Eine Datenspezifikationsarchitektur

Stefan Widmann[1]

Abstract: Eingebettete Systeme werden in zunehmendem Maße in sicherheitsgerichteten Anwendungen eingesetzt, so z. B. in Form des Lenkens „steer-by-wire“ und Bremsens „brake-by-wire“ im Automobilbereich. Die Komplexität der in diesen Systemen eingesetzten Hard- und Software steigt, und die fortwährende Verkleinerung der Strukturbreiten in integrierten Schaltkreisen macht diese immer empfindlicher gegenüber Umgebungseinflüssen wie Strahlung, was sich in einer steigenden Wahrscheinlichkeit von Datenflussfehlern auswirkt. Anstatt dem Trend zur Fehlererkennung durch immer komplexere Software auf konventionellen Prozessorarchitekturen zu folgen, wurde in der Arbeit eine neuartige Prozessorarchitektur mit umfassenden hardwarebasierten Fehlererkennungsmerkmalen vorgestellt. Diese ermöglichen die einfache und zuverlässige Erkennung von zur Laufzeit auftretenden Datenflussfehlern, wodurch die Architektur bisherigen Ansätzen deutlich überlegen ist.

1 Einführung

Es besteht deutlicher Bedarf an der Verbesserung heutiger Fehlervermeidungs- und -erkennungsmaßnahmen, die in sicherheitsgerichteten Systemen zum Einsatz kommen, also Systemen, die Verantwortung für Mensch, Umwelt und Investitionsgüter tragen. Es ist notwendig, Fehler weitestgehend zu vermeiden, und trotz aller Vermeidungsmaßnahmen trotzdem auftretende Fehler so frühzeitig wie möglich zu erkennen: idealerweise im Moment ihres Auftretens, und nicht erst, wenn ihre Auswirkungen (z. B. durch Vergleich von Ergebnissen oder Zwischenergebnissen) sichtbar werden und ggf. bereits zu gefährlichen Ausgaben bis hin zu fatalen Unfällen führen. Die möglichen Folgen nicht erkannter oder nicht korrekt behandelter Fehler, die während Spezifikation, Entwurf oder Implementierung von Hard- oder Software in ein System eingebracht wurden oder zur Laufzeit auftreten, zeigen einige Vorfälle aus der Vergangenheit eindrücklich:

Recht bekannt ist die Selbstzerstörung der Rakete Ariane 5, die hohen Sachschaden verursachte. Nach [Li96] kam es bei der Umwandlung einer 64-Bit-Gleitkommazahl in eine vorzeichenbehaftete 16-Bit-Ganzzahl zu einer Überschreitung des Wertebereichs, weil die horizontale Geschwindigkeit der Ariane 5 deutlich höher war als die der Ariane 4. Der Fehler wurde als solcher erkannt und ein Diagnose-Bitmuster an den Bordrechner gesendet, der dieses fehlerhafterweise als Flugdaten interpretierte und die Ablenkdüsen voll aussteuerte, was in der Folge die Selbstzerstörung der Rakete auslöste.

Ebenfalls großen Sachschaden verursachte der Verlust des Mars Climate Orbiter MCO, einer Sonde der NASA. Nach [St99] war die Trägerrakete durch die Verwendung inkompatibler Einheiten – eine Softwarekomponente rechnete in metrischen SI-Einheiten, die andere in angloamerikanischen Maßeinheiten – zum Zeitpunkt des Absetzens der Sonde im Marsorbit rund 170 km zu tief, was entweder zu deren Abdriften oder Verglühen führte.

