# Digit-On-Line-Architekturen und VHDL-Cores für die Umsetzung von schnellen seriellen MSD-First-Signalverarbeitungsalgorithmen

*Steffen Dolling, Dirk Timmermann, Andreas Wassatsch*

Universität Rostock,

Fachbereich Elektrotechnik und Informationstechnik,

Institut für Angewandte Mikroelektronik und Datentechnik,

R.-Wagner-Str. 31, D-18119 Rostock,

email: dol@baltic.e-technik.uni-rostock.de

## Abstract

In dieser Arbeit werden Digit-On-Line-Module für eine schnelle serielle Signalverarbeitung vorgestellt, die nach dem Most-Significant-Digit-First-Prinzip arbeiten. Diese gestattet im Gegensatz zu der bekannten Least-Significant-Digit-First-Technik, die nur für die Addition, Subtraktion und Multiplikation ohne Einschränkungen zu verwenden ist, die Implementierung nahezu aller praktisch vorkommenden Funktionen. Üblicherweise können, verglichen mit parallel kommunizierenden Strukturen, für kaskadierbare Aufgabenstellungen i.a. niedrigere Latenzzeiten erreicht werden. Somit wird auch der Einsatz in Multiprozessorsystemen und allgemein verteilten Systemen vorteilhaft. Die den Modulen zugrunde liegenden Algorithmen werden auf die Funktionalität generischer Grundzellen zurückgeführt, die als VHDL-Modelle beschrieben wurden. In einem entsprechenden VLSI-Entwicklungszyklus lassen sich aus diesen flexible parametrierbare Funktionsblöcke bilden. Durch die modulare Anordnung dieser Blöcke sind konfigurierbare Digit-On-Line-Elemente für die Umsetzung der elementaren arithmetischen Funktionen realisierbar, die sich wiederum zur Abbildung komplexer Operationen verbinden lassen. Diese synthetisierbare Modulbibliothek wurde zur Verifikation auf eine ASIC-Technologie abgebildet.

## 1. Einleitung

Bei der Realisierung von Signalverarbeitungs-Anwendungen mit einem großen Rechen- oder Kommunikationsbedarf werden vielfach arithmetische Strukturen verwendet, die die Eingabe der Operanden und die Ausgabe des Ergebnisses in serieller Form gestatten. Zur effizienten Implementierung der on-chip Signalführung, der inter-chip Kommunikation und einer modularen Problempartitionierung wird häufig der Einsatz von Rechenelementen mit zeichenseri-

eller, least-significant-digit-first Arithmetik (LSDF) vorgeschlagen [1,2]. Die Vorteile dieser gebräuchlichsten Serialisierung für arithmetische Operationen sind die einfache Realisierung einiger Grundoperationen wie Addition und Multiplikation sowie die leichte Pipelinebarkeit. Nachteilig ist jedoch, daß bei gleichzeitiger Übertragung von *k* Bits eines *n*-Bit Wortes die effektive Übertragungsrate bei einer durch die Technologie gegebenen maximalen Taktrate $f_T$ nur $k \cdot f_T / n$ beträgt. Außerdem kann die Berechnung bei intern entscheidungsbasierten Funktionen (Division, Quadratwurzel, etc.) im Empfänger erst starten, wenn alle Operandenbits empfangen wurden. Da in diesen Fällen eine Überlappung aufeinanderfolgender Operationen nicht möglich ist, steigt die Latenzzeit ganz erheblich an. Die Anwendbarkeit von LSDF-Algorithmen ist dann auch naturgemäß bei der Implementierung entscheidungsgestützter Systemanwendungen mit Performanceeinbußen behaftet. Ebenso entstehen gravierende Probleme, da einige Grundoperationen wie die Division ihre Resultate most-significant-digit-first (MSDF) ausgeben und daher eine interne, zeitraubende Umwandlung in die LSDF-Form erfordern.
Aus den geschilderten Gründen sind serielle Kommunikationsformen im LSDF-Format für den VLSI-Einsatz häufig auf spezielle Anwendungen beschränkt. Eine effiziente Alternative dazu bilden arithmetische Systeme, deren interne Struktur die Eingabe der Operanden und die Ausgabe des Ergebnisses im MSDF-Format gestatten. Diese ermöglichen die Implementierung nahezu aller praktisch vorkommenden Funktionen. Damit eröffnet sich, anders als bei LSDF-Algorithmen, eine geschlossene Methodik zur Umsetzung vieler Systemanwendungen. Als eine frappierende Eigenschaft dieser sogenannten Digit-On-Line-Algorithmen gilt, daß sie bei stark kaskadierten Strukturen eine insgesamt niedrigere Latenzzeit verglichen mit parallel rechnenden Strukturen aufweisen. In den folgenden Ausführungen werden zunächst die Eigenschaften der Digit-Online-Algorithmen dargelegt. Im Anschluß wird eine umfangreiche, in VHDL entwickelte und voll synthetisierbare Modulbibliothek beschrieben, die eine Portierung von bisher bitparallel ausgelegten Algorithmen und Architekturen auf die MSDF-Form ermöglicht.

