This is the author's version of the work. The definitive work was published in Proceedings of Zuverlässigkeit und Entwurf (ZuD 2007), Munich, Germany, pp. 141-148, 2007. The work is supported in part by the German Science Foundation (DFG), SFB 694 and SPP 1148.

Synthese zuverlässiger und flexibler Systeme*

Michael Glaß, Martin Lukasiewycz, Thilo Streichert, Christian Haubelt und Jürgen Teich
Universität Erlangen-Nürnberg
{glass, martin.lukasiewycz, streichert, haubelt, teich}@cs.fau.de

Kurzfassung

Die Erhöhung der Zuverlässigkeit ist eines der wichtigsten Entwurfsziele derzeitiger und zukünftiger eingebetteter Systeme. In diesem Beitrag konzentrieren wir uns auf die Entwurfsphase, in der die Zuverlässigkeit nur eine von vielen, häufig konkurrierenden Zielgrößen darstellt. Existierende Ansätze betrachteten die gleichzeitige Optimierung von Zuverlässigkeit gemeinsam mit allen anderen Zielgrößen als zu aufwändig. Aus diesem Grund erfolgt in einem ersten Schritt der Systementwurf, in einem zweiten Schritt eine Analyse des Systems hinsichtlich dessen Zuverlässigkeit und im dritten Schritt eine Optimierung bzw. Einhaltung von Beschränkungen hinsichtlich der Zuverlässigkeit durch Austausch kritischer Ressourcen bzw. Einführung von Redundanz. Dieses Vorgehen führt bei Betrachtung aller Zielgrößen im Allgemeinen zu suboptimalen Lösungen. Wir präsentieren darum a) einen neuen Ansatz, der *Zuverlässigkeit* gleichzeitig mit allen anderen Zielgrößen optimiert und *Funktionsalternativen* als funktionale Redundanzen nutzt, b) eine *Analysetechnik*, die es erlaubt, auch für reale Beispiele einen quantitativen Vergleich der Zuverlässigkeit der gefundenen Lösungen in akzeptabler Zeit durchzuführen, sowie c) experimentelle Ergebnisse, die die Effektivität unseres Ansatzes verdeutlichen.

1 Einleitung

Eingebettete elektrische Systeme befinden sich heute in Produkten aller Art, von Unterhaltungselektronik bis zu Steuerungen in Fahrzeugen oder industriellen Anlagen. Die Systeme selbst sollen dabei verschiedenste und häufig konkurrierende Anforderungen z.B. an Größe, Energieverbrauch oder monetäre Kosten erfüllen, wobei das System unter Einhaltung dieser Anforderungen möglichst ein Optimum bezüglich aller Zielgrößen, ein sogenanntes *Pareto-Optimum*, darstellen sollte. Mit dem Einzug eingebetteter Systeme in sicherheitskritische Bereiche, in denen sie zudem häufig verschiedensten äußeren Bedingungen und Einflüssen ausgesetzt sind, steigen die Anforderungen an deren Zuverlässigkeit.

Während hohe Temperaturschwankungen die Elektronik belasten und den Alterungsprozess vorantreiben, führt eine hohe Strahlung zu sogenannten *Single Event Upsets* (SEUs). Diese SEUs werden durch die Ladung einzelner Transistoren verursacht und führen zu einem Fehler in der Schaltung. Ob dieser Fehler einen permanenten oder temporären Charakter hat, hängt von der Art der Schaltung und der Position des Fehlers innerhalb der Schaltung ab. Man nehme als Beispiel ein SRAM-basiertes FPGA, welches mit einer bestimmten Konfiguration geladen wurde und in dem ein Bit in einem bestimmten Teil gekippt ist. Je nach Position dieses *Bitflips* offenbart sich dieser Fehler als permanentes oder transientes Fehlverhalten. Tritt dieses SEU in einer Konfigurationsspeicherzelle auf, so liegt ein Fehlverhalten bis zur nächsten Rekonfiguration vor. Im anderen Fall, bei dem das SEU nur FlipFlops und Inverter betrifft, liegt ein transientes Fehlverhalten vor. Der Einfluss von SEUs auf FPGAs wurde intensiv in [12] untersucht.

Am Schwerionendetektor in Cern [1] müssen FPGA-basierte ECUs unter dem Einfluss von starker Strahlung arbeiten. Interessanterweise werden diese FPGAs periodisch rekonfiguriert, um einen fehlerfreien Betrieb zu erreichen. Auf der anderen Seite sollen transiente Fehler häufiger in ASICs mit einer besonders hohen Integrationsdichte auftreten [17]. Auf Grund der sinkenden Transistorgrößen können hier die stark reduzierten Kapazitäten der Gatter durch Strahlung geladen werden.

Reliability Engineering ist eine Disziplin, die den kompletten Lebenszyklus eines Systems, von der Entwurfs-, über die Test- und Integrationsphase bis zum Betrieb des Systems abdeckt. In all diesen Phasen des Lebenszyklus wird unter anderem versucht, die *Zuverlässigkeit* $R(t)=P[LD>t]$, d.h. die Wahrscheinlichkeit dafür, dass die Lebensdauer LD eines Systems größer als die Zeit t ist, zu erhöhen. Dies wird im Wesentlichen durch eine entsprechende Auswahl von zuverlässigen Hardwareressourcen sowie Fehlertoleranzmaßnahmen erreicht. Bestehende Ansätze für den Entwurf zuverlässiger Systeme führen nach dem klassischen Systementwurf eine Analysephase durch, in der die Zuverlässigkeit des Entwurfspunktes bestimmt wird. Daran anschließend werden kritische Komponenten ausgetauscht oder Fehlertoleranzmaßnahmen wie redundante Komponenten eingesetzt, um die vorgegebenen Zuverlässigkeitseigenschaften des Systems zu

* mit Unterstützung der Deutschen Forschungsgemeinschaft (DFG), SFB 694, SPP 1148

erreichen. Dieses Vorgehen führt im Allgemeinen jedoch zu suboptimalen Lösungen, da der Schritt der Zuverlässigkeitsoptimierung unabhängig von der vorherigen Entwurfsraumexploration stattfindet. In diesem Beitrag präsentieren wir darum ein neues Verfahren für die Synthese von zuverlässigen und flexiblen integrierten Systemen auf Systemebene. Ausgehend von einer Verhaltensbeschreibung in Form eines *hierarchischen Datenflussgraphen*, welcher *Funktionsalternativen* beinhaltet, sowie einem *Architekturgraphen*, der alle Ressourcenalternativen beinhaltet, sind wir in der Lage, Datenpfade zu synthetisieren, die mehrzieloptimiert bezüglich aller Zielgrößen inklusive der Zuverlässigkeit sind.

