# Erkennung von transienten Fehlern in Schaltungen mit reduzierter Verlustleistung
# Detection of transient faults in circuits with reduced power dissipation

Michael E. Imhof, Institut für Technische Informatik, Universität Stuttgart, D-70569 Stuttgart
Hans-Joachim Wunderlich, Institut für Technische Informatik, Universität Stuttgart, D-70569 Stuttgart
Christian G. Zoellin, Institut für Technische Informatik, Universität Stuttgart, D-70569 Stuttgart

## Kurzfassung / Abstract

Für Speicherfelder sind fehlerkorrigierende Codes die vorherrschende Methode, um akzeptable Fehlerraten zu erreichen. In vielen aktuellen Schaltungen erreicht die Zahl der Speicherelemente in freier Logik die Größenordnung der Zahl von SRAM-Zellen vor wenigen Jahren. Zur Reduktion der Verlustleistung wird häufig der Takt der pegelgesteuerten Speicherelemente unterdrückt und die Speicherelemente müssen ihren Zustand über lange Zeitintervalle halten. Die Notwendigkeit Speicherzellen abzusichern wird zusätzlich durch die Miniaturisierung verstärkt, die zu einer erhöhten Empfindlichkeit der Speicherelemente geführt hat.
Dieser Artikel stellt eine Methode zur fehlertoleranten Anordnung von pegelgesteuerten Speicherelementen vor, die bei unterdrücktem Takt Einfachfehler lokalisieren und Mehrfachfehler erkennen kann. Bei aktiviertem Takt können Einfach- und Mehrfachfehler erkannt werden. Die Register können ähnlich wie Prüfpfade effizient in den Entwurfsgang integriert werden. Die Diagnoseinformation kann auf Modulebene leicht berechnet und genutzt werden.
For memories error correcting codes are the method of choice to guarantee acceptable error rates. In many current designs the number of storage elements in random logic reaches the number of SRAM-cells some years ago. Clock-gating is often employed to reduce the power dissipation of level-sensitive storage elements while the elements have to retain their state over long periods of time. The necessity to protect storage elements is amplified by the miniaturization, which leads to an increased susceptibility of the storage elements.
This article proposes a method for the fault-tolerant arrangement of level-sensitive storage elements, which can locate single faults and detect multiple faults while being clock-gated. With active clock single and multiple faults can be detected. The registers can be efficiently integrated similar to the scan design flow. The diagnostic information can be easily computed and used at module level.

## Schlüsselwörter / Keywords

Robustes Design, Fehlertoleranz, Verlustleistung, Latch, Register, Single Event Effect
Robust design, fault tolerance, power dissipation, latch, register, single event effects

## 1 Einführung

Die Technologieskalierung hat Auswirkungen auf Verlustleistung und Zuverlässigkeit. Der vorliegende Beitrag behandelt die Kombination beider Effekte. Durch Strahlung hervorgerufene Single Event Effects (SEE) waren schon in den 70er Jahren Gegenstand von Untersuchungen bei Speicherzellen [1], heute sind SEEs für statische Speicher [2], Latches und Flip-Flops [3] und selbst für frei angeordnete Logik [4], [5] von Bedeutung.

Die Soft Error Rate (SER) bei Speicherelementen in freier Logik steigt stetig [6] und die Menge der Flip-Flops und Latches wächst überproportional, so daß der Schutz dieser Speicherelemente in neuen Technologien immer mehr Bedeutung gewinnt.

Neben der hohen Anfälligkeit gegenüber SEEs führt die Skalierung zu einer wachsenden Leistungsdichte auf den Chips, welche, wie in den letzten Jahren zu beobachten war, weitere Freuquenzerhöhungen begrenzt. Die klassischen Techniken zur Reduktion der Verlustleistung behalten ihre Relevanz [7], [8], müssen aber von massivem Parallelismus [9] und Power Management in Networks- und Systems-on-a-Chip unterstützt werden. Während die Unterbrechung der Stromzufuhr eingesetzt wird, wenn Module über einen langen Zeitraum nicht benutzt werden und ihr Zustand für weitere Berechnungen nicht mehr benötigt wird, wird das Unterdrücken des Takts für kürzere Pausen bevorzugt, nach denen die Berechnung fortgesetzt wird und somit der Zustand erhalten werden muss.

Es existiert eine Vielzahl von Schemata für den Schutz von Flip-Flops gegen Single-Event-Upsets (SEUs) [10], [11], [12], [13], [3], [14], [15], [16], [17], [18], [19], alle fügen Redundanz ein und bringen zusätzliche Aktivität und zusätzliche Verlustleistung mit sich. Signifikante Fortschritte zur Limitierung der Verlustleis-

tung wurden durch spezielle Designs wie BISER [20], RAZOR [12], DF-DICE [21] und andere gemacht. Allerdings haben diese Schemata nicht die Phase mit unterdrücktem Takt zum Ziel, die aber zumeist eine längere zu schützende Zeitspanne darstellt als die Phase mit aktivem Takt.