[1] FernUniversität in Hagen, stefan.widmann@gmx.de

Beim Einsatz des für medizinische Zwecke genutzten Teilchenbeschleunigers Therac-25 kam es zu mindestens sechs schweren Unfällen, bei denen nach [LT93] Patienten aufgrund von Softwarefehlern massiven Strahlungsüberdosen mit teils tödlichen Folgen ausgesetzt wurden. Während das Vorgängergerät Therac-20 noch zusätzliche hardwaretechnische Sicherheitseinrichtungen und mechanische Verriegelungen nutzte, wurden diese Maßnahmen beim Therac-25 durch eine reine Softwarelösung ersetzt – so groß war das Vertrauen in die Software des Geräts. Die Unfälle wurden dadurch ausgelöst, dass es bei entsprechender Nutzereingabe aufgrund unzureichender Synchronisierungsmechanismen zu einer Vermischung vorhergehender und aktueller Betriebsparameter kommen konnte, wodurch sich unzulässige Kombinationen von Bestrahlungsart, -dosen und -zeiten ergaben.

Konventionelle Prozessorarchitekturen sind vor allem auf maximalen Datendurchsatz hin ausgelegt, nicht auf Einfachheit, Fehlervermeidung und -erkennung. Diese – eigentlich ungeeigneten Systeme – kommen meist aus ökonomischen Gründen in den beschriebenen Anwendungen zum Einsatz und werden als „commercial-off-the-shelf (COTS)“ bezeichnet. Die Datenworte in den Speichern dieser Systeme enthalten neben dem eigentlichen Datenwert keinerlei weiterführende Informationen, die dessen Eigenschaften beschreiben. In solchen Architekturen und Systemen kann den steigenden Anforderungen und Fehlerwahrscheinlichkeiten nur durch weiter steigende Komplexität der Software begegnet werden, z. B. durch den Einsatz von softwarebasierter arithmetischer Kodierung wie Software Encoded Processing (SEP) und Compiler Encoded Processing (CEP) [Sc11].

In [Go14] wurde der Frage nachgegangen, wie sich der Kontrollfluss innerhalb sicherheitsgerichteter Echtzeitsysteme mit möglichst einfachen hardwarebasierten Mitteln überwachen lässt. Allerdings wird in [Te87] davon gesprochen, dass 80 - 90 % aller Programmfehler Datenflussfehler sind und nur die verbleibenden 10 - 20 % auf Kontrollflussfehler entfallen.

In der diesem Beitrag zugrundeliegenden Arbeit [Wi17] wurden 20 datenflussbezogene Fehler- und Angriffsarten identifiziert und eine neuartige, leistungsfähige Prozessorarchitektur für die hardwarebasierte Überwachung des Datenflusses innerhalb von sicherheitsgerichteten Echtzeitsystemen vorgestellt. Sie fügt Datenwerten eine umfassende, hardwareles- und -prüfbare Beschreibung ihrer Eigenschaften hinzu, die untrennbar mit den Datenwerten verbunden ist und mit ihnen gespeichert und verarbeitet wird. Dadurch ist die Architektur in der Lage, alle 20 Fehler- und Angriffsarten zur Laufzeit aufzudecken und entsprechende Fehlerbehandlungmaßnahmen einzuleiten und ist bisherigen Ansätzen damit deutlich überlegen.

2 Der Stand von Wissenschaft und Technik und dessen Nachteile

Die konventionellen Architekturen x86 im geschützten und 64-Bit-Modus [In91, AM12] und ARM [AR11] können trotz größter Bemühungen, unter Nutzung komplexester Maßnahmen, einen maximalen Datendurchsatz zu bieten, nur wenige datenflussbezogene Fehler- und Angriffsarten erkennen.

Die auf den konventionellen Architekturen basierenden, für sicherheitsgerichtete Systeme spezialisierten Prozessoren, wie z. B. der TI Hercules [Te15], sind in der Lage, mehr Fehlerarten aufzudecken, basierend auf dem Einsatz gewisser Redundanz- und Diversitätsarten. Allerdings ist die Reichweite der Fehlererkennung begrenzt und viele Fehlerarten bleiben weiterhin unerkannt.

Die weitestgehend in Vergessenheit geratenen Architekturarten Datentyp-, Datenstruktur- und Befähigungsarchitekturen [Gi93, Fe73, AE] fügen Speicherinhalten Kennungen hinzu, die hardwareverständlich die Inhalte des Speichers beschreiben. Damit können durch die Architekturen bestimmte Fehlerarten erkannt werden, doch auch hier bleiben viele weitere unentdeckt.

