Knowledge-Intensive, Causal Reasoning for Analog Circuit Topology Synthesis in Emergent and Innovative Applications

Fanshu Jiao, Sergio Montano, Alex Doboli
Department of Electrical and Computer Engineering
State University of New York at Stony Brook, Stony Brook, NY 11794-2350
Email: {fanshu.jiao,sergio.montano,alex.doboli}@stonybrook.edu

***Abstract*—Analog circuit topology design has been difficult to automate. Topology synthesis involves searching an open-ended, widely extensible, and strongly discontinuous solution space. Existing algorithms cannot generate topologies beyond a constrained set of structures, or experience difficulties in evolving performance-effective yet minimal circuits. This paper proposes a new topology synthesis method that implements a design knowledge-intensive reasoning process to create novel circuit structures with all their features justified by the problem requirements. Two synthesis experiments demonstrate the capability of the method to generate circuits beyond the capabilities of existing topology synthesis algorithms.**

I. Introduction

There has been a constant need to innovate in analog circuit design as new applications and manufacturing processes emerged. Design specifications have changed with respect to the required size, speed, bandwidth, linearity, power consumption, and robustness. Moreover, novel design principles, features, schematics, and models have been invented to tackle existing challenges. EDA tools must rapidly and effectively incorporate such innovations to reduce the gap between their capabilities and state-of-the-art in (manual) design.

While there has been significant advancing in synthesis tools for layout design and circuit sizing, automatically devising or refining circuit topologies (schematics) remains difficult. However, circuit sizing and layout design are often tightly coupled to topology synthesis as incremental modifications in the schematics can simplify transistor sizing and layout design. Circuit topology design is knowledge intensive and designer experience is critical. Moreover, circuit topology design is difficult to frame as an optimization problem (the main approach used in synthesis) as its solution space is open-ended, extensible, and strongly discontinuous. For example, inventing new structural features changes the nature of the solution space because these features can be utilized to produce more circuits, which otherwise are hard to conceptualize.

There are four main approaches used in circuit topology synthesis. The first approach utilizes a database of *if-then* rules to indicate the circuit structures expected to perform best for given requirements [1], [2]. Limitations include the difficulty of devising effective circuit selection rules if complex performance trade-offs must be tackled and the restriction of the results to the topologies encoded by the rules. The second approach utilizes a library of analog cells (OpAmps, comparators, Mux) and a set of transformation rules to convert a signal-flow graph expressed in a language like VHDL-AMS into an implementation [3]. The third approach describes a class of circuit topologies as a template with all possible feedforwards and feedback signal paths, and then uses the template to decide which of the paths should be used in an implementation. The two latter approaches are limited to structured systems, which follow systematic signal flows, like state-space filters and $\Delta\Sigma$ ADC [4]. However, less-systematic structures, like OpAmps or OTAs, are hard to synthesize. The fourth approach uses genetic (GA) or evolutionary algorithms to create structures by interconnecting CMOS devices or simple sub-structures [5], [6]. Design-inspired constraints are embedded into the algorithms to limit the evolutionary process and increase the likelihood of producing effective structures [6]. While this approach can, in theory, evolve any circuit topology, in reality, creating performance-efficient yet minimal structures is hard as it involves searching an open-ended, widely extensible, and strongly discontinuous solution space.

This paper proposes a knowledge-intensive, reasoning-based approach to creating analog circuit topologies for emergent and innovative applications, e.g., problems that involve tackling of novel performance trade-offs and bottlenecks. The reasoning method begins with a set of starting ideas and then continues with a sequence of design steps to complete the solution. Starting ideas are structural features, which correspond to the main conceptual ideas utilized in creating a new topology. Each step of the design sequence is *justified* by the fact that it either introduces a new structural feature that further improves performance or relaxes the constraints of the design. By analyzing circuit topologies created by designers, we observed that starting ideas can be of five types: Type 1 include structural features from different circuits, Type 2 are a mixture of physical and more abstract features, Type 3 involve only abstract features, Type 4 represent ideas of not using certain features, and Type 5 combine physical and abstract features but using them for different purposes than in previous circuits. The paper discusses five different reasoning flows depending on the nature of starting ideas. The reasoning flow uses an expandable design knowledge representation [7], [8] that stores structural features at various level of abstraction and the impact of the features on performance trade-offs.

The paper has the following structure. Section II offers an overview of the synthesis method. The related algorithms are discussed in Section III. Experiments are offered next, and conclusions end the paper.