Der vorliegende Beitrag stellt ein Verfahren speziell zum Schutz der Phase mit unterdrücktem Takt vor, welches aber auch zur Fehlererkennung mit aktivem Takt eingesetzt werden kann und besonders für Entwürfe mit reduzierter Verlustleistung geeignet ist. Für eine effiziente und automatisierte Implementierung werden Basiszellen entwickelt, die genauso wie Scanelemente zur automatisierten Integration benutzt werden können und einzelne Register effizient mittels Parität schützen. Zur Fehlerlokalisation kann ein ressourcensparender Baum synthetisiert werden, der anzeigt, welcher Registerinhalt verändert wurde und ob ein Neustart nach dem unterdrücken des Takts nötig ist. Kleine Erweiterungen erlauben die Wiederverwendung des Schemas für die Fehlererkennung mit aktivem Takt und den Offline-Test.

Der Hardwaremehraufwand für die eingeführte Technik ist deutlich geringer als der Overhead bekannter Verfahren wie RAZOR oder BISER.

Das nächste Kapitel stellt den grundlegenden Registeraufbau vor und berücksichtigt dabei Leistungs- und Fehlertoleranzaspekte. Kapitel 3 beschreibt die Fehlererkennungs- und Korrekturbäume mit quantitativen Analysen. Kapitel 4 erweitert die Basisregister für die Phase mit aktivem Takt. Kapitel 5 untersucht den LEON Prozessor als Teil eines Systems mit mehreren Kernen, analysiert den für den Einbau des vorgestellen Fehlerdetektionsschemas nötigen Aufwand und vergleicht ihn mit anderen fehlererkennenden Registern aus der Literatur.

# 2 Erkennung von transienten Fehlern auf Gatterebene bei begrenzter Verlustleistung

Beim Entwurf von integrierten Systemen mit niedriger Verlustleistung muss sowohl auf die Schaltaktivität als auch auf Leckströme geachtet werden. Methoden zur Reduzierung der Leckströme, wie mehrere Schwellspannungen (Multi-$V_t$) oder Power Gating arbeiten auf Technologie- bzw. Schaltungsebene und sind nicht Gegenstand dieses Artikels. Methoden zur Reduzierung der Schaltaktivität vermeiden alles, wodurch unnötige Aktivität im Schaltkreis hervorgerufen wird. In gewissem Maß widerspricht dieser Ansatz den gängigen Methoden des robusten Entwurfs beispielsweise durch Fehlertoleranz oder Redundanz.

*Prüfpfadelemente:* Das Level Sensitive Scan Design (LSSD) [22] gilt als besonders robust gegenüber Timing-Variationen. Abbildung 1 zeigt das pegelgesteuerte $L1/L2$ Schieberegister-Latch (SRL), welches von einem Systemtakt und zwei Takten $A$ und $B$ gesteuert wird.

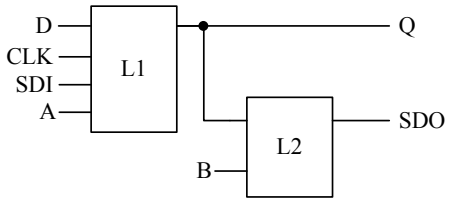


**Bild 1** Schieberegister-Latch

Im einfachsten Ansatz wird das $L1$ Latch zum Speichern der Daten am Eingang $D$ durch den Takt $CLK$ und der Eingang "scan data in" wird von den beiden nicht-überlappenden Taktsignalen $A$ und $B$ gesteuert. Folglich wird das Latch $L2$ nur zur Implementierung des Schiebebetriebs verwendet und wirkt sich somit nachteilig auf die Verlustleistung aus. Es vergrössert die aktive Fläche und damit den Leckstrom, und es erhöht die Last des $L1$ Latches, wodurch sich die Energie für einen Schaltvorgang erhöht. Darum wird in verlustleistungssensitiven Schaltungen häufig ein partieller Prüfpfad verwendet. In Pipelines ohne Rückkopplung wird die Testbarkeit durch einen partiellen Prüfpfad nicht verringert [23], und z.B. im Cell Prozessor sind nur etwa $1/4$ der Latches Teil eines Prüfpfads [24].

Im Systembetrieb müssen die Latches von einem nichtüberlappenden Takt gesteuert werden. Abbildung 2 zeigt den einfachsten Fall. Die Zeiten $\tau_1^l$ and $\tau_1^h$ sind die 0- resp. 1-Phasen von $CLK_1$, $\tau_2^l$ and $\tau_2^h$ sind die Phasen von $CLK_2$, und beide sind niemals zur selben Zeit auf dem 1-Pegel ($\tau_a$, $\tau_b$).

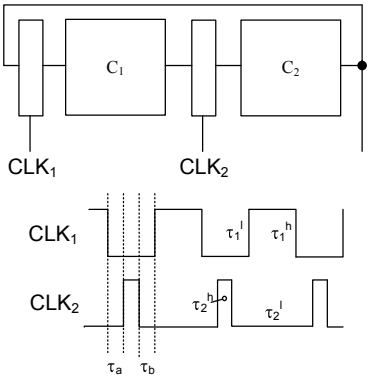


**Bild 2** Nicht-überlappendes Takt-Schema

Diese Latches können zu sogenannten $L1/L2^*$ Latches kombiniert werden um ein einzelnes SRL zu bilden (Abbildung 3). Dadurch ist das $L2$ Latch nicht mehr redundant und vergrössert weder die aktive Fläche noch den Leckstrom.

