from: ITG-Fachtagung Mikroelektronik für die Informationstechnik, Stuttgart, 1992

Funktionsblockredundanz als kostengünstiges Fehlertoleranzverfahren für mikroelektronische Steuerungen

R. Ernst, P. Nowottnick

Institut für Datenverarbeitungsanlagen, TU Braunschweig,
Hans-Sommer-Str. 66, 3300 Braunschweig,
0531/391 3730, ernst@ida.ing.tu-bs.de

Kurzfassung

Funktionsblockredundanz ist eine dynamische Redundanztechnik für Fehlertoleranz in VLSI-Schaltkreisen. Sie ist allgemein einsetzbar auch für Schaltungen mit nichtregulärer Logikstruktur, wie sie in Gate-Arrays implementiert werden. Funktionsblockredundanz nutzt funktionale Ähnlichkeit, wie etwa mehrfach auftretende Zähler- und Schieberegisterfunktionen, um den Zusatzaufwand für Standby-Module zu reduzieren. Am Beispiel eines manuell optimierten industriellen Gate-Arrays zeigt sich ein nur sehr geringer Overhead von nur 80% der ursprünglichen Schaltungsfunktion, der bislang für derartige Schaltungen nicht erreichbar war.

1 Einleitung

VLSI-Schaltungen dringen zunehmend in Bereiche mit entscheidenden Zuverlässigkeits- und Verfügbarkeitsanforderungen ein, darunter auch Massenprodukte, wie etwa die Automobilelektronik. Fehlertolerante Steuerungen könnten die Verfügbarkeit und Zuverlässigkeit dieser Produkte erhöhen, jedoch sind gegenwärtige Redundanztechniken [2], [6], [12] zu aufwendig für kostensensitive Massenprodukte.

Mit zunehmender Komplexität mikroelektronischer Steuerungen wird der Aspekt der Fehlertoleranz erheblich an Bedeutung gewinnen, zumal wenn aufgrund eines verteilten zu steuernden Gesamtsystems der Integration Grenzen gesetzt sind. Ein gutes Beispiel hierfür ist die Automobilelektronik. Nehmen wir einmal an, daß in einem künftigen Automobil 100 VLSI-Komponenten verwendet werden. Für ein industrielles Gate-Array wird bei einer Sperrschichttemperatur von 115°C als Fehlerrate 297 FIT (1 FIT = 10^{-6} Fehler/1000h) angegeben[1] [8]. Nimmt man weiterhin 500 Betriebsstunden pro Jahr an, dann werden jährlich etwa 1,5% der Fahrzeuge eine defekte Komponente aufweisen[2]. Damit wird Fehlertoleranz nicht nur zur Sicherheits-, sondern bereits zur Qualitätsfrage.

Es gilt daher, Fehlertoleranzverfahren mit geringem Aufwand zu finden. Besonders interessant sind dabei Verfahren, bei denen gleichzeitig, bei geringer Leistungseinbuße, auch Fehlertoleranz für die Verbindungsleitungen geboten wird.

Jedes Fehlertoleranzverfahren bezieht sich auf ein Fehlermodell. Im folgenden betrachten wir Einzelfehlertoleranz für die Haftfehler ständig-1, ständig-0 und ständig-offen, wie sie auch beim Test verwendet werden. Heutige Fehlertoleranzverfahren kommen immer dann mit geringem Aufwand aus, wenn hochreguläre Schaltungsstrukturen vorliegen, wie Speicher [7], [12] und Prozessor-Arrays [3], [4], [9]. Hier gibt es unterschiedliche Verfahren, von redundanten Codes, bei denen Wortparallelismus ausgenutzt wird, bis hin zum Multiprozessor mit "Graceful Degradation". Ansonsten liegt der gesamte zusätzliche Aufwand für Einzelfehlertoleranz erheblich höher als 100%. Regu-

---

[1] Die Annahme einer relativ hohen Sperrschichttemperatur ist durch die Zunahme der Verlustleistung mit der Taktrate gerechtfertigt, denn die Gegenmaßnahme, das Absenken der Betriebsspannung, ist aus Gründen der Störsicherheit begrenzt.