Datenflussarchitekturen [Gi93] sind auf die Bearbeitung von Datenflüssen spezialisiert, wodurch bestimmte Fehlerarten in der Theorie grundsätzlich nicht auftreten können. Werden diese Architekturen jedoch auf unterster Ebene wiederum durch konventionelle Architekturen realisiert, so z. B. die Verarbeitungseinheiten in [Ł95], dann besteht die genannte Beschränkung der Fehlerbandbreite nicht mehr. Bei der Verarbeitung der Daten findet keinerlei Prüfung derer Eigenschaften statt, wodurch Fehlerarten wie z. B. inkompatible Datentypen der Operanden nicht erkannt werden können.

Das Fehlererkennungsmerkmal Application Data Integrity ADI, später Silicon Secured Memory SSM genannt, des Oracle SPARC M7 Prozessors [Or14] fügt 64-Byte-Datenblöcken eine Versionskennung hinzu und nutzt Teile von Zeigern, um dort eine erwartete Version zu hinterlegen. Dadurch können Abweichungen zwischen erwarteter und vorgefundener Datenversion aufgedeckt werden. Nachteilig ist, dass Versionen nur für Datenblöcke, nicht jedoch für einzelne Datenwerte vergeben werden können. Weiterhin müssen die Versionsangaben aufwendig durch die Software aktualisiert werden.

Die dynamische Datenflussprüfung DDFV [MS07] erlaubt die signaturbasierte Prüfung der Datenflüsse auf Registerebene, ist aber nicht in der Lage, in diese Prüfungen den oder die Speicher einzubeziehen, geschweige denn systemweite Datenflüsse zu überwachen. Zudem kann eine Abweichung vom vorgesehenen Datenfluss erst am Ende eines Überwachungsblocks, und nicht im Moment des Auftretens erkannt werden. Dies kann zu spät sein, und die ungültigen Ergebnisse können unter Umständen bereits zu gefährlichen Ausgaben geführt haben.

Die arithmetische AN-Kodierung [Br60] kann zusammen mit den Erweiterungen zur ANBD-Kodierung [Fo89] einige wichtige Datenflussfehler erkennen, ist jedoch nur für bestimmte Operationsarten und Datentypen geeignet und verursacht erhöhten Laufzeitbedarf. Zudem sind die kodierten Werte nicht mehr ohne Weiteres menschenlesbar, wodurch die Fehlersuche erschwert wird.

Die Datenflussüberwachung in Netzwerkprotokollen wie z. B. TCP [RF81] weist nur wenige Fehlererkennungsmerkmale auf, während Protokolle für sicherheitsgerichtete Feldbusse [IE16] deutlich mehr Fehlererkennungsmechanismen einsetzen. Allerdings wird nur die erfolgreiche Übertragung der Daten über die verschiedenen Kommunikationsstrecken geprüft, nicht jedoch deren Weiterverarbeitung innerhalb der Datenverarbeitungseinheiten.

Allgemein fehlt eine systemweite, ganzheitliche Betrachtung von Daten, ihrer Eigenschaften und ihrer Wege durch ein System. Selbst wenn bestimmte Dateneigenschaften von der Software lokal zur Laufzeit betrachtet oder sogar überwacht werden, so werden diese getrennt von den eigentlichen Daten verwaltet. Es besteht die Gefahr von Inkonsistenzen, und die Information geht bei der Übertragung der Daten zwischen verschiedenen Softwareprogrammen, spätestens jedoch bei der Übertragung der Daten an andere Systemkomponenten verloren. Die Hauptverantwortung für die Fehlererkennung trägt meist die Software, was deren Komplexität und Fehlerwahrscheinlichkeit weiter erhöht.

