Adv. Radio Sci., 6, 195–199, 2008
www.adv-radio-sci.net/6/195/2008/

Advances in Radio Science

Synthese von analogen Filtern auf einer rekonfigurierbaren Hardware-Architektur mittels eines Genetischen Algorithmus

S. Trendelenburg, J. Becker, F. Henrici, and Y. Manoli

Lehrstuhl für Mikroelektronik, Institut für Mikrosystemtechnik (IMTEK), Albert-Ludwigs-Universität Freiburg, Germany

Zusammenfassung. Rekonfigurierbare Analog-Arrays (FPAAs) sind der Versuch, die Vorteile der aus der digitalen Welt bekannten FPGAs (Flexibilität, Entwurfsgeschwindigkeit) auch für analoge Anwendungen verfügbar zu machen. Aufgrund der Vielfalt der analogen Schaltungstechnik ist die Abbildung von vorgegebenen Schaltungskonzepten auf eine FPAA-Architektur nicht immer einfach lösbar.

Diese Arbeit stellt einen neuen Ansatz für die Synthese von Filtern auf einer FPAA-Architektur für zeitkontinuierliche Analogfilter mittels eines Genetischen Algorithmus (GA) vor. Anhand eines Matlab-Modells des FPAA, das eine gute Übereinstimmung mit Simulationen des FPAA auf Transistorebene aufweist, wurde gezeigt, dass eine große Vielzahl verschiedener Filterstrukturen auf dieser Architektur dargestellt werden kann. Daraufhin wurde ein Genetischer Algorithmus entwickelt, der es erlaubt, aus einer gegebenen Filterspezifikation Konfigurationsdatensätze zu synthetisieren, die den gewünschten Filter auf die FPAA-Architektur abbilden.

1 Einleitung

Die Filterung von analogen Signalen stellt auch im Zeitalter der zunehmenden Digitalisierung ein wichtiges Anwendungsgebiet für Analogschaltungen dar. Die Einsatzgebiete reichen hierbei von Sensorschnittstellen bis zur drahtlosen Datenübertragung. Trotz der großen Verbreitung von digitalen Filtern und Signalprozessoren besteht immer noch ein großer Bedarf an schnellen, zeitkontinuierlichen Analogfiltern. Anders als bei zeitdiskreten Methoden kann hier auf hohe Überabtastraten verzichtet werden, was höhere Signalbandbreiten erlaubt und den Energieverbrauch senkt.

Rekonfigurierbare Hardware-Architekturen sind in der digitalen Schaltungstechnik v.a. in der Form von FPGAs weit verbreitet. Zu ihren Vorteilen gehören neben der Flexibilität die Geschwindigkeit und schnelle Rekonfigurierbarkeit, was sie vor allem für Rapid-Prototyping-Anwendungen prädestiniert. Seit einiger Zeit bestehen Bestrebungen, diese Vorteile auch für die analoge Schaltungstechnik nutzbar zu machen, in der Form so genannter Feldprogrammierbarer Analog-Arrays (FPAAs).

FPAAs unterschieden sich von ihrem Aufbau her in solche, die größte Flexibilität durch eine Vielzahl von kleinsten rekonfigurierbaren Elementen, meist einzelne Transistoren, erreichen und solchen, die wenigere, jedoch spezialisierte Elemente, wie z.B. Operations- oder Transkonduktanzverstärker, als aktive Grundelemente verwenden (Hall et al., 2005). Die dieser Arbeit zugrundeliegende Architektur gehört zur letzteren Gruppe, und ist für die Instantiierung zeitkontinuierlicher analoger Filter geeignet.

Für den Einsatz von FPAAs als rekonfigurierbares Filterelement in Rapid-Prototyping-Anwendungen spielt die Komplexität der Handhabung eine wichtige Rolle. Es ist wünschenswert, dem Anwender die Möglichkeit zu geben in kurzer Zeit und ohne tiefergehendes Wissen über die zugrundeliegende Theorie einen für eine vorliegende Spezifikation geeigneten Filter auf dem FPAA zu synthetisieren. Es besteht zwar die Möglichkeit, Filter-Grundstrukturen vorzugeben und diese später nur noch durch deren Kombination und Parametervariation anzupassen, jedoch wird dies der Flexibilität der FPAA-Architektur nicht gerecht. Diese Arbeit beschäftigt sich daher mit dem Ansatz, Filterstrukturen auf einem FPAA aus einer vorgegebenen Spezifikation automatisch zu synthetisieren.

