Aplicação de algoritmos evolutivos em circuitos eletrônicos

Caio Cestari Silva Paulo Henrique Floriano
Prof. Alfredo Goldman

14 de dezembro de 2009

Resumo

Este trabalho tem como objetivo estudar e analisar os resultados obtidos pelo artigo *Evolution of Robustness in an Electronics Design* [3]. Neste, são discutidas as aplicações de algoritmos evolutivos para a criação de um circuito eletrônico e a capacidade de tal abordagem de produzir circuitos cujas características não podem ser geradas a partir de nenhum outro método. Além disso, analisa-se a possibilidade de deixar com que a evolução ignore certas restrições dos padrões de design digital para atingir maior eficácia e robustez.

1 Introdução

Algoritmos genéticos são meta-heurísticas utilizadas para a resolução de problemas de busca ou otimização. Neste método, inicia-se com um conjunto de candidatos a solução (chamado população) e aplicam-se operadores de combinação e mutação sobre os elementos deste conjunto para obter uma nova geração, na qual só persistem soluções iguais ou melhores às da geração anterior. Repetindo este processo por muitas gerações, obtemos soluções cada vez mais próximas da ótima.

Diversos experimentos já foram feitos utilizando algoritmos genéticos em projeto de circuitos eletrônicos geralmente muito complexos para se projetar manualmente. Este processo produz resultados significativos pois independe dos preconceitos que dirigem o pensamento humano e explora candidatos a solução que uma pessoa poderia desconsiderar. A partir deste processo o artigo [2] relata que diversos circuitos patenteados no passado por mostrar boa funcionalidade foram redescobertos por algoritmos genéticos que buscavam soluções melhores para aqueles problemas.

O trabalho estudado relata um experimento em evolução de circuitos chamados *emergentes*, que são aqueles cujo comportamento não pode ser facilmente previsto com base apenas no conhecimento das componentes e suas interconexões. Este tipo de circuito não pode ser facilmente criado pelos métodos convencionais devido à necessidade de modelar o comportamento das componentes, porém, um algoritmo genético, que depende apenas da análise do resultado do circuito e não de seu comportamento, encontra mais facilidade. Desta forma, um circuito evoluído pode ultrapassar o escopo dos padrões atuais de design de circuitos eletrônicos.

O objetivo do experimento é evoluir um circuito para apresentar maior robustez, ou seja, a possibilidade de operar adequadamente sob variações especificadas de temperatura, voltagem e outros fatores físicos. A utilização das restrições dos modelos tradicionais de circuitos facilita a obtenção de robustez, característica geralmente ignorada por circuitos evoluídos a partir de algoritmos genéticos. No artigo estudado [3], mostra-se uma forma de utilizar a evolução para obtermos circuitos robustos e suas consequências. Apresentaremos tais resultados no decorrer deste trabalho.

Nas próximas seções, descrevemos em detalhes as características e o funcionamento dos algoritmos genéticos, bem como sua utilização na criação de circuitos eletrônicos. Em seguida, detalhamos o experimento realizado no artigo estudado.

2 Algoritmos Genéticos

Um algoritmo genético é uma meta-heurística utilizada para resolução de problemas de busca e otimização. Por tratar-se de uma meta-heurística, um algoritmo genético pode ser aplicado a qualquer tipo de problema que se encaixe na modelagem. Esta técnica baseia-se no princípio biológico da evolução pois modela o espaço de busca através de populações com diferentes genótipos (características) e suas interações através de reproduções, mutações e recombinações de genes. As principais etapas deste tipo de abordagem são:

- **Inicialização** - Gera-se uma população inicial escolhida aleatoriamente;
- **Passo** - A partir da população atual, construímos a próxima levando em conta os indivíduos mais adaptados (soluções com melhor avaliação), e;
- **Terminação** - O algoritmo para quando atingimos uma população com soluções aceitáveis ou quando atingirmos um certo número de gerações.

Podemos utilizar qualquer tipo de estrutura para representar uma solução; a representação mais comum é por meio de uma sequência de *bits*. Esta forma é boa pois é bastante concisa e facilita a comparação e avaliação das soluções.

Para comparar as soluções de um dado problema, construímos uma função de adaptação (*fitness*) que avalia cada indivíduo da população por meio de um número real. Quanto maior o valor de adaptação, melhor é a solução. O cálculo da função de adaptação pode levar muito tempo, dependendo da representação utilizada para as soluções, o que torna a escolha da avaliação da população uma das principais dificuldades na aplicação de algoritmos genéticos.