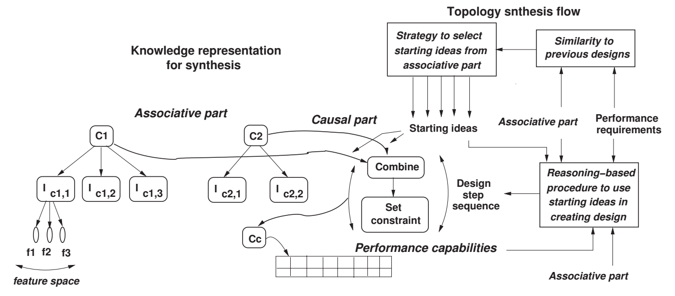


Fig. 1. **Reasoning-based synthesis using knowledge representations**

II. Overview of the Synthesis Methodology

Figure 1 presents an overview of the proposed reasoning-based methodology for synthesis of analog circuit topologies. The methodology includes two main steps: (i) a strategy to select starting ideas from the associative part of the domain knowledge representation and (ii) a reasoning-based procedure to use the starting ideas in creating a design that is a solution to the problem specification, e.g., the considered performance requirements. Both steps utilize information produced by a step that finds the similarity of the current requirements to the requirements of previously solved problems [7].

Starting ideas are often considered to be the most essential elements in the process of devising an innovative solution [9]. There is active research in cognitive psychology [9] and neuroscience [10] attempting to understand the mechanism through which starting ideas emerge. This process is not always conscious, therefore making it difficult to understand the process. However, once they were selected, starting ideas are characterized as either similar to previous designs (analogies [11]), combinations of existing design features [12], generalizations, e.g., through induction [7], and sudden insight (like through restructuring the knowledge representation). The design flow in Figure 1 produces starting ideas using similarity, idea combination, and induction rather than attempting to model the neural, neuro-cognitive mechanism of idea emergence.

The reasoning-based procedure utilizes the starting ideas to identify sequences of design steps that create a performance-satisfying solution. Each step of the sequence is justified by the fact that it either improves performance or it relaxes the design constraints, so that the subsequent steps can further improve performance. Hence, design step sequences are produced through decision making, in which every step is causaly-explained (*justified*) by its design improvements. The decision making process can be expressed by various, invariant patterns, which can then be utilized to tackle starting ideas of a certain kind [13]. The reasoning-based procedure in Figure 1 includes five different reasoning algorithms for each type of starting ideas (Section III).

The synthesis flow in Figure 1 offers a systematic way of implementing divergent - convergent thinking, well known to be the cognitive mechanism that originates innovations [9], [11]. Starting ideas implement divergent thinking and are the main way of introducing novel features that are beyond the concepts already utilized in existing designs. The reasoning-based procedure offers convergent thinking by working out the implementation details that make the starting ideas operational (functional) in a circuit design. Section III present the algorithms to select starting ideas and implement the reasoning-based procedure.

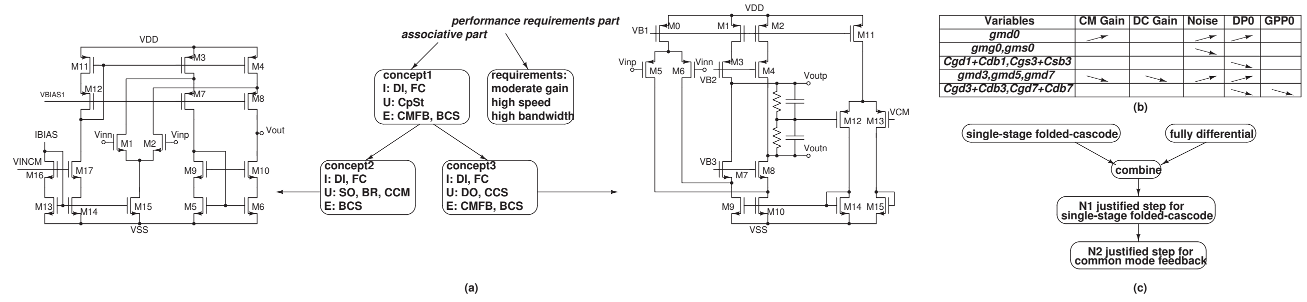
Design knowledge representation. The remaining part of this section summarizes the design knowledge representation used by the reasoning-based flow to synthesize new circuit topologies. A detailed description of the representation is offered in [7] [8]. The knowledge representation has three components: (a) the associative part, (b) performance capabilities, and (c) causal part as shown in Figure 1.

