

Ressortforschungsberichte zur Sicherheit der nuklearen Entsorgung

Komplexität und Fehlerpotential bei softwarebasierter digitaler Sicherheitsleittechnik – Vorhaben 4714R01310

Auftragnehmer:
TÜV Rheinland ISTec GmbH, Garching

R. Heigl
F. Hellie
A. Lindner
A. Mölleken



Bundesamt
für die Sicherheit
der nuklearen Entsorgung

Dieser Band enthält einen Ergebnisbericht eines vom Bundesamt für die Sicherheit der nuklearen Entsorgung im Rahmen der Ressortforschung des BMU (ReFoPlan) in Auftrag gegebenen Untersuchungsvorhabens. Verantwortlich für den Inhalt sind allein die Autoren. Das BASE übernimmt keine Gewähr für die Richtigkeit, die Genauigkeit und Vollständigkeit der Angaben sowie die Beachtung privater Rechte Dritter. Der Auftraggeber behält sich alle Rechte vor. Insbesondere darf dieser Bericht nur mit seiner Zustimmung ganz oder teilweise vervielfältigt werden.

Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der des BASE übereinstimmen.

BfE-RESFOR-007/21

Bitte beziehen Sie sich beim Zitieren dieses Dokumentes immer auf folgende URN:
urn:nbn:de:0221-2021012825222

Berlin, Januar 2021



TÜVRheinland®

ISTec

Genau. Richtig.

KOMPLEXITÄT UND FEHLERPOTENTIAL BEI SOFTWAREBASIERTER DIGITALER SICHERHEITS- LEITTECHNIK

Abschlussbericht

R. Heigl
F. Hellie
A. Lindner
A. Mölleken

ISTec - A - 2915
Rev. 1
September 2017

Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der Meinung der Auftraggeberin übereinstimmen

INHALTSVERZEICHNIS

1	EINLEITUNG	11
1.1	Beauftragung	11
1.2	Zielsetzung	11
1.3	Technologien	12
2	BEGRIFFE UND ABKÜRZUNGEN	15
2.1	Begriffe	15
2.2	Abkürzungen	19
3	STAND VON WISSENSCHAFT UND TECHNIK ZUR KOMPLEXITÄTSMESSUNG SOFTWAREBASIERTER LEITTECHNIK	21
3.1	Komplexitätsmetriken	21
3.1.1	In der Literatur dokumentierter Stand für Computerprogramme	21
3.1.2	In der Literatur dokumentierter Stand für programmierbare Logik (FPGAs)	21
3.1.3	Vorangegangene eigenen Projekte	22
3.2	Zuverlässigkeitsbewertung anhand von Komplexitätsmetriken	26
3.2.1	Ansätze aus der Literatur	26
3.2.2	Eigene Arbeiten	26
4	ERWEITERUNGEN DER METHODIK	28
4.1	Erweiterung des Komplexitätsmaßes	28
4.2	Modifikation des Komplexitätsvektors	30
4.2.1	Komplexitätsvektor für CPU-basierte Steuerungen	30
4.2.2	Komplexitätsvektor für programmierbare Logik	31
5	WERKZEUGE ZUR KOMPLEXITÄTSMESSUNG	32
5.1	Analyse des Werkzeugprototypen für TXS	32
5.2	Anpassung der Werkzeuge zur Komplexitätsmessung	32
5.2.1	Schnittstelle von Funktionsplan zum Berechnungswerkzeug	32
5.2.2	Skriptprogramm zur Auswertung von EDIF-Netzlisten (analyzer)	33
5.2.3	Skriptprogramm zur Berechnung der Verflechtungen (vfb)	36
5.2.4	Skriptprogramm zur Auflösung von internen Verbindungen in Funktionsplänen (connector)	39
5.2.5	Skriptprogramm zur Anonymisierung von Funktionsplänen (anonymizer)	39

5.3	Validierung der Werkzeuge zur Komplexitätsmessung	40
5.3.1	Skriptprogramm zur Auswertung von EDIF-Netzlisten (analyzer)	40
5.3.2	Skriptprogramm zur Berechnung der Verflechtungen (vfb)	40
5.3.3	Skriptprogramm zur Auflösung von internen Verbindungen in Funktionsplänen (connector)	41
5.3.4	Skriptprogramm zur Anonymisierung von Funktionsplänen (anonymizer)	41
6	ERMITTLUNG DER KOMPLEXITÄTSCHARAKTERISTIK DIGITALER LEITTECHNIKSYSTEME DER KERNTECHNIK	42
6.1	CPU-basierte Steuerungen	42
6.1.1	Betrachtungsebenen	42
6.1.1.1	Funktionsbausteine	43
6.1.1.2	Funktionspläne	45
6.1.2	Beispiele	46
6.1.2.1	Hersteller 1	47
6.1.2.2	Hersteller 2	50
6.1.2.3	Hersteller 3	52
6.1.3	Schlussfolgerung für Modifikation der Komplexitätscharakteristika	55
6.2	FPGA-basierte Steuerungen	57
6.2.1	Betrachtungsebenen	57
6.2.1.1	Basisblöcke programmierbarer Logikblöcke	58
6.2.1.2	Netzlisten – das EDIF-Format	59
6.2.2	Beispiele	63
6.2.2.1	Hersteller 4	63
6.2.2.2	Hersteller 5	64
6.2.3	Schlussfolgerung für Modifikation der Komplexitätscharakteristika	65
7	ANWENDUNG DER METHODIK DER KOMPLEXITÄTSMESSUNG FÜR REALE EINSATZFÄLLE IN DER KERNTECHNIK	65
7.1	Anwendung auf CPU-basierte Steuerungen	65
7.1.1	Auswahl	65
7.1.2	Bestimmung der Kenngrößen	66
7.1.3	Ergebnisvergleich	73
7.2	Anwendung auf FPGA-basierte Steuerungen	78
7.2.1	Auswahl	78

Komplexität und Fehlerpotential		ISTec
7.2.2	Bestimmung der Kenngrößen	78
7.2.3	Ergebnisvergleich	80
8	VERÖFFENTLICHUNG DER IM VORHABEN ERZIELTEN ERGEBNISSE	81
9	BEWERTUNG DER ERGEBNISSE IM HINBLICK AUF DIE ZUVERLÄSSIGKEIT	82
9.1	Grundannahmen	82
9.2	Ableitung von Kriterien zur Bewertung der Zuverlässigkeit	83
10	ZUSAMMENFASSUNG	88
11	REFERENZEN	90
Anhang A	ISTec-A-2891, Teilbericht	92
Anhang B	Detaillierte Beschreibung des Skriptprogramms zur Berechnung der Ver- flechtungen (vfb)	93
Anhang C	Tool Validierung	103
Anhang D	Vortrag NPIC&HMIT	119
Anhang E	PowerPoint Folien zum Beitrag des ISTec zur NPIC	129
Anhang F	Interne Projektverfolgung	138

VERZEICHNIS DER BILDER

Bild 1.1:	Architekturtypen von programmierbarer Logik [11]	15
Bild 4.1:	Zusammenhang von $V_{\text{intern}}(\text{FP})$, $V_{\text{mod}}(\text{FP})$ und $V(\text{FP})$	29
Bild 5.1:	Eintrag der Verbindungen in einer Textdatei (Verbindungsliste) zu dem Funktionsplan in Bild 5.3.	32
Bild 5.2:	Benutzerschnittstelle des Skriptprogramms zur Auswertung von EDIF-Netzlisten	35
Bild 5.3:	Funktionsplan [2] zur Veranschaulichung der Matrizendarstellung	36
Bild 5.4:	Matrixdarstellung zum Funktionsplan aus Abbildung 5.3	37
Bild 5.5:	Gesamtstruktur der automatisierten Komplexitätsmessung [1]	38
Bild 5.6:	Auflösung von Verbindungen	39
Bild 5.7:	Beispiellauf 5.2.5 Skriptprogramm zur Anonymisierung von Funktionsplänen	40
Bild 5.8:	Validierung des Skriptprogramms zur Auflösung von internen Verbindungen in Funktionsplänen	41
Bild 5.9:	Validierung des Skriptprogramms zur Anonymisierung von Funktionsplänen	42
Bild 6.1:	APEX II Device Block Diagram [12]	58
Bild 6.2:	Look-Up Table	58
Bild 6.3:	MegaLAB Struktur der APEX II Familie [12]	60
Bild 6.4:	FPGA Architektur: Reihen- und Spaltenstruktur [12]	61
Bild 7.1:	Verschaltung der Funktionspläne in Funktion F1	70
Bild 7.2:	Verschaltung der Funktionspläne in Funktion F2	72
Bild 7.3:	Gegenüberstellung der Komponenten der Komplexitätsvektoren von S1 und S2	74
Bild 7.4:	Gegenüberstellung der Komponenten der Komplexitätsvektoren von A1-3, A4, A5 und A6	76
Bild 7.5:	Gegenüberstellung der Komponenten der Komplexitätsvektoren von F1 und F2	77
Bild 9.1:	Beispielhafter Kontrollflussgraph, erzeugt mit LDRA Testbed	86
Bild B.1:	Funktionsplan [2]	93
Bild B.2:	Schnittstellenbeschreibung des Funktionsplans aus Bild B.1	94
Bild B.3:	Nutzerinterface des Werkzeugs zur Berechnung der Verflechtung	95
Bild B.4:	Ergebnis Batchbetrieb mit grafischer Aufbereitung	102

VERZEICHNIS DER TABELLEN

Tabelle 1.1:	Technologien softwarebasierter Leittechnik - programmierbare Kontroller-Systeme	12
Tabelle 1.2:	Technologien softwarebasierter Leittechnik - programmierbare Schaltungen	13
Tabelle 3.1:	Komplexitätsmerkmale [1]	23
Tabelle 3.2:	Komponenten des Komplexitätsvektors nach [1]	27
Tabelle 4.1:	Komponenten K_{ex} des erweiterten Komplexitätsvektors	29
Tabelle 4.2:	Komponenten des modifizierten Komplexitätsvektors	30
Tabelle 4.3:	Komponenten des Komplexitätsvektors für FPGA-basierte Steuerungen	31
Tabelle 6.1:	Komplexitätsmatrix der Funktionsbausteine Teil 1 (Signale, Parameter)	44
Tabelle 6.2:	Komplexitätsmatrix der Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)	44
Tabelle 6.3:	Ermittlung der Bausteinkomplexität K9	45
Tabelle 6.4:	Komponenten des erweiterten Komplexitätsvektors	45
Tabelle 6.5:	Komplexitätsmatrix der L1 Funktionsbausteine Teil 1 (Signale, Parameter)	47
Tabelle 6.6:	Komplexitätsmatrix der L1 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)	48
Tabelle 6.7:	Ermittlung der Bausteinkomplexität für L1 Funktionsbausteine	48
Tabelle 6.8:	Ermittelbare Komponenten des erweiterten Komplexitätsvektors für einen L1 Funktionsplan	49
Tabelle 6.9:	Komplexitätsmatrix der L2 Funktionsbausteine Teil 1 (Signale, Parameter)	50
Tabelle 6.10:	Komplexitätsmatrix der L2 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)	50
Tabelle 6.11:	Ermittlung der Bausteinkomplexität für L2 Funktionsbausteine	51
Tabelle 6.12:	Ermittelbare Komponenten des erweiterten Komplexitätsvektors für einen L2 Funktionsplan	51
Tabelle 6.13:	Komplexitätsmatrix der L3 Funktionsbausteine Teil 1 (Signale, Parameter)	52
Tabelle 6.14:	Komplexitätsmatrix der L3 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)	53
Tabelle 6.15:	Ermittlung der Bausteinkomplexität für L3 Funktionsbausteine	53

Tabelle 6.16:	Ermittelbare Komponenten des erweiterten Komplexitätsvektors für einen L3 Funktionsplan	53
Tabelle 6.17:	Modifizierte Komplexitätsmatrix der Funktionsbausteine Teil 1 (Signale, Parameter)	55
Tabelle 6.18:	Modifizierte Komplexitätsmatrix der Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)	56
Tabelle 6.19:	Komponenten des modifizierten Komplexitätsvektors	56
Tabelle 6.20:	Komponenten des Komplexitätsvektors für FPGA Netzlisten	62
Tabelle 6.21:	Ermittelbare Komponenten des Komplexitätsvektors einer FPGA Netzliste von Hersteller 4	63
Tabelle 6.22:	Ermittelbare Komponenten des Komplexitätsvektors einer FPGA Netzliste von Hersteller 5	64
Tabelle 6.23:	Komplexitätsvektor für FPGA-basierte Steuerungen	65
Tabelle 7.1:	Modifizierte Komplexitätsmatrix der L1 Funktionsbausteine Teil 1 (Signale, Parameter)	67
Tabelle 7.2:	Modifizierte Komplexitätsmatrix der L1 Funktionsbausteine Teil 2 (Interne Variable, Ressourcenbedarf)	67
Tabelle 7.3:	Modifizierte Tabelle zur Ermittlung der Bausteinkomplexität für L1 Funktionsbausteine	69
Tabelle 7.4:	Komplexitätsvektoren der einzelnen Funktionspläne der Funktion F1	70
Tabelle 7.5:	Komplexitätsvektoren der Funktion F1 mit und ohne Berücksichtigung der Funktionspläne B1 bis B3	71
Tabelle 7.6:	Komplexitätsvektoren der einzelnen Funktionspläne und der Funktion F2	72
Tabelle 7.7:	Komplexitätsvektoren weiterer Funktionspläne von Hersteller 1	73
Tabelle 7.8:	Gegenüberstellung der Komplexitätsvektoren der Funktionspläne S1 und S2	74
Tabelle 7.9:	Gegenüberstellung der Komplexitätsvektoren der Funktionspläne der Aggregate A1 bis A3 und A4 bis A6	75
Tabelle 7.10:	Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2	77
Tabelle 7.11:	Komplexitätsvektoren der FPGA Anwendung von Hersteller 4 für die hierarchische aufgebauten Designebenen	79
Tabelle 7.12:	Komplexitätsvektoren der FPGA Anwendung von Hersteller 5 für die Designs 1, 2 und 3	80
Tabelle 7.13:	Gegenüberstellung der Komplexitätsvektoren von FPGA-basierten Steuerungen	81

Tabelle 9.1:	Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2	84
Tabelle 9.2:	Gegenüberstellung der Komplexitätsvektoren von FPGA-basierten Steuerungen	85
Tabelle 9.3:	Bewertungskriterien	87

REVISIONSBLATT

Datum	Rev.	Änderungen	Bearbeiter
16.08.2017	0	Entwurf	
22.09.2017	1	Berücksichtigung der Anmerkungen des Auftraggebers Redaktionelle Korrekturen	Heigl, Lindner

KURZFASSUNG

Die Einführung und Weiterentwicklung immer komplexerer verfahrenstechnischer Systeme in den verschiedensten Bereichen führt auch in der Kerntechnik zu einer erhöhten Verwendung softwarebasierter Leittechniksysteme mit sicherheitstechnischer Bedeutung. Die Frage nach einer Ermittlung der Zuverlässigkeit softwarebasierter Systeme mithilfe eines allgemein anerkannten Verfahrens ist dabei noch nicht geklärt.

Wie die praktische Erfahrung zeigt, hängt das Risiko einer Fehlfunktion eines Software-Systems entscheidend von der Komplexität der Software in Verbindung mit der Vielfalt der Anwendungsprofile ab. Deshalb stellt sowohl die Bestimmung der Komplexität der Software als auch der Ansatz, die Software-Zuverlässigkeit auf der Grundlage der Software-Komplexität zu quantifizieren, eine wichtige Zielsetzung dar.

In dem vorliegenden Bericht wird eine Methodik zur Messung der Komplexität der Software CPU-basierter Steuerungen und deren Anwendbarkeit zur Messung der Komplexität programmierbarer Logik FPGA-basierter Steuerungen dargestellt. Aus den verschiedenen Komplexitätscharakteristiken wird ein Komplexitätsvektor gebildet, womit der Vielschichtigkeit der Komplexität digitaler Leittechniksysteme Rechnung getragen wird. Der Komplexitätsvektor ist Ausgangspunkt für die Ableitung von Kriterien zur Bewertung der Zuverlässigkeit digitaler Leittechniksysteme.

Das Komplexitätsmessverfahren wird im vorliegenden Bericht beschrieben und seine praktische Anwendbarkeit auf graphisch spezifizierte Leittechnikfunktionen durch die exemplarische Anwendung auf Leittechniksysteme verschiedener Hersteller nachgewiesen. Dabei wird die Aussagekraft des Verfahrens anhand der Gegenüberstellung der Komplexitätsvektoren zweier ähnlicher Anwendungen aus der Kerntechnik verifiziert.

ABSTRACT

Introduction and advancement of more complex technical systems in many branches is accompanied by the implementation of computerized I&C systems important to safety. The determination of dependability of software-based I&C systems by means of a commonly accepted method is not yet solved.

Practical experiences show that the complexity of software systems in conjunction with the variety of use cases is essential for the risk of malfunction of the software. Thus, both determination of the complexity of software and the approach to quantify the reliability of software on the basis of its complexity are an essential goal.

In the present report a method is described for measuring of the complexity of CPU software as well as of programmable (FPGA) logic and its applicability is demonstrated. A complexity vector is compiled from different complexity characteristics taking into account the multifaceted nature of the notion of software complexity. The complexity vector is the starting point for deduction of criteria for the evaluation of the reliability of digital I&C systems.

The complexity measurement procedure is described in the present report and its practical usability for graphically specified I&C functions is demonstrated by application to I&C systems of different suppliers. The significance of the method is verified by comparison of the complexity vectors of two similar nuclear applications.

1 EINLEITUNG

1.1 Beauftragung

Mit Schreiben vom 29.01.2015 beauftragte das Bundesamt für Strahlenschutz (BfS) im Auftrag des Bundesministeriums für Umwelt, Naturschutz, Bau und Reaktorsicherheit (BMUB) die TÜV Rheinland ISTec GmbH mit der Erarbeitung von Methoden zur sicherheitstechnischen Bewertung des Funktionsverhaltens von softwarebasierter Leittechnik auf der Basis der Ermittlung der Komplexität der eingesetzten Geräte und Systeme. Diese Arbeiten sollen auch zur Aktualisierung des deutschen und internationalen kerntechnischen Regelwerks beitragen. Die im Vorhaben angewandte und dabei erweiterte Methodik setzt auf den Ergebnissen von Vorläuferprojekten auf [1], [2].

Der Ablauf des Vorhabens wird aus der internen Projektverfolgung (siehe Anhang F) ersichtlich. Die Protokolle der Projektmeetings sind aufgrund darin enthaltener vertraulicher Herstellerinformationen, die im Rahmen des Vorhabens zur Erhebung von Daten kontaktiert wurden, dem Bericht nicht beigelegt. Die im Angebot zum Vorhaben erstellte Planung wurde eingehalten.

1.2 Zielsetzung

Die sicherheitstechnische Bewertung des Funktionsverhaltens von softwarebasierter Leittechnik sowohl von Komplettsystemen als auch in der Peripherie des Sicherheitssystems ist nach wie vor für die deutschen Anlagen relevant, da davon auszugehen ist, dass auch in der Nachbetriebsphase bis in die Phase der Stilllegung hinein über etwa 15 bis 20 Jahre Leittechniksysteme mit beschränkter Funktionalität oder Austauschkomponenten eingesetzt werden.

Darüber hinaus sind in einigen benachbarten ausländischen Alt- und Neuanlagen softwarebasierte Komplettsysteme für die Sicherheitsleittechnik im Einsatz.

Diese softwarebasierte Leittechnik beruht maßgeblich auf Technologien wie Programmable Logical Devices (PLD) oder Field Programmable Gate Arrays (FPGA).

Wie praktische Erfahrungen zeigen, hängt die Wahrscheinlichkeit für eine Fehlfunktion leittechnischer Systeme entscheidend von der Komplexität dieser Systeme ab. Das gilt insbesondere für softwarebasierte leittechnische Systeme, deren Funktionalität wesentlich von der Software bestimmt wird. Deshalb stellt die Bestimmung der Komplexität der Software eine wichtige Aufgabe zur Bewertung ihrer Sicherheit dar, wenn man dem Ansatz, die Software-Zuverlässigkeit auf der Grundlage der Software-Komplexität zu bewerten, folgt

Hierzu hat die TÜV Rheinland ISTec GmbH im Rahmen vorangegangener Projekte die Grundlagen einer Methode zur Messung der Komplexität der Anwendungssoftware von Automatisierungssystemen und -geräten entwickelt. Die der Messung zugrundeliegende Metrik ist auf die Spezifika der Software digitaler Leittechniksysteme, d.h. auf Software, die durch die Verschaltung von Funktionsbausteinen zu Funktionsplänen gekennzeichnet ist, ausgerichtet. Die Methode wurde an einem funktionsplanprogrammierten digitalen Leittechniksystem (TXS) getestet [1].

Im aktuellen Projekt wird eine applikationsabhängige Methode mit einem zugehörigen Instrumentarium zur Bestimmung der Software-Komplexität entwickelt, die für unterschiedliche

Softwarestrukturen anwendbar ist. Dabei erstreckt sich der Anwendungsbereich von Anwendungsprogrammen für CPUs bis zu FPGA-Designs. Anhand der mit dieser Methode ermittelten Komplexitätsbewertung werden Schlüsse bezüglich der Zuverlässigkeit der jeweiligen Anwendungssoftware abgeleitet.

Die Bewertung der Komplexität ist umso gebotener, da das deutsche kerntechnische Regelwerk neben Anforderungen zur Zuverlässigkeit auch Anforderungen an die Begrenzung der Komplexität von Sicherheitsleittechnik enthält [16], [17].

1.3 Technologien

Mit der schnellen Entwicklung der Mikroelektroniktechnologien in den 1970er und 80er Jahren wurden zwei Entwicklungen initiiert die für die Leittechnik von Bedeutung sind.

Einerseits wurde durch die Verbesserung der Herstellungstechnologien eine zunehmende Integration von Funktionalität auf integrierten Prozessor-Schaltkreisen möglich. Die Entwicklung verlief dabei von zentralen Rechenanlagen zur Steuerung und Regelung von Prozessen zu dezentralen Industrie-Computern und zur Anwendung von Mikrocontrollern bzw. Speicherprogrammierbare Steuerung (SPS; engl. Programmable Logic Controller - PLC oder Programmable Automation Controller – PAC).

Tabelle 1.1: Technologien softwarebasierter Leittechnik - programmierbare Controller-Systeme

Bezeichnung	Beschreibung
µC / MCU	<p>Mikrocontroller</p> <ul style="list-style-type: none"> - sind integrierte Schaltkreise die Verarbeitungseinheit/Prozessor, Speicher und Peripheriefunktionen wie A/D-Wandler oder Interface- und Netzwerktreiber auf einem Chip vereinen - ca. ab Mitte der 1970 verfügbar
PLC	<p>Programmable Logic Controller (→SPS)</p> <ul style="list-style-type: none"> - ein Gerät für die Automation von typischen Industrieprozessen - ca. ab Mitte der 1970 verfügbar
PAC	<p>Programmable Automation Controller (→SPS)</p> <ul style="list-style-type: none"> - stellt eine Weiterentwicklung der PLCs dar, die die Steuerungs- und Regelungsmöglichkeiten von PLC mit der Flexibilität eines IPCs bei Darstellungen und Berechnungen kombiniert - ca. ab 2000 verfügbar

Bezeichnung	Beschreibung
SPS	<p>Speicherprogrammierbare Steuerung</p> <ul style="list-style-type: none"> - überwiegend basierend auf Mikrocontrollern - moderne leistungsfähige SPS sind meist modular aufgebaut, bestehend aus einer Zentralen Verarbeitungseinheit und einer variablen Anzahl von Ein-/Ausgabebaugruppen für Ansteuerung dezentrale Peripheriebaugruppen - ca. ab Mitte der 1970 verfügbar
IPC	<p>Industrie-PC</p> <ul style="list-style-type: none"> - im Unterschied zu Büro-PC für Einsatzbereiche wie Prozessvisualisierung, Industrierobotersteuerung, Test- und Prüfstände oder Industrieautomation vorgesehen - physikalisch robuster gegen Umwelteinflüsse als Büro-PCs

Diese Entwicklung ermöglichte einen zunehmenden Einsatz softwarebasierter Systeme auf allen Leittechnikebenen.

Andererseits wurde es möglich mit der Verbesserung der Herstellungstechnologien integrierter Schaltkreise (IC - Integrated Circuit) anwendungs- bzw. kundenspezifische Schaltkreise (ASIC - Application Specific Integrated Circuit) und programmierbare Schaltkreise (PLD - Programmable Logic Device) wie z.B. PROMs (Programmable Read-Only Memory) oder PLA (Programmable Logic Array) kostengünstig zu produzieren.

Über FPLAs (Field Programmable Logic Array) ab Mitte der 1980er führte die Entwicklung zu den aktuellen hochintegrierten CPLDs (Complex Programmable Logic Device) und FPGAs (Field Programmable Gate Array) der heutigen Zeit.

Tabelle 1.2: Technologien softwarebasierter Leittechnik - programmierbare Schaltungen

Bezeichnung	Beschreibung
PLD	<p>Programmable Logic Device (z.B. PLA, PROMs)</p> <ul style="list-style-type: none"> - programmierbare integrierter Schaltkreise - ca. ab 1980 verfügbar

Bezeichnung	Beschreibung
PLA, FPLA	Programmable Logic Array, Field Programmable Logic Array <ul style="list-style-type: none"> - integrierter Schaltkreise mit programmierbare Logik-Feldern die logische Grundverknüpfungen wie UND/ODER- Gatter, Addierer, Zähler, Flipflops usw. ermöglichen - mit geringer bis mittlerer Integrationsgrad - ca. ab 1980 verfügbar
PAL	Programmable Array Logic, sind spezielle PLAs <ul style="list-style-type: none"> - integrierter Schaltkreise mit programmierbaren Logik-Feldern (nur UND-Verknüpfungen) - geringer Integrationsgrad - ca. ab 1980 verfügbar
CPLD / SPLD	Complex Programmable Logic Device <ul style="list-style-type: none"> - bestehen aus zusammengeschalteten Simple Programmable Logic Device - CPLDs sind integrierte Schaltkreise mit: <ul style="list-style-type: none"> - programmierbaren UND/ODER-Matrix und logischen Funktionen - programmierbaren Verknüpfungslogiken und Rückkopplungen mit bestimmbarem Zeitverhalten - programmierbaren Eingabeblöcke/ Ausgabeblöcke - mit hohem bis sehr hohem Integrationsgrad (VLSI-Technologie, bis 10.000 Gatter), - ca. ab Anfang der 1990 verfügbar
FPGA	Field Programmable Gate Array <ul style="list-style-type: none"> - integrierte Schaltkreise mit: <ul style="list-style-type: none"> - programmierbaren Logik-, Speicher-, Ein- und Ausgabe- und Routingfeldern - verschieden programmierbarer Verknüpfungslogik mit Einfluss auf das Zeitverhalten der Logikpfade - sehr hoher Integrationsgrad (VLSI-Technologie, bis 100.000 Gatter-Äquivalente) - ca. ab erste Hälfte der 1990 verfügbar

Der Entwicklungsgang der unterschiedlichen Technologien, bis hin zum System on a Chip (SOC), ist in Bild 1.1 dargestellt.

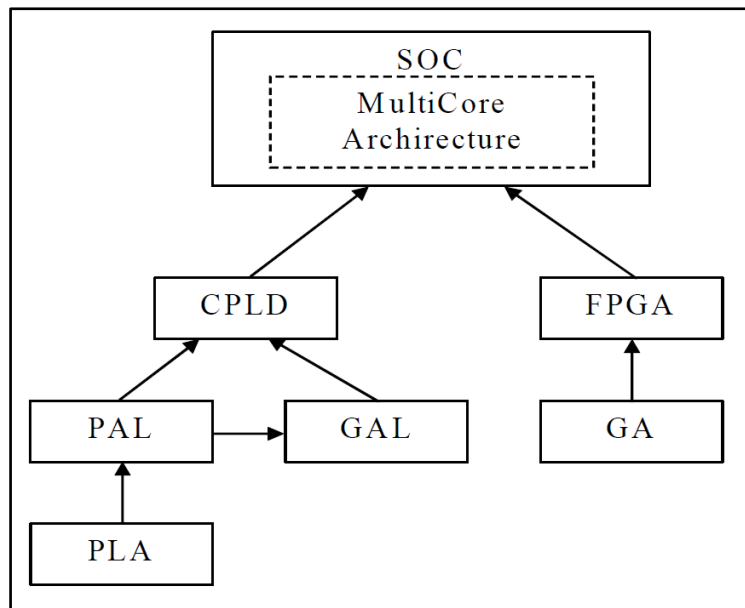


Bild 1.1: Architekturtypen von programmierbarer Logik [11]

In [11] werden FPGAs (Field Programmable Gate Array) und CPLDs (Complex Programmable Logic Device) wie folgt gegenübergestellt. CPLDs haben ihren Ursprung in PALs (Programmable Arrays Logic), die wiederum aus den PLAs (Programmable Logic Arrays) hervorgegangen sind und GALs (Generic Arrays Logic). Die funktionelle Einheit der CPLD besteht aus Mikrozellen, von denen jede eine kombinatorische UND/ODER Registerfunktion ausführt. Die funktionelle Logik in einem Block ist eine Matrix logischer Terme. Über ein Term-Verteilungs-Diagramm kann jede Mikrozele auf eine Teilmenge dieser Terme zugreifen. Umschaltmatrizen ordnen die Signale der Ausgänge der funktionalen Einheit und der E/A-Einheit zu. Im Gegensatz zu FPGAs, bei denen die Verbindungen segmentiert sind (parallele Abarbeitung), weisen CPLDs ein kontinuierliches System von Verbindungen (serielle Abarbeitung) auf. Die FPGA Architektur hat ihren Ursprung in verbundenen GAs (Gate Array). Die interne Architektur besteht aus einem Satz konfigurierbarer logischer Einheiten, die regulär angeordnet und mit Leitkanälen verbunden sind. Module wie z.B. Transistor-Paare, logische Zellen, und Look-Up Tables (LUT) werden als konfigurierbare Logikblöcke verwendet. Diese Module weisen wiederum eine gegliederte Architektur von internen Verbindungen auf.

2 BEGRIFFE UND ABKÜRZUNGEN

2.1 Begriffe

Basisblock

Beschreibung der grundlegenden Elementarfunktionen in der Netzliste einer FPGA Anwendung

Anmerkung: Die leittechnischen Funktionen einer FPGA Anwendung ergeben sich aus der Verschaltung der Basisblöcke, die aus einer Hierarchie von Bibliotheken in der Netzliste des FPGAs hervorgehen.

Complex Programmable Logic Device

Integrierte Schaltkreise mit

- programmierbaren UND/ODER-Matrix und logischen Funktionen,
- programmierbaren Verknüpfungslogiken und Rückkopplungen mit bestimmbarem Zeitverhalten und
- programmierbaren Eingabe-/ Ausgabeblocks

Anmerkung: CPLDs besitzen einen hohen bis sehr hohen Integrationsgrad (VLSI-Technologie, bis 10.000 Gatter).

Computer-basierte Einheit

Einheit, deren Funktion auf Softwareanweisungen, die auf einen Mikroprozessor oder einen Mikrocontroller ablaufen, beruht

Anmerkung 1: In dieser Begriffsdefinition kann der Terminus "Einheit" durch "System" oder „Einrichtung“ oder „Gerät“ ersetzt werden.

Anmerkung 2: Computer-basierte Einheiten sind programmierbare digitale Einheiten.

Anmerkung 3: Dieser Begriff ist zum Begriff software-basierte Einheit äquivalent.

Effektive Komplexität

Grad der Schwierigkeit, um das Design, die Implementierung oder das Verhalten eines Systems oder einer Komponente zu verstehen und zu verifizieren [6]

Anmerkung 1: Diese Definition besitzt eine hochgradig subjektive Komponente. Folgendes Beispiel möge diesen Sachverhalt illustrieren.

Eine relativ einfache leittechnische Funktion werde durch

- einen Programmierer mit einem Computerprogramm in C,
- einen Elektroniker mit einer einfachen elektronischen Schaltung mit Operationsverstärkern, Gattern, Widerständen und Kondensatoren,
- von einem Systemingenieur auf logischer Ebene mit abstrakten Funktionsbausteinen

realisiert. Jeder Fachexperte wird die Lösung des anderen Fachexperten für schwerer zu verstehen und zu verifizieren ansehen als seine eigene.

Anmerkung 2: Dieser Begriff wird auf Grund seiner Abhängigkeit von subjektiven Faktoren in diesem Bericht nicht verwendet.

Elektrische/Elektronische/Programmierbare Elektronische Einheit (E/E/PE Einheit):

Einheit, die auf elektrischer (E) und/oder elektronischer (E) und/oder programmierbarer elektronischer (PE) Technologie basiert

Anmerkung 1: In dieser Begriffsdefinition kann der Terminus "Einheit" durch "System" oder „Einrichtung“ oder „Gerät“ ersetzt werden.

Festverdrahtete Einheit

Einheit, deren Funktion auf Relais, analoger Elektronik oder diskreter digitaler Logik beruht

Anmerkung 1: In dieser Begriffsdefinition kann der Terminus "Einheit" durch "System" oder „Einrichtung“ oder „Gerät“ ersetzt werden.

Anmerkung 2: Festverdrahtete Einheiten werden auch konventionelle Einheiten genannt.

Field Programmable Gate Array

Integrierte Schaltkreise mit

- programmierbaren Logik-, Speicher-, Ein- und Ausgabe- sowie Routingfeldern und
- verschiedenen programmierbaren Verknüpfungslogiken mit Einfluss auf das Zeitverhalten der Logikpfade

Anmerkung: FPGAs besitzen einen sehr hohem Integrationsgrad (VLSI-Technologie, bis 100.000 Gatter-Äquivalente),

Funktionsbaustein

Logiksymbol in einer graphisch orientierten Programmiersprache

Anmerkung: Mit Hilfe eines Funktionsbausteins können neben Elementarfunktionen nahezu beliebige Funktionalitäten implementiert werden. Aus der Verschaltung von Funktionsbausteinen, dem Funktionsplan, ergeben sich die leittechnischen Funktionen einer Speicherprogrammierbaren Steuerung.

HDL-programmierte Baugruppe [18]

Integrierter Schaltkreis der (für kerntechnische Leittechniksysteme) mit Hardwarebeschreibungssprachen (HDL) und zugehörigen Softwarewerkzeugen konfiguriert wird

Anmerkung 1: Hardwarebeschreibungssprachen und die zugehörigen Werkzeuge (z.B. Simulatoren, Synthesewerkzeuge) werden benutzt, um die Anforderungen in eine geeignete Anordnung vorgefertigter mikroelektronischer Ressourcen umzusetzen.

Anmerkung 2: In der Entwicklung von HDL-programmierten Baugruppen (HPD) können vorgefertigte Blöcke (PDB) benutzt werden.

Anmerkung 3: HDL-programmierte Baugruppen (HPD) basieren typischerweise auf leeren FPGAs, PLDs oder ähnlichen mikroelektronischen Technologien.

Anmerkung 4: HDL-programmierte Baugruppen sind programmierbare digitale Einheiten.

Integrationsgrad

Anzahl von Transistoren in einem integrierten Schaltkreis

Anmerkung: Der Integrationsgrad ergibt sich aus der Integrationsdichte (Anzahl Transistoren pro Flächeneinheit) und der Chipgröße (Fläche des Chips).

Komplexität

Anmerkung: Es gibt keine allgemeine Definition des Begriffs „Komplexität“, deshalb wird dieser Begriff in unterschiedlichen Gebieten unterschiedlich definiert. Daher soll in diesem Bericht „Komplexität“ als nachvollziehbare Kombination messbarer Größen angesehen werden. Sowohl die Kombination als auch die einzelnen Messgrößen werden als Komplexitätsmaß (bzw. synonym als Komplexitätsmetrik) bezeichnet.

Komplexitätsanalyse

Prozess zur Bestimmung eines geeigneten Komplexitätsvektors für eine Klasse von Objekten

Komplexitätsbewertung

Prozess der Beurteilung einer Komplexitätsmessung

Komplexitätsmessung

Prozess der Bestimmung der Werte des Komplexitätsvektors für ein bestimmtes Objekt einer Objektklasse

Komplexitätsmetrik, Komplexitätsmaß

Softwarequalitätsmetrik, die die Qualitätseigenschaft „Komplexität“ abbildet

Anmerkung 1: Hauptsächliche Komplexitätsmetriken sind das McCabe-Maß (zyklomatische Komplexität), die Knotenmetrik und das Verknüpfungsmaß nach ISTec-A-1569 [1].

Anmerkung 2: Die zyklomatische Komplexität und die Knotenmetrik sind Kontrollflussmaße; Das Verknüpfungsmaß nach ISTec-A-1569 [1] ist ein Datenflussmaß.

Komplexitätsvektor

Zusammenfassung von Komplexitätswerten in einem Vektor

Komplexitätswert

Messbare oder berechenbare Maßzahl

Anmerkung: Dieser Wert kann eine Komponente eines Komplexitätsvektors sein.

Look-Up Table (LUT)

Programmierbare Logikgatter eines FPGAs, die die technische Realisierung der Basisblöcke darstellen

Programmable Array Logic

Integrierter Schaltkreise mit programmierbaren Logik-Feldern (nur UND-Verknüpfungen) und geringem Integrationsgrad

Programmable Automation Controller

Weiterentwicklung des PLCs, die die Steuerungs- und Regelungsmöglichkeiten von PLCs mit der Flexibilität eines IPCs bei Darstellungen und Berechnungen kombiniert

Programmable Logic Array, Field programmable Logic Array

Integrierter Schaltkreis mit programmierbaren Logik-Feldern, die logische Grundverknüpfungen wie UND/ODER-Gatter, Addierer, Zähler, Flipflops usw. ermöglichen

Anmerkung: Der Integrationsgrad ist gering bis mittel.

Programmable Logic Controller

Gerät für die Automation von typischen Industrieprozessen

Programmable Logic Device (PLD)

Programmierbarer integrierter Schaltkreis (z.B. PLA, PROM)

Programmierbare digitale Einheit

Einheit, deren Funktion auf Softwareanweisungen oder programmierbarer Logik beruht

Anmerkung: In dieser Begriffsdefinition kann der Terminus "Einheit" durch "System" oder „Einrichtung“ oder „Gerät“ ersetzt werden.

Softwarequalitätsmetrik

Funktion, die eine Software-Einheit in einen Zahlenwert abbildet, der als der Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit interpretierbar ist [7]

Anmerkung: Beispiele für Software-Qualitätseigenschaften sind z.B. die Anzahl der Programmzeilen (Lines of Code, LOC), die zyklomatische Komplexität nach McCabe und das Verflechtungsmaß nach [1].