2 Hardware-Architektur

Bei der zugrundeliegenden FPAA-Architektur (Becker and Manoli, 2004) handelt es sich um eine hexagonale Grundstruktur aus sogenannten Configurable Analog Blocks (CABs) als Grundelementen. Abbildung 1 zeigt einen FPAA mit 7 CABs. Für größere Arrays kann die Struktur in alle Richtungen fortgesetzt werden. Die CABs sind mit den Ein- und Ausgängen sowie untereinander durch digital programmierbare Transkonduktanzverstärker (G_m-Zellen) verbunden. Diese stellen zusammen mit der parasitären Kapazität am zentralen Knoten einer CAB die Grundelemente zum Aufbau von Filterstrukturen in G_m-C Technik dar.

Correspondence to: S. Trendelenburg (stanis.trendelenburg@imtek.uni-freiburg.de)

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

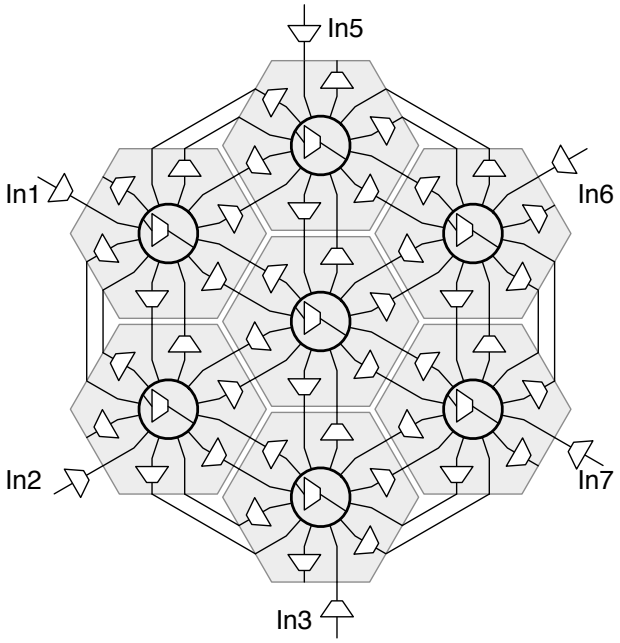

Abb. 1. Schematische Ansicht des FPAA mit 7 CABs.

Durch die hexagonale Form ist es möglich, sowohl lineare G_m-C-Filterstrukturen als auch solche mit Feedback gerader oder ungerader Ordnung auf dem FPAA zu instantiieren.

Die Verstärkung der G_m-Zellen lässt sich in 13 diskreten Schritten von -6 bis $+6\,g_m$ einstellen, wobei g_m die Einheitsverstärkung bezeichnet. Die Einstellung 0 entspricht dem Ausschalten der G_m-Zelle. Da jede CAB mit ihren Nachbarn bidirektional über solche Zellen verbunden ist, ist kein separates Routing-Netzwerk nötig, um die Signale von CAB zu CAB zu leiten. Zur genaueren Charakterisierung der G_m-Zelle sei auf (Henrici et al., 2007) verwiesen.

3 Modellierung

Um verschiedene Ansätze zur Filtersynthese auf der vorgestellten FPAA-Architektur effizient untersuchen zu können, wurde zunächst ein Matlab-Modell des FPAA erstellt. Da zum Zeitpunkt dieses Teils der Arbeit noch kein Prototyp des Chips verfügbar war, wurde das Modell zunächst anhand von Daten aus einer Simulation des Chips auf Transistor-Ebene verifiziert.