Ein veranschaulichendes Beispiel für die Methodik, die in diesem Beitrag vorgestellt wird, findet sich in **Bild 1**. Diese Anwendung liest Videodaten ein, um sie im Schritt *EnCo* zu kodieren. Für die Kodierung stehen drei verschiedene Kodieralgorithmen in Form von Subgraphen zur Verfügung: Das Video Texture Coding (VTC) besteht hierbei aus *5*, MPEG4 aus *32* und H.261 aus *27* Operationen. Nachfolgend werden die kodierten Videodaten durch die Operation *EnCry* verschlüsselt, wofür ebenfalls drei verschiedene Verschlüsselungsalgorithmen in Form von Subgraphen zur Verfügung stehen: Fiestal besteht hierbei aus *9*, Substitution/Permutation aus *16* und Square aus *12* Operationen. Anschließend werden die kodierten und verschlüsselten Daten ausgegeben. Für die Funktion des Kodierens und Verschlüsselns genügt der Anwendung jeweils eine implementierte bzw. korrekt funktionierende Funktionsalternative. Aus Sicht des Anwenders erhöht sich die sogenannte *Flexibilität*, d.h. der Grad der Funktionalität, die das System als Alternative zu seiner Grundfunktionalität anbietet [10]. Aus Sicht der Zuverlässigkeit kann eine *funktionale Redundanz* der Funktionsalternativen angenommen werden. Auf andere Zielfunktionen wie Platz- und Energieverbrauch sowie monetäre Kosten wirkt sich die Implementierung mehrerer Funktionsalternativen hingegen negativ aus. Dieser Trade-Off-Charakter zwischen den verschiedenen Zielgrößen unter Berücksichtigung von Allokation und Bindung von Funktionsalternativen erweitert den Lösungsraum des Problems stark und erfordert eine gleichzeitige Optimierung aller Zielgrößen, insbesondere der Zuverlässigkeit der ausgewählten Implementierung.

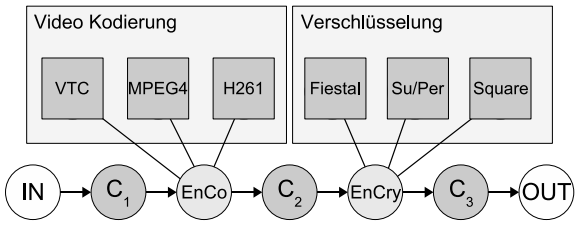


Bild 1 Modell einer Anwendung, die Videodaten einliest, kodiert und nachfolgend verschlüsselt ausgibt.

Bis jetzt lag die Entscheidung der Auswahl einer Implementierung beim Entwickler des Systems. Ohne dem Entwickler diese Entscheidung vollständig abzunehmen, bietet unser vorgestellter Syntheseansatz die Möglichkeit, optimale Lösungen, auch bezüglich der Zuverlässigkeit des Systems, automatisch zur Verfügung zu stellen und so die Möglichkeit zur Entscheidungsfindung qualitativ zu verbessern. Um dies zu erreichen, präsentieren wir in diesem Beitrag eine Methodik mit den folgenden Eigenschaften:

- Auswahl von *funktional äquivalenten Ressourcen*, die sich jedoch in Platzverbrauch, Latenz oder Zuverlässigkeit unterscheiden
- Gleichzeitige Optimierung hinsichtlich *mehrerer Zielgrößen* wie Platzverbrauch, Latenz und Zuverlässigkeit
- *Wiederverwendung von Ressourcen*: Verschiedene Operationen des Datenflussgraphen können auf dieselbe Ressource gebunden werden
- Automatische redundante Auslegung (*Replizierung*) von Operationen
- Nutzung von *Funktionsalternativen* als funktionale Redundanzen

Der Rest dieses Beitrag gliedert sich wie folgt: Abschnitt 2 beschreibt den derzeitigen Stand der Forschung. Abschnitt 3 erläutert unseren neuen Ansatz, Redundanz, Funktionsalternativen und Zuverlässigkeitsanalyse direkt in ein *Mehrzieloptimierungsproblem* einzubetten. Abschnitt 4 präsentiert erste experimentelle Ergebnisse. Dieser Beitrag wird in Abschnitt 5 mit einer Zusammenfassung und einem Ausblick abgeschlossen.

2 Stand der Forschung

Bis heute wurde bereits eine Vielzahl an Ansätzen präsentiert, die sich mit der Analyse von Systemen in Bezug auf ihre Zuverlässigkeit und Fehlertoleranz beschäftigen. Ein Überblick über diese Techniken findet sich in [5]. In den letzten Jahren wurden diese Techniken auch in die Synthese integriert, um zuverlässigere bzw. fehlertolerantere Systeme zu entwerfen.

Ein früher Ansatz im Bereich der Fehlertoleranz wurde von Karri et al. [11] vorgestellt. Hierbei wird der Overhead minimiert, der durch die Einführung von Replikas entsteht. Als Eingangsdatum für ihren Syntheseansatz wird ein Datenflussgraph (DFG), eine Redundanzrate und die Latenz des Systems benötigt. Mittels dieser Information werden verschiedene Transformationen auf dem DFG ausgeführt, welche die Kosten für Redundanz verringern. Die damit erzielten Resultate sind eindrucksvoll, da die Einsparungen bei den Hardwarekosten bei bis zu 35.71% liegen, wobei der Grad der Redundanz in ihren Beispielen von 2 bis 7 reicht.