Inicialmente, gera-se um conjunto aleatório de candidatos a solução (indivíduos) para formar a primeira geração. Se for conhecida alguma informação sobre o domínio que permita inferir o local onde as soluções ótimas têm maior probabilidade de serem encontradas, é possível limitar a população inicial a candidatos próximos deste local.

A partir da população inicial e das subsequentes, temos que obter os indivíduos da próxima geração. Para isto, utilizam-se os processos de seleção, mutação e reprodução,

tomando como base a função de adaptação aplicada a cada indivíduo. O processo de seleção consiste na escolha de certos candidatos a solução da população atual para fazer parte da nova geração. Normalmente, os indivíduos com maior valor de função de adaptação têm maior probabilidade de serem selecionados. Nos casos em que a população é muito grande e, portanto, avaliar todas as soluções é muito custoso, pode-se colher uma amostra menor para realizar a seleção.

Além da seleção, outra forma de compor a próxima população é através de mutações. Neste processo, são escolhidos alguns indivíduos ao acaso (independentemente do valor da função de adaptação) para sofrerem pequenas modificações e passarem para a geração seguinte. Se a solução está representada por meio de um *array* de *bits*, a mutação geralmente escolhida é a inversão de um destes *bits*. Por fim, o processo de reprodução consiste em gerar indivíduos da próxima geração através de um processo de combinação genética entre dois ou mais indivíduos da geração atual. A forma usual de reprodução de soluções (em *arrays* de *bits*) é selecionando um ou dois pontos de *crossover*, onde ocorre a quebra dos *arrays* e a devida recombinação das partes.

Ao final destes três processos, teremos uma população nova, na qual a média de *fitness* é geralmente maior que a da geração anterior. Apesar disso, mantem-se uma parcela dos indivíduos candidatos à solução que apresentam valores pequenos da função de avaliação. Age-se desta forma pois tais candidatos podem ser modificados, ainda que ligeiramente, gerando soluções com valores maiores de *fitness*, superiores inclusive às que aparentavam ser melhores.

Exemplo: o problema da mochila (booleana) pode ser modelado e resolvido utilizando-se um algoritmo genético. Sabemos que a mochila possui capacidade máxima W, e que temos n produtos, indivisíveis, com pesos $w_1, w_2, \ldots, w_n$ e os valores $v_1, v_2, \ldots, v_n$, associados a cada produto. Toma-se um vetor x de dimensão n, onde cada índice i representa um produto, no qual:

$$x_i = \begin{cases} 1, & \text{se o produto } i \text{ não é escolhido} \\ 0, & \text{caso contrário} \end{cases}$$

O objetivo é maximizar o valor de $\sum_{i=1}^{n} x_i v_i$, sem que $\sum_{i=1}^{n} x_i w_i$ seja maior que a capacidade W da mochila.

Para modelar este problema de forma favorável à resolução com um algoritmo genético, precisamos definir a representação de uma solução e uma função de adaptação. A representação pode ser feita com uma sequência de *bits*, na qual o i-ésimo *bit* é equivalente a x_i. A função de avaliação é definida da seguinte maneira:

$$f(x) = \begin{cases} \sum_{i=1}^{n} x_i v_i, & \text{se } \sum_{i=1}^{n} x_i w_i \leq W \\ 0, & \text{caso contrário} \end{cases}$$

Com esta representação, podemos utilizar um algoritmo genético para resolver o problema da mochila.

3 Design Eletrônico Restrito e Irrestrito

No processo de desenvolvimento de circuitos eletrônicos, existem alguns paradigmas que devem ser seguidos para que o comportamento desejado do produto final seja atingido e seu funcionamento não seja facilmente afetado por ruídos externos. Estas restrições de *design* são particularmente úteis na criação manual de circuitos (não automatizada), pois permitem uma maior facilidade em sua modelagem e avaliação. Existem dois padrões principais de *design* eletrônico: digital e analógico.

