Interactive Procedural Computer-Aided Design

Carlo H. Séquin
CS Division, University of California, Berkeley
sequin@cs.berkeley.edu

Abstract

The typical engineering design process can be decomposed into several phases: creative exploration of ideas, testing soundness of proposed concepts, refining concepts to realizable solutions, optimizing viable solutions with respect to performance/cost. Powerful computer algorithms have been developed for many of these tasks. Often these modules are rigid, allowing for little intervention by the designer, and the management of the interactions of these tasks mostly relies on human intelligence. Better user interfaces are required to integrate more fully human ingenuity and the assistance of the computer into the overall design process. The most powerful CAD systems should combine the power of programming, graphical visualization, and interactive adjustment of crucial design parameters.

1. Introduction

Computer-aided tools play an important role in many areas of design, including integrated circuits, architecture, engine design, avionics, etc. Many of today's designs, such as bridges, airplanes, or integrated circuits could not be done without computers. In general, the engineering design process can be structured into three key stages: exploration, development, refinement (Fig.1). In stage 1, ideas are collected, relevant related designs are searched for, and first tentative designs are checked for their viability. In stage 2, promising concepts from stage 1 are adjusted to meet the current constraints and optimized to approach the desired goal functions as closely as possible. In stage 3, the best designs are refined to fit the intended implementation process and are further embellished to become as attractive as possible will still serving the desired functions.

Powerful CAD modules exist for the tasks relating to all three phases. The later stages first received support from CAD tools, mostly in the form of sophisticated drafting tools. Later CAD tools started to perform detailed analysis, functional simulation or parametric optimization. More recently CAD tools have attempted to also support the first stage of idea generation and conceptual design, but their effectiveness in this area is still rather limited.

I. EXPLORATION
Concept Generation, Viability Checks
➔ **Schematic Design**

II. DEVELOPMENT
Observing Constraints, Optimization
➔ **Detailed Design**

III. REFINEMENT
Design for Realization, Embellishment
➔ **Construction Drawings**

Figure 1. Three stages in the design process.

Moreover, in most CAD environments the integration between these phases is seriously lacking. The various CAD modules often run on different computers, and it may require significant data conversion and re-coding to take a design from one algorithm to the next one. Human interaction is often limited to selecting which CAD tool to use next and to setting a few explicit parameters. In this paper we make the point that more interaction, giving the user more direct control, is desirable in all modules. Through an improved symbiosis of optimization, visualization, human intelligence, and engineering expertise, CAD environments will advance to the next more powerful level. Giving the human more direct control will pay off for almost all CAD modules. In this tutorial review we will discuss illustrative examples from all three stages of the design process.

2. Exploration Phase

The first stage is the most unstructured one. So far, CAD support has been the weakest in the early conceptual, more creative phases of design. There are few programs that can propose radically new approaches. Computer searches have been less

successful in the early conceptual stages of design, where creativity, "gut feelings," and insightful sparks are needed to come up with truly new approaches.

The most creative programs are based on evolutionary techniques, such as genetic algorithms (GA) [5]. These algorithms have been demonstrated in many playful settings and have often produced truly surprising results [14]. However, they very rarely produce practical, usable engineering solutions for real-world problems. The reason is that the possible solution space for such problems is always humongous, and the sampling produced by GA in any finite amount of time is thus only very sparse. Therefore any "solution" found by GA, although good enough to beat out other competing machine-generated solutions, is rarely good enough to meet the stringent engineering requirements of a successful system or consumer product.

Because of this, it is advantageous to subject the most promising "solutions" found by GA to a more narrowly focused search or to a greedy optimization that allows to match exactly some non-negotiable design requirements and to approach more closely other design objectives. This fine-tuning will not typically change the given structure, but will only adjust a few aspects of the current design, and optimize a few pre-defined parameters. Such a combination of stochastic search and greedy optimization can be quite powerful if the search domain is kept focused narrowly enough.