In einem weiteren Ansatz [16, 17] wird versucht die Zuverlässigkeit des Systems zu maximieren, indem ausgewählte Redundanzen eingefügt werden, um sogenannte *Soft-Errors*, d.h. das Kippen einzelner Bits in Speicherbausteinen oder kombinatorischen Schaltungen, zu erkennen. Dieser Ansatz kann ebenfalls Ressourcen mit gleicher Funktionalität, jedoch verschiedenen Werten für Flächenbedarf, Latenz oder Zuverlässigkeit berücksichtigen. Im ersten Schritt werden die zuverlässigsten Ressourcen ausgewählt. Die Operationen des DFG werden dann auf diese gebunden sowie ein gültiger Ablaufplan erstellt. Wenn dieser Entwurfspunkt die Beschränkungen für den Platzverbrauch oder die Latenz nicht einhält, werden schrittweise Ressourcen durch entsprechend kleinere oder schnellere, jedoch unzuverlässigere Ressourcen ausgetauscht. Mit dieser Austauschstrategie wird versucht, unter gegebenen Beschränkungen für den Platzverbrauch und die Latenz eine größtmögliche Zuverlässigkeit des Systems zu erzielen. Es ist jedoch zu bemerken, dass bei diesem Ansatz nur die Zuverlässigkeit optimiert wird, während die anderen Zielgrößen Platzverbrauch und Latenz als Beschränkungen gegeben sind.

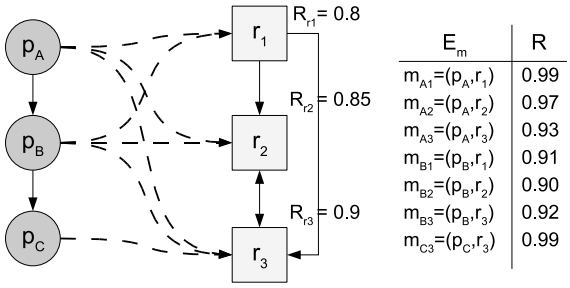
Ein Co-Design-Framework, welches Zuverlässigkeitseigenschaften auf Systemebene berücksichtigen kann, wurde von Bolchini et al. [7] eingeführt, die hierbei ebenfalls einen zweistufigen Ansatz verwenden. Im ersten Schritt führen die Autoren eine Partitionierung durch, die Zielgrößen wie Platz- und Energieverbrauch, monetäre Kosten etc. beachtet. In einem zweiten Schritt wird die Zuverlässigkeit kritischer Bereiche erhöht. Es ist zu bemerken, dass dieser zweite Schritt die globale Pareto-Optimalität der gefundenen Lösung aus dem ersten Schritt im Allgemeinen nicht mehr gewährleisten kann. Weiterhin gehen optimale Lösungen verloren, die sich aus suboptimalen Lösungen aus dem ersten Schritt und den Umformungen aus dem zweiten Schritt ergeben können. Trotzdem ist dies ein interessanter Ansatz, da im zweiten Schritt verschiedenste Techniken zur Zuverlässigkeitssteigerung Anwendung finden können.

Coit und Smith [9] schlagen einen weiteren Ansatz zur Synthese zuverlässiger Systeme mittels Genetischer Algorithmen (GA) vor. Der GA wählt hierbei angemessene Ressourcen aus und legt einen Grad an Redundanz fest, so dass die Kosten für eine bestimmte Zuverlässigkeit minimal sind. Um die Zuverlässigkeit der generierten Lösung, welche eine Zielgröße des GAs ist, zu bestimmen, verwenden die Autoren ein Neuronales Netzwerk.

Ähnlich dem letztgenannten Ansatz wird in unserem Verfahren der Systemsynthese eine Entwurfsraumexploration mittels Evolutionärer Algorithmen durchgeführt. Im Gegensatz zu dem vorher genannten Ansatz erlauben wir jedoch die Optimierung mehrerer Zielgrößen durch moderne *Mehrzieloptimierende Evolutionäre Algorithmen* (MOEAs) [6]. Nach unserem Wissen wurden in allen bisherigen Syntheseansätzen feste Sätze von Verhaltensbeschreibungen unter Berücksichtigung von Zuverlässigkeit verwendet. In dem hier vorgestellten Ansatz eröffnen wir durch *Funktionsalternativen* völlig neue Lösungsmöglichkeiten, da je nach Anzahl und Permutation der implementierten Funktionsalternativen alle anderen Zielgrößen, insbesondere die Zuverlässigkeit, entsprechend beeinflusst werden. Die Berechnung der Zuverlässigkeit erfolgt in unserem Ansatz auf Basis der Analyse komplexer Systemstrukturen mittels *Binärer Entscheidungsdiagramme* (engl. Binary Decision Diagram, BDD), was im folgenden Abschnitt erläutert wird.

3 Zuverlässigkeitsoptimierung in der Systemsynthese

Die gleichzeitige Optimierung von Zuverlässigkeit mit anderen, häufig konkurrierenden Zielgrößen betrachtete man in der Vergangenheit als zu aufwendig [7]. In diesem Abschnitt werden wir eine Methodik vorstellen, die es erlaubt, Zuverlässigkeit zusammen mit allen anderen Zielgrößen zu optimieren und dabei implizit von Techniken zur redundanten Auslegung des Systems Gebrauch zu machen. Nach einer Einführung unseres Systemmodells werden wir die Ressourcenauswahl sowie die *Mehrfachbindung* von Operationen beschreiben. Anschließend wird unsere Technik zur effizienten Analyse der Zuverlässigkeit von Systemen mit Funktionsalternativen unter Verwendung von BDDs erläutert.



E_m	R
$m_{A1}=(p_A,r_1)$	0.99
$m_{A2}=(p_A,r_2)$	0.97
$m_{A3}=(p_A,r_3)$	0.93
$m_{B1}=(p_B,r_1)$	0.91
$m_{B2}=(p_B,r_2)$	0.90
$m_{B3}=(p_B,r_3)$	0.92
$m_{C3}=(p_C,r_3)$	0.99

Bild 2 Spezifikationsgraph $G_{s,}$ bestehend aus Problemgraph G_p, Architekturgraph G_a und Abbildungskanten E_m.