Speicherprogrammierbare Steuerung

Gerät für die Automation von typischen Industrieprozessen

Zuverlässigkeit

Wahrscheinlichkeit, dass eine Vorrichtung, ein System oder eine Anlage die vorgesehene Funktion unter festgelegten Betriebsbedingungen zufriedenstellend während eines genau bezeichneten Zeitraumes erfüllen wird [15]

2.2 Abkürzungen

ANSI	American National Standards Institute
ASIC	Application Specific Integrated Circuit
BBN	Bayesian Belief Network
BfS	Bundesamt Für Strahlenschutz
BMUB	Naturschutz Bau Und Reaktorsicherheit
CAM	Content-Adressable Memory
CD-ROM	Compact Disk Read Only Memory
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
CSV	Comma-separated values
E/A	Ein-/Aus- (in Wortverbindungen)
E/E/PE	Elektrische/Elektronische/Programmierbare Elektronische Einheit
EDIF	Electronic Design Interchange Format
EN	Europäische Norm
ESB	Embedded System Block
FB	Funktion Block
FD	Function Diagram
FIFO	First In, First Out
FP	Funktionsplan
FPGA	Field Programmable Gate Arrays
FPLA	Field Programmable Logic Array

GAL	Generic Arrays Logic
HARMONICS	Harmonised Assessment of Reliability of MOdern Nuclear I&C Software
HDL	Hardware Description Language (Hardwarebeschreibungssprache)
HPC	High Performance Computing
HPD	HDL programmed device (HDL programmiertes Gerät)
HW	Hardware
I&C	Instrumentation and Control
IAEA	International Atomic Energy Agency
IC	Integrated Circuit
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IOE	I/O element (E/A-Element)
IPC	Industrie-PC
ISOFIC	International Symposium on Future I&C for Nuclear Power Plants
KKW	Kernkraftwerk
LAB	Logic Arrays Block
LD	Logic Diagram
LE	Logic Element
LOC	Lines of Code
LUT	Look-Up Table
MCU	Main Control Unit
NPIC&HMIT	Nuclear Plant Instrumentation, Control & Human-Machine Interface Technologies
NPP	Nuclear Power Plant
PAC	Programmable Automation Controller
PAL	Programmable Array Logic
PDB	Pre-developed block (Vorgefertigter Block)
PDF	Portable Document Format
PLA	Programmable Logic Array
PLC	Programmable Logical Devices
PLD	Programmable Logic Device
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
SAFECOMP	International Conference on Computer Safety, Reliability and Security
SOC	System On A Chip
SPLD	Simple Programmable Logic Device
SPS	Speicherprogrammierbare Steuerung
SRAM	Static Random Access Memory
TÜV	Technischer Überwachungsverein
TXS	TELEPERM XS
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e.V.
VDI	Verein Deutscher Ingenieure

3 STAND VON WISSENSCHAFT UND TECHNIK ZUR KOMPLEXITÄTSMESSUNG SOFTWAREBASIERTER LEITTECHNIK

3.1 Komplexitätsmetriken

3.1.1 In der Literatur dokumentierter Stand für Computerprogramme

Der Teilbericht [3] (siehe Anhang A) fasst unter anderem den Stand von Wissenschaft und Technik zur Komplexitätsmessung von softwarebasierter Leittechnik zusammen. Es zeigte sich, dass sich für Software im Allgemeinen eine Komplexität nur durch sehr allgemeine und damit auch nur entsprechend ungenaue Eigenschaften beschreiben lässt, wobei meist bestimmte Facetten des seinerseits komplizierten Komplexitätsbegriffs im Vordergrund stehen. Die für manuell programmierte Software vorrangig herangezogenen Komplexitätsmaße (z.B. Komplexität nach McCabe) haben für automatisch generierten Code leittechnischer Anwendungen wenig Aussagekraft [1]. Bei digitalen Leittechnik-Systemen, die mittels Integrierter Entwicklungsumgebungen realisiert werden, kann die Komplexität mit der in [1] entwickelten Methode deutlich exakter betrachtet werden. Die quantitative Bestimmung der Zuverlässigkeit softwarebasierter Systeme anhand eines standardisierten, allgemein akzeptierten Verfahrens ist nach wie vor ein offenes Problem.

Als Konsequenz aus dieser Situation wurde im Rahmen des vorherigen Projekts (Vorhaben-Nr. 1501334) ein Komplexitätsmaß entwickelt, das nicht nur auf einer einfachen Maßzahl basiert, sondern das einen Komplexitätsvektor beinhaltet, der unterschiedliche Maßzahlen für die Komponenten der Software digitaler Leittechniksysteme kombiniert. Die Wirksamkeit von Verfahren, die situationsbezogen mehrere Metriken kombinieren, wird durch Smidts et.al. [4] bestätigt. In [4] wird explizit darauf verwiesen, dass die Kombination der Metriken von der Art der zu bewertenden Software abhängig ist.

3.1.2 In der Literatur dokumentierter Stand für programmierbare Logik (FPGAs)

Für FPGAs sind keine Metriken, die zur Bewertung der Zuverlässigkeit einer FPGA-Anwendung herangezogen werden können, bekannt. In [5] wird erwähnt, dass eine Komplexitätsbewertung von FPGAs unter Anwendung von Komplexitätsmaßen zur Fehlerminimierung genutzt werden kann, indem komplexe Programmblöcke in kleinere Einheiten zerlegt werden. Es wird aber in [5] keine konkrete Komplexitätsmetrik angegeben.

In [8] werden vier zu messende Kenngrößen von FPGA-Anwendungen als wesentlich herausgestellt:

- Korrekte Funktion ("Ability of the program to produce the correct answers"),
- Leistungsfähigkeit ("Performance of the FPGA implementation of the program"),
- Portierbarkeit ("Portability of the FPGA implementation of the program"),
- Wirtschaftlichkeit ("Time and effort required to produce the FPGA implementation of the program").

Insbesondere der erste Anstrich macht deutlich, dass der Nachweis der Zuverlässigkeit einer programmierbaren Logikschaltung ganz wesentlich durch ihre Komplexität bestimmt wird.

Andere Quellen wie z.B. [9] und [10] befassen sich mit Metriken zur Verbesserung der Verteilung der Logikblöcke und zur Verbesserung des Routings.

Auf Grund der Tatsachen, dass die in diesem Projekt angewandten Verflechtungsmaße (Definitionen siehe 3.1.3 und 4.1) Datenflussmaße sind und dass die Eigenschaften von FPGA-Anwendungen wegen der massiv parallelen Arbeitsweise dieser Schaltkreise wesentlich durch den Datenfluss bestimmt werden, besitzen die Verflechtungsmaße ein hohes Potenzial für die Bestimmung der Komplexität einer FPGA-Anwendung.

3.1.3 Vorangegangene eigenen Projekte

Durch die zunehmende Bedeutung der Software in sicherheitsrelevanten Systemen wird eine Zuverlässigkeitsbewertung dieser Software dringend benötigt. Die erforderlichen Nachrüstungen und Modernisierungen analoger Leittechnik-Systeme in Kernkraftwerken durch digitale Systeme verlangen Aussagen zur Zuverlässigkeit und Verfügbarkeit dieser digitalen Systeme. Dabei handelt es sich bei der Anwendungssoftware digitaler Leittechnik-Systeme in KKW's typischerweise um typgeprüfte, hochqualifizierte Software, die auf der Grundlage einer graphischen Spezifikation von Codegeneratoren als normierter Code mit vordefinierten Eigenschaften erzeugt wird. Der Mangel an geeigneten Modellen um eine Aussage über die Zuverlässigkeit zu treffen führte in einer Reihe von Technologiebereichen zur Anwendung stringenter Entwicklungsvorschriften (z.B. im Bereich der Avionik DO-178B, im kerntechnischen Bereich EN IEC 61513 und EN IEC 60880).

Wie die praktische Erfahrung zeigt, hängt das Risiko für eine Fehlfunktion eines Software-Systems entscheidend von der Komplexität der Software in Verbindung mit der Vielfalt der Anwendungsprofile ab. Deshalb stellt sowohl die Bestimmung der Komplexität der Software per se eine wichtige Aufgabe dar, wie auch als Option im Hinblick auf den prinzipiellen Ansatz, die Software-Zuverlässigkeit auf der Grundlage der Software-Komplexität zu bewerten.

In ISTec-A-1569 [1] wurden die Ergebnisse zur Messung der Komplexität und der zu erwartenden Zuverlässigkeit der Anwendungssoftware von Automatisierungssystemen und -geräten dargestellt. Durch die Eingrenzung auf Software die durch die Verschaltung von Funktionsbausteinen zu Funktionsplänen gekennzeichnet ist, wurde es möglich, eine Komplexitätsmetrik zu definieren, die die Spezifika der Software leittechnischer Systeme berücksichtigt.

Die Methode zur Messung der Komplexität stützt sich auf ein strukturiertes Vorgehen, bei dem bottom-up zuerst die Funktionsbausteine und dann die Funktionspläne bewertet werden können. Um dieses Konzept für verschiedene digitale Leittechniksysteme verwendbar zu gestalten, wurde die Bewertung der Funktionsbausteine und Funktionspläne unabhängig voneinander vollzogen, d.h. die Bewertung der Funktionspläne ist unabhängig von der Bewertung der Komplexität der Funktionsbausteine möglich.

Zur Bewertung der Komplexität der Funktionsbausteine wurden zwei komplementäre Sichtweisen, der Black-Box-View und der White-Box-View, angewandt. Diese Dualität erlaubt es, die Komplexitätsbewertung der Funktionsbausteine auch ohne Zugriff auf den Quellcode der Funktionsbausteine auszuführen. Zur Bewertung der Komplexität der Funktionspläne wurde ein Verflechtungsmaß definiert. Die Berechnung des Verflechtungsmaßes basiert auf der Verschaltung der Funktionsbausteine und Eingabe-Signale des Funktionsplans. Als weitere Komplexitätsmaße wurden die internen Speicher und die Parametrierbarkeit der Funktions-

bausteine berücksichtigt, womit der Variabilitäts-Komplexität digitaler Leittechnik-Systeme Rechnung getragen wird.

In Tabelle 3.1 sind Komplexitätsmerkmale dargelegt, die im Rahmen vorangegangener eigener Projekte im Zusammenhang mit der leittechnischen Systemplattform Teleperm XS ermittelt wurden.

Tabelle 3.1: Komplexitätsmerkmale [1]

Merkmal	Begründung
Informational fan-out	In [14] ¹ wurde eine signifikante Korrelation zwischen diesem Merkmal und der Häufigkeit von Fehlern in der Software sowie mit der Zahl der Änderungen festgestellt. Dieses Merkmal kann z.B. aus den Daten des LDRA-Testbeds ² ermittelt werden, wenn der Quellcode verfügbar ist. ³
Signale	Die Zahl der Ein-/Ausgangssignale sowie das Vorhandensein eines Meldesignals bezieht sich auf der möglichen Komplexität der Kopplung des analysierten Funktionsbausteins mit anderen.
Parameter	Die Parameterwerte können durch Spezifikations- oder Übertragungsfehler zu einem Fehlverhalten der leittechnischen Funktion führen. Nicht änderbare Parameter werden ein einziges Mal spezifiziert, bei der (automatischen) Erstellung der Software. Änderbare Parameter werden im Laufe eines Brennelement-Zyklus mehrmals geändert. Hierbei sind weitere Angaben wie z. B. wie oft diese Parameter im Laufe eines Brennelement-Zyklus geändert werden für eine Beurteilung der Fehlerwahrscheinlichkeit relevant
Zustandspeicher	Hängt direkt mit der Komplexität eines Funktionsbausteins zusammen.
Speicherbedarf	Da bei TELEPERM XS nur eine statische Speicherverwaltung implementiert wurde, genügen die Angaben des absoluten Speicherbedarfs für Code, Daten und Stack zur Bewertung der Komplexität. Es ist nicht möglich eine einfache, lineare Beziehung zwischen Komplexität und Speicherbedarf anzugeben. Bemerkung: für die ausgewählten Funktionsbausteine war der Speicherbedarf für Daten jeweils 0, daher wird dieses Merkmal hier nicht verwendet.

¹ Kitchenham, B. ; Linkman, S.: Design Metrics in Practice, 31.05.89 (REQUEST/STL-bak/124/SI/QL-RP/00)

² Ein kommerzielles Softwarewerkzeug zur statischen und dynamischen Analyse manuell programmierter Software.

³ Da der Quellcode in allen betrachteten Anwendungen nicht verfügbar war, wird dieses Merkmal nicht weiter berücksichtigt.

Merkmal	Begründung
Laufzeit	Je höher der Laufzeit-Bedarf umso komplexer (zumindest umfangreicher) die Berechnungen. Da die TELEPERM XS-Software unterschiedliche Zustände kennt (Initialisierung, Parametrierung und Berechnung) werden die Laufzeiten danach unterschieden.
Fehlercodes	Die Zahl der zu unterscheidenden Fehlerfälle kann einen Hinweis auf die Komplexität des Funktionsbausteins geben.
Funktion	Es wird unterstellt, dass die Mächtigkeit der Beschreibung eines Funktionsbausteins Rückschlüsse auf dessen Komplexität erlaubt. Dabei wirken sich Bilder bzw. Tabellen mindernd auf die Fehleranfälligkeit eines Systemdesigns aus. Da es durchaus Unterschiede zwischen der Design-Unterlage (vom Designer und Gutachter verwendet) und dem Handbuch (vom End-Anwender genutzt) geben kann, wurden beide Dokumente ausgewertet.
Zahl der Meldungen	Wenn ein Funktionsbaustein ein Meldesignal besitzt, dann wirkt sich die Zahl der auszugebenden Meldungen auf die Komplexität des Funktionsbausteins aus.
Statusverarbeitung ⁴	Wirkt sich sowohl auf die Komplexität des Funktionsbausteins als auch auf die der Funktionspläne aus.
Interne Variablen	Eine höhere Zahl interner Variablen weist auf eine höhere Komplexität des Funktionsbausteins hin.
Abgeleitete Parameter	Wie Parameter.

Die Berechnung des Verflechtungsmaßes basiert auf der Verschaltung der Funktionsbausteine und Eingabe-Signale des Funktionsplans. In ISTec-A-1569 [1] wird beschrieben, dass in Funktionsplänen einzelne Funktionsbausteine oder auch ganze Sätze von Funktionsbausteinen zur Berechnung von mehr als nur einem Ausgangssignal verwendet werden. Diese Verflechtung $V(FP)$ hinsichtlich der Berechnung verschiedener Ausgangssignale ist kennzeichnend für die Komplexität eines Funktionsplans. Eine weitere wesentliche Rolle spielt die Verflechtung, die durch die Verwendung gemeinsamer Eingang-Signale gegeben ist.

Die Menge der Funktionsbausteine, die zur Berechnung eines Ausgangssignals S_{Ai} beitragen, bilden den Vorbereich $VB_{FB}(S_{Ai})$ des Ausgangssignals S_{Ai} .

Die Menge der Eingang-Signale, die zur Bildung des Ausgangssignals S_{Ai} verwendet werden bilden den Input $IN(S_{Ai})$ des Ausgangssignals S_{Ai} .

Die Verflechtung $V(FP)$ eines Funktionsplans ist definiert durch die Vorbereiche $VB_{FB}(S_{Ai})$ und der Inputs $IN(S_{Ai})$ aller Ausgangssignale eines Funktionsplans.

⁴ Mitführen von Informationen zur Qualität eines Signals (z.B. korrekt, fehlerhaft, Ersatzwert, ...)

$$V(FP) = \sum_i \frac{|VB_{FB}(S_{Ai})| + |IN(S_{Ai})|}{|BFP| + |SIN|} \quad (1)$$

Dabei bezeichnet:

S_{Ai}	die einzelnen Ausgabesignale des Funktionsplans FP
$VB_{FB}(S_{Ai})$	die Menge der Funktionsbausteine, die zur Berechnung des Ausgabesignals S_{Ai} beitragen
$IN(S_{Ai})$	die Menge der Eingang-Signale, die zur Berechnung des Ausgabe-Signals S_{Ai} beitragen
BFP	die Menge der Funktionsbausteine, aus denen der Funktionsplan FP besteht
SIN	die Menge der Eingang-Signale des Funktionsplans FP
$ I $	die Mächtigkeit (Anzahl der Elemente) einer Menge

Die Verflechtung ist im Wesentlichen ein Mittelwert, der die Vielfachheit widerspiegelt, mit der die einzelnen Funktionsbausteine und Eingangssignale für die Berechnung der verschiedenen Ausgabesignale verwendet werden.

Das Verflechtungsmaß ist unabhängig von der Komplexität der Funktionsbausteine. Die Komplexität der Funktionsbausteine wird durch den Komplexitätsvektor in die Betrachtung einbezogen.

Die Verflechtung $V(FP)$ ist modular in Bezug auf Prozessoren, Funktionspläne und Einzelsignale:

- infolge der systeminhärenten Zuordnung zwischen Prozessoren und Funktionsplänen ist das Maß $V(FP)$ auch in Bezug auf Prozessoren anwendbar
- durch den Begriff des Vorbereichs $VB_{FB}(S_{Ai})$ ist es auch in Bezug auf Einzelsignale anwendbar.

Ein weiteres Komplexitätsmerkmal, die Bausteinkomplexität lässt sich aus den bisher dargestellten generischen Merkmalen ermitteln. Die Ermittlung der Bausteinkomplexität erfolgt mittels folgender Gleichung:

$$FB_{Cplx} = \frac{in_ports + out_ports + params + memories}{\log(quantity)} \quad (2)$$

in_ports	Anzahl der Eingangsports
out_ports	Anzahl der Ausgangsports
params	Anzahl der Parameter
memories	Anzahl der internen Speicher
quantity	Verwendungshäufigkeit

Bei der Ermittlung der Bausteinkomplexität wird unterstellt, dass für alle generischen Merkmale eines Funktionsbausteins so viele Fehlermöglichkeiten angenommen werden, wie der Baustein Merkmale aufweist. Dabei handelt es sich um Fehler wie Fehlbeschriftung von Ein- und Ausgängen, Fehlinterpretation zwischengespeicherter Werte (interne Speicher), oder fehlerhaft belegte Parameter. Die so definierte Komplexität eines Funktionsbausteins wird durch die Berücksichtigung der Verwendungshäufigkeit ergänzt.

Aus den oben genannten Komplexitätsmaßen wurde ein Komplexitätsvektor gebildet. Der Komplexitätsvektor war Ausgangspunkt für die Herstellung der Korrelation zu Zuverlässigkeitsaussagen. Dabei wurden zur Gewinnung dieser Zuverlässigkeitsaussagen Bayesian Belief Netzwerke (BBNs) verwendet [1]. Sie gestatteten eine über mehrere Teilschritte hinweg durchzuführende und aufgrund klar definierter Transformationsregeln nachvollziehbare Transformation der objektiv gemessenen Komplexitätseigenschaften in eine Vorhersage der Zuverlässigkeit bzw. der Zuverlässigkeitsparameter eines digitalen Leittechnik-Systems.

3.2 Zuverlässigkeitsbewertung anhand von Komplexitätsmetriken

3.2.1 Ansätze aus der Literatur

In der Vergangenheit wurden insbesondere Komplexitätsmaße bezüglich ihrer Aussagekraft zu Softwarefehlern untersucht [10]. Diese Arbeit kommt zu dem Schluss, dass die zyklomatische Komplexität ein brauchbares Maß zur Abschätzung einer Fehlerwahrscheinlichkeit ist bzw. dass geeignete Kombinationen von Softwaremetriken für diesen Zweck herangezogen werden können [4]. Diese Aussagen sind tragfähig für manuell, von einem Programmierer erstellten Quellcode. Bei automatisch generiertem Code können Programmkonstrukte entstehen, die eine hohe (z.B. zyklomatische) Komplexität aufweisen, die aber zu einem einfachen und gut verständlichen Programmfluss führen und die für die Leittechnik von KKW's vorteilhaft sind. Deshalb ist die zyklomatische Komplexität für automatisch generierten Code im Allgemeinen bezüglich der Zuverlässigkeit dieser Software nicht aussagekräftig.

3.2.2 Eigene Arbeiten

Durch Software bedingtes Versagen beruht auf systematischen Fehlern in der Software, die sich entweder aus fehlerhaften oder unvollständigen Anforderungsspezifikationen oder aus Fehlern im Lebenszyklus ergeben. Software ändert sich nicht durch Alterungsprozesse. Durch Softwaremodifikationen wie z.B. Software-Updates aber auch einfache Parameteränderungen können jedoch Fehler in die Software hineingetragen werden.

Bei der Entwicklung und Pflege qualitativ hochwertiger Software, die durch die Anwendung standardisierter Qualitätssicherungsmaßnahmen gekennzeichnet ist, wird die Anzahl der Softwarefehler, auch der eventuell später eingetragenen Fehler, minimiert.

Nichtdestotrotz ist das Vorhandensein von Fehlern in der Software eine unabdingbare Voraussetzung für ein Softwareversagen. Diese Fehler können durch Ereignisse, die nur durch zeitliche Bedingungen oder durch die zu verarbeitenden Eingangsdatentrajektorie⁵ bedingt

⁵ Eine Trajektorie beinhaltet sowohl die Abfolge der Eingangsdaten als auch die dadurch bedingte Abfolge der internen Systemzustände.

sind, zur Wirkung gebracht werden. Man sagt auch, dass die Ereignisse die in der Software vorhandenen Fehler triggern.

Entscheidend für die Wahrscheinlichkeit, dass ein Softwareversagen auftritt, ist die Wahrscheinlichkeit für das Auftreten der versagensauslösenden Ereignisse, die den (oder einen der) in der Software vorhandenen Fehler triggern. Natürlich steigt diese Wahrscheinlichkeit auch mit der Anzahl der vorhandenen (und durch Qualitätssicherungsmaßnahmen und Tests nicht erkannten) Fehler an. Die qualitätssichernden Maßnahmen und Tests sind jedoch in ihrer Wirksamkeit stark von der Komplexität der Software abhängig. Während man für einfache Softwarebausteine durch Verifizierung und Validierung oftmals Fehlerfreiheit nachweisen kann, gelingt das für komplexere Systemstrukturen nicht.

Um der spezifischen Struktur eines Softwareentwurfs, der auf der Verschaltung von Funktionsbausteinen zu Funktionsplänen beruht, Rechnung tragen zu können, wurde ein Komplexitätsvektor für Funktionspläne definiert [1]. Im Bericht ISTec-A-1311 [2] wird beschrieben, dass - falls erforderlich - in übergeordneten, probabilistischen Zuverlässigkeits- bzw. Sicherheitsbewertungen eine Zusammenfassung von Komponenten des Komplexitätsvektors erfolgen kann:

Tabelle 3.2: Komponenten des Komplexitätsvektors nach [1]

ID	Bedeutung
K1:	Anzahl der Funktionsbausteine
K2:	Anzahl der Eingangssignale
K3:	Anzahl der Ausgabesignale
K4:	Anzahl der vorgelagerten Funktionspläne
K5:	Anzahl der nachfolgenden Funktionspläne
K6:	Komplexität der Verschaltung $V(FP)$
K7:	Anzahl änderbarer Parameter
K8:	Anzahl der internen Speicher
K9:	Komplexität der Funktionsbausteine des Funktionsplans

Die getrennte Darstellung der einzelnen Komplexitätsfaktoren in einem Komplexitätsvektor eignet sich als Grundlage für die Anwendung von Bayesian Belief Network Methoden (BBN-Methoden). Durch die Anwendung von BBN-Methoden kann der Komplexitätsvektor in einem wohldefinierten Verfahren - wie es BBN-Methoden darstellen - in eine Zuverlässigkeitsaussage transformiert werden. Der Komplexitätsvektor liefert dabei die Beobachtungen aufgrund derer eine Zuverlässigkeitsaussage abgeleitet wird. Es zeigte sich aber auch, dass die Struk-

turen der verwendeten Bayesian-Belief-Networks die Ergebnisse relativ stark beeinflussten [1].

Weiterhin ermöglicht die getrennte Darstellung durch die neun Komponenten eines Komplexitätsvektors eine detaillierte Identifizierung und Auswertung der verschiedenen Komplexitätsfaktoren.

4 ERWEITERUNGEN DER METHODIK

4.1 Erweiterung des Komplexitätsmaßes

Das im Bericht ISTec-A-1569 [1] beschriebene Komplexitätsmaß ist auf die Ausgangssignale eines Funktionsplans zugeschnitten und kann mit der Anzahl der Ausgangssignale normiert werden. Dieses Komplexitätsmaß wurde durch ein zweites Verflechtungsmaß $V_{mod}(FP)$ ergänzt. Dieses Maß spiegelt die interne Verknüpfung stärker wider. Es ist aber nicht mehr normierbar. Hierbei steht S_i für alle Signale (Ausgabesignale und internen Signale) des Funktionsplans. Somit stellt $VB_{FB}(S_i)$ die Menge der Funktionsbausteine, die zur Berechnung der Ausgabesignale bzw. der internen Signale beitragen, dar.

$$V_{mod}(FP) = \sum_i \frac{|VB_{FB}(S_i)| + |IN(S_i)|}{|BFP| + |SIN|} \quad (3)$$

Die Verflechtung wird hierbei auch über die Vorbereiche und die zur Berechnung herangezogenen Eingangssignale der internen Signale S_i berechnet.

Bildet man die Differenz

$$V_{intern}(FP) = V_{mod}(FP) - V(FP) \geq 0 \quad (4)$$

ergibt sich eine Maßzahl $V_{intern}(FP)$, die unabhängig von der Anzahl der Ausgangssignale ist. Diese Zahl kann als eine Charakteristik der inneren Komplexität des Funktionsplans verstanden werden. Diese Differenz ist immer größer oder gleich Null, da $V_{mod}(FP)$ immer größer oder gleich $V(FP)$ ist.

Im Bild 4.1 ist der Zusammenhang der Größen $V_{intern}(FP)$, $V_{mod}(FP)$ und $V(FP)$ für neun unterschiedliche Funktionspläne beispielhaft grafisch dargestellt.

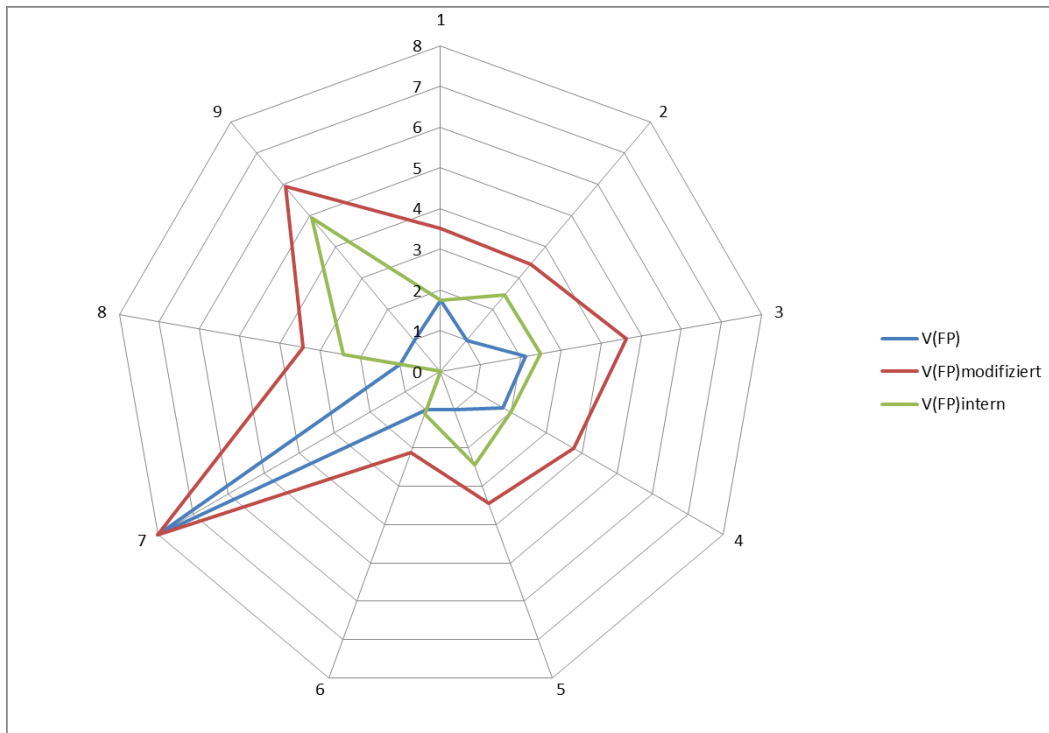


Bild 4.1: Zusammenhang von $V_{intern}(FP)$, $V_{mod}(FP)$ und $V(FP)$

Ein weiteres Komplexitätsmerkmal spiegelt sich in der Anzahl der Verbindungen eines Funktionsplans wider. Als Verbindung wird jede Verknüpfung eines Bausteins (Eingangssignal, Ausgangssignal, Funktionsbaustein) mit anderen Bausteinen aufgefasst. Je höher die Anzahl der Verbindungen in einem Funktionsplan ist, desto öfter werden die Signale verarbeitet und die Wahrscheinlichkeit sowie die Auswirkungen eines möglichen Fehlers steigen.

Durch die Ergänzung des Komplexitätsvektors aus 3.2.2 um die Komplexitätsmerkmale Anzahl der Verbindungen, Komplexität der Verschaltung $V_{mod}(FP)$ und Komplexität der Verschaltung $V_{intern}(FP)$ ergibt sich der erweiterte Komplexitätsvektor.

Tabelle 4.1: Komponenten $K_{e,x}$ des erweiterten Komplexitätsvektors

ID	Bedeutung
K_{e1} :	Anzahl der Funktionsbausteine
K_{e2} :	Anzahl der Eingangssignale
K_{e3} :	Anzahl der Ausgabesignale
K_{e4} :	Anzahl der Verbindungen
K_{e5} :	Anzahl der vorgelagerten Funktionspläne
K_{e6} :	Anzahl der nachfolgenden Funktionspläne

ID	Bedeutung
K _e 7:	Komplexität der Verschaltung V(FP)
K _e 8:	Komplexität der Verschaltung V _{mod} (FP)
K _e 9:	Komplexität der Verschaltung V _{intern} (FP)
K _e 10:	Anzahl änderbarer Parameter
K _e 11:	Anzahl der internen Speicher
K _e 12:	Komplexität der Funktionsbausteine des Funktionsplans

4.2 Modifikation des Komplexitätsvektors

4.2.1 Komplexitätsvektor für CPU-basierte Steuerungen

Aufgrund der im vorangegangenen Abschnitt eingeführten Erweiterungen des Komplexitätsmaßes und der zu ermittelnden Informationen wurde der Komplexitätsvektor gemäß 6.1.3 modifiziert. Im Zuge der Modifizierung wurden die Komponente K_e11 für die Anzahl der internen Speicher sowie die Komponente K_e12 für die Komplexität der Funktionsbausteine des Funktionsplans im modifizierten Komplexitätsvektor nicht berücksichtigt, da diese Informationen für die untersuchten Leittechnik-Plattformen nicht aus den Herstellerangaben (z.B. Datenblättern, Handbüchern usw.) ermittelt werden konnten (siehe Abschnitt 6.1.2 und 6.1.3).

Tabelle 4.2: Komponenten des modifizierten Komplexitätsvektors

ID	Bedeutung
K _m 1:	Anzahl der Funktionsbausteine
K _m 2:	Anzahl der Eingangssignale
K _m 3:	Anzahl der Ausgabesignale
K _m 4:	Anzahl der Verbindungen
K _m 5:	Anzahl der vorgelagerten Funktionspläne
K _m 6:	Anzahl der nachfolgenden Funktionspläne
K _m 7:	Komplexität der Verschaltung V(FP)
K _m 8:	Komplexität der Verschaltung V _{mod} (FP)

ID	Bedeutung
K _m 9:	Komplexität der Verschaltung V _{intern} (FP)
K _m 10:	Anzahl änderbarer Parameter

Dieser Komplexitätsvektor wurde bei der Projektbearbeitung für Funktionspläne CPU-basierter Steuerungen angewandt. Ein Funktionsplan kann einen zusammenhängenden Teilaspekt einer leittechnischen Funktion auf verschiedenen Hierarchiestufen bis hin zu einer kompletten leittechnischen Funktion in einem System repräsentieren. Ein Funktionsplan einer vollständigen Funktion besitzt naturgemäß keine vor- bzw. nachgelagerten Funktionspläne. In diesem Fall werden K_m5 und K_m6 Null gesetzt.

4.2.2 Komplexitätsvektor für programmierbare Logik

In Anlehnung an den erweiterten Komplexitätsvektor für Funktionspläne CPU-basierter Steuerungen (siehe 6.2.1.2) und den zu ermittelnden Informationen für FPGA-basierte Steuerungen ergibt sich der Komplexitätsvektor für FPGA-basierte Steuerungen gemäß 6.2.3.

Tabelle 4.3: Komponenten des Komplexitätsvektors für FPGA-basierte Steuerungen

ID	Bedeutung
K _f 1:	Anzahl der Basisblöcke
K _f 2:	Anzahl der Eingangssignale
K _f 3:	Anzahl der Ausgabesignale
K _f 4:	Anzahl der Verbindungen
K _f 5:	Anzahl der vorgelagerten Netzlisten
K _f 6:	Anzahl der nachfolgenden Netzlisten
K _f 7:	Komplexität der Verschaltung V(FP)
K _f 8:	Komplexität der Verschaltung V _{mod} (FP)
K _f 9:	Komplexität der Verschaltung V _{intern} (FP)

Dieser Komplexitätsvektor wurde bei der Projektbearbeitung für FPGA-basierte Steuerungen angewandt.

5 WERKZEUGE ZUR KOMPLEXITÄTSMESSUNG

5.1 Analyse des Werkzeugprototypen für TXS

Im Rahmen des Projekts „Komplexitätsmessung der Software digitaler Leittechnik-Systeme“ wurde ein Gerüst für die benötigte Werkzeugumgebung zur Ermittlung der Komplexitätswerte für eine Plattform entwickelt [1].

Dieses Gerüst setzte unmittelbar auf die SPACE-Datenbank auf [1]. Zudem war dieses Gerüst auf die Berechnung der Verflechtung $V(FP)$ beschränkt und beinhaltete keine Konsistenzprüfungen der Funktionspläne. Die Konsistenzprüfung war bereits in SPACE enthalten. Es waren deshalb Anpassungen in folgenden Aufgabenfeldern notwendig:

- a) Definition einer universellen Schnittstelle von Funktionsplänen zum Berechnungswerkzeug.
- b) Erweiterung der Berechnungsalgorithmen auf $V_{\text{intern}}(FP)$, $V_{\text{mod}}(FP)$.
- c) Ergänzung von Konsistenzprüfungen.
- d) Analyse und Funktionsplangenerierung aus EDIF-Netzlisten (Beschreibung der Verbindungen von E/A-Ports von Logikblöcken im Design)
- e) Auflösung funktionsplanübergreifender interner Verbindungen
- f) Anonymisierung von Funktionsplänen

5.2 Anpassung der Werkzeuge zur Komplexitätsmessung

5.2.1 Schnittstelle von Funktionsplan zum Berechnungswerkzeug

Für Speicherprogrammierbare Steuerungen werden zunächst die Verbindungen aller Bausteine aus der graphischen Darstellung eines Funktionsplans der CPU-basierten Steuerung manuell in einer Textdatei (Verbindungsliste) erfasst.

Die Verbindungen werden in einer Textdatei anhand von einfachen Regeln eingetragen, so dass das Werkzeug diese auswerten kann. Eingänge werden mit Anfangsbuchstaben „e“ aufgelistet, Ausgänge mit „a“, Funktionsbausteine mit „f“ und Konnektoren mit „c“. Die Verbindung eines Bausteins zu einem anderen wird mit „->“ dargestellt, weshalb keine dieser Zeichen in der ansonsten freien Namensgebung vorkommen darf. Die Textdatei kann z.B. folgende Form haben.

e1	->	fb
fa	->	fb
fb	->	fc
e2	->	ff
ff	->	fe
ff	->	fd
fc	->	fd
fc	->	as1
fd	->	as2
fe	->	as3

Bild 5.1: Eintrag der Verbindungen in einer Textdatei (Verbindungsliste) zu dem Funktionsplan in Bild 5.3.

Das Werkzeug wertet diese Textdatei anschließend aus und ermittelt die Anzahl der Vorbereiche und Funktionsbausteine, um das Verflechtungsmaß gemäß (1), (3) und (4) zu berechnen.

Für FPGA-basierte Steuerungen wird die zugehörige Netzliste meist durch eine Datei im EDIF (Electronic Design Interchange Format) beschrieben. Auf Grund der Standardisierung ist eine EDIF-Netzliste unabhängig von Hersteller und Typ des jeweiligen FPGA-Chips. Mit Hilfe dieser Netzlisten lassen sich alle Verbindungen zwischen den einzelnen Bausteinen ermitteln. Die Netzlisten enthalten keine Informationen zur internen Struktur der verbundenen Bausteine (siehe Ausschnitt EDIF-Netzliste). Die ausgewerteten EDIF-Netzlisten brechen die Verbindungen bis auf die Ebene der Basisblöcke herunter. Die Basisblöcke bestanden dabei aus LUTs und zustandsgesteuerten Flipflops. Beide Elemente besitzen eine vergleichbare Komplexität.

Die in einer EDIF-Netzliste beschriebenen Verbindungen können in eine Textdatei gemäß Bild 5.1 transformiert werden und die Verflechtungsmaße analog zu den Funktionsplänen einer CPU-basierten Steuerung ermittelt werden.

Im Gegensatz zum manuellen Eintragen der Verbindungen eines Funktionsplans kann die Netzliste einer FPGA-basierten Steuerung, welche durch ein standardisiertes EDIF Dateiformat beschrieben ist automatisiert ausgewertet werden, um die Verbindungen aller Blöcke aufzulisten.

Eine detaillierte Beschreibung der Schnittstelle ist im Anhang B angegeben.

5.2.2 Skriptprogramm zur Auswertung von EDIF-Netzlisten (analyzer)

Die Werkzeuge zur Entwicklung von FPGA-Anwendungen setzen standardmäßig auf das standardisierte Electronic Design Interchange Format (EDIF) auf. Im Folgenden ist ein Ausschnitt aus einer EDIF-Netzliste beispielhaft dargestellt.

```
(edif g_BP
(edifVersion 2 0 0)
(edifLevel 0)
(keywordMap (keywordLevel 0))
(status
(written
(timestamp 2015 8 24 13 29 58)
.
.
.
)
)
(library PA3
(edifLevel 0)
(technology (numberDefinition ))
(cell XNOR2 (cellType GENERIC)
(property dont_touch (string "false"))
(view prim (viewType NETLIST)
(interface
(port Y (direction OUTPUT)
(property function (string "!(A ^ B)"))
)
(port A (direction INPUT))
(port B (direction INPUT))
)
(property is_combinational (integer 1))
)
```



```

Cell Analyzer
© Quell- und Zieldatei auswählen Cell-Name Analysieren Einzelzelle Komplettanalyse Option Ende
Dateigröße: 516727
ausgewählte Zelle: AND_BOOL_2_0
ausgewählte Instanz:
Instanz: Zelle: AND_BOOL_2_0
4 Inputs
MOVE_BOOL_1_wire_248_OUT
MOVE_BOOL_1_wire_251_OUT
clk_c
rst_c
1 Outputs
AND_BOOL_2_wire_257_OUT1
4 Instance + zugehörige cellRef
process_1_R_o_2 NOR2A f
R_o DFN1C1 f
VCC_i VCC f
GND_i GND f
8 Net-Name + zugehörige Verbindung
R_o_2
(joined
    (portRef Y (instanceRef process_1_R_o_2))
    (portRef D (instanceRef R_o))
)
MOVE_BOOL_1_wire_248_OUT
(joined
    (portRef MOVE_BOOL_1_wire_248_OUT)
    (portRef A (instanceRef process_1_R_o_2))
)
MOVE_BOOL_1_wire_251_OUT
(joined
    (portRef MOVE_BOOL_1_wire_251_OUT)
    (portRef B (instanceRef process_1_R_o_2))
)
AND_BOOL_2_wire_257_OUT1
(joined
    (portRef Q (instanceRef R_o))
    (portRef AND_BOOL_2_wire_257_OUT1)
)
clk_c
(joined
    (portRef clk_c)
    (portRef CLK (instanceRef R_o))
)
rst_c
(joined
    (portRef rst_c)
    (portRef CLR (instanceRef R_o))
)
GND
(joined
    (portRef Y (instanceRef GND_i))
)
VCC
(joined
    (portRef Y (instanceRef VCC_i))
)
1 s
1 s

```

Bild 5.2: Benutzerschnittstelle des Skriptprogramms zur Auswertung von EDIF-Netzlisten

Der Programmteil zur Analyse von einzelnen Zellen wurde benötigt, um den Aufbau der EDIF-Netzlisten besser verstehen und das Skriptprogramm zur Extraktion der Verbindungen erarbeiten zu können. Die Extraktion der Verbindungen wird mit dem Menüpunkt „Komplettanalyse“ angestoßen.