Ausgehend von der idealen Beschreibung der G_m-C Kombination als Integrator mit vorgeschaltetem Verstärker wurde eine Matlab-Modell des elementaren Filterelements erstellt. Dieses wurde um die wichtigsten nichtidealen Effekte, in Form des parasitären Pols und des Eingangs- und Ausgangswiderstands der G_m-Zelle ergänzt. Die Ein- und Ausgangswiderstände sind zudem von der eingestellten Verstärkung

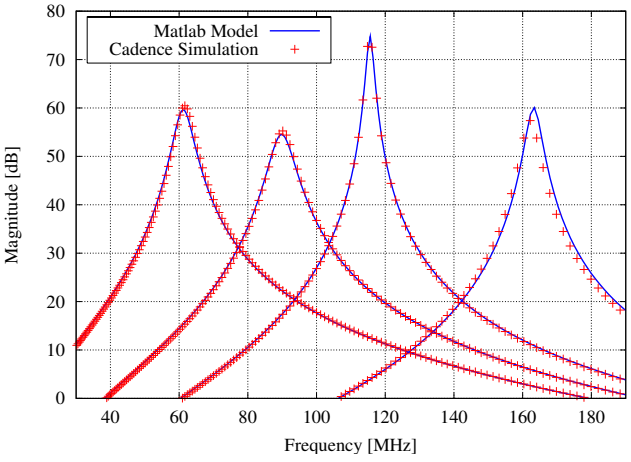

Abb. 2. Vergleich zwischen Matlab-Modell und Cadence-Simulation anhand verschiedener Bandpässe 4. Ordnung.

abhängig. Als weiterer nichtidealer Effekt wurde der parasitäre Pol der G_m-Zelle bei der Frequenz ω_2 berücksichtigt.

$$T(s)_{G_m} = n \cdot G_m \cdot \frac{1}{\frac{s}{\omega_2} + 1} \tag{1}$$

$$T(s)_{C_L||r_{\text{out}}} = \frac{1}{s \cdot C_L + \frac{n}{r_{\text{out}}}} \tag{2}$$

Wird die G_m-Zelle eingebettet in der Struktur des FPAA betrachtet, müssen die Ausgangswiderstände aller mit dem Knoten verbundenen G_m-Zellen berücksichtigt werden: Dies geschieht durch den Faktor $\Sigma n'$, der die Summe der Parameter n aller mit diesem Knoten verbundenen G_m-Zellen darstellt.

$$T(s) = n \cdot G_m \cdot \frac{1}{\frac{s}{\omega_2} + 1} \cdot \frac{1}{s \cdot C_L + \frac{\Sigma n'}{r_{\text{out}}}} \tag{3}$$

Das Modell für eine gegebene Filterkonfiguration kann nun in Matlab programmatisch aufgebaut werden, indem die elementaren Übertragungsfunktion für G_m-Zelle und Kapazität der Konfiguration entsprechend erzeugt und miteinander zur Gesamtstruktur verbunden werden. Man erhält so ein LTI-Modell des FPAAs für diese spezifische Konfiguration, aus dem sich das Übertragungsverhalten zwischen beliebigen Ein- und Ausgängen extrahieren läßt. Die benötigte Rechenzeit zum Erzeugen des Modells liegt dabei im Bereich von einigen Sekunden, während die Simulation des Frequenzgangs auf Transistorebene mehrere Stunden in Anspruch nimmt. Abbildung 2 zeigt die gute Übereinstimmung des Matlab-Modells mit der Simulation auf Transistorebene anhand eines Bandpasses 4ter Ordnung.

4 Filtersynthese mit einem Genetischen Algorithmus

Die klassische Herangehensweise zur Filtersynthese sieht zunächst die Auswahl einer geeigneten Filterfunktion und deren Parameter aus Tabellen, und in einem weiteren Schritt die Realisierung der ausgewählten Funktion in der gewünschten Technologie vor. Probleme ergeben sich dabei vor allem durch zwei Effekte: Zum einen können die Filterparameter aufgrund der diskreten Abstufung der G_m-Zellen-Parameter nicht immer exakt getroffen werden. Zum Anderen müsste, um die Funktion des instantiierten Filters genau vorherzusagen, der Pol der G_m-Zelle bei der Modellierung mit berücksichtigt werden, da dieser abhängig von der Filterfunktion einen mehr oder weniger großen Einfluß auf die Gesamtübertragungsfunktion hat. Dies würde jedoch die Ordnung des Systems extrem erhöhen und eine direkte Abbildung von bekannten Filtern erschweren, da kein nicht-integrierendes Grundelement mehr zur Verfügung stehen würde.

