of the list to the opposite power supply, all other circuit nodes in those lists are also taken into account. For instance, when considering transistor m4 in the circuit shown in figure 8 with current-flow lists in equation (4), current-flow list L_5 is also considered because L_3 that hosts transistor m4 shares a current node at the end of the list with L_5 , and another end of L_3 is connected to VDD, while another end of L_5 is connected to VSS. The current-flow list L_4 is not considered for transistor m4 because the end of the list L_4 is connected to VDD although it also shares another end of the list with L_3 and L_5 .

When the transistor region is known, a transistor that performs in the wrong region can be reconnected or removed. Again, from figure 8 and equation (4), transistor m6 can easily be identified to perform in the saturation region, because its gate transistor is connected to the circuit node defined within the same current-flow list that hosts the transistor as explained above.

The current-flow analysis is a simple and short task, but, at the same time, is useful for the analysis of circuits evolved under evolutionary algorithms. In other automated circuit design systems based on a knowledge base, the correction procedure can be easily performed with accuracy because the knowledge base provides the information of the correction procedure together with the architecture or building blocks that the system is using to design for the particular specifications. But in the systems which are based on evolutionary algorithms with the facility of topological invention, such information is not available, the current-flow analysis is therefore an alternative procedure to identify and correct any faulty design, since the current-flow analysis is based upon fundamental Kirchhoff's Laws. The result can be used to guide the correction and refinement of the circuit, or to screen the obscure design so that circuits containing only non-useful components will not spend the valuable simulation time.

C. Evolutionary process

The main evolutionary process of the proposed system is similar to the process used in genetic programming. Its genetic operations are *reproduction*, *crossover*, and *mutation*. The reproduction is the same as canonical genetic programming. The crossover and mutation are a little more complicated, as the proposed representation contains cross-linked connections. The crossover is done by selecting the swapping point for both parents. Both parents are cut at the swapping locations, along with any lower hierarchical subtrees connected underneath. This causes all cross-linked connections made across the swapping point to be disconnected. The sections are then swapped between the parents. The mutation operation performs by selecting a point within the parent. All GP-nodes below the selected point are removed and new GP-nodes are evolved using randomly selected evolving functions. Again, the cross-linked connections made between GP-nodes under the cutting point and above the cutting point are disconnected. Thus, the crossover and mutation operations require the final procedure to reconnect all disrupted cross-linked connections by randomly selecting new locations from available referencing nodes and other global circuit nodes, including input circuit nodes and power supplies.

In the proposed system, we use tournament selection as the algorithm for natural selection of parents. The tournament selection algorithm selects a few individuals randomly. The selected individuals are then compared to each other by their fitness values. The best individual will be used as a parent.

D. Fitness measurement

The fitness values are divided into two groups, which are *hard constraints* and *soft constraints*. The fitness measurement process, which evaluates all fitness values, are shown in figure 9.

The hard constraints are the user specifications for the desired circuit. The constraints can be a single boundary, which the acceptable performance is either under of above, or presented as desired range, with the acceptable performance falling within this range. The hard constraints are required to meet the target for the evolution to be considered a success.

The soft constraints comprise some user specifications and other program configurations. The soft constraints are considered as lower priority than hard constraints. The soft constraints are not required to meet the target. However, after all hard constraints have met the target, the evolution can perform further to improve the fitness values controlled by the soft constraints and this provides better optimised circuits depending upon particular soft constraints.

For the evolution of CMOS amplifier used in the proposed system. The hard constraints include:

1. **Open-loop gain:** This constraint has a lower boundary to notify the minimum value of acceptable gain.
2. **Gain bandwidth product:** This constraint also has a lower boundary to notify the minimum value of acceptable GBP.
3. **Phase margin:** This constraint has a lower boundary to notify the minimum value of phase margin for the whole range of acceptable GBP.
4. **DC-offset:** This constraint has lower and higher boundaries. It notifies the acceptable range of DC offset.
5. **Power dissipation:** This constraint has a higher boundary to notify that the power dissipation at quiescent point should not be beyond the boundary.
6. **Voltage swing:** This constraint possesses lower and higher boundaries to notify the minimum range of acceptable output voltage range that the circuit can perform is linear.
7. **Linearity penalty:** This constraint has a higher boundary point. It notifies the possible highest value of the differential of the lowest gain point and the highest gain point within acceptable voltage swing. The example of DC sweep in figure 10 shows how to notify the highest gain point and the lowest gain point.

All hard constraints except the voltage swing are normalised into a general scale, where the boundaries can be eliminated. Only the normalised fitness values that are greater than 0 (except voltage swing) are summed together as an *overall fitness value*, and will be used as the highest priority for the ranking process.

The soft constraints contain the general parameters apart from desired circuit specifications. As the benefit of optimising these constraints is mostly for economic reasons rather than requirement. The soft constraints for the evolution of CMOS circuits are:

1. **CMOS transistors violation penalty:** This is due to transistors that perform in other regions than the saturation region. This constraint could be considered as a hard constraint as it is practical for CMOS amplifiers to have only transistors performing in the saturation region. However, from several trails, it was found that the evolution process often starts evolving circuits which contain a number of transistors in the wrong region. The evolution will approach a situation where fewer transistors will be kept in the wrong region.
2. **Diffitness:** The *differential overall fitness value*, or in short, *diffitness*, is sought from the differentiation of overall fitness value of a circuit, to the overall fitness value of its better parent. The positive value identifies that the circuit performs better than its parents, the negative value identifies otherwise.
3. **Complexity:** This constraint is sought from the component parameters. Each component type has a particular weight and function for the calculation. The larger size of components and more components mean the higher complexity.

The overall fitness value and other soft constraints are used to evaluate the rank of each individual. The priority level is ordered from the overall fitness value, diffitness, CMOS violation, and complexity respectively. The rank of each individual is used for the *tournament selection* of parents for the evolution of the next generation.

Please note that the normalised fitness value is valid from $-\infty$ to 100. The highest possible value, 100, notifies the worse performance for a particular specification. The value greater than 100 notifies the failure in fitness

evaluation. The acceptable value is 0. The negative value identifies the *over-specification* situation; e.g. the circuit outperforms the desired specification.

The overall fitness value also provides another benefit. It allows all hard-constraints with over-specification not to influence the rank position, while soft constraints can still influence the rank. For the design of amplifiers, the complexity penalty is calculated from component parameters, which also affects circuit performance. For instance, transistor size has a direct effect on transistor gain. The reduction in transistor size means the reduction of the complexity penalty, it also means the transistor gain would decrease, and thus, it could decrease the fitness value which governs the amplifier gain. Thus, this technique is relevant to design *trade-offs*.

E. Variable population size and user-defined library

A problem which arises when using genetic algorithms and genetic programming is how to obtain the proper population size with the minimum use of computational resource, and at the same time, to guarantee a successful result. In [27], there is a procedure to determine the possibility of success in searching for an answer using genetic programming. Unfortunately, the procedure requires a number of runs to provide a near-solved solution of various population sizes for the determination of the proper population size. For a general design when the specifications can be varied, the best action is to use the largest possible population size to avoid the *premature convergence*; e.g. the situation when the evolution stops before reaching its target.

In general, the best way to reduce the risk of such a condition is to use as large a population size as possible. However, this means longer computational time. A solution to reduce computational time is to skip the fitness measurement procedure for a few first generations as in [38].

In the proposed system, we introduce another solution to reduce computational time by using a small or moderate population size at the first generation. In such a case that premature convergence may occur while evolving, the population size will be expanded to allow more circuits to be evolved for each generation. This promotes more diversity in the population. We determine the premature convergence by checking the consistency in the fitness values of the best circuit and the appearance of several circuits that are the same structure as the best circuit over several generations. If the evolution improves over several contiguous generations, the population size may reduce but not smaller than the starting size.

The user-defined library is another simple but useful technique. Knowledge is provided by the users by adding known basic modules as additional information for the evolution. For instance, a simple gain cell comprising P-type and N-type transistors is useful for evolving amplifiers. Basic modules that we use for the evolution of CMOS amplifiers are shown in figure 11. The mutation process randomly selects a module from the user-defined library and inserts it into an evolving circuit in the same way as it evolves components. Thus, the use of basic modules is also controlled by the natural selection as with other components.