## 2. Eigenschaften von Digit-On-Line-Algorithmen

Eine Voraussetzung für die Digit-On-Line-Arithmetik ist eine redundante Zahlendarstellung, Diese erlaubt Flexibilität in der Wahl der Resultatzeichen, um, wenn nötig, noch in den niederwertigen Zeichen geringe Ergebniskorrekturen zu algebraisch kleineren oder größeren Werten durchzuführen. Sie ist dadurch gekennzeichnet, daß jede Stelle einer redundanten Zahl mit der Basis *r* mehr als *r* verschiedene ganzzahlige Werte annehmen kann. Ein Element $x_i$ eines Digitsatzes kann entsprechend folgende Werte haben:

$$x_i \in \left\{\overline{(r-1)}, \overline{(r-2)}, \ldots, \bar{1}, 0, 1, \ldots, (r-1)\right\} \quad mit \quad \bar{i} = -i \tag{1}$$

Die Wahl der Basis *r* gestattet dabei einen frei wählbaren Tradeoff zwischen einer geringen Bandbreite und einem hohen Datendurchsatz. In diesem Zusammenhang ist der erscheinende Nachteil des erhöhten Aufwands in Form mehrerer Bits zur Darstellung eines Digits einer differenzierten Betrachtung zu unterziehen. Auf der einen Seite ist eine Reduktion der Redundanz durch eine Beschränkung wie folgt erreichbar:

$$x_i \in \left\{\bar{a}, \overline{(a-1)}, \ldots, \bar{1}, 0, 1, \ldots, a\right\} \quad mit \quad \left\lceil \frac{r-1}{2} \right\rceil \le a \le r-1 \tag{2}$$

Andererseits werden arithmetische Operationen wie z.B. die Division im Empfänger und Sender aus Zeitgründen ohnehin intern in einer redundanten Zahlendarstellung durchgeführt, die anschließend vor der Weiterverarbeitung mit einer zumindest $\log_2(n)$-Zeiteinheiten erfordernden Operation in ein nichtredundantes Format umgewandelt werden müssen. Diese zeitaufwendige Operation läßt sich beim Weiterverarbeiten in redundanter Form einsparen.
Sowohl Operanden als auch Ergebnisse werden nach dem MSDF-Prinzip abgearbeitet. Dies ermöglicht eine Kommunikation mit der gewünscht niedrigen Bandbreite, weil nicht alle Zeichen des Operanden gleichzeitig am Eingang anliegen müssen. Dabei benötigen die On-Line-Verfahren $\delta$ anliegende Digits des Eingangsoperanden, bevor die erste Ziffer des Ergebnisses generiert werden kann. Für die Leistungsfähigkeit des Verfahrens ist dabei entscheidend, daß $\delta$ allgemein wesentlich kleiner als die Operandenlänge *n* ist. Der Wert $\delta$ wird gemeinhin als On-Line-Delay bezeichnet und ist lediglich von der Operation und der Zahlenbasis *r*, aber nicht von *n* abhängig. Daraus resultiert ein weiterer Vorteil - die genauigkeitsunabhängige Latenzzeit. Abbildung 1 zeigt das Zeitverhalten einer Funktion $c=g(a,b)$. Die Taktperiode $1/f_T$ und damit die Zeitspanne zur Berechnung des nächsten Resultatszeichens ist abhängig von der jeweils realisierten Funktion und bewegt sich in der Größenordnung von 2 bis 4 Volladdiererlaufzeiten.