a. The associative part [7] groups circuit designs (e.g., instances $I_{ci,k}$) into more abstract concepts (C_j) based on the similarity of their structural features. Every instance and concept is characterized by three sets of features: (i) Set I is the set of features common to all instances that correspond to the same concept. (ii) Set U is the set of features that are unique to a instance or concept, hence distinguish it from other concepts. (iii) Set E is the set of constraints that must be met for the instance or concept to be functionally feasible, i.e. constraints expressing that MOSFET devices must operate in saturation. Features f_h are symbolic (mathematical) relations that link circuit parameters, nodal currents and voltages, and performance attributes.

b. Performance capabilities indicate the performance trade-offs and bottlenecks of circuit instances and more abstract concepts of the knowledge representation. Trade-offs present how controlling circuit parameters (e.g., device parameters) improve certain performance attributes while worsening other attributes. Bottlenecks are performance attributes that cannot be improved beyond a certain limit for the entire value ranges of the parameters. The methods to compute performance capabilities for circuit instances and concepts are shown in [8].

c. Causal parts present how the features of two or more concepts are combined to form a new concept that offers improved performance and/or leads to a less constrained solution. The causal part indicates the starting ideas and design step sequence for each circuit instance and concept of the associative part [8]. For example, in Figure 1, the causal part for concept C_c includes the starting ideas representing features of the two concepts C_1 and C_2 and a two-step design sequence that combines the features and then sets a set of constraints on the feature parameters (e.g., matching their values).

Example: Figure 2 presents the design knowledge representation built for two single-stage folded-cascode OTA circuits. Figure 2(a) shows the associative part, with $concept_2$ corresponding to the left circuit and $concept_3$ representing the right circuit. $Concept_1$ is the abstraction of concepts $concept_2$ and $concept_3$. It includes the common attributes of the two circuits and models the concept of differential input, folded cascode circuits. These concepts also have distinguishing attributes and enabling conditions respectively. The description of the features i.e. DI, FC, etc. is in [8]. Figure 2(b) is a fragment of the performance capability table of the right circuit. It shows the performance trade-offs for common mode gain, DC



Variables	CM Gain	DC Gain	Noise	DP0	GPP0
gmd0	↗		↗	↗	
gmg0,gms0			↘		
Cgd1+Cdb1,Cgs3+Csb3				↘	
gmd3,gmd5,gmd7	↘	↘	↗	↗	
Cgd3+Cdb3,Cgd7+Cdb7				↘	↘

Fig. 2. **Design knowledge representations for two circuits**

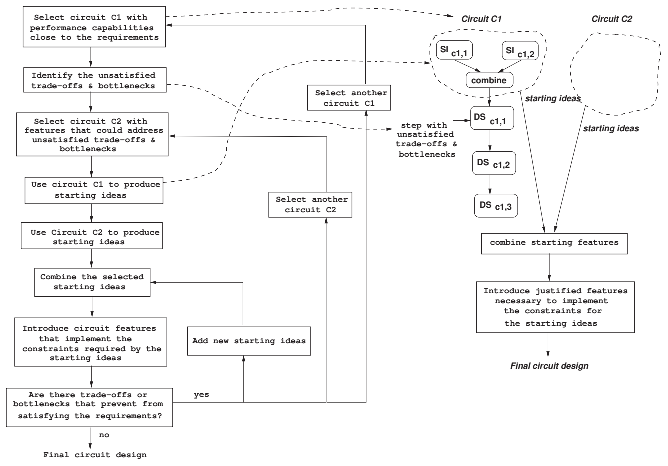

Select circuit C1 with performance capabilities close to the requirements
Identify the unsatisfied trade-offs & bottlenecks
Select circuit C2 with features that could address unsatisfied trade-offs & bottlenecks
Use circuit C1 to produce starting ideas
Use Circuit C2 to produce starting ideas
Combine the selected starting ideas
Introduce circuit features that implement the constraints required by the starting ideas
Are there trade-offs or bottlenecks that prevent from satisfying the requirements?
yes
no
Final circuit design
Add new starting ideas
Select another circuit C2
Select another circuit C1
step with unsatisfied trade-offs & bottlenecks
Circuit C1
Circuit C2
SI c1,1
SI c1,2
combine
DS c1,1
DS c1,2
DS c1,3
starting ideas
starting ideas
combine starting features
Introduce justified features necessary to implement the constraints for the starting ideas
Final circuit design

Fig. 3. **Reasoning methodology for starting ideas of Type 1**

gain, noise, dominant pole, and gain-pole product. Upward arrows express that increasing the variable values improves the performance attribute. Figure 2(c) illustrates the starting ideas and the corresponding design sequence of the right circuit.