F. Candidate selection and refinement

Although the current-flow analysis is performed before simulation, there is no change within the circuit representation of each individual. This allows unused information to remain in the individual and could be evolved further into useful components in later generations. However, in most cases, this unused information can propagate when the individual grows more complex, and unfortunately this prevents the possibility of selecting proper genetic information from evolving into useful components.

To allow the smaller individual a better chance to thrive, a small group of *candidates* is selected from the whole population and processed by current-flow analysis. A simple improving task is to apply current-flow analysis to notify unused information and remove it if possible. The results are then regenerated and inserted into the population.

In the proposed system, we demonstrate the use of a simple hill-climbing algorithm to refine component parameters of candidates. The process changes one of component parameters at a time. It then evaluates the fitness value and compares the altered circuit with the original and selects the better one. The evaluation procedure is the same module to the one used by genetic programming. However, we consider the altered circuit to be better than the original under a condition where the CMOS violation penalty is considered as higher priority than the overall fitness value.

The candidates are selected using the following process:

- 1) Select a candidate whose CMOS violation penalty is lower (better) than the best individual,
- 2) Select a candidate whose design is valid but can not pass the simulation, and
- 3) Select remaining candidates from the whole population using the tournament selection procedure.

IV. THE EXPERIMENT

A. Configuration settings

The aim of the experiment is to demonstrate the use of current-flow analysis for the design of CMOS amplifiers evolved using genetic programming. Currently, we decided to evolve amplifiers with one input and one output as in previous work [22,37,38]. The system was written in C and ran on a Pentium-II 400 MHz PC. The operating system was Linux and the circuit simulator was PSPICE3. The minimum population size was 300 and maximum size was 1000 with expansion/reduction rate at 5 percent. Five candidates were selected from each generation for parameter refinement using hill climbing as previously described. The mutation and crossover were 50 percent chance. The limitation for the serial component was set to 4 for subtrees at lower hierarchy and 32 for the main tree. Power supplies used by the testing harness were -5 and 5 volt unless stated otherwise. Most runs used a general set of specifications, which are shown in table 1.

B. Results

The system was thoroughly tested with several runs. However, under such a tight configuration for the evolutionary process, several runs could not reproduce results and so they were discarded. Some runs satisfied the desired specifications but unfortunately the number of transistors operating in the invalid region did not decrease over several generations.

The referencing circuit shown in figure 14 was evolved using a set of evolving functions and a genetic representation technique which are similar to the WYWIWYG system. Design optimisation technique, GCL and DPS are disabled. The population size was 10000. The circuit was difficult for human designer to understand and such a condition can also be found in resulting circuit shown in [37].

When the proposed genetic representation technique and current-flow analysis were implemented, the resulting circuits evolved by the system were significantly better. The resulting circuits in figure 15 and 16 were from two

runs with the phase margin penalty not restricted. They evolved from an embryonic circuit shown in figure 12. The testing harness for the simulation is shown in figure 13.

The resulting circuits in figure 17 and 18 were from another two runs where the transistor length was set to $10\mu\text{m}$ instead of $5\mu\text{m}$. This demonstrates how the proposed system adapts itself to a new configuration where the change could directly affect gain of each transistor.

When phase margin penalty were also used for fitness evaluation, the proposed system adjusted the evolution by automatically applying frequency compensation if necessary, as the resulting circuit shown in figure 19. The resulting circuit comprises 3 amplification stages and a compensating subcircuit which comprises a capacitor and a low-gain amplification stage which also performs as a biasing circuit for the first and final stage.

Figure 23, 24 and 25 show resulting circuits of various runs that different specifications were applied. The resulting circuits shown in figure 23 and 24 was evolved using the target gain of 90dB and 120 dB respectively, while the resulting circuits shown in figure 25 uses different supply voltages from the rest, which are 3 and -3 volt. The output voltage swing was reduced to ± 2 volt and amplification gain was 60dB.

The result in figure 28 was evolved from an embryonic circuit shown in figure 26. This demonstrates how the proposed system reacts with the predefined topologies that are used as embryonic circuits. A similar condition has been previously implemented by DARWIN[20]. And the last result in figure 29 was also evolved from the same predefined topology but the transistor length was set to $3\mu\text{m}$ and amplification gain 75dB.

Under the user specifications shown in table 1, the proposed system under went around 20 to 100 generations before all hard constraints were satisfied. The system then proceeded with further optimisation of the soft constraints, e.g. the number of transistors operating in the wrong region. Table 2 shows the total number of individuals and generations that each resulting circuit was used to evolve from an embryonic circuit to a status that all hard constraints were satisfied. The referencing circuit requires significantly more computational resources than other circuits which were evolved under the proposed methodology. However, evolving circuits using the same set of specifications require uncertain amount of resources, as the comparison of total number of individuals required to evolve circuits in figure 15 and 16, and the similar condition for circuits in figure 17 and 18. Another noticeable condition comes from the evolution of 60dB, 90dB and 120dB of circuit shown in figure 19, 23 and 24. According to the total number of individuals shown in table 2, evolving higher amplification gain requires more computational resources. The average time for the runs except the run that produced referencing circuit was about two to three days for each successful run on a Pentium-II 400 MHz PC with the total generations around 100 to 250. Failure can be identified within a day by checking to see if the overall fitness value of the best circuit could not improve over several generations.

C. Discussion

According to the results and the proposed techniques, there are various issues to be raised:

- 1) *Resulting circuits in comparison to other similar systems:* The experimental results have shown that the proposed techniques can actually improve the design when compared to the results produced by the WYWIWYG system [21] and the system by Lohn et al.[22]. Results in [21] contain redundancy and unconventional transistor connections. Results in [22] shows that the system could eliminate redundancy without the need for design correction algorithms. However, several transistors are connected improperly.

The proposed systems produced resulting circuits which not had only redundancy eliminated, but the transistor connections were also correct.

- 2) *The variation in the resulting circuits:* The experimental results show that the proposed system produces various circuit topologies for the same configuration setting. This is different from systems based upon a knowledge base which generally produce consistent results. In the knowledge based technique, user specifications are used to identify which topology will be used or how a topology is synthesised by design rules. While the proposed system uses genetic programming to provide circuit topologies by evolving them from a collection of components. This allows a higher degree of freedom for topology invention. Currently, the design optimisation algorithm is designed to correct circuits. In the future research, the facility for design optimisation using small-signal models of transistors can be done. This could reduce the variation of design by replacing a complex structure with an equivalent one.
- 3) *Strategy for improving the design:* Although current-flow analysis provides information which is useful for the design optimisation purpose, it is possible that such information may not be known for some components, e.g. the transistor region may not be known for some transistors. Thus, no correction will be done for components where there is inefficient design information. After a circuit has been sent for simulation, the transistor region can now be obtained and used for the evaluation of the CMOS violation penalty. The purpose of current-flow analysis is therefore to correct the design using as much information as it can gather before the simulation. This would decrease the search space used by the genetic programming.
- 4) *The use of the refining process:* The refining process of the candidates is designed to improve potential individuals by means of their performance by searching within the local neighbourhood. The move to a new point within the search space should be done carefully not to strongly perturb the move of the whole population. Currently, we use the refining process as component parameter refinement for the trade-offs between the hard constraints and complexity penalty, while the CMOS violation penalty will not be traded with other penalties unless the action causes a better move for the CMOS violation penalty. From the experiment, we have also found a situation where the design is valid but the circuit can not function properly. In this case, the use of parameter refinement that allows all parameters to be randomly reset could salvage such design.
- 5) *The use of dynamic population sizing:* The purpose of dynamic population sizing is to reduce the time for evolution in the few first generations. Another purpose is to allow the population size to be expanded when the users set the population size too small. And finally, to allow the population size to be expanded if there is no improvement after several generations. The larger population size would promote more diversity and hence increase the chance that evolution would jump over the local maxima. The population size may then be decreased to the previous size to limit the computational time.
- 6) *Strategy for reducing the size of search space:* The evolutionary process that invents circuit topologies and optimises circuit parameters is actually a search algorithm that performs the searches within a search space. The size of the search space for a circuit design problem depends on number of possible component values for each component, number of components, and available component connections. The number of possible component values depends on the move step of each component parameter during optimisation and its acceptable range. A smaller step means more accuracy but also a larger search space. An application can be found in [34] where a small number of preferred component values is introduced to reduce the number of