[2] bei Gleichverteilung der Fehler

läre Strukturen bilden jedoch nur einen kleinen Teil der meisten Steuerungen, besonders wenn sie mit ASICs implementiert werden.

Wir stellen ein Fehlertoleranzverfahren vor, die *Funktionsblockredundanz*, die bereits bei *Ähnlichkeiten* von Schaltungs*funktionen* einen geringeren Schaltungsaufwand erzielt. Bei dem Verfahren kann z.B. das mehrfache Auftreten von Zählern und Registern in einer Schaltung ausgenutzt werden. An einem Beispiel werden wir zeigen, daß das Verfahren vollständige Fehlertoleranz mit einem weit geringeren Aufwand erreichen kann als irgend ein anderes bekanntes Verfahren.

2 Funktionsblockredundanz

Zur Identifikation von ähnlichen Schaltungsfunktionen muß die Schaltung in Funktionsblöcke zerlegt werden. Im Falle eines hierarchischen Entwurfsstils ist die Blockhierarchie ein guter Ausgangspunkt.

Funktional ähnliche Schaltungsblöcke werden zu Gruppen zusammengefaßt. Für jede dieser Gruppen wird ein *Superblock* definiert, der die Funktionen aller Gruppenmitglieder überdeckt. Der Superblock wird als Reservemodul (Spare) für alle Blöcke der Gruppe verwendet. Er kann durch externe Signale auf die Funktion des jeweils zu ersetzenden Blocks eingestellt werden. Auch wortparallele Schaltungsstrukturen, die in Bit-Slices vorliegen, lassen sich so zu Gruppen zusammen fassen. Alle verbleibenden Schaltungsblöcke werden zu Gruppen der Größe 1 zusammengefaßt, d.h. sie erhalten ein identisches Reservemodul.