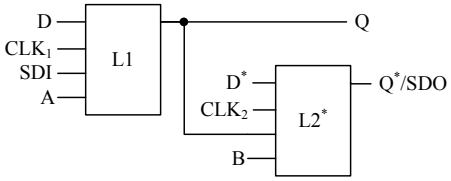


**Bild 3** $L1/L2^*$ Schieberegister-Latch

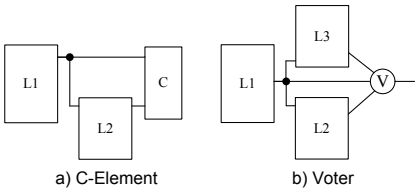


**Bild 4** Prinzip des robusten Latch-Designs

*Robuste und fehlertolerante Register:* Schaltungen können auf allen Design-Ebenen gehärtet werden, jedoch existiert in vielen Fällen eine erhebliche Auswirkung auf Leck- und Schaltströme. Auf der physikalischen Ebene kann etwa durch die Auslegung von Zellen- und Transistorgrößen oder das Hinzufügen von Dioden und Kapazitäten die Soft Error Rate (SER) vermindert werden [25], [26], wobei sich jedoch sowohl die Schaltströme als auch die Leckströme vergrössern. Auf Transistor- und Gatterebene werden redundante Speicherelemente hinzugefügt und ein Muller-C-Element oder ein Voter sichern den Ausgang. Ein beispiel ist das BISER (Built-In Soft Error Correction) Element [3], und zahlreiche Vorgänger und Variationen finden sich in der Literatur [17], [14], [10], [13], [15], [18], [19], [12].

Das Grundprinzip aller dieser Strukturen besteht darin, zwei weitere Speicherelemente hinzuzufügen. Eines der Speicherelemente kann ein C-Element sein, andernfalls wird ein zusätzlicher Voter benötigt (Abbildung 4).

Falls das $L2$ Latch ohnehin nur für den Prüfpfad verwendet wird, dann ist der Overhead einer solchen Struktur im Bezug auf Fläche, Verlustleistung und Signalverzögerung begrenzt. Falls das Latch jedoch nicht Teil eines Prüfpfads ist und wenn zumindest ein partieller Prüfpfad verwendet wird, kann der Mehraufwand für eine Lösung nach Abbildung 4 auf das Siebenfache steigen. Auch bei Verwendung einer $L1/L2^*$ Struktur ist eine direkte Implementierung unmöglich. Dieser Aufwand wird nicht reduziert, indem die Register durch einen Hammingcode geschützt werden [27].

*Erkennung transienter Fehler:* Wenn auf Registerebene keine Fehlertoleranz und Fehlerkorrektur sondern nur Fehlererkennung benötigt wird, ist eine Implementierung mit wesentlich geringeren Auswirkungen auf Verlustleistung, Signalverzögerung und Fläche möglich. Dann genügt es, ein einfaches Paritätssignal über ein 8 oder 16 Bit Register zu berechnen, und den Paritätsbaum durch das invertierte Clock-Gating Signal zu steuern, um die Schaltaktivität während des Betriebs zu reduzieren. Die Paritätsgenerierung ist Teil der Latchzelle und erlaubt die Synthese durch einfaches Aneinanderreihen (engl. Abutment). Das abschliessende Verdrahten der Paritätssignale unterscheidet sich nicht von der Verdrahtung des Prüfpfads. Abbildung 5 zeigt das Schaltbild der Paritätsberechnung von 2 Latches.

Mit diesen Paritätspaar-Latches (PPL) wird ein Register wie in Abbildung 6 gebildet. Da die Paritätssignale ohnehin verstärkt werden müssen, ist der Aufwand für das oben erwähnte Gating nur der Unterschied zwischen einem Inverter und einem NAND-Gatter. Für den vollständigen Baum werden das PPL und ein weiteres XOR-Gatter wie in Abbildung 6 benötigt. Zelltyp $A$ enthält ein Latch und die für das PPL nötige zusätzliche Logik während Zelltyp $B$ ein Latch und ein weiteres XOR-Gatter für den internen Paritätsbaum enthält.

Mit dieser Struktur besteht jede Zelle nur aus einem Latch und zwei halben XOR-Gattern, wobei höchstens zwei Leitungen über jede Zelle verlaufen. Der kritische Pfad besteht aus 4 Invertern, 1 NAND-Gatter und 3 Pass-Transistoren, und ist somit höchstens das Dreifache der Verzögerungszeit durch ein Latch und in der gleichen Grössenordnung wie die oben erwähnten Standardlösungen. Es werden lediglich zwei verschiedene Zelltypen benötigt, und die Registersynthese ist nicht aufwändiger als die übliche Prüfpfadsynthese. Indem zwei Register direkt verbunden werden, können 16-Bit Register entworfen werden, für die nur ein weiteres XOR-Gatter hinzugefügt wird (Abbildung 7).

# 3 Lokalisierung von transienten Fehlern auf Modulebene

Die Fehlerinformation der Register muss zur obersten Ebene des Moduls weitergeleitet werden. Sie wird dort in einem Register gespeichert und zur Fehlerbehandlung verwendet. Falls nur Fehlererkennung benötigt wird, um Berechnungen als Rollback neu zu starten, reicht ein einfacher XOR-Baum, um alle Register zu verbinden. Zusätzlicher Aufwand ist erforder-