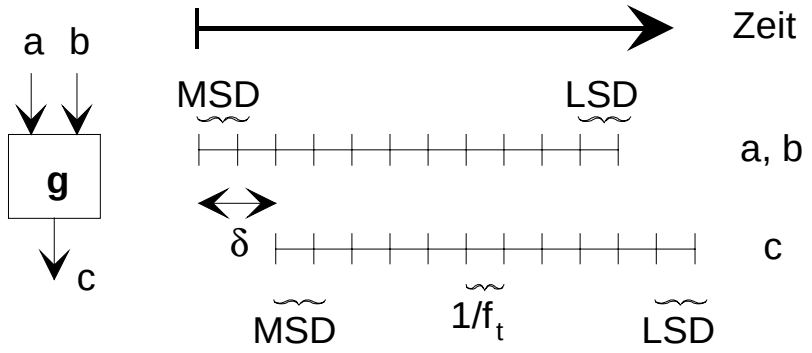


**Abb. 1:** Zeitverhalten einer Digit-On-Line-Funktion g

Da die Resultatzeichen schon während der eingangsseitigen Einspeisung der Operandendigits erzeugt werden, ist eine hohe Überlappung von aufeinanderfolgenden und möglicherweise voneinander abhängigen Operationen erreichbar. Dieses der Digit-On-Line-Arithmetik innewohnende Potential ist in der Abbildung 2 skizziert.

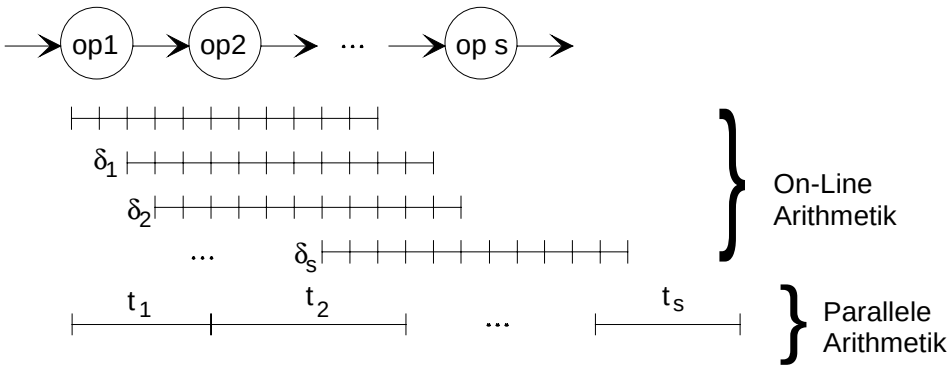


**Abb. 2:** Vergleich sequentieller und paralleler Berechnungsmethoden

So benötigt z.B. eine vollparallele, *n=32* Bit breite, Verarbeitung und Kommunikation bei *s=40* sequentiellen Rechenoperationen mit durchschnittlich $log_2(n)$ Taktzyklen pro Operation eine gesamte Rechenzeit von 200 Taktzyklen. Demgegenüber benötigt eine äquivalente On-Line-Architektur für derart rein sequentiell aufeinanderfolgende Operationen mit durchschnittlich $\delta = 3$ entsprechend

$$n + \sum_{i=1}^{s} \delta_i \tag{3}$$

eine Verarbeitungszeit von 152 Taktzyklen.
Allgemein setzen sich Digit-Online-Algorithmen aus zwei Schritten - einer Rekursion und der Generierung des Outputdigits - zusammen. Die allgemeine Darstellung der Rekursionsgleichung und der Ergebnisfunktion für zwei Operanden kann wie folgt angegeben werden.

$$W_{j+1} \leftarrow f\left(W_j, X_j, x_{j+\delta}, Y_j, y_{j+\delta}, s_j\right) \tag{4}$$

| | |
|---|---|
| W | Rekursionsvektor |
| X, Y | Operandenvektoren |
| x, y | Operandendigits |
| s | Ergebnisdigit |
| ∂ | Online-Delay |
| j | Nummer der Iteration |

$$s_{j-1} \leftarrow S\left(W_j\right) \tag{5}$$

Selektionsfunktion

## 3. On-Line-Arithmetik-Architektur und Algorithmenentwicklung

Hinsichtlich einer VLSI-gerechten Implementierung der Digit-Online-Algorithmen müssen die Gleichungen (4) und (5) in Formen überführt werden, die durch folgende elementare Grundelemente realisierbar sind:

*a) Signed-Digit-Addierer*
Das zentrale Element eines Digit-On-Line-Moduls bilden Signed-Digit (SD) - Addierer [4]. Eine entsprechende 4:2-CMOS-Zelle (2 Eingabedigits, 1 Ausgabedigit) läßt sich mit 42 Transistoren realisieren. Für eine flächeneffiziente und geschwindigkeitsoptimale Implementierung können in verschiedenen On-Line-Elementen auch die vereinfachten redundanten 3:2- und 2:2-CMOS-Zellen eingesetzt werden.