Bild 2 zeigt unser Systemmodell, welches durch den sogenannten *Spezifikationsgraphen* $G_s = (G_p, G_a, E_m)$ [2] dargestellt wird. Das Systemverhalten wird durch den azyklischen, hierarchischen *Problemgraphen* $G_p = (V_p, E_p, \Psi_p, \Gamma_p)$, dargestellt links, beschrieben, der in diesem Falle als ein DFG mit Operationen $p \in V_p$ und Datenabhängigkeiten $e \in E_p$ angesehen werden kann. Ψ_p ist die Menge der sogenannten *Interfaces*, welche hierarchischen Operationen entsprechen. Diese werden jeweils durch *Funktionsalternativen* $\gamma \in \Gamma_p$ verfeinert, welche in diesem Falle wiederum als *Subproblemgraphen* modelliert

werden. Der *Architekturgraph* $G_a = (V_a, E_a)$, in Bild 2 rechts, modelliert Ressourcen $r \in V_a$ und deren Kommunikationsverbindungen $e \in E_a$. Gestrichelt dargestellt sind die *Abbildungskanten* $m=(p, r) \in E_m$, welche der Möglichkeit entsprechen, eine Operation p auf einer Ressource r zu implementieren.
Um die Zuverlässigkeit der Ressourcen zu modellieren, werden die Knoten im Architekturgraphen mit Zuverlässigkeitsattributen versehen. Hierbei können zeitunabhängige Werte für die Zuverlässigkeit R in Form von Wahrscheinlichkeiten verwendet werden. Ebenso ist es gestattet Ausfallraten λ anzutragen, die mittels verschiedener Verteilungen, wie zum Beispiel der Weibull-Verteilung, interpretiert werden können und damit eine zeitabhängige Zuverlässigkeit $R(t)$ modellieren. Die Abbildungskanten können ebenfalls mit Zuverlässigkeitsattributen versehen werden, um damit die Zuverlässigkeit der Ausführung einer Operation auf einer Ressource zu modellieren.
Die simultane Optimierung der Zuverlässigkeit mit anderen Zielgrößen erfolgt mittels einer Entwurfsraumexploration, welche ebenfalls die Systemsynthese durchführt. Die drei Aufgaben der Systemsynthese bestehen hierbei aus 1) der Auswahl (*Allokation*) einer Menge von Ressourcen, 2) dem gültigen *Binden* von jeder Operation auf eine allokierte Ressource und 3) dem Bestimmen eines gültigen *Ablaufplans*. Da existierende Entwurfsraumexplorationsverfahren [3, 13, 15] die Bindung einer Operation auf maximal eine Ressource beschränken, wird eine Steigerung der Zuverlässigkeit der gefundenen Lösungen durch redundante Auslegung einer Operation verhindert. Aus diesem Grund wurde unsere Entwurfsraumexploration [3] wie folgt erweitert:

- Generierung von *mehrfachen Instanzen* einer Operation durch Mehrfachbindung,
- Nutzung von *Funktionsalternativen* als funktionale Redundanzen,
- *Analyse der Zuverlässigkeit* einer gefundenen Lösung anhand unseres Systemmodells.

Unsere Entwurfsraumexploration benutzt Mehrzieloptimierende Evolutionäre Algorithmen, siehe [6]. Die Allokation $\alpha \subseteq V_a \cup \Psi_p$ der Ressourcen und Funktionsalternativen wird in einem Bitvektor kodiert, in dem jedes Bit angibt, ob die entsprechende Ressource bzw. Funktionsalternative allokiert ist oder nicht. Bei der Allokation von Funktionsalternativen ist zu beachten, dass für jedes Interface mindestens eine Verfeinerung allokiert sein muss. Die verschiedenen möglichen Permutationen von allokierten Verfeinerungen haben dabei großen Einfluss auf die späteren Zielgrößen des Systems. Die Bindung $\beta \subseteq E_m$ wird in Prioritätslisten kodiert. Diese Prioritätslisten geben an, in welcher Reihenfolge die Operationen auf die Ressourcen gebunden werden. Um nun eine Operation auf mehrere Ressourcen binden zu können, wird ein weiterer Parameter in das Chromosom kodiert, der die maximale Anzahl an auswählbaren Ressourcen pro Operation angibt. All diese Parameter werden mittels genetischer Operatoren, z.B. Mutation und Kreuzung, variiert, um die Suche in Richtung optimaler Entwurfspunkte zu leiten.

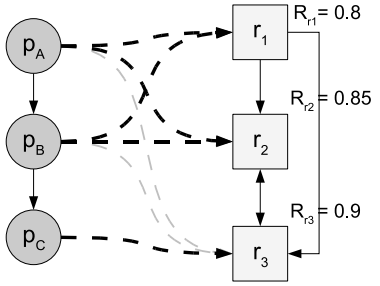
3.1 Redundanz durch Mehrfachbindung

Das Konzept der *Redundanz*, was ein Vorhandensein von mehr Mitteln als zur Erfüllung einer Aufgabe notwendig sind beschreibt, ist dafür bekannt die Zuverlässigkeit eines Systems steigern zu können. In herkömmlichen Systemsynthesemodellen wird eine Operation auf genau eine Ressource gebunden, was eine Redundanz auf Operationsniveau ausschließt und die Analyse der Zuverlässigkeit des Gesamtsystems auf eine Serienstruktur aller Systemkomponenten vereinfacht. Betrachtet man das Beispiel aus Bild 2 und bindet p_A auf r_1, p_B auf r_2 und p_C auf r_3, so berechnet sich die Zuverlässigkeit dieser Lösung aus $R_{r1} \cdot R_{r2} \cdot R_{r3} \cdot R_{mA1} \cdot R_{mB2} \cdot R_{mC3} = 0.564$.
Da unsere Analysetechnik effizient genug ist, komplexe Systemstrukturen zu bewältigen, führen wir die Möglichkeit ein, eine Operation auf mehrere Ressourcen zu binden, um so verschiedene Instanzen einer Operation im Gesamtsystem verfügbar zu machen.
Unsere neue Definition einer gültigen Bindung $\beta = \bigcup_{p \in V_p} I_p$ lautet darum wie folgt:

1. Für jede Operation $p \in V_p$ ist eine Menge $I_i \subseteq E_m$ ausgehender Abbildungskanten aktiviert. Jede Abbildungskante $m \in I_i$ repräsentiert hierbei eine Instanz der Operation p.
2. Jede Operationsinstanz $m = (v_p, v_a) \in \beta$ endet auf einer allokierten Ressource $r \in \alpha$.
3. Für mindestens eine Menge $L \subseteq \beta$, welche genau eine Operationsinstanz einer jeden Operation enthält, gilt, dass die Instanzen $m = (p, r)$ und $m' = (p', r')$ einer jeden Datenabhängigkeit $(p, p') \in E_p$ auf identische $r = r'$ oder adjazente Ressourcen $(r, r') \in E_a$ gebunden sind.