III. Synthesis Algorithms

There are five synthesis flows depending on the nature of the starting ideas. Starting ideas can be classified into the following categories: Type 1 are starting ideas that combine physical features of existing circuits, Type 2 correspond to combining physical and more abstract features, Type 3 are combinations of abstract features, Type 4 are starting ideas of not using certain features, and Type 5 represent a novel abstraction created from combined physical features of existing circuits. The synthesis flow for each type of starting ideas is presented next.

Starting ideas of Type 1. **Example**: To address high gain, low power applications, a new OpAmp can be created by combining physical features, like cross-coupled floating batteries class AB input and gain boosted common source feedback OpAmp. The two physical features form the starting ideas and are the circuit gain stages. The design sequence adds a low voltage current mirror to bias and improve output resistance.

This reasoning process is conceptually presented in Figure 3. A circuit C_1 is first selected from the design knowledge representation as being the one closest to meeting the performance requirements, e.g., gain-boosted cascode OpAmp in our example. The figure shows the causal part for circuit C_1 including its starting ideas $SI_{c1,1}$ and $SI_{c1,2}$ and the related design step sequence $DS_{c1,1}$, $DS_{c1,2}$, and $DS_{c1,3}$. Let's assume that the trade-offs that prevent from meeting the problem requirements are introduced in step $DS_{c1,1}$. Then, circuit C_2 is selected because it incorporates features that can address the unsatisfied trade-off in circuit C_1, i.e. using class AB input OpAmp in our example. The starting ideas for reasoning include two sources: all features of circuit C_1 up to the step that introduces the unsatisfied trade-offs (e.g., the two starting features $SI_{c1,1}$ and $SI_{c1,2}$) and circuit C_2's features that tackle the unsatisfied trade-offs. These features are combined. The remaining sequence adds the design steps that are justified by the operational needs of the selected features, i.e. biasing, current sources and mirrors.

Algorithm. Figure 3 illustrates the reasoning methodology for circuit topology synthesis. The first step selects a circuit with performance close to the problem requirements. The features of this circuit, including the corresponding concepts in the associative and causal parts, are used next to identify which features and design step introduce the trade-offs that prevent from satisfying the performance requirements. Then, the methodology selects a second circuit with features that can tackle the unsatisfied trade-offs of the first circuit. The features of the two circuits are utilized to create the starting ideas for the methodology. The starting ideas are followed by adding the structural features required to implement the constraints needed for correct operation of the structures represented by the starting ideas. The final design is produced, if there are no trade-offs preventing the satisfaction of the problem requirements. Otherwise, the reasoning flow iterates by considering new starting features to address any unsatisfied requirements. If unsuccessful, other options for the second circuit are analyzed. If still no constraint-satisfying design is created then the features of another circuit are used as starting ideas.

Starting ideas of Type 2. These ideas combine features of a physical circuit with features of an abstract concept.

Example. In Figure 4(a), the structural, physical features of a differential OpAmp are combined with the abstract feature representing common-mode feedback(CMFB). Figure 4(b) presents the associative part corresponding to CMFB concept,

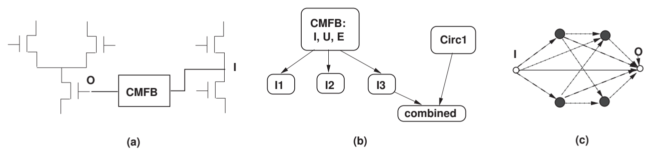


Fig. 4. **Reasoning for starting ideas of Type 2**

in which three specific instances I_i describe physical implementations of the concept. Moreover, the causal part indicates the way in which the features are utilized in actual circuits, e.g., feature I_3 is combined with the structural features of $Circ_1$ to create a new circuit topology.

Algorithm. The first two steps of the reasoning-based synthesis flow are the same as for the flow in Figure 3(a). However, the third step selects an abstract concept, e.g., CMFB, instead of physical features. Each of the instances available in the associative part for the concept are considered next as candidates of starting ideas to be combined with the physical features. The pursued flow is similar to steps 4-8 in Figure 3. If none of the available feature alternatives can tackle the unsatisfied requirements then the reasoning flow uses a bottom-up induction step to create more structural feature alternatives that correspond to the CMFB concept. Each of the generated features is added to the associative part and then analyzed to decide if it is part of the solution by being combined with the features of the first circuit. This process is similar to steps 4-8 in Figure 3.

The bottom-up induction step attempts to create new alternatives for a concept by using the following mechanism. The input and output nodes of the alternative are similar to all instances of the concepts, e.g., nodes I and O for the concept in Figure 4(a). In addition, the alternative will include all nodes and node couplings that are common to all instances of the concept, hence are part of the set $I \cup U$ describing the abstraction. We indicated such couplings with a continuous line in Figure 4(c). The bottom-up process creates new structures by gradually adding new nodes to the structure until a feasible alternative is identified and the structure is not too complex, e.g., there are too many new nodes added to the network. These nodes are darkened in the figure. Every time a new node is added, all possible couplings are identified as shown by the dashed arcs in the figure. These arcs represent the total solution space available for a given set of nodes to create a new alternative for the concept. Note that induction creates new features (like mutation in genetic algorithms).