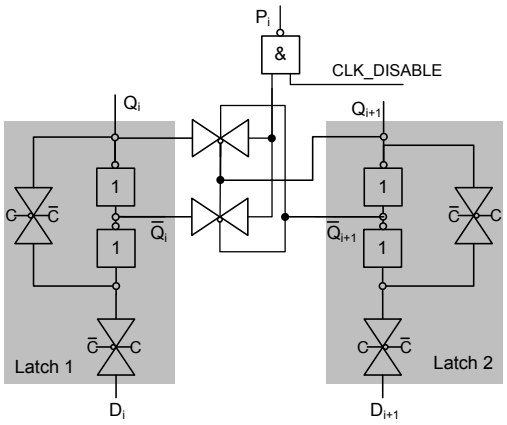


**Bild 5** Paritätsberechnung zwischen zwei Latches

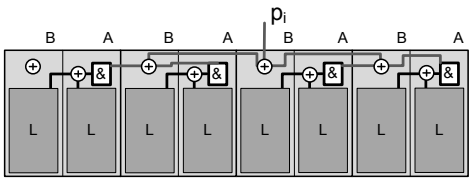


**Bild 6** Paritätsbaum, bestehend aus zwei Zelltypen

lich, wenn auch Mehrfachfehler erkannt werden sollen, oder wenn Fehlerlokalisierung benötigt wird.

In eingebetteten Speichern kann der transparente Online-Test zur Fehlerlokalisierung verwendet werden [28], [29], der im Folgenden an freie Logik angepasst wird. Die Modulo-2 Addresscharakteristik eines bit-orientierten Speichers ist die bitweise XOR-Verknüpfung der Adressen, deren Speicherzellen eine 1 enthalten (Abbildung 8).

Die Charakteristik hat einige wesentliche Vorteile im Vergleich zur Signaturanalyse oder fehlerkorrigierenden Codes [30], [31]. So ist die Lokalisierung von Einfachfehlern besonders einfach: Angenommen $c_k$ ist die Charakteristik des korrekten Speicherinhalts, und $c_f$ ist die Charakteristik eines Fehlers in einem einzigen Bit. Die bit-weise Berechnung von $add_f = c_k \oplus c_f$ erhält die Adresse des fehlerhaften Bits. Doppelfehler werden immer erkannt und die Aliasing-Wahrscheinlichkeit für beliebige Fehler ist $2^{-N}$, wobei $N$ die Zahl der Adressbits ist.

## 3.1 Charakteristik-Berechnung

Wenn alle Register des Moduls von $r_1$ bis $r_{N-1}$ numeriert sind, könnte die Gesamtcharakteristik sequentiell, Bit für Bit, berechnet werden, wie in Abbildung 8 dargestellt. Jedoch ist die sequentielle Berechnung der Charakteristik nur für Speicherfelder geeignet, und in freier Logik mit Clock-Gating ist eine kombinatorische Logik erforderlich. Es sei $p_i$ die Parität des Registers

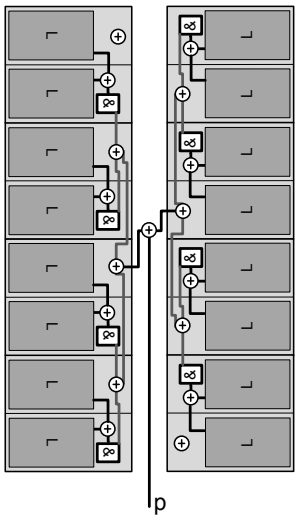


**Bild 7** Fehlererkennendes 16-Bit Register

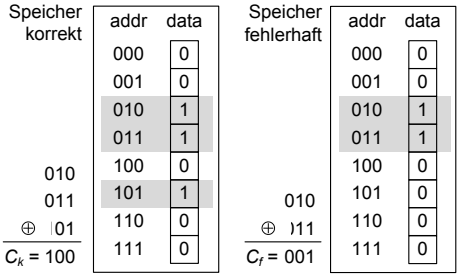


**Bild 8** Modulo-2 Adresscharakteristik

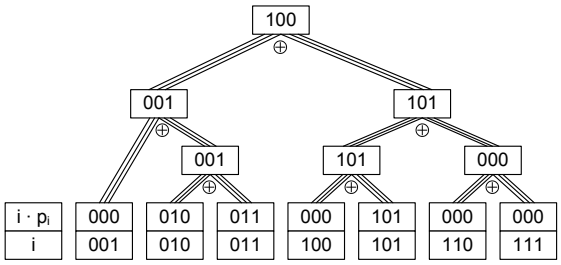


**Bild 9** Sub-optimale Berechnung der Modulo-2 Charakteristik

$r_i$. Die N-bit Worte $p_i \wedge i$ entsprechen der Registeradresse $i$ falls $p_i$ gleich 1 ist, ansonsten sind sie 0. Sie bilden die Blätter eines Baumes, dessen Knoten eine bitweise XOR-Operation implementieren. Abbildung 9 veranschaulicht dies für 7 Register und die Paritätsbits $p_1, \ldots, p_7$.

Die Zahl der Knoten im Baum in Abbildung 9 ist $2^{N+1} - N$, da Adresse 0 nicht benutzt wird. Da die Wurzel nicht weiter verbunden werden muss, gibt es

$$N \cdot (2^{N+1} - N - 1) \tag{1}$$

Punkt-zu-Punkt Verbindungen. Ein erheblicher Teil der Hardware kann gespart werden, wenn nur zählende Bits zwischen den Ebenen weitergereicht werden (Abbildung 10).