Num circuito digital, toda vez que um sinal é utilizado por uma componente, ele é arredondado para um dos extremos de voltagem (geralmente 0 e 1). Deste modo, um sinal só pode ser transmitido de um elemento para outro do circuito se estiver em um dos estados extremos. Esta abordagem permite que um circuito seja modelado através de uma álgebra booleana, podendo ignorar quase que completamente a interação física das componentes com o meio externo. Além disso, circuitos digitais são extremamente tolerantes a corrupção de sinais, uma vez que este é restaurado a um dos valores extremos em quase todas as oportunidades. Mas, estas vantagens vêm com o preço das fortes restrições que este modelo impõe para simplificar a realidade.

Um modelo um pouco mais complexo é o analógico. Neste, os sinais não são mais arredondados; sua natureza contínua é preservada. Esta abordagem permite que uma gama maior de circuitos sejam criados, pois as restrições são diminuídas consideravelmente. Porém, isto gera certas dificuldades, como a grande complexidade das relações entre as componentes e a baixa tolerância a falhas devido aos sinais contínuos. Por isso, o design de circuitos analógicos geralmente fica restrito ao uso de blocos pré-fabricados.

Se considerarmos um espaço de busca no domínio dos circuitos eletrônicos, os *designs* mencionados abrangeriam apenas uma pequena parte dele. Apesar disto, é muito difícil para um humano desenvolver circuitos fora das convenções tradicionais. Para procurar circuitos em locais desconhecidos deste espaço de busca é uma tarefa que apenas uma ferramenta automática de design poderia realizar.

4 Evolução no Design Eletrônico

Tendo como objetivo gerar bons circuitos automaticamente para certas aplicações, surgiu a ideia de utilizar algoritmos genéticos. Esta abordagem permite que uma grande quantidade de circuitos sejam criados e testados sob diversas condições rapidamente, permitindo que o circuito gerado além de ter bom funcionamento, consiga uma certa margem de tolerância a falhas.

A evolução é muito poderosa quanto a exploração do espaço de busca. Este poder cresce ainda mais se as restrições de *design* convencionais puderem ser ignoradas no desenvolvimento das gerações. Com a possibilidade de buscar circuitos que não poderiam ser gerados de nenhuma outra maneira, a evolução irrestrita pode explorar *assíncronos* cujo comportamento não pode ser facilmente previsto, aumentando as chances de que seja encontrado um circuito adequado.

Como mencionado anteriormente, para se modelar um problema para que possa ser resolvido com um algoritmo genético, devemos definir três fatores: a representação genética de uma solução, a função de adaptação calculada sobre cada indivíduo e a forma de obtenção de uma nova geração, a partir da população atual.

Nos trabalhos que mostram a utilização de algoritmos genéticos para evolução de circuitos eletrônicos, a representação genética de um circuito é o próprio. Esta representação não é compacta, mas é adequada devido à forma de cálculo da função de adaptação. Diferentemente dos problemas tradicionais, a função de *fitness* não depende do formato das conexões internas do circuito, mas apenas de sua resposta a alguns padrões de entrada. Geralmente, alimenta-se o circuito com um conjunto aleatório de entradas e calcula-se seu valor de adaptação baseado nos valores da saída.

A modelagem da transição de gerações varia conforme o experimento. Alguns deles utilizam todos os operadores já mencionados, outros deixam alguns, como o cruzamento, de lado. No experimento estudado, apenas as mutações foram utilizadas, pois a população utilizada possui apenas um indivíduo. Geralmente, uma mutação consiste na mudança da sequência reconhecida por um dos multiplexadores do circuito.

5 O chip *FPGA*

Um chip costuma vir pré-programado, mas e se não for esta a necessidade do usuário para um certo aparelho? Surge a idéia dos hardwares configuráveis e reconfiguráveis, abrindo caminho para a programação de portas lógicas em uma estrutura conhecida como FPGA, *Field Programmable Gate Array*. Semicondutor, criado pela Xilinx Inc., é composto por:

- **Blocos Lógicos Configuráveis**: reunião de flip-flops, de 2 a 4, com lógica combinacional;
- **Blocos de Entrada e Saída**: interface das saídas dos blocos lógicos configuráveis, funcionando como buffers, e;
- **Chaves de Interconexões**: trilhas que conectam os dois blocos em redes apropriadas, utilizando funções lógicas.