available component values and thus reduces the search space. The total number of component for an evolving circuit is indirectly controlled by limiting the growth of the circuit representation structure. While the connection of components is controlled by basic rules presented in circuit representation technique and the current-flow analysis. The basic component connection rules affect the search space used by the evolutionary process. In contrast, since the current-flow analysis performs after the evolution of each generation, it affects the searches by selectively presenting valid choices within the whole search space. The result is the population size and number of generations can be decreased. The proposed system has successfully evolved 60dB CMOS amplifiers using the dynamic population size between 300 and 1000 individuals with the total generation around 100 to 250. In comparison, the WYWIWYG system generated a similar 60dB amplifier using the population size of 640,000 and total generation of 109 [37], while the system by Lohn et al. generated a 75dB amplifier using the population size of 1200 and total generation of 4866 [22]. However, when comparing to other techniques that apply manually designed topologies with topology-independent parameter optimisation techniques, the proposed system searches on a larger search space because the searches for proper constructs of circuit topologies are taken into account. The search space would increase further when evolving more complex circuits. Several techniques can be done to counterbalance the size of the search space, including improving circuit representation technique and topology correction process, providing predefined building blocks, using predefined circuit topologies as embryonic circuits, as a result shown in figure 28. In very complex circuit design cases, the proposed system may be applied to invent subcircuits within the hierarchical design process when the design system cannot provide a satisfying subcircuit to fit the required performances.

V. CONCLUSION

This paper introduces a system for the evolution of CMOS amplifiers. The system possesses a new circuit representation technique, which is designed to cope with the complexity of connections between transistors. It also possesses a novel current-flow analysis, a topology-independent design correction algorithm that corrects circuits evolved by the evolutionary process. The system also allows other optimisation techniques to be used along with genetic programming. A simple hill-climbing algorithm is used as a demonstration for this case. The experimental results have demonstrated the feasibility of an evolved design compatible to a human design with similar performance.

Future research is geared towards improving the circuit presentation technique and current-flow analysis algorithm to correct some transistors that are still in wrong region and speed up the design process. New fitness measurement modules will be added so other types of circuits can be investigated and tested. We believe that when such system becomes available, novice circuit designers will be one step closer to inventing basic analogue circuits.

ACKNOWLEDGEMENT

The authors would like to thank all reviewers for their invaluable suggestions regarding the paper. In particular the authors would like to thank Prof. J.R. Koza for extremely valuable comments which have helped to enormously improve paper.

REFERENCES

- [1] C. Toumazou and Costas A. Makris, *Analog IC design Automation: Part I- Automated Circuit Generation: New Concepts and Methods*, IEEE Transaction on CAD, Feb. 1994 Vol.14, No. 2, pp. 218-238.
- [2] R. Spence, C. Toumazou, P. Cheung, C. Makris, C. Berrah, M. Singha, and J. Xiao Xiangmin, *Approaches to Analogue IC Synthesis*, IEE Colloquium on VLSI Analogue Design, 1989, pp. 1/1 -1/6.
- [3] R. A. Rutenbar, *Analog Design Automation: Where are we? Where are we going?*, Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993 , 1993 , pp. 13.1.1 -13.1.7.
- [4] D.H. Horrocks, and Y.M.A. Khalifa, *Evolutionary Design of Analogue Electronic Circuits; Current status*, IEE Third one-day Colloquium on Analog Signal Processing (Digest No: 1996/236), 1996 , pp. 7/1 -7/8.
- [5] G.E. Gielen and R.A. Rutenbar, *Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits*, Proceeding of the IEEE, Volume 88, no.12, 2000, pp.1825-1852.
- [6] W. Nye, D.C. Riley, A. Sangiovanni-Vincentelli and A.L. Tits, *DELIGHT.SPICE: an optimization-based system for the design of integrated circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 7 4 , April 1988 , pp. 501 -519.
- [7] Shyu Jyuo-Min and A. Sangiovanni-Vincentelli, *ECSTASY: a new environment for IC design optimization*, IEEE International Conference on Computer-Aided Design, 1988. ICCAD-88. Digest of Technical Papers , 1988 , pp. 484 -487.
- [8] J.C. Lai, J.S. Kueng, H.J. Chen and F.J. Fernandez, *ADOPT-a CAD system for analog circuit design*, Proceedings of the IEEE Custom Integrated Circuits Conference, 1988., pp. 3.2/1 -3.2/4.
- [9] G.G.E. Gielen, H.C.C. Walcharts and W.M.C. Sansen, *Analog circuit design optimization based on symbolic simulation and simulated annealing*, IEEE Journal of Solid-State Circuits, volume: 25 3, 1990, pp. 707-713.
- [10] G.G.E. Gielen, H.C.C. Walcharts and W.M.C. Sansen, *ISAAC: a symbolic simulator for analog integrated circuits*, IEEE Journal of Solid-State Circuits, volume: 24 6, 1989, pp. 1587-1597.
- [11] E.S. Ochotta, R.A. Rutenbar and L.R. Carley, *Synthesis of high-performance analog circuits in ASTRX/OBLX*, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on Volume: 153, March 1996, pp. 273-294.
- [12] R. Phelps, M. Krasnicki, R.A. Rutenbar, L.R. Carley and J.R. Hellums, *ANACONDA: Robust Synthesis of Analog Circuit Via Stochastic Pattern Search*, IEEE Conference on Custom Integrated Circuits, 1999, pp. 26.3.1-26.3.4.
- [13] M.M. Hershenson, S.P. Boyd and T.H. Lee, *GPCAD: A Tool for CMOS Op-Amp Synthesis*, International Conference on Computer-Aided Design, 1998, pp.296-303.
- [14] J. Stoffels and C. van Reevwijk, *Ampdes: a program for the synthesis of high-performance amplifiers*, European Conference on Design Automation, 1992, pp.474-479.
- [15] F. El-Turky and E.E. Perry, *BLADES: an artificial intelligence approach to analog circuit design*, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, volume 86, 1989, pp.680-692.
- [16] A.H. Fung, D.J. Chen, Y.N. Li and B.J. Sheu, *Knowledge-based analog circuit synthesis with flexible architecture*, Computer Design: Proceedings of the IEEE International Conference on VLSI in Computers and Processors, 1988, pp. 48 -51.
- [17] C. Toumazou, C.A. Makris and C.M. Berrah, *ISAID - a methodology for automated analog IC design*, IEEE International Symposium on Circuit and Systems, volume 1, 1990, pp. 531-535.
- [18] C. Toumazou and C.A. Makris, *Analog IC design automation: Part I- Automated circuit generation: New concepts and methods*, IEEE Transaction on CAD, 1994, volume 14, no 2, pp.218-238.
- [19] C. A. Makris and C. Toumazou, *Analog design automation: Part II- Automated circuit correction by qualitative reasoning*, IEEE Transactions on CAD, Feb. 1995 volume 14, no. 2 pp. 239-254.
- [20] W. Kruiskamp and D. Leenaerts, *DARWIN: CMOS opamp synthesis by means of a genetic algorithm*, Proceeding of the 23rd Design Automation Conference: Association for Computing Machinery, 1995, pp.433-438
- [21] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, *Automated WYWIWYG Design of Both the Topology and Component Values of Electrical Circuits Using Genetic Programming*, Proceeding of the First Annual Conference, July 28-31, Stanford University, 1996.
- [22] J.D. Lohn, and S.P. Colombano, *A circuit representation technique for automated circuit design*, IEEE Transactions on Evolutionary Computation, Volume: 3 3 , Sept. 1999 , pp. 205 -219.
- [23] H.Y. Koh, C.H. Sequin and P.R. Gray, *OPASYN: a compiler for CMOS operational amplifiers*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, volume: 9 2 , Feb. 1990 , pp. 113 -125.
- [24] J. Jongsma, C. Meixenberger, B. Goffart, J. Litsios, M. Pierre, S. Seda, G. Di Domenico, P. Deck, L. Menevaut and M. Degrauwe, *An open design tool for analog circuits*, IEEE International Symposium on Circuits and Systems, volume 4, 1991, pp.2000-2003.