Das Skriptprogramm nutzt Eigenschaften der Struktur der EDIF-Netzlisten der im Vorhaben verwendeten Beispiele aus, um die Bearbeitungszeiten kurz zu halten (wenige Minuten für „flache Designs“ und kleiner eine Stunde für „hierarchische Designs“). Es ist deshalb nicht auf beliebige EDIF-Netzlisten anwendbar, sondern muss gegebenenfalls angepasst werden.

5.2.3 Skriptprogramm zur Berechnung der Verflechtungen (vfb)

Aufgrund der notwendigen Schritte zur Nutzbarmachung des Gerüsts [1] für die Berechnungen für CPU- und FPGA-basierte Steuerungen wurde entschieden, einen vollständig neuen Prototyp des Berechnungswerkzeugs zu entwickeln. Die Beschreibung dieses Werkzeugs ist im Anhang B dargestellt.

Der ursprüngliche Algorithmus arbeitete alle Vorbereiche der Funktionsbausteine und der Ausgangssignale rekursiv ab. Dadurch entstanden erhebliche Rechenzeiten für komplexere Funktionspläne (im Bereich von Tagen).

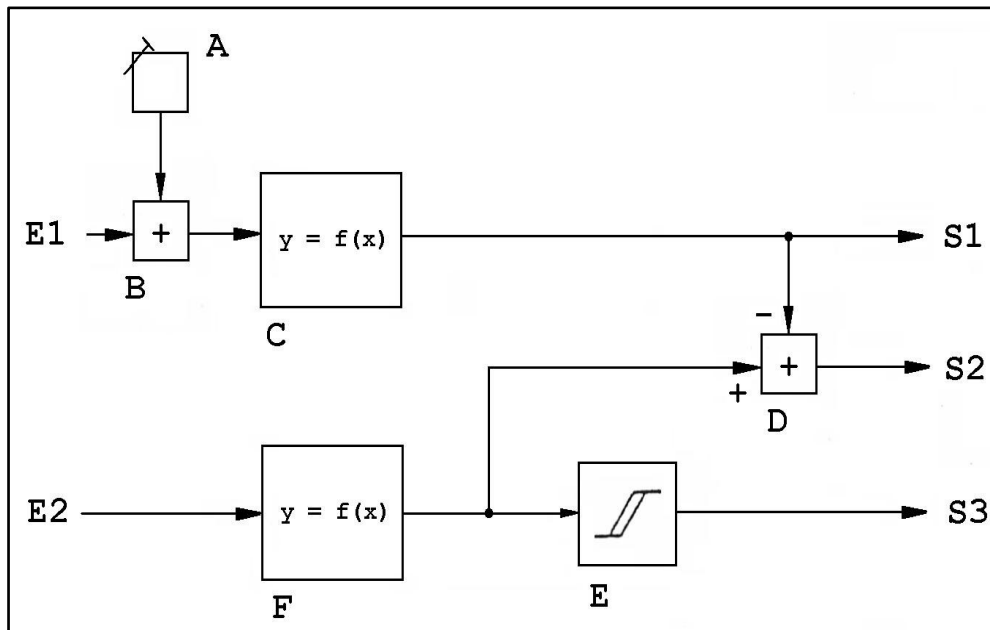


Bild 5.3: Funktionsplan [2] zur Veranschaulichung der Matrizendarstellung

Zur Optimierung der Berechnungszeit wurde der Algorithmus des Werkzeugs [1] dahingehend geändert, dass bereits ermittelte Vorbereiche nicht erneut berechnet werden. Dies konnte über eine Matrizendarstellung der Verbindungs- bzw. Knotenpunkte der einzelnen Bausteine umgesetzt werden.

Die Zeilen und Spalten der Matrix stellen die Ein- und Ausgänge, sowie die Funktionsbausteine des Funktionsplans (CPU-basierte Steuerung) bzw. der Netzliste (FPGA) dar. Für jeden Matrixeintrag (x,y) zur Zeile „x“ und Spalte „y“ bedeutet der Eintrag „1“, dass „x“ ein Vorbereich von „y“ ist und „0“, dass zwischen „x“ und „y“ keine Verbindung besteht. Damit lassen sich Schleifen leicht durch Prüfen des Eintrags (x,x) erkennen, da der Eintrag „1“ gleichbedeutend mit einer Verbindung eines Bausteins „x“ zu sich selbst ist. Die Mächtigkeit der Vorbereiche eines Bausteins wird über die Spaltensumme $\sum (VB)$ eines Bausteins ermittelt. Das Verflechtungsmaß lässt sich anschließend gemäß 3.1.3 bzw. 3.1.4 anhand der Mächtigkeit der Vorbereiche und der Anzahl der Bausteine ermitteln.

	E1	E2	A	B	C	D	E	F	S1	S2	S3
E1	0	0	0	1	1	1	0	0	1	1	0
E2	0	0	0	0	0	1	1	1	0	1	1
A	0	0	0	1	1	1	0	0	1	1	0
B	0	0	0	0	1	1	0	0	1	1	0
C	0	0	0	0	0	1	0	0	1	1	0
D	0	0	0	0	0	0	0	0	0	1	0
E	0	0	0	0	0	0	0	0	0	0	1
F	0	0	0	0	0	1	1	0	0	1	1
S1	0	0	0	0	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0	0	0
S3	0	0	0	0	0	0	0	0	0	0	0
Σ (VB)	0	0	0	2	3	6	2	1	4	7	3

Bild 5.4: Matrixdarstellung zum Funktionsplan aus Abbildung 5.3

Wird ein bereits ermittelter Vorbereich eines Bausteins abgefragt, so wird dieser nicht erneut berechnet, sondern die Einträge der Spalte, welche diesen Baustein in der Matrix repräsentiert, übernommen. Mit der Übernahme der Spalteneinträge eines Bausteins ist auch der Vorbereich des Bausteins vollständig berücksichtigt. Diese Vorgehensweise ermöglicht, wie bereits erwähnt, eine deutliche Verkürzung der Berechnungszeit und konnte am Beispiel der FPGA Netzliste (siehe Abschnitt 7.2) von über 24 Stunden auf ungefähr 20 Minuten gesenkt werden.

Mit dem Tool, das für die Berechnung der Verflechtungsmaße verwendet wird, lässt sich neben der Identifikation von Schleifen auch prüfen, ob der Funktionsplan bzw. die Netzliste zusammenhängend ist. Ein Funktionsplan ist zusammenhängend, wenn er sich nicht in zwei oder mehr separate Funktionspläne aufteilen lässt. Die Prüfung des Zusammenhangs läuft hierbei ebenfalls automatisiert ab. Für die Validierung des Tools (siehe Anhang C) wurden zusätzliche Fehler in Funktionspläne eingebaut, wie z.B. isolierte Knoten (Bausteine ohne Verbindung), fehlende Ausgänge oder fehlerhafte Verwendung von Funktionsbausteinen, um die Fehlererkennung zu validieren. Der Funktionsbaustein A in Bild 5.3 beispielsweise stellt einen Funktionsbaustein dar, der wie ein Eingang agiert, was zu einer Warnmeldung führt.

Das Werkzeug ist in der Lage die Verflechtungsmaße von nahezu beliebig umfangreichen Funktionsplänen und Netzlisten zu bestimmen und diese auf Zusammenhang, Schleifen und Fehler zu prüfen. Im Batchbetrieb können von dem Werkzeug auch mehrere Funktionspläne und Netzlisten automatisiert bearbeitet werden, was z.B. die Auswertung einer kompletten Anwendung oder Funktion ermöglicht.

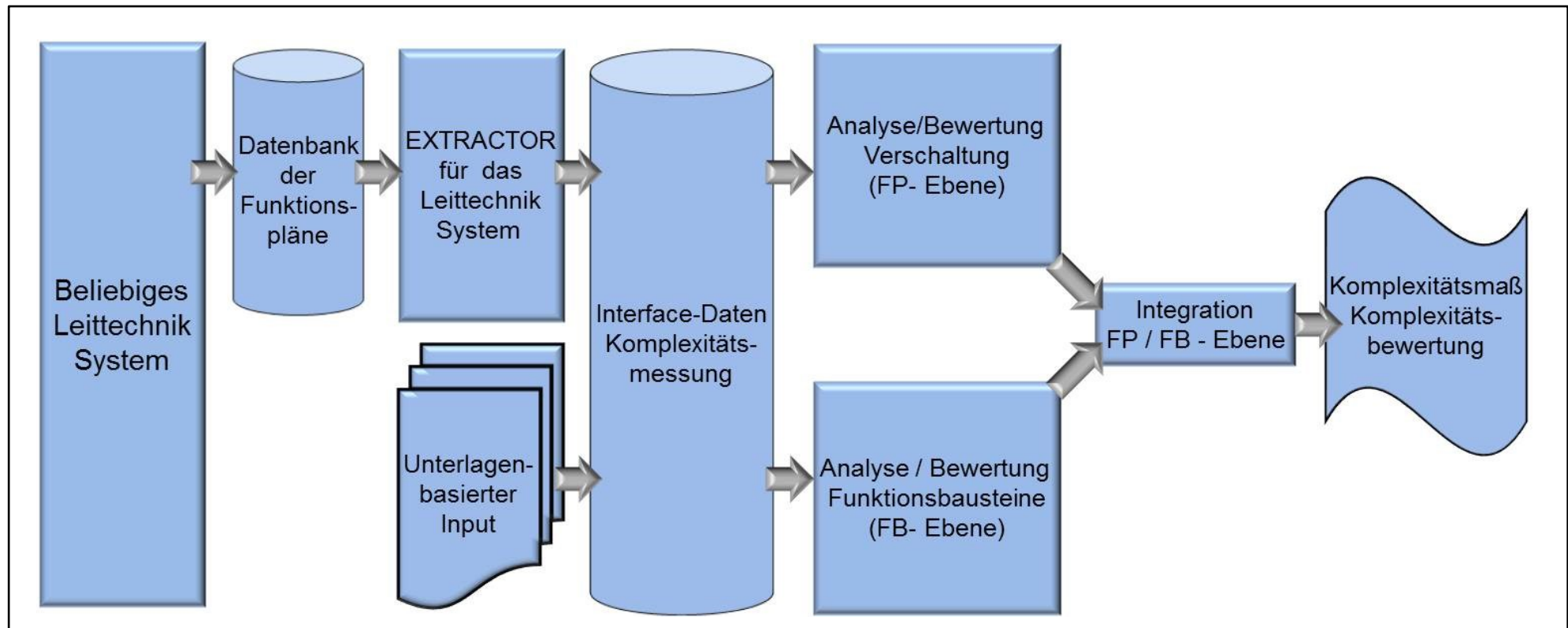


Bild 5.5: Gesamtstruktur der automatisierten Komplexitätsmessung [1]

Eine detaillierte Beschreibung des Werkzeugs zur Berechnung der Verflechtungen, einschließlich der Optionen zur Prüfung der Funktionspläne ist im Anhang B angegeben.

5.2.4 Skriptprogramm zur Auflösung von internen Verbindungen in Funktionsplänen (connector)

Die Werkzeuge zur grafischen Spezifikation von Funktionsplänen stellen diese oftmals auf mehreren Bildschirmseiten dar, die auch im Fall eines Ausdrucks auf mehrere Seiten verteilt sind. Dabei erlauben sie die Verwendung interner Verbindungen zwischen den Seiten. Diese müssen nicht zwangsläufig 1-zu-1 Verbindungen sein. Es sind auch Verbindungen von einer Menge von Funktionsbausteinen über einen solchen Verbindungspunkt zu einer zweiten Menge von Funktionsbausteinen möglich. Derartige Verbindungspunkte (Konnektoren) werden in einem Funktionsplan mit dem Zeichen „c“ markiert und wie Funktionsbausteine verwendet (Bsp. „fb1 -> c34“, „c34 -> fb3“).

Das Skriptprogramm zur Auflösung dieser Verbindung erzeugt aus einer Funktionsplanbeschreibung mit Konnektoren, eine solche, in der alle Verbindungen entsprechend aufgelöst sind (Bsp. „fb1 -> c34“, „c34 -> fb3“ wird zu „fb1 -> fb3“).

Eingangsdatei	Ausgangsdatei
eE1 -> fB	eE1 -> fB
fA -> fB	fA -> fB
fB -> fC	fB -> fC
fC -> aS1	fC -> aS1
fC -> c1	eE1 -> fF
eE1 -> fF	fF -> fE
fF -> fE	fD -> aS2
fF -> c1	fE -> aS3
c1 -> fD	fC -> fD
fD -> aS2	fF -> fD
fE -> aS3	

Bild 5.6: Auflösung von Verbindungen

Das Skriptprogramm öffnet ein Konsolfenster und fordert über Eingabedialoge die Namen der Eingabe- und der Ausgabedatei an. Der Abschluss der Bearbeitung wird im Konsolfenster angezeigt. Das Konsolfenster wird durch einen Mausklick in das Konsolfenster oder eine beliebige Tastatureingabe bei aktivem Konsolfenster geschlossen.

In übergeordneten Skripten oder in Batch-Dateien kann das Skriptprogramm mit der Angabe des Eingabe- und des Ausgabedateinamens als Parameter aufgerufen werden. In diesem Fall wird zwar das Konsolfenster geöffnet, es werden aber die Eingabedialoge nicht aufgerufen. Das Konsolfenster wird immer automatisch geschlossen. Falls die Ausgabedatei bereits existiert, wird sie ohne Rückfrage durch die neue Ausgabe überschrieben. Ein fehlerhafter Eingabedateiname führt zum Aufruf der Eingabedialoge. Ein fehlerhafter Ausgabedateiname führt zum Programmabbruch.

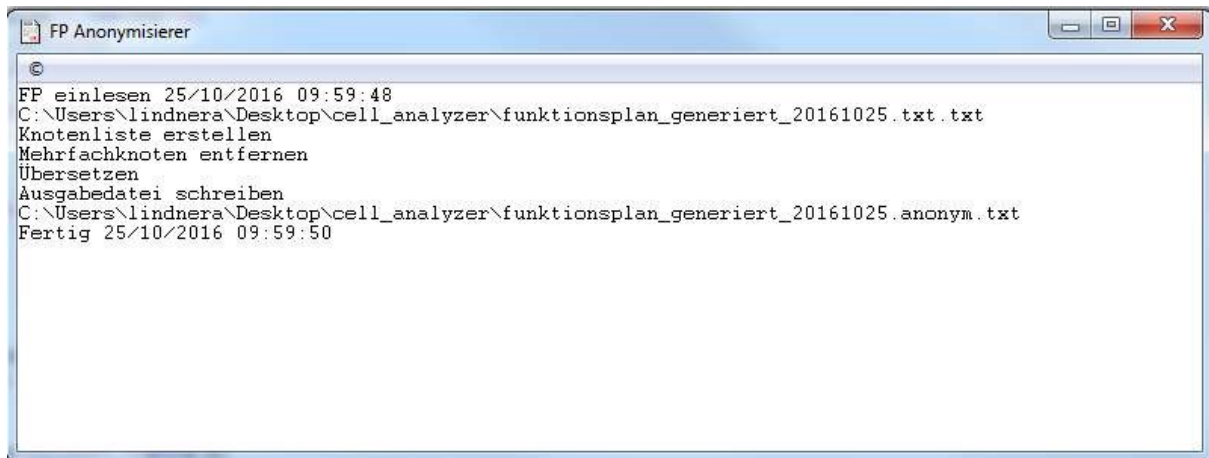
5.2.5 Skriptprogramm zur Anonymisierung von Funktionsplänen (anonymizer)

Um Funktionspläne anonymisieren zu können wurde ein Skriptprogramm implementiert, das alle Eingangssignale, Ausgangssignale und Funktionsbausteine nach dem Schema

- e lfd. Nr. (Bsp. e_15) für Eingangssignale

- a lfd. Nr. (Bsp. a_3) für Ausgangssignale
- f lfd. Nr. (Bsp. f_125) für Funktionsbausteine

umbenennt. Ein Beispiellauf ist in Bild 5.7 dargestellt.



```
FP Anonymisierer
©
FP einlesen 25/10/2016 09:59:48
C:\Users\lindnera\Desktop\cell_analyzer\funktionsplan_generiert_20161025.txt.txt
Knotenliste erstellen
Mehrfachknoten entfernen
Übersetzen
Ausgabedatei schreiben
C:\Users\lindnera\Desktop\cell_analyzer\funktionsplan_generiert_20161025.anonym.txt
Fertig 25/10/2016 09:59:50
```

Bild 5.7: Beispiellauf 5.2.5 Skriptprogramm zur Anonymisierung von Funktionsplänen

5.3 Validierung der Werkzeuge zur Komplexitätsmessung

5.3.1 Skriptprogramm zur Auswertung von EDIF-Netzlisten (analyzer)

Zur Validierung des Skriptprogramms wurde anhand der grafischen Darstellung eines FPGA-Designs manuell eine Verbindungsliste gemäß Abschnitt 5.2.1 erstellt. Diese Netzliste enthielt 75 Eingangssignale, 23 Ausgangssignale, 513 Basisblöcke und 1224 Verbindungen.

Der aus der zugehörigen EDIF-Netzliste mittels des Skriptprogramms analyzer generierte Verbindungsliste wies 10 Abweichungen zu der manuell erstellten Verbindungsliste auf. Die Analyse ergab:

- In der manuell erstellten Verbindungsliste fehlte eine Verbindung. Diese Abweichung konnte nachvollzogen und korrigiert werden.
- Bei 9 Verbindungen war die Übersetzung von externen und internen Signalbezeichnungen eines Bausteins in der manuell erstellten Verbindungsliste unterblieben. Diese Abweichungen konnte nachvollzogen und korrigiert werden.

Nach Korrektur der Fehler waren beide Verbindungslisten (bis auf die unterschiedlichen Namen der Eingangs- und Ausgangssignale sowie der Basisblöcke) identisch.

Die Validierung wurde nach jeder Änderung des Skriptprogramms wiederholt, wobei die Wiederholungen gegen die als fehlerfrei ermittelte Verbindungsliste geprüft wurden.

5.3.2 Skriptprogramm zur Berechnung der Verflechtungen (vfb)

Zur Validierung der Werkzeuge zur Komplexitätsmessung wurde eine Testsuite von Funktionsplänen erstellt, die folgende Eigenschaften aufweist:

- Es wird die Korrektheit der Berechnung geprüft.

- Es wird geprüft, dass die Reihenfolge der im Funktionsplan angegebenen Verbindungen keinen Einfluss auf das Ergebnis hat.
- Es werden alle Fehlerbedingungen mit Ausnahme der Fehlerbedingung „Zu viele Knoten im Funktionsplan“ geprüft. Die Fehlerbedingung „Zu viele Knoten im Funktionsplan“ wird aus Zeitgründen mit einer Softwareversion mit stark begrenzter Knotenzahl geprüft.

Die Testsuite ist im Anhang C detailliert beschrieben. Die Validierung wurde nach jeder Änderung des Skriptprogramms erneut durchgeführt. Im Anhang sind die Ergebnisse der Validierung der aktuellen Version des Skriptprogramms dargestellt.

5.3.3 Skriptprogramm zur Auflösung von internen Verbindungen in Funktionsplänen (connector)

Das Skriptprogramm wurde mit einer Datei einer realen Anwendung validiert. Ein Ausschnitt der Eingangsdatei ist in Bild 5.8 der Ausgangsdatei gegenübergestellt.

Eingangsdatei	Ausgangsdatei
eAmanuellerGebabs -> fAnegl	eAmanuellerGebabs -> fAnegl
c15.1 -> fA>11	fAnegl -> fA>11
fAnegl -> fA>11	fA>11 -> aAGebäudeabschluss
fA>11 -> aAGebäudeabschluss	eAHandRESA -> fAneg2
eAHandRESA -> fAneg2	fAneg2 -> fA>12
fAneg2 -> fA>12	eARESAreset -> fAff
eARESAreset -> fAff	fA>12 -> fAff
fA>12 -> fAff	fA>12 -> fAPuls
fA>12 -> fAPuls	fAff -> aARESA
fAff -> aARESA	fAPuls -> aANotkühlung
fAPuls -> aANotkühlung	f2RSkorNF -> f2RSggw
c15.1 -> fA>12	eNF-LB -> f2RSkorNF
c2.4 -> fA>12	eDL-PS -> f2RSkorNF
c3.2 -> fA>12	eNF-LB -> f3RSkorNF
c3.3 -> fA>12	eDL-PS -> f3RSkorNF
c4 -> fA>12	f3RSkorNF -> f3RSggw
c5 -> fA>12	eNF-LB -> f4RSggw
c6.2 -> fA>12	eNF-LB -> f5RSggw
c7.1 -> fA>12	f6RSkorNF -> f6RSggw

Bild 5.8: Validierung des Skriptprogramms zur Auflösung von internen Verbindungen in Funktionsplänen

Anhand von Bild 5.8 ist die Arbeitsweise des Skriptprogramms ersichtlich. Für die Validierung wurde das Beispiel vollständig manuell nachvollzogen.

5.3.4 Skriptprogramm zur Anonymisierung von Funktionsplänen (anonymizer)

Das Skriptprogramm wurde mit der Ausgangsdatei des in Abschnitt 5.3.3 dargestellten Beispiels validiert. Der in Bild 5.8 gezeigte Ausschnitt der Ausgangsdatei ist in Bild 5.9 den anonymisierten Daten gegenübergestellt.

Eingangsdatei	Ausgangsdatei
eAmanuellerGebabs -> fAnegl	e_9 -> f_64
fAnegl -> fA>11	f_64 -> f_61
fA>11 -> aAGebäudeabschluss	f_61 -> a_2
eAHandRESA -> fAneg2	e_8 -> f_65
fAneg2 -> fA>12	f_65 -> f_62
eARESAreset -> fAff	e_10 -> f_63
fA>12 -> fAff	f_62 -> f_63
fA>12 -> fAPuls	f_62 -> f_66
fAff -> aARESA	f_63 -> a_4
fAPuls -> aANotkühlung	f_66 -> a_3
f2RSkorNF -> f2RSggw	f_46 -> f_45
eNF-LB -> f2RSkorNF	e_22 -> f_46
eDL-PS -> f2RSkorNF	e_16 -> f_46
eNF-LB -> f3RSkorNF	e_22 -> f_50
eDL-PS -> f3RSkorNF	e_16 -> f_50
f3RSkorNF -> f3RSggw	f_50 -> f_49
eNF-LB -> f4RSggw	e_22 -> f_51
eNF-LB -> f5RSggw	e_22 -> f_52
f6RSkorNF -> f6RSggw	f_56 -> f_55

Bild 5.9: Validierung des Skriptprogramms zur Anonymisierung von Funktionsplänen

Anhand von Bild 5.9 ist die Arbeitsweise des Skriptprogramms ersichtlich. Für die Validierung wurde das Beispiel vollständig manuell nachvollzogen.

6 ERMITTLUNG DER KOMPLEXITÄTSCHARAKTERISTIK DIGITALER LEIT-TECHNIKSYSTEME DER KERNTECHNIK

6.1 CPU-basierte Steuerungen

6.1.1 Betrachtungsebenen

Wie bereits im Abschnitt 3 und speziell in [4] ausgeführt wird, sind Software-Komplexitätsmerkmale geeignet, die Software bezüglich ihrer zu erwartenden Zuverlässigkeit zu beurteilen. Die Software für digitale Leittechnikssysteme wird entsprechend dem Stand von Wissenschaft und Technik mit grafischen Spezifikationswerkzeugen erstellt, mit deren Hilfe vorgefertigte Funktionsbausteine zu Funktionsplänen verschaltet werden. Diese Funktionspläne werden danach in sequenzielle Instruktionen einer Programmiersprache (z.B. C) übersetzt. Daraus ergeben sich für CPU-basierte Steuerungen zwei unterschiedliche Betrachtungsebenen:

- Funktionsbausteine, die leittechnische Elementarfunktionen implementieren
- Funktionspläne als Verschaltung der leittechnischen Elementarfunktionen

Die Elemente der Betrachtungsebenen besitzen jeweils eigenständige Komplexitätsmerkmale, die gemäß der im Bericht ISTec-A-1569 [1] beschriebenen Methodik in Komplexitätsmatrizen und einem Komplexitätsvektor aggregiert werden.

6.1.1.1 Funktionsbausteine

Um die Komplexitätsmerkmale von Funktionsbausteinen einheitlich darzustellen wurden im Bericht ISTec-A-1569 [1] Tabellen erstellt, die als Komplexitätsmatrizen bezeichnet sind. Im Folgenden sind die Templates der Komplexitätsmatrizen in tabellarischer Form dargestellt, die eingeführt wurden, um die Komplexitätsmerkmale von TELEPERM XS aufzulisten. Daran anschließend werden diese Komplexitätsmatrizen mit den Daten der in diesem Vorhaben verfügbaren Beispiele veranschaulicht. Eine Erklärung der in der Komplexitätsmatrix dargestellten Komplexitätsmerkmale ist unter 3.1.3 zu finden.

Tabelle 6.1: Komplexitätsmatrix der Funktionsbausteine Teil 1 (Signale, Parameter)

Betrachtungsebene	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
Funktionsbaustein										

Tabelle 6.2: Komplexitätsmatrix der Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)

Betrachtungsebene	interne Variable	Zustandsspeicher	Speicherbedarf		Laufzeit [μ s]			Fehler- codes	Funktion	
			Code (Bytes)	Stack (Bytes)	Initialisie- ren	Parametrieren	Berechnen		Text	Bild, Tabelle
Funktionsbaustein										

Zusätzlich zu den Komplexitätsmatrizen wurde im Bericht ISTec-A-1569 [1] eine Tabelle erarbeitet, welche die Parameter der Funktionsbausteine auflistet, die für die Ermittlung der Bausteinkomplexität benötigt werden. Die Bausteinkomplexität K9 ist wiederum eine der Komponenten des Komplexitätsvektors (siehe 3.2.2).

Tabelle 6.3: Ermittlung der Bausteinkomplexität K9

Betrachtungsebene	in_ports	out_ports	params	memories	quantity
Funktionsbaustein					

6.1.1.2 Funktionspläne

Um die Komplexität der Funktionspläne darzustellen, wird der in 3.1.3 eingeführte Komplexitätsvektor, gemäß Abschnitt 4.2 erweitert. Der erweiterte Komplexitätsvektor stellt sich wie folgt dar.

Tabelle 6.4: Komponenten des erweiterten Komplexitätsvektors

ID	Bedeutung
K _e 1:	Anzahl der Funktionsbausteine
K _e 2:	Anzahl der Eingangssignale
K _e 3:	Anzahl der Ausgabesignale
K _e 4:	Anzahl der Verbindungen
K _e 5:	Anzahl der vorgelagerten Funktionspläne
K _e 6:	Anzahl der nachfolgenden Funktionspläne
K _e 7:	Komplexität der Verschaltung V(FP)
K _e 8:	Komplexität der Verschaltung V _{mod} (FP)
K _e 9:	Komplexität der Verschaltung V _{intern} (FP)
K _e 10:	Anzahl änderbarer Parameter
K _e 11:	Anzahl der internen Speicher
K _e 12:	Komplexität der Funktionsbausteine des Funktionsplans

6.1.2 Beispiele

Die im Abschnitt 6.1.1.1 dargestellten Komplexitätsmatrizen und der im Abschnitt 6.1.1.2 eingeführte erweiterte Komplexitätsvektor werden für beispielhafte digitale Leittechnikssysteme verschiedener Hersteller ausgewertet. Die folgende Auswertung erfolgt dabei anhand der Komplexitätsmerkmale, die für das jeweilige Leittechnikssystem eines bestimmten Herstellers ermittelbar sind.

Das digitale Leittechnikssystem L1 von Hersteller 1 und das digitale Leittechnikssystem L2 von Hersteller 2 sind für Anwendungen aus der Kerntechnik ausgelegt. Um einen Vergleich der ermittelbaren Daten mit nicht nuklearen Anwendungen zu gewährleisten, wurde außerdem ein Leittechnikssystem L3 von Hersteller 3 ausgewählt, von dem uns nach jetzigem Kenntnisstand keine Anwendung in der Kerntechnik bekannt ist.

6.1.2.1 Hersteller 1

Tabelle 6.5: Komplexitätsmatrix der L1 Funktionsbausteine Teil 1 (Signale, Parameter)

Betrachtungsebene	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
Funktionsbaustein	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒

Tabelle 6.6: Komplexitätsmatrix der L1 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)

	interne Variable	Zustandsspeicher	Speicherbedarf ⁶		Laufzeit [µs]			Fehler-codes	Funktion	
			Code (Worte)	Parameter (Worte)	Initialisieren	Parametrieren	Berechnen ⁷		Text	Bild, Tabelle
ermittelbar	☒	☐	☒	☒	☐	☐	☒	☒	☒	☒

Im Gegensatz zu dem im vorangegangenen Forschungsvorhaben untersuchten TELEPERM XS (ISTec-A-1569 [1]) ist der Speicherbedarf der Funktionsbausteine nicht in Bytes sondern in Worten angegeben. Aus der Angabe Worte lässt sich jedoch der Speicherbedarf in Bytes errechnen.

Tabelle 6.7: Ermittlung der Bausteinkomplexität für L1 Funktionsbausteine

	in_ports	out_ports	params	memories	quantity
ermittelbar	☒	☒	☒	☒	☒

Die Anzahl der Speicher (memories) lässt sich für die CPU-basierte Steuerung von Hersteller 1 nicht vollständig aus der softwarebasierten Funktionsplandarstellung ziehen, da bei diesem Hersteller die Speicher in der Hardwarebeschreibung hinterlegt sind. Da in diesem Fall die Verwendungshäufigkeit einzelner Funktionsbausteine (quantity) gering ist, ist diese Größe nicht belastbar.

⁶ Die Grundeinheiten (Byte, Worte) wurden angepasst.

⁷ In den zur Auswertung der Komplexitätsmerkmale herangezogenen Dokumenten sind nur Gesamtlaufzeiten angegeben. Diese sind in der Spalte „Berechnen“ eingetragen.

Anhand der verfügbaren Komplexitätsmerkmale sind für die CPU-basierte Steuerung von Hersteller 1 folgende Komponenten des erweiterten Komplexitätsvektors (siehe 3.2.2) ermittelbar.

Tabelle 6.8: Ermittlung der Komponenten des erweiterten Komplexitätsvektors für einen L1 Funktionsplan

ID	Bedeutung	ermittelbar
K _e 1:	Anzahl der Funktionsbausteine	☒
K _e 2:	Anzahl der Eingangssignale	☒
K _e 3:	Anzahl der Ausgabesignale	☒
K _e 4:	Anzahl der Verbindungen	☒
K _e 5:	Anzahl der vorgelagerten Funktionspläne	☒
K _e 6:	Anzahl der nachfolgenden Funktionspläne	☒
K _e 7:	Komplexität der Verschaltung V(FP)	☒
K _e 8:	Komplexität der Verschaltung V _{mod} (FP)	☒
K _e 9:	Komplexität der Verschaltung V _{intern} (FP)	☒
K _e 10:	Anzahl änderbarer Parameter	☒
K _e 11:	Anzahl der internen Speicher	☒
K _e 12:	Komplexität der Funktionsbausteine des Funktionsplans	☒

6.1.2.2 Hersteller 2

Tabelle 6.9: Komplexitätsmatrix der L2 Funktionsbausteine Teil 1 (Signale, Parameter)

Betrachtungsebene	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
Funktionsbaustein	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Tabelle 6.10: Komplexitätsmatrix der L2 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)

	interne Variable	Zustandsspeicher	Speicherbedarf		Laufzeit [µs]			Fehler- codes	Funktion	
			Code (Bytes)	Stack (Bytes)	Initialisie- ren	Parametrieren	Berechnen		Text	Bild, Tabelle
ermittelbar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Tabelle 6.11: Ermittlung der Bausteinkomplexität für L2 Funktionsbausteine

	in_ports	out_ports	params	memories	quantity
ermittelbar	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Anhand der verfügbaren Komplexitätsmerkmale sind für die CPU-basierte Steuerung von Hersteller 2 folgende Komponenten des erweiterten Komplexitätsvektors (siehe 3.2.2) ermittelbar.

Tabelle 6.12: Ermittlung der Komponenten des erweiterten Komplexitätsvektors für einen L2 Funktionsplan

ID	Bedeutung	ermittelbar
K _e 1:	Anzahl der Funktionsbausteine	<input checked="" type="checkbox"/>
K _e 2:	Anzahl der Eingangssignale	<input checked="" type="checkbox"/>
K _e 3:	Anzahl der Ausgabesignale	<input checked="" type="checkbox"/>
K _e 4:	Anzahl der Verbindungen	<input checked="" type="checkbox"/>
K _e 5:	Anzahl der vorgelagerten Funktionspläne	<input type="checkbox"/>
K _e 6:	Anzahl der nachfolgenden Funktionspläne	<input type="checkbox"/>
K _e 7:	Komplexität der Verschaltung V(FP)	<input checked="" type="checkbox"/>
K _e 8:	Komplexität der Verschaltung V _{mod} (FP)	<input checked="" type="checkbox"/>

ID	Bedeutung	ermittelbar
K _e 9:	Komplexität der Verschaltung V _{intern} (FP)	<input checked="" type="checkbox"/>
K _e 10:	Anzahl änderbarer Parameter	<input type="checkbox"/>
K _e 11:	Anzahl der internen Speicher	<input type="checkbox"/>
K _e 12:	Komplexität der Funktionsbausteine des Funktionsplans	<input type="checkbox"/>

6.1.2.3 Hersteller 3

Tabelle 6.13: Komplexitätsmatrix der L3 Funktionsbausteine Teil 1 (Signale, Parameter)

Betrachtungsebene	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
Funktionsbaustein	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Tabelle 6.14: Komplexitätsmatrix der L3 Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)

	interne Variable	Zustandsspeicher	Speicherbedarf		Laufzeit [μ s]			Fehler-codes	Funktion	
			Code (Bytes)	Stack (Bytes)	Initialisieren	Parametrieren	Berechnen		Text	Bild, Tabelle
ermittelbar	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Tabelle 6.15: Ermittlung der Bausteinkomplexität für L3 Funktionsbausteine

	in_ports	out_ports	params	memories	quantity
ermittelbar	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Anhand der verfügbaren Komplexitätsmerkmale sind für die CPU-basierte Steuerung von Hersteller 3 folgende Komponenten des erweiterten Komplexitätsvektors (siehe 3.2.2) ermittelbar.

Tabelle 6.16: Ermittlung der Komponenten des erweiterten Komplexitätsvektors für einen L3 Funktionsplan

ID	Bedeutung	ermittelbar
K _e 1:	Anzahl der Funktionsbausteine	<input checked="" type="checkbox"/>
K _e 2:	Anzahl der Eingangssignale	<input checked="" type="checkbox"/>

ID	Bedeutung	ermittelbar
K _e 3:	Anzahl der Ausgabesignale	<input checked="" type="checkbox"/>
K _e 4:	Anzahl der Verbindungen	<input checked="" type="checkbox"/>
K _e 5:	Anzahl der vorgelagerten Funktionspläne	<input type="checkbox"/>
K _e 6:	Anzahl der nachfolgenden Funktionspläne	<input type="checkbox"/>
K _e 7:	Komplexität der Verschaltung $V(\text{FP})$	<input checked="" type="checkbox"/>
K _e 8:	Komplexität der Verschaltung $V_{\text{mod}}(\text{FP})$	<input checked="" type="checkbox"/>
K _e 9:	Komplexität der Verschaltung $V_{\text{intern}}(\text{FP})$	<input checked="" type="checkbox"/>
K _e 10:	Anzahl änderbarer Parameter	<input checked="" type="checkbox"/>
K11:	Anzahl der internen Speicher	<input type="checkbox"/>
K12:	Komplexität der Funktionsbausteine des Funktionsplans	<input type="checkbox"/>

6.1.3 Schlussfolgerung für Modifikation der Komplexitätscharakteristika

Die Komplexitätsmatrizen und der Komplexitätsvektor in ISTec-A-1569 [1] wurden auf Grundlage der Daten zu TELEPERM XS eingeführt. Bezüglich der Auswertung der Komplexitätsmerkmale standen für TELEPERM XS projektspezifisch sehr umfassende und detaillierte Daten zur Verfügung. Im Zuge der Erweiterung der Methodik wurden weitere Komplexitätsmerkmale in die Betrachtung einbezogen und es ergibt sich der erweiterte Komplexitätsvektor gemäß 4.1. Die Prüfung der Anwendbarkeit der Methodik auf andere digitale Leittechnikssysteme zeigt, dass eine Modifizierung der Komplexitätsmatrizen und des Komplexitätsvektors nötig ist, da die Komplexitätsmerkmale zu anderen digitalen Leittechnikssystemen verglichen mit TELEPERM XS im Allgemeinen nicht so umfassend und detailliert ermittelbar sind.

Tabelle 6.17: Modifizierte Komplexitätsmatrix der Funktionsbausteine Teil 1 (Signale, Parameter)

Betrachtungsebene	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
Funktionsbaustein										

Bezüglich der Merkmale Interne Variablen, Zustandsspeicher und Ressourcenbedarf (Teil 2 der Komplexitätsmatrix) konnte bei keiner der zur Verfügung stehenden Steuerungen die Anzahl der Zustandsspeicher ermittelt werden. Diese Kenngröße wird somit in der modifizierten Komplexitätsberechnung nicht berücksichtigt. Darüber hinaus geht aus den Dokumenten lediglich die Gesamtlaufzeit der Funktionsbausteine hervor. Diese wird nicht weiter in Initialisierungs-, Parametrierungs-, oder Berechnungszeit unterteilt. Aus diesem Grund stellt die Laufzeit in der modifizierten Tabelle die Gesamtlaufzeit des Funktionsbausteins dar.

Tabelle 6.18: Modifizierte Komplexitätsmatrix der Funktionsbausteine Teil 2 (Interne Variable, Zustandsspeicher, Ressourcenbedarf)

Betrachtungsebene	interne Variable	Speicherbedarf		Laufzeit [µs]	Fehlercodes	Funktion	
		Code	Parameter			Text	Bild, Tabelle
Funktionsbaustein							

Bezüglich der Ermittlung der Bausteinkomplexität lassen sich für zwei von drei Herstellern die Anzahl der Speicher (memories) sowie der Anzahl des verwendeten Funktionsbausteins (quantity) nicht ermitteln. Für die CPU-basierte Steuerung von Hersteller 1 sind diese Kenngrößen grundsätzlich ermittelbar. Die beiden Kenngrößen können jedoch auch bei der CPU-basierten Steuerung von Hersteller 1 nur repräsentativ für separierte Funktionen ermittelt werden und stellen in diesem Fall keine belastbaren Größen im Sinne der Komplexitätsmessung dar.

Da für die Ermittlung der Bausteinkomplexität nicht alle relevanten Komplexitätsmerkmale (z.B. Verwendungshäufigkeit) bestimmt werden können, ist eine Berechnung der Bausteinkomplexität (siehe 3.1.3) nicht möglich.

Anhand der verfügbaren Komplexitätsmerkmale reduziert sich der erweiterte Komplexitätsvektor (siehe 3.2.2) auf den modifizierten Komplexitätsvektor (siehe 4.2).