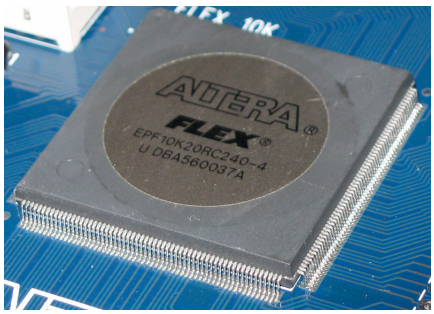

Figura 1: Um chip FPGA

Os circuitos que podem ser desenhados numa FGPA vão desde alguns blocos até grandes proporções, e as células são capazes o suficiente para implementar funções lógicas e rotear a comunicação entre elas. Tal comunicação possui uma arquitetura, que posiciona barramentos e chaves de interconexões de forma a abrigar os blocos lógicos, e que deve permitir comunicação completa e grande densidade de portas lógicas.

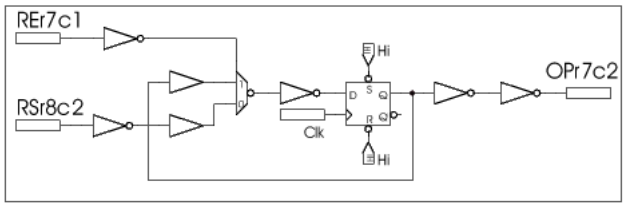

Figura 2: Um bloco lógico de uma FPGA

Neste meio, estão envolvidos conceitos como pinos, conexões, redes, blocos de comutação, segmentos de trilhas, canais de roteamento, blocos de conexão, dentre outros, e é sobre essa estrutura lógica do FPGA que se operou o experimento que será descrito a seguir.

6 O Experimento

6.1 Descrição e Estrutura

O experimento consiste em, dada uma onda quadrada de 1kHz ou 10kHz, determinar a tal frequência da entrada pela sua tensão de saída, que oscila entre extremos altos e baixos, identificando cada uma. Dada esta estrutura inicial, a partir da qual será aplicado o algoritmo genético, objetiva-se que cada candidato a solução seja testado por quatro diferentes *chips* FPGA XC6216, com diferentes fabricantes, interfaces, temperaturas, PSUs (*Power Supply Units*), etc. Dentre os quatro experimentos, o que apresentasse pior valor seria determinante para a avaliação final de *fitness* daquela geração.

Uma FPGA possui 64 x 64 células, mas apenas 10 x 10 foram utilizadas no processo de evolução. Para cada *chip* diferente, foi utilizada uma região distinta, apenas ativando as conexões entre as células envolvidas no processo. Os 10 multiplexadores localizados nas células possuem a seguinte distribuição:

- Quatro multiplexadores 4:1, entre células vizinhas;
- Três multiplexadores X, que selecionam as entradas para o funcionamento de cada célula;
- Dois multiplexadores 4:1, de funcionamento interno, e;
- Um multiplexador seletor 2:1 CS (*Combinacional* ou *Sequencial*).

Entra-se, então, com um sinal de *clock* para todas as *FPGAs*, originado de um cristal único externo com frequência nominal de 6MHz. Sabe-se que cada célula pode ser síncrona ao clock, assíncrona ao *clock* ou ainda mista, baseando-se na utilização ou não da saída do *flip-flop*; porém, no processo evolutivo não se produz um design síncrono, pois não há regras sobre o mesmo impostas para as modificações de uma geração para a outra.

6.2 O Algoritmo Evolutivo

Um operador de mutação seleciona uma dentre as 100 células, de forma aleatória, e então um dos seus multiplexadores, também de forma aleatória, reconfigurando-o para

selecionar uma entrada diferente. Na técnica conhecida como Estratégia Evolutiva, aplica-se a mutação três vezes a cada nova variante a ser produzida. Então, um oscilador de um milhão de MHz dispara frequências de 1kHz e 10kHz, aleatoriamente, com um intervalo de 100 milissegundos, de forma que haja números iguais de disparo para ambas. Circuitos análogos integram os resultados de cada chip e calculam uma média para a tensão de saída nesse intervalo.