- [25] R. Harjani, R.A. Rutenbar and L.R. Carley, *Analog circuit synthesis and exploration in OASYS*, Proceeding of the 1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1988, pp.44-47.
- [26] D.E. Goldberg, *Genetic algorithms in search, optimisation & machine learning*, Addison-Wesley Publishing Company, Inc, 1989.
- [27] J.R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, MIT press, 1992.
- [28] J.R. Koza, *Genetic programming II: Automatic discovery of reusable programs*, The MIT press, 1994.
- [29] J.R. Koza, F.H. Bennett III, D. Andre and M.A. Keane, *Genetic programming III: Darwinian invention and problem solving*, Morgan Kaufmann Publishers Inc., 1999.
- [30] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, *Genetic programming: An introduction on the automatic evolution of computer programs and its applications*, Morgan Kaufmann Publishers Inc., 1998.
- [31] M. Wo'jcikowski, J. Glinianowicz and M. Bialko, *System for optimisation of electronic circuits using genetic algorithms*, Proceedings of the Third IEEE International Conference on Electronics, Circuits, and Systems, 1996, volume 1, pp.247-250.
- [32] R.S. Zebulum, M.A. Pacheco and M. Vellasco, *Synthesis of CMOS operational amplifiers through genetic algorithms*, Proceedings of XI Brazillian Symposium on Integrated Circuit Design, 1998, pp. 125-128.
- [33] T. Arslan and D.H. Horrocks, *The design of analogue and digital filters using genetic algorithms*, IEE 15th SARAGA Colloquium on Digital and Analogue Filters and Filtering Systems, 1995, pp.2/1-2/5.
- [34] D.H. Horrocks and Y.M.A. Khalifa, *Genetically derived filter circuits using preferred value components*, IEE Colloquium on Analogue Signal Processing, 1994, pp. 4/4-4/5.
- [35] J.B. Grimbleby, *Automatic synthesis of active electronic networks using genetic algorithms*, Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1997, pp.103-107.
- [36] A. Thompson, P. Layzell and R.S. Zebulum, *Exploration in design space: Unconventional electronics design through artificial evolution*, IEEE Transaction on Evolutionary Computation, Sep. 1999, volume 3, no 3, pp.167-196.
- [37] J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane, *Evolution of a Low-Distortion, Low-Bias 60 Decibel Op Amp with Good Frequency Generalization using Genetic Programming*, presented at the International Conference on Evolvable System: From Biology to Hardware (ICES-96), Tsukuba, Japan, October 1996.
- [38] J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane, *Design of a 96 Decibel Operational Amplifier and Other Problems for Which a Computer Program Evolved by Genetic Programming is Competitive with Human Performance*, presented at the 1996 Japan-China Joint International Workshop on Information Systems at the Ashikaga: Ashikaga Institute of Technology, Ashikaga, Japan.
- [39] D. Andre, F.H. Bennett, J.R. Koza and M.A. Keane, *On the theory of designing circuits using genetic programming and a minimum of domain knowledge*, Evolutionary Computation Proceeding, IEEE World Congress on Computational Intelligence, 1998.

Open loop gain	60dB
GBP	1MHz
Phase margin	60 degree minimum
Offset	± 0.025 volt
Power Dissipation (including biasing circuits)	100mW
Voltage swing	± 3 volt minimum
Linear penalty	10dB

Table 1. general specifications for the experiment

Circuit in figure#	special specifications	population size (min-max)	expanding/reducing rate (percent)	number of generations evolved	total individual
14	referencing circuit,60dB	10000	(constant size)	200	2000000
15	phase not restricted, 60dB,l=5 μ m	400-1000	5%	12	4800
16	phase not restricted, 60dB,l=5 μ m	400-1000	5%	25	10000
17	phase not restricted, 60dB,l=10 μ m	300-1000	5%	11	3300
18	phase not restricted, 60dB,l=10 μ m	300-1000	5%	43	11700
19	60dB,l=5 μ m	100-1000	10%	48	5577
23	90dB,l=5 μ m	300-1000	5%	64	19646
24	120dB,l=5 μ m	300-1000	5%	92	32888
25	60dB,l=5 μ m, VDD/VSS= ± 3 V	300-1000	5%	144	43764
28	60dB,l=5 μ m, good-started	300-1000	5%	7	2100
29	75dB,l=3 μ m, good-started	100-1000	10%	5	500

Table 2. number of generations and individuals used for the evolution of all runs

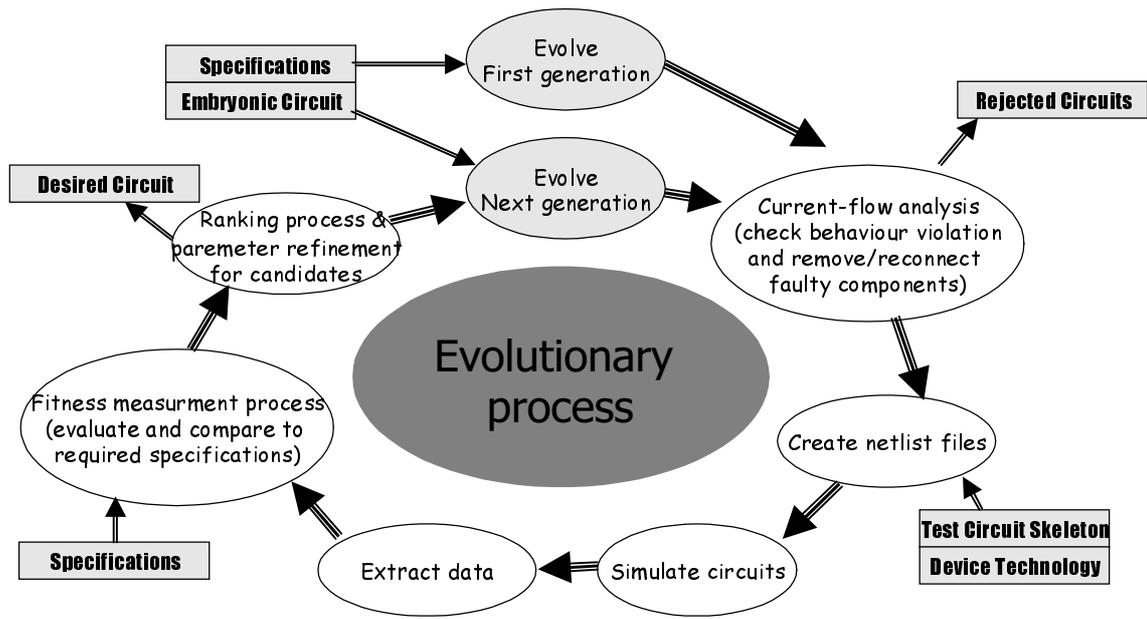


Figure 1. diagram of the proposed system

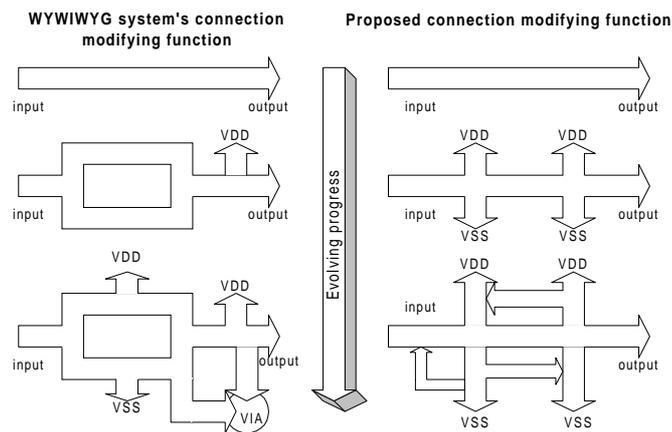


Figure 2 comparison of growth direction between the circuit representation used in the WYWIWYG system and the proposed system