Tabelle 6.19: Komponenten des modifizierten Komplexitätsvektors

ID	Bedeutung
K_m1 :	Anzahl der Funktionsbausteine
K_m2 :	Anzahl der Eingangssignale
K_m3 :	Anzahl der Ausgabesignale
K_m4 :	Anzahl der Verbindungen
K_m5 :	Anzahl der vorgelagerten Funktionspläne
K_m6 :	Anzahl der nachfolgenden Funktionspläne
K_m7 :	Komplexität der Verschaltung $V(FP)$
K_m8 :	Komplexität der Verschaltung $V_{mod}(FP)$
K_m9 :	Komplexität der Verschaltung $V_{intern}(FP)$

ID	Bedeutung
K _m 10:	Anzahl änderbarer Parameter

Die Anzahl der vorgelagerten und nachgelagerten Funktionspläne ist lediglich für die CPU-basierte Steuerung von Hersteller 1 ermittelbar. Grundsätzlich lassen sich diese beiden Größen nur für komplette leittechnische Funktionen, die mehrere Funktionspläne umfassen, bestimmen. Für die Betrachtung der Komplexität einzelner Funktionspläne sind diese beiden Größen nicht aussagekräftig. Da die Anzahl der vor- bzw. nachgelagerten Funktionspläne wichtige Komplexitätsmerkmale darstellen, bleiben diese Teil des modifizierten Komplexitätsvektors und werden für die Betrachtung alleinstehender Funktionspläne gleich Null gesetzt.

6.2 FPGA-basierte Steuerungen

6.2.1 Betrachtungsebenen

Wie bereits im Abschnitt 3 ausgeführt wird, sind Komplexitätsmerkmale von Logikschaltungen geeignet, die Logikschaltung bezüglich ihrer zu erwartenden Zuverlässigkeit zu beurteilen. Die Logikschaltung für digitale Leittechniksysteme wird entsprechend dem Stand von Wissenschaft und Technik mit grafischen Spezifikationswerkzeugen erstellt, mit deren Hilfe vorgefertigte Basisblöcke zu Netzlisten verschaltet werden. Diese Netzlisten werden danach in Konfigurationsdaten für programmierbare Logikschaltkreise (z.B. FPGAs) übersetzt.

Die Netzliste einer FPGA Anwendung weist eine Hierarchie von Bibliotheken auf, welche die leittechnischen Funktionen in sogenannten Zellen aufteilt und beschreibt. In Analogie zu der Beschreibung der Verschaltung in CPU-basierten Steuerungen können diese Zellen als Analogon zu Funktionsbausteinen aufgefasst werden. In der Netzliste von FPGAs sind diese Zellen in grundlegendere Elementarfunktionen unterteilt die mit Basisblöcken implementiert werden. Die Basisblöcke eines FPGA sind technisch als Look-Up Table (LUT) realisiert. Ein Basisblock umfasst gegebenenfalls neben der LUT auch Speicher einschließlich Flipflops und Product Terms (siehe Bild 6.1) und stellt daher eine verallgemeinerte Beschreibung der leittechnischen Elementarfunktion dar. Letztlich wird die Struktur einer FPGA Anwendung durch die Anweisung „design“ in der EDIF-Netzliste aufgerufen. Damit werden die einzelnen Basisblöcke (bzw. LUT) miteinander zu Zellen verknüpft, die wiederum (analog zu einem Funktionsplan einer CPU basierten Steuerung) verschaltet werden, um die leittechnische Funktion zu realisieren. Daraus ergeben sich zwei unterschiedliche Betrachtungsebenen für derartige Software:

- Basisblöcke programmierbarer Logikschaltkreise (LUT)
- Netzlisten als Verschaltung der leittechnischen Elementarfunktionen

Die Elemente der beiden Betrachtungsebenen besitzen jeweils eigenständige Komplexitätsmerkmale, die analog zu der im Bericht ISTec-A-1569 [1] beschriebenen Methodik in einen Komplexitätsvektor aggregiert werden.

6.2.1.1 Basisblöcke programmierbarer Logikblöcke

Programmierbare Logikblöcke⁸, die in leittechnischen Anwendungen eingesetzt werden, sind gegenwärtig meist Field Programmable Gate Arrays (FPGA). Die Beschreibung der Basisstrukturen dieser Schaltkreise variiert von Schaltkreishersteller zu Schaltkreishersteller. Eine hinreichend allgemeine Darstellung ist in [12] enthalten. Die in [12] beschriebene Schaltkreisfamilie wird z.B. von der Firma Research and Production Corporation „Radiy“ für sicherheitsrelevante kerntechnische Anwendungen eingesetzt [11]. Es wird jedoch darauf hingewiesen, dass Details der FPGA-Struktur in den Dokumentationen anderer Schaltkreisfamilien des gleichen Herstellers oder anderer Hersteller sowohl in den benutzten Bezeichnungen als auch in der Architektur von der nachfolgenden Beschreibung abweichen können.

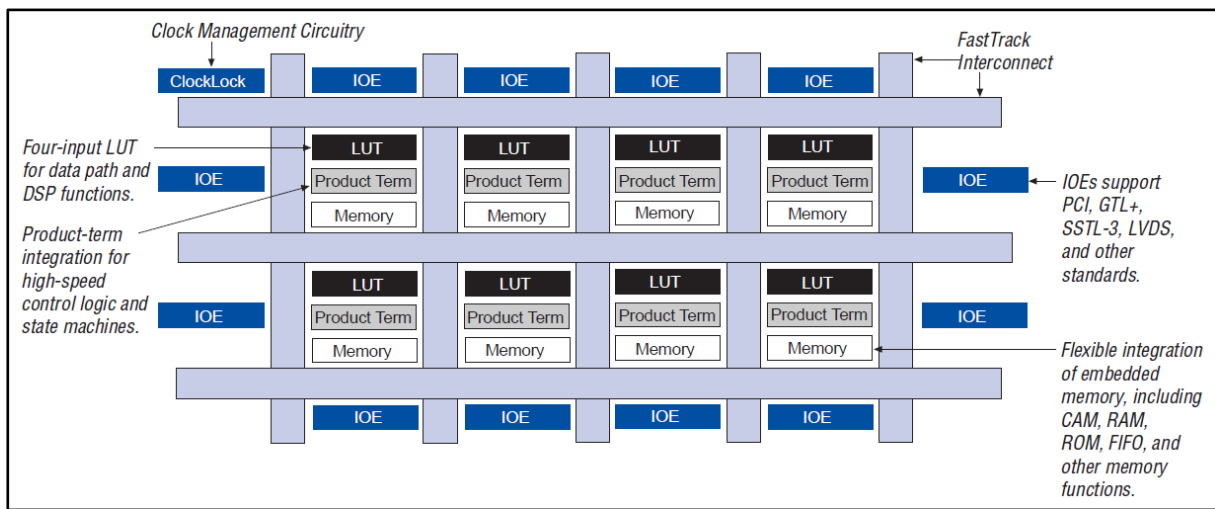


Bild 6.1: APEX II Device Block Diagram [12]

Ein Look-Up Table (LUT) besitzt in seiner einfachsten Form zwei Eingänge und einen Ausgang.

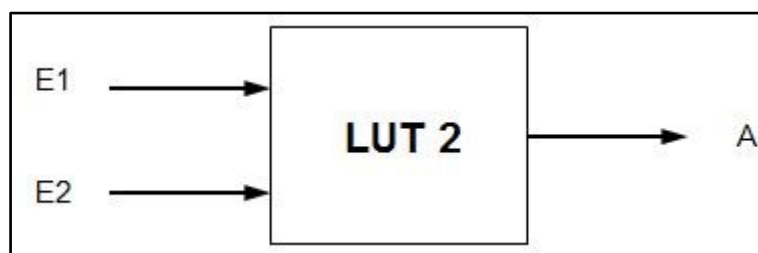


Bild 6.2: Look-Up Table

Das Verhalten ist frei programmierbar. Zum Beispiel kann das Verhalten eines logischen UND, logischen ODER oder eines logischen Exklusiv-ODER programmiert werden. Wie der

⁸ Um CPU-basierte Technik klar von programmierbarer Logik zu unterscheiden werden für CPU-basierte Technik die Begriffe „Funktionsbaustein“ und „Funktionsplan“ und für programmierbare Logik die Begriffe „Basisblock“ und „Logikblock“ verwendet.

Name Look-Up Table schon sagt, werden in einem LUT keine sequentiellen Befehle oder Microbefehle abgearbeitet, sondern wie bei einem Speicherchip der Ausgang A entsprechend der Adressleitungen [E1, E2] belegt.

Look-Up Tables mit mehr Ein- und Ausgängen erhält man durch Zusammenschaltung/Kaskadierung einfacherer Look-Up Tables.

Die Product Terms können komplexere (logische) Verarbeitungsfunktionen enthalten (z.B. zur schnellen Signalverarbeitung).

Ergänzt werden die Look-Up Tables und Product Terms durch Speicher, der auch als Dual Port Speicher oder Flipflop ausgeführt sein kann.

Für die Komplexitätsbewertung einer leittechnischen Funktion sind jeweils nur die für die Realisierung dieser Funktion herangezogenen Elemente von Belang.

6.2.1.2 Netzlisten – das EDIF-Format

Die FPGA-Anwendungen werden an dem Beispiel der FPGA Chips der APEX II Familie [12] veranschaulicht. Die FPGA Chips aus dem Beispiel der APEX II Familie sind wie folgt aufgebaut. Das Logic Element (LE) stellt die kleinste Logikeinheit in der APEX II Architektur dar (siehe Bild 6.3). Jede LE beinhaltet eine LUT Umwandlungstabelle mit vier Eingängen, die als ein funktionaler Generator dient, der in der Lage ist jede Funktion mit vier Variablen auszuführen. Außerdem beinhaltet jedes Logic Element ein programmierbares Register und Carry sowie Cascade Ketten [12]. Das Carry Eingangssignal eines Bits wird über die Carry Kette in ein Bit höherer Ordnung geschoben, um Funktionen wie Zähler, Addition und Vergleich zu beschleunigen. Die Länge der Carry Kette ist dabei nicht durch ein Logic Arrays Block (LAB), bestehend aus 10 LEs, beschränkt und kann sich somit über mehrere Logic Arrays Blocks erstrecken. Die Cascade Kette kann logische UND bzw. ODER Funktionen durchführen, die aus mehreren LE Ausgängen stammen. Die Anzahl der möglichen UND/ODER Auswahl ist dabei genau wie bei der Carry Kette nicht auf 10 LEs beschränkt. Somit kann sich die Cascade Kette über mehrere Logic Arrays Blocks erstrecken. Die Logic Elements der APEX Familie laufen funktional in einem von drei verschiedenen Modi, dem Arithmetik-, Zähler-, oder Normalmodus. Dabei verwendet jeder Modus die Ressourcen des Logic Elements unterschiedlich. Im Normalmodus werden generelle logische Anwendungen, kombinatorische Funktionen, oder umfangreiche Decodier-Funktionen, für die eine Cascade Kette nützlich sein kann, umgesetzt. Im Arithmetik-Modus werden Additions-, Sammel- und Vergleichs-Funktionen umgesetzt. Zusammengefasst hat ein Logic Element in jedem Modus insgesamt sieben verfügbare Eingänge bestehend aus den vier Eingängen die mit dem Logic Arrays Block verbunden sind, die Eingänge aus dem Register, der Carry und der Cascade Kette. Darüber hinaus verfügt jedes Logic Element über zwei Ausgänge, die lokale, MegaLAB oder FastTrack Leitstrukturen ansteuern. Die nächstgrößere Struktur LAB beinhaltet standartmäßig zehn LEs, kann jedoch unter Verwendung schnell agierender lokaler Verbindungen maximal 30 LEs kontrollieren. Jeder Logic Arrays Block beinhaltet eine ihm zugeordnete Logik um die Kontrollsignale an seine LEs und Embedded System Blocks (ESB) zu vermitteln. Diese Kontrollsignale sind für alle LE Moden verfügbar. Maximal sechs Kontrollsignale, wie z.B. Takteinstellung, synchrone und asynchrone Signaleinstellungen, können gleichzeitig verwendet werden. Dabei kann jede LAB Struktur mit zwei unterschiedlichen Taktungen gleichzeitig arbeiten.

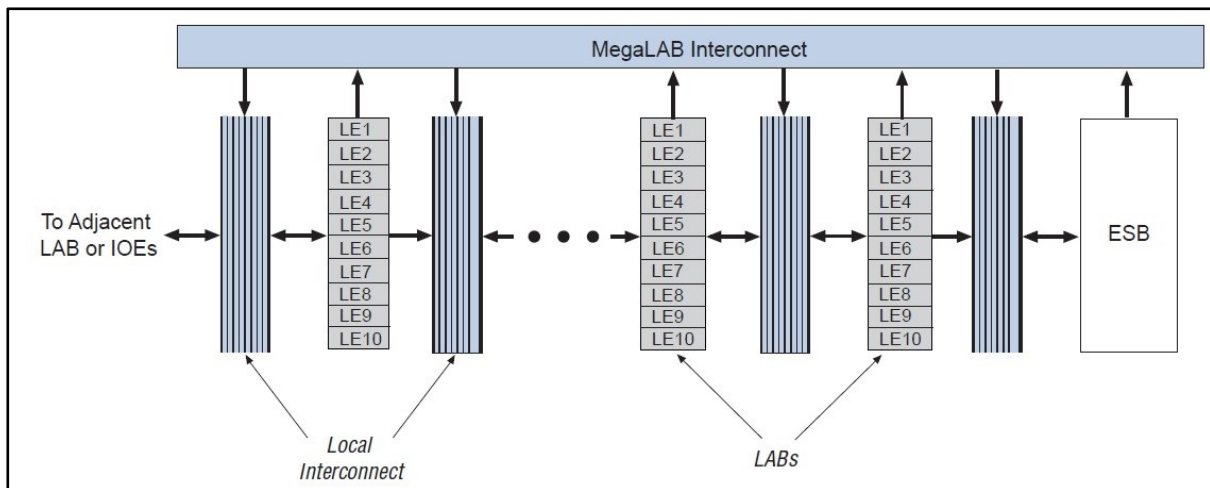


Bild 6.3: MegaLAB Struktur der APEX II Familie [12]

Die übergeordnete MegaLAB Struktur umfasst wiederum 16 bis 24 LABs sowie einen eingebauten Systemspeicher ESB und Verbindungen, die Signale innerhalb oder zwischen mehreren MegaLAB Strukturen und mittels FastTrack zu E/A-Elementen übertragen. Das gesamte FPGA besteht aus mehreren MegaLAB Strukturen, wobei die genaue Anzahl vom Typ abhängt. Die enorme Flexibilität des FPGA wird durch die Vielzahl an Verschaltungsmöglichkeiten zwischen den Logic Elements gewährleistet. Jedes LE kann über eine von 1084 Verbindungen mit anderen LEs derselben oder einer benachbarten LAB verbunden werden.

Logik, Schaltkreis und Verbindungen werden in der FPGA Architektur über einen SRAM (Static Random Access Memory) konfiguriert. Somit können FPGAs je nach benötigter Funktionalität beim Hochfahren innerhalb von 100 Millisekunden neu konfiguriert werden. Die FPGAs verfügen zudem über eine spezielle Schnittstelle, über die sie von einem Mikroprozessor seriell, parallel, synchron oder auch asynchron konfiguriert werden können, wobei die parallele Konfiguration die benötigte Konfigurationszeit deutlich senkt. Darüber hinaus erlaubt diese Schnittstelle die FPGA als Speicher zu nutzen und die Konfiguration auf einen virtuellen Speicher zu schreiben, was die Rekonfiguration erleichtert. Außerdem lassen sich hiermit Änderungen in Echtzeit während dem Systembetrieb durchführen. Damit besitzen diese FPGAs typische Softwareeigenschaften.

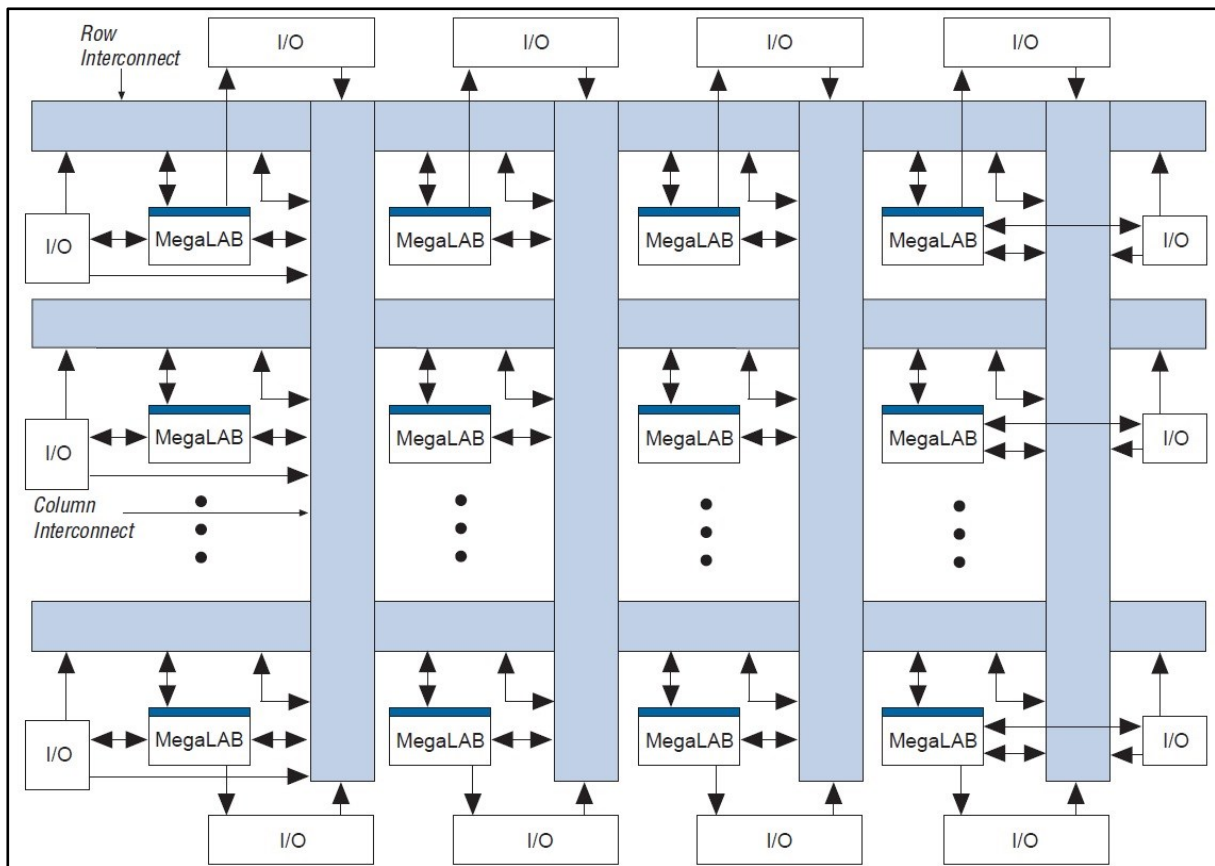


Bild 6.4: FPGA Architektur: Reihen- und Spaltenstruktur [12]

Bei FPGAs findet die Signalübertragung über variabel konfigurierbare Verbindungen statt. Die Struktur der Verbindungen lässt sich in Reihen und Spalten darstellen (siehe Bild 6.4). Die hohe Rechengeschwindigkeit wird durch eine Reihe schneller, kontinuierlicher Reihen- und Spaltenkanäle realisiert, welche die gesamte Länge und Breite des Bausteins durchziehen. Wie bereits dargelegt, können Logic Elements benachbarter Logic Arrays Blocks direkt miteinander verschaltet werden, um die Anzahl an benötigten Reihen- und Spaltenverbindungen zu minimieren.

Die E/A-Kontakte werden durch E/A-Elemente (IOE) angesteuert, welche sich am Ende jeder Reihe bzw. Spalte befinden. Jedes IOE beinhaltet einen doppelt gerichteten Puffer und sechs Register, die für Ein- und Ausgabesignale sowie eine Ausgabefreischtaltung verwendet werden. Je nach FPGA Typ können mit diesen IOEs zusätzliche Funktionen realisiert werden, weshalb sie als eine spezielle Art von LEs zu betrachten sind.

Die Systemspeicher ESB können viele Speicherarten, wie z.B. RAM, CAM (Content-Adressable Memory), ROM und FIFO Funktionen ausführen. Die Fülle an verschaltbaren ESBs ermöglicht eine hohe Speicherdichte der FPGAs, verglichen mit anderen Systemen. Zusätzlich kann jeder ESB Systemspeicher flexibel, abhängig von der Anforderung der jeweiligen Anwendung, ökonomisch mit einem niedrigen Strombedarf, oder als schneller Speicher mit erhöhtem Strombedarf eingesetzt werden.

Die Geräte der APEX Familie stellen acht Takteingänge sowie vier schnelle E/A-Kontakte, die für die globale Taktung zuständig sind, bereit. Damit wird eine beschleunigte Übertra-

gung von Signalen bei geringem Bitversatz ermöglicht. Die Komplexität der Verschaltung hängt für FPGAs somit von der Funktion bzw. den Funktionen ab, die realisiert werden sollen.

Ähnlich zu der Verschaltung von Funktionsbausteinen in Funktionsplänen bei CPU-basierten Steuerungen, spiegelt sich die Verschaltung der Basisblöcke (in sogenannten Zellen) bei einer FPGA Anwendung in einer Netzliste wider. Es bietet sich daher analog zu der Methodik für Speicherprogrammierbare Steuerungen an, die Verflechtungsmaße anhand der Verschaltung der Basisblöcke, die aus der Netzliste hervorgeht, zu ermitteln.

Anders als bei einer CPU-basierten Steuerung ist die Verschaltung in Form einer Netzliste zumeist im EDIF-Format beschrieben. Das **E**lectronic **D**esign **I**nterchange **F**ormat (kurz EDIF) ist ein durch EN 61690-1 und EN 61690-2 standardisiertes Dateiformat. Daher ist es möglich Komplexitätsmerkmale wie z.B. die Verflechtungsmaße in gleicher Weise für eine Vielzahl verschiedener FPGA Anwendungen, z.B. mit einem Skriptprogramm, auszuwerten.

Der erweiterte Komplexitätsvektor (siehe 6.1.1.2), welcher von den Komplexitätsmerkmalen von CPU-basierten Steuerungen abgeleitet ist, kann generell auch für die Komplexitätsmessung einer FPGA-basierten Anwendung verwendet werden. Die Komponenten K_{e1} bis K_{e4} , welche die Anzahl der Bausteine, der Ein- bzw. Ausgänge und der Verbindungen umfassen, können ebenso ermittelt werden, wie die Verflechtungsmaße K_{e7} bis K_{e9} . Analog zu einer CPU-basierte Steuerung könnten die Komponenten K_{e5} und K_{e6} die Anzahl der vor- bzw. nachgelagerten Netzlisten einer übergeordneten FPGA-Funktion erfassen. Die Anzahl der Änderbaren Parameter und Speicher K_{e10} und K_{e11} sind nicht in der Netzliste einer FPGA Anwendung hinterlegt, wodurch die Ermittlung der Komplexität der Basisblöcke (in Analogie zur Bausteinkomplexität) allein aus der Beschreibung durch die Netzliste nicht möglich ist. Diese Informationen könnten jedoch aus der Hardwarebeschreibung des FPGA-Chips bzw. aus der Beschreibung der Anwendung hervorgehen. Somit ergibt sich für die Betrachtungsebene der Netzliste einer FPGA Anwendung folgender Komplexitätsvektor.

Tabelle 6.20: Komponenten des Komplexitätsvektors für FPGA Netzlisten

ID	Bedeutung
K_{fe1} :	Anzahl der Basisblöcke
K_{fe2} :	Anzahl der Eingangssignale
K_{fe3} :	Anzahl der Ausgabesignale
K_{fe4} :	Anzahl der Verbindungen
K_{fe5} :	Anzahl der vorgelagerten Netzlisten
K_{fe6} :	Anzahl der nachfolgenden Netzlisten
K_{fe7} :	Komplexität der Verschaltung $V(FP)$

ID	Bedeutung
K _{fe} 8:	Komplexität der Verschaltung V _{mod} (FP)
K _{fe} 9:	Komplexität der Verschaltung V _{intern} (FP)
K _{fe} 10:	Anzahl änderbarer Parameter
K _{fe} 11:	Anzahl der internen Speicher
K _{fe} 12:	Komplexität der Basisblöcke der Netzliste

In Abhängigkeit der ermittelbaren Komplexitätsmerkmale ist eine Modifikation dieses Komplexitätsvektors notwendig (siehe 6.2.3).

6.2.2 Beispiele

Die in 6.2.1.1 dargestellten Komplexitätsmatrizen und der in 6.2.1.2 eingeführte Komplexitätsvektor werden für beispielhafte FPGA Anwendungen verschiedener Hersteller ausgewertet. Die folgende Auswertung erfolgt dabei anhand der Komplexitätsmerkmale, die für die jeweilige FPGA eines bestimmten Herstellers ermittelbar sind.

6.2.2.1 Hersteller 4

Anhand der Netzliste lassen sich für die FPGA Anwendung von Hersteller 4 folgende Komponenten des Komplexitätsvektors (siehe 6.2.1.2) ermitteln.

Tabelle 6.21: Ermitteltbare Komponenten des Komplexitätsvektors einer FPGA Netzliste von Hersteller 4

ID	Bedeutung	ermittelbar
K _{fe} 1:	Anzahl der Basisblöcke	<input checked="" type="checkbox"/>
K _{fe} 2:	Anzahl der Eingangssignale	<input checked="" type="checkbox"/>
K _{fe} 3:	Anzahl der Ausgabesignale	<input checked="" type="checkbox"/>
K _{fe} 4:	Anzahl der Verbindungen	<input checked="" type="checkbox"/>
K _{fe} 5:	Anzahl der vorgelagerten Netzlisten	<input type="checkbox"/>
K _{fe} 6:	Anzahl der nachfolgenden Netzlisten	<input type="checkbox"/>
K _{fe} 7:	Komplexität der Verschaltung V(FP)	<input checked="" type="checkbox"/>
K _{fe} 8:	Komplexität der Verschaltung V _{mod} (FP)	<input checked="" type="checkbox"/>

ID	Bedeutung	ermittelbar
K _{fe} 9:	Komplexität der Verschaltung $V_{\text{intern}}(\text{FP})$	<input checked="" type="checkbox"/>
K _{fe} 10:	Anzahl änderbarer Parameter	<input type="checkbox"/>
K _{fe} 11:	Anzahl der internen Speicher	<input type="checkbox"/>
K _{fe} 12:	Komplexität der Basisblöcke der Netzliste	<input type="checkbox"/>

Aus der Netzliste im EDIF-Format der entsprechenden FPGA Anwendung von Hersteller 4 lassen sich keine detaillierten Informationen über die Komplexitätsmerkmale zu den Basisblöcken ziehen. Eine Hardwarebeschreibung der FPGA liegt nicht vor.

6.2.2.2 Hersteller 5

Anhand der Netzliste lassen sich für die FPGA Anwendung von Hersteller 5 folgende Komponenten des Komplexitätsvektors (siehe 6.2.1.2) ermitteln.

Tabelle 6.22: Ermittlere Komponenten des Komplexitätsvektors einer FPGA Netzliste von Hersteller 5

ID	Bedeutung	ermittelbar
K _{fe} 1:	Anzahl der Basisblöcke	<input checked="" type="checkbox"/>
K _{fe} 2:	Anzahl der Eingangssignale	<input checked="" type="checkbox"/>
K _{fe} 3:	Anzahl der Ausgabesignale	<input checked="" type="checkbox"/>
K _{fe} 4:	Anzahl der Verbindungen	<input checked="" type="checkbox"/>
K _{fe} 5:	Anzahl der vorgelagerten Netzlisten	<input type="checkbox"/>
K _{fe} 6:	Anzahl der nachfolgenden Netzlisten	<input type="checkbox"/>
K _{fe} 7:	Komplexität der Verschaltung $V(\text{FP})$	<input checked="" type="checkbox"/>
K _{fe} 8:	Komplexität der Verschaltung $V_{\text{mod}}(\text{FP})$	<input checked="" type="checkbox"/>
K _{fe} 9:	Komplexität der Verschaltung $V_{\text{intern}}(\text{FP})$	<input checked="" type="checkbox"/>
K _{fe} 10:	Anzahl änderbarer Parameter	<input type="checkbox"/>
K _{fe} 11:	Anzahl der internen Speicher	<input type="checkbox"/>
K _{fe} 12:	Komplexität der Basisblöcke der Netzliste	<input type="checkbox"/>

6.2.3 Schlussfolgerung für Modifikation der Komplexitätscharakteristika

Für die betrachteten FPGA Netzlisten beider Hersteller sind jeweils dieselben Komponenten des Komplexitätsvektors $K_{fe}1$ bis $K_{fe}4$ und $K_{fe}7$ bis $K_{fe}9$ ermittelbar. Die verfügbaren Daten der betrachteten FPGA Anwendungen beider Hersteller ermöglichen die Betrachtung der jeweiligen Netzlisten als Teil einer übergeordneten Funktion nicht, weshalb die Komponenten $K_{fe}5$ und $K_{fe}6$ in beiden Fällen gleich Null gesetzt werden. Die Komponenten für die Anzahl der vorgelagerten und nachfolgenden Netzlisten sind jedoch für FPGA Anwendungen, die in einer übergeordneten Funktion durch Verschaltung von mehreren Netzlisten realisiert sind, ermittelbar und werden daher auch im modifizierten Komplexitätsvektor für FPGA-basierte Steuerungen berücksichtigt. Die Komponenten $K_{fe}10$ bis $K_{fe}12$ konnten für keine der FPGA Netzlisten beider Hersteller ermittelt werden und sind deshalb nicht im modifizierten Komplexitätsvektor enthalten.

Tabelle 6.23: Komplexitätsvektor für FPGA-basierte Steuerungen

ID	Bedeutung
K_f1 :	Anzahl der Basisblöcke
K_f2 :	Anzahl der Eingangssignale
K_f3 :	Anzahl der Ausgabesignale
K_f4 :	Anzahl der Verbindungen
K_f5 :	Anzahl der vorgelagerten Netzlisten
K_f6 :	Anzahl der nachfolgenden Netzlisten
K_f7 :	Komplexität der Verschaltung $V(FP)$
K_f8 :	Komplexität der Verschaltung $V_{mod}(FP)$
K_f9 :	Komplexität der Verschaltung $V_{intern}(FP)$

7 ANWENDUNG DER METHODIK DER KOMPLEXITÄTSMESSUNG FÜR REALE EINSATZFÄLLE IN DER KERntechnik

7.1 Anwendung auf CPU-basierte Steuerungen

7.1.1 Auswahl

Die Methodik wird im Folgenden auf Funktionspläne des Leittechniksystemtyps 1 des Herstellers 1 (kurz L1) angewandt. Die Komplexitätsmessung wird anhand der modifizierten Komplexitätsmatrizen und des modifizierten Komplexitätsvektors 6.1.3 durchgeführt.

7.1.2 Bestimmung der Kenngrößen

Zunächst wird die Komplexitätsmessung der Funktionsbausteine für einen Satz von Funktionsbausteinen von Hersteller 1 durchgeführt, die in L1 verwendet werden. Das System L1 ist für kerntechnische Anwendungen ausgelegt. Die Kenngrößen zur Berechnung der Komplexitätsmatrix sind in der nachfolgenden Tabelle für einige Funktionsbausteine exemplarisch angegeben.

Tabelle 7.1: Modifizierte Komplexitätsmatrix der L1 Funktionsbausteine Teil 1 (Signale, Parameter)

Funktionsbaustein	Signal					Parameter				
	Eingänge		Ausgänge			Nicht änderbar	änderbar	Summe	Bedingungen	abgeleitet
	binär	analog	binär	analog	Meldung					
SCHALTER	1	2	0	1	0	0	0	0	0	0
ADDITION	0	2	1	1	1	0	6	6	0	0
MAXIMUM	0	4	1	4	0	0	0	0	0	0
GRENZWERT	7	3	3	0	1	5	2	7	2	2
ODER	15	0	2	0	0	0	0	0	0	0
UND	16	0	1	0	0	0	2	2	0	0
VERZÖGERUNG	1	2	0	1	3	0	2	2	1	0
PULS	4	0	2	0	0	0	1	1	0	0
PUFFER	21	0	20	0	0	0	1	1	0	0

Tabelle 7.2: Modifizierte Komplexitätsmatrix der L1 Funktionsbausteine Teil 2 (Interne Variable, Ressourcenbedarf)

Funktionsbaustein	interne Variable	Speicherbedarf ¹		Laufzeit [μ s]	Fehler-codes	Funktion	
		Code (Worte)	Parameter (Worte)			Text	Bild, Tabelle
SCHALTER	0	220	10	100	0	5	3
ADDITION	6	1100	2	2000	12	15	6
MAXIMUM	0	308	14	200	0	5	3
GRENZWERT	2	528	14	100	0	21	5
ODER	0	93	2	30	0	5	4
UND	2	84	2	100	0	3	6
VERZÖGERUNG	1	515	10	200	0	5	4
PULS	1	226	1	43	0	9	3
PUFFER	1	270	7	100	0	41	4

¹ Die Grundeinheiten (Byte, Worte) wurden angepasst.

Tabelle 7.3: Modifizierte Tabelle zur Ermittlung der Bausteinkomplexität für L1 Funktionsbausteine

Funktionsbaustein	in_ports	out_ports	params
SCHALTER	3	1	0
ADDITION	2	2	6
MAXIMUM	4	5	0
GRENZWERT	10	3	7
ODER	15	2	0
UND	16	1	2
VERZÖGERUNG	3	1	2
PULS	4	2	1
PUFFER	21	20	1

Wie in Abschnitt 6.1.3 dargelegt, können für die Ermittlung der Bausteinkomplexität nicht alle Komplexitätsmerkmale bestimmt werden.

Für die Bestimmung des Komplexitätsvektors von L1 Funktionsplänen stehen mehrere beispielhafte Funktionen und Funktionspläne aus nuklearer Anwendung zur Verfügung. Da zwei dieser beispielhaften Funktionen (F1 und F2) funktional gleich sind und für denselben Leittechniksystemtyp L1 entwickelt wurden, ist eine Gegenüberstellung der Komplexitätsvektoren dieser beiden Funktionen möglich. Das wird in Kapitel 7.1.3 durchgeführt.

Die im Folgenden betrachtete Funktion F1 umfasst in Summe sieben Funktionspläne. Innerhalb dieser abgegrenzten Funktion sind die Funktionspläne Steuerung S1, Aggregat A1, A2, A3 und Bedienelement B1, B2 und B3 miteinander verknüpft. Die Funktionspläne A1 bis A3 sind jeweils identisch. Zunächst wurden die Komplexitätsvektoren der einzelnen Funktionspläne jeweils gesondert ausgewertet.

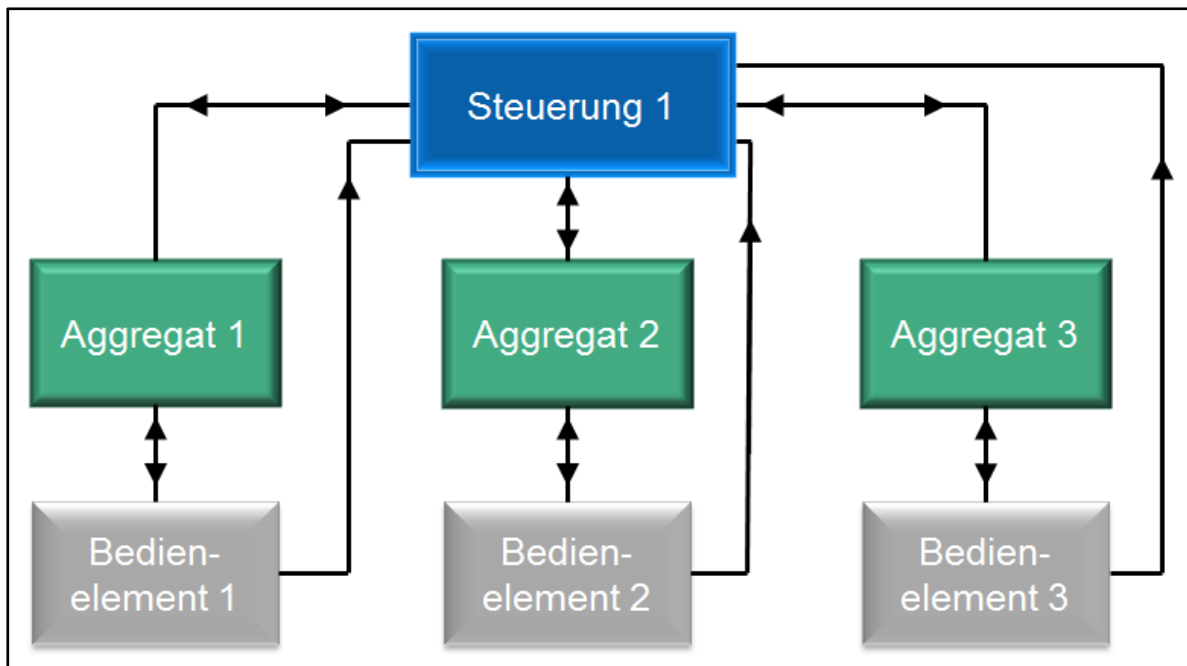


Bild 7.1: Verschaltung der Funktionspläne in Funktion F1

Die gesamte Funktion F1 beinhaltet jeden dieser sieben Funktionspläne und die Verbindungen, die zwischen den einzelnen Funktionsplänen bestehen. Hierbei ergibt sich der Komplexitätsvektor der Funktion F1 jedoch nicht einfach aus der Summe der Komplexitätsvektoren der einzelnen direkt beitragenden Funktionen, da bestimmte Ein- und Ausgänge der gesonderten Funktionspläne in der übergeordneten Funktion F1 miteinander verschaltet sind.

Der Komplexitätsvektor zur Funktion F1, der für die spätere Gegenüberstellung in Kapitel 7.1.3 benötigt wird, beinhaltet weder die Funktionspläne B1 bis B3 noch die Verbindungen zu diesen, um die Vergleichbarkeit mit F2 besser zu veranschaulichen.

In der Tabelle 7.4 sind die Komplexitätsvektoren der einzelnen Funktionspläne zusammengefasst.

Tabelle 7.4: Komplexitätsvektoren der einzelnen Funktionspläne der Funktion F1

ID	S1	A1	A2	A3	B1	B2	B3
K _m 1:	17	11	11	11	5	5	5
K _m 2:	14	18	18	18	5	5	5
K _m 3:	14	16	16	16	8	8	8
K _m 4:	55	46	46	46	19	19	19
K _m 5:	3	2	2	2	2	2	2

ID	S1	A1	A2	A3	B1	B2	B3
K _m 6:	6	2	2	2	1	1	1
K _m 7:	9,3	14,4	14,4	14,4	8,0	8,0	8,0
K _m 8:	16,0	20,3	20,3	20,3	9,9	9,9	9,9
K _m 9:	6,7	5,9	5,9	5,9	1,9	1,9	1,9
K _m 10:	6	18	18	18	25	25	25

Die Tabelle 7.5 beinhaltet die Komplexitätsvektoren der gesamten Funktion F1 sowohl mit als auch ohne Berücksichtigung der Funktionen B1, B2 und B3.