*b) Selektionsfunktion $S(W_j)$*
In allen elementaren On-Line-Verfahren werden die Ergebnisdigits über ein Auswahlverfahren bestimmt, bei der eine Auswertung von Betrag und Vorzeichen des Rekursionsvektors $W_j$ erfolgt. Dabei erweist sich eine approximative Auswertung der *t* höchstwertigen Digits als ausreichend.

*c) Multiplikation eines Digit-Vektors mit einem einzelnen Digit*
Diese Operation der Digit-On-Line-Algorithmen wird auf eine Grundzelle, die die Multiplikation zweier Digits mit Hilfe eines Digit-Negators und eines Multiplexers realisiert, zurückgeführt.

*d) Multiplikation einer Digitstelle mit $2^{-i}$ bzw. $2^i$*
Das vom jeweiligen Iterationsschritt abhängige Verschieben eines Digitvektors wird hierbei durch einen oder mehrere Shifter ausgeführt.

*e) Truncation-Elemente*

Diese Elemente dienen der Korrektur von Pseudoüberläufen bei binären Signed-Digit-Operanden.

Die genannten, für eine Digit-On-Line-Architektur markanten Grundelemente, sind als synthesefähige VHDL-Modelle auf der Register-Transfer-Ebene entwickelt und simuliert worden. Auf deren Basis können nun einfache und elementare Digit-Online-Operationen durch eine intuitive Vorgehensweise ermittelt werden. So lassen sich z.B. durch Einfügen von sequentiellen Elementen in redundante Addiererstrukturen Digit-Online-Operatoren für einen Operanden (Inkrement) bzw. für zwei Operanden (Addition/Subtraktion) ableiten.
Eine weitere Möglichkeit ist eine geschlossene und systematische Methodik, die auf einem in [3] vorgestellten Verfahren beruht. Dabei werden die Iterationsgleichungen für den Rekursionsvektor $W_j$, die Selektionsfunktion $S(W_j)$ sowie die entsprechenden Fehler- und Konvergenzkriterien und Initialisierungsbedingungen aufgestellt. Das wesentliche Ziel dieses Verfahrens ist das Erreichen eines minimalen Online-Delays δ. Die folgenden Gleichungen zeigen beispielhaft die Berechnungsvorschrift für die Digit-On-Line-Addition von zwei Operanden.

Initialisierung: $W_{-1} \leftarrow 0; \quad s_{-2} = s_{-1} \leftarrow 0$

Rekursion: für $j = 0,1,\ldots,m+1$ gilt

$$W_j \leftarrow r\left(W_{j-1} - s_{j-2}\right) + r^{-\delta}\left(x_j + y_j\right)$$

$$s_{j-1} \leftarrow S\left(W_j\right)$$

Selektionsfunktion:

$$S\left(W_j\right) = \begin{cases} sign\left(W_j\right)\left\lfloor \left|W_j + \frac{1}{2}\right| \right\rfloor & wenn \quad \left|W_j\right| \le a \\ sign\left(W_j\right)\left\lfloor \left|W_j\right| \right\rfloor & sonst \end{cases} \qquad (6)$$

## 4. Digit-On-Line-Architekturen

Die in Abschnitt 3 dargelegte konsequente Rückführung der Digit-On-Line-Funktionalität auf elementare Grundzellen gestattete die Generierung von flexiblen parametrierbaren Funktionsblöcken, wie z.B. Additionsstufen mit variabler Operandenbreite. Auf der Basis dieser Funktionsblöcke wurden zahlreiche Digit-On-Line-Module zur Realisierung elementarer Funktionen entwickelt, die auf Grund ihrer inneren Struktur ebenfalls parametrierbar sind. Die als VHDL-Beschreibungen vorliegenden komplexen Anordnungen lassen sich somit in einem entsprechenden VLSI-Entwicklungszyklus an die gewünschten Zielvorgaben anpassen. In den folgenden Ausführungen wird insbesondere auf spezifische Ergebnisse und Besonderheiten eingegangen.