2.1. MEMS Resonator Suspension

To illustrate the above comments, let's look at a possible design scenario involving the design of a MEMS (Micro-Electro-Mechanical System) resonator, designed to serve as a narrow band-pass filter in a signal processing application. The task is to find the best possible suspension design for some central mass to which are attached two capacitive comb drives, – one acting as an input driver, and the other one acting as an output sensor (Fig. 2a). The central mass must be suspended by some kind of spring to form a spring-mass system with the proper resonant frequency. The mass with the attached drive and sensing combs must oscillate preferentially in the direction of the comb fingers and be rather stiff in the direction orthogonal to it. The stiffness in the two directions should differ by at least a factor of 10. Design experience with MEMS readily informs us that a practical solution should employ at least four suspension springs to keep this oscillating mass in its initial plane. Thus, at this point, the design task may be formulated as finding four string-like poly-line beams connected with one end to the central mass and with their other ends anchored to the silicon substrate (Fig. 2b).

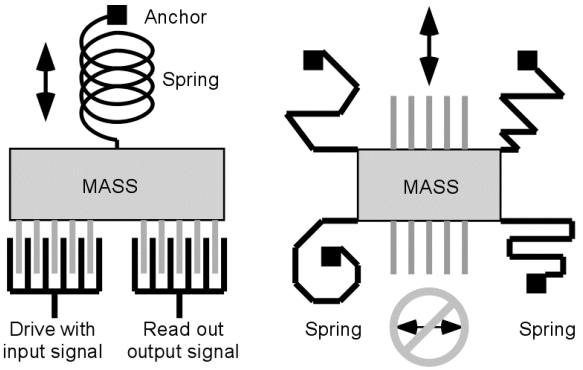

Figure 2. MEMS resonantor: (a) schematic diagram, (b) suspension with 4 unconstrained polyline springs.

This task was subjected to a genetic algorithm (GA) to discover potentially new, innovative geometries for the suspension springs [17]. The genotype employed was flexible enough to allow poly-lines with arbitrary many piecewise linear links of various width and length, connected at arbitrary angles. Beam geometries that led to intersections were ruled out. The remaining "legal" geometries were analyzed as to their resonant behavior with the MEMS simulation program SUGAR [16]. This formulation of the problem leads to a very large, high-dimensional solution space. It thus takes a very long time for the GA algorithm to find a layout geometry that gives the desired resonance frequency and also has a stiffness ratio of at least 10 between the desirable direction of oscillation (in the direction of the comb fingers) and the direction perpendicular to it. The GA search did indeed find solutions that met the above criteria. However, the crooked legs and asymmetrical layouts produced are not geometries that one would seriously consider for actual fabrication.

Even minimal experience with integrated circuit fabrication readily tells us that we want to use symmetry to cancel out to first order the effects of processing variations and of residual stresses in the surfaces of the suspension beams. Adding bilateral, or, better yet, 4-fold symmetry constraints to the GA search, reduces by a factor of four the number of parameters that need to be adjusted and thus the dimension of the search space. The run-times are dramatically reduced, solutions closer to the desired design goals are found, and the generated layout geometries are much more suitable for manufacturing.

On the other hand, an open-ended search using GA may find intriguing patterns that trigger some useful associations in a designer's brain. Emerging pheno-types may suggest spirals, serpentines, or frame structures that the designer may not have considered initially. While the patterns produced by the GA may

only be tenuous, the designer can readily distill out the new concept, and create simpler and more regular spirals, serpentines, or frames that are characterized with just a few geometrical parameters. Such new higher-level primitives can then be added to the library of components, from which the GA search or other optimization programs can draw, in order to further improve the best designs found so far (Fig.3).

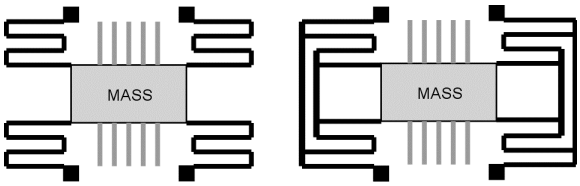

Figure 3. MEMS resonator suspension: (a) using 4-fold symmetry and parameterized serpentine springs, (b) using braces between serpentines to increase spring stiffness in the horizontal direction.