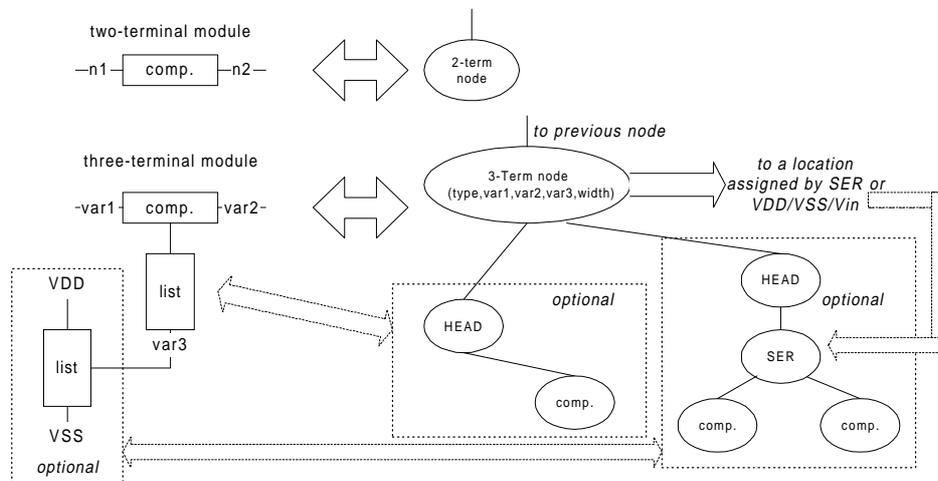


Figure 3 the component creating functions of two-terminal components and three-terminal components

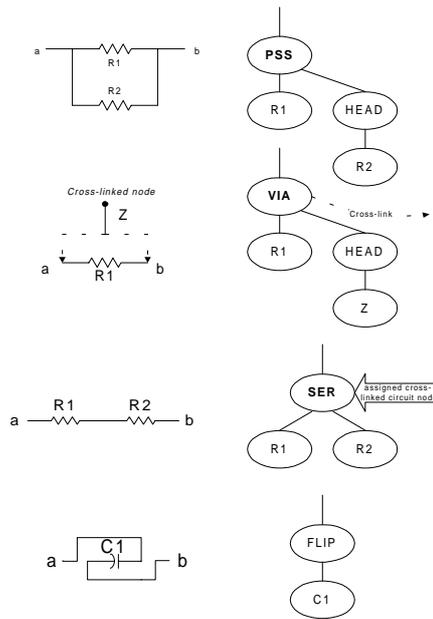


Figure 4 connection-modifying functions.