*a) On-Line-Addition mit zwei Operanden (OL_ADD2)*

Die auf den in Abschnitt 3 genannten Iterationsgleichungen basierende On-Line-Addition von zwei Operanden besitzt den in Abbildung 3 gezeigten Datenfluß. Die einzelnen Funktionsblöcke sind dabei parallel aufgebaut und werden sequentiell durchlaufen. Für eine vollständige Operation ergibt sich bei $r = 2$ eine On-Line-Verzögerung von $\delta = 2$.

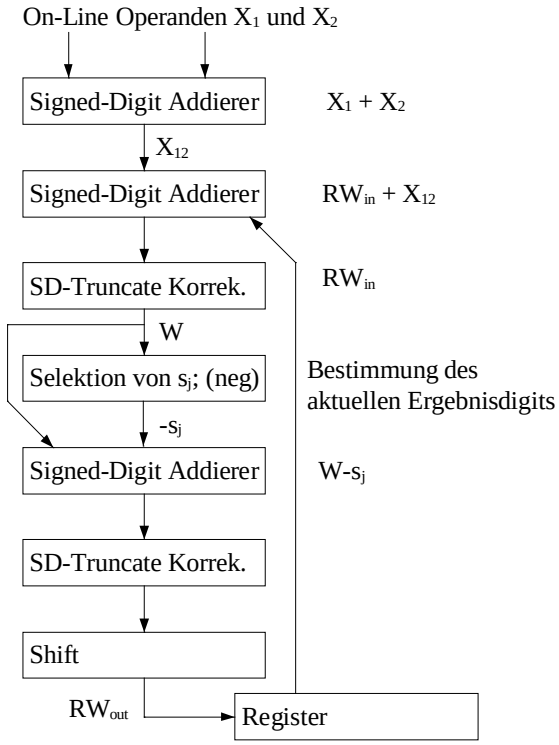


**Abb. 3:** Datenfluß eines Digit-On-Line-Additionsmoduls für zwei Operanden

*b) Multiplikation (OL_MUL)*

Hier lassen sich zwei Varianten realisieren. Während die erste Möglichkeit eine direkte Realisierung der Rekursionsgleichungen darstellt, wird bei der zweiten Lösung das Ergebnisdigit einen Iterationszyklus später berechnet und ausgegeben. Dadurch wird eine Einsparung von Zwischenregistern auf Kosten eines größeren Online-Delays $\delta$ erreicht.

*c) Multiplikation mit einem festen Faktor (OL_SKAL)*

Als wesentlich vereinfachte Variante ist dieses Element zur Multiplikation eines Digitvektors mit einem festen Faktor von der On-Line-Multiplikation abgeleitet worden. Dieses Modul wird vor allem für die Implementierung von zusammengesetzten Funktionen sinnvoll, bei denen eine Bewertung eines Operanden benötigt wird.

*d) Quadrat (OL_EXP2)*

Diese Funktion basiert ebenfalls auf dem Verfahren der On-Line-Multiplikation. Allerdings ließen sich auf Grund der Gleichheit der Operandenvektoren erhebliche Vereinfachungen in der Berechnungsvorschrift und damit in der schaltungstechnischen Umsetzung erreichen. Abbildung 4 zeigt den Datenfluß dieses Moduls. Hieran wird die erreichte parallele Arbeit zwischen den Funktionsblöcken deutlich.

*e) Division (OL_DIV)*

*f) Quadratwurzel (OL_SQRT)*

*g) On-Line-Verzögerung (OL_DELAY))*

Diese Baugruppe realisiert keine arithmetische oder logische Funktion, sondern stellt ein in Bezug auf das gewünschte On-Line-Delay konfigurierbares Verzögerungsglied dar. Damit kann bei der Zusammensetzung mehrerer elementarer Digit-On-Line-Module das zeitrichtige und stellenrichtige Eintreffen von Operandendigits garantiert werden.