Der hier vorgestellte Ansatz unterscheidet sich darin, daß er nicht von der theoretisch besten Filterfunktion ausgeht, und dann versucht diese so gut es geht auf die bestehende Architektur abzubilden. Statdessen wird versucht, in einem iterativen Optimierungsprozess den FPAA so zu konfigurieren, daß er die Spezifikation unter Berücksichtigung aller zur Verfügung stehenden Möglichkeiten möglichst gut erfüllt.

Die Synthese eines Filters anhand einer gegebenen Spezifikation kann als Optimierungsproblem betrachtet werden, wenn man bedenkt, daß die Funktion einer instantiierten Filterstruktur durch die FPAA-Konfiguration vorgegeben ist, die als Matrix von 49 Werten aus dem Bereich -6 bis $+6$ vorliegt, welche die Einstellung jeder G_m-Zelle auf dem FPAA festlegt. Durch Messung oder Berechnung der Übertragungscharakteristik des Filters und den Vergleich mit der Spezifikation lässt sich jeder Konfiguration ein Wert E zuordnen, der den Fehler gegenüber der Spezifikation beschreibt. Das Optimierungsproblem besteht somit darin, eine FPAA-Konfiguration C zu finden, für die der Fehler E minimal wird.

$$C = \begin{pmatrix} n_{11} & n_{12} & \dots & n_{1j} \\ n_{21} & n_{22} & \dots & n_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ n_{i1} & n_{i2} & \dots & n_{ij} \end{pmatrix}; \quad \begin{aligned} &\text{CAB: } i = 1, 2, \dots, 7 \\ &G_m\text{: } \quad j = 1, 2, \dots, 7 \\ &\qquad\quad n \in \{-6, \dots, +6\} \end{aligned} \tag{4}$$

Genetische Algorithmen (GAs) fallen in die Kategorie der Globalen Such- oder Optimierungsalgorithmen. Sie werden meist für Probleme eingesetzt, für die andere Optimierungsverfahren keine guten Lösungen finden. Einfachere, gradientenbasierte Verfahren liefern z.B. keine guten Ergebnisse, wenn der abzusuchende Parameterraum sehr groß und nichtlinear ist. GAs zeichnen sich dadurch aus, daß sie vergleichsweise robust gegenüber lokalen Maxima im Suchraum sind und außerdem keinerlei Eigenschaften der zu Optimierenden Funktion wie z.B. Differenzierbarkeit oder Stetigkeit voraussetzen.

Grundlage der Optimierung ist die FPAA-Konfiguration, im GA-Jargon auch als „Individuum“ bezeichnet. Durch die Evaluation der Übertragungsfunktion, die die von einer Konfiguration dargestellte Struktur auf dem FPAA besitzt, kann ihr ein Fehler zugeordnet werden. Das Inverse des Fehlers wird auch als Fitnesswert F bezeichnet. Da die Optimierung ausschließlich anhand dieses Wertes erfolgt, müssen sich alle Eigenschaften des Zielfilters, wie Stabilität und möglichst geringer Platzbedarf, in der Berechnung dieses Wertes wiederfinden. Dies geschieht durch eine gewichtete Summe, die den quadratischen Fehler, gemittelt über n_s Samples, der Abweichung im Frequenzverhalten zwischen Spezifikation S und der betrachteten Konfiguration K , die Anzahl aller aktiven G_m-Zellen n_{Gm} und die Anzahl der Pole auf oder rechts der imaginären Achse n_p, enthält. Die Gewichtungsfaktoren k_x müssen problemspezifisch angepasst werden.

$$E = \frac{1}{F} = k_a \cdot \frac{1}{n_s} \sum_{i=1}^{n_s} (S_i - K_i)^2 + k_n \cdot n_{Gm} + k_p \cdot n_p \tag{5}$$