Tabelle 7.5: Komplexitätsvektoren der Funktion F1 mit und ohne Berücksichtigung der Funktionspläne B1 bis B3

ID	Funktion F1 (inkl. B1 bis B3)	Funktion F1
K _m 1:	65	50
K _m 2:	59	53
K _m 3:	62	47
K _m 4:	226	178
K _m 5:	0	0
K _m 6:	0	0
K _m 7:	57,6	41,6
K _m 8:	95,9	67,9
K _m 9:	38,3	26,3
K _m 10:	135	60

Die nächste betrachtete Funktion F2 umfasst in Summe vier Funktionspläne. Innerhalb dieser abgegrenzten Funktion wurden die Funktionspläne Steuerung S2 und die Aggregate A4, A5 und A6 miteinander verknüpft. Zunächst ist die Auswertung der Komplexitätsvektoren der einzelnen Funktionspläne jeweils gesondert dargestellt.

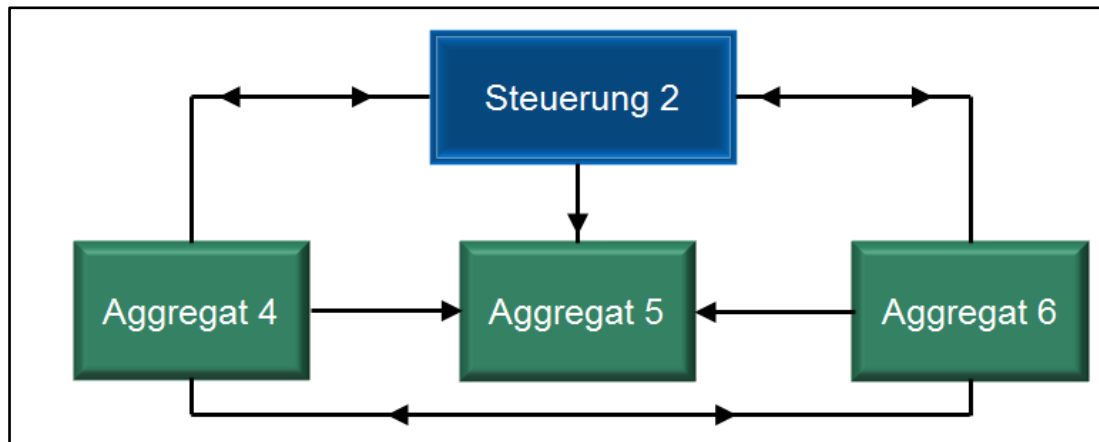


Bild 7.2: Verschaltung der Funktionspläne in Funktion F2

Die gesamte Funktion F2 beinhaltet jeden dieser vier Funktionspläne und die Verbindungen, die zwischen den einzelnen Funktionsplänen bestehen. Der Komplexitätsvektor der Funktion F2 ergibt sich wiederum nicht einfach aus der Summe der Komplexitätsvektoren der einzelnen Funktionen, da bestimmte Ein- und Ausgänge der gesonderten Funktionspläne in der übergeordneten Funktion F2 miteinander verschaltet sind.

Tabelle 7.6: Komplexitätsvektoren der einzelnen Funktionspläne und der Funktion F2

ID	S2	A4	A5	A6	Funktion F2
K _m 1:	68	112	103	100	383
K _m 2:	21	17	14	19	45
K _m 3:	50	27	16	31	99
K _m 4:	177	224	201	204	778
K _m 5:	2	3	3	3	0
K _m 6:	3	3	1	3	0
K _m 7:	43,1	12,8	8,4	11,2	66,8
K _m 8:	71,8	67,6	61,5	63,5	282,2
K _m 9:	28,7	54,8	53,1	52,3	215,4
K _m 10:	26	27	25	28	106

Neben den Funktionsplänen der Funktionen F1 und F2 standen nur wenige weitere Funktionspläne für eine Auswertung zur Verfügung. Auch deren Komplexitätsvektoren wurden ermittelt. Die Funktionspläne R1 und R2 sind Teil der Funktion F3 und beschreiben eine Leittechnische Anwendung aus der Kerntechnik, die sich funktional essentiell von F1 und F2 unterscheidet. Deshalb wurde F3 bei der Gegenüberstellung von F1 und F2 in Kapitel 7.1.3 nicht herangezogen. Außerdem wurden die Komplexitätsvektoren des Funktionsplans R3 und jeweils ein Funktionsplan (FPA), der eine Analogwerterfassung und –aufbereitung beinhaltet und ein Funktionsplan (FPB), der eine Binärwerterfassung und –aufbereitung beinhaltet ausgewertet. Die Funktionspläne R3, FPA und FPB entstammen ebenfalls kerntechnischen Anwendungen, sind jedoch keiner übergeordneten Funktion zugehörig.

Tabelle 7.7: Komplexitätsvektoren weiterer Funktionspläne von Hersteller 1

ID	R1	R2	F3	R3	FPA	FPB
K _m 1:	8	14	22	74	27	11
K _m 2:	9	10	17	80	6	3
K _m 3:	9	5	12	16	7	11
K _m 4:	25	30	53	199	55	26
K _m 5:	1	1	0	0	0	0
K _m 6:	1	1	0	0	0	0
K _m 7:	8,1	2,3	8,1	7,4	2,9	2,5
K _m 8:	11,4	5,7	16,6	17,9	15,7	5,1
K _m 9:	3,3	3,4	8,5	10,5	12,8	2,6
K _m 10:	25	31	56	45	5	34

7.1.3 Ergebnisvergleich

Da die beiden beispielhaften Funktionen F1 und F2 wie eingangs erwähnt funktional gleich sind und für denselben Leittechniksystemtyp L1 entwickelt wurden, ist eine Gegenüberstellung der Komplexitätsvektoren der beiden Funktionen und von deren Funktionsplänen möglich. Die beiden Funktionen F1 und F2 sind für unterschiedliche Anlagen und insbesondere unter unterschiedlichen Vorgaben bezüglich der Funktionsplanerstellung entwickelt worden.

Zunächst erfolgt eine Gegenüberstellung der Komplexitätsvektoren der Funktionspläne S1 und S2. Diese beiden Funktionspläne weisen trotz unterschiedlicher Anzahl von Ein- und Ausgängen die gleiche Steuerungs-Funktionalität auf.

Tabelle 7.8: Gegenüberstellung der Komplexitätsvektoren der Funktionspläne S1 und S2

ID	S1	S2
K _m 1:	17	68
K _m 2:	14	21
K _m 3:	14	50
K _m 4:	55	177
K _m 5:	3	2
K _m 6:	6	3
K _m 7:	9,3	43,1
K _m 8:	16,0	71,8
K _m 9:	6,7	28,7
K _m 10:	6	26

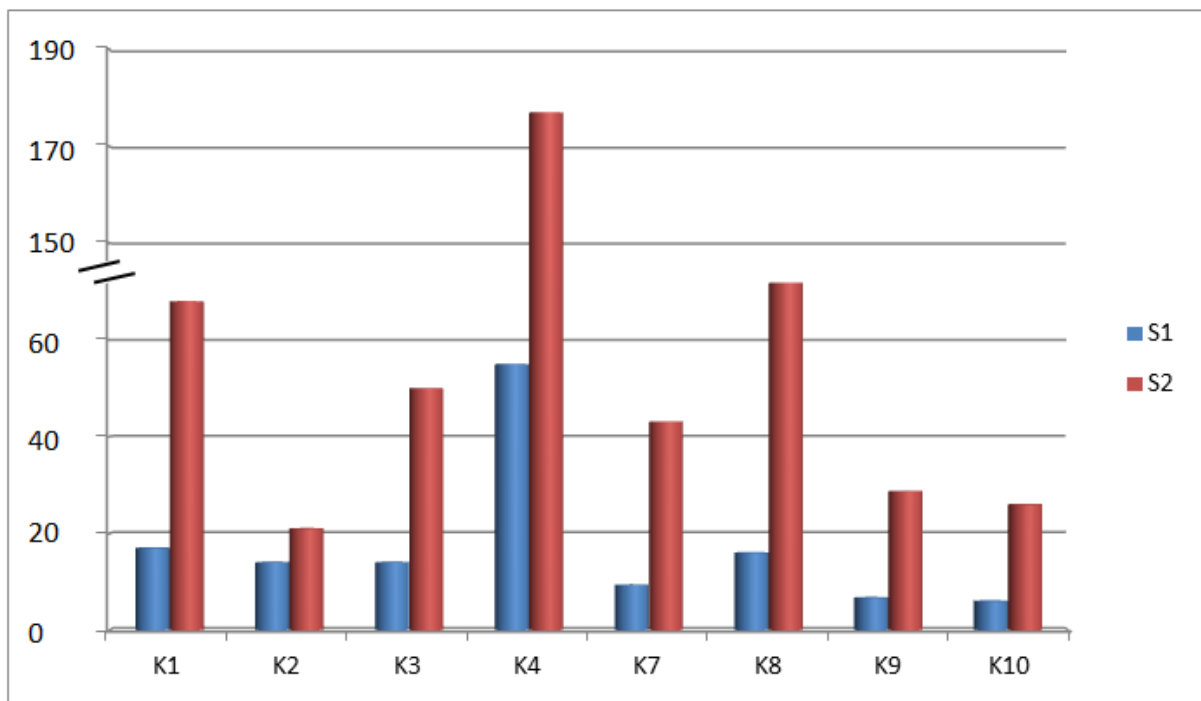


Bild 7.3: Gegenüberstellung der Komponenten der Komplexitätsvektoren von S1 und S2

Die graphische Darstellung der Gegenüberstellung in Bild 7.3 umfasst aus Gründen der Übersichtlichkeit die Komponenten K5 und K6 nicht. Um die Lesbarkeit der Daten (Unterscheidbarkeit der kleineren Werte) zu verbessern, wurde die Ordinatenachse an der mit zwei Schrägstrichen gekennzeichneten Stelle gestaucht.

Die Gegenüberstellung der Komplexitätsvektoren von Steuerung S1 und Steuerung S2 zeigt, dass die Werte fast aller Komponenten des Komplexitätsvektors S2 verglichen mit S1 signifikant höher sind. Im Vergleich zu S1 werden bei dem Funktionsplan S2 deutlich mehr Funktionsbausteine eingesetzt. Zudem sind die Verflechtungsmaße von S2 in etwa 4- bis 5-mal höher als bei S1. Bedenkt man, dass bei der Berechnung der Verflechtungsmaße die Anzahl der verwendeten Funktionsbausteine im Nenner steht und sich damit bei einer höheren Anzahl von Funktionsbausteinen in der Regel eine niedrigere Verflechtung ergibt, so ist der Funktionsplan S2 im Vergleich zu S1 als deutlich komplexer einzustufen.

Als nächstes werden die Komplexitätsvektoren der Funktionspläne der Aggregate A1 bis A3 der Funktion F1 und die Funktionspläne der Aggregate A4 bis A6 der Funktion F2 gegenübergestellt. Da die Komplexitätsvektoren der Aggregate A1 bis A3 identisch sind, werden diese in einer Spalte zusammengefasst.

Tabelle 7.9: Gegenüberstellung der Komplexitätsvektoren der Funktionspläne der Aggregate A1 bis A3 und A4 bis A6

ID	A1-3	A4	A5	A6
K _m 1:	11	112	103	100
K _m 2:	18	17	14	19
K _m 3:	16	27	16	31
K _m 4:	46	224	201	204
K _m 5:	2	3	3	3
K _m 6:	2	3	1	3
K _m 7:	14,4	12,8	8,4	11,2
K _m 8:	20,3	67,6	61,5	63,5
K _m 9:	5,9	54,8	53,1	52,3
K _m 10:	18	27	25	28

Die Gegenüberstellung der Komplexitätsvektoren der Aggregate A1 bis A3 und A4 bis A6 zeigt, dass die Anzahl der Verbindungen (K_m4) für A4 bis A6 verglichen mit A1 bis A3 etwa fünfmal höher sind und die Werte der verwendeten Funktionsbausteine (K_m1) und der internen Verflechtung (K_m9) für A4 bis A6 verglichen mit A1 bis A3 etwa zehnmal höher sind,

während sich für die übrigen Komponenten des Komplexitätsvektors keine nennenswerten Unterschiede ergeben. Dies ist insofern bemerkenswert, weil sich trotz ähnlicher Werte für das Verflechtungsmaß signifikant höhere Werte für die interne Verflechtung ergeben. Hinzu kommt, dass, wie bereits bei der Gegenüberstellung von Steuerung S1 und Steuerung S2 dargelegt, bei einer höheren Anzahl von Funktionsbausteinen in der Regel eine niedrigere Verflechtung ergibt. Somit stellt sich trotz gleicher Funktionalität für die Funktionspläne der Funktion F2 im Vergleich zu F1 eine deutlich höhere Komplexität dar.

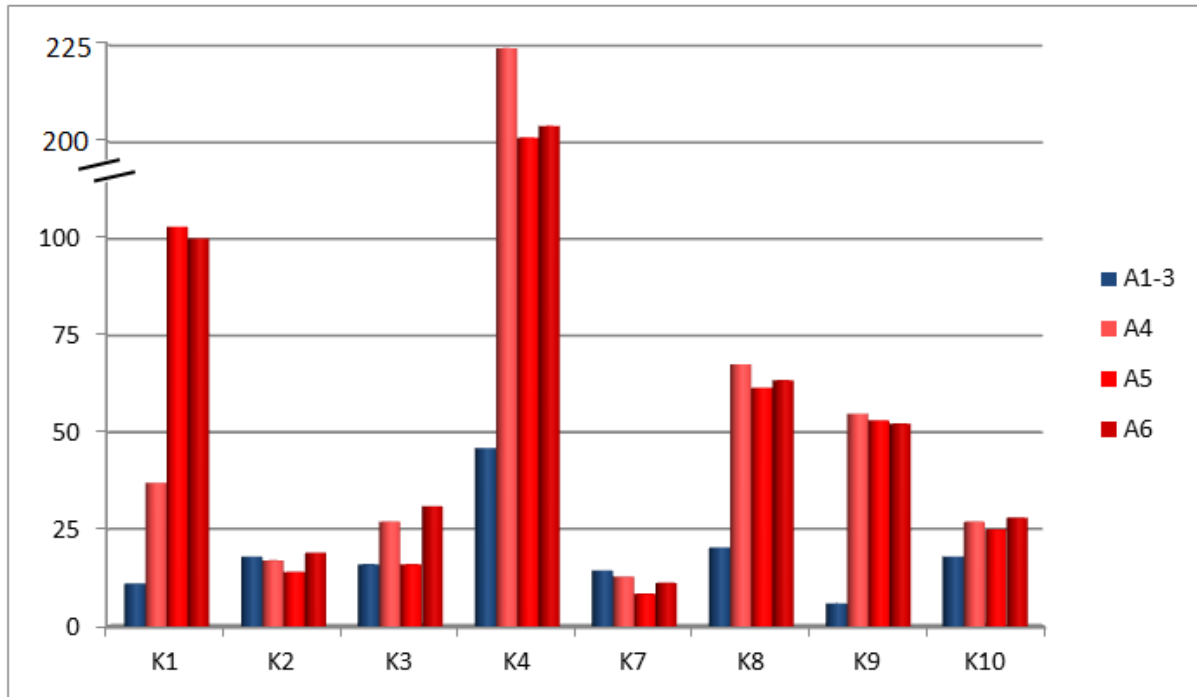


Bild 7.4: Gegenüberstellung der Komponenten der Komplexitätsvektoren von A1-3, A4, A5 und A6

Die graphische Darstellung der Gegenüberstellung in Bild 7.4 umfasst aus Gründen der Übersichtlichkeit die Komponenten K5 und K6 ebenfalls nicht. Um die Lesbarkeit der Daten (Unterscheidbarkeit der kleineren Werte) zu verbessern, wurde die Ordinatenachse an der mit zwei Schrägstrichen gekennzeichneten Stelle ebenfalls gestaucht.

Im Gegensatz zu F1 weist die Funktion F2 keine Funktionspläne mit einer zu den Bedienelementen B1, B2, oder B3 vergleichbaren Funktionalität auf. Um die jeweiligen Funktionen F1 und F2 vergleichen zu können wurde für die Funktion F1 der Komplexitätsvektor herangezogen, welcher weder die Funktionspläne der Bedienelemente B1 bis B3 noch die Verbindungen zu diesen beinhaltet.

Tabelle 7.10: Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2

ID	Funktion F1	Funktion F2
K _m 1:	50	383
K _m 2:	53	45
K _m 3:	47	99
K _m 4:	178	778
K _m 5:	0	0
K _m 6:	0	0
K _m 7:	41,6	66,8
K _m 8:	67,9	282,2
K _m 9:	26,3	215,4
K _m 10:	60	106

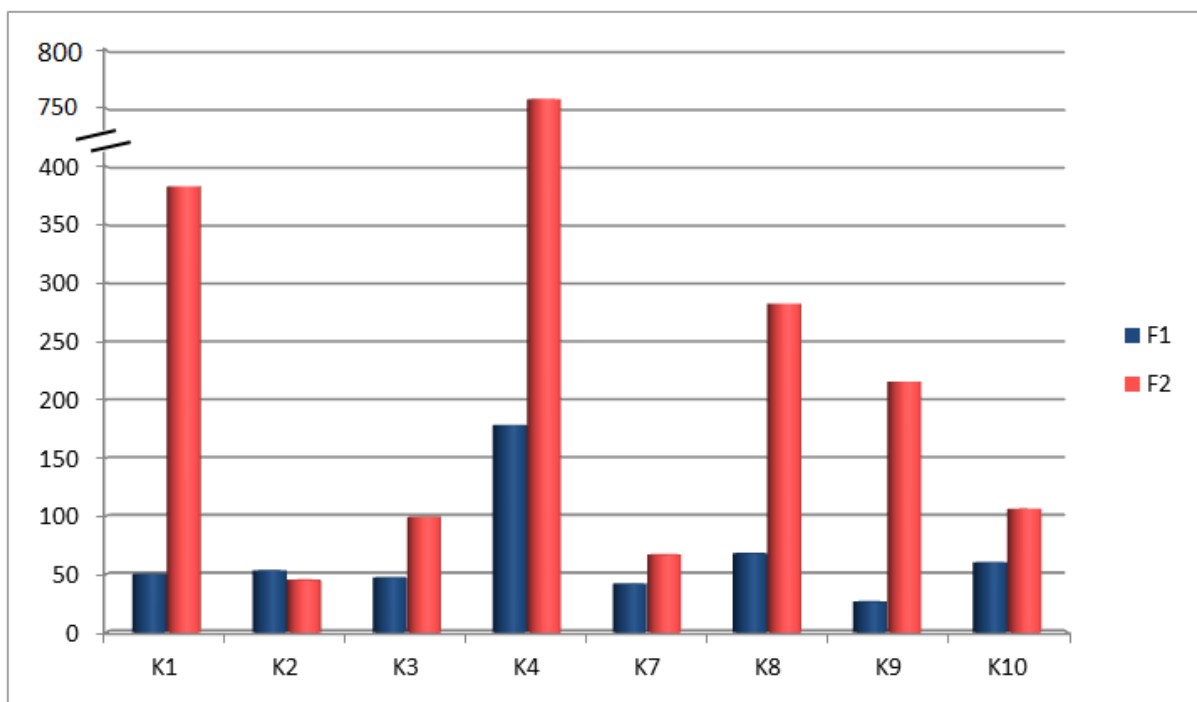


Bild 7.5: Gegenüberstellung der Komponenten der Komplexitätsvektoren von F1 und F2

Die graphische Darstellung der Gegenüberstellung in Bild 7.5 umfasst aus Gründen der Übersichtlichkeit die Komponenten K5 und K6 ebenfalls nicht. Um die Lesbarkeit der Daten (Unterscheidbarkeit der kleineren Werte) zu verbessern, wurde die Ordinatenachse an der mit zwei Schrägstrichen gekennzeichneten Stelle ebenfalls gestaucht.

Die Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2 zeigt, dass die Werte fast aller Komponenten des Komplexitätsvektors in der Funktion F2 verglichen mit F1 höher sind. Da die beiden Funktionen eine Verschaltung ihrer jeweiligen Funktionspläne darstellen und unter Berücksichtigung der bisherigen Ergebnisse ergeben sich folglich insbesondere für die Anzahl der verwendeten Funktionsbausteine bzw. Verbindungen und das modifizierte und interne Verflechtungsmaß für F2 verglichen mit F1 deutlich höhere Werte. Trotz der ähnlichen Werte für das Verflechtungsmaß ergeben sich für die Funktion F2 signifikant höhere Werte für die interne Verflechtung als für F1.

Die beiden Funktionen F1 und F2 sind bezüglich der Steuerungsfunktion funktional gleich und für denselben Leittechniksystemtyp L1, jedoch für unterschiedliche Anlagen und unter unterschiedlichen Vorgaben betreffend der Funktionsplanerstellung, entwickelt worden. Für die Funktion F2 wurden verglichen mit F1 nicht nur deutlich mehr Funktionsbausteine verschaltet, die interne Verflechtung ist zudem erheblich viel höher. Bedingt durch die unterschiedliche Gestaltung der Funktionspläne weist die Funktion F2 verglichen mit F1, trotz der gleichen Funktionalität, eine deutlich höhere Komplexität auf.

Es erfolgt keine Gegenüberstellung der Komplexitätsvektoren der Funktion F3 mit den Funktionen F1 und F2, da die Funktion F3 eine leittechnische Anwendung darstellt, die sich funktional von F1 und F2 essentiell unterscheidet.

7.2 Anwendung auf FPGA-basierte Steuerungen

7.2.1 Auswahl

Die Methodik wird im Folgenden auf die Netzlisten der FPGA-Anwendungen von Hersteller 4 und Hersteller 5 angewandt. Die Komplexitätsmessung wird anhand des modifizierten Komplexitätsvektors 6.2.3 durchgeführt.

7.2.2 Bestimmung der Kenngrößen

Die FPGA Anwendung von Hersteller 4 ist in der Darstellung der Netzliste in mehrere Designebenen unterteilt. Während sich die Anzahl der Ein- und Ausgänge nicht ändert, werden Zellen, die als umfangreiche Logikblöcke (oder Superblöcke) aufgefasst werden können, in jeder höheren Ebene in kleinere Zellen bzw. weniger umfangreiche Logikblöcke unterteilt. In der Darstellung der Ebene 0 sind alle Basisblöcke und daraus bestehende Zellen in einer großen Zelle bzw. in einem Superblock zusammengefasst. In der Darstellung der Ebene 3 ist die Netzliste bis auf nicht mehr zerlegbare Basisblöcke aufgeteilt und die Signale sind auf einzelne Bits heruntergebrochen. Anhand dieses Beispiels lässt sich veranschaulichen wie sich die Werte der Anzahl der Logikblöcke und der internen Verflechtung erhöhen, wenn die umfangreicheren (oder auch komplexeren) Logikblöcke in einfachere kleinere Unterblöcke aufgeteilt werden. Die Besonderheit ist hierbei, dass die Funktionalität für jede Ebene dieselbe ist.

Tabelle 7.11: Komplexitätsvektoren der FPGA Anwendung von Hersteller 4 für die hierarchische aufgebauten Designebenen

ID	Ebene 0	Ebene 1	Ebene 2	Ebene 3
K _f 1:	100	107	225	506
K _f 2:	75	75	75	75
K _f 3:	23	23	23	23
K _f 4:	198	228	620	1227
K _f 5:	0	0	0	0
K _f 6:	0	0	0	0
K _f 7:	20,1	17,0	13,4	4,8
K _f 8:	41,4	39,4	94,9	114,6
K _f 9:	21,3	22,4	81,5	109,7

Stellt man die Komplexitätsvektoren der Designebenen gegenüber zeigt sich, dass der Wert des Verflechtungsmaßes $V(FP)$ für jede höhere Eben kleiner wird. Dies liegt vor allem daran, dass bei der Berechnung bei gleicher Anzahl von Ein- und Ausgängen im Zähler des Verflechtungsmaßes die Anzahl der Basisblöcke im Nenner steigt und somit zu einem kleineren Wert führt. Der Beitrag der internen Verflechtung zur modifizierten Verflechtung wird jedoch gleichzeitig größer, da bei der Berechnung des modifizierten Verflechtungsmaßes die Mächtigkeit der Vorbereiche eines jeden Blocks zunimmt.

Die FPGA Anwendungen von Hersteller 5 sind in der Darstellung der Netzliste nicht wie bei der FPGA Anwendung von Hersteller 4 in mehrere Ebenen unterteilt, sondern weisen ähnlich wie die betrachteten Funktionspläne der CPU-basierten Steuerungen eine flache Struktur auf einer Ebene auf, die der Ebene 3 im Design von Hersteller 4 entspricht.

Die FPGA-basierten Steuerungen von Hersteller 5 repräsentieren drei verschiedene Funktionen, die im Folgenden als Design 1, Design 2 und Design 3 bezeichnet werden.

Tabelle 7.12: Komplexitätsvektoren der FPGA Anwendung von Hersteller 5 für die Designs 1, 2 und 3

ID	Design 1	Design 2	Design 3
K _f 1:	504	673	727
K _f 2:	18	16	18
K _f 3:	19	19	19
K _f 4:	1543	2094	2418
K _f 5:	0	0	0
K _f 6:	0	0	0
K _f 7:	6,7	3,9	9,2
K _f 8:	115,9	69,0	181,5
K _f 9:	109,2	65,1	172,3

7.2.3 Ergebnisvergleich

Die Gegenüberstellung der Komplexitätsvektoren der FPGA-basierten Steuerung von Hersteller 4 (Ebene 3) und Hersteller 5 (Design 1) zeigt, dass trotz der unterschiedlichen Funktionalität der beiden FPGA Anwendungen, der unterschiedlichen Hersteller und der unterschiedlichen Anzahl von Ein- und Ausgangssignalen die Komponenten beider Komplexitätsvektoren nahezu identisch sind. Die Komponenten der Komplexitätsvektoren von Design 2 und Design 3 (jeweils Hersteller 5) liegen in einem vergleichbaren Wertebereich.

Bei der Gegenüberstellung wird bei der FPGA-basierten Steuerung von Hersteller 4 die Ebene 3 herangezogen, da die zugehörige Darstellung in der Netzliste ebenso wie bei den Netzlisten der Anwendungen von Hersteller 5 bis auf nicht mehr zerlegbare Basisblöcke heruntergebrochen ist.

Die Designs 2 und 3 von Hersteller 5 kommen auf derselben Baugruppe zum Einsatz. Bei der Gegenüberstellung der Komplexitätsvektoren fällt jedoch auf, dass für Design 2 trotz der höheren Anzahl von verwendeten Blöcken und Verbindungen verglichen mit Design 1 niedrigere Werte für die Verflechtungsmaße ermittelt wurden. Im Gegensatz dazu treten bei Design 3 verglichen mit den anderen Designs für alle Komponenten des Komplexitätsvektors höhere Werte auf. Die Gegenüberstellung zeigt, dass sich trotz der geringen Unterschiede bezüglich der Anzahl der Blöcke, der Ein- und Ausgangssignale und der Verbindungen für Design 3 größere Werte der Verflechtungsmaße ergeben als für Design 2. Daraus lässt sich schließen, dass die Mächtigkeit der Vorbereiche der jeweiligen Blöcke aufgrund der unterschiedlichen Verschaltung für Design 3 im Vergleich zu Design 2 im Mittel höher ist und sich somit für Design 3 eine komplexere Verflechtung ergibt.

In Tabelle 7.13 sind die vergleichbaren Komplexitätsvektoren zusammengefasst.

Tabelle 7.13: Gegenüberstellung der Komplexitätsvektoren von FPGA-basierten Steuerungen

ID	Ebene 3	Design 1	Design 2	Design 3
	Hersteller 4	Hersteller 5	Hersteller 5	Hersteller 5
K _f 1:	506	504	673	727
K _f 2:	75	18	16	18
K _f 3:	23	19	19	19
K _f 4:	1227	1543	2094	2418
K _f 5:	0	0	0	0
K _f 6:	0	0	0	0
K _f 7:	4,8	6,7	3,9	9,2
K _f 8:	114,6	115,9	69,0	181,5
K _f 9:	109,7	109,2	65,1	172,3

8 VERÖFFENTLICHUNG DER IM VORHABEN ERZIELTEN ERGEBNISSE

Die im Rahmen des Vorhabens erarbeiteten Ergebnisse wurden in einem Vortrag mit dem Titel „Complexity and Error Potential of Digital I&C“ [14] auf der vom 11. bis 15. Juni 2017 stattgefundenen 10. Tagung der Tagungsreihe „International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies“ (NPIC & HMIT 2017) in San Francisco, USA, vorgetragen. Die Tagungsreihe ist die zurzeit weltweit bedeutendste internationale Fachtagung auf dem Gebiet der KKW-Leittechnik. Der Vortrag des ISTec war der technischen Sitzung „Digital System Reliability“, einer technischen Sitzung, die insgesamt zwölf Vorträge aufwies, zugeordnet.

Es ist nicht einfach, alle komplexen mathematischen Zusammenhänge des Vorhabens in der begrenzten Vortragszeit von 20 Minuten den Zuhörern zu vermitteln. Der Beitrag konzentrierte sich deshalb auf die Darstellung der Anwendbarkeit der Methodik der Komplexitätsmessung sowohl auf CPU-basierte, als auch FPGA-basierte technische Systeme. Es wurden die bisherigen Ergebnisse anhand der Gegenüberstellung von zwei exemplarischen CPU-basierten Leittechnikfunktionen in nuklearen Anwendungen und einer FPGA-basierten Leittechnikfunktion in einer nuklearen Anwendung veranschaulicht. In der dem Vortrag folgenden Diskussion mit den Teilnehmern der Sitzung wurde ein Vergleich der in dem Vorhaben erzielten Ergebnisse mit anderen Verfahren zur Komplexitätsmessung angeregt.

Der Beitrag in den Proceedings der Tagung ist im Anhang D und die zugehörige PowerPoint-Präsentation ist im Anhang E diesem Bericht angefügt.

9 BEWERTUNG DER ERGEBNISSE IM HINBLICK AUF DIE ZUVERLÄSSIGKEIT

9.1 Grundannahmen

Der Zusammenhang zwischen der Komplexität und der Zuverlässigkeit von Software lässt sich folgendermaßen darstellen. Entscheidend für einen durch Software bedingten Fehler ist zunächst einmal, dass ein Fehler in der Software vorhanden ist, der durch versagensauslösende Eingangsdaten zur Wirkung gebracht wird (der Fehler wird in der Software „getriggert“). Dabei steigt die Wahrscheinlichkeit für ein Versagen der Software (bzw. es sinkt die Zuverlässigkeit der Software) mit der Anzahl der Fehler in der Software. Um die Menge der Fehler in der Software zu minimieren, werden Qualitätssicherungsmaßnahmen, wie z.B. umfangreiche Verifikations- und Validierungsverfahren angewendet. Dabei beinhaltet die Validierung umfangreiche Funktionstests. Während für einfache Softwarebausteine nahezu alle möglichen Pfade in der Software getestet werden können, ist dies für komplexere Systemstrukturen jedoch kaum bzw. gar nicht möglich. Je komplexer die zugrundeliegende Software ist, desto geringer ist die erreichbare Testabdeckung [13]. Somit steigt mit der Komplexität der Software die Möglichkeit, dass Fehler, die eventuell in der Software enthalten sind durch die angewandten Qualitätssicherungsmaßnahmen und Tests unentdeckt bleiben und damit die Wahrscheinlichkeit eines potenziellen Fehlers. Das heißt, mit steigender Komplexität sinkt potenziell die Zuverlässigkeit von Software.

Im Gegensatz zur Hardware unterliegt die Software keiner „Verschlechterung“ durch Alterung. Softwarefehler sind deshalb immer systematische Fehler, die von Anfang der Verwendung der Software in dieser vorhanden sind.

Unter Anwendung von Bayesian-Belief-Netzen (BBN) wurde in dem Vorläuferprojekt [1] die Zuverlässigkeit von Funktionsplänen anhand deren Komplexität bewertet. Dieses Verfahren kann ebenfalls auf die Software der Steuerungen, die während des aktuellen Vorhabens untersucht wurden, angewendet werden. Die im Vorläuferprojekt ermittelten Maßstäbe des BBN Verfahrens für die Bewertung der Zuverlässigkeit können unter der Voraussetzung, dass die entsprechenden Komplexitätscharakteristika ermittelbar sind, nach bisherigem Kenntnisstand auf beliebige CPU-basierte Steuerungen angewendet werden.

FPGA basierte Steuerungen weisen im Allgemeinen eine deutlich feinere Granularität verglichen mit CPU-basierten Steuerungen auf. Während für einen Funktionsplan einer CPU-basierten Anwendung die Verschaltung von 100 oder mehr Funktionsbausteinen in der Regel als hoch komplex angesehen wird und zu einer negativen Bewertung der Zuverlässigkeit führt, stellt eine Anzahl von 100 oder mehr Basisblöcken in der Netzliste einer FPGA Anwendung, bedingt durch die Verschaltung elementarster Logikblöcke eher den aktuellen Stand der Technik dar. Daher ist für eine Anwendung von BBNs auf FPGAs eine andere Struktur des BBN erforderlich und es bedarf der ingenieurtechnischen Bewertung von Zuverlässigkeitseigenschaften von FPGA-Strukturen. Beides ist zum gegenwärtigen Zeitpunkt nicht verfügbar. Da die elementaren Basisblöcke in FPGA-Anwendungen im Vergleich zu Funktionsbausteinen CPU-basierter Anwendungen weniger starke Variationen bezüglich

ihrer Komplexität aufweisen, ist zu erwarten, dass die Komplexität einer FPGA-Anwendung von den Verflechtungsmaßen dominiert wird.

9.2 Ableitung von Kriterien zur Bewertung der Zuverlässigkeit

Ein erster Ansatz für die Ableitung von Kriterien zur Bewertung der Zuverlässigkeit ist durch die Verflechtungsmaße gegeben. Die Bewertung der Zuverlässigkeit auf Grundlage der Bayesian Belief Networks (BBN) in dem Vorläuferprojekt hat unter anderem gezeigt, dass besonders die Funktionspläne als unzuverlässig eingestuft wurden, die eine hohe Komplexität der Verschaltung aufwiesen. Es wurde deutlich, dass die Zuverlässigkeit eines Funktionsplans in nahezu allen untersuchten Funktionsplänen aus der Komplexität der Verschaltung ersichtlich wird. In diesem Zusammenhang wurde jedoch auch darauf hingewiesen, dass sich die Komplexität der Verschaltung eines Funktionsplans durch eine hohe Komplexität einzelner Funktionsbausteine verschleiern lässt. In dem aktuellen Vorhaben wurde deutlich, dass sich dieser Vorbehalt für FPGA-basierte Steuerungen nicht ergibt, da die einzelnen Signale bis zu Bit-Ebene heruntergebrochen werden. Somit können die Verflechtungsmaße, insbesondere für die programmierbare Logik von FPGAs, für die Bewertung der Zuverlässigkeit herangezogen werden.

Die im aktuellen Vorhaben erarbeitete Erweiterung der Methodik ermöglicht die Ermittlung des internen Verflechtungsmaßes. Bei der Ermittlung des Komplexitätsvektors auf Grundlage der bisher gesammelten Daten hat sich das interne Verflechtungsmaß als das augenscheinlichste Merkmal für die Komplexität der Funktionspläne erwiesen. Bei einer programmierbaren Logik sind die einzelnen Logikblöcke elementar aufgebaut, weshalb sich für diese Technologie aus der strukturellen Komplexität, die sich am besten an dem internen Verflechtungsmaß ablesen lässt, ein potentiell Kriterium für die Bewertung der Zuverlässigkeit ableiten lässt. Für CPU-basierte Anwendungen kann diese Aussage nicht gleichermaßen getroffen werden, da sich durch die Realisierung einer höheren Bausteinkomplexität die strukturelle Komplexität senken und somit künstlich ein niedrigerer Wert für das interne Verflechtungsmaß erzielen lässt. Für beide Technologien gilt, dass eine Festlegung von Akzeptanzkriterien rein nach Zahlenwerten der Metriken per-se nicht immer zielführend ist, da die ermittelten Werte vor allem von der Funktionalität abhängen und somit anwendungsspezifisch sind. Die bisher gesammelten Daten haben jedoch gezeigt, dass eine Gegenüberstellung von ähnlichen Funktionen für die Ermittlung von Unterschieden der Komplexität und der zu erwartenden Zuverlässigkeit herangezogen werden kann. Eine Gegenüberstellung der Komplexitätsvektoren von Funktionsplänen ähnlicher Funktionalität ist in der Lage Unterschiede beim Softwaredesign und dessen Auswirkung auf die Komplexität zu ermitteln. Mit der Zielsetzung die Zuverlässigkeit der Software zu erhöhen kann diese Vorgehensweise verwendet werden, um das Softwaredesign hinsichtlich der Verringerung der Komplexität zu optimieren.

Die in der Tabelle 9.1 enthaltene Gegenüberstellung der Komplexitätsvektoren ähnlicher Funktionen zweier CPU-basierter Anwendungen zeigt ihre Unterschiede auf. Die beiden Funktionen wurden für die gleiche Leittechnikplattform von zwei unterschiedlichen Leittechnikprojektanten für die Verwendung in nuklearen Anlagen für sehr ähnliche Funktionen der Leittechnik entworfen.

Im Gegensatz zu Funktion F1 sind die Funktionspläne der Funktion F2 nicht nach den Entwurfsvorgaben des Herstellers der Leittechnikplattform gestaltet worden.

Tabelle 9.1: Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2

ID	Funktion F1	Funktion F2
K _m 1:	50	383
K _m 2:	53	45
K _m 3:	47	99
K _m 4:	178	778
K _m 5:	0	0
K _m 6:	0	0
K _m 7:	41,6	66,8
K _m 8:	67,9	282,2
K _m 9:	26,3	215,4
K _m 10:	60	106

Die Gegenüberstellung der Komplexitätsvektoren der Funktionen F1 und F2 zeigt, dass die Werte fast aller Komponenten des Komplexitätsvektors in der Funktion F2 verglichen mit F1 höher sind. Für die Funktion F2 wurden verglichen mit F1 nicht nur deutlich mehr Funktionsbausteine verschaltet, die interne Verflechtung ist nahezu eine Größenordnung höher. Bedingt durch die unterschiedliche Gestaltung der Funktionspläne weist die Funktion F2 verglichen mit F1, trotz gleicher Funktionalität, eine deutlich höhere Komplexität auf. In dem hier dargestellten Beispiel lassen die deutlichen Unterschiede der Werte der Komplexitätsvektoren trotz gleicher Funktionalität der beiden Funktionen den Schluss zu, dass die Funktionspläne der Funktion F2 potenziell eher Fehler enthalten und damit als weniger zuverlässig anzusehen sind, als die Funktionspläne der Funktion F1.

Dieses Beispiel zeigt die Unterschiede der Komplexität aufgrund des unterschiedlichen Funktionsplandesigns auf. Dies ermöglicht sowohl Herstellern solcher Systeme Anforderungen nach Einfachheit von Sicherheitsleittechnik zu erfüllen als auch Prüfern diese Systemeigenschaften objektiv zu bewerten.

Wie bereits dargelegt wurde in dem Vorhaben deutlich, dass die Netzlisten von FPGAs bis zu einer Verschaltung elementarster Logikbausteine (LUTs, Flipflops) reichen. Diese sind in Ihrer internen Komplexität nicht wesentlich unterschiedlich (z.B. besteht das Ersatzschaltbild eines Flipflop aus vier Nand-Gattern). Da sich in dieser Darstellung der Verschaltungsebene im Gegensatz zu CPU-basierten Anwendungen keine Komplexität in die Betrachtungsebene eines Bausteins verschieben lässt, bietet sich für die Bewertung der Zuverlässigkeit die Betrachtung des internen Verflechtungsmaßes des Komplexitätsvektors für FPGA-basierte

Steuerungen besonders an⁹. Es ist offensichtlich, dass die Werte der Verflechtungsmaße CPU-basierter und FPGA-basierter Anwendungen nicht direkt vergleichen lassen, sondern einer Interpretation bedürfen.