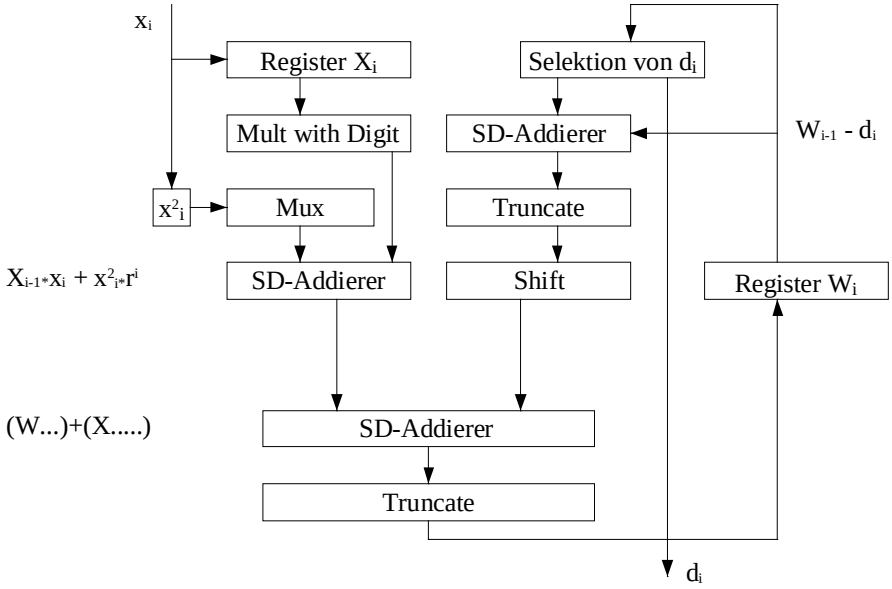


**Abb. 4:** Digit-On-Line-Modul zur Realisierung der Potenz zweiten Grades

Auf der Grundlage einer Aufwands- und Genauigkeitsanalyse konnte festgestellt werden, daß eine Zusammenfassung von elementaren Operationen in einem Digit-Online-Algorithmus zu einer Verbesserung der Kostenfunktion bzgl. des Schaltungsaufwandes und des Online-Delays gegenüber einer Lösung mit Rechenelementen führt, die durch eine Verkettung der Elementarfunktionen gebildet werden. Diese Aussage wird auch durch Ausführungen in [5] gestützt. Unter diesem Gesichtspunkt wurden die folgenden Digit-Online-Module erarbeitet.

*h) On-Line-Addition mit vier bzw. acht Operanden (OL_ADD4 bzw. OL_ ADD8)*

Diese Algorithmen stellen eine Erweiterung des unter a) geschilderten Verfahrens dar und arbeiten mit einem $\delta$ von 3 bzw. 4. Durch die Begrenzung der Operandenzahl auf ein Vielfaches von zwei konnte eine regelmäßige Schaltungsstruktur erreicht werden, die eine flächeneffizientere und geschwindigkeitsoptimalere Realisierung als die sonst übliche Verknüpfung von Additions-Modulen ($\delta$=4 bzw. $\delta$=6) für zwei Operanden darstellt.

*i) On-Line-Addierer mit skalierten Summanden (OL_SA2)*

Bei diesem Modul werden anstelle einer vollständigen Multiplikationsstruktur zwei Vektorzeilen, die die Multiplikation eines Digitvektors und eines Digits ausführen, verwendet. Im Gegensatz zu der gängigen Lösung, der Verkettung eines On-Line-Multiplikations-Moduls mit einem On-Line-Addierer läßt sich zum einen die On-Line-Verzögerung der Anordnung verringern und zum anderen der Schaltungsaufwand minimieren.

*j) lineare Funktion ax+b (OL_LIN)*

Für die Berechnung der linearen Funktion wurde durch die Verschmelzung des On-Line-Additions- und des On-Line-Multiplikations-Verfahrens ein eigenständiger Algorithmus entwickelt, wodurch wiederum eine zeit- und kostengünstigere Lösung entsteht.

Neben den dargelegten elementaren arithmetischen Operationen wurden auch Implementierungen für komplexe Funktionen erarbeitet (Sinus, Arcsinus, Cosinus, Tangens, Logarithmus, Exponentialfunktion). Die Realisierung dieser umfangreicheren Strukturen erfolgte im Gegensatz zu der sonst üblichen Vorgehensweise auf der Basis einer Potenzreihenentwicklung. Der große Vorteil dieser Lösung besteht darin, daß sich auf dieser Grundlage die genannten Funk-

tionen durch eine modulare Verknüpfung der elementaren Elemente umsetzen lassen. Der Ausgangspunkt für diese universell konfigurierbare Struktur ist die baumartige Anordnung nach Abbildung 5, die die Bereitstellung der ersten bis achten Potenz verdeutlicht.