Die erste Bedingung ermöglicht es eine Operation auf mehrere Ressourcen zu binden und erstellt damit Operationsinstanzen, die Redundanz gewährleisten. Es ist zu bemerken, dass die dritte Bedingung nur mindestens eine Permutation von Instanzen sichert, die einem funktionierenden Gesamtsystem entsprechen. Diese lässt sich jedoch für striktere Anforderungen erweitern.
Eine gültige Bindung ist in **Bild 3** annotiert. Die erste Bedingung für eine gültige Bindung erlaubt es, die Operationen p_A und p_B auf die Ressourcen r_1 und r_2 sowie Operation p_C auf r_3 zu binden. Unter der Annahme, dass alle Ressourcen allokiert wurden, ist auch Bedingung 2 erfüllt. Bedingung 3 kann z.B. durch die

Menge ($m_{A1} = (p_A, r_1)$, $m_{B2} = (p_B, r_2)$, $m_{C3} = (p_C, r_3)$) erfüllt werden. Durch das Einbringen von mehreren Operationsinstanzen für p_A und p_B wird Redundanz erzeugt, womit die Zuverlässigkeit des Systems erhöht werden kann.



E_m	R
$m_{A1}=(p_A,r_1)$	0.99
$m_{A2}=(p_A,r_2)$	0.97
$m_{A3}=(p_A,r_3)$	0.93
$m_{B1}=(p_B,r_1)$	0.91
$m_{B2}=(p_B,r_2)$	0.90
$m_{B3}=(p_B,r_3)$	0.92
$m_{C3}=(p_C,r_3)$	0.99

Bild 3 Die schwarzen Abbildungskanten zeigen eine gültige Mehrfachbindung des Problemgraphen auf die gegebene Architektur.

Für eine anschauliche Darstellung der Operationsinstanzen und ihrer Kommunikation sowie für spätere Definitionen in Abschnitt 3.2 sei an dieser Stelle ein neues Modell, der sogenannte *Instanzengraph* $G_i = (G_i, E_i)$, eingeführt. Die Menge der Knoten V_i ist definiert als $V_i = \beta$. Jede Kante $e = (m, m') \in E_i$ entspricht einem Paar von Operationsinstanzen $m = (p, r)$ und $m' = (p', r')$, zwischen denen eine Datenabhängigkeit $(p, p') \in E_p$ im Problemgraph besteht und für die $r = r'$ oder $(r, r') \in E_a$ gilt. Der Graph bietet zudem die interessante Information, ob eine Permutation, also eine Operationsinstanz jeder Operation, ein Funktionieren des Gesamtsystems gewährleisten würde. Besteht zwischen all den gewählten Operationsinstanzen jeweils eine gerichtete Verbindung, so entspricht dies einem funktionierenden System. **Bild 4** zeigt den Instanzengraphen für die Beispielbindung aus Bild 3.

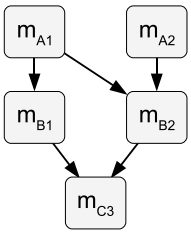


Bild 4 Instanzengraph der Bindung aus Bild 3.

3.2 Zuverlässigkeitsanalyse von Funktionsalternativen

Um die Zuverlässigkeit eines Systems quantitativ bestimmen zu können, muss die verwendete Analysetechnik in der Lage sein, komplexe Systemstrukturen auswerten zu können, welche sich in unserem Modell aus der Wiederverwendung von Ressourcen ergeben. Weiterhin wollen wir sowohl Ausfälle von Komponenten und Versagen von Operationsausführungen auf einer Komponente modellieren als auch Funktionsalternativen als funktionale Redundanzen nutzen. Wir stellen im Folgenden eine Technik vor, die direkt anhand des Spezifikationsgraphen, der Allokation α und der Bindung β die sogenannte *Strukturfunktion* φ des Gesamtsystems in Form eines Binären Entscheidungsdiagramms (BDD) kodiert und diese anschließend zur Bestimmung der Zuverlässigkeit analysiert.

Bei der Strukturfunktion $\varphi(x)$ mit $x = (a(r_1), \ldots, a(r_{|\alpha|}), a(m_1), \ldots, a(m_{|\beta|}))$ handelt es sich um eine Boolesche Funktion die angibt, ob das System unter einem Eingabevektor x funktionstüchtig ist. Die Funktion $a: V_a \cup E_m \rightarrow \{0, 1\}$ übersetzt hierbei jede aktivierte Ressource und jede Abbildungskante in eine binäre Variable, wobei der Wert *1* ein korrektes Funktionieren, *0* hingegen einen Ausfall der Komponente bzw. ein Versagen der Operation angibt.

$$\varphi(x) = \begin{cases} 1, & System\ funktioniert \\ 0, & System\ versagt \end{cases} \tag{1}$$

Die Strukturfunktion $\varphi(x)$ definiert sich über die folgenden Gleichungen (2), (3) und (4) sowie unter Verwendung der Funktion $d: E_m \rightarrow \{0, 1\}$. d übersetzt hierbei Operationsinstanzen in binäre Variablen die angeben, ob eine Instanz verwendet wird um die Funktion des Systems sicherzustellen.

$$\varphi_{G_s,\alpha,\beta}(x) = \exists_{m \in \beta}\, d(m): \tag{2}$$

$$\bigwedge_{p \in V_p} \left[\bigvee_{m=(p,r) \in \beta} d(m) \right] \wedge \bigwedge_{\hat{p} \in \Psi_p} \left[\bigvee_{\gamma_{\hat{p}} \in \Gamma_p} T_{\gamma_{\hat{p}}}(x) \right] \wedge \tag{2.1}$$

$$\bigwedge_{m \in \beta} d(m) \rightarrow a(m) \wedge \tag{2.2}$$

$$\bigwedge_{(p,p') \in E_p} \left[\bigwedge_{\substack{m=(p,r),\\ m'=(p',r') \in \beta}} d(m) d(m') \rightarrow C_{m,m'}(x) \right] \tag{2.3}$$