3 Ziel und Ergebnisse der Arbeit

Ein Ziel der Arbeit [Wi17] war die Identifikation relevanter datenflussbezogener Fehler- und Angriffsarten und der Eigenschaften von Daten im Anwendungsbereich sicherheitsgerichteter Echtzeitsysteme, um

- einfache Fehlervermeidungs- und -erkennungsmöglichkeiten auf Hardwareebene,
- einfache Überwachungsmöglichkeiten von Echtzeitbedingungen durch die Hardware selbst und
- eine ganzheitliche Betrachtung eines Gesamtsystems inklusive aller eingesetzten Hard- und Software

zu erreichen, ohne dem Trend zur unnötigen weiteren Erhöhung der Komplexität der eingesetzten Entwicklungswerkzeuge (z. B. Übersetzer) oder der entstehenden Software zu folgen. In Datentyp-, Datenstruktur- und Befähigungsarchitekturen wurden sehr leistungsfähige Fehlererkennungsmerkmale geboten, und dies mit einfachsten Mitteln. Es galt, diese Merkmale zu nutzen und deutlich zu erweitern.

Feustel zitiert Iliffe in [Fe72] dahingehend, dass die Eigenschaften von Datenfeldern untrennbar von den eigentlichen Daten in den Feldern selbst unterzubringen seien, anstatt in den auf die Felder zugreifenden Algorithmen. Bezog sich diese Anforderung von Iliffe zunächst nur auf die Anzahl der Elemente innerhalb des Felds und deren Datentypen, so wurde sie für die Arbeit zum folgenden zentralen Entwurfsparadigma erweitert:

> *Alle ein Datenspeicherelement beschreibenden Eigenschaften sollen untrennbar mit diesem verknüpft, gespeichert, übertragen, verarbeitet und in einer hardwareverständlichen und -überprüfbaren Form dargestellt werden.*

Im Zuge der Arbeit [Wi17] entstand – basierend auf dieser Forderung – eine neuartige Prozessorarchitektur, die aufgrund der umfassenden, hardwareverständlichen Beschreibung von Dateneigenschaften als „Datenspezifikationsarchitektur“ bezeichnet wird. Die Beiträge der Arbeit zum Stand von Wissenschaft und Technik sind dabei:

- die Identifikation von insgesamt 20 datenflussbezogenen Fehler- und Angriffsarten, die in diesem Beitrag in Tabelle 1 aufgelistet werden,
- eine umfassende Sammlung der Eigenschaften von Daten in sicherheitsgerichteten Echtzeitsystemen und
- die Vorstellung der auf dieser Grundlage entwickelten Datenspezifikationsarchitektur, welche die identifizierten Dateneigenschaften in Form von hardwareverständlichen Kennungen den Datenwerten hinzufügt.

Die Neuheiten der Fehlererkennungsmerkmale der Datenspezifikationsarchitektur gegenüber dem Stand von Wissenschaft und Technik, von denen 8 zum Patent beim Deutschen Marken- und Patentamt angemeldet wurden, sind:

- die Definition von Messwertdatentypen in Form eines Werteintervalls zur Darstellung fehlerbehafteter Werte (Abb. 1), um die Fehlerfortpflanzung bei der Werteverarbeitung durch Intervallarithmetik zu verfolgen und eventuelle Genauigkeitsprobleme zu erkennen, zusammen mit speziellen Befehlen zur Prüfung der Genauigkeit,

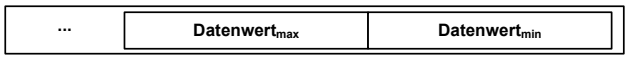

Abb. 1: Datenwert als Werteintervall

- eine Wertebereichskennung [WH17h] (Abb. 2), die es der Hardware erlaubt, einerseits eine Plausibilitätsprüfung beim Lesen eines Datenspeicherelements durchzuführen, indem geprüft wird, ob der Datenwert innerhalb des spezifizierten Wertebereichs liegt, und es ihr andererseits beim Schreiben eines Datenwerts ermöglicht, sicherzustellen, dass dieser innerhalb des spezifizierten Wertebereichs liegt,