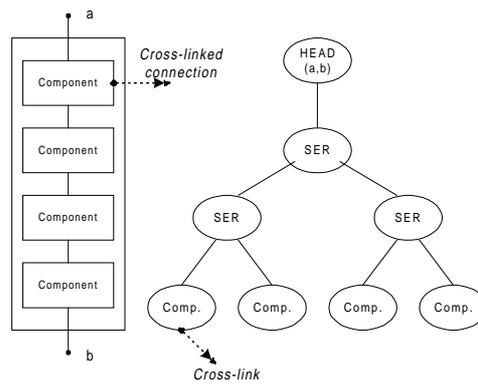


Figure 5 the representation of a single current-flow list as GP-tree

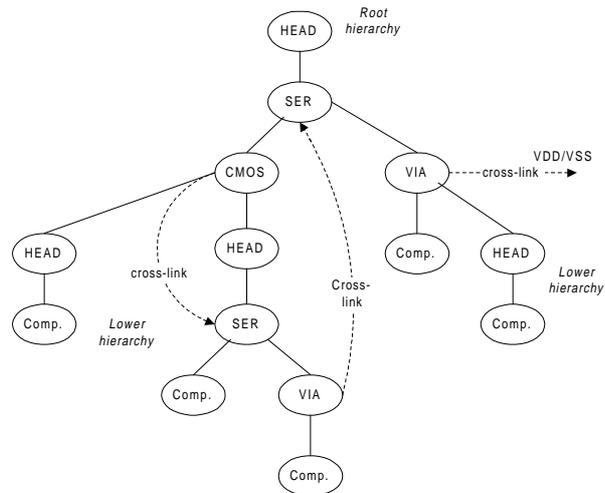


Figure 6 an example of hierarchical GP-tree representation

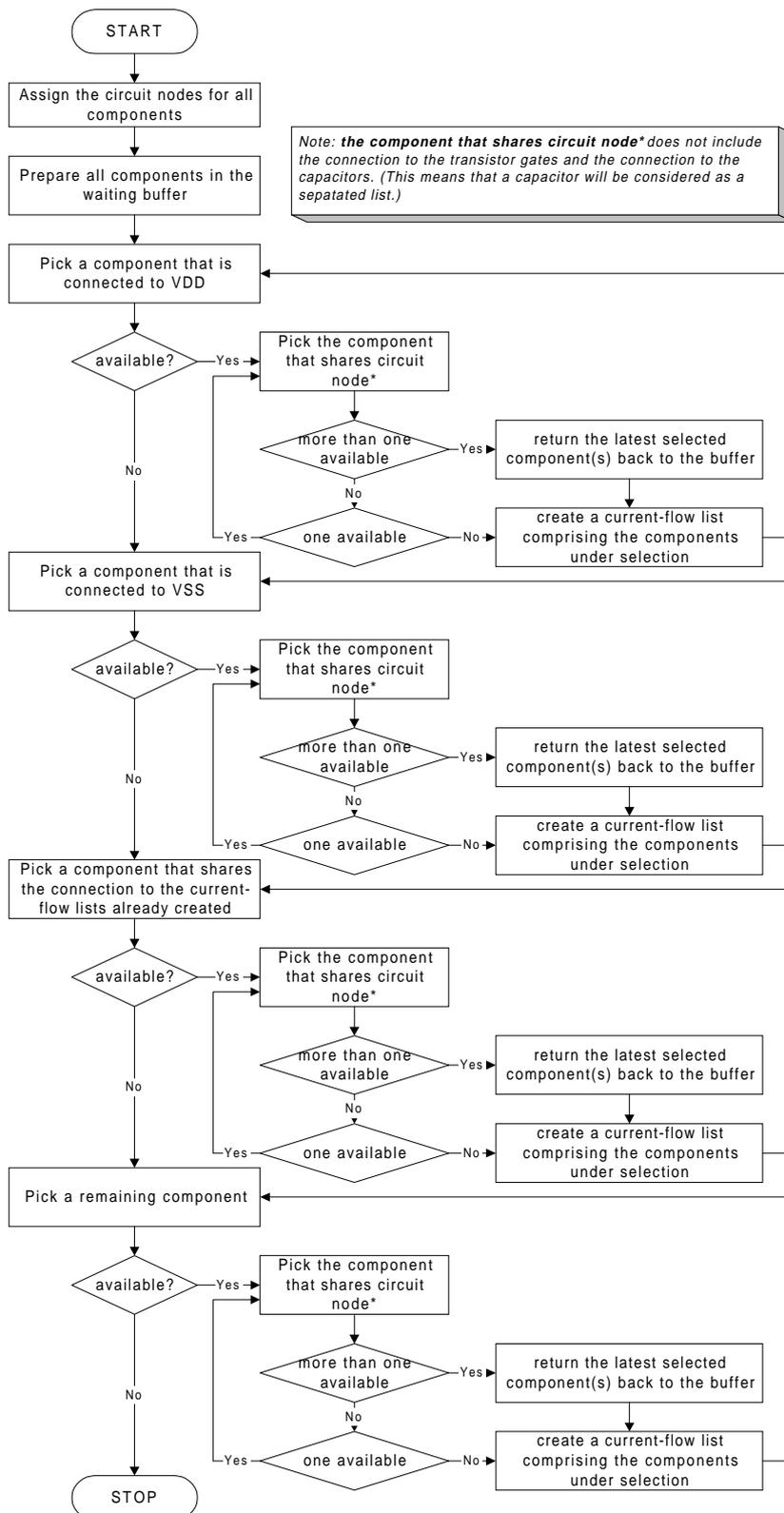


Figure 7 The creation of current-flow lists from an evolving circuit

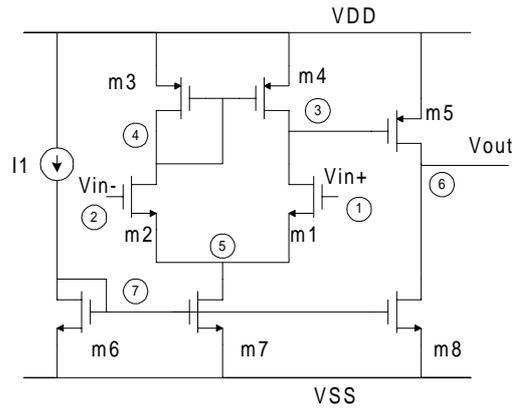


figure 8 differential operational amplifier

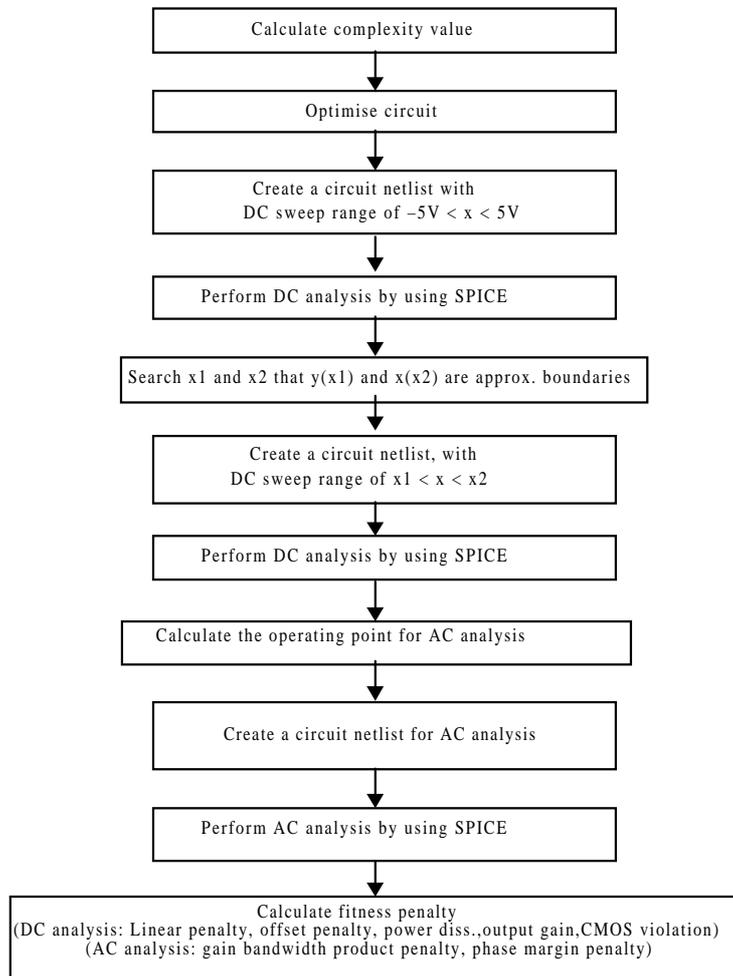


Figure 9 Fitness measuring procedure

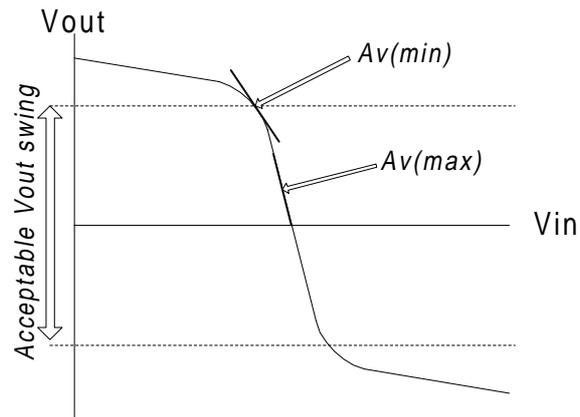


Figure 10 example of DC sweep for the calculation of linearity penalty.

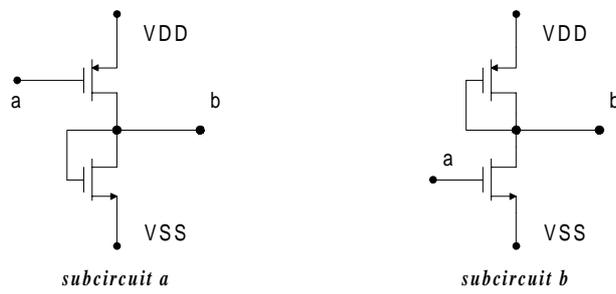


Figure 11 basic modules in the user-defined library

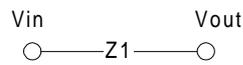


Figure 12 embryonic circuit

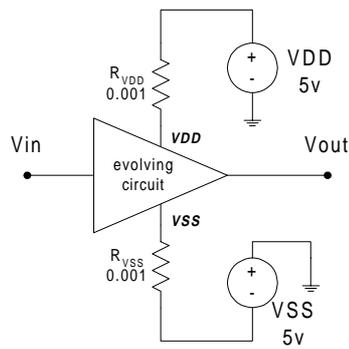


Figure 13 testing harness

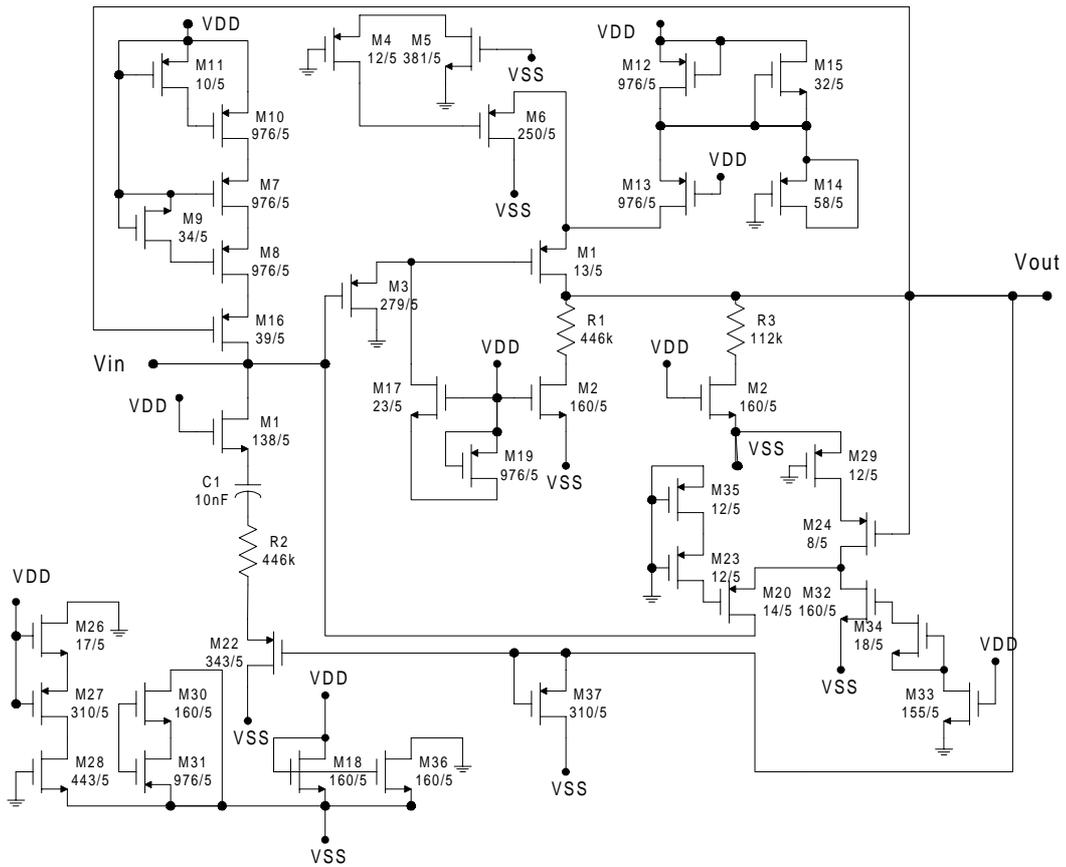


Figure 14 referencing 60dB CMOS amplifier evolved under a similar condition found in the WYWIWYG system