Der Genetische Algorithmus läuft wie folgt ab: Zunächst wird eine zufällig generierte Startpopulation der Größe p erzeugt. Die Individuen dieser Population entsprechen Filterkonfigurationen des FPAA bzw. Punkte im Lösungsraum. Sodann wird ein iterativer Prozess durchlaufen:

```
while Abbruchkriterium nicht erfüllt do
    Kopiere die n besten Individuen aus der alten in die
    neue Population;
    while neue Population < alte Population do
        Selektiere 2 Individuen aus alter Population;
        if rand()<r_xover then
            Führe Cross-over durch;
        end
        if rand()<r_mut then
            Führe Mutation durch;
        end
        Addiere Individuen zu neuer Population;
    end
    Berechne Fitness aller Individuen;
    Ersetze alte durch neue Population;
end
```

Abbruchkriterium ist in der Regel das Erreichen einer festgelegten Anzahl von Iterationen oder das Erreichen einer bestimmten Fitness. In jeder Iteration wird die Population durch eine neue Population gleicher Größe ersetzt, die zum Großteil durch Selektion und Rekombination der bereits bestehenden gebildet wird. Dabei kommen 3 Mechanismen zum Einsatz: „Elitismus“, d.h. die besten n Individuen werden unverändert übernommen. Dies ist notwendig, damit die aktuell beste Lösung niemals verlorengeht. Die neue Population wird dadurch erzeugt, indem wiederholt jeweils 2 Individuen aus der alten Population Selektiert, und mit einer Wahrscheinlichkeit $r_{x\text{over}}$ rekombiniert werden.

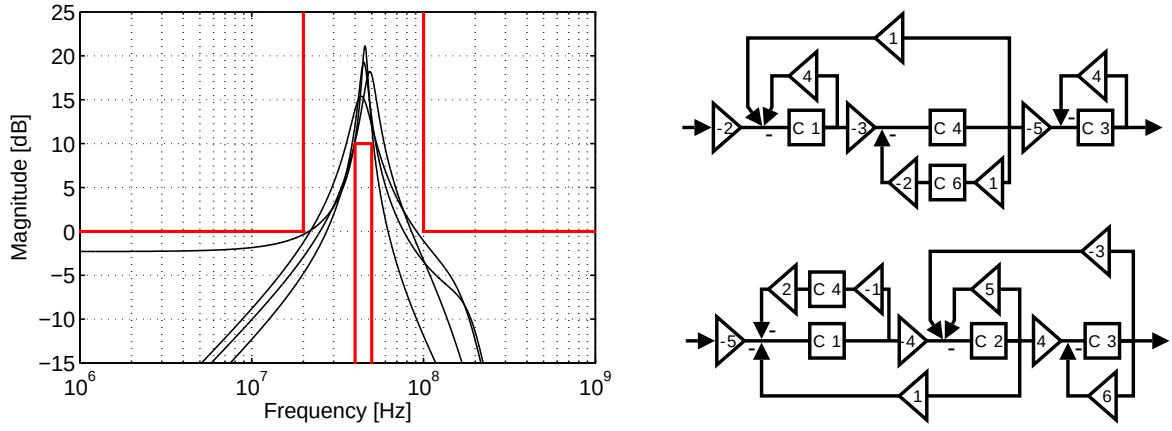

Abb. 3. Synthese von Filtern für eine Bandpass-Spezifikation.

Tabelle 1. GA-Parameter.

Name	Wert
p	100
t	2
r_{mut}	5%
$r_{x\text{over}}$	50%
k_a	100
k_n	0.1
k_p	10^5
Abbruchkriterium	25 Generationen ohne Verbesserung

Die Selektion findet nach einem Mechanismus statt, der Individuen mit höherem Fitnesswert in der Regel bevorzugt, jedoch auch schwächeren Individuen die Chance gibt, selektiert zu werden. Hier hat sich die sogenannte „Tournament-Selection“ als vorteilhaft erwiesen, bei der 2 mal t Individuen zufällig ausgewählt und jedes mal das Beste selektiert wird.

Bei der Rekombination findet ein Austausch von Informationen der Individuen statt, d.h. es werden zusammenhängende Blöcke der FPAA-Konfiguration zwischen den beiden Individuen ausgetauscht. Zusätzlich dazu wird eine Mutation durchgeführt, was einer zufälligen Variation der Parameter jedes der beiden Individuen mit einer Wahrscheinlichkeit r_{mut} entspricht.