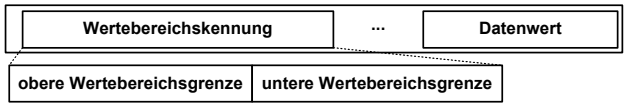

Abb. 2: Wertebereichskennung im Datenspeicherelement

- die Erweiterung der von Datentyparchitekturen bekannten Datentypkennungen um abgeleitete Unterdatentypen mit Einschränkungen der gestatteten Operationen in einer Typberechtigungskennung [WH17a] (Abb. 3), wodurch die versuchte Anwendung nicht zulässiger Operationen von der Hardware als Fehler erkannt wird,

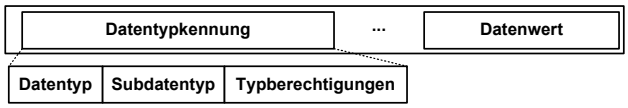

Abb. 3: Erweiterte Datentypkennung im Datenspeicherelement

- eine Einheitenkennung [WH17c] (Abb. 4), die die Einheit des Datenwerts eines Datenspeicherelements in Form von Potenzen der sieben SI-Basiseinheiten beschreibt und der Hardware umfassende Kompatibilitätsprüfungen der Operanden und die Ermittlung der Einheit des Ergebnisses einer Operation gestattet,

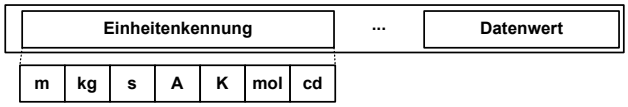

Abb. 4: Einheitenkennung im Datenspeicherelement

- eine Verarbeitungswegkennung [WH17e] (Abb. 5), die beschreibt, wer die Daten erzeugt hat, welche Stationen die Daten auf ihrem Weg von Datenquelle bis -senke verarbeiten dürfen und wer die Daten schlussendlich entgegennehmen darf, so dass die Hardware basierend auf diesen Informationen fehlgeleitete Daten zuverlässig als solche erkennen kann,

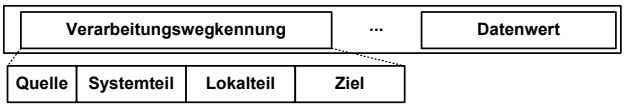

Abb. 5: Verarbeitungswegkennung im Datenspeicherelement

- eine Zeitschrittkennung [WH17f] (Abb. 6), die beschreibt, zu welchem diskreten Zeitpunkt der betroffene Datenwert generiert worden ist, zusammen mit einer Erweiterung des Befehlssatzes um eine Kennung, die die erwartete temporale Beziehung der Operanden einer Operation beschreibt, wodurch die Hardware verlorengegangene Datenaktualisierungen und Synchronisationsfehler erkennen kann,

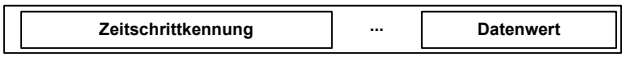

Abb. 6: Zeitschrittkennung im Datenspeicherelement

- eine Fristkennung [WH17b] (Abb. 7), die es gestattet, den Gültigkeitszeitraum der Daten einzugrenzen, wodurch die Hardware die Verwendung von Daten außerhalb dieses Zeitraums als Fehler erkennen kann,

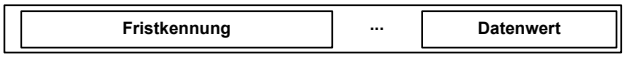

Abb. 7: Fristkennung im Datenspeicherelement

- eine Zykluszeitkennung [WH17g] (Abb. 8), die beschreibt, innerhalb welcher zeitlicher Grenzen eine aktualisierte Version eines Datenwerts erwartet wird, wodurch die Hardware ausbleibende und verfrühte Aktualisierungen des betreffenden Datenwerts als Fehler erkennen kann,