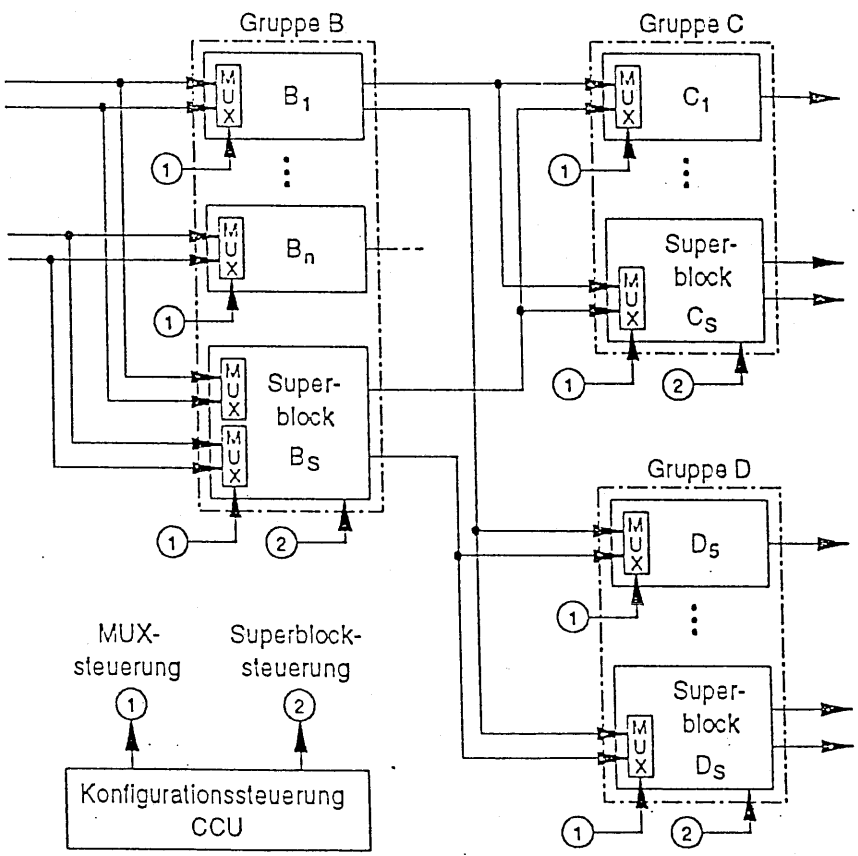

Bild 1: Funktionsblocksredundanz (FBR) - Prinzip

Bild 1 zeigt das Prinzip. Block B besteht aus n Blöcken B_1 bis B_n. In dem Beispiel werden die Ausgänge von B_1 mit C_1 und D_5 der Gruppen C bzw. D verbunden. Superblock B_S muß B_1 ersetzen können, und stellt daher ebenfalls Ausgangssignale für diese Blöcke. Da die Superblöcke C_S und D_S der Gruppen C und D Reservemodule für C_1 und D_5 sind, sind sie ebenso mit diesen Signalen verbunden. Multiplexer an den Blockeingängen werden zum Umschalten verwendet. Um B_1 durch B_S zu ersetzen, werden die gezeichneten Multiplexer an den Eingängen von C_1, D_5, C_S und D_S auf die Ausgangssignale von B_S eingestellt, und BS wird auf die Funktion von B_1 programmiert. Multiplexer- und Superblock-Steuersignale werden von einer zentralen Konfigurationssteuerungseinheit gebildet.

Die Ähnlichkeit von Schaltungsfunktionen ist keine eindeutige Größe und läßt damit Freiheiten bei der Gruppenbildung. Daraus entsteht ein Optimierungsproblem:

- Große Gruppen mit kleinen Superblöcken können gefunden werden, wenn die Schaltung in eine große Zahl funktionaler Blöcke zerlegt wird (Extremfall: Einzelgatter).

- Multiplexer- und Verdrahtungsaufwand steigen aber mit der Zahl an Funktionsblöcken.

- Größere Gruppen bei gleichbleibender Zahl an Funktionsblöcken erfordern wenige Superblöcke, wodurch der Aufwand in erster Betrachtung herabgesetzt wird.

- Bei gleichen Funktionsblöcken werden größere Gruppen durch geringere Ähnlichkeit der Funktionen einer Gruppe ermöglicht. Dies geht auf Kosten komplexerer Superblockfunktionen, womit der Schaltungsaufwand für den einzelnen Superblock größer wird.

- Der Multiplexer- und Verdrahtungsaufwand ist davon abhängig, wie die Grenzen zwischen den Funktionsblöcken gewählt werden. Hier hilft die Vorarbeit im hierarchischen Entwurf.

Als obere Grenze für den Aufwand läßt sich die Verdopplung der Schaltung angeben. Das entspricht gerade der bekannten dynamischen Redundanz mit Standby.

Funktionsblockredundanz (FBR) erschien uns ein geeigneter Name, da Funktionsblöcke im Mittelpunkt stehen und das Verfahren auf der Funktionsblockebene eines Entwurfs aufsetzt.

Der Schaltungsaufwand für die zentrale *Konfigurationssteuerung* (CCU) kann sehr klein sein. Aufgrund des Einzelfehlermodells muß nur höchstens ein fehlerbehafteter Block angenommen werden. Wenn dieser Block durch den zugehörigen Superblock ersetzt wird, kann gleichzeitig je eine beliebige Ersetzung in allen anderen Gruppen erfolgen, denn die Schaltungsfunktion wird nicht verändert, wenn ein funktionsfähiger Block durch seinen (fehlerfreien) Superblock ersetzt wird. Damit kommt man mit einer einzigen Steuerfunktion für die gesamte Schaltung aus. Die Zahl der möglichen Konfigurationen ist damit gleich der maximalen Gruppengröße. Die gemeinsame Steuerung kann als Nebeneffekt zur Verringerung des Multiplexeraufwands eingesetzt werden.