Starting ideas of Type 3. Starting ideas of Type 3 combine the features of two more abstract concepts.

Example. For high gain, high bandwidth applications, the starting ideas describe combining a multi-path OpAmp with the idea of feedforward compensation. Both ideas are abstract. Next, the related design sequence adds three gain stages (e.g., telescopic cascode, current source, current source stage) to implement the abstract multi-path OpAmp, common source feedforward path to implement the abstract idea of feedforward compensation, and current source biasing.

Algorithm. After combining the abstract features into a new abstract concept, the reasoning process of the synthesis method

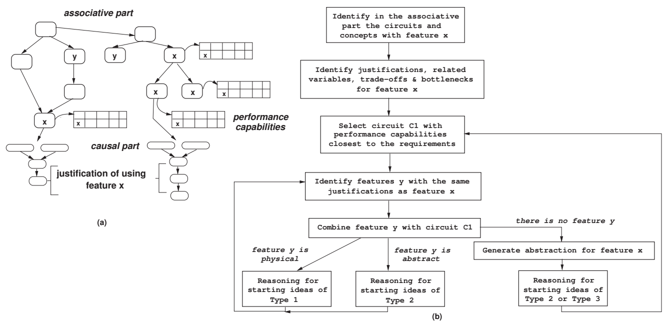

associative part
y
y
x
x
x
x
x
x
x
x
performance capabilities
causal part
justification of using feature x
(a)
Identify in the associative part the circuits and concepts with feature x
Identify justifications, related variables, trade-offs & bottlenecks for feature x
Select circuit C1 with performance capabilities closest to the requirements
Identify features y with the same justifications as feature x
Combine feature y with circuit C1
there is no feature y
feature y is physical
feature y is abstract
Generate abstraction for feature x
Reasoning for starting ideas of Type 1
Reasoning for starting ideas of Type 2
Reasoning for starting ideas of Type 2 or Type 3
(b)

Fig. 5. **Reasoning methodology for starting ideas of Type 4**

instantiates the features following the same sequence of steps as the instancing that is performed for starting ideas of Type 2.

Starting ideas of Type 4. Starting ideas of Type 4 start from the consideration that certain features should not be utilized in the solution due to their unwanted influence on performance.

Example. For high bandwidth OpAmps, designs might not include Miller compensation capacitor since the capacitor pushes the dominant pole to lower frequency in spite of good phase margin. Then, the reasoning strategy attempts to identify alternative features that offer the same justification (e.g., performance improvement or constraint relaxation) like the unwanted feature but not the undesired trade-offs. For example, a feedforward path improves speed and increases bandwith through the introduced zeros to cancel second dominant poles for better phase margin. The related design steps add common source path to a regular three-stage OpAmp.

Algorithm. Figure 5(b) presents the proposed reasoning methodology for starting ideas of Type 4, and Figure 5(a) shows a simple knowledge representation to illustrate the methodology. The first step of the methodology identifies the circuit instances and concepts that include an unwanted feature x. Then, the causal parts of the identified circuits and concepts are utilized to find the justifications for using feature x, e.g., the specific performance improvement due to the feature. In addition, the tables expressing performance capabilities indicate the variables due to feature x and the related performance trade-offs and bottlenecks. The justifications relevant to the problem requirements are used next to identify homonym features y, e.g., features that can create the same justifications as feature x, but do not introduce trade-offs that negatively impact the problem-relevant trade-offs.

The third step of the methodology selects circuit C_1 with performance capabilities closest to the desired problem requirements. Next, the features of the selected circuit is combined with feature y. If feature y is at the physical level then the flow continues with the reasoning methodology used for starting ideas of Type 1 (Figure 3) as the process resembles now that of combining starting ideas that are physical features. If feature y is abstract, then the reasoning methodology continues by using the flow for combining ideas of Type 2 (Figure 4) as the process is similar to combining physical features (from circuit C_1) with abstract features (e.g., feature x).