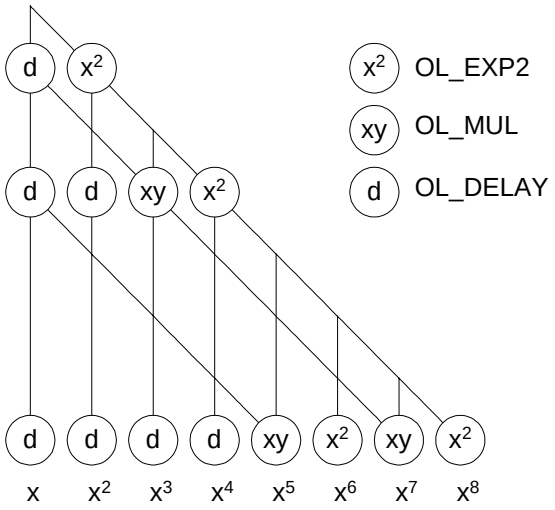


**Abb. 5:** Baumstruktur zur Bereitstellung der Potenzen

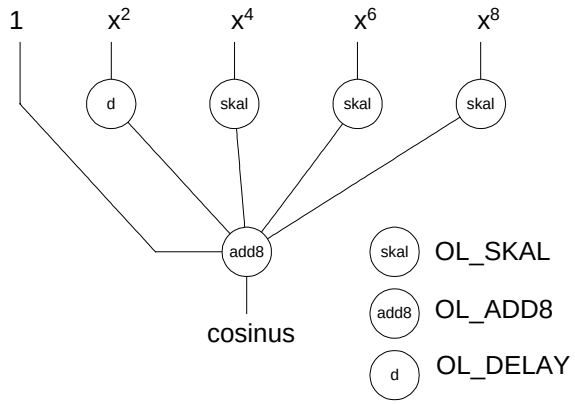


**Abb. 6:** Realisierung der Cosinus-Funktion

Ausgehend von den so bereitgestellten Zwischenergebnissen zur Funktionsberechnung, kann nach einer Wichtung der einzelnen Potenzen mit den sich aus der jeweiligen Reihe ergebenden Faktoren der Funktionswert durch eine abschließende Additionsoperation berechnet werden. Dies ist beispielhaft für die Cosinusfunktion in Abbildung 6 gezeigt. An dieser Stelle sei bemerkt, daß für diese Funktion eine optimierte Potenzreihenanordnung, die nur die notwendigen geraden Potenzen berechnet, implementiert wurde.
Der Nachteil dieses Verfahrens liegt in der Verschlechterung des Verhältnisses zwischen dem benötigten Berechnungsaufwand und dem damit erreichten Genauigkeitsgewinn.

Eine wesentliche Rolle für die wirtschaftliche Anwendbarkeit der Digit-On-Line-Algorithmen spielen Konvertierungsfunktionen, die den Übergang von der seriellen redundanten Zahlendarstellung in eine nicht-redundante Form realisieren. Dafür stehen zwei Hilfsmodule zur Verfügung:

*a) On-the-Fly-Conversion (OL_CONV)*
Mit Hilfe von zwei Registern und einer Steuereinheit wird hier ein serieller redundanter Datenstrom in eine äquivalente parallele nichtredundante Darstellung transformiert. [7]

*b) On-the-Fly-Rounding (OL_ROUND)*
Aufbauend auf der On-the-Fly-Conversion wird hier ein kombinierter Rundungs- und Konvertierungsalgorithmus umgesetzt. [7]

Wie bereits erwähnt, sind die gezeigten On-Line-Grundzellen sowie elementaren und komplexen Digit-On-Line-Module als konfigurierbare VHDL-Beschreibungen entwickelt worden. Für eine weitere Verifikation und Validierung erfolgte eine Synthese auf der Grundlage der ASIC-Technologie ES2 für einen 1.0 µm-CMOS-Prozeß. In diesem Zusammenhang zeigt Abbildung 7 das Layout der Digit-On-Line-Division. In der Tabelle 1 sind die erreichten Ergebnisse hinsichtlich der On-Line-Verzögerung und der notwendigen Fläche bei einer Genauigkeit von 6 Digit und einer Basis $r = 2$ dargestellt.