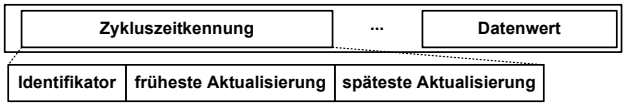

Abb. 8: Zykluszeitkennung im Datenspeicherelement

- eine Signaturkennung [WH17d] (Abb. 9) für besonders anspruchsvolle Anwendungen wie Chipkarten, bei denen der hohe Laufzeitaufwand für Prüfung und Erstellung einer kryptographischen Signatur jedes einzelnen Datenspeicherelements zur Sicherstellung der Schutzziele Integrität und Authentizität zu rechtfertigen ist,

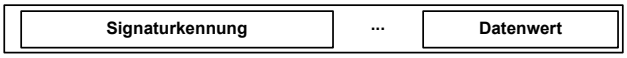

Abb. 9: Signaturkennung im Datenspeicherelement

- Datenportale in Form von Dateneingangs- und -ausgangsportalen, die es ermöglichen, Daten mit Einbeziehung der Speicheradresse in die Integritätsprüfung bzw. Signatur zwischen Systemkomponenten zu übertragen, wobei die Dateneingangsportale bei Nutzung einer Signaturkennung zusätzlich eine Prüfung der Signatur und die Umsignierung auf den eigenen geheimen Schlüssel durchführen, und
- die Vorstellung einer Realisierungsmöglichkeit der Merkmale einer Datenspezifikationsarchitektur in Datenflussarchitekturen durch Erweiterung der Verarbeitungseinheiten, um auch in diesen eine Vielzahl der identifizierten Fehler- und Angriffsarten erkennen zu können.

Unter Nutzung aller in der Arbeit vorgestellten Kennungen ergibt sich für alle innerhalb einer Datenspezifikationsarchitektur genutzten Datenspeicherelemente der in Abbildung 10 gezeigte atomare Aufbau.

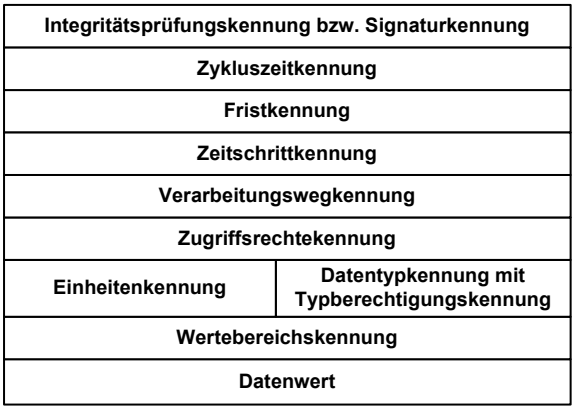

Abb. 10: Aufbau der Datenspeicherelemente einer DSA

Durch die hardwarebasierte Prüfung der in den Kennungen spezifizierten Dateneigenschaften ist es der Datenspezifikationsarchitektur möglich, alle 20 Fehler- und Angriffsarten zur Laufzeit zu erkennen. Dies wird in Tabelle 1 ersichtlich, in der die neue Architektur den konventionellen Architekturen x86 und ARM gegenübergestellt wird. Dabei wird ihre Überlegenheit in Bezug auf die Fehler- bzw. Angriffserkennung deutlich. Die meisten der Prüfungen können durch die Hardware parallel zur Ausführung der eigentlichen Operationen stattfinden, wodurch sich der Laufzeitaufwand nicht erhöht.

Durch den Einsatz der vorgestellten Fehlererkennungsmerkmale einer Datenspezifikationsarchitektur hätten in den eingangs vorgestellten Praxisbeispielen Ariane 5, Mars Climate Orbiter und Therac 25 die ursächlichen Fehler entweder vermieden oder deren Auswirkungen durch deren Erkennung und angemessene Behandlung deutlich reduziert werden können.