By introducing a specially coded serpentine element with two or more regular rectilinear hairpin turns, the resulting designs start to resemble more closely what an experienced designer might have started with in the first place (Fig.3a). Now the role of the search algorithm is to determine the parameters of these serpentine springs as well as their placement and orientation with respect to the central mass. This leads to some viable designs that are not too far off the desired resonance frequency and stiffness ratio.

Because of its sparse discrete sampling, a GA is unlikely to find a solution that meets the frequency requirement precisely. At this point it is preferable to introduce an inner loop with a continuous optimization procedure, such as gradient descent, to fine-tune the spring parameters so as to obtain the desired resonant behavior. These locally optimized designs, reached from different sampling points found by the GA, can now be evaluated with respect to other design objectives, such as minimizing layout area, or gaining robustness against manufacturing variations.

In applying this process, it became apparent that going from two to three serpentine loops could further reduce layout area, but it led to an unsatisfactory stiffness ratio. Adding cross braces between symmetrical pairs of serpentines (Fig.3b) dramatically improves the situation. It prevents the flare-out of the serpentines that occurs when the central mass moves along the undesirable axis. Design experience or engineering insight may invoke this modification of the basic suspension system. But there is no way that the GA can stumble across this solution, unless the genotype is capable of forming connected graphs with more complex connectivity between the spring elements than just four individual poly-lines. Such a genotype would make the search space even larger and further increase the time to find any viable solution. Also, such a change in the capabilities of the genotype may require a serious re-programming effort, which will disrupt the designer's creative thought process.

Ideally, I would like an interactive system that allows me to control the GA search while it is running. Through a graphical interface, I would like to select phenotypes that I like and enhance their "survival" value, and readily eliminate designs that I feel have no chance of leading to a useful solution. Some GA environments already provide such user interfaces [3]. In addition, I also would like to be able to sketch a promising looking modification to a specific phenotype and insert that individual into the gene pool, while the algorithm is running. This presents a much more challenging programming problem!

Perhaps an even more productive strategy for the first conceptual stage of design is to look for already existing solutions. Very often the problem at hand has been solved before, or an inspiring solution can be found in a related field. This suggests that we need much better support for such wide-spread searches over many engineering branches and databases. In the example above, the students who programmed the initial approach using GA could have found the much better approach depicted in Figure 3b by studying the many existing resonator designs described in the literature. We need better tools to make such a directed search practical for non-expert users.

In summary, I consider GA an interesting exploration tool to look for new conceptual ideas. But, so far, I am not aware of any practical solutions to any real-world engineering problem found by a GA that had not been known previously, or which could not be readily improved once an engineer takes a hard look at it. Evolution only produces adequate designs for a particular niche, not optimized solutions obeying hard constraints. However, combining GA with other search techniques and putting it all under human control may well be a winning approach for the future.

3. Development Phase

Once a conceptual solution has been found that promises to solve the problem at hand, the emerging design needs to be tailored to the exact specifications and constraints. Today, powerful optimization routines of many types can assist a designer in almost all domains of engineering.

3.1. Automated Layout of Operational Amplifiers

As an illustration I will discuss OPASYN [8], a program that produces custom-made IC (integrated circuit) layouts of an operational amplifier (op-amp)

tailored to a particular application. A typical op-amp may have anywhere from a 20 to more than a hundred discrete circuit elements (transistors, resistors, capacitors). An open-ended search over all possible collections of such elements and their interconnection topologies is thus out of the question. Fortunately, several decades of design experience with such circuits [6] have distilled out a logical approach that breaks such circuits into a few generic stages with well defined functions (voltage dividers, current mirrors, differential stages, source-follower buffers).

Furthermore the relationships between these clusters have been carefully choreographed so as to make viable amplifier stages. Thus the vast majority of all op-amp designs fall into a few well-established design classes, characterized by the total number of stages, and the generic types of input and output stages they employ (differential, single sided, push-pull). For each of these generic amplifier designs, there are only 5 to 8 key decisions to be made (the base current through a particular stage, the voltage swing desired at the output, etc), and these decision then define most directly the other parameters in the op-amp circuit. Thus when fine-tuning a new amplifier to a particular application, one has to find the right values for those 5 to 8 parameters and then follow well-established design practices to define the whole circuit in all its detail. This approach reduces this potentially very large design task to a search space of only 5 to 8 dimensions.