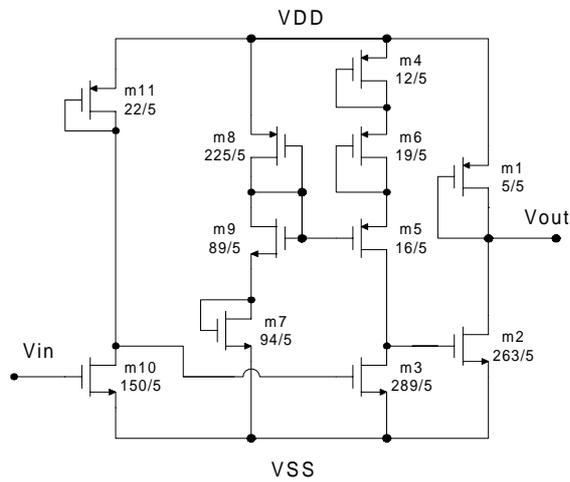


Figure 15 result from a run where phase margin is not restricted

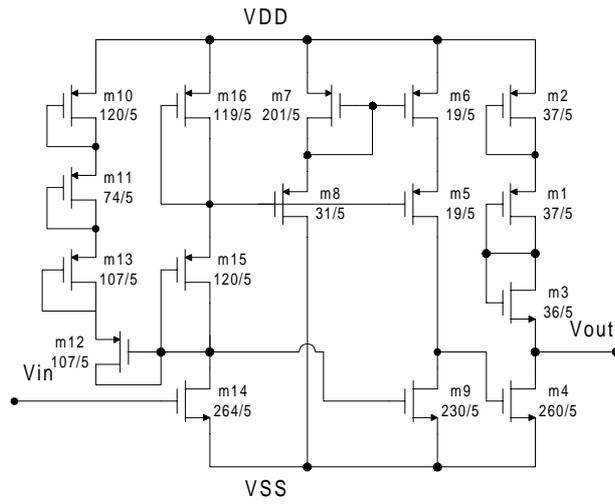


Figure 16 a result from another run

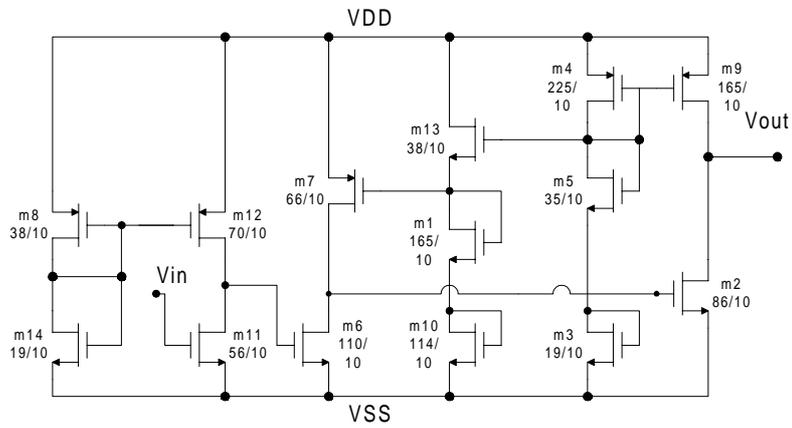


Figure 17 result from a run where transistor length was set to 10 μm

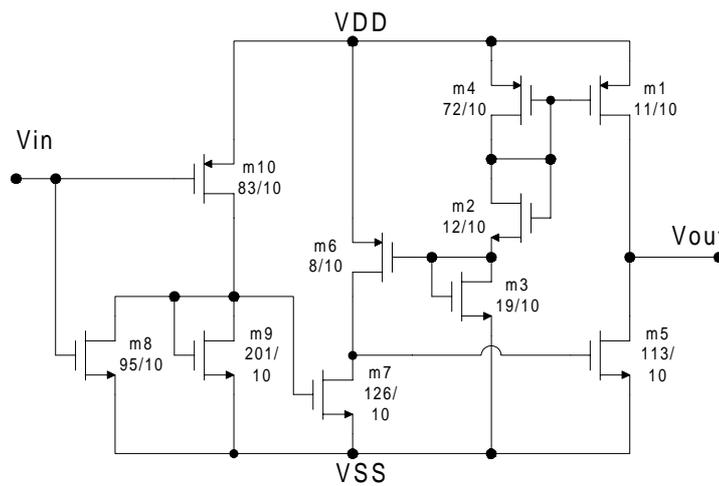


Figure 18 result from another run where transistor length was set to 10 μm

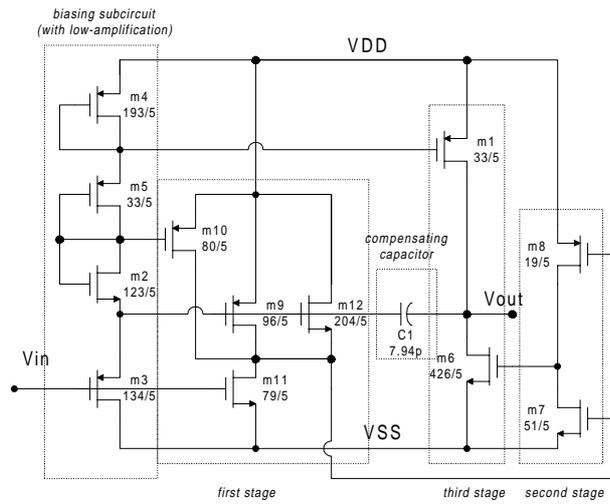


Figure 19 resulting circuit under a restriction of phase margin

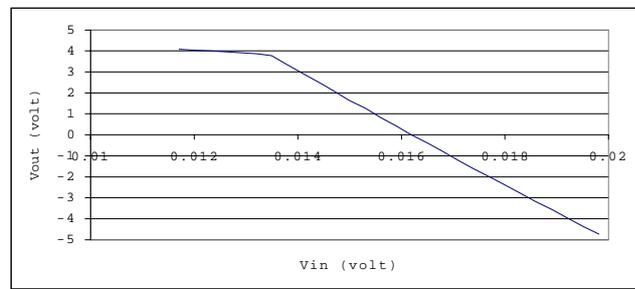


Figure 20 DC sweep of circuit in figure 19

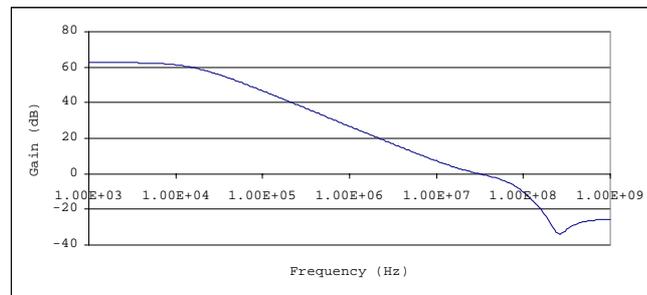


Figure 21 open-loop transfer characteristic of circuit in figure 19

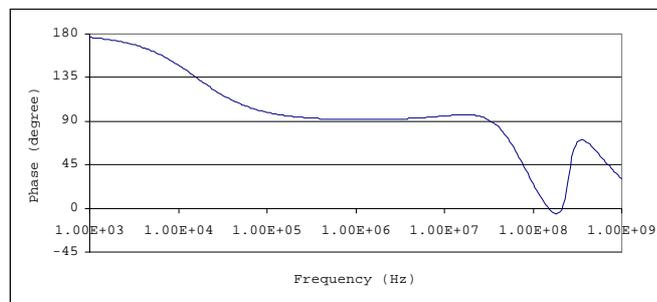


Figure 22 phase margin of circuit in figure 19

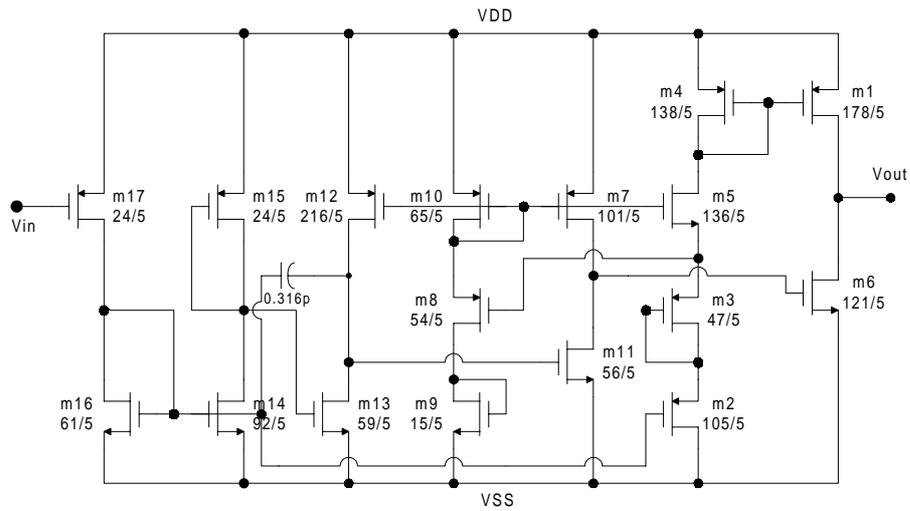