Bei der in der Tabelle 9.2 enthaltenen Gegenüberstellung wird bei der FPGA-basierten Steuerung von Hersteller 4 die Ebene 3 herangezogen, da die zugehörige Darstellung in der Netzliste ebenso wie bei den Netzlisten der Anwendungen von Hersteller 5 bis auf nicht mehr zerlegbare Basisblöcke aufgeteilt ist.

Tabelle 9.2: Gegenüberstellung der Komplexitätsvektoren von FPGA-basierten Steuerungen

ID	Ebene 3	Design 1	Design 2	Design 3
	Hersteller 4	Hersteller 5	Hersteller 5	Hersteller 5
K _f 1:	506	504	673	727
K _f 2:	75	18	16	18
K _f 3:	23	19	19	19
K _f 4:	1227	1543	2094	2418
K _f 5:	0	0	0	0
K _f 6:	0	0	0	0
K _f 7:	4,8	6,7	3,9	9,2
K _f 8:	114,6	115,9	69,0	181,5
K _f 9:	109,7	109,2	65,1	172,3

Die Gegenüberstellung der Komplexitätsvektoren der FPGA-basierten Steuerung von Hersteller 4 (Ebene 3) und Hersteller 5 (Design 1) zeigt, dass trotz der unterschiedlichen Funktionalität der beiden FPGA Anwendungen, der unterschiedlichen Hersteller und der unterschiedlichen Anzahl von Ein- und Ausgangssignalen die Komponenten beider Komplexitätsvektoren nahezu identisch sind. Für diese beiden FPGA-basierten Steuerungen ist demnach anhand der Betrachtung ihrer Komplexität von einer gleichen Zuverlässigkeit auszugehen.

Die Gegenüberstellung der unterschiedlichen Designs von Hersteller 5 zeigt, dass sich trotz einer höheren Anzahl an Verbindungen niedrigere Werte für die Komplexitätsmaße ergeben

⁹ In komplexeren Anwendungen, die möglicherweise Basisblöcke mit stärker differenzierter Komplexität beinhalten, kann diese ihren Niederschlag im Komplexitätsvektor finden. Bei sicherheitsrelevanten Steuerungsfunktionen ist damit aufgrund der Forderung nach Einfachheit aber nicht zu rechnen.

können, wie es bei Design 2 der Fall ist. Wie die nähere Betrachtung zeigt, ist die durchschnittliche Mächtigkeit der Vorbereiche der jeweiligen Basisblöcke aufgrund der unterschiedlichen Verschaltung für Design 2 verglichen mit den anderen FPGA-basierten Anwendungen niedriger, wodurch sich für Design 2 niedrigere Verflechtungsmaße ergeben. Dies lässt den Schluss zu, dass verglichen mit den anderen FPGA-Anwendungen Design 2 eine potenziell höhere Zuverlässigkeit aufweist, was sich insbesondere an dem vergleichsweise kleinen Wert des Verflechtungsmaßes ableiten lässt.

Für CPU-basierte Systemanwendungen gehören ein „top-down“ Aufbau und eine modulare Struktur zu den wichtigsten Konzepten, um den Problemen einer unvermeidlichen Komplexität entgegenzuwirken [13]. Diese Modularisierung bezieht sich immer auf den Kontrollfluss des Programms. Bild 9.1 zeigt beispielhaft einen Kontrollflussgraph, wie er von statischen Analysewerkzeugen erzeugt wird.

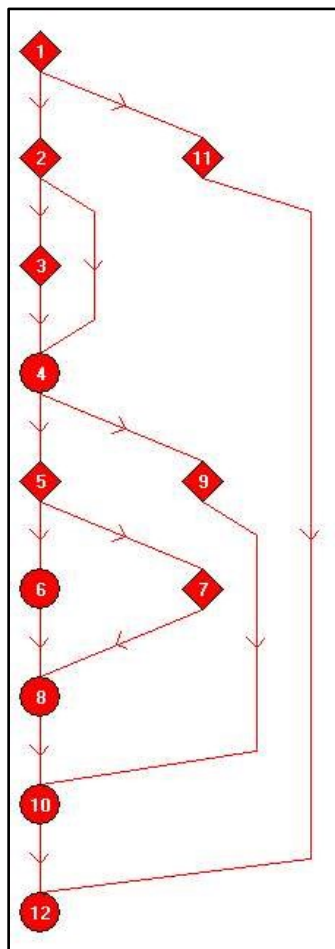


Bild 9.1: Beispielhafter Kontrollflussgraph, erzeugt mit LDRA Testbed

Eine Modularisierung von FPGA-basierten Anwendungen ist zwar ebenfalls möglich, diese hat jedoch auf Grund der massiv parallelen Bearbeitung der Funktionen keinen Einfluss auf den Kontrollfluss, sondern bezieht sich auf die Darstellung des Datenflusses. In der Netzliste wird der Datenfluss immer bis auf die Ebene der elementaren Basisblöcke aufgelöst. Ein anschauliches Beispiel dafür ist die Gegenüberstellung von CPU-basierten und FPGA-basierten Steuerungen. Während die einzelnen Bausteine einer typischen CPU-basierten Realisierung zumeist deutlich komplexer sind als die elementaren Logikblöcke einer FPGA

Anwendung, ist die Verschaltung der Logikblöcke einer FPGA Anwendung zumeist deutlich komplexer, als die Verschaltung der Funktionsbausteine einer CPU-basierten Anwendung, um die gewünschte Funktionalität zu ermöglichen.

Wie im Anhang B erläutert, ermittelt das Verfahren zur Berechnung der Verflechtungsmaße strukturelle Eigenschaften, die auf den Datenfluss bezogen sind. Dabei handelt es sich um Aussagen bezüglich:

- Isolierter Teile eines Funktionsplans bzw. einer Netzliste.
- Vorhandene Schleifen im Funktionsplan bzw. in einer Netzliste, d.h. direkte Rückkopplungen von einem Ausgang eines Funktionsbausteins bzw. eines Basisblocks auf einen Eingang desselben Funktionsbausteins bzw. eines Basisblocks.

Anhand der strukturellen Merkmale lässt sich ebenfalls ein Kriterium für die Bewertung der Zuverlässigkeit ableiten, da sowohl die Verfügbarkeit als auch die Zuverlässigkeit der Software davon abhängen [13].

Dabei trifft eine Analyse der strukturellen Merkmale einer Software Aussagen darüber, ob einzelne Softwareteile isoliert oder unerreichbar sind. Die Analyse der strukturellen Merkmale einer programmierbaren Logik liefert Aussagen darüber, ob einzelne Bereiche mit logischen Blöcken auf einem FPGA isoliert oder unerreichbar sind, bzw. ob Redundanzen auf einem FPGA wirklich separiert sind.

Obwohl die strukturelle Analyse theoretisch manuell durchgeführt werden kann, ist eine manuelle Analyse der Objekte¹⁰ sehr fehleranfällig. Mit steigender Komplexität und Größe der zu prüfenden Objekte wird die Verwendung von Werkzeugen erforderlich [13].

Bezüglich der Aussagekraft der Komplexitätsmessung für die in den Abschnitten 6.1.1 und 6.2.1 eingeführten Betrachtungsebenen ergibt sich das in Tabelle 9.3 wiedergegebene Bild.

Tabelle 9.3: Bewertungskriterien

Betrachtungsebene	Bedeutung für Zuverlässigkeitsbewertung CPU-basierter Anwendung	Bedeutung für Zuverlässigkeitsbewertung FPGA-basierter Anwendung
Komplexität der elementaren Bausteine	hoch	niedrig
Verflechtungsmaße	hoch	hoch

Daraus und aus dem unterschiedlichen Aufbau der Komplexitätsvektoren für CPU-basierte und FPGA-basierte Anwendungen lässt sich ableiten, dass ein direkter Vergleich der Komplexität der unterschiedlichen Technologien nur schwer möglich ist.

¹⁰ Code der Software oder Netzliste einer FPGA-Anwendung.

10 ZUSAMMENFASSUNG

Die Einführung und Weiterentwicklung immer komplexerer, verfahrenstechnischer Systeme in den verschiedensten Bereichen führt auch in der Kerntechnik zu einer erhöhten Verwendung programmierbarer digitaler Leittechniksysteme mit sicherheitstechnischer Bedeutung. Die Frage nach einer Ermittlung der Zuverlässigkeit dieser Systeme mithilfe eines allgemein anerkannten Verfahrens ist dabei noch nicht abschließend geklärt. Es gibt eine Reihe von Ansätzen, die meist Teilprobleme behandeln [13]. Die Ergebnisse des Vorhabens leisten ebenfalls einen Beitrag zur Bewertung von Zuverlässigkeitseigenschaften auf der Basis von Komplexitätsberechnungen.

Wie die praktische Erfahrung zeigt, ist es die Komplexität eines programmierbaren digitalen Systems in Verbindung mit der Vielfalt der Anwendungsprofile, wovon das Risiko einer Fehlfunktion des Systems entscheidend abhängt. Deshalb stellt sowohl die Bestimmung der Komplexität der Software/programmierbaren Logik, als auch der Ansatz, die Zuverlässigkeit programmierbarer digitaler Leittechniksysteme auf der Grundlage der Komplexität der Software/programmierbaren Logik zu bewerten eine wichtige Zielsetzung dar.

Eine Methode zur Messung der Komplexität, die sich auf ein strukturiertes Vorgehen stützt, wurde in einem Vorläuferprojekt für eine spezifische CPU-basierte Leittechnikplattform entwickelt. Um das entwickelte Konzept für verschiedene digitale Leittechniksysteme verwendbar zu gestalten, wurde die Methode sowohl für CPU-basierte Anwendungen weiterentwickelt als auch für FPGA-basierte Anwendungen nutzbar gemacht.

Das zur Bewertung der Komplexität definierte Verflechtungsmaß der Funktionspläne wurde um das modifizierte und interne Verflechtungsmaß erweitert, um eine höhere Aussagekraft der Messung der Komplexität zu erzielen. Die Berechnung des Verflechtungsmaßes basiert auf der Verschaltung der Funktionsbausteine bzw. Basisblöcke und der Eingabe-Signale des Funktionsplans bzw. der Netzliste und ist sowohl auf Funktionspläne CPU-basierter Steuerungen als auch auf Netzlisten von FPGA basierten Steuerungen anwendbar. Neben den Verflechtungsmaßen werden weitere Komplexitätscharakteristiken, wie Anzahl der Ein- und Ausgangssignale, Anzahl der Bausteine bzw. Blöcke eines Funktionsplans / einer Netzliste, Anzahl der Verbindungen sowie vor- und nachgelagerte Funktionspläne / Netzlisten berücksichtigt.

Aus den verschiedenen Komplexitätscharakteristiken wird ein Komplexitätsvektor gebildet, womit der Variabilitäts-Komplexität digitaler Leittechnik-Systeme Rechnung getragen wird. Der Komplexitätsvektor ist Ausgangspunkt für die Ableitung von Kriterien zur Bewertung der Zuverlässigkeit digitaler Leittechniksysteme.

Für die automatisierte Ermittlung der Komplexitätsmerkmale und der Komponenten des Komplexitätsvektors wurden prototypische Skriptprogramme entwickelt. Mithilfe dieser Werkzeuge lassen sich sowohl Funktionspläne (CPU-basierte Anwendung) als auch Netzlisten (FPGA-basierte Anwendungen) auswerten, womit die Automatisierbarkeit des Messverfahrens nachgewiesen wurde. Die damit gewonnenen Ergebnisse fließen direkt in die Messung der Komplexität ein und können anschließend zur Ableitung von Kriterien zur Bewertung der Zuverlässigkeit des jeweiligen digitalen Leittechniksystems herangezogen werden.

Für die Anwendung des Werkzeugs auf programmierbare Logik lassen sich aus dem standardisierten EDIF-Netzlistenformat alle notwendigen Komplexitätscharakteristiken für die

Ermittlung der Verflechtungsmaße direkt gewinnen, sodass ein Zwischenschritt über eine graphische Darstellung der Verschaltung nicht mehr notwendig ist. Dies ermöglicht eine automatisierte Bestimmung der Verflechtungsmaße für eine nahezu beliebige Anzahl von FPGAs, sofern deren Netzliste im EDIF-Dateiformat vorliegt. Auf Grund der Standardisierung ist die Netzliste unabhängig von Hersteller und Typ des jeweiligen FPGA-Chips.

Die praktische Anwendbarkeit des Messverfahrens auf mit graphischen Spezifikationen generierten Software bzw. FPGA-Strukturen wurde durch die Anwendung auf Leittechnik-Systeme verschiedener Hersteller nachgewiesen. Dabei wurde auch die Aussagekraft des Verfahrens anhand der Gegenüberstellung der Komplexitätsvektoren ähnlicher Anwendungen aus der Kerntechnik dargelegt. Hierbei wurden sowohl zwei CPU-basierte Steuerungen als auch die FPGA-basierten Steuerungen zweier Hersteller gegenübergestellt.

Für die Ableitung von Kriterien zur Bewertung der Zuverlässigkeit programmierbarer digitaler Leittechniksysteme ist eine Bewertung mittels Bayesian-Belief-Netze (BBN) grundsätzlich möglich. Für die Bewertung der Zuverlässigkeit CPU-basierter und FPGA-basierter Anwendungen sind dazu jedoch unterschiedliche BBNs erforderlich.

Im Vorhaben wurde deutlich, dass der Zugang zu den für die Ermittlung der Komplexität notwendigen Daten der digitalen Sicherheitsleittechnik mit großen Schwierigkeiten behaftet ist. Für die Ableitung von statistisch belastbaren Kriterien zur Bewertung der Zuverlässigkeit digitaler Leittechniksysteme ist die Erhebung von größeren Datenmengen erforderlich. Kriterien zur Bewertung der Zuverlässigkeit wurden deshalb qualitativ formuliert.

Das in diesem Vorhaben erzielte Ergebnis umfasst eine automatisierte Methodik zur Ermittlung der Komplexität von softwarebasierter Sicherheitsleittechnik und bietet damit die Möglichkeit verschiedene Softwaredesigns anhand ihrer Komplexität vergleichend zu bewerten. Dies ermöglicht sowohl Herstellern solcher Systeme Anforderungen nach Einfachheit von Sicherheitsleittechnik zu erfüllen als auch Prüfern diese Systemeigenschaft objektiv zu bewerten.

11 REFERENZEN

- [1] März, J., Miedl, H., Lindner, A., Gerst, C., Komplexitätsmessung der Software digitaler Leittechnik-Systeme
ISTec-A-1569
ISTec Garching, 2010
- [2] März, J., Fachberatung zur Um- und Nachrüstung der Sicherheitsleittechnik in deutschen Kernkraftwerken Sicherheitstechnische Bewertung zum Einsatz und Betrieb rechnergestützter Sicherheitsleittechnik in deutschen Kernkraftwerken AP 3: Analyse und Bewertung der Komplexität der Software und software-basierter Sicherheitsleittechnik
ISTec-A-1311
ISTec Garching, 2010
- [3] Hellie, F., Lindner, A., Mölleken, A., Komplexität und Fehlerpotenzial bei softwarebasierter digitaler Leittechnik, Teilbericht
ISTec-A-2891, Juni 2015
- [4] Smidts, C., Li, M., Brill, R. W., From Measures to Reliability, International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2000). Washington. DC. November. 2000.
- [5] Yastrebenetsky, M., Ed., Nuclear Power Plant Instrumentation and Control Systems for Safety and Security, Hershey, Pennsylvania, IGI Global, [2014]
- [6] VDI/VDE Richtlinie Anforderungen an Serienprodukte und Kriterien für deren Einsatz in der Sicherheitsleittechnik in Kernkraftwerken, Allgemeiner Teil, VDI/VDE 3528 Blatt 1, August 2011
- [7] 1061-1992 – IEEE Standard for a software Quality Metrics Methodology
- [8] Pressel, M., FPGAs and HPC
Army Research Laboratory
ARL-SR147
January 2007
- [9] Septián, J., Mozos, D., Mecha, H., Tabero J. and García de Dios, M. A., Perimeter Quadrature-based Metric for Estimating FPGA Fragmentation in 2D HW Multi-tasking, IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008
- [10] Herzner, W., et.al., Comparing Software measures with Fault Counts Derived from Unit-Testing of Safety Critical Software, SAFECOMP 2005, pp. 81-93

-
- [11] Karchenko, V., S., Sklyar, V., V., (Ed.) FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment, Kirovograd – Kharkiv – 2008.
- [12] APEX II – Programmable Logic Device Family - Data Sheet, Altera, August 2002, ver. 3.0
- [13] Dependability Assessment of Software for Safety Instrumentation and Control Systems at Nuclear Power Plants, International Atomic Energy Agency, 28.02.2017, Draft V4.21
- [14] Heigl, R., Lindner, A., Complexity and Error Potential of Digital I&C“, „10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC & HMIT 2017), San Francisco, USA, June 11 – 15, 2017
- [15] Begriffe-Sammlung
KTA-GS-12
Stand: Juni 2016
- [16] KTA 3501 Reaktorschutzsystem und Überwachungseinrichtungen des Sicherheitssystems, Fassung 2015-11
- [17] Sicherheitsanforderungen an Kernkraftwerke vom 3. März 2015 (BANz AT 30.03.2015 B2)
- [18] DIN EN 62566 (VDE 0491-3-5) Kernkraftwerke – Leittechnik für Systeme mit sicherheitstechnischer Bedeutung – Entwicklung HDL-programmierter integrierter Schaltkreise für Systeme, die Funktionen der Kategorie A ausführen (IEC 62566:2012); Deutsche Fassung EN 62566:2014

Anhang A: Teilbericht

Das Dokument ist in der PDF-Datei verlinkt.



TÜV Rheinland ISTec GmbH
Institut für
Sicherheitstechnologie

KOMPLEXITÄT UND FEH- LERPOTENTIAL BEI SOFT- WAREBASIERTER DIGITA- LER SICHERHEITSTECH- NIK

Teilbericht

F. Hellie
A. Lindner
A. Mülleken

ISTec - A - 2891

Rev. 4
Juli 2017

Das diesem Bericht zugrundeliegende F&E-Vorhaben wurde im Auftrag des Bundesministeriums für Umwelt, Naturschutz, Bau und Reaktorsicherheit unter dem Kennzeichen 3614RD1310 durchgeführt. Der Bericht gibt die Auffassung und Meinung des Auftragnehmers wieder und muss nicht mit der Meinung der Auftraggeberin übereinstimmen.

TÜV Rheinland ISTec GmbH - Postfach 12 13 - 85740 Garching - Telefon +49 89 5151466-0 - Telefax +49 89 5151466-20

Anhang B: Detaillierte Beschreibung des Skriptprogramms zur Berechnung der Verflechtungen (vfb)

Schnittstellendefinition

Nicht in allen Fällen steht eine Datenbank der Funktionspläne zur Verfügung. Deshalb ist es gegebenenfalls erforderlich, Daten für die Analyse / Bewertung der Verschaltung der Funktionspläne händisch aus den grafischen Darstellungen der Funktionspläne abzuleiten. Aus diesem Grund wurde für das Interface zwischen einem „Extractor für I&C Plattform“ und „Analyse / Bewertung Verschaltung (FP Ebene)“ eine universelle Zwischenschicht in folgender Weise definiert:

- 1) Eingangssignale werden in der Form „eSignalname“ benannt (Bsp.: eSignal1)
- 2) Ausgangssignale werden in der Form „aSignalname“ benannt (Bsp.: aSignal1)
- 3) Funktionsbausteine werden in der Form „fBausteinname“ benannt (Bsp.: fB1)
- 4) Connector zur Verbindung von Funktionsplanteile werden in der Form „cConnectorname“ benannt (Bsp.: cFP3.2)
- 5) Verbindungen werden in der Form „Signalquelle->Signalsenke“ benannt.
- 6) Die Zeichenfolge „->“, Leerzeichen, Tabulatoren und das Zeichen „#“ dürfen in den Signal-, Funktionsbaustein und Connectornamen nicht verwendet werden.

Connectoren werden mit dem Skriptprogramm zur Auflösung von internen Verbindungen in Funktionsplänen durch die realen Verbindungen ersetzt. Steht eine Datenbank bzw. eine maschinenlesbare Datei im EDIF-Netzlistenformat mit den Informationen zu den zu untersuchenden Funktionsplänen zur Verfügung, so ist ein Werkzeug erforderlich, das die Datenbankinhalte/EDIF-Netzlisten auf die oben definierte Zwischenschicht abbildet.

Das folgende Beispiel demonstriert den Zusammenhang zwischen grafischer Funktionsplandarstellung und der Repräsentation des Funktionsplans auf der Ebene der Zwischenschicht.

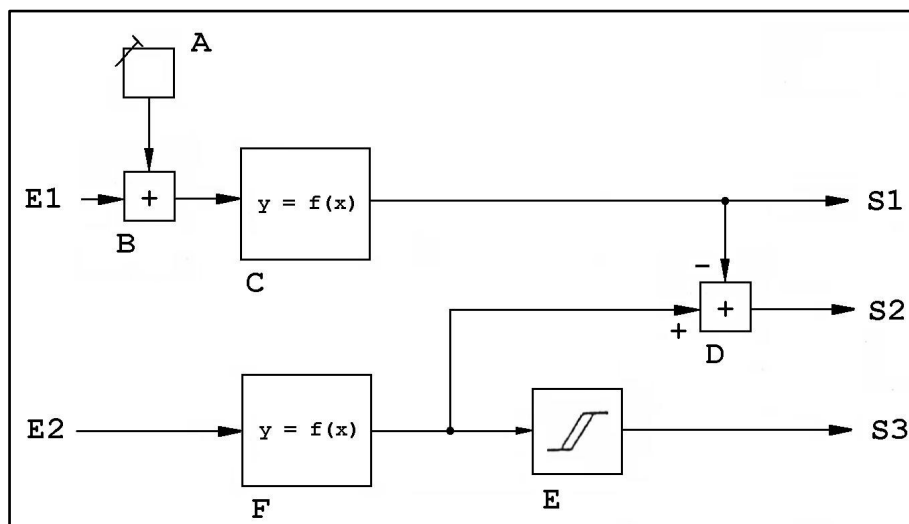


Bild B.1: Funktionsplan [2]

Die Zugehörige Beschreibung auf der Zwischenschicht ist in Bild B.2 dargestellt.

```
eE1 -> fB
fA -> fB
fB -> fC
fC -> aS1
fC -> fD
eE1 -> fF
fF->fE
fF->fD
fD->aS2
fE->aS3
```

Bild B.2: Schnittstellenbeschreibung des Funktionsplans aus Bild B.1

Diese Beschreibung ist auch für Netzlisten von FPGAs und PLDs anwendbar. An die Stelle der Funktionsbausteine des Funktionsplans treten hier die Basisblöcke des FPGAs bzw. des PLDs.

Wie aus Bild B.2 ersichtlich können zur Formatierung Leerzeichen und Tabulatoren in beliebiger Anzahl verwendet werden. Damit wird eine gute Lesbarkeit der Datei gesichert.

Es wurde ein skriptbasiertes Werkzeug implementiert, das sowohl einzelne Funktionspläne (Einzelplanbetrieb) als auch einen Satz von Dateien (Batchbetrieb), die in einem Format entsprechend Bild B.2 vorliegen, verarbeiten kann.

Skriptprogramm zur Berechnung der Verflechtungen

Im Rahmen des Projekts „Komplexitätsmessung der Software digitaler Leittechnik-Systeme“ wurde ein Gerüst für die benötigte Werkzeugumgebung zur Ermittlung der Komplexitätswerte für eine Plattform entwickelt [1].

Für Speicherprogrammierbare Steuerungen werden zunächst die Verbindungen aller Bausteine aus der graphischen Darstellung eines Funktionsplans der CPU-basierte Steuerung manuell in einer Textdatei aufgelistet.

Die Verbindungen werden in einer Textdatei anhand von einfachen Regeln eingetragen, sodass das Werkzeug diese auswerten kann. Eingänge werden mit Anfangsbuchstaben „e“ aufgelistet, Ausgänge mit „a“ und Funktionsbausteine mit „f“. Die Verbindung eines Bausteins zu einem anderen wird mit „->“ dargestellt, weshalb keine dieser Zeichen in der ansonsten freien Namensgebung vorkommen darf. Die Textdatei kann z.B. das in Bild B.2 dargestellte Format haben.

Das Werkzeug wertet diese Textdatei anschließend aus und ermittelt die Anzahl der Vorbereiche und Funktionsbausteine, um das Verflechtungsmaß gemäß 3.1.3 bzw. 3.1.4 zu berechnen. Für FPGAs wird die Netzliste meist durch eine Datei im EDIF (Electronic Design Interchange Format) beschrieben. Mit Hilfe dieses Dateiformats lassen sich alle Verbindungen zwischen den einzelnen Bausteinen ermitteln. Diese Verbindungen können wiederum gesondert in einer Textdatei aufgelistet werden und die Verflechtungsmaße analog zu den Funktionsplänen einer CPU-basierte Steuerung ermittelt werden. Im Gegensatz zum manuellen Eintragen der Verbindungen eines Funktionsplans kann die Netzliste, welche durch ein standardisiertes EDIF Dateiformat beschrieben ist, automatisiert ausgewertet werden, um die Verbindungen aller Bausteine aufzulisten.

Das Werkzeug ist in der Lage, die Verflechtungsmaße von nahezu beliebig umfangreichen Funktionsplänen und Netzlisten zu bestimmen und diese auf Zusammenhang, Schleifen und Fehler zu prüfen. Im Batchbetrieb können von dem Werkzeug auch mehrere Funktionspläne und Netzlisten automatisiert bearbeitet werden, was z.B. die Auswertung einer kompletten Anwendung oder Funktion ermöglicht.

Die Funktionsplanbeschreibung darf keine Konnektoren beinhalten. Diese sind gegebenenfalls mit dem Skriptprogramm zur Auflösung interner Verbindungen aufzulösen.

Nutzerinterface

Das Nutzerinterface für Einzelplanberechnung ist in Bild B.3 dargestellt.

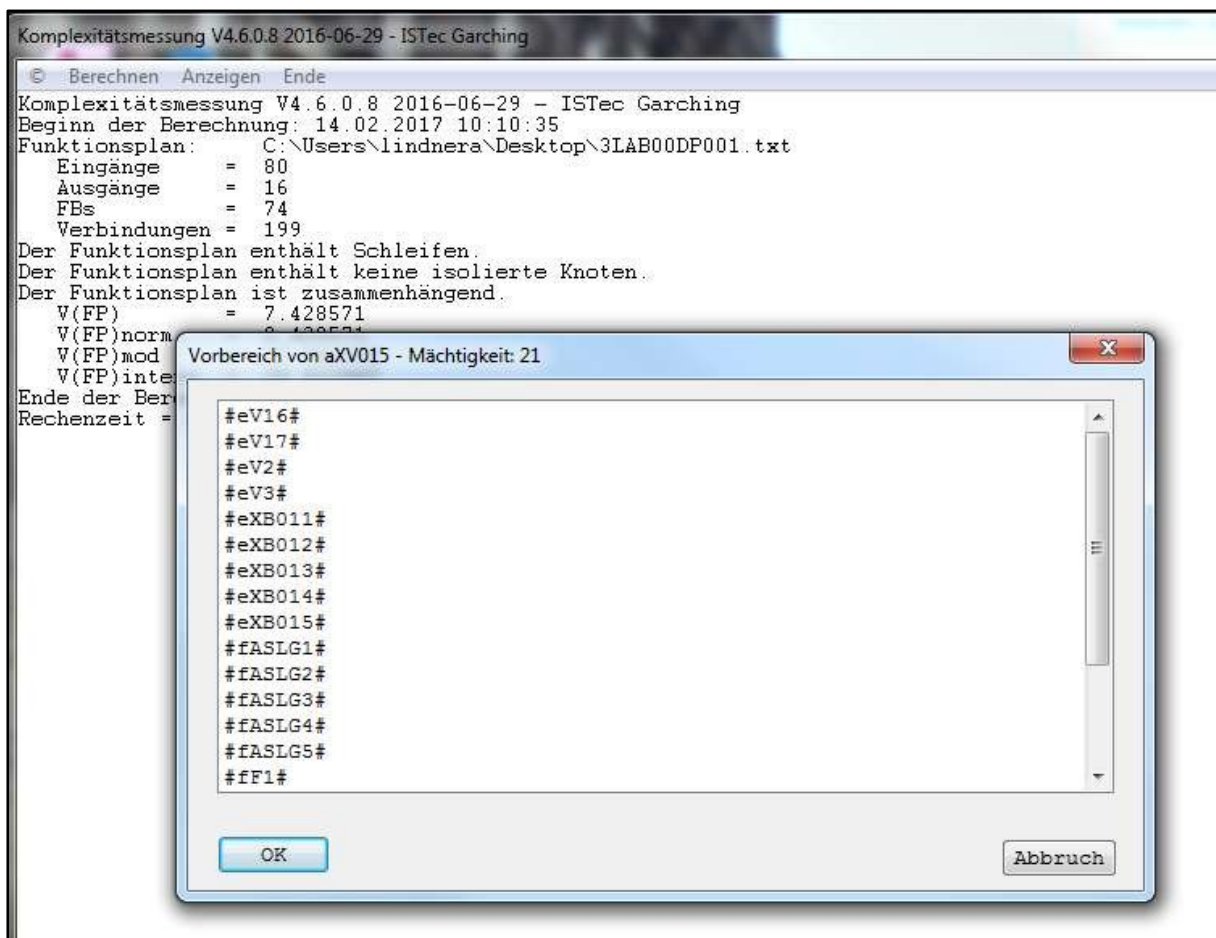


Bild B.3: Nutzerinterface des Werkzeugs zur Berechnung der Verflechtung

Das Skriptprogramm arbeitet Menügesteuert. Folgende Menüpunkte stehen zur Verfügung:

- Berechnen
 - Einzelplan
Es wird ein einzelner Funktionsplan ausgewertet (Ergebnis wie in Bild B.3 dargestellt).
 - Batchbetrieb mit löschen von Schleifen
Es wird eine Folge von Funktionsplänen abgearbeitet. Die Funktionsplandateien müssen in der Batchdatei aufgelistete sein. Im Ergebnis wird eine CSV-Datei erstellt,

- die alle Berechnungsergebnisse beinhaltet. Diese kann mit MS-Excel geöffnet und weiterbearbeitet werden.
- Einlesen von Ergebnissen
Es können Ergebnisdateien vorausgegangener Berechnungen eingelesen werden und die Ergebnisse können mit dem Menüpunkt „Anzeigen“ angezeigt werden. Das ist insbesondere zur Anzeige von Vorbereichen einzelner Funktionsbausteine und von Ausgangssignalen bei Funktionsplänen mit einer langen Auswertzeit hilfreich.
 - Anzeigen
 - Eingangssignale
Die Menge der Eingangssignale wird in einem Fenster dargestellt. Funktionsbausteine, die wie Eingangssignale verwendet werden, werden dabei nicht mit dargestellt. Diese sind im Ergebnisbildschirm sichtbar (siehe Bild B.3).
 - Funktionsbausteine
Die Menge der Funktionsbausteine wird in einem Fenster dargestellt.
 - Ausgangssignale
Die Menge der Ausgangssignale wird in einem Fenster dargestellt.
 - Funktionsplan
Der Funktionsplan wird in einem Fenster dargestellt.
 - Vorbereiche
Es werden alle Funktionsbausteine und Ausgangssignale in einem Auswahlfenster dargestellt. Nach Auswahl eines Funktionsbausteins oder eines Ausgangssignals wird der Vorbereich in einem Fenster dargestellt. Die Mächtigkeit des Vorbereichs wird in der Titelzeile des Fensters angegeben (siehe Bild B.3).
 - Ergebnisdatei
Dieser Menüpunkt ist nur im Anschluss an die Abarbeitung einer Batch-Datei anwählbar. Es wird MS-Excel unter Angabe der CSV-Datei gestartet
 - Log-Datei
Die während der Berechnung (Einzelplan oder Batchbetrieb) angelegte Log-Datei wird mit dem windowseigenen Editor angezeigt.
 - Ende
Das Programm lässt sich nur über diesen Menüpunkt beenden.

Ergebnis- und Log-Dateien

Für die Ergebnis- und Log-Dateien werden Dateinamen automatisch nach folgenden Regeln erzeugt:

- Ergebnisdatei Einzelplanberechnung
Name_der_Funktionsplandatei.result.txt
- Log-Datei Einzelplanberechnung
Name_der_Funktionsplandatei.log.txt
- Ergebnisdatei Batchbetrieb
Frei wählbarer Name. Die Datei sollte den Typ .csv haben, damit sie mit MS-Excel di-

rekt geöffnet werden kann.

Zusätzlich zur Ergebnisdatei des Batchbetriebs werden die Ergebnisdatei und die Log-Datei für die einzelnen Funktionspläne gemäß der oben beschriebenen Namenskonvention erzeugt.

- Log-Datei Batchbetrieb
Name_der_Ergebnisdatei_Batchbetrieb.logfile.txt

Fehlermeldungen und Warnungen werden in den Log-Dateien gespeichert. Falls ein Funktionsplan schwerwiegende Fehler aufweist, so dass ein berechnetes Ergebnis unbrauchbar wäre, wird die Berechnung der Verflechtungen abgebrochen.

Falls bereits Ergebnisdateien mit dem gleichen Namen existieren, werden diese nach Nachfrage überschrieben. Log-Dateien werden fortlaufend weitergeschrieben.

Aufbau der Ergebnisdatei im Batchbetrieb:

Die Ergebnisdatei im Batchbetrieb enthält folgende Daten:

- Kopfinformationen (siehe Beispiel)
- FP
- Anzahl Inputsignale
- Anzahl FBs
- Anzahl Output-signale
- Anzahl Verbindungen
- $V(\text{FP})$
- $V(\text{FP})_{\text{normiert}}$
- $V(\text{FP})_{\text{modifiziert}}$ $V(\text{FP})_{\text{intern}}$
- Isolierte Knoten
- Zusammenhang

In der Spalte „Zusammenhang“ wird angezeigt, ob der Funktionsplan einen zusammenhängenden Graphen repräsentiert oder ob der Funktionsplan in unabhängige Einzelpläne zerlegt werden kann. Im zweiten Fall ist das Berechnungsergebnis nicht korrekt, da das Verflechtungsmaß nur für zusammenhängende Funktionspläne definiert ist.

- Schleifen
In der Spalte Schleifen wird angezeigt, ob der Funktionsplan Signalverbindungen der Art „fb1->fb1“ (d.h. eine Aufschaltung eines Signalausgangs eines Funktionsbausteines auf einen Signaleingang des gleichen Bausteins) beinhaltet.
- Rechenzeit [ms]

Beispiele**Ergebnisdatei Einzelplanberechnung (fehlerfreier Plan, Ausschnitte):**

```
14.02.2017 10:10:35
C:\Users\lindnera\Desktop\3LAB00DP001.txt
80
16
74
199
1
0
1
7.428571
0.428571
17.941558
10.512987
0
14.02.2017 10:09:52
32
Knotenmenge
#eXC121#
#fO1#
...
#aXV016#
Kantenmenge
#eXC121->fO1#
#eXB021->fO1#
...
#fF13->aXV016#
Eingänge
#eXC121#
#eXB021#
...
#eV12#
Ausgänge
#aXC931#
#aXQ011#
...
#aXV016#
FBs
#fO1#
#fO2#
..
#fOSPCb#
Matrix
1 2
1 14
...
167 168
```

Die Datei enthält 3324 Zeilen.

Zugehörige Log-Datei:

Dienstag, der 14. Februar 2017 - 10:10

Komplexitätsmessung V4.6.0.8 2016-06-29 - ISTec Garching

Beginn der Berechnung: 14.02.2017 10:10:35

Funktionsplan: C:\Users\lindnera\Desktop\3LAB00DP001.txt

Eingänge = 80

Ausgänge = 16

FBs = 74

Verbindungen = 199

Der Funktionsplan enthält Schleifen.

Der Funktionsplan enthält keine isolierte Knoten.

Der Funktionsplan ist zusammenhängend.