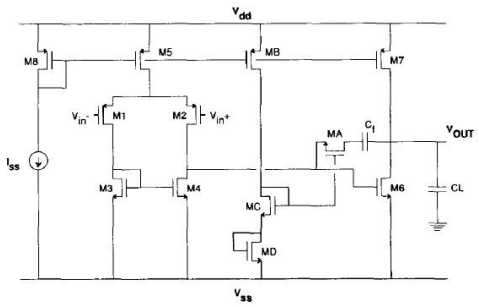

Figure 4. Basic two-stage op-amp.

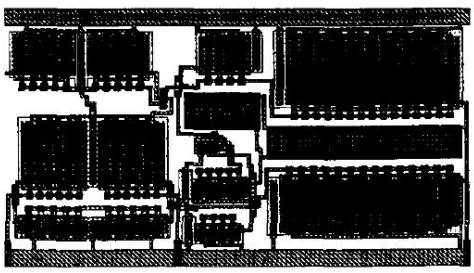

Figure 5. Synthesized amplifier layout.

For any such set of parameters, the other circuit elements can then be calculated, and the individual stages can be laid out as integrated circuit blocks. Again, some design experience is explicitly built into the program. The sensitive elements of the input stage are kept close together and are laid out in a symmetrical manner, so as to keep that stage balanced even in the presence of manufacturing uncertainties. The automated layout process allows for tailoring the floor plan to some fixed width or height, or for specifying a preferred aspect ratio.

Some exploration revealed that the cost function profile in this solution space is rather well-behaved, showing large smooth "hills" with only a few discrete basins with local optima. Thus for a given set of constraints dictated by the application, it was possible to sparsely sample that solution space and then refine these solutions with a gradient descent optimization to find all the local minima that might be compatible with the specifications. If there is more than one such "optimal" solution, they all get presented to the designer, who can then select one based on criteria which may not have been stated explicitly.

In this program, strong use was made of the vast engineering knowledge that had been built up over several decades in the design of analog integrated circuits [6]. But it took considerable engineering ingenuity to find out how to capture this knowledge in the computer, so that an automated design tool could be built around it. The open challenge is how to make this process easier, so it can be used in many other engineering applications. Clearly a similar approach should be applicable to the design of lens systems, car engines, bridges, or large exhibition halls.

3.2. Placement and Routing in Digital Circuits

There are other, less structured and less narrowly defined design problems, where the solution space is not as simple and smooth as in the case of OPASYN, and where thousands of local optima may exist. Such problems need a stronger stochastic component in their optimization phase. Noteworthy examples are placement and routing programs for digital ICs. Simulated annealing [7] has been used to produce impressive results [13]. With carefully selected move sets and suitable automatic cooling schedules, they can escape a particular local minimum in solution space and find better solutions based on layouts that might be quite different from the starting configuration.

Like genetic algorithms, this process may run through states that show some desirable partial solutions. However, these may get lost again, because they happen to be combined mostly with other non-viable features. That is the point where an experienced designer acting as an observer should be able to

intervene and mark such features as "highly desirable" that should not get lost. In the context of placement and routing, this situation is often encountered when a small change needs to be made at the periphery of the layout area, which acts as one of the constraints for the optimization for the layout task. Such small changes often change everything, and good partial designs in other regions of the layout are lost forever. I envision a graphical user interface that allows to mark some areas of the current solution as intangible, which means that their current perimeter must be added as further a constraint for the remaining layout task.

4. Refinement Phase

In stage 3 we assume that a design has progressed to a state where it is already lying in the proper valley of the solution space and where it can be moved to the closest local optimum with greedy optimization techniques. Once an engineering solution has been found that meets all the functional requirements, additional design effort may be spent to make the final product as attractive as possible, or to give it a distinctive style to represent a particular brand.