Tab. 1: Fehlererkennung durch konventionelle Architekturen und die Datenspezifikationsarchitektur

Fehlerart	**x86, ARM**	**DSA**	**Merkmal der DSA**
Inkompatible Datentypen	–	+	Datentypkennung
Inkompatible Einheiten	–	+	Einheitenkennung
Wertebereichsunter- bzw. -überschreitung	◯ x86	+	Wertebereichskennung
Genauigkeitsproblem	–	+	Messwertdatentypen mit Werteintervall
Falsche Operandenauswahl	–	+	Speicheradresse in Integritätsprüfungs- oder Signaturkennung
Falsche Operatorauswahl	–	+	Diversitäre ALE
Fehlerhaftes Operationsergebnis	–	+	Diversitäre ALE
Fristüberschreitung	–	+	Fristkennung
Zyklusunterschreitung	–	+	Zykluszeitkennung
Zyklusüberschreitung	–	+	Zykluszeitkennung
Verlorengegangene Datenaktualisierung	–	+	Zeitschrittkennung
Synchronisationsfehler oder unvollständige Datenübertragung	–	+	Zeitschrittkennung
Pufferunter- oder -überläufe	(+) x86	+	Sichere Felder, Datentypkennung
Fehlerhafter Datenfluss (falsche Adressaten, ...)	–	+	Verarbeitungswegkennung
Duplizierte Daten	–	+	Zeitschrittkennung
Durch Fehler oder Störungen verfälschte Daten	◯	+	Integritäts- oder Signaturkennung
Fehlerhafter Datenzugriff (fehlende Zugriffsrechte)	◯	+	Zugriffsrechtekennung
Nutzung nicht initialisierter Daten	–	+	Zugriffsrechtekennung
Angriffsart			
Gezielt verfälschte Daten	–	+	Signaturkennung
Wiedereinspielungsattacke	–	+	Signaturkennung mit Zeitschritt-, Frist- und Zykluszeitkennung

Fehlererkennung: – nicht möglich, ◯ begrenzt möglich, (+) mit Einschränkungen möglich, + möglich;
DSA: Datenspezifikationsarchitektur; ALE: Arithmetisch-Logische Einheit

Literaturverzeichnis

[AE] AEG Datenverarbeitung: TR 4 Bedienungshandbuch.

[AM12] AMD: AMD64 Architecture Programmer's Manual Vol. 2: System Programming. http://developer.amd.com/wordpress/media/2012/10/24593_APM_v2.pdf, 2012.

[AR11] ARM Limited: Migrating from IA-32 to ARM. Application Note 274, ARM DAI 0274, 2011.

[Br60] Brown, D. T.: Error Detecting and Correcting Binary Codes for Arithmetic Operations. IRE Transactions on Electronic Computers, Vol. EC-9, Issue 3, 1960.

[Fe72] Feustel, E.: The Rice research computer: a tagged architecture. AFIPS '72 (Spring) Proceedings of the May 16-18, 1972, spring joint computer conference, S. 369–377, 1972.

[Fe73] Feustel, E.: On the Advantages of Tagged Architectures. IEEE Transactions on Computers, Vol. C-22, No. 7:644–656, 1973.

[Fo89] Forin, P.: Vital Coded Microprocessor Principles and Application for Various Transit Systems. IFAC Control, Computers, Communications, S. 79–84, 1989.

[Gi93] Giloi, W.: Rechnerarchitektur. Springer-Verlag, 2. Auflage, 1993.

[Go14] Gollub, L.: Verfahren zur Kontrollflussüberwachung in sicherheitsgerichteten Rechensystemen. VDI Verlag GmbH, 2014.

[IE16] IEC 61784-3: Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions. 2016.

[In91] Intel: 80386 System Software Writer's Guide. 1991.

[Ł95] Łent, B.: A Contribution To The Design Of A Disjunctive Computer Architecture For Real Time Control Systems. Dissertation, FernUniversität in Hagen, 1995.

[Li96] Lions, J.-L. et al.: Ariane 501 Inquiry Board report. http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf, 1996.

[LT93] Leveson, N. G.; Turner, C. S.: An Investigation of the Therac-25 Accidents. Computer, Vol. 26, Issue 7:18–41, 1993.