Fehlerdiagnose und Rekonfiguration können mit einem iterativen Verfahren erfolgen, da die Gruppengröße im allgemeinen auf < 32 beschränkt sein wird. Dazu wird successive in jeder Konfiguration ein Schaltungstest für die momentan selektierten Funktionsblöcke durchgeführt, bis eine fehlerfreie Konfiguration gefunden ist. Die Steuerung selbst kann nicht rekonfiguriert werden. Da sie eine sehr einfache Struktur hat (im wesentlichen ein Zähler), kann sie statisch redundant ausgelegt werden, in diesem Fall mit TMR, d.h. verdreifacht mit einem 2-aus-3 Voter. Die Teststeuerung ist mit Verdopplung und Vergleich aufgebaut [5], und bei einem Fehler wird sie deaktiviert (Prüfredundanz).

Mit diesen Maßnahmen wird es für alle einzelnen Haftfehler immer eine fehlerfreie Konfiguration geben.

- Fehler in Funktionsblöcken und ihren Eingangsmultiplexern werden durch Superblock-Ersetzung maskiert. Fehler in Superblöcken oder in der Superblocksteuerung der CCU werden maskiert, indem die Superblöcke nicht genutzt werden.

- IC-Pins, Schnittstellensignale und bidirektionale Busse werden als Funktionsblöcke behandelt.

- Die CCU ist statisch fehlertolerant. Selbst Voter-Fehler werden toleriert, denn die Voter-Ausgänge steuern Multiplexer in Funktionsblöcken, die geschlossen ersetzt werden können. Ausnahme sind stuck-open Fehler der Voter. Statische Redundanz wird auch auf die Takttreiber angewendet, wobei das Taktnetz doppelt ausgelegt und in den Funktionsblöcken gemultiplext wird.

Da für jeden Einzelfehler eine fehlerfreie Konfiguration gefunden werden kann, ist die Gesamtschaltung fehlertolerant.

Der gegenwärtige Schaltungszustand geht während der Fehlerdiagnose und Rekonfiguration verloren. Das System muß daher entweder einen Rücksetzvorgang auslösen, oder es ist eine Rückwärtskorrektur [2] ("Roll Back and Recovery") unter Programmkontrolle notwendig.

3 Test

Der Test dient zwei Zielen. Zum einen soll für eine gegebene Konfiguration festgestellt werden, ob sie fehlerfrei funktioniert. Dies genügt während des Betriebs. Zum anderen muß getestet werden können, ob die gesamte Schaltung einschließlich Reservekomponenten fehlerfrei ist, denn nur dann ist sie fehlertolerant. Dieser zweite Test, der u.a. bei der Herstellung gefordert ist, macht auch einen Test der Multiplexer-Selektfunktion, der CCU und der Teststeuerung erforderlich.

Für den ersten Testfall genügt der übliche Schaltungstest. Das Testverfahren und die Testmuster der ursprünglichen Schaltung können dabei übernommen werden, d.h. Funktionstest ist ebenso möglich wie Strukturtest mit deterministischen oder stochastischen Mustern. Bei Verwendung von Scan-Pfaden müssen diese Pfade auch geschaltet werden, um eine korrekt funktionierende Konfiguration auch im Fehlerfall erkennen zu können.

Aufwendiger ist der zweite Fall. Da jeder Funktions- und Superblock in wenigstens einer Konfiguration eingesetzt wird, genügt es für den Test dieser Blöcke, in jeder Konfiguration einen Test der Schaltung durchzuführen. Wenn alle Konfigurationen getestet werden und dann ein Betriebstest anschließt, werden dabei auch alle Funktionen der CCU und der Teststeuerung vollständig durchlaufen, und ein Fehler muß sich in einer Differenz der wenigen Steuersignale zeigen, die zur Erkennung mit Komparatoren versehen werden. Das Resultat ist ein vollständiger Funktionstest.

Problematischer ist der Multiplexertest. Bedingt durch die Redundanz durch Superblöcke werden Fehler der Selektierung mindestens dann nicht erkannt, wenn Funktionsblock und Superblock identisch aufgebaut sind. Bei der statistischen Fehlerüberdeckung mag dies kaum ins Gewicht fallen, dennoch können als Konsequenz ganze Funktionsblöcke unentdeckt fehlerhaft sein. Für den Multiplexertest muß die Redundanz daher aufgehoben werden. Dazu werden in die Ausgangssignale der Superblöcke, wo notwendig, Exclusiv-Oder-Gatter eingebracht, die die Ausgangsfunktion während eines getrennten Multiplexertests invertieren. Damit ist die Redundanz aufgehoben und Fehler der Selektierung werden entdeckt.