V(FP) = 7.428571

V(FP)norm = 0.428571

V(FP)mod = 17.941558

V(FP)intern = 10.512987

Ende der Berechnung: 14.02.2017 10:11:15

Rechenzeit = 40279 ms

Ergebnisdatei Batchbetrieb:

```
Komplexitätsmessung V4.6.0.8 2016-06-29 - ISTec Garching
Funktionsplanliste: C:\Users\lindnera\Desktop\Validierung\Val_Batchbetrieb.txt
FP;Anzahl Inputs;Anzahl FBs;Anzahl Outputs;Anzahl Verbindungen;V(FP);V(FP)normiert;V(FP)modifiziert;V(FP)intern; Iso-
lierte Knoten;Zusammenhang;Schleifen;Rechenzeit [ms]
Val_Testplan_01.txt;1;99;1;100;1,000000;0,000000;50,500000;49,500000;nein;ja;nein;33727
Val_Testplan_02.txt;1;10;1;11;1,000000;0,000000;5,545455;4,545455;nein;ja;nein;3682
Val_Testplan_03.txt;1;999;1;1000;1,000000;0,000000;500,500000;499,500000;nein;ja;nein;480012
Val_Testplan_04.txt;Funktionsbaustein ist Ausgangssignal;;;;;;;;;;;;;16
Val_Testplan_05.txt;Funktionsbaustein ist Ausgangssignal;;;;;;;;;;;;;15
Val_Testplan_06.txt;1;9;1;11;1,000000;0,000000;5,500000;4,500000;nein;ja;ja;3682
Val_Testplan_07.txt;1;8;1;10;1,000000;0,000000;8,111111;7,111111;nein;ja;nein;3604
Val_Testplan_07a.txt;1;8;1;10;1,000000;0,000000;8,111111;7,111111;nein;ja;nein;3261
Val_Testplan_07b.txt;1;9;1;12;1,000000;0,000000;9,100000;8,100000;nein;ja;nein;3712
Val_Testplan_08.txt;Ausgang als Quellknoten;;;;;;;;;;;;;0
Val_Testplan_09a.txt;3;36;6;60;6,000000;1,000000;41,076923;35,076923;nein;ja;nein;82758
Val_Testplan_09b.txt;3;36;6;60;6,000000;1,000000;41,076923;35,076923;nein;ja;nein;81058
Val_Testplan_09c.txt;1;8;4;14;4,000000;1,000000;11,111111;7,111111;nein;ja;nein;3806
Val_Testplan_09d.txt;1;8;4;14;4,000000;1,000000;11,111111;7,111111;nein;ja;nein;4540
Val_Testplan_10.txt;Eingang als Zielknoten;;;;;;;;;;;;;15
Val_Testplan_11.txt;0;8;1;9;1,000000;0,000000;8,000000;7,000000;nein;ja;nein;2917
Val_Testplan_11a.txt;Keine Ausgaenge im Funktionsplan;;;;;;;;;;;;;16
Val_Testplan_12.txt;Funktionsbaustein ist Ausgangssignal;;;;;;;;;;;;;16
Val_Testplan_13a.txt;99;1;1;100;1,000000;0,000000;1,990000;0,990000;nein;ja;nein;17737
Val_Testplan_13b.txt;Ausgang wird mehrfach angesteuert;;;;;;;;;;;;;78
Val_Testplan_13c.txt;99;99;99;198;1,000000;0,000000;1,500000;0,500000;nein;nein;nein;253500
Val_Testplan_13d.txt;99;1;99;198;99,000000;1,000000;99,990000;0,990000;nein;ja;nein;102430
Val_Testplan_13e.txt;1;1;99;100;99,000000;1,000000;99,500000;0,500000;nein;ja;nein;32603
Val_Testplan_13f.txt;Ausgang wird mehrfach angesteuert;;;;;;;;;;;;;78
Val_Testplan_14a.txt;Ausgang wird mehrfach angesteuert;;;;;;;;;;;;;0
Val_Testplan_14b.txt;1;0;99;99;99,000000;1,000000;99,000000;0,000000;nein;ja;nein;31848
```

Zugehörige Log-Datei:

```
-----  
Donnerstag, der 9. Februar 2017 - 12:46  
-----  
Komplexitätsmessung V4.6.0.8 2016-06-29 - ISTec Garching  
Beginn 09.02.2017 12:46:36  
Funktionsplanliste: C:\Users\lindnera\Desktop\Validierung\Val_Batchbetrieb.txt  
Ergebnisdatei: C:\Users\lindnera\Desktop\Validierung\ergebnis_2017_02_09.csv  
bearbeiten Val_Testplan_01.txt  
bearbeiten Val_Testplan_02.txt  
#f10# ist Eingangssignal  
bearbeiten Val_Testplan_03.txt  
bearbeiten Val_Testplan_04.txt  
#f5# ist Ausgangssignal  
#f6# ist Eingangssignal  
bearbeiten Val_Testplan_05.txt  
#f6# ist Ausgangssignal  
#f5# ist Eingangssignal  
bearbeiten Val_Testplan_06.txt  
bearbeiten Val_Testplan_07.txt  
bearbeiten Val_Testplan_07a.txt  
bearbeiten Val_Testplan_07b.txt  
bearbeiten Val_Testplan_08.txt  
bearbeiten Val_Testplan_09a.txt  
bearbeiten Val_Testplan_09b.txt  
bearbeiten Val_Testplan_09c.txt  
bearbeiten Val_Testplan_09d.txt  
bearbeiten Val_Testplan_10.txt  
bearbeiten Val_Testplan_11.txt  
bearbeiten Val_Testplan_11a.txt  
bearbeiten Val_Testplan_12.txt  
#f9# ist Ausgangssignal  
bearbeiten Val_Testplan_13a.txt  
bearbeiten Val_Testplan_13b.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_13c.txt  
bearbeiten Val_Testplan_13d.txt  
bearbeiten Val_Testplan_13e.txt  
bearbeiten Val_Testplan_13f.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_14a.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_14b.txt  
Fertig 09.02.2017 13:05:41
```

Ergebnis Batchbetrieb als Excel-Blatt mit grafischer Aufbereitung der Ergebnisse

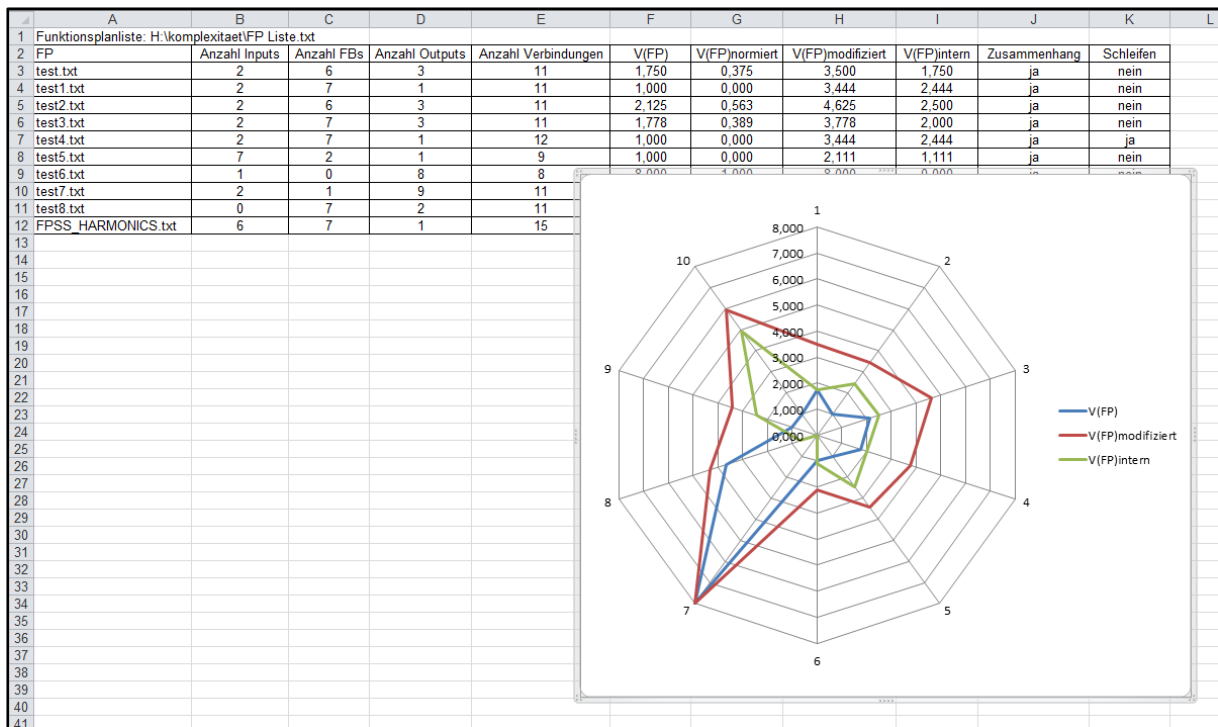


Bild B.4: Ergebnis Batchbetrieb mit grafischer Aufbereitung

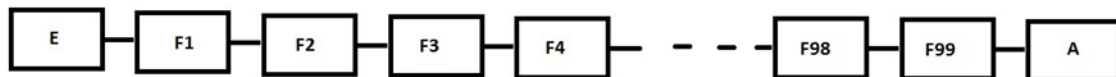
Anhang C: Validierung des Tools zur Berechnung des Verflechtungsmaßes

ALLGEMEINE ANMERKUNGEN

- Die aktuelle Version des Tools ist V 4.6.0.8. Die Laufzeit für die Validierung beträgt ca. 20 Min.
- Als „Schleifen“ werden direkte Rückkopplungen über einen einzelnen Funktionsbaustein bezeichnet (siehe Testfall 6). Im Gegensatz dazu sind „Kreise“ Rückkopplungen über mehrere Funktionsbausteine (siehe Testfall 7).
- Doppelt eingetragene Verbindungen werden vom Tool erkannt und korrekterweise nicht berücksichtigt. Der Nachweis hierfür wird nicht gesondert dargelegt.
- Fehlerhafte Bezeichnungen von Bausteinen und unzulässige Zeichenfolgen in der Textdatei werden vom Tool erkannt. Der Nachweis hierfür wird nicht gesondert dargelegt.
- Isolierte Knoten (Bausteine, die ohne Verbindung in die Textdatei eingefügt sind) werden vom Tool erkannt.
- Obwohl grundsätzlich keine Obergrenze für den Umfang eines Funktionsplans nötig ist, wurde für das Skript eine maximale Anzahl an Knoten im Funktionsplan als Abbruchkriterium festgelegt, um die Berechnungszeit des Skripts im Falle eines fehlerhaften oder unangemessen umfangreichen Funktionsplans einzudämmen. Der Nachweis dieses Abbruchkriteriums wird nicht gesondert dargelegt.
- Ist der eingetragene Testplan nicht zusammenhängend oder enthält eine Schleife wird dadurch das Ergebnis der Berechnung eventuell verfälscht. In diesem Zusammenhang wird dies jedoch nicht als Abweichung gewertet.
- Eine Fehlermeldung wird in der Validierung dahingehend geprüft, dass der korrekte Fehler gemeldet wird.

AUSWERTUNG DER TESTFÄLLE

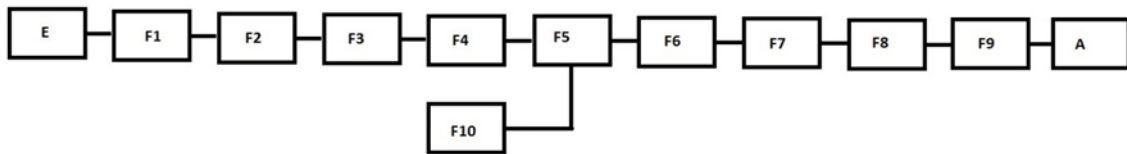
Testfall 1



Ziel: Prüfung der Berechnung des Verflechtungsmaßes an einem einfachen Beispiel.

Ergebnis: Erwartetes Ergebnis erzielt

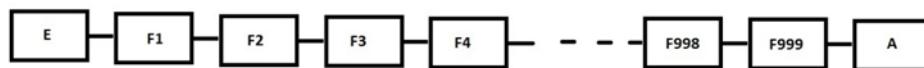
Anmerkung: keine.

Testfall 2

Ziel: Prüfung der Berechnung des Verflechtungsmaßes an einem einfachen Beispiel.

Ergebnis: Erwartetes Ergebnis erzielt

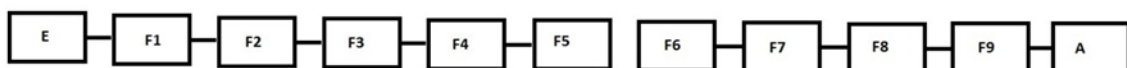
Anmerkung: keine

Testfall 3

Ziel: Prüfung der Berechnung des Verflechtungsmaßes an einem Beispiel mit einer sehr hohen Anzahl an Bausteinen bzw. Verbindungen

Ergebnis: Erwartetes Ergebnis erzielt

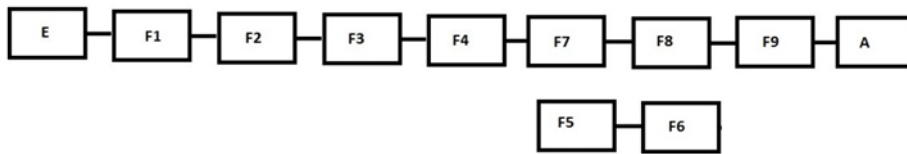
Anmerkung: Es gibt eine festgelegte Grenze, die eine Maximale Anzahl an Bausteinen im Funktionsplan zulässt.

Testfall 4

Ziel: Prüfung der Erkennung des Nicht-Zusammenhangs.

Ergebnis: Erwartetes Ergebnis erzielt

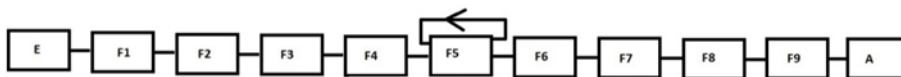
Anmerkung: Es wird folgende Fehlermeldung erzeugt „Funktionsbaustein ist Ausgangssignal“, dies ist ein Abbruchkriterium. Der Zusammenhang wird nicht mehr geprüft.

Testfall 5

Ziel: Prüfung der Erkennung des Zusammenhangs.

Ergebnis: Erwartetes Ergebnis erzielt

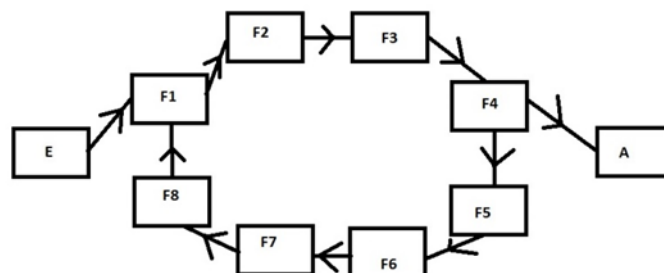
Anmerkung: Es wird folgende Fehlermeldung erzeugt „Funktionsbaustein ist Ausgangssignal“, dies ist ein Abbruchkriterium. Der Zusammenhang wird nicht mehr geprüft.

Testfall 6

Ziel: Prüfung der Erkennung einer Schleife.

Ergebnis: Erwartetes Ergebnis erzielt.

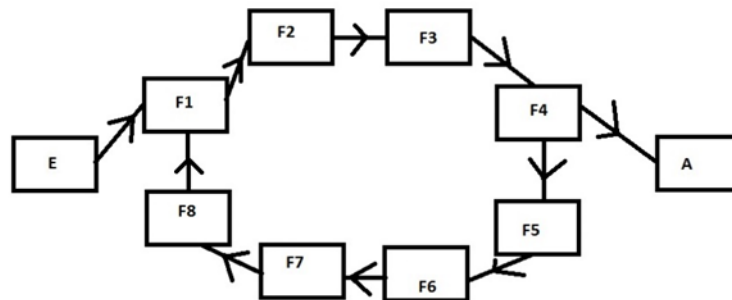
Anmerkung: Die Schleife wird gemeldet. Um die Berechnung zu ermöglichen wird die Schleife korrekterweise nicht berücksichtigt.

Testfall 7**Testfall 7 „richtige“ Reihenfolge**

Ziel: Prüfung der Berechnung anhand eines Kreises.

Ergebnis: Erwartetes Ergebnis erzielt

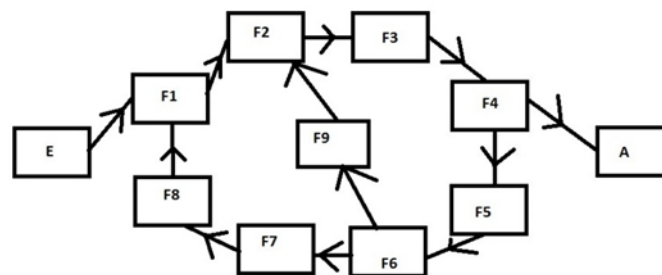
Anmerkung: Jeder Funktionsblock ist korrekterweise im Vorbereich der anderen Funktionsblöcke. Kein Funktionsblock ist als sein eigener Vorbereich deklariert.

Testfall 7a geänderte Reihenfolge

Ziel: Prüfung der Berechnung anhand geänderter Reihenfolge der Verbindungen in der Textdatei.

Ergebnis: Erwartetes Ergebnis erzielt

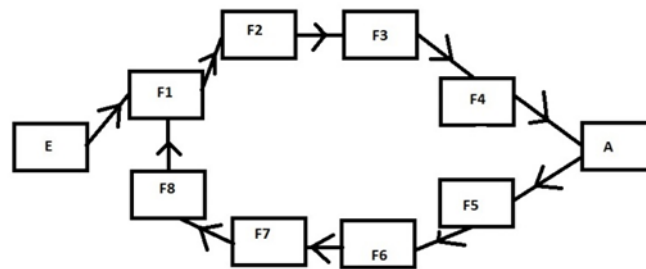
Anmerkung: Trotz geänderter Reihenfolge bleibt das beobachtete Ergebnis unverändert.

Testfall 7b geänderte Reihenfolge mit zusätzlicher Querverbindung

Ziel: Prüfung der Berechnung anhand geänderter Reihenfolge der Verbindungen in der Textdatei.

Ergebnis: Erwartetes Ergebnis erzielt

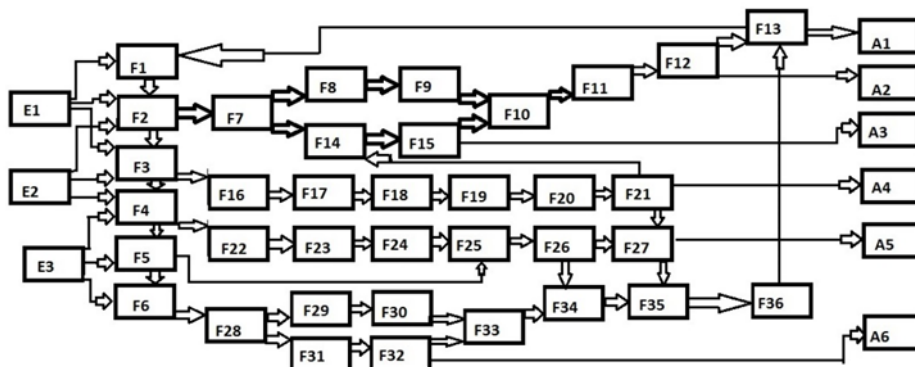
Anmerkung: keine.

Testfall 8

Ziel: Prüfung der Erkennung eines fehlerhaften Ausgangs.

Ergebnis: Erwartetes Ergebnis nicht erzielt.

Anmerkung: Es wird folgende Fehlermeldung erzeugt „Ausgang als Quellknoten“, dies ist ein Abbruchkriterium.

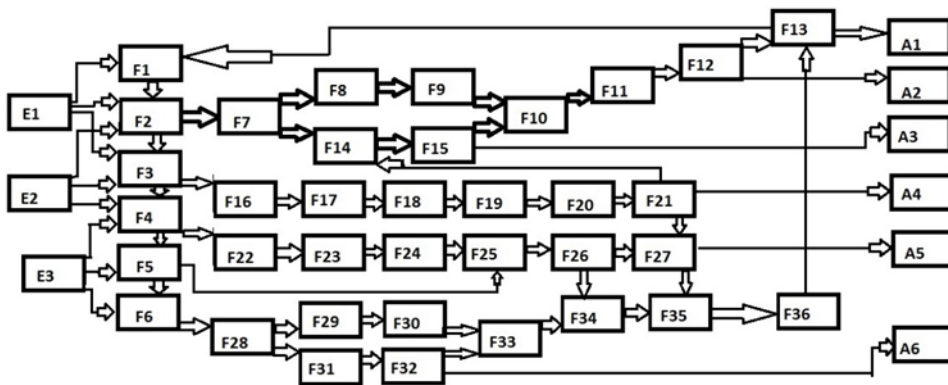
Testfall 9**Testfall 9a „richtige“ Reihenfolge**

Ziel: Prüfung der Berechnung anhand mehrerer Kreise.

Ergebnis: Erwartetes Ergebnis erzielt

Anmerkung: keine.

Testfall 9b geänderte Reihenfolge

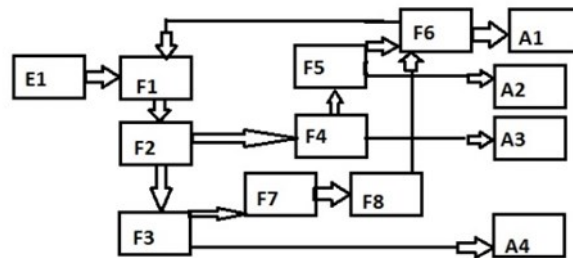


Ziel: Prüfung der Berechnung anhand geänderter Reihenfolge der Verbindungen in der Textdatei.

Ergebnis: Erwartetes Ergebnis erzielt

Anmerkung: Trotzdem die Verbindungen in einer völlig willkürlichen Reihenfolge eingetragen sind und der Funktionsplan über mehrere Kreise verfügt, ist das beobachtete Ergebnis unverändert.

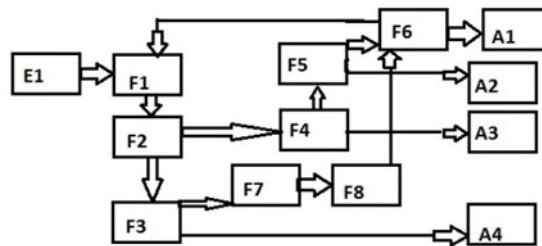
Testfall 9c Vereinfachter Testplan



Ziel: Prüfung der Berechnung anhand eines Kreises.

Ergebnis: Erwartetes Ergebnis erzielt

Anmerkung: keine.

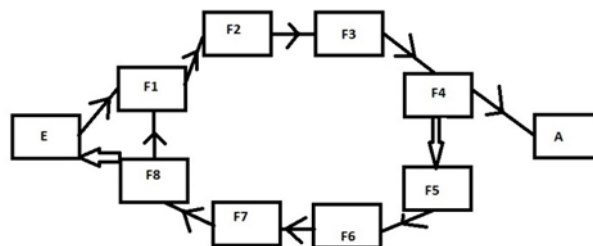
Testfall 9d Vereinfachter Testplan mit geänderter Reihenfolge

Ziel: Prüfung der Berechnung anhand eines Kreises.

Ergebnis: Prüfung der Berechnung anhand geänderter Reihenfolge der Verbindungen in der Textdatei.

Ergebnis: Erwartetes Ergebnis erzielt

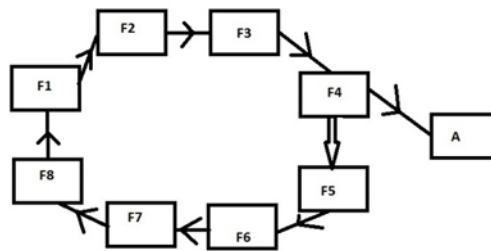
Anmerkung: keine.

Testfall 10

Ziel: Prüfung der Erkennung eines fehlerhaften Eingangs.

Ergebnis: Erwartetes Ergebnis erzielt.

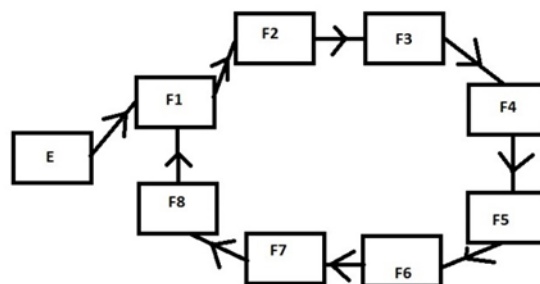
Anmerkung: Es wird folgende Fehlermeldung erzeugt „Eingang als Zielknoten“, dies ist ein Abbruchkriterium.

Testfall 11**Testfall 11 fehlender Eingang**

Ziel: Prüfung der Berechnung trotz fehlenden Eingangs.

Ergebnis: Erwartetes Ergebnis erzielt.

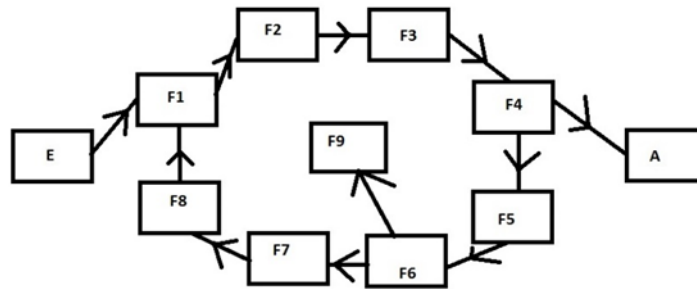
Anmerkung: Die Berechnung ist korrekt. Die Anzahl der Eingänge beträgt 0.

Testfall 11a fehlender Ausgang

Ziel: Prüfung der Erkennung eines fehlenden Ausgangs.

Ergebnis: Erwartetes Ergebnis erzielt.

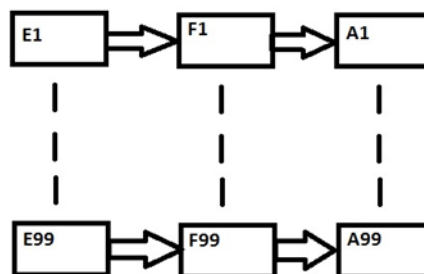
Anmerkung: Es wird folgende Fehlermeldung erzeugt „Keine Ausgänge im Funktionsplan“, dies ist ein Abbruchkriterium.

Testfall 12

Ziel: Prüfung der Erkennung eines fehlerhaften Funktionsblocks.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Es wird folgende Fehlermeldung erzeugt „Funktionsbaustein ist Ausgangssignal“, dies ist ein Abbruchkriterium.

Testfall 13**Testfall 13a: 99 Eingänge auf 1 Funktionsblock und 1 Ausgang**

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Die Berechnung ist korrekt. Der Funktionsplan wird richtigerweise als zusammenhängend deklariert.

Testfall 13b: 99 Eingänge auf 99 Funktionsblöcke und 1 Ausgang

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen und einem fehlerhaften Ausgang der mehrfach angesteuert wird.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Es wird folgende Fehlermeldung erzeugt „Ausgang wird mehrfach angesteuert“, dies ist ein Abbruchkriterium.

Testfall 13c: 99 Eingänge auf 99 Funktionsblöcke und 99 Ausgänge

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Die Berechnung ist korrekt. Der Funktionsplan wird richtigerweise als nicht zusammenhängend deklariert.

Testfall 13d: 99 Eingänge auf 1 Funktionsblock und 99 Ausgänge

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Die Berechnung ist korrekt. Der Funktionsplan wird richtigerweise als zusammenhängend deklariert.

Testfall 13e: 1 Eingang auf 1 Funktionsblock und 99 Ausgänge

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen.

Ergebnis: Erwartetes Ergebnis erzielt.

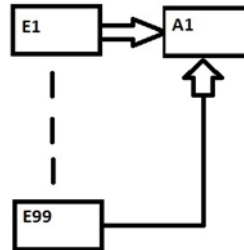
Anmerkung: Die Berechnung ist korrekt. Der Funktionsplan wird richtigerweise als zusammenhängend deklariert.

Testfall 13f: 1 Eingang auf 99 Funktionsblöcke und 1 Ausgang

Ziel: Prüfung der Berechnung an einfachen Beispielen mit einer Vielzahl an Bausteinen in verschiedenen Permutationen und einem fehlerhaften Ausgang der mehrfach angesteuert wird.

Ergebnis: Erwartetes Ergebnis erzielt.

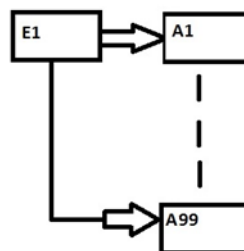
Anmerkung: Es wird folgende Fehlermeldung erzeugt „Ausgang wird mehrfach angesteuert“, dies ist ein Abbruchkriterium.

Testfall 14**Testfall 14a fehlende Funktionsbausteine**

Ziel: Prüfung der Berechnung bei fehlenden Funktionsbausteinen und einem fehlerhaften Ausgang, der mehrfach angesteuert wird.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Es wird folgende Fehlermeldung erzeugt „Ausgang wird mehrfach angesteuert“, dies ist ein Abbruchkriterium.

Testfall 14b fehlende Funktionsbausteine

Ziel: Prüfung der Berechnung bei fehlenden Funktionsbausteinen.

Ergebnis: Erwartetes Ergebnis erzielt.

Anmerkung: Das Ergebnis der Berechnung ist korrekt. Die Anzahl der Funktionsbausteine ist 0.

ABGLEICH DES BEOBACHTETEN ERGEBNISSES MIT DEM ERWARTETEN ERGEBNIS

Die aufgelisteten Testfälle werden von dem Tool im Batchbetrieb verarbeitet und die beobachteten Ergebnisse anschließend mit den erwarteten Ergebnissen abgeglichen. Bei einer Abweichung der erwarteten von den beobachteten Ergebnissen ist das entsprechende Feld in der Tabelle rot hinterlegt.

FP	Anzahl Inputs	Anzahl FBs	Anzahl Outputs	Anzahl Verbindungen	V(FP)	V(FP) _n ormiert	V(FP) _{modifiziert}	V(FP) _{intern}	Isolierte Knoten	Zusammenhang	Schleifen
Val_Testplan_01.txt	1	99	1	100	1	0	50,50000	49,5	nein	ja	nein
Val_Testplan_02.txt	1	10	1	11	1	0	5,545455	4,545455	nein	ja	nein
Val_Testplan_03.txt	1	999	1	1000	1	0	500,50000	499,5	nein	ja	nein
Val_Testplan_04.txt	Funktionsbaustein ist Ausgangssignal										
Val_Testplan_05.txt	Funktionsbaustein ist Ausgangssignal										
Val_Testplan_06.txt	1	9	1	11	1	0	5,50000	4,5	nein	ja	ja
Val_Testplan_07.txt	1	8	1	10	1	0	8,11111	7,111111	nein	ja	nein
Val_Testplan_07a.txt	1	8	1	10	1	0	8,11111	7,111111	nein	ja	nein

FP	Anzahl Inputs	Anzahl FBs	Anzahl Outputs	Anzahl Verbindungen	V(FP)	V(FP) _n ormiert	V(FP) _{modifiziert}	V(FP) _{intern}	Isolierte Knoten	Zusammenhang	Schleifen
Val_Testplan_07b.txt	1	9	1	12	1	0	9,10000	8,1	nein	ja	nein
Val_Testplan_08.txt	Ausgang als Quellknoten										
Val_Testplan_09a.txt	3	36	6	60	6	1	41,07692	35,076923	nein	ja	nein
Val_Testplan_09b.txt	3	36	6	60	6	1	41,07692	35,076923	nein	ja	nein
Val_Testplan_09c.txt	1	8	4	14	4	1	11,11111	7,111111	nein	ja	nein
Val_Testplan_09d.txt	1	8	4	14	4	1	11,11111	7,111111	nein	ja	nein
Val_Testplan_10.txt	Eingang als Zielknoten										
Val_Testplan_11.txt	0	8	1	9	1	0	8,00000	7	nein	ja	nein
Val_Testplan_11a.txt	Keine Ausgaenge im Funktionsplan										
Val_Testplan_12.txt	Funktionsbaustein ist Ausgangssignal										
Val_Testplan_13a.txt	99	1	1	100	1	0	1,99000	0,99	nein	ja	nein

FP	Anzahl Inputs	Anzahl FBs	Anzahl Outputs	Anzahl Verbindungen	V(FP)	V(FP) _{normiert}	V(FP) _{modifiziert}	V(FP) _{intern}	Isolierte Knoten	Zusammenhang	Schleifen
Val_Testplan_13b.txt	Ausgang wird mehrfach angesteuert										
Val_Testplan_13c.txt	99	99	99	198	1	0	1,50000	0,5	nein	nein	nein
Val_Testplan_13d.txt	99	1	99	198	99	1	99,99000	0,99	nein	ja	nein
Val_Testplan_13e.txt	1	1	99	100	99	1	99,50000	0,5	nein	ja	nein
Val_Testplan_13f.txt	Ausgang wird mehrfach angesteuert										
Val_Testplan_14a.txt	Ausgang wird mehrfach angesteuert										
Val_Testplan_14b.txt	1	0	99	99	99	1	99,00000	0	nein	ja	nein

Es stimmen alle beobachteten Ergebnisse einschließlich der generierten Fehlermeldungen mit den erwarteten Ergebnissen überein.

Log-Datei des Validierungslaufs

```
-----  
Donnerstag, der 9. Februar 2017 - 12:46  
-----  
Komplexitätsmessung V4.6.0.8 2016-06-29 - ISTec Garching  
Beginn 09.02.2017 12:46:36  
Funktionsplanliste:  
C:\Users\lindnera\Desktop\Validierung\Val_Batchbetrieb.txt  
Ergebnisdatei:  
C:\Users\lindnera\Desktop\Validierung\ergebnis_2017_02_09.csv  
bearbeiten Val_Testplan_01.txt  
bearbeiten Val_Testplan_02.txt  
#f10# ist Eingangssignal  
bearbeiten Val_Testplan_03.txt  
bearbeiten Val_Testplan_04.txt  
#f5# ist Ausgangssignal  
#f6# ist Eingangssignal  
bearbeiten Val_Testplan_05.txt  
#f6# ist Ausgangssignal  
#f5# ist Eingangssignal  
bearbeiten Val_Testplan_06.txt  
bearbeiten Val_Testplan_07.txt  
bearbeiten Val_Testplan_07a.txt  
bearbeiten Val_Testplan_07b.txt  
bearbeiten Val_Testplan_08.txt  
bearbeiten Val_Testplan_09a.txt  
bearbeiten Val_Testplan_09b.txt  
bearbeiten Val_Testplan_09c.txt  
bearbeiten Val_Testplan_09d.txt  
bearbeiten Val_Testplan_10.txt  
bearbeiten Val_Testplan_11.txt  
bearbeiten Val_Testplan_11a.txt  
bearbeiten Val_Testplan_12.txt  
#f9# ist Ausgangssignal  
bearbeiten Val_Testplan_13a.txt  
bearbeiten Val_Testplan_13b.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_13c.txt  
bearbeiten Val_Testplan_13d.txt  
bearbeiten Val_Testplan_13e.txt  
bearbeiten Val_Testplan_13f.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_14a.txt  
#a1# wird mehrfach angesteuert  
bearbeiten Val_Testplan_14b.txt  
Fertig 09.02.2017 13:05:41
```

ZUSAMMENFASSUNG

Folgende Funktionalitäten wurden validiert:

- Identifikation isolierter Konten.
- Identifikation fehlerhaft verwendeter Funktionsbausteine,
- Identifikation mehrfach angesteuerter Ausgänge.
- Identifikation von Schleifen.
- Ausgabe der Anzahl von Bausteinen und Verbindungen.

-
- Unabhängigkeit der Berechnungen von der Reihenfolge der Notation der Verbindungen.
 - Berechnung der Verflechtungsmaße für nahezu beliebig umfangreiche Funktionspläne unabhängig von der Reihenfolge des Eintrags der Verbindungen in der Textdatei, für Funktionspläne, die Schleifen und Kreise enthalten und Funktionspläne die keine Eingänge aufweisen.
 - Ermittlung des Zusammenhangs allgemein, aber insbesondere unabhängig von der Reihenfolge des Eintrags der Verbindungen in der Textdatei, für Funktionspläne, die Kreise enthalten.

Anhang D: Vortrag NPIC&HMIT**COMPLEXITY AND ERROR POTENTIAL OF DIGITAL I&C****R. J. Heigl and A. Lindner**

TÜV Rheinland ISTec GmbH – Institut für Sicherheitstechnologie
Boltzmannstraße 14
85748 Garching near Munich, Germany
Raimund.Heigl@de.tuv.com; Arndt.Lindner@de.tuv.com

ABSTRACT

Utilization of digital I&C for application in nuclear power plants led to a growing discussion on how to create methods for a probabilistic evaluation on such systems. Issues arise on the one hand due to a lack of commonly accepted models for quantitative evaluation of availability and reliability and on the other hand due to the integration of such models into the evaluation of reliability of an overall system. Nevertheless programmable and even more complex systems are implemented to satisfy growing functionality requirements. Therefore the question for a method to evaluate the reliability of systems based on software or on programmable logic became even more urgent. The approach described is based on the commonly accepted assumption that more complex designs are less reliable.

Key Words: complexity, PLC function diagram, function block, FPGA netlist, interconnection

1 INTRODUCTION

The average age of NPPs currently operated in Europe exceeds 20 years. Thus the need for backfitting and modernization will inescapably grow in the next future. Most of the installed I&C systems are based on analogue technologies such as electromagnetic relays and analogue electronic modules. The technical solutions available on the market mainly rely on digital technologies, applying microcontrollers or Field Programmable Gate Arrays (FPGA) [9]. Those technologies still raise many technical and procedural issues. One of them is the quantification of software reliability.

Since the time that computer-based systems have been used in other safety relevant applications (e.g. Aviation, Telecommunication) there is an increasing discussion concerning the probabilistic safety assessment of software. The problems are due to the lack of widely accepted models for determining the reliability of software systems.

As known from practical experience it is the complexity of a software-system together with the high variety of its operational profile, which determines the hazard of the software to fail, i.e. its dependability. Therefore, prediction of software dependability, based on complexity measurement is one of the promising approaches for quantifying the reliability of software [7].

In order to facilitate this approach for digital I&C systems applied in the nuclear field, research effort was spent to identify and quantify complexity of digital I&C systems and its correlation to reliability [1-3]. The approach introduced in previous projects was modified in order to enhance significance and also to apply the complexity measurement to a wide range of programmable I&C systems including Programmable Logic Controller (PLC) and FPGA technology. The focus of this paper is directed to the application of the methodology to different technologies [8], [10], [11].

2 CONCEPT FOR COMPLEXITY MEASUREMENT

The concept was introduced in [1], [2], and [4] and can now be applied to any generic, graphic-based, digital I&C platforms (e.g. TELEPERM XS, COMMON Q, TRICON CX, RadICS). The application software of I&C systems based on such platforms is represented as logic diagram (LD) of elementary functions based on either PLC or FPGA technology.

The elementary functions can be classified as:

- logic or arithmetic functions such as OR, AND, ADD etc.,
- basic I&C functions e.g. for implementing a comparison or an interpolation curve,
- specific functions such as ramp generator or sorter.

The elementary functions represented by elementary blocks in the graphical representation will further on be called function blocks in case of PLC logic and basic blocks in case of FPGA logic. They are implemented as modules of a library. They are defined as the lowest level of programming items in current digital I&C systems. Large and complex I&C functions can be designed and generated by combination of these elementary blocks.

The functionality of the I&C system is implemented by logic diagrams of either function blocks or basic blocks. The logic diagram is represented as function diagram in case of PLC logic or as netlists in case of FPGA logic.

The complexity measurement of the application software is subdivided into two levels, according to the structure of the logic diagram modules implementing the I&C functionality. These levels are

- Measurement of the elementary blocks,
- Measurement of the logic diagram.

The complexity of the different blocks is the first level for determining the overall complexity of an I&C function.

There is a limited and fixed set of elementary blocks available to implement the logic diagrams. Therefore a meaningful procedure for complexity measurement of an I&C system should generate a matrix describing the complexity features of all the blocks which are available. This complexity matrix is independent of any specific application and of any logic diagram.

On the second level a complexity metric for the interconnection of elementary blocks of logic diagrams has to be defined and evaluated. The overall complexity of a digital I&C system will be quantified by the complexity of the application specific logic diagrams taking into account the complexity matrix generated on the first level for elementary blocks.

3 COMPLEXITY MATRIX

The function blocks represent a set of software functions, which are implemented in some standardized high level language such as ANSI-C, ADA, etc.

For these software functions implementing the function blocks a complexity matrix was generated. The complexity matrix comprises all typical function blocks and refers to the relevant aspects describing the complexity of the function blocks. These aspects are compiled using a White-Box-View based on the code and a Black-Box-View based on documentation and user-manuals [2], [4].

The complexity characteristics obtained are put together in a complexity matrix. Examples for details of a complexity matrix are given in two separate tables (table I. and table II.) for reasons of clarity.

Table I. Excerpt of a Complexity Matrix (signals, parameter)

		signal					parameter				
		inputs		outputs			not change- able	change- able	sum	condi- tions	de- rived
Fan Out		bi-	ana-	bi-	ana-	re-					

		nary	log	nary	log	port					
FB01 PAR	1	0	0	0	1	0	2	1	3	0	0
FB02 ADD	1	0	3	0	1	0	0	0	0	0	0
FB03 CURVE	2	0	1	0	1	0	1	2N+1	2N+2	N+2	0
FB04 COMP	1	0	3	1	0	0	1	4	5	2	6

Table II. Excerpt of a Complexity Matrix (variables, resources)

	int. var.	memory	memory usage		runtime [µs]			error-codes	function	
			code (bytes)	stack (bytes)	init.	param.	calc.		text	picture table
FB01 PAR	0	0	125	24	13	5	5	1	1	0
FB02 ADD	2	0	276	34	19	10	10	2	4	0
FB03 CURVE	8	0	592	48	101	92	92	2	7	0
FB04 COMP	2	1	503	32	28	19	12	1	11	0

The examples are taken from a PLC based platform.

4 METHOD FOR MEASURING THE COMPLEXITY OF LOGIC DIAGRAMS

Logic diagrams are implementing the I&C functions. They are established by interconnection of elementary functions.

The features describing the complexity of a logic diagram can be subdivided into three groups:

- Basic counting metrics, that can be taken directly from the graphic representation of the logic diagram namely number of input and output signals, the number of elementary blocks and number of interconnections,
- Metrics to describe the complexity of the interconnection of the elementary blocks
- Variability features of digital I&C systems that have an impact on complexity, such as number of internal memories and changeable parameters

It is in the nature of the logic diagrams that individual elementary blocks or even sets of individual elementary blocks are often used for computing more than only one output signal. This overlapping interconnection of function blocks for the computation of various output signals is the normal case for logic diagrams. Another essential impact on complexity is introduced by the usage of common input signals.