Term (2.1) stellt sicher, dass wenigstens eine[1] Operationsinstanz $m = (p, r)$ einer jeden Operation $p \in V_p$ und wenigstens eine[1] hierarchische Verfeinerung eines jeden Interfaces aktiv und damit das Gesamtsystem funktionstüchtig ist. Term (2.2) sichert die korrekte Ausführung einer verwendeten Operationsinstanz $m = (p, r)$ auf der Ressource r. Term (2.3) sichert eine korrekte Kommunikation zwischen den Instanzen mittels der Funktion $C_{m,m'}(x)$:

$$C_{m,m'}(x) = \begin{cases} a(r)a(r'), & wenn\ (m = (p,r), m' = (p',r')) \in E_i \\ 0, & sonst \end{cases} \tag{3}$$

Um einen rekursiven Abstieg in die Graphenhierarchie zu gewährleisten, verwendet Term (2.1) die Funktion $T_\gamma(x)$, die wie folgt definiert ist:

$$T_\gamma(x) = \bigwedge_{p \in V_{p,\gamma}} \left[\bigvee_{m=(p,r) \in \beta} d(m) \right] \wedge \bigwedge_{\hat{p} \in \Psi_{p,\gamma}} \left[\bigvee_{\tilde{\gamma}_{\hat{p}} \in \Gamma_{p,\gamma}} T_{\tilde{\gamma}_{\hat{p}}}(x) \right] \tag{4}$$

[1] Diese Bedingung entspricht einer *1*-aus-*n* Redundanz und kann durch entsprechendes Umformen der Bedingung auf eine *k*-aus-*n* Redundanz erweitert werden.

Anhand der Terme (2.1), (2.2) und (2.3) sowie unter Verwendung der Gleichungen (3) und (4) wird nun ein BDD generiert. Dieses enthält die Information, unter welchen Mengen $\alpha' \subseteq \alpha$, $\beta' \subseteq \beta$ sowie benutzten Operationsinstanzen das System funktionstüchtig ist. Da für die Berechnung der Zuverlässigkeit jedoch unerheblich ist, welche Operationsinstanzen gerade Verwendung finden, werden diese mittels des Existenz-Quantifikators[2] eliminiert. Das resultierende BDD liefert dann und nur dann *1* zurück, wenn es unter der in *x* kodierten Mengen $\alpha' \subseteq \alpha$ und $\beta' \subseteq \beta$ mindestens eine Menge von Operationsinstanzen gibt, die eine Funktionstüchtigkeit des Systems gewährleistet.
Nachdem die Strukturfunktion des Gesamtsystems generiert wurde, kann die Zuverlässigkeit durch Berechnung der Wahrscheinlichkeit des Wurzelknotens des BDDs berechnet werden. Hierzu findet die in [14] präsentierte Form der Shannon-Zerlegung Anwendung. Werden an die Operationsinstanzen und Ressourcen zeitunabhängige Zuverlässigkeitsattribute in Form von Wahrscheinlichkeiten annotiert, entspricht die Wahrscheinlichkeit des Wurzelknotens der Gesamtzuverlässigkeit des Systems und kann direkt als Wert für die Zielgröße bei der Optimierung genutzt werden. Da bei der Beispielbindung in Bild 3 nur Wahrscheinlichkeiten für die Modellierung verwendet wurden, entspricht somit die Wahrscheinlichkeit des Wurzelknotens von *0.829* der Zuverlässigkeit des Gesamtsystems. Werden Ausfallraten als Zuverlässigkeitsattribute verwendet und durch eine vorgegebene Verteilung interpretiert, greifen wir auf die Verwendung der *Mean Time To Failure*, $MTTF = \int_0^\infty R(t)dt$, des Gesamtsystems als Zielgröße zurück. Hierfür wird der Wert *R(t)* des Wurzelknotens für jeden benötigten Zeitpunkt *t* bestimmt und einem Verfahren zur numerischen Integration zugeführt.

4 Experimentelle Ergebnisse

Im Folgenden werden wir die experimentellen Ergebnisse unserer neuen Entwurfsmethodik für drei Beispiele, einen FIR-Filter vierter Ordnung und eine komplexe Steuerung aus dem Automobilbau ohne Funktionsalternativen sowie dem Videokodierer aus Bild 1 mit Funktionsalternativen vorstellen. Alle Experimente wurden mit dem *SystemCoDesigner*-Framework [18, 19] berechnet.

4.1 FIR-Filter vierter Ordnung

Bild 5 zeigt einen FIR-Filter vierter Ordnung. Für die Synthese dieses Filters stehen drei verschiedene Arten von Addierern und zwei Arten von Multiplizierern zur Verfügung, die alle verschiedene Werte für die Zielgrößen Fläche, Latenz und Zuverlässigkeit aufweisen. Mittels dieser Addierer und Multiplizierer ergibt sich eine Vielzahl von verschiedenen Implementierungen für den Filter, wobei die Lösungen gesucht sind, die Pareto-optimal bezüglich aller Zielgrößen Fläche, Latenz und Zuverlässigkeit sind.

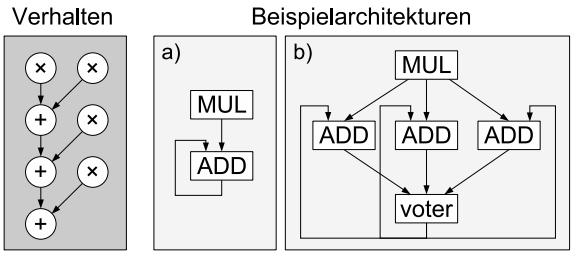


Bild 5 Verhaltensmodell eines FIR-Filters vierter Ordnung in Form eines DFG und zwei Beispielarchitekturen a) ohne und b) mit redundanter Auslegung der Ressourcen.

Bild 5 a) und b) zeigt zwei solche Beispielarchitekturen. Die verwendeten Werte für die Zielgrößen der Ressourcen finden sich in der folgenden Tabelle [16]:

Ressource	Fläche	Latenz	Zuverlässigkeit *R*
ADD1	1	2	0.999
ADD2	2	1	0.969
ADD3	4	1	0.987
MUL1	2	2	0.999
MUL2	4	1	0.969

Nachdem der FIR-Filter vierter Ordnung entsprechend unseres Modells in einen Spezifikationsgraphen umgewandelt wurde, erfolgte der besseren Visualisierung wegen noch eine Beschränkung der Architektur der Implementierung auf maximal drei Addierer und einen Multiplizierer. Weiterhin wurde für das Voting, d.h. die Mehrheitsfindung bei Vorhandensein mehrerer Operationsinstanzen, in diesem Beispiel eine $\left\lfloor \frac{n}{2}+1 \right\rfloor$-aus-*n* Mehrheit gefordert und durch einen Voter, Bild 5 b), realisiert. Die Werte für die Zielgrößen der gefundenen Lösungen finden sich in **Bild 6**.