4 Anwendungsbereiche

Die möglichen Anwendungsbereiche werden durch die Anforderungen des zu steuernden Systems bestimmt. Es gibt zwei Dimensionen, Anforderungen an die Integrität und die Stetigkeit einer Steuerung [5]. Eine Steuerung mit *Integrität* erzeugt nie eine fehlerhafte Ausgabe, bleibt aber gegebenenfalls stehen, wenn ein Fehler auftritt ("Fail Stop"). Eine Steuerung mit *Stetigkeit* wird auch im Fehlerfall nicht unterbrochen, kann aber für eine begrenzte Zeit fehlerhafte Signale erzeugen. Nur statische Fehlertoleranztechniken mit hohem Aufwand liefern beide Eigenschaften. Die meisten zu steuernden Systeme tolerieren jedoch fehlerhafte Steuersignale bis zur einer Systemzeitkonstanten, der Toleranzfrist ("Grace Time").

Aufgrund der wenigen Konfigurationen wird die Testzeit bei der iterativen Rekonfiguration 100 ms üblicherweise nicht übersteigen. Das ist weniger als die Toleranzzeit vieler mechanischer Systeme in Massenprodukten, so daß deren Stetigkeitsanforderungen erfüllt werden, wenn Test und Rekonfiguration im laufenden Betrieb durchgeführt werden. Integrität jedoch kann zwischen zwei Tests nicht garantiert werden. Aber das ist in vielen Anwendungen kein Hindernis. Kurzzeitige Ausfälle der Klimatisierung, eines Telefax-Gerätes oder des Fahrzeugnavigationssystems erscheinen akzeptabel, wenn das System dann nach einigen Sekunden überhaupt wieder in einen funktionsfähigen Zustand zurückkehrt.

Im übrigen kann FBR durch einen nebenläufigen Test (Concurrent Test) ergänzt werden. Stetigkeit und Integrität werden dabei erheblich verbessert [11].

5 Resultate bei einem realistischen Beispiel

Um das Verfahren näher zu untersuchen, wurde es auf eine industrielle Schaltung angewandt [10]. Gewählt wurde eine ABUS-Interfaceschaltung, die als handoptimiertes Gate-Array mit 7000 Gatteräquivalenten vorlag [1]. Die Schaltung zeichnete sich durch geringe funktionale oder strukturelle Ähnlichkeit ihrer Schaltungsblöcke aus, so

hatten die wenigen Zähler unterschiedliche Wortbreiten und Steuerfunktionen. Bidirektionale Busse, optimierte, mehrstufige Dekoder und eine starke Verknüpfung der Teilschaltungen bildeten ungünstige Ausgangsbedingungen.

Bild 2 zeigt eine Übersicht über das Ergebnis. Insgesamt wurden 26 Gruppen mit bis zu 13 Blöcken gebildet. Nur für die Gruppen der Größe 1, d.h. wo keine Ähnlichkeit genutzt wurde, betrug der Zusatzaufwand mehr als 100%. Insgesamt stellen diese Gruppen jedoch nicht einmal die Hälfte der Gesamtschaltung, in allen anderen Fällen brachte FBR einen Vorteil gegenüber dem bisher günstigsten Verfahren, d.h. der Verdopplung. Der Aufwand für die CCU ist nahezu vernachlässigbar.

Dies erklärt das gute Ergebnis. Die ursprüngliche Schaltung hat 7034 Gatteräquivalente, während der modifizierte, fehlertolerante Entwurf auf 12639 Gatteräquivalente kommt. Um den Verdrahtungsaufwand zu schätzen, zerlegten wir für die Zählung alle Mehrpunktnetze in Zweipunktnetze[3]. Die Anzahl der Netze steigt damit von 6492 auf 12141. Der gesamte Mehraufwand beträgt damit 80% für die Schaltung und 87% für die Verdrahtung. Ein vergleichbar geringer Aufwand wurde bisher ausschließlich für reguläre Schaltungsstrukturen erreicht.