Finally, if there are no valid features y in the associative part then the reasoning flow proceeds by creating an abstraction

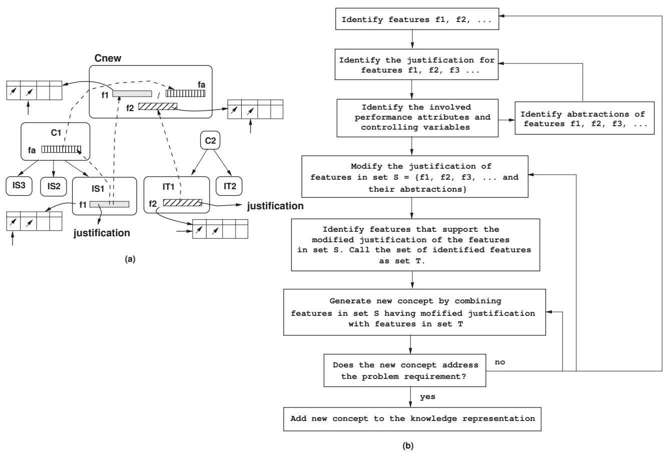

Cnew
f1
f2
fa
C1
fa
IS3
IS2
IS1
f1
justification
IT1
f2
C2
IT2
justification
(a)
Identify features f1, f2, ...
Identify the justification for features f1, f2, f3 ...
Identify the involved performance attributes and controlling variables
Identify abstractions of features f1, f2, f3, ...
Modify the justification of features in set S = {f1, f2, f3, ... and their abstractions}
Identify features that support the modified justification of the features in set S. Call the set of identified features as set T.
Generate new concept by combining features in set S having mofified justification with features in set T
Does the new concept address the problem requirement?
no
yes
Add new concept to the knowledge representation
(b)

Fig. 6. **Reasoning methodology for starting ideas of Type 5**

of feature x. New physical features are produced for this abstraction by using the same bottom-up induction step also used for combining ideas of Type 2. The reasoning process proceeds iteratively to analyze the set of homonym features y as well as other circuits C_1, if the currently considered circuits do not produce a performance-satisfying solution.

Starting ideas of Type 5. The reasoning methodology produces a new abstract concept starting from existing design features but changing their justification. Then, the new concept is combined with physical features by following the methodology for starting ideas of Type 2, or is combined with an existing concept by following the methodology for ideas of Type 3.

Example. To design a highly linear, high bandwidth circuit, the starting ideas include a pseudodifferential OTA circuit and a feedforward path crossing input stage, which is justified for nonlinearity. However, the feedforward path is extended from input stage to output stage to provide also frequency compensation (the new justification) besides nonlinearity cancellation (the traditional justification). The related design step is the implementation of the feedforward path, adding common mode feedback circuit.

Figure 6(a) presents a more conceptual example for this reasoning methodology. The steps of the methodology are shown in Figure 6(b). The example illustrates a new concept C_{new} that is created starting from features f_1 used in circuit IS_1 and feature f_2 incorporated in circuit IT_1. The two features have precise justifications, e.g., they improve the pointed performance attributes, like the first attributes in the performance capabilities tables. However, in concept C_{new}, the justifications of the two features change, i.e. they are now used to improve the second attributes in the performance capabilities. In order to achieve this goal, the two features are combined with other features in the associative part, so that concept C_{new} tackles the requirements of the problem. Similarly, an alternative of this process is the one in which features f_1 or f_2 are used to first identify their abstractions, e.g., feature f_a of concept C_1 abstracts feature f_1 (circuit IS_1 is an instance of C_1). Next, the reasoning flow follows the same steps as before but using the abstract feature f_a.

Algorithm. The algorithm in Figure 6(b) starts by identifying the physical features f_1, f_2, ... that are candidates to produce a new concept. Next, the justifications of each feature are found by examining the causal parts of the circuits that include the feature (e.g., the performance improvement created by feature f_1 in circuit IS_1). The attributes and the controlling variables of each of the selected features are also identified using the performance capability tables of the knowledge representation. Feature abstractions are also added to the set of candidate features at this step. Next, the justifications of the candidate features are changed by selecting problem-related performance attributes (that are part of the requirements), which are improved by the feature. The resulting set is called set S. The following steps identify features in other concepts of the associative part, such that a new concept results by combining the features with those in set S. If the new concept addresses the problem requirements, then the concept is added to the knowledge structure. Otherwise, the reasoning flow iterates by considering other sets of features.