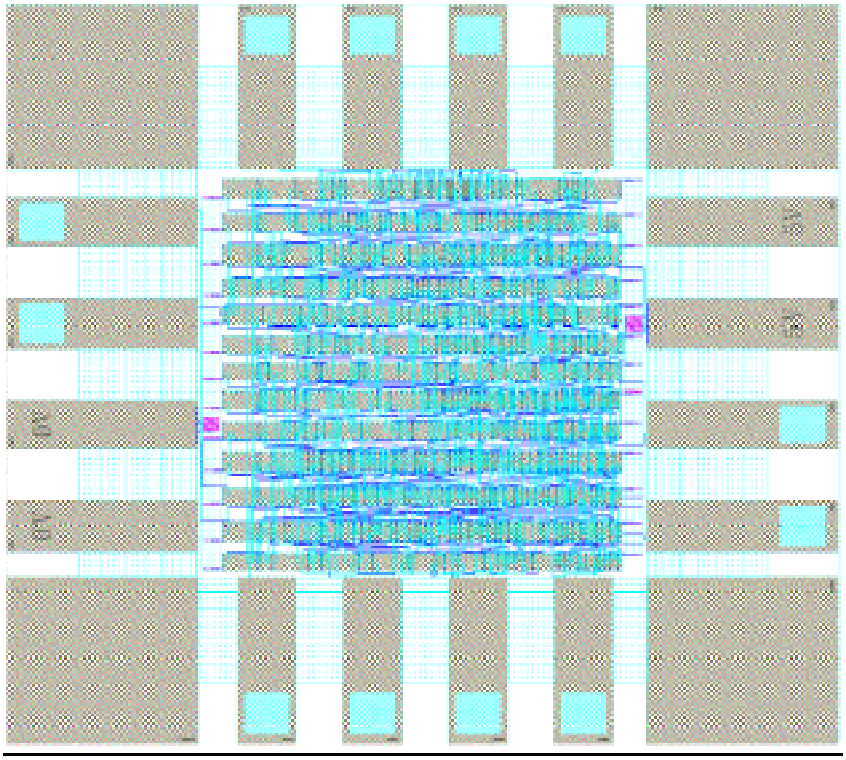

**Abb. 7:** Layout der Digit-On-Line-Division

| Modul | δ | komb. Fläche in $mm^2$ | sequent. Fläche in $mm^2$ | Fläche gesamt in $mm^2$ |
|---|---|---|---|---|
| Addition mit zwei Operanden | 2 | 0,049 | 0,036 | 0,085 |
| Addition mit vier Operanden | 3 | 0,116 | 0,044 | 0,160 |
| Addition mit acht Operanden | 4 | 0,213 | 0,052 | 0,265 |
| Addition mit skal. Summanden | 3 | 0,613 | 0,068 | 0,681 |
| Multiplikation | 3 | 0,277 | 0,026 | 0,404 |
| Potenz zweiten Grades | 3 | 0,156 | 0,088 | 0,243 |
| Mult. mit einem festen Faktor | 3 | 0,158 | 0,060 | 0,219 |
| Division | 4 | 0,348 | 0,123 | 0,472 |
| lineare Funktion | 4 | 0,327 | 0,134 | 0,461 |
| Quadratwurzel | 2 | 0,202 | 0,090 | 0,293 |
| Sinus | 15 | 1,604 | 0,774 | 2,377 |
| Arcsinus | 15 | 1,604 | 0,774 | 2,377 |
| Cosinus | 16 | 1,442 | 0,656 | 2,098 |
| Tangens | 15 | 1,604 | 0,773 | 2,377 |
| Logarithmus | 16 | 2,638 | 1,210 | 3,847 |
| Exponentialfunktion | 16 | 2,796 | 1,270 | 4,066 |
| On-the-Fly-Conversion | n | 0,054 | 0,052 | 0,106 |
| On-the-Fly-Rounding | n | 0,076 | 0,077 | 0,153 |

**Tabelle 1:** Charakteristische Größen der realisierten Digit-On-Line-Module

## 5. Zusammenfassung

Die in dieser Arbeit betrachteten konfigurierbaren und in einem VHDL-Entwicklungsprozeß umgesetzten Digit-On-Line-Module erfüllen minimale Kommunikationsanforderungen, sowohl on-chip als auch off-chip. Sie sind durch eine einfache innere Struktur der zugrundeliegenden primitiven Operatoren und eine hohe Modularität gekennzeichnet. Die Taktrate $f_T$ liegt über 30 MHz und durch die durchgängige Verwendung der redundanten Zahlendarstellung werden langsame, da nicht-redundante arithmetische Operationen vermieden.
Eine weitestmögliche Parallelisierung und Ausnutzung der inneren Abhängigkeiten zwischen verschiedenen Operationen wird möglich. Auf der Basis dieser Digit-On-Line-Module wird derzeit eine komplexe rekonfigurierbare On-Line-Struktur entwickelt, die eine flexible Implementierung verschiedener Berechnungen der inversen Kinematik ermöglicht.

**