4.1. Smooth 3D Shape Optimization

To discuss issues of stage 3 of the design process, we look to the domain of 3D shape design. CAD tool development in this area has been dominated by the needs of content creation for the insatiable video game and film industries. The most challenging tasks are to make realistic looking animated human faces, and complex models of fluid simulation that (seem to) obey physical laws, yet are controllable enough to allow the film makers to stage the particular (disaster) scene that they have in mind. Clever combinations of search and optimization techniques have achieved amazing results in the domain of a flood washing through New York City [4] or to create a sequence of motions that moves a football player from one specified pose and motion state to another one [1].

Here I want to focus on the design of 3D shapes based on aesthetics, whether this concerns a vase, the hood of a car, or abstract geometric sculpture [12]. The key difficulty, of course, is that there is no absolute way to measure beauty. Tastes vary, and even experienced designers cannot express their preferences as explicit cost functions. The human designer must thus be integrated tightly into the design loop. This requires direct, real-time, interactive, continuous control of the displayed result.

4.2. Sculpture Generator I

A decade ago I implemented such a program with the very specific purpose of capturing a paradigm of coupled holes and saddles to design abstract geometrical sculptures, inspired by the work of Brent Collins [2]. Even though *'Sculpture Generator I'*, could produce "only" Scherk-Collins Toroids [11], it was well worth its development effort. Over the last 10 years I have made close to a hundred different shapes and sculpture models such as: '*Molecule'*, '*Totem_3'*, or *'Whirled White Web'* (Fig.6a), which has been realized as a 12-foot tall snow sculpture. In these shapes the appearance of what looks like minimal saddle surfaces is hard-coded as approximations based on hyperboloid surfaces. The key parameters were attached to sliders that allowed real-time interactive fine-tuning [15] of the sculptural shape. This allowed a designer to quickly explore many of different shapes in just minutes, – which rendered the computer an effective amplifier of the creative process [10].

Figure 6. *'Whirled White Web'* and *'Volution_5'*

In later sculptures I used closer approximations to minimal surfaces by relying on Brakke's surface evolver. This is an iterative optimization program that moves the vertices of a triangulated mesh in such a way as to minimize surface area or bending energy. The result was a series of *Volution* surfaces (Fig.6b). In this process, the interactive setting and modification of parameters and constraints has been lost. Clearly, I would like to see an environment that combines all these capabilities.

4.3. "Beauty Functionals"

In the *Volution* sculptures, the functional used to optimize the surface was inspired by nature. Minimal surfaces as assumed by soap films look very pleasing to most humans. However, to an artists it is too restrictive to stick with functionals found in nature. We can readily capture our own concepts of the crucial elements of beauty in different functionals. As an example consider the Minimum Variation Surfaces (MVS) (Fig.7) developed with Henry Moreton [9]. This functional is based on the premise that curvature should not be penalized a priori. The most perfect closed shape is a sphere, which thus should have an

overall penalty of zero. Only **deviations** from this highly regular shape should be penalized with extra cost. This led most naturally to the MVS cost functional, which integrates the square of the **change** in curvature over the surface of the shape.

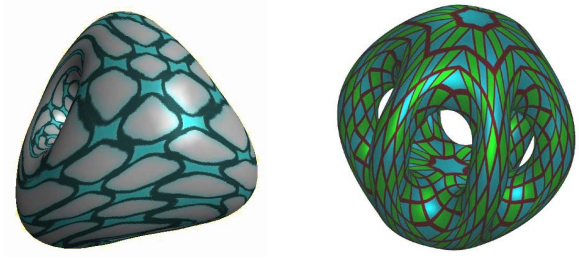

Figure 7. MVS-optimized surfaces.

Finding such surfaces requires some heavy-duty computation, but then allows a designer to specify a shape with just a few high-level constraints without the need to do any subsequent "fairing" of the surface. In a mature CAD system, I would like to see a whole arsenal of such beauty functionals, allowing the designer to select the appropriate one for a particular application, or even to pick different ones for different parts of a single design. What I envision here is a catalog of possible **shape styles**, similar to the style sheets found in desktop publishing. The designer could then apply the desired style to a region with a click of the mouse. The design process in stage 3 would then become an interactive play, adjusting some external constraints and trying out various beauty functionals.