Chip	Fabrication	Package	Interface	Temperature	PSU	Output load	Position
1	Seiko	PQFP	parallel	in PC	PC's 5V s.m.	-	(37,30)
2	Yamaha	PLCC	serial	ambient	5V lin.	1kΩ	(32,0)
3	Yamaha	PGA	serial	60°C	4.75V s.m.	-	(63,0)
4	Seiko	PGA	serial	−27°C	5.25V s.m.	-	(37,54)

Figura 3: Chips utilizados no experimento, com suas características nominais

Seja um chip dito como "integrador" do circuito, denominado c, considerado ao final do sinal de teste t, representado por i_t^e, com $(t = 1, 2, ...T)$, e seja S_1 o conjunto dos sinais de 1kHz e S_{10} o conjunto dos sinais de 10kHz. A função de avaliação da configuração de c é calculada como:

$$E^c = \frac{1}{2T} \left| \sum_{t \in S_1} i_t^c - \sum_{t \in S_{10}} i_t^c \right| - P^c$$

A função de *fitness* F é considerada como o $min_{c=1}^{4}(E^c)$. Há, além disso, uma função de penalização para os circuitos que apresentam ruídos na saída (oscilações não esperadas), que considera, para cada *FPGA*, um contador de transições lógicas de alto para baixo a cada pulso. Se esse número no sinal de teste t é dado por $N^c(t)$, então o contador retornaria um valor dado por $R^c(t)$, que é o mínimo entre o número de transições lógicas e 255, uma constante. Dessa forma, define-se P^c, para w um peso (inicialmente nulo), como:

$$P^c = w \text{ x } \frac{1}{2T} \sum_{t=1}^{T} max(R^c(t) - 1, 0)$$

A base do algoritmo é a existência de caminhos que podem não melhorar imediatamente o valor de *fitness*, mas que podem ser responsáveis por grandes alterações em gerações futuras. Os pais iniciais são encontrados por meio de uma busca aleatória, com um ponto inicial de performance melhor que a trivial. É necessário utilizar uma reconfiguração

parcial pela mutação pois algumas FPGAs são reconfiguradas por meio de uma interface serial lenta. O critério de parada é definido pelo usuário, e esse é o ciclo do algoritmo genético aplicado aos circuitos *FPGA*.

Obervações sobre o algoritmo: alguns valores e configurações, como a dos pais iniciais, são encontrados por meio de testes aleatórios e independentes, tendo como base o resultado do *fitness*. Tais configurações provaram ser eficazes após várias gerações, reduzindo o número de ruídos e distorções na saída e eliminando os indivíduos considerados "não adaptados".

6.3 Resultado e Análise do Experimento

O circuito, surpreendentemente, funcionou bem em todos os modelos e condições diferentes a que foi submetido. Há apenas um ponto de melhora: quando altera-se a frequência de entrada, são necessários alguns ciclos até que a saída se modifique, em vez de um simples meio-ciclo. O tempo não influiu negativamente na robustez: após 24 horas, não houve transições indesejadas. Nem a queda de de temperatura até o os -50^oC foi suficiente para influir no funcionamento do circuito, que continuava intacto.

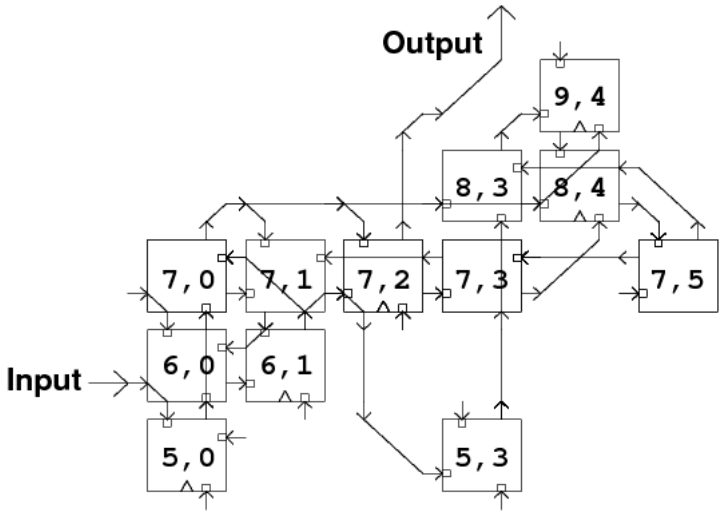

Figura 4: Parte funcional do circuito gerado

Foi verificado, após a geração do modelo final, se este representava algum modelo de operação lógica digital conhecido, aplicando-o a um simulador amplamente utilizado, o *PSpice*. Utilizando sinais sem ruído, binários, e com uma descrição da propagação interna

(para cada componente) de suas entradas até sua saída, foi possível reproduzir um bom funcionamento do circuito sem a necessidade de um ajuste mais fino das configurações do simulador. Em contrapartida, a simulação foi executada um pouco mais lentamente: em um Pentium 233MHz, apresentou mais de 60 mil vezes mais demora que se executado no mundo real.