Der Algorithmus sucht also den Suchraum parallel an p Punkten ab. Die Lage bzw die Bewegung dieser Punkte folgt dabei nicht der lokalen Steigung, wie bei gradientenbasierten Verfahren, sondern setzt sich aus zwei Anteilen zusammen: Zum einen gehen neue Punkte aus der Rekombination der Parameter der besten vorhandenen Punkte hervor. Zum anderen werden durch die Mutation bestehende Punkte in Ihrer Lage zufällig variiert. Diese Kombination läßt den GA bei geeigneter Wahl seiner Parameter sehr robust gegen lokale Maxima werden. Der Algorithmus konvergiert, da die Selektion Individuen mit höherer Fitness bevorzugt.

5 Ergebnisse

Der im vorherigen Abschnitt beschriebene GA wurde in Matlab implementiert, wobei die Fitnessberechung anhand des Matlab-Modells des FPAA stattfindet. Die Parameter des GA wurden empirisch ermittelt und sind in Tabelle 1 zusammengefasst.

5.1 Filtersynthese

Im folgenden wurde die Eignung des GA untersucht, Filterstrukturen gemäß verschiedener vorgegebener Spezifikationen zu synthetisieren. Abbildung 3 zeigt auf der linken Seite eine Bandpass-Spezifikation (als rote Linien dargestellt) und die Frequenzgänge der Lösungen aus 4 Durchläufen des GA. Auf der rechten Seite ist die Struktur der zwei besten Lösungen dargestellt. Die mit „C n“ markierten Blöcke stellen die CABs dar. Die Transkonduktoren (dreieckige Elemente) sind mit ihrer Einstellung markiert. Zur besseren Übersicht sind die Elemente linear von Eingang zum Ausgang angeordnet, und nicht entsprechend ihrer Lage auf dem FPAA.

Alle gefundenen Lösungen liegen innerhalb der durch die Spezifikation vorgegebenen Grenzen und benötigen exakt 4 CABs auf dem FPAA, es handelt sich also um Filter 4ter Ordnung. Die Anzahl der aktiven Elemente (eingeschaltete G_m-Zellen) variiert jedoch. In beiden Fällen ist die charakteristische Gyrator-Struktur zu erkennen (C 6 bzw. C 4 im Rückkopplungspfad mit Vorzeichenwechsel des Signals), die ein Induktives Element simuliert.

5.2 Ausgleichen von Temperatureinflüssen

Außer zur Filtersynthese kann der GA auch dazu eingesetzt werden, vorhandene Filter zu optimieren, bzw. externe

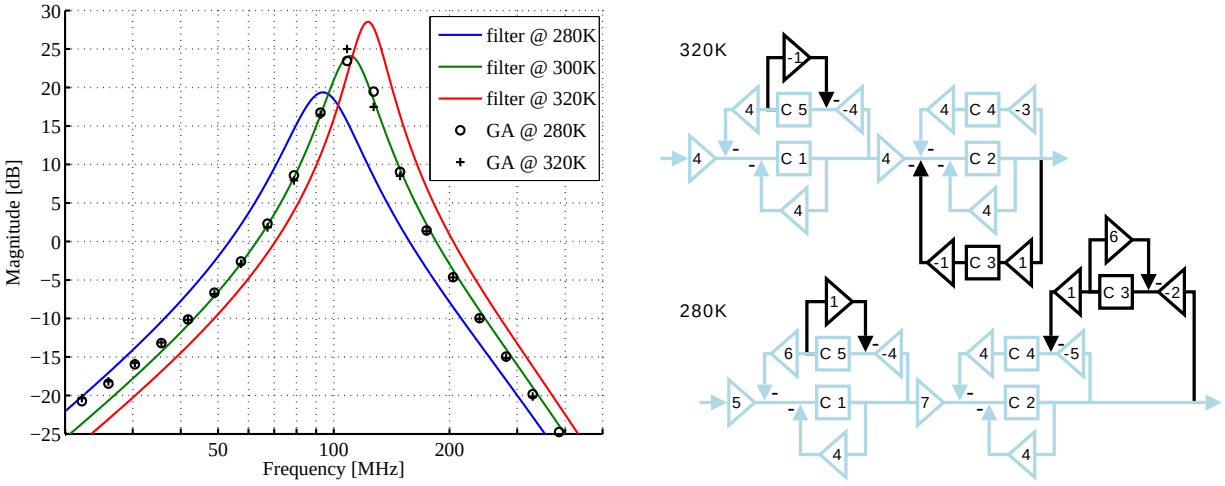

Abb. 4. Ausgleichen eines simulierten Temperaturdrifts.