Im Augenblick liegt die Schaltung als Logikmodell vor, zur Zeit wird sie als Standardzellenschaltung implementiert. Die Implementierung soll vor allem Aufschluß über das Zeitverhalten geben.

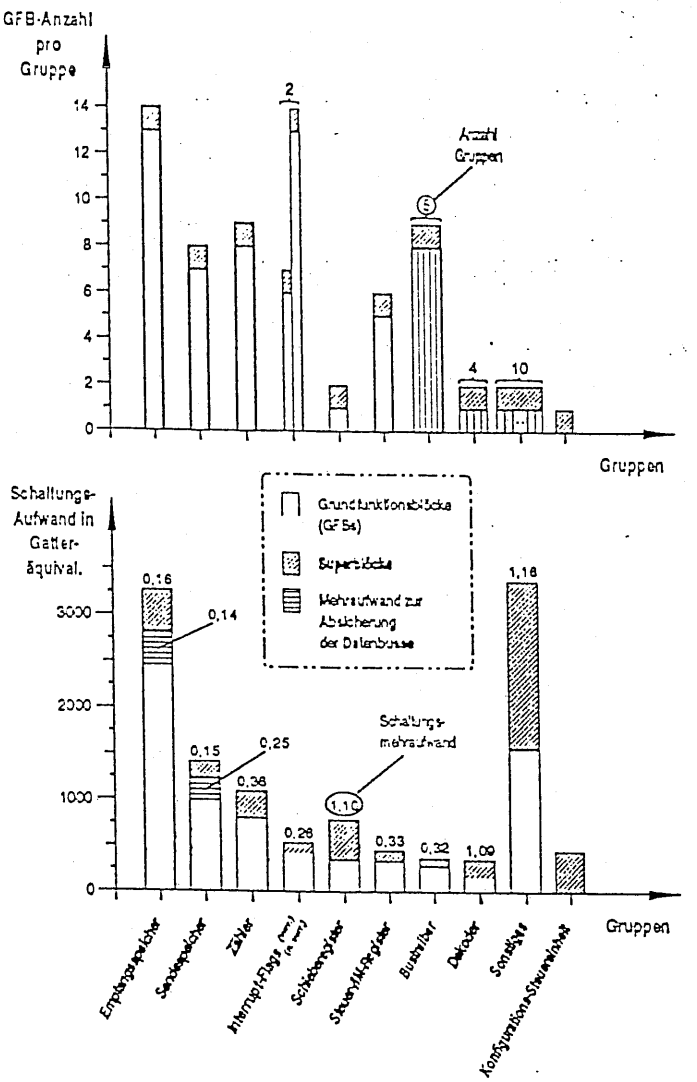

Bild 2: Schaltungsaufwand für ein Businterface

6 Relevanz des Fehlermodells

FBR liefert Fehlertoleranz für alle einzelnen Haftfehler. Das zugrundeliegende Einzelfehlermodell deckt viele physikalische Fehlertypen ab, insbesondere die meisten Fehler des Chip-Interface, die den überwiegenden Teil der heutigen IC-Ausfälle verursachen. Jedoch können nicht alle der vielen verschiedenen Fehlertypen [13] abgedeckt werden, ein Beispiel sind Kurzschlüsse. Folglich hängt die Relevanz des Fehlermodells von der Fehlerverteilung ab, und die ist technologie- und prozeßabhängig. Daten hierzu werden üblicherweise nicht publiziert. Grundsätzlich könnte der Anteil nicht erfaßter physikalischer Fehler jedoch auf Kosten zusätzlicher Chipfläche reduziert werden, z.B durch Macrozell-Layout und spezielle Verdrahtungsregeln. FBR ist allerdings nicht dazu gedacht, maximale Fehlertoleranz bei extremen Sicherheitsanforderungen zu erreichen, sondern es bietet einen guten Kompromiß zwischen der Toleranz physikalischer Fehler und Schaltungsaufwand. In dieser Hinsicht könnte man es durchaus mit dem Partial-Scan-Ansatz im Schaltungstest vergleichen.