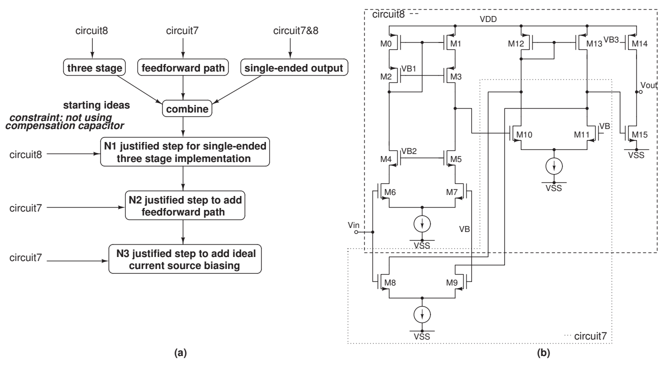


Fig. 7. **Starting ideas and design sequences for synthesized circuit**

IV. Experiments

This section discusses the creation of two new circuit topologies using the methodology in Figure 1. The used design knowledge representation was based on 30 state-of-the-art high-frequency OpAmps and OTAs. The circuit schematics are shown in [8].

Case study 1. The goal of this synthesis experiment was to create a low-power amplifier that optimizes the gain-bandwidth product. Figure 7(a) illustrates the starting ideas and the design step sequence that produced the circuit shown in Figure 7(b). The starting ideas were of Type 4 as they added the constraint of not using compensation capacitors. The starting ideas combined three stage, feedforward compensation path, and single-ended output features. These ideas were selected from circuits 7, 8, and 11, which are the high gain, high frequency circuits in the knowledge representation. Feedforward compensation path was selected as a starting feature as it is in the knowledge set the only way to compensate frequency without using Miller capacitor.

The design step sequence includes the following steps. Step $N1$ implemented three gain stages by cascode, current mirror and common source stages, and the circuit is single-input, single-ended output (features from circuit 8). The step is justified by multistage boosted gain and higher power

TABLE I
PERFORMANCE COMPARISON

Performance	7	8	new circuit
Technology[μm]	0.6 CMOS	0.6 CMOS	0.6 CMOS
Supply voltage[V]	$\pm$1.25	2	$\pm$1.25
Static power[mw]	0.63	0.42	0.65
Gain[dB]	71	80	73
Bandwidth[MHz]	0.15	0.012	0.15
GainBwProd[MHz]	539	123	620
Noise[V^2/Hz]@20MHz	1.8e-12	2.7e-14	2.3e-12
Setting time[ns]@20MHz	33.16	63.22	27.24
Slew rate@20MHz	40.55e6	56.12e6	7.02e9
Thd@20MHz	28.29%	39.43%	13.96%

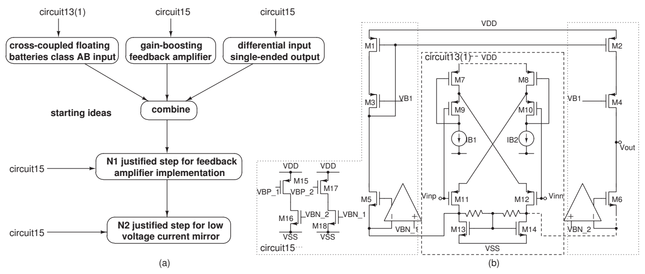

circuit13(1)
circuit15
circuit15
cross-coupled floating batteries class AB input
gain-boosting feedback amplifier
differential input single-ended output
starting ideas
combine
circuit15
N1 justified step for feedback amplifier implementation
circuit15
N2 justified step for low voltage current mirror
(a)
VDD
M1
M2
circuit13(1)
M7
M8
M9
M10
M3
VB1
VB1
M4
IB1
IB2
Vout
VDD
VBP_1
VBP_2
M15
M17
M16
VBN_2
VBN_1
M18
VSS
circuit15
M5
Vinp
M11
M12
Vinn
M6
M13
M14
VBN_1
VBN_2
VSS
(b)

Fig. 8. **Starting ideas and design sequences for synthesized circuit**

efficiency. Step $N2$ introduced common source to implement the feedforward path (feature from circuit 7). The step is justified by high gain-bandwidth product and good phase response. Step $N3$ added ideal current source biasing required for correct operation of the circuit (feature from circuit 7). The resulting circuit topology is shown in Figure 7(b).

Table I summarizes the performance of the new circuit as compared to circuits 7 and 8. The gain-bandwidth product is by 13% higher than that of circuit 7 and by 80% higher than that of circuit 8. As a result of the maximized gain-bandwidth product, the new circuit has the fastest step response with best setting time and slew rate accordingly.