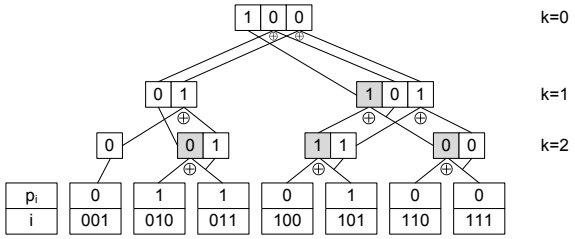


**Bild 10** Optimale Organisation des Baums

## 3.2 Hardwareoptimale Berechnung

In Ebene $k \in \{0, \ldots, N-1\}$ gibt es $2^k$ Knoten und jeder Knoten $v_{k,l}$, $l \in \{0, \ldots, 2^k - 1\}$ entspricht $2^{N-k}$ Adressen. $c_{k,l} = (c_{k,l,N-1}, \ldots, c_{k,l,0})$ ist der Ausgang von Knoten $v_{k,l}$. Da die $k$ höchstwertigen Bits $(a_{N-1}, \ldots, a_{N-k+1})$ der Adressen der Vorgänger von $v_{k,l}$ identisch sind, hängen die Ergebnisbits $(c_{k,i,N-1}, \ldots, c_{k,i,N-k+1})$ nur von der Parität der $2^{N-k}$ Blattknoten bzw. Registerparitäten ab:

- $(c_{k,l,N-1}, \ldots, c_{k,l,N-k+1}) = (0, 0, \ldots, 0)$
  wenn die Parität 0 ist
- $(c_{k,l,N-1}, \ldots, c_{k,l,N-k+1}) = (a_{N-1}, \ldots, a_{N-k+1})$
  wenn die Parität 1 ist

Es muss nicht der ganze Vektor von $k$ Bits berechnet werden. Stattdessen, kann die Generierung dieser Information zur nachfolgenden Ebene $k-1$ delegiert werden, und es muss hier nur die Parität berechnet und weitergeleitet werden:

$$p_{k,l} = p_{k+1,2\cdot l} \oplus p_{k+1,2\cdot l+1}$$

Die $N-k-1$ bits $(c_{k,l,N-k-2}, \ldots, c_{k,l,0})$ werden aus der Charakteristik der Vorgänger von $v_{k,l}$ berechnet:

$$c_{k,l,j} = c_{k+1,2\cdot l,j} \oplus c_{k+1,2\cdot l+1,j}$$
$$j \in \{N-k-2, \ldots, 0\}$$

Schließlich korrespondiert $c_{k,l,N-k-1}$ zum höchstwertigen Adressbit, das zwischen den beiden Vorgängern von $v_{k,l}$ unterscheidet. Aus den vorhergehenden Betrachtungen wissen wir, dass jedes höherwertige Bit der Charakteristik aus den Paritätsbits $p_{k,2\cdot l}$ and $p_{k,2\cdot l+1}$ abgeleitet werden kann. Hier wissen wir, dass Addressbit $N-k-1$ des Vorgängers $v_{k,2\cdot l}$ eine 0 ist, und entsprechend 1 für $v_{k,2\cdot l+1}$. Es folgt:

$$\begin{aligned} c_{k,l,N-k-1} &= (0 \wedge p_{k+1,2\cdot l}) \oplus (1 \wedge p_{k+1,2\cdot l+1}) \\ &= p_{k,2\cdot l+1} \end{aligned}$$

Jeder Knoten auf Ebene $k$ hat 2 Verbindungen, um die Parität der beiden Vorgänger zu lesen, und $2\cdot(N-k-1)$ Verbindungen, um die Charakteristik der Vorgänger zu lesen. Folglich ist die Zahl der Verbindungen auf Ebene $k$ gleich $2^k \cdot 2 \cdot (N-k)$.

Da das niederstwertige Partitätsbit nicht zur Gesamtcharakteristik beiträgt, muss diese Information nicht berechnet werden (siehe Abbildung 10). Die Gesamtzahl der Punkt-zu-Punkt Verbindungen ist:

$$\sum_{k=0}^{N-1} 2^{k+1} \cdot (N-k) - N$$

Der Summenanteil in dieser Gleichung kann nun als die Evaluierung der Potenzreihe $p(x) = \sum_{k=0}^{N-1} x^{k+1} \cdot (N-k)$ für $x = 2$ betrachtet werden. Durch Umformen erhält man eine Summe von geometrischen Reihen, und kommt schließlich zu folgender geschlossenen Form für die Gesamtzahl der Punkt-zu-Punkt Verbindungen:

$$2^{N+2} - 3 \cdot N - 4 \qquad (2)$$

Dies ist erheblich geringer als (1).

# 4 Online-Test und Produktionstest

Die bisher vorgestellte Struktur kann auch bei aktiviertem Taktsignal verwendet werden, wenn die Register aus Abschnitt 2 geringfügig erweitert werden. Für LSSD-Strukturen, wie sie hier behandelt werden, wurde die sogenannte GRAAL [32] Struktur entwickelt, die jedem Systemlatch ein Schattenlatch hinzufügt und die Zustände beider Latches vergleicht (Abbildung 11).

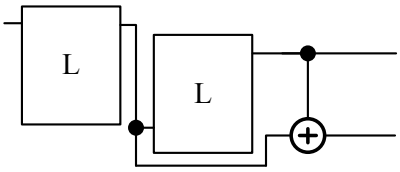


**Bild 11** GRAAL

Die Grundidee des hier vorgestellten Ansatzes ist es, nicht die Registerinhalte zu vergleichen, sondern nur deren Parität (Abbildung 12).