---

[3] Das ist eine pessimistische Schätzung, denn die Folge ist eine übermäßige Wichtung des Verdrahtungsaufwands für die 3-Punktnetze zwischen Funktionsblöcken (siehe Bild 1).

7 Zusammenfassung und Ausblick

Funktionsblockredundanz (FBR) ist ein dynamisches Redundanzverfahren für die Funktionsblockebene eines Entwurfs, das besonders für mikroelektronische Steuerungen geeignet ist. Die Resultate zeigen, daß FBR in der Lage ist, mit weit geringerem Aufwand als heutige Verfahren, Fehlertoleranz für alle einzelnen ständig-1 und ständig-0, sowie fast alle ständig-offen Fehler zu bieten. FBR ist damit ein Kompromiß zwischen hoher Zuverlässigkeit und niedrigem Schaltungsaufwand.

Für die Zukunft sind die Implementierung weiterer Beispiele, die Kombination mit effizienten Fehlertoleranzverfahren für Speicher und PLA, eine Erweiterung von FBR auf die Systemebene und die Einbeziehung schaltbarer Analogkomponenten geplant. Vielversprechend wäre eine automatisierte Lösung des Optimierungsproblems.

Danksagung

Die Autoren möchten sich bei der *Volkswagen AG* und dem *Institut für Angewandte Mikroelektronik (IAM)* für die Überlassung des Entwurfsbeispiels bedanken.

Literaturverzeichnis

[1] "ABUS-Interface Specification Manual", Volkswagen AG, Wolfsburg 1991.

[2] Belli, F.; Echtle, K.; Görke, W.: "Methoden und Modelle der Fehlertoleranz", Informatik Spektrum 9, 1986, S. 68-81.

[3] Gay, F.A.; Ketelsen, M.L.: "Performance evaluation for gracefully degrading systems", Proc. FTCS-9, 51, 1979.

[4] Johnson, D.: "The Intel 432: A VLSI Architecture for Fault-Tolerant Computer Systems", IEEE Computer, August 1984, S. 40-48.

[5] Kirrmann, H.D.: "Industrieller Einsatz fehlertoleranter Rechner", Informationstechnik it 30 (1988) 3, S. 186-195.

[6] Lala, P.K.: "Fault Tolerant & Fault Testable Hardware Design", London, Prentice-Hall International 1985, S. 69-134.

[7] Moore, W.R.: "A Review of Fault-Tolerant Techniques for the Enhancement of Integrated Circuit Yield", Proc. of IEEE, Vol. 74, No. 5, May 1986, S. 684-698.

[8] "MOTOROLA MCA3 Series Design Manual", Motorola Inc., 1988

[9] Negrini, R.; Sami, M.; Stefanelli, R.: "Fault-Tolerance Approaches for VLSI/WSI Arrays", Fourth Int. Conf.: Phoenix Conference on Computers and Communications, 1985, S. 460-468.

[10] Nowottnick, P.: "Maßnahmen zur Erhöhung der Zuverlässigkeit einer integrierten Schaltung", Diplomarbeit, Braunschweig, 1991.

[11] Nowottnick, P.; Ernst, R.: "Concurrent Testing for Fault Tolerant VLSI with Functional Block Redundancy", (zur Veröffentlichung eingereicht).

[12] Pradhan, D.K. (editor): "Fault Tolerant Computing - Theory and Techniques", Vol. 1&2, Englewood Cliffs, New Jersey, Prentice Hall International, 1986.

[13] Schnable, G.L.: "Failure Mechanisms in Microelectronic Devices" in: Hakim, E.B. (editor): "Microelectronic Reliability, Vol. 1, Reliability, Test and Diagnostics", Norwood, MA, Artech House, 1989, S. 25-87.