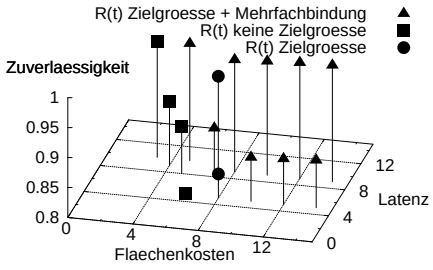


Bild 6 Pareto-Optimale Lösungen des FIR-Filters vierter Ordnung im Zielgrößenraum

[2] $\exists i : f = f_{i=0} \vee f_{i=1}$

Die durch die Vierecke dargestellten Lösungen sind die Pareto-optimalen Lösungen, die gefunden werden, falls Zuverlässigkeit nicht als Zielgröße betrachtet wird. Die durch Kreise repräsentierten Lösungen sind die, die unsere Methodik zusätzlich zu den durch Vierecke dargestellten Lösungen finden konnte, wenn Zuverlässigkeit als Zielgröße beachtet, jedoch keine Mehrfachbindung genutzt wird. Unser Verfahren, welches Zuverlässigkeit als Zielgröße und Redundanz durch Mehrfachbindung berücksichtigt, findet alle dargestellten Punkte, wobei die mittels Dreiecken dargestellten Punkte exklusiv von dieser Methodik gefunden werden.

4.2 Adaptive Lichtsteuerung

Im Folgenden präsentieren wir die Ergebnisse unserer Methodik, angewendet auf ein Beispiel aus der Automobilindustrie, die sogenannte Adaptive Lichtsteuerung (ALC). Mit *234* Operationen, *1103* Ressourcen und *1851* Abbildungskanten erlaubt diese Spezifikation ca. 2^{375} mögliche Bindungen, was einer Komplexität entspricht, die derzeitige Mehrzieloptimierungsverfahren herausfordern.

Bild 7 zeigt die durchschnittliche minimale und maximale MTTF über *10* Explorationen für die ALC. Es ist dabei besonders anzumerken, dass wir bis zu ca. *20%* zuverlässigere Lösungen bestimmen können als die Lösungen, die mit herkömmlichen Mehrzieloptimierungsmethodiken gefunden werden, welche Zuverlässigkeit nicht als Zielgröße betrachten können. Weiterhin sind all diese gefundenen Lösungen bezüglich aller betrachteten Zielgrößen gleichzeitig optimiert. Dieses große Beispiel zeigt die Fähigkeit unseres Ansatzes, auch komplexeste Beispiele aus der Realität zu optimieren und fügt sich damit vorbehaltlos in unser Werkzeug zur Systemsynthese ein.

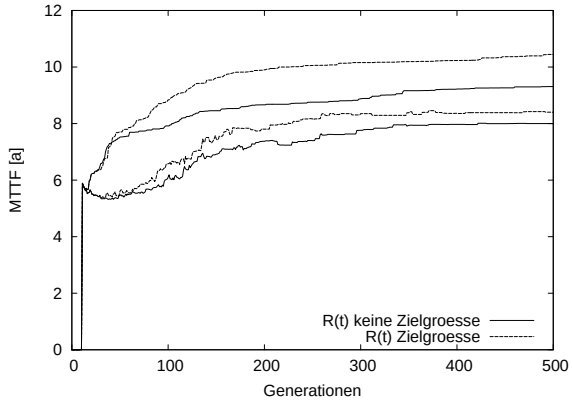


Bild 7 Adaptive Lichtsteuerung: Maximale und Minimale MTTF während des Explorationsprozesses

4.3 Videokodierer

Als drittes Experiment erfolgte eine Optimierung des Videokodierers, welcher bereits im ersten Abschnitt kurz vorgestellt wurde und über Funktionsalternativen für die Videokodierungs- und Verschlüsselungsoperation verfügt. Das Beispiel besteht aus *106* Operationen, die auf *17* Ressourcen gebunden werden können.

Bild 8 zeigt eine über *10* Testläufe gemittelte minimale und maximale Zuverlässigkeit der gefundenen Lösungen für den Videokodierer. Interessanterweise ist unter Berücksichtigung der Zuverlässigkeit als Zielgröße nur eine Steigerung der Zuverlässigkeit um ca. *5%* erreicht worden. Dies hängt vor allem damit zusammen, dass sich *106* Operationen nur *17* Ressourcen teilen und sich damit relativ wenig Redundanz in das System einbringen lässt. Zu bemerken ist jedoch, dass die unzuverlässigsten Lösungen mit Zuverlässigkeit als Zielgröße um ca. *28%* zuverlässiger sind als ohne. Auch hier gilt wieder, dass die betrachteten Lösungen bezüglicher aller Zielgrößen mehrzieloptimiert sind, welche natürlich auch Mehrkosten für redundante Auslegung wie Flächen- und Energieverbrauch etc. beinhalten.

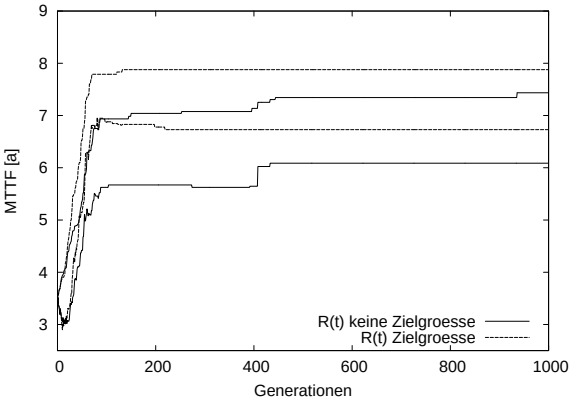


Bild 8 Videokodierer: Maximale und Minimale MTTF während des Explorationsprozesses