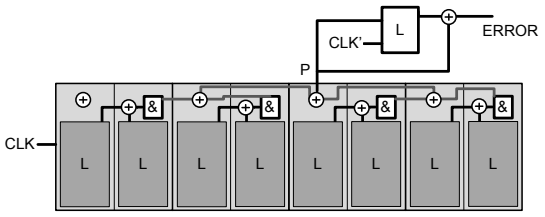


**Bild 12** Fehlererkennung durch Paritätsprüfung

Ein zusätzliches Latch pro Register speichert das Paritätsbit, das für die Charakteristik ohnehin berechnet wird. Falls das Paritätslatch mit dem gleichen Takt betrieben wie das Register, wird der kritische Pfad verlängert und die Frequenz geringfügig beeinträchtigt. Dieser Nachteil kann ohne Verlust an Fehlerschutz behoben werden, indem $CLK'$ wie bei üblichem "Time borrowing" aus $CLK$ mit einer Verzögerung abgeleitet wird, die dem kritischen Pfad durch den Paritätsbaum entspricht. Nachdem die Parität berechnet wurde, wird der gespeicherte Wert mit dem gleichzeitig durch den XOR-Baum kontinuierlich neu berechneten Wert verglichen. Dazu erzeugt ein weiteres XOR-Gatter die Fehlerinformation auf Registerebene. Dieses Fehlersignal kann auf die gleiche Art und Weise für eine Fehlerbehandlung verwendet werden, wie beim RAZOR-Ansatz.

Die vorgestellte, fehlererkennende Struktur ist besonders geeignet für Schaltungen mit partiellem Prüfpfad und kann zusätzlich für den Offline-Test verwendet werden. Die Charakteristik auf Modul-Ebene erhöht die Beobachtbarkeit aller Latche und komprimiert die Test-Antwort.

# 5 Experimentelle Ergebnisse

Im Folgenden wird der Aufwand untersucht, der durch die vorgestellte Fehlerlokalisierung in eingebetten Prozessorkernen für Multiprozessor System on Chip (MPSoC) wie z.B. beim Leon3 entsteht. Wir vergleichen die in diesem Artikel vorgestellte Struktur mit den fehlermaskierenden Flipflops aus [3] und dem fehlererkennenden RAZOR Schattenregister aus [12], das zwar für die Erkennung und Maskierung von Timing Fehlern entwickelt wurde, aber für SEEs verwendet werden kann. Für jede Methode wird die Zahl von Transistoren in der gesamten notwendigen Logik geschätzt und verglichen. Dadurch ergeben sich entsprechende Implikationen für die dynamische Verlustleistung und besonders für die statische Verlustleistung, welche jedoch hier noch nicht speziell betrachtet werden.

Leon3 ist ein hochgradig modularer Prozessorkern, der es erlaubt einzelne Komponenten getrennt auszuwählen, wie z.B. FPU, MMU, Cache, Speicherkontroller, etc.. Die hier verwendete Konfiguration hat etwa 16k Speicherelemente in freier Logik. Die Speicherelemente wurden in 2047 Register von bis zu 8 Bit Breite segmentiert. Die hier verwendeten Latche benötigen 8 Transistoren. Der Aufwand für Taktgenerierung und Verteilung wird nicht berücksichtigt.

Tabelle 1 zeigt den für die hier vorgestellte Struktur nötigen Mehraufwand. Die Spalte *# Trans* enthält für jeden der verwendeten Gattertypen (Spalte *Gattertyp*) jeweils die Zahl der Transistoren. Die Spalte *# Gatter* enthält entsprechend die Anzahl der Gatter des jeweiligen Typs und die zugehörige Gesamtzahl der Transistoren (Spalte *Gesamt*).

| | Gattertyp | # Trans. | # Gatter | Gesamt |
|---|---|---|---|---|
| Pro Register | PPL | 24 | 4 | 96 |
| | XOR | 6 | 3 | 18 |
| | Gesamt | | | 114 |
| Alle Register | | 114 | 2047 | 233358 |
| Characteristik | XOR | 6 | 4072 | 24432 |
| Komparator 11bit | Latch | 8 | 11 | 88 |
| | XOR | 6 | 11 | 66 |
| | OR | 4 | 10 | 40 |
| | Gesamt | | | 194 |
| Online-Test | Latch | 8 | 2047 | 16376 |
| | XOR | 6 | 2047 | 12282 |
| | OR Baum | 4 | 2046 | 8184 |
| | Gesamt | | | 36842 |
| Gesamt (o. online) | | | | 257984 |
| Gesamt (mit online) | | | | 294826 |

**Tabelle 1** Transistorzahl für die vorgestellte Methode

Jedes Register von 8 bit Breite besteht aus 4 PPL und 3 zusätzlichen XOR-Gattern, wie in Abbildung 6. Die Zahl der XOR-Gatter, die benötigt werden, um die Gesamtcharakteristik zu berechnen, wird wie in Kapitel 2 beschrieben bestimmt, und deckt sich mit den Syntheseergebnissen. Ein einfacher Komparator vergleicht die Referenzcharakteristik und eine evtl. fehlerbehaftete Charakteristik. Der letzte Abschnitt in Tabelle 1 betrifft den Aufwand für die Online-Fehlererkennung, wie sie in Abschnitt 4 vorgestellt wurde. Um auf Core-Ebene ein Signal zu erhalten, dass ein Fehler aufgetreten ist, müssen die $ERROR$ Leitungen durch einen einfachen OR-Baum verknüpft werden. Da die Online-Fehlererkennung optional ist, ist der Aufwand für beide Varianten angegeben.