Einflüsse, zum Beispiel Temperaturschwankungen, auszugleichen. Dafür wird die Spezifikation so angepasst, daß sie genau auf den bereits instantiierten Filter zutrifft. Anstatt hier einen komplett neuen Filter zu synthetisieren, empfiehlt es sich bei der Optimierung von der ursprünglichen Filterstruktur auszugehen. Daher wird die Population zu Beginn des GA nicht zufällig, sondern mit sondern mit Kopien der Referenzstruktur initialisiert (dies Entspricht einer Optimierung mit Wahl der Startposition).

Der gegenüber der Temperatur empfindlichste Parameter ist das G_m der G_m-Zelle, für das gilt:

$$G_m \approx \sqrt{2\,\beta\,I_{SS}} \tag{6}$$

I_{SS} ist ein konstanter Bias-Strom, der wiederum von β_0 und V_T abhängt. Der Einfluß der Temperatur auf diese Parameter wurde durch folgende Näherung modelliert:

$$\mu = \mu_0 \left(\frac{300\text{K}}{T}\right)^{\frac{3}{2}}; \quad V_T = V_{T0} - 1\text{mV}\,(T - 300\text{K}) \tag{7}$$

Die Simulationen wurden mit dem FPAA-Modell durchgeführt, das um die beschriebenen Temperatur-Effekte ergänzt wurde. Als Ausgangspunkt diente ein Bandpass 4ter Ordnung. Abbildung 4 zeigt auf der linken Seite die Übertragungskennlinie des Referenzfilters bei 300 K, und die sich durch eine Änderung der Temperatur um ±20 K ergebende Verschiebung als durchgezogene Linien. Die Optimierung wurde unter den gleichen simulierten Temperaturbedingungen durchgeführt. In beiden Fällen gelang es, mit dem GA den Filter so umzustrukturieren, daß die ursprüngliche Übertragungskennlinie bis auf kleine Abweichungen wiederhergestellt wird (gepunktete Linien). In den gefundenen Lösungen (rechte Seite) ist die Struktur des Ursprungsfilters (blau dargestellt) jeweils gut zu erkennen. Die Anpassung wurde in beiden Fällen durch einer Kombination von Parametervariationen und Hinzufügen einzelner Rückkopplungspfade erreicht.

6 Zusammenfassung

Es wurde eine neuer Ansatz für die Synthese von analogen Filtern auf einer FPAA-Architektur durch einen Genetischen Algorithmus vorgestellt. Der Ansatz unterscheidet sich von der klassischen Filtersynthese dadurch, daß der Raum der möglichen Filter direkt durch ein heuristisches Verfahren nach Lösungen durchsucht wird, die eine gute Übereinstimmung mit der Spezifikation aufweisen. Es wurde gezeigt, daß sich dieses Verfahren sowohl für die Synthese als auch Optimierung von Filtern eignet. Gegenstand der gegenwärtigen Forschung ist die Wiederholung der Experimente mit einem Prototypen der FPAA-Hardware.

Literatur

Becker, J. and Manoli, Y.: A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M-cells, Proc. ISCAS'04, 1, 1092–1095, doi:10.1109/ISCAS.2004.1328389, 2004.

Hall, T., Twigg, C., Gray, J., Hasler, P., and Anderson, D.: Large-scale field-programmable analog arrays for analog signal processing, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 52, 2298–2307, doi:10.1109/TCSI.2005.853401, 2005.

Henrici, F., Becker, J., Buhmann, A., Ortmanns, M., and Manoli, Y.: A Continuous-Time Field Programmable Analog Array Using Parasitic Capacitance Gm-C Filters, Proc. ISCAS'07, pp. 2236–2239, doi:10.1109/ISCAS.2007.378727, 2007.