[MS07] Meixner, A.; Sorin, D. J.: Error Detection Using Dynamic Dataflow Verification. 16th International Conference on Parallel Architecture and Compilation Techniques (PACT 2007), S. 104–118, 2007.

[Or14] Oracle: Introduction to SPARC M7 and Application Data Integrity (ADI). https://swisdev.oracle.com/_files/What-Is-ADI.html, 2014.

[RF81] RFC 793: Transmission Control Protocol. DARPA Internet Program, Protocol Specification, 1981.

[Sc11] Schiffel, U.: Hardware Error Detection Using AN-Codes. Dissertation, Technische Universität Dresden, 2011.

[St99] Stephenson, A. G. et al.: Mars Climate Orbiter Mishap Investigation Board Phase I Report. ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf, 1999.

[Te87] Teller, J.: Problematik der Datenflussfehler. Angewandte Informatik, Ausgabe 29, Nr. 6:240–247, 1987.

[Te15] Texas Instrument: Safety Manual for RM48x Hercules ARM-Based Safety Critical Microcontrollers. User's Guide, Dokumentennummer SPNU577D, 2015.

[WH17a] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Einschränkung der zulässigen Operationen auf Daten in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 005 945.4 beim Deutschen Patent- und Markenamt, 2017.

[WH17b] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung der Datennutzung außerhalb ihres Gültigkeitszeitraums in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 005 974.8 beim Deutschen Patent- und Markenamt, 2017.

[WH17c] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung inkompatibler Operandeneinheiten in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 005 975.6 beim Deutschen Patent- und Markenamt, 2017.

[WH17d] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung von absichtlichen oder durch Störungen und / oder Fehler verursachten Datenverfälschungen in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 006 354.0 beim Deutschen Patent- und Markenamt, 2017.

[WH17e] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung von Datenflussfehlern in Datenverarbeitungseinheiten und -systemen. Patentanmeldung 10 2017 005 972.1 beim Deutschen Patent- und Markenamt, 2017.

[WH17f] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung von Synchronisations- und Datenaktualisierungsfehlern in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 005 970.5 beim Deutschen Patent- und Markenamt, 2017.

[WH17g] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung von Verletzungen von zyklischen Echtzeitbedingungen in Datenverarbeitungseinheiten und -systemen. Patentanmeldung 10 2017 005 944.6 beim Deutschen Patent- und Markenamt, 2017.

[WH17h] Widmann, S.; Halang, W. A.: Vorrichtung und Verfahren zur gerätetechnischen Erkennung von Wertebereichsverletzungen von Datenwerten in Datenverarbeitungseinheiten. Patentanmeldung 10 2017 005 971.3 beim Deutschen Patent- und Markenamt, 2017.

[Wi17] Widmann, S.: Eine Datenspezifikationsarchitektur. VDI Verlag GmbH, 2017.

Stefan Widmann wurde in Schwäbisch Gmünd geboren und studierte nach dem Abitur Industrieelektronik an der Fachhochschule Ulm. Nach Erlangung seines Diploms im Jahr 2004 arbeitete er zunächst bei einem Mittelständler als Entwicklungsingenieur im Bereich Leistungselektronik. 2006 wechselte er zur Siemens AG in Amberg, wo er seither als Projektleiter, Firmwarearchitekt und -entwickler in der Leistungsschalterentwicklung tätig ist. Seit 2008 betreibt er zudem als Freiberufler ein eigenes Ingenieurbüro. Nebenberuflich absolvierte er von 2010 bis 2014 das Masterstudium der Elektro- und Informationstechnik an der FernUniversität in Hagen und wurde dort anschließend unter der Betreuung von Prof. Dr. Dr. W. A. Halang im Jahr 2017 zum Doktor-Ingenieur promoviert. Für seinen Masterabschluss und seine Dissertation wurde er jeweils mit einem Preis ausgezeichnet. Sowohl seine Masterarbeit, als auch seine Dissertation sind als Bücher im VDI Verlag erschienen.