5. Conclusions

Over the last three decades CAD tools have evolved from glorified drafting tools to sophisticated analysis and optimization programs. CAD tools will become even more sophisticated and will incorporate ever more powerful search, analysis, and optimization modules. However, the original dream of fully *Computer-Automated Design*, which as popular in the 1970's, has been realized in only a few niches of narrowly defined design tasks. The main reason is that every novel design challenge raises new issues, and requires one to draw on new ideas that have not previously been programmed into the available CAD tools. The level of general knowledge and learning abilities in CAD tools today is still extremely limited and is not sufficient to deal with these new situations in an autonomous way.

Thus, human intelligence will not disappear from the design process. Engineering experience and plain common sense will continue to play key roles. If these can be suitably integrated into the design process and into the individual CAD modules, it will have a huge pay-off in terms of better designs and shorter turn-around times. Almost every CAD module can benefit from an enhanced user interface that gives the designer additional control over the task that this module is performing. We believe that such a symbiosis between computer algorithms and human intelligence will yield a very powerful design environment that can give practical solutions superior to those that any single monolithic synthesis or optimization program can find.

Acknowledgements

This work is supported through NSF Grant CRR – DES/CC-0306557 and MICRO Research Grant 05-066.

References

[1] Arikan, O., and, D. A. Forsyth, Interactive Motion Generation from Examples. *Proc. of ACM SIGGRAPH'02*, Vol: 21, No: 3, pp 483--490, 2002.

[2] Collins, B., Evolving an Aesthetic of Surface Economy in Sculpture. *Leonardo* Vol 30, No 2, pp 85-88, 1997.

[3] Dawkins, R., *The Blind Watchmaker*. W. W. Norton & Co. New York, 1987.

[4] Emmerich R., *The Day After Tomorrow*. Movie by Fox Home Entertainement, 2004.

[5] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[6] Gray, P. G., and R. G. Meier, *Analysis and Design of Analog Integrated Circuits*. Wiley, New York, 1984.

[7] Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi, Optimization by Simulated Annealing. *Science Vol 220,* May 1983.

[8] Koh, H.Y., C. H. Séquin, and P. R. Gray: "OPASYN: A Compiler for CMOS Operational Amplifiers," *IEEE Trans. on CAD*, Vol 9, No 2, pp 113-125, 1990.

[9] Moreton, H., and C. H. Séquin., Functional optimization for fair surface design. *Proc. of ACM SIGGRAPH'92*, pp. 167-176. (1992).

[10] Séquin, C. H., Computer-Augmented Inspiration. *Proc. ISAMA'99*, San Sebastian, pp. 419-428, (1999).

[11] Séquin, C. H., Virtual Prototyping of Scherk-Collins Saddle Rings. *Leonardo* Vol 30, No 2, pp 85-88, 1997.

[12] Séquin, C. H., CAD Tools for Aesthetic Engineering. *JCAD* Vol 37, No 7, pp 737-750, June 2005.

[13] Shin, H., and A. Sangiovanni-Vincentelli, MIGHTY: A 'Rip-up and reroute' detailed router. *Proc IEEE ICCD*, pp. 10-13, 1986.

[14] Sims, K., Artificial Evolution for Computer Graphics. *Computer Graphics*, 25(4), July 1991, pp. 319-328.

[15] SLIDE design environment: http://www.cs.berkeley.edu/~ug/slide/docs/slide/spec

[16] SUGAR (MEMS simulator): http://www.bsac.eecs.berkeley.edu/cadtools/sugar/sugar

[17] Zhang, Y., R. Kamalian, A. M. Agogino, and C. H. Séquin, Hierarchical MEMS Synthesis and Optimization. *SPIE Conference on Smart Structures and Materials, Conf. 5763,* Mar. 2005, San Diego CA.