A metric $V(LD)$ for these two types of interconnection within a logic diagram is given by

$$V(LD) = \sum_{S_{ai}} \frac{|VB(S_{ai})| + |IN(S_{ai})|}{|BLD| + |SIN|} \tag{1}$$

Where S_{ai} indicates all the individual output signals of the LD

$VB(S_{ai})$	indicates the set of elementary blocks that are involved in the computation of the output signal S_{ai}
$IN(S_{ai})$	indicates the set of input signals that are involved in the computation of the output signal S_{ai}
BLD	indicates the set of elementary blocks making up the LD
SIN	indicates the set of input signals of the LD
	indicates cardinality (number of elements of a set)

This metric basically is an average value reflecting the complexity regarding the output signals only. For LD with only one output signal the value of $V(LD)$ is always 1, no matter how many input signals are used for computing the output signal and how the signal is processed via elementary blocks.

Therefore a metric $V(LD)_{mod}$, for the modified interconnection complexity is introduced in [5]. It is given by

$$V(LD)_{mod} = \sum_{S_i} \frac{|VB(S_i)| + |IN(S_i)|}{|BLD| + |SIN|} \quad (2)$$

Where	S_i	indicates all the individual signals (internal and output) of the LD
	$VB(S_i)$	indicates the set of elementary blocks that are involved in the computation of the signal S_i
	$IN(S_i)$	indicates the set of input signals that are involved in the computation of the signal S_i

Determining the difference

$$V(LD)_{mod} - V(LD) = V(LD)_{int} \quad (3)$$

yields a metric for the internal interconnection complexity $V(LD)_{int}$, since $V(LD)_{mod}$ considers the interconnection of all signals (of elementary blocks and output signals) and $V(LD)$ only takes into account the interconnection of signals contributing to output signals. That modification is an essential extension of the methodology described in [4] and presented at previous meetings [1-2]. The modified approach exhibits a more detailed view on the complexity characteristics of logic diagrams. It should be mentioned that the calculation of $V(LD)_{mod}$ requires long calculating times in case of large logic diagrams.

5 COMPLEXITY VECTOR

The characteristics identifying the complexity of logic diagrams are made up by simple volume measures, by the complexity of the interconnection and by variability characteristics of digital I&C systems.

These characteristics comprise a wide range of complexity properties. To combine these to a single number would hide all the detailed information gained during complexity analysis. Therefore, the complexity features of a logic diagram should be kept separated and represented in a complexity vector.

According to the previous considerations the components of the complexity vector are defined by

- K1: Number of elementary (function or basic) blocks
- K2: Number of inputs signals of a LD

- K3: Number of output signals of a LD
- K4: Number of interconnections between inputs, elementary blocks and outputs
- K5: Number of upstream FDs for the relevant FD (as part of an I&C function)
- K6: Number of downstream FDs for the relevant FD (as part of an I&C function)
- K7: Interconnection complexity $V(LD)$ for the interconnection of elementary blocks
- K8: Modified interconnection complexity $V(LD)_{mod}$
- K9: Internal interconnection complexity $V(LD)_{int}$

The components K1, K2, K3 and K4 can directly be taken from the graphical representation of the logic diagram or determined tool-based for large function diagrams e.g. for an FPGA netlist directly out of an Electronic Design Interchange Format (EDIF) file.

The components K5 and K6 are taken into account in case the relevant FD acts as a part of an I&C function, where the functionality is achieved via interconnection of several FDs. When the complexity vector of a whole I&C function or a whole system is determined, the value of both K5 and K6 will be zero.

The components K7, K8 and K9 can in principle be computed by hand, but for large function diagrams such as FPGA logic or the examples given in chapter 6, the metrics for the interconnection complexity were computed by a script [6].

6 COMPARISON OF TWO SIMILAR APPLICATIONS

The Complexity Vectors of two similar setups of PLC logic in nuclear application are compared. The two functions are developed for the same type of I&C system and constitute the same functionality but are used in different NPPs and have been developed by different teams under different provisions concerning the development of function diagrams. The two functions are simply called function 1 and function 2. The function diagrams compounded to the respective functions (see Figure 1) are called controls and engines in the following.

The functionality is achieved by a “control” and three “engines”. For function 1 redundant engines of the same type are used, while for function 2 three different engines are in place. The interconnections between the items are slightly different.

As opposed to function 1, function 2 was developed using specific macros built from the original function blocks. The approach applied to the function diagram design of function 2 can save time during function diagram development, but in turn may result in a utilization of more function blocks and interconnections as required for the intended function.

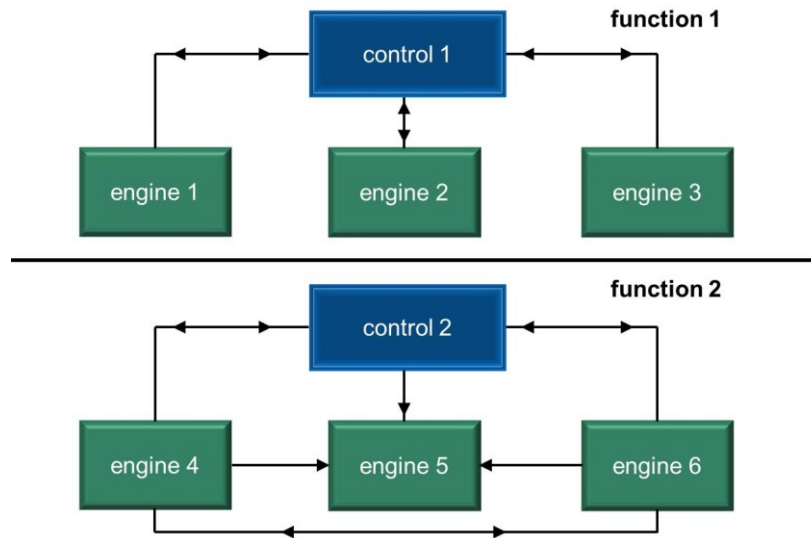


Figure 1. Comparison of two similar setups in nuclear application with the same functionality.

Comparison of the Complexity Vectors (see table III.) of the two controls shows, that almost all values of the complexity vector of control 2 are higher than those of control 1. The number of used function blocks is significantly higher for control 2. Additionally the value of the interconnection complexity $V(LD)$ is more than four times higher. Considering the determination of the interconnection complexity, where the number of function blocks appears in the denominator, the complexity of the function diagram of control 2 is identified as being more complex.

Table III. Comparison of Complexity Vectors for PLC logic

PLC	Definition	control 1	control 2	engine 1	engine 4	function 1	function 2
K1:	# function blocks	17	68	11	112	50	383
K2:	# inputs	14	21	18	17	53	45
K3:	# outputs	14	50	16	27	47	99
K4:	# connections	55	177	46	224	178	778
K5:	# upstream LD	3	2	2	3	0	0
K6:	# downstream LD	6	3	2	3	0	0
K7:	$V(LD)$	9	43	14	13	42	67
K8:	$V(LD)_{mod}$	16	72	20	68	68	282
K9:	$V(LD)_{int}$	7	29	6	55	26	215

The comparison of the Complexity Vectors of the engines draws a similar picture. The internal interconnection complexity is almost 10 times higher for engine 4 as compared to engine 1, while the interconnection complexity $V(LD)$ has a similar value. This is remarkable because the significantly higher number of used function blocks compensates the number of signals contributing to outputs for the calculation of $V(LD)$, but not for the calculation of the internal interconnection complexity

$V(LD)_{int}$. This demonstrates the higher complexity of the engines of function 2 compared to function 1.

Finally the Complexity Vector for each function, which consists of the control and the three engines, respectively, is compared. Almost all values of the complexity vector are higher for function 2. As a logical consequence of considering the previous results the values for number of function blocks, interconnections, modified interconnection complexity and internal interconnection complexity are significantly higher for function 2 as compared to function 1. Again despite a similar value for the interconnection complexity $V(LD)$, the value of the internal interconnection complexity is much higher for function 2. This clearly shows that the crucial difference between the two function diagram designs has an impact on the complexity of the interconnection of the function blocks of which the respective function diagram is composed of.

7 APPLICATION TO FPGA LOGIC

The method for measuring the complexity can also be used for FPGA logic. FPGA logic cells have a finer granularity than PLCs. As a result, for the same functionality, it takes more cells to implement a function in an FPGA than in a PLC [9]. The example used to show the implementation of the method was provided by [8]. The I&C function implemented by the FPGA logic is a hysteresis function for the reactor trip of a NPP and is illustrated in figure 2.

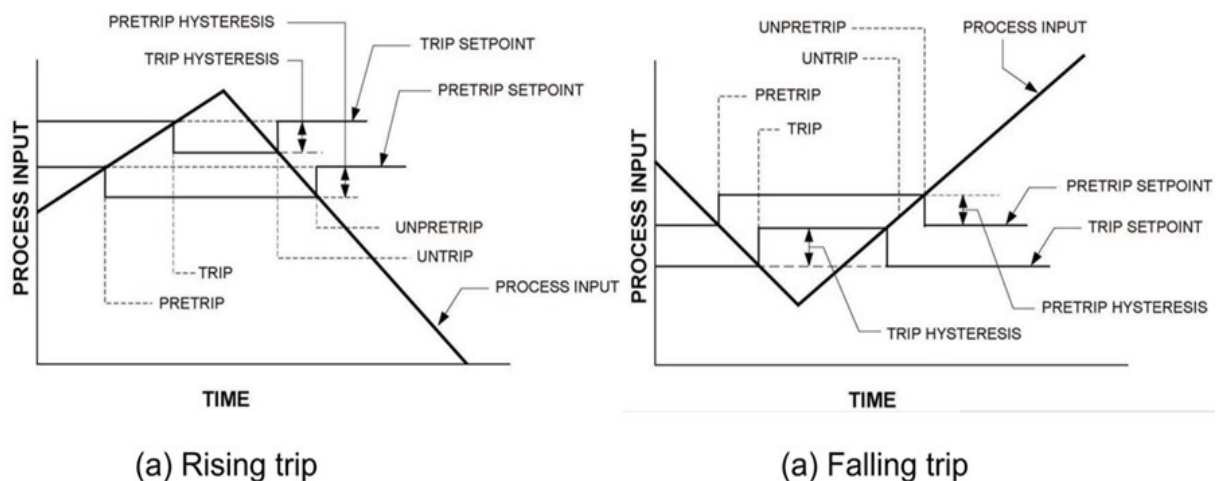
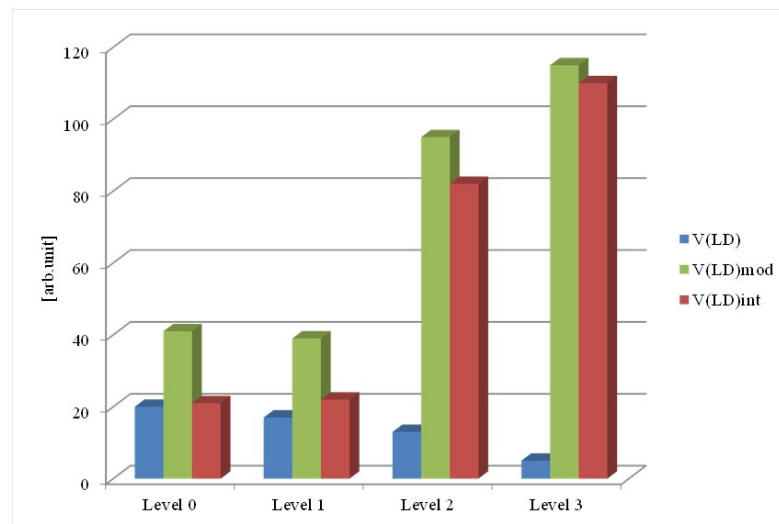


Figure 2. Example of a function implemented by FPGA logic in nuclear application [8].

FPGAs have segmented interconnect structures [9]. The FPGA logic is divided into several levels in the visual representation. In the selected example the netlist can be divided into four levels (0 to 3). The elementary functions are represented as cells on each level. With each higher level, a cell is split up into several sub-cells where the functionality of a mother cell is realized by interconnection of daughter-cells, which feature a more and more elemental functionality. On the level having the finest granularity (level 3 in the given example), the remaining cells are represented by basic blocks featuring the fundamental basic functions. In order to depict the changing functionality on each level the complexity vector is presented for each level of the visual representation of the FPGA logic in table IV and fig. 3. At the end the most relevant level concerning the complexity of the FPGA logic is the base-level complexity vector (level 3 in the given example).

Table IV. Complexity Vector of FPGA Application separated in levels

FPGA	Definition	Level 0	Level 1	Level 2	Level 3
K1:	# basic blocks	100	107	225	506
K2:	# inputs	75	75	75	75
K3:	# outputs	23	23	23	23
K4:	# connections	198	228	620	1227
K5:	# upstream LD	0	0	0	0
K6:	# downstream LD	0	0	0	0
K7:	$V(LD)$	20	17	13	5
K8:	$V(LD)_{mod}$	41	39	95	115
K9:	$V(LD)_{int}$	21	22	82	110

**Figure 3. Comparison of the interconnection complexity of different levels of the FPGA logic.**

The number of input and output signals is the same for each level of the visual representation, since only the cells are split up into sub-cells. Comparison of the complexity vectors of each level shows, that the interconnection complexity $V(LD)$ is descending with each higher level but the internal complexity $V(LD)_{int}$ is increasing. This behavior meets exactly one's expectations.

8 CONCLUSIONS

Modern digital I&C systems, programmed by graphic-based specification of its functionality have a common structure. Therefore a unitary two-level concept can be used for description and quantification of their complexity.

The lower level is defined by elementary functions, the function blocks or basic blocks. Their

complexity is calculated using a White-Box-View based on the code and a Black-Box-View based on documentation and user-manuals. The results of the White-Box-View and Black-Box-View are collected and represented in the corresponding complexity-matrices for the function blocks of a function diagram or the basic blocks of a FPGA design respectively.

The second level is defined in a graphical manner by the function diagrams or netlists. They represent the functionality of the I&C function and are represented by logic diagrams of the function blocks or the basic blocks. Their complexity is defined on basis of the signal-interconnection taken from their graphical representation of the LD in case of function diagrams or taken from a FPGA netlist. The quantification of the complexity features is compiled in a complexity-vector.

The quantification of complexity in two levels fits to the structure of digital I&C systems. It is rather descriptive and allows:

- identification of regions with raised complexity within a digital I&C design,
- comparison of complexity between I&C functions.

Additionally, the method developed for complexity measurement allows:

- identification of errors in the function diagrams or netlists respectively such as loops, isolated parts etc.,
- signal tracing e.g. of signals corresponding to a specific output and therefore supporting the generation of test cases.

The approach for complexity measurement was applied to two PLC-based functions, which have the same functionality but were designed with different methods for different NPPs. Comparison of the respective complexity vectors showed that the internal interconnection complexity is significantly different. The two compared PLC-based functions possess a different complexity as a result of a different function diagram design.

The presented method of complexity measurement is also applicable to FPGA logic, although further research is required to compare the complexity of different FPGA applications.

The extended approach now allows the analysis of a broad range of generic, graphic-based, digital I&C platforms for different technologies like PLC and FPGA logic and exhibits a more detailed view on the complexity characteristics of logic diagrams as compared to the method that has been described formerly.

9 ACKNOWLEDGMENTS

The work on complexity measurement and reliability of software in digital I&C systems is funded by the German Federal Ministry for the Environment, Nature Conservation, Building and Nuclear Safety (BMUB) under the project number 3614R01310 within its scope of research in reactor safety. Special thanks go to Prof. Junbeom Yoo for providing data of FPGA-based logic for controllers in NPPs.

1. REFERENCES

1. J. März, A. Lindner, H. Miedl, M. Baleanu “COMPLEXITY MEASUREMENT AND RELIABILITY OF SOFTWARE IN DIGITAL I&C-SYSTEMS”, *NPIC&HMIT 2004*, Columbus, Ohio, September, 2004.
2. J. März, A. Lindner, H. Miedl, “COMPLEXITY MEASUREMENT OF SOFTWARE IN DIGITAL I&C-SYSTEMS”, *NPIC&HMIT 2009*, Knoxville, Tennessee, April 5-9, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009).
3. J. Holmberg, S. Guerra, N. Thuy, J. März, B. Liwång, “HARMONICS-EU FP7 PROJECT ON THE RELIABILITY ASSESSMENT OF MODERN NUCLEAR I&C SOFTWARE”, *NPIC&HMIT 2012*, San Diego, CA, July 22-26, 2012.

4. J. März, H. Miedl, A. Lindner, C. Gerst, „KOMPLEXITÄTSMESSUNG DER SOFTWARE DIGITALER LEITTECHNIKSYSTEME“, ISTec-A-1569, February 2010.
5. F. Hellie, A. Lindner, A. Mölleken, „KOMPLEXITÄT UND FEHLERPOTENTIAL BEI SOFTWAREBASIERTER DIGITALER SICHERHEITSTECHNIK“, Interim Report, ISTec-A-2891, February 2016.
6. R. Heigl, F. Hellie, A. Lindner, A. Mölleken, „KOMPLEXITÄT UND FEHLERPOTENTIAL BEI SOFTWAREBASIERTER DIGITALER SICHERHEITSTECHNIK“, Final Report, (in preparation).
7. Smidts, C., „FROM MEASURES TO RELIABILITY“, *NPIC&HMIT 2000*, Washington D.C., November 13-16, 2000.
8. J. Yoo, E. Kim, D. Lee, J. Choi, „AN INTEGRATED SOFTWARE DEVELOPMENT FRAMEWORK FOR PLC & FPGA-BASED DIGITAL I&C“, *ISOFIG 2014*, Jechu, Rep. of Korea, August 24-28, 2014.
9. IAEA Nuclear Energy Series No. NP-T-3.17, APPLICATION OF FIELD PROGRAMMABLE GATE ARRAYS IN INSTRUMENTATION AND CONTROL SYSTEMS OF NUCLEAR POWER PLANTS, International Atomic Energy Agency, Vienna (2016).
10. J.-E. Holmberg, P. Bishop, S. Guerra, N. Thuy, „SAFETY CASE FRAMEWORK TO PROVIDE JUSTIFIABLE RELIABILITY NUMBERS FOR SOFTWARE SYSTEMS,“ *Proc. of 11th International Probabilistic Safety Assessment & Management Conference, PSAM 11*, Helsinki, June 25–29, 2012(2012).
11. S. Guerra, N. Thuy, „SAFETY JUSTIFICATION FRAMEWORKS: INTEGRATING RULE-BASED, GOAL-BASED, AND RISK-INFORMED APPROACHES,“ *Proc. of 8th International Conference on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC & HMIT)*, July 22-26, 2012, San Diego, CA(2012).

Anhang E: PowerPoint Folien zum Beitrag des ISTec zur NPIC


TÜV Rheinland ISTec GmbH – Institut für Sicherheitstechnologie 


Complexity and error potential of digital I&C

NPIC&HMIT 2017
June 13, 2017

R. Heigl / A. Lindner
TÜV Rheinland ISTec GmbH
- Institut für Sicherheitstechnologie -
Boltzmannstraße 14
85748 Garching
Germany



 TÜV Rheinland®
ISTec
Genau. Richtig.


TÜV Rheinland ISTec GmbH – Institut für Sicherheitstechnologie 

Motivation


“The assignment of the application functions to the systems shall attempt to minimize the complexity of class 1 systems.”

IEC 61513:2011

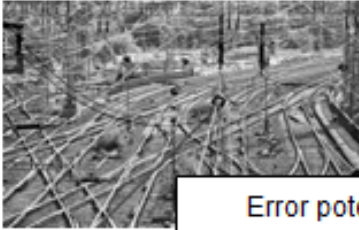
OPERATING A NUCLEAR PLANT MADE SIMPLE



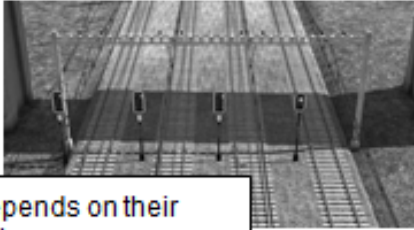
source: Licensing experience Reconstruction of NPP Unit Computer Systems, Károly Hamar, HAEA

 TÜV Rheinland®
ISTec
Genau. Richtig.

TÜV Rheinland ISTec GmbH – Institut für Sicherheitstechnologie **ISTec**



Motivation




Error potential of I&C Systems depends on their respective complexity

↓

Determination of complexity on the basis of functional diagrams (e.g. TXS)

↓

Amplification of the complexity measurement on FPGA- and PLC- technologies and modification of the procedure




TÜVRheinland®
ISTec
Genau. Richtig.

TÜV Rheinland ISTec GmbH – Institut für Sicherheitstechnologie **ISTec**

Complexity Vector

components of the complexity vector	
K1:	number of function blocks
K2:	number of input signals
K3:	number of output signals
K4:	number of interconnections
K5:	number of upstream function diagrams
K6:	number of downstream function diagrams
K7:	interconnection complexity $V(FD)$
K8:	modified interconnection complexity $V_{mod}(FD)$
K9:	internal interconnection complexity $V_{intern}(FD)$
K10:	number of changeable parameters
K11:	number of internal memories
K12:	complexity of the function blocks of the function plan

□ $V(FD)$:Determination of interconnection complexity



TÜVRheinland®
ISTec
Genau. Richtig.

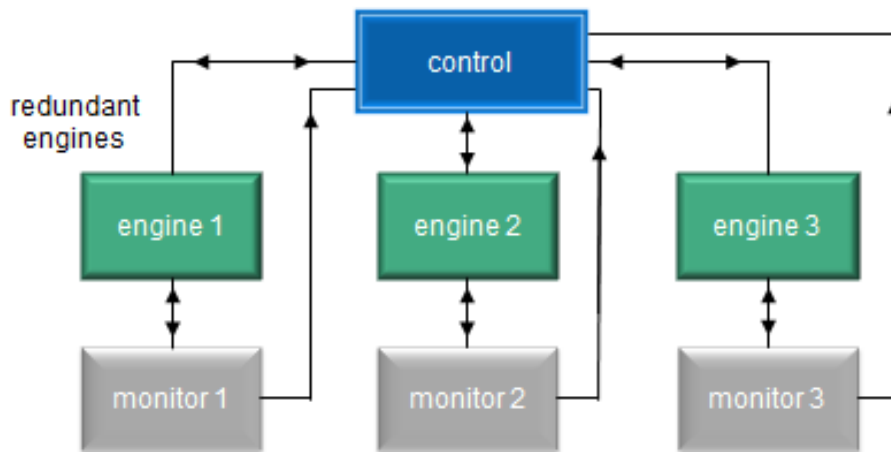
Determination of Complexity Vector

- either manually or tool-based determination (e.g. for large FDs) possible
- Number of I/O, FB, interconnections
- Interconnection complexity, „normal“, modified, internal
- Identification of discontinuity and loops
- Identification and localization of errors in the function plan, such as:
 - missing I/O
 - I/O acting like FB
 - FB acting like I/O
 - dead ends

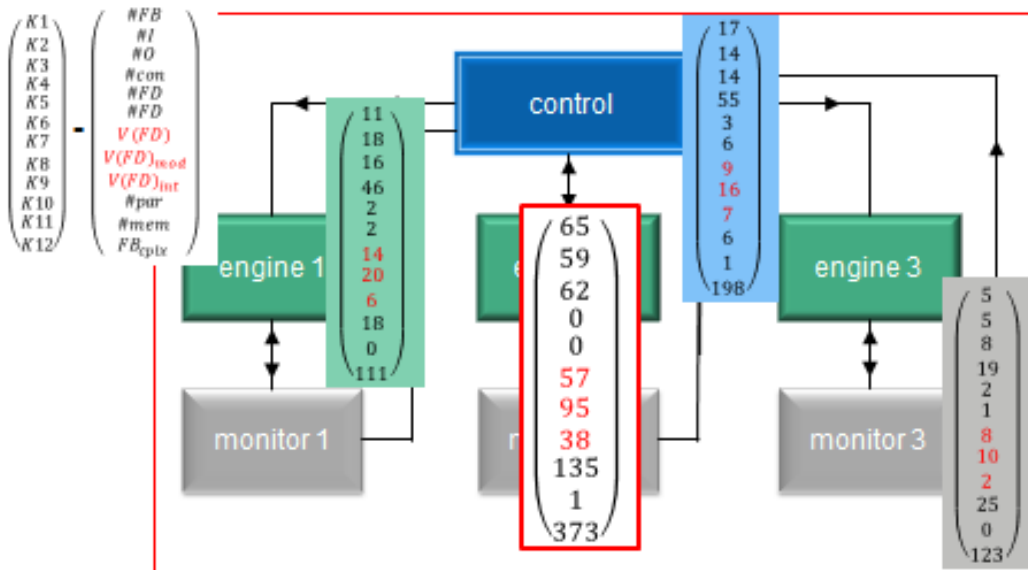
```

Komplexitätsmessung (neuer Algorithmus) V4.3.0.14 2016-03-11 - ISTec Garching
C Berechnen Anzeigen Ende
Komplexitätsmessung (neuer Algorithmus) V4.3.0.14 2016-03-11 - ISTec Garching
Dauer der Berechnung 22.03.2016 14:40:53
Funktionsplan C:\TIS-ISTec-Projekte\Abteilung 117\469014-11_Tooler\FPGA-FPGA_Ebese1(korrekt).tat
Einträge = 75
Ausgänge = 23
FDB = 157
Verbindungen = 228
Der Funktionsplan enthält keine Schleifen.
Der Funktionsplan enthält keine isolierte Knoten.
Der Funktionsplan ist zusammenhängend.
V(FP) = 14.94105
V(FP)act = 0.927623
V(FP)mod = 39.406593
V(FP)intern = 22.412088
Ende der Berechnung 22.03.2016 14:42:15
Rechenzeit = 78125 ms
    
```

Example on a nuclear PLC-based application

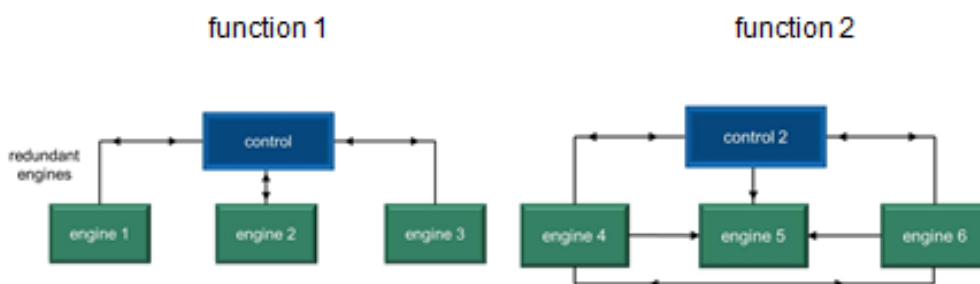


Example on a nuclear PLC-based application



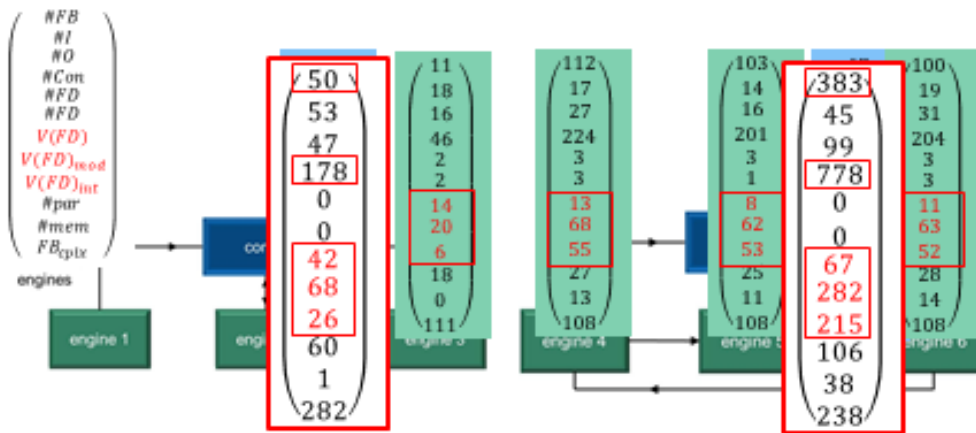
Comparison between two similar applications

- Same functionality, same platform
- different NPP, different approach in the FD-design



- discard monitors of function 1 in order to compare

Comparison between two similar applications



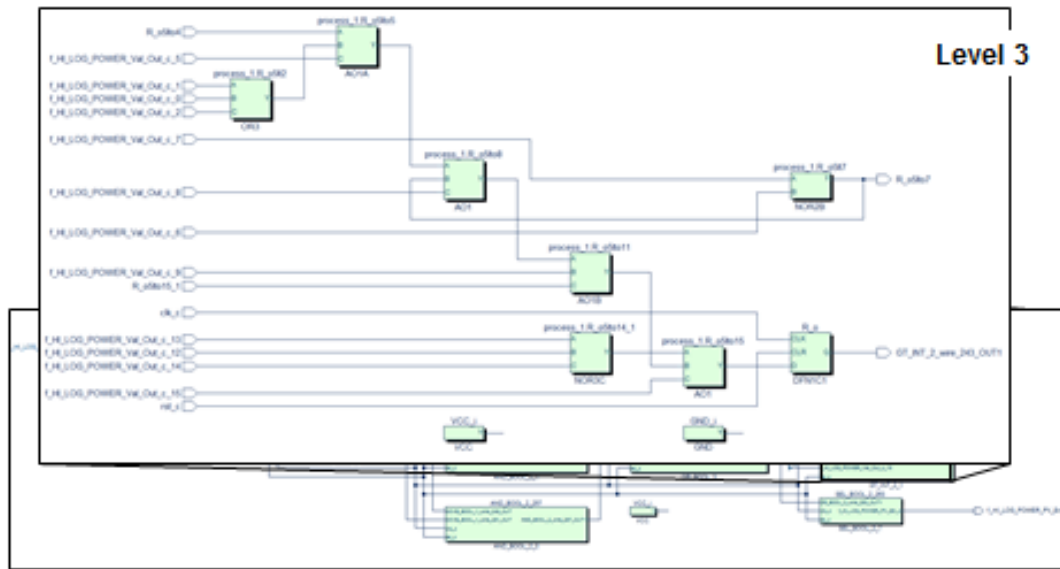
➤ Differences in the FD-design lead to a different complexity

Application on FPGAs

	complexity vector PLC		complexity vector FPGA
K1:	number of function blocks		K1: number of basic blocks (LUT)
K2:	number of input signals		K2: number of input signals
K3:	number of output signals		K3: number of output signals
K4:	number of interconnections		K4: number of interconnections
K5:	number of upstream function diagrams		K5: number of upstream netlists
K6:	number of downstream function diagrams	↔	K6: number of downstream netlists
K7:	interconnection complexity $V(FD)$		K7: interconnection complexity $V(FD)$
K8:	interconnection complexity $V_{mod}(FD)$		K8: interconnection complexity $V_{mod}(FD)$
K9:	interconnection complexity $V_{intern}(FD)$		K9: interconnection complexity $V_{intern}(FD)$
K10:	number of changeable parameters		K10: number of changeable parameters
K11:	number of internal memories		K11: number of internal memories
K12:	complexity of function blocks		K12: complexity of basis blocks

➤ Determination of interconnection complexity feasible for FPGAs

FPGA netlist – Levels of graphical representation



Auto-processing of FPGA netlists

- Description of FPGAs netlists as EDIF-files
EDIF – Electronic Design Interchange Format

```
(edif g_BP
(edifVersion 2 0 0)
(edifLevel 0)
(keywordMap (keywordLevel 0))
(status
(written
(timestamp 2015 8 24 13 29 58)
(author "Synopsys, Inc.")
(program "Synplify Pro" (version "I-2013.09M-SP1 , mapper mapact, build 1154R")))
)
)
(library PA3
(edifLevel 0)
(technology (numberDefinition ))
(cell xnor2 (celltype GENERIC)
(property dont_touch (string "false"))
(view prim (viewtype NETLIST)
(interface
(port Y (direction OUTPUT)
(property function (string "!(A ^ B)"))
```

- EDIF format exhibits a universal character
- Possibility of auto-processing the interconnections directly from EDIF

Auto-processing of FPGA netlists

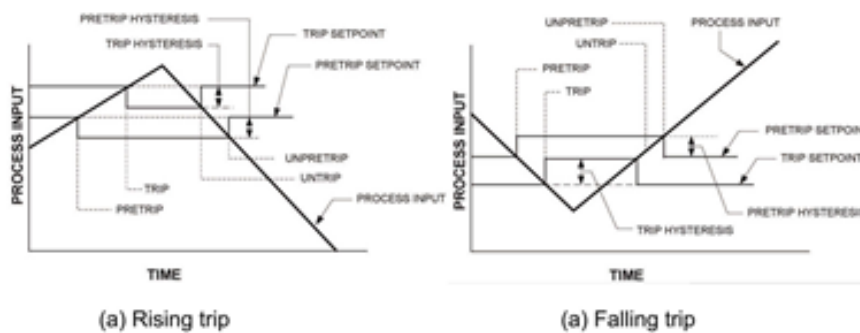
➤ Analysis of the whole netlist – generated function plan (extract)

```

.
.
.
f_._ast_pad->          f_th_HI_LOG_POWER_Trip_Logic_44.status_RNO_1
f_._ast_pad->          f_th_HI_LOG_POWER_Trip_Logic_44.unl_th_HI_LOG_POWER_Trip_Logic_t0_RNO
f_._ast_pad->          f_th_HI_LOG_POWER_Trip_Logic_44.unl_th_HI_LOG_POWER_Trip_Logic_t0_RNO_0
f_th_HI_LOG_POWER_Trip_Logic_pad ->      a_th_HI_LOG_POWER_Trip_Logic
f_th_HI_LOG_POWER_Trip_Logic_pad ->      a_th_HI_LOG_POWER_Trip_Logic
f_AND_BOOL_2_254.process_1_R_o_2 ->      f_AND_BOOL_2_254.R_o
f_AND_BOOL_2_254.R_o ->                  f_OR_BOOL_3_253.process_1_R_o_2
f_AND_BOOL_2_257.process_1_R_o_2 ->      f_AND_BOOL_2_257.R_o
f_AND_BOOL_2_257.R_o ->                  f_OR_BOOL_3_253.process_1_R_o_2
f_AND_BOOL_2_260.process_1_R_o_2 ->      f_AND_BOOL_2_260.R_o
f_AND_BOOL_2_260.R_o ->                  f_OR_BOOL_3_253.process_1_R_o_2
f_AND_BOOL_2_287.process_1_R_o_2 ->      f_AND_BOOL_2_287.R_o
f_AND_BOOL_2_287.R_o ->                  f_AND_BOOL_2_298.process_1_R_o_2
f_AND_BOOL_2_287.R_o ->                  f_AND_BOOL_2_301.process_1_R_o_2
f_AND_BOOL_2_287.R_o ->                  f_AND_BOOL_2_304.process_1_R_o_2
f_AND_BOOL_2_298.process_1_R_o_2 ->      f_AND_BOOL_2_298.R_o
f_AND_BOOL_2_298.R_o ->                  f_OR_BOOL_3_297.process_1_R_o_2
f_AND_BOOL_2_301.process_1_R_o_2 ->      f_AND_BOOL_2_301.R_o
.
.
.
    
```

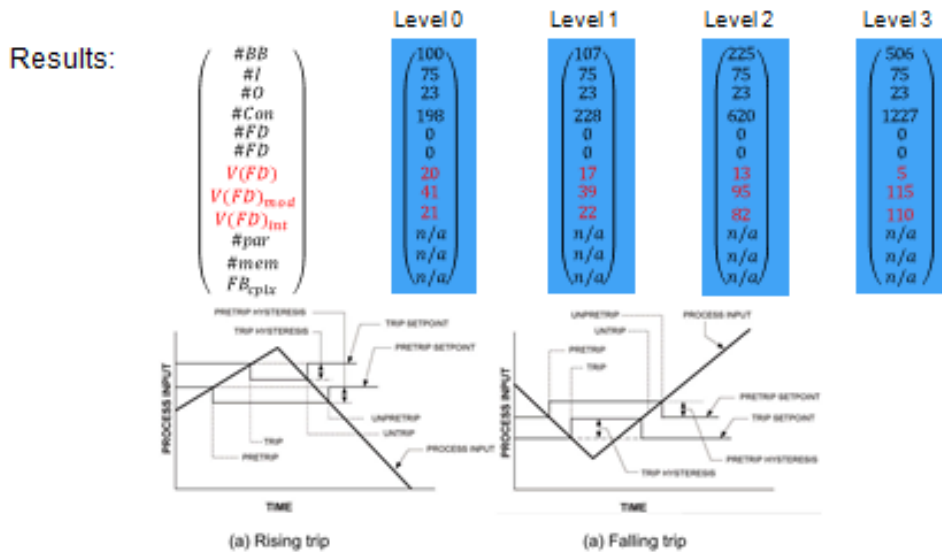
Application on FPGAs: specific example

example of FPGA in nuclear application: Fixed Rising Trip



source: Junbeom Yoo, et al., In Integrated Software Development Framework for PLC & FPGA based Digital IECs, ISOFIC2016 2016, Jeju, Rep. of Korea, August 24–26, 2016

Application on FPGAs: specific example



Conclusions and Outlook

- ✓ Determination of the complexity for different technologies possible
- ✓ Different FD-Designs lead to a different complexity
- ✓ Differing complexity for PLC and FPGA-based logic

- Examination of different FPGA designs
- Comparison of FPGA applications
- Considering the comparison of PLC and FPGA applications (e.g. based on a similar functionality)
- Examination of consequences from differing complexity on error potential



Thanks for your attention!

Are there any questions?

Anhang F: Interne Projektverfolgung

2015-03-20-Protokoll Kick-off-Meeting.docx

2015-04-16-Protokoll Projekt-Meeting.docx

2015-04-29-Protokoll Projekt-Meeting.docx

2015-05-08-Protokoll Projekt-Meeting.docx

2015-05-18-Protokoll Projekt-Meeting.docx

2015-05-26-Protokoll Projekt-Meeting.docx

2015-06-18-Protokoll Projekt-Meeting.docx

2015-06-23-Protokoll Projekt-Meeting.docx

2015-07-06-Protokoll Projekt-Meeting.docx

2015-07-17-Protokoll Projekt-Meeting.docx

2016-02-19-Protokoll Projekt-Meeting.docx

2016-03-01-Protokoll Projekt-Meeting.docx

2016-04-05-Protokoll Projekt-Meeting.docx

2016-05-09-Protokoll Projekt-Meeting.docx

2016-05-31-Protokoll Projekt-Meeting.docx

2016-06-23-Protokoll Projekt-Meeting.docx

2016-08-22-Protokoll Projekt-Meeting.docx

2016-09-21-Protokoll Projekt-Meeting.docx

2016-11-17-Protokoll Projekt-Meeting.docx

2016-11-22-Protokoll Projekt-Meeting.docx

2016-11-22-Protokoll Projekt-Meeting.docx

2016-11-29-Protokoll Projekt-Meeting.docx

2017-01-31-Protokoll Projekt-Meeting.docx

2017-02-13-Protokoll Projekt-Meeting.docx

2017-03-20-Protokoll Projekt-Meeting.docx

2017-05-04-Protokoll Projekt-Meeting.docx

2017-05-22-Protokoll Projekt-Meeting.docx

2017-06-29-Protokoll Projekt-Meeting.docx

2017-08-16-Protokoll Projekt-Meeting.docx

Kontakt:

Bundesamt für die Sicherheit der nuklearen Entsorgung

Wegelystr. 8

10623 Berlin

Telefon: + 49 30 18 4321 0

Internet: www.base.bund.de

Gedruckt auf Recyclingpapier aus 100% Altpapier



Bundesamt
für die Sicherheit
der nuklearen Entsorgung