Case study 2. The goal was to create a low-power, high-gain OpAmp that uses a low supply voltage. Figure 8(a) illustrates the starting ideas and the design step sequence that produced the circuit in Figure 8(b). The starting ideas were of Type 1 as they combined two physical features: class AB input stage and feedback amplifier. The starting ideas were identified as follows. The knowledge representation contains the following low power circuits: 3, 13(1), 13(2), 13(3), 17, 19, and 30. The corresponding abstract concept includes the features: class AB input, local common-mode feedback, and class AB output. Class AB input stage and local common-mode feedback (from circuit 13) were selected as this combination achieves near optimal current efficiency. In contrast, the feature of adaptive biasing class AB input stage (used in circuit 3) was found only in switched capacitor (SC) circuits. Also, the feedback of circuit 15 was selected as it is used in a two-stage amplifier. The design step sequence includes the following steps. Step $N1$ introduced a simple gain-boosting stage justified by improving the gain for low supply voltage. Step $N2$ added a low voltage current mirror (used in circuit 15) justified by increased output resistance. The resulting circuit topology is shown in Figure 8(b).

Table II summarizes the performance of the new circuit as compared to circuits 13(1) and 15. Gain of the new circuit is by 86% higher than that of circuit 13(1) and by 75% higher than that of circuit 15. The power consumption is only 3% higher than that of circuit 13(1) and 0.4% of circuit 15.

TABLE II
PERFORMANCE COMPARISON

Performance	13(1)	15	new circuit
Technology[μm]	0.6 CMOS	0.6 CMOS	0.6 CMOS
Supply voltage[V]	$\pm$1	5	$\pm$1
Static power[mw]	0.097	23.8	0.1
Gain[dB]	45	50	62
Bandwidth[MHz]	0.18	0.32	0.02
GainBwProd[MHz]	32	104.8	27.4
PhaseMargin	61°	78.5°	54°
Noise[V^2/Hz]@20MHz	5.3e-16	5.1e-15	1.2e-14

The synthesized topologies are similar to designer-created circuits. Even though we do not have formal evidence, we think that such circuits are hard to be evolved by GAs.

V. Conclusions

This paper presents a new topology synthesis method that implements a design knowledge-intensive reasoning process to create novel circuit structures with all their features justified by the problem requirements. Five methodologies are described depending on the nature of the starting design features. Two synthesis experiments demonstrate the capability of the method to generate circuits beyond the capabilities of existing topology synthesis algorithms. Future work will study the reasoning flows for other types of starting ideas.

References

[1] R. Harjani, R. A. Rutenbar, and L. R. Carley, "Oasys: A framework for analog circuit synthesis," *IEEE Trans. CADICS*, vol. 8, no. 12, pp. 1247–1266, 1989.

[2] F. El-Turky and E. E. Perry, "Blades: An artificial intelligence approach to analog circuit design," *IEEE Trans. CADICS*, vol. 8, no. 6, pp. 680–692, 1989.

[3] A. Doboli, N. Dhanwada, A. Nunez-Aldana, and R. Vemuri, "A two-layer library-based approach to synthesis of analog systems from vhdl-ams specifications," *ACM Transactions on Design Automation*, vol. 9, no. 2, pp. 238–271, 2004.

[4] Y. Wei, A. Doboli, and H. Tang, "Systematic methodology for designing reconfigurable σ modulator topologies for multimode communication systems," *IEEE Trans. CADICS*, vol. 26, no. 3, pp. 480–496, 2007.

[5] W. Kruiskamp and D. Leenaerts, "Darwin: Cmos opamp synthesis by means of a genetic algorithm," in *Proceedings of the 32nd annual ACM/IEEE design automation conference*. ACM, 1995, pp. 433–438.

[6] T. McConaghy, *Variation-aware analog structural synthesis: a computational intelligence approach*. Springer, 2009.

[7] C. Ferent and A. Doboli, "Symbolic matching and constraint generation for systematic comparison of analog circuits," *IEEE Trans. CADICS*, vol. 32, no. 4, pp. 616–629, 2013.

[8] F. Jiao and et al., "Analog circuit design knowledge mining: Discovering topological similarities and uncovering design reasoning strategies," *IEEE Trans. CADICS, accepted*, 2014.

[9] P. Thagard and T. C. Stewart, "The aha! experience: Creativity through emergent binding in neural networks," *Cognitive science*, vol. 35, no. 1, pp. 1–33, 2011.

[10] J. Luo and K. Niki, "Function of hippocampus in insight of problem solving," *Hippocampus*, vol. 13, no. 3, pp. 316–323, 2003.

[11] S. Vosniadou and A. Ortony, *Similarity and analogical reasoning*. Cambridge University Press, 1989.

[12] F. J. Costello and M. T. Keane, "Efficient creativity: Constraint-guided conceptual combination," *Cognitive Science*, vol. 24, no. 2, pp. 299–349, 2000.

[13] D. Kahneman and A. Tversky, "Choices, values, and frames." *American psychologist*, vol. 39, no. 4, p. 341, 1984.