Figure 23 a 90dB amplifier evolved using the proposed system

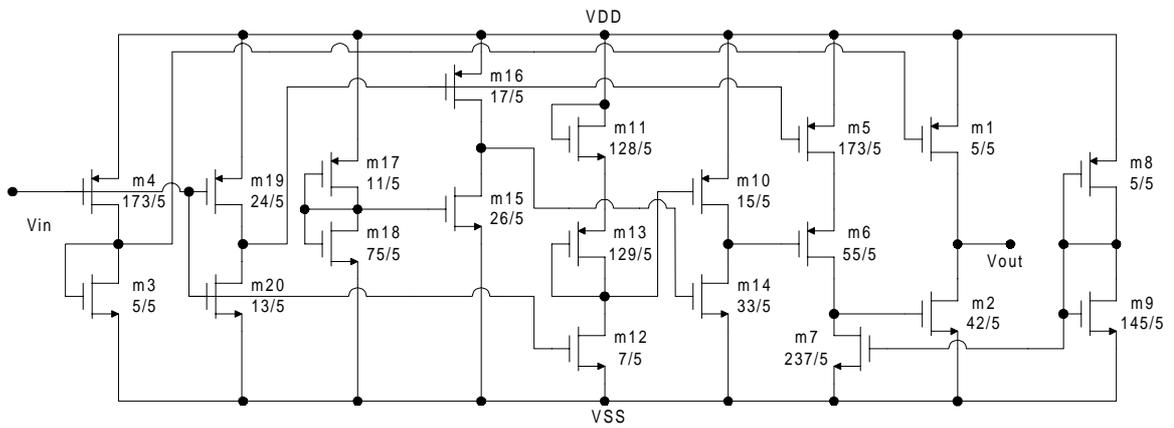


Figure 24 a 120dB amplifier evolved using the proposed system

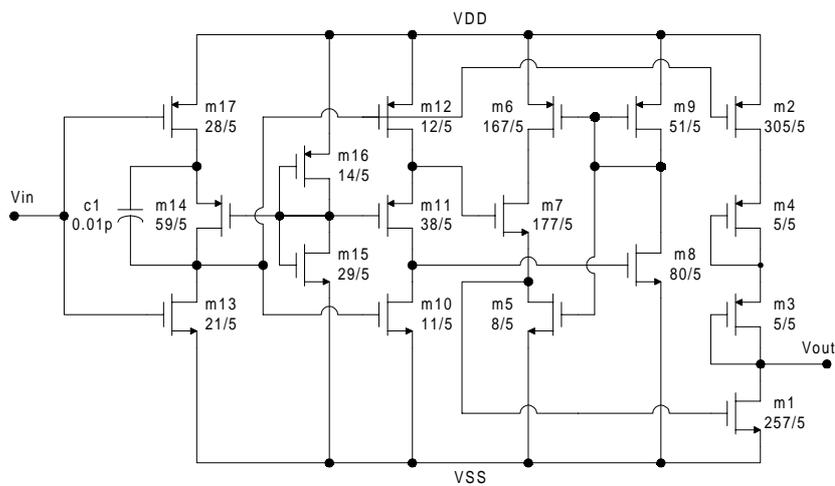


Figure 25 a 60 dB amplifier evolved using the proposed system. Its power supplies were 3 and -3 volt for VDD and VSS respectively

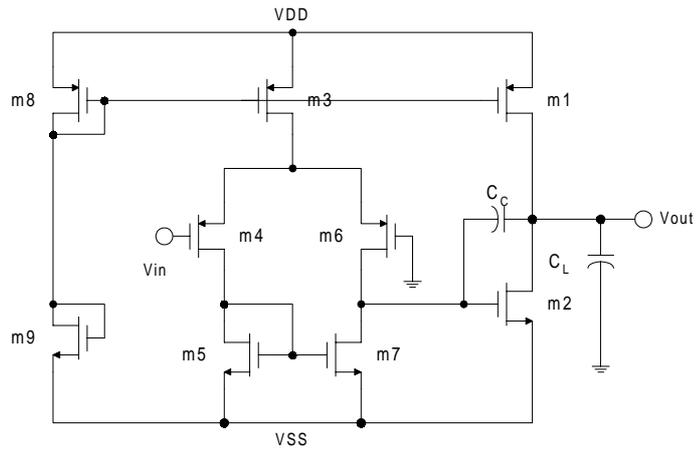


Figure 26 embryonic circuit for a good-start experiment

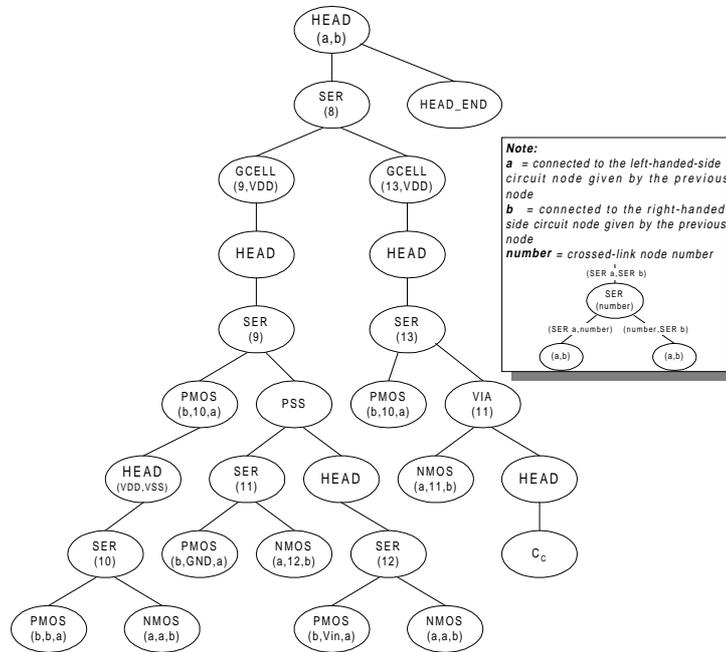


Figure 27 GP-tree representation of the embryonic circuit in Figure 26

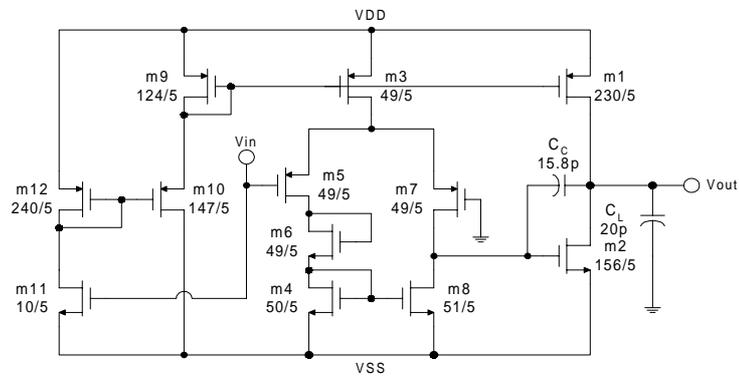


Figure 28 resulting circuit from a run using a good-start strategy

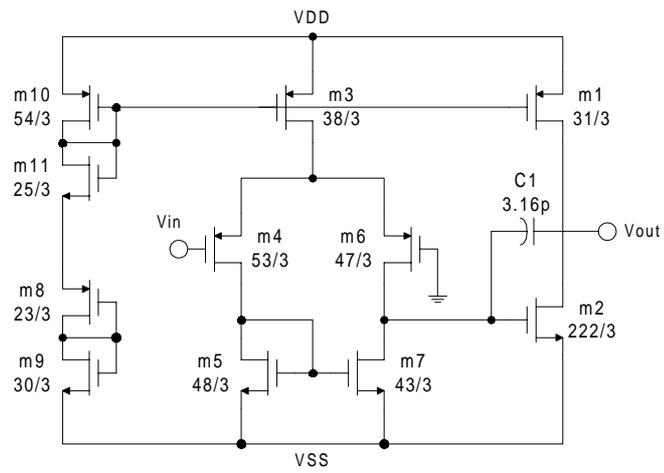


Figure 29 resulting circuit from a run where its target gain was 75dB and transistor length was $3\mu\text{m}$, using embryonic circuit shown in figure 26