O mecanismo central do circuito consiste em uma oscilação diferente para cada frequência de entrada: o oscilador pode variar um número ímpar de vezes, dada uma entrada, terminando em um estado diferente do inicial, ou não. Tal mecanismo utiliza apenas três células, tendo as outras o papel de atrasar a entrada realimentada, corrigindo as situações em que a saída é constante para uma entrada e alternada a cada ciclo para a outra.

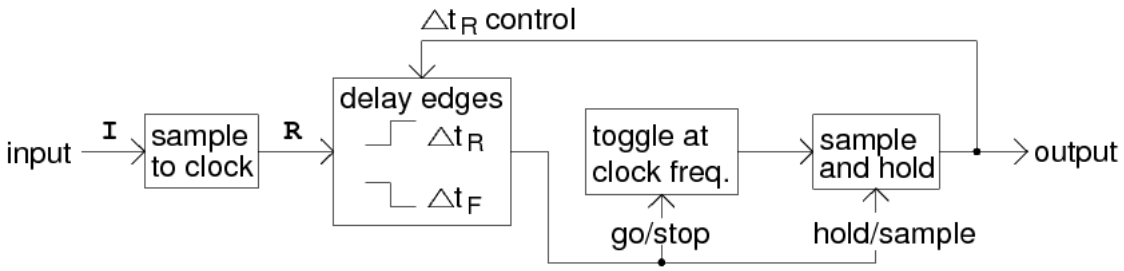

Figura 5: O mecanismo central do circuito

O simulador, além disso, tem a habilidade de apontar quando há uma falha de flip-flop, isto é, quando o sinal de entrada muda mais rapidamente que o sinal do clock, necessário para produzir a saída que torna-se incerta. Apenas ocorrem falhas deste tipo na célula (5,0), onde há entrada realimentada.

7 Conclusão

A partir deste curioso experimento, podemos perceber que nosso conhecimento sobre os circuitos lógicos é bastante limitado. Utilizando um algoritmo bastante simples, implementado em uma placa com número pequeno de transístores (se comparado a um processador atual, por exemplo), foi possível a geração de um circuito cujo comportamento ainda não foi entendido por completo.

Outro ponto interessante a ser notado é a eficiência dos algoritmos genéticos na geração

automática de circuitos eletrônicos. Mesmo uma variante bastante rudimentar deste algoritmo (com apenas um indivíduo na população e mutações simples) conseguiu chegar a um circuito final cujo comportamento é quase perfeito para o contexto, salvo pequenos ruídos. Esta eficiência se deve ao fato de que os algoritmos genéticos não são viesados, ou seja, não priorizam uma solução sobre outra baseando-se em fatores que não a função de *fitness*. Deste modo, é possível uma exploração mais profunda do espaço de busca, encontrando soluções em locais inacessíveis pelos métodos convencionais de *design*.

Para uma busca ainda mais abrangente, utilizando a evolução, uma estratégia utilizada é abandonar as convenções tradicionais de *design* de circuitos eletrônicos. Estas, como visto, impõem diversas restrições que, apesar de fundamentais para o entendimento do funcionamento e manuseio do circuito por um ser humano, são desnecessárias para o algoritmo genético e acabam apenas limitando sua gama de possibilidades.

Em suma, nós só conseguimos entender bem uma quantidade ínfima de circuitos eletrônicos, se compararmos com o tamanho deste domínio. Meta-heurísticas são uma boa escolha na abordagem de um problema deste tipo, pois são independentes de nosso conhecimento limitado.

Referências

[1] I. Harvey and A. Thompson. Through the labyrinth evolution finds a way: A silicon ridge. *Lecture Notes in Computer Science*, 1259:406–422, 1997.

[2] J.R. Koza, F.H. Bennett III, M.A. Keane, J. Yu, W. Mydlowec, and O. Stiffelman. Searching for the impossible using genetic programming. In *Proc. Genetic and Evolutionary Computation conference (GECCO-99)*, pages 1083–1091.

[3] A. Thompson and P. Layzell. Evolution of robustness in an electronics design. In *Evolvable systems: from biology to hardware: third international conference, ICES 2000, Edinburgh, Scotland, April 17-19, 2000, proceedings*, page 218. Springer Verlag, 2000.

[4] A. Thompson, P. Layzell, and R.S. Zebulum. Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, 1999.