In den Tabellen 2 und 3 wird die gleiche Information für die RAZOR and BISER Strukturen aufgelistet. Bei RAZOR wurde die Zahl der Transistoren für die Implementierung aus [12] ermittelt. Jedes Latch der Schaltung wird auf diese Art implementiert. RAZOR wurde ursprünglich für die Erkennung und Korrektur von Timing-Fehlern entworfen, kann jedoch auch zur Erkennung von SEUs verwendet werden. Bei RAZOR wird ein einfacher OR-Baum benötigt um ein Fehlersignal auf Core-Ebene zu erhalten, in diesem Fall jedoch über alle Latches statt nur über die Register. Die BISER Methode wurde für Fehlermaskierung statt Fehlerdetektierung entworfen. Dadurch kann der Aufwand für einen OR-Baum vermieden werden. Die hier betrachtete Implementierung von BISER wurde dem Artikel in [3] entnommen. BISER basiert auf der Wiederverwendung von Strukturen in Registern für den At-Speed Scan-Test. Da wie bereits erwähnt im Low-Power Design nur möglichst wenige Register mit LSSD o.ä. ausgestattet werden sollen, wird diese Austattung mit eingerechnet. Kürzlich vorgestellte Variationen von BISER [19] enthalten zusätzliche Funktionalität wie etwa Alterungssensoren, benötigen jedoch noch zusätzliche Fläche.

| | Gattertyp | # Trans. | # Gatter | Gesamt |
|---|---|---|---|---|
| Pro Latch | INV | 2 | 11 | 22 |
| | XOR | 6 | 1 | 6 |
| | OR | 4 | 1 | 4 |
| | MUX | 4 | 1 | 4 |
| | Transm. Gate | 2 | 3 | 6 |
| | AND | 4 | 1 | 4 |
| | Gesamt | | | 46 |
| Pro Register | | 46 | 8 | 368 |
| Alle Register | | 368 | 2047 | 753296 |
| OR Baum | OR | 4 | 16376 | 65504 |
| Gesamt | | | | 818800 |

**Tabelle 2** Transistorzahl bei einer RAZOR-Implementierung

| | Gattertyp | # Trans. | # Gatter | Gesamt |
|---|---|---|---|---|
| Pro Latch | Non-scan Latch | 8 | 2 | 16 |
| | Scan Latch | 20 | 2 | 40 |
| | C-Element | 6 | 1 | 6 |
| | INV | 2 | 5 | 10 |
| | AND | 4 | 1 | 4 |
| | OR | 4 | 1 | 4 |
| | Total | | | 80 |
| Pro Register | | 80 | 8 | 640 |
| Gesamt | | 640 | 2047 | 1310080 |

**Tabelle 3** Transistoranzahl bei einer BISER-Implementierung

Aus diesen Analysen zeigt sich, dass der Aufwand für die vorgestellte Methode um eine Grössenordnung geringer ist als für die Standardmethoden, die auf Verdoppelung oder Verdreifachung beruhen. Im Vergleich zur vollständigen Implementierung der hier vorgestellten Methode einschliesslich der Online-Fehlererkennung benötigt die Schaltung mit RAZOR 177% mehr Transistoren und das selbe Design mit BISER sogar 344% mehr Transistoren. Wenn die Online-Fehlererkennung nicht benötigt wird, verschieben sich diese Zahlen nochmals und RAZOR hat nun entsprechend 217% und BISER 407% mehr Transistoren.

Die geringere Zahl von Transistoren hat direkten Einfluss auf die Siliziumfläche und damit ebenfalls auf die Leistungsaufnahme, was besonders für die gezeigten Anwendungen relevant ist. Darüber hinaus ist der Einfluss auf die Schaltaktivität durch die vorgestellte Technik äusserst gering, da das im PPL integrierte NAND-Gatter unnötige Schaltvorgänge im Charakteristik-Baum vermeidet.

# 6 Zusammenfassung

Das in diesem Artikel vorgestellte Verfahren zur Fehlererkennung und -lokalisierung überträgt die Vorteile von fehlerkorrigierender Kodierung bei Speicherfeldern auf Speicherlemente in freier Logik. Die Methode zielt im besonderen auf Schaltungsentwürfe die zur Veringerung der Verlustleistung den Takt vorübergehend abschalten, wodurch die Speicherelemente den Zustand über einen langen Zeitraum halten müssen.

Das Verfahren ermöglicht es, das von einem transienten Fehler betroffene Register zu identifizieren. Dazu werden wenige Zellen benötigt, die einfach in den Entwurfsgang integriert werden können. Die Gesamtstruktur kann effizient für den Offline- und Online-Test weiterverwendet werden. Im Vergleich zu anderen robusten Speicherelementen, erfordert die vorgestellte Struktur bis zu 77% weniger Transistoren.

# 7 Danksagung

Diese Arbeit wurde von der Deutschen Forschungsgesellschaft im Projekt Realtest unter dem Zeichen Wu245/5-1 gefördert.

# 8 Literatur

[1] J. Ziegler, H. Curtis, H. Muhlfeld, C. Montrose, B. Chin, M. Nicewicz, C. Russell, W. Wang, L. Freeman, P. Hosier, *et al.*, "IBM experiments in soft fails in computer electronics (1978-1994)," *Journal of Research*, vol. 40, no. 1, 1998.