Die für alle Experimente beispielhaft verwendeten Zuverlässigkeitswerte finden sich z.B. in [4]. Alle Experimente wurden auf einem Intel Pentium 4, *3.20*GHz PC mit *1*GB RAM durchgeführt. Beispielhaft sei hier die Explorationszeit für die ALC für *500* Generationen und eine Populationsgröße von *100* mit *3*h und *37*min angegeben. In unseren Experimenten mit der ALC, welche die Komplexesten waren, verlängerte unsere Zuverlässigkeitsanalysetechnik die Explorationszeit um einen Faktor von ca. *10* gegenüber Analyseverfahren die z.B. auf schlichter Multiplikation oder Addition beruhen. Die Komplexität der Analyse liegt in der Konstruktion des BDDs, welche im Worst-Case $O(2^{|\alpha|+2\cdot|\beta|})$ beträgt, wobei $|\alpha|+2\cdot|\beta|$ die maximale Anzahl der Variablen während der Konstruktion des BDDs ist. Entscheidend ist jedoch die Anzahl der Knoten im BDD ist jedoch stark von der

Systemstruktur abhängig. Eine hohe Anzahl an Knoten wird insbesondere durch einen hohen Grad an Redundanz erzeugt und erhöht folglich die Komplexität beim Aufbau des BDDs.

5 Zusammenfassung und Ausblick

In diesem Beitrag haben wir zum ersten Mal eine Technik präsentiert, welche automatisch die Zuverlässigkeit eines Systems steigern kann, indem sie die Zuverlässigkeit als Zielgröße in einen Mehrzieloptimierenden Evolutionären Algorithmus zur Entwurfsraumexploration auf Systemebene einbringt. Neben der optimierten Allokation und der impliziten Nutzung von Redundanz mittels Mehrfachbindung von Prozessen und Ressourcenwiederverwendung benutzen wir eine effiziente Analysetechnik, die es uns erlaubt, komplexe Systemstrukturen mit Funktionsalternativen zu analysieren. Unsere experimentellen Ergebnisse zeigen, dass unsere Technik effizient genug ist, um sogar komplexeste Beispiele aus der Praxis zu analysieren und damit dem Entwickler eine größere Menge an, auch hinsichtlich Zuverlässigkeit, mehrzieloptimierten Lösungen zur Verfügung zu stellen.
In zukünftigen Arbeiten werden wir unsere Methodik für den Entwurf von zuverlässigen Netzwerken erweitern. Sogenannte *Network on Chips* (NoCs) bieten Routingfunktionen und komplexere Kommunikationsmöglichkeiten zwischen den Ressourcen, welche dann ebenfalls hinsichtlich mehrerer Zielgrößen und insbesondere der Zuverlässigkeit optimiert werden sollen.

6 Literatur

[1] Cern – Heavy Ion Collider. http://aliceinfo.cern.ch/static/Documents/LHHC/cr5/cr5tpcfee.ppt.

[2] T. Blickle, J. Teich, and L. Thiele. System-Level Synthesis Using Evolutionary Algorithms. J. Design Automation for Embedded Systems, Vol. 3, No. 1, pages 23-58, Kluwer Academic Publishers, Jan. 1998.

[3] T. Schlichter, M. Lukasiewycz, C. Haubelt, and J. Teich. Improving System Level Design Space Exploration by Incorporating SAT-Solvers into Multi-Objective Evolutionary Algorithms. In Proceedings of Annual Symposium on VLSI, pages 309-314, Karlsruhe, Germany, Mar. 2006. IEEE.

[4] Military Handbook 217FN2, Reliability Prediction of Electronic Equipment. Department of Defense, 1995.

[5] A. Birolini. Reliability Engineering – Theory and Practice. Springer, 4th Edition, Berlin, Heidelberg, New York, 2004.

[6] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA – A Platform and Programming Language Independent Interface for Search Algorithms. In Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science (LNCS), volume 2632, pages 494-508, Faro, Portugal, Aug. 2003.

[7] C. Bolchini, L. Pomante, F. Salice, and D. Sciuto. Reliability Properties Assessment at System Level: A Co-Design Framework. Journal of Electronic Testing, 18(3):351-356, 2002.

[8] R. E. Bryant. Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers, 35(8):677-691, 1986.

[9] D. W. Coit and A. E. Smith. Reliability Optimization of Series-Parallel Systems using a Genetic Algorithm. IEEE Transactions on Reliability, 45(1):254-260, 1996.

[10] C. Haubelt, J. Teich, K. Richter, and R. Ernst. System Design for Flexibility. In Proceedings of Design, Automation and Test in Europe (DATE'02), pages 854-861, Paris, France, 2002. IEEE.

[11] R. Karri and A. Orailoğu. Transformation-Based High-Level Synthesis of Fault-Tolerant ASICs. In Proceedings of 29th Design Automation Conference (DAC'92), pages 662-665, USA, June 1992. ACM, IEEE.

[12] F. L. Kastensmidt, G. Neuberger, L. Carro, and R. Reis. Designing and Testing Fault-Tolerant Techniques for SRAM-based FPGAs. In Proceedings of Computing Frontiers 2004, Ischia, Italy, Apr. 2004. ACM.

[13] V. Kianzad and S. S. Bhattacharyya. CHARMED: A Multi-Objective Co-Synthesis Framework for Multi-Mode Embedded Systems. In Proceedings of the 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04), pages 28-40, Galveston, USA, Sept. 2004.

[14] A. Rauzy New Algorithms for Fault Tree Analysis. Reliability Eng. And System Safety, 40:202-211, 1993.

[15] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design Space Exploration of Network Processor Architectures. Network Processor Design: Issues and Practices, 1:55-89, Oct. 2002.

[16] S. Tuson, N. Mansouri, E. Arvas, M.Kandemir, and Y. Xie. Reliability-Centric High-Level Synthesis. In Proceedings of Design Automation and Test in Europe (DATE'05), pages 1258-1263, Munich, Germany, Mar. 2005. IEEE.

[17] Y. Xie, L- Li, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Reliability-Aware Cosynthesis for Embedded Systems. In Proceedings of the 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04), pages 41-50, Galveston, USA, Sept. 2004. ACM.

[18] SystemCoDesigner. http://www12.cs.fau.de/research/scd/

[19] Ch. Haubelt. Automatic Model-Based Design Space Exploration for Embedded Systems - A System Level Approach. Dissertation, University of Erlangen-Nuremberg, Verlag Dr. Köster, Berlin, 2005.