[2] B. Gill, M. Nicolaidis, and C. Papachristou, "Radiation Induced Single-Word Multiple-Bit Upsets Correction in SRAM," in *11th IEEE International On-Line Test Symposium (IOLTS)*, 2005, pp. 266–271.

[3] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *IEEE Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[4] A. Nieuwland, S. Jasarevic, and G. Jerin, "Combinational Logic Soft Error Analysis and Protection," *12th IEEE International On-Line Test Symposium (IOLTS)*, pp. 99–104, 2006.

[5] M. Nicolaidis, "Design for soft error mitigation," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 405–418, 2005.

[6] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.

[7] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," *32nd Conference on Design Automation (DAC), San Francisco, CA, USA, June 12-16, 1995*, pp. 242–247, 1995.

[8] M. Pedram and J. Rabaey, *Power Aware Design Methodologies*. Kluwer Academic Publishers Norwell, MA, USA, 2002.

[9] S. Borkar, "Thousand core chipsa technology perspective," in *Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, 2007*, 2007, pp. 746–749.

[10] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology," *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, pp. 2874–2878, 1996.

[11] T. Monnier, F. Roche, and G. Cathebras, "Flipflop hardening for space applications," *Intl. Workshop on Memory Technology*, 1998.

[12] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation," *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 7–18, 2003.

[13] S. Krishnamohan and N. Mahapatra, "A highly-efficient technique for reducing soft errors in static CMOS circuits," *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*, pp. 126–131, 2004.

[14] M. Nicolaidis, "Design for mitigation of single event effects," in *IOLTS*. IEEE Computer Society, 2005, pp. 95–96.

[15] A. Drake, A. KleinOsowski, and A. Martin, "A Self-Correcting Soft Error Tolerant Flop-Flop," *12th NASA Symposium on VLSI Design, Coeur dAlene, Idaho, USA, Oct*, pp. 4–5, 2005.

[16] A. Goel, S. Bhunia, H. Mahmoodi, and K. Roy, "Low-overhead design of soft-error-tolerant scan flip-flops with enhanced-scan capability," *Proceedings of the 2006 conference on Asia South Pacific design automation*, pp. 665–670, 2006.

[17] M. Omaña, D. Rossi, and C. Metra, "Latch susceptibility to transient faults and new hardening approach," *IEEE Trans. Computers*, vol. 56, no. 9, pp. 1255–1268, 2007.

[18] M. Fazeli, A. Patooghy, S. Miremadi, and A. Ejlali, "Feedback Redundancy: A Power Efficient SEU-Tolerant Latch Design for Deep Sub-Micron Technologies," *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 276–285, 2007.

[19] M. Zhang, T. Mak, J. Tschanz, K. Kim, N. Seifert, and D. Lu, "Design for Resilience to Soft Errors and Variations," *Proceedings of the 13th IEEE International On-Line Testing Symposium*, pp. 23–28, 2007.

[20] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel, "Sequential element design with built-in soft error resilience," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 12, pp. 1368–1378, 2006.

[21] R. Naseer and J. Draper, "The DF-dice storage element for immunity to soft errors," *Proceedings of the 48th IEEE International Midwest Symposium on Circuits and Systems, August*, 2005.

[22] E. Eichelberger and T. Williams, "A logic design structure for LSI testability," *Proceedings of the 14th design automation conference*, pp. 462–468, 1977.

[23] H.-J. Wunderlich and A. Kunzmann, "An analytical approach to the partial scan problem," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 1, no. 2, pp. 163–174, 1990.

[24] O. Takahashi, S. R. Cottier, S. H. Dhong, B. K. Flachs, and J. Silberman, "Power-conscious design of the cell processor's synergistic processor element," *IEEE Micro*, vol. 25, no. 5, pp. 10–18, 2005.

[25] M. R. Choudhury, Q. Zhou, and K. Mohanram, "Design optimization for single-event upset robustness using simultaneous dual-vdd and sizing techniques," in *2006 International Conference on Computer-Aided Design (ICCAD'06), November 5-9, 2006, San Jose, CA, USA*, 2006, pp. 204–209.

[26] R. Garg, N. Jayakumar, S. Khatri, and G. Choi, "A design approach for radiation-hard digital electronics," in *Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*, 2006, pp. 773–778.

[27] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy," *Proceedings of the 15th Symposium on Integrated Circuits and Systems Design*, p. 95, 2002.

[28] S. Hellebrand, H.-J. Wunderlich, A. A. Ivaniuk, Y. V. Klimets, and V. N. Yarmolik, "Efficient online and offline testing of embedded drams," *IEEE Trans. Computers*, vol. 51, no. 7, pp. 801–809, 2002.

[29] S. Boutobza, M. Nicolaidis, K. M. Lamara, and A. Costa, "A transparent based programmable memory bist," in *11th European Test Symposium (ETS 2006), 21-24 May 2006, Southhampton, UK*, 2006, pp. 89–96.

[30] T. Damarla, C. Stroud, and A. Sathaye, "Multiple error detection and identification via signature analysis," *Journal of Electronic Testing (JETTA)*, vol. 7, no. 3, pp. 193–207, 1995.

[31] R. W. Hamming, "Error Correcting and Error Detecting Codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[32] M. Nicolaidis, "Graal: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies," *Test Conference, 2007. ITC 2007. IEEE International*, pp. 1–10, 21-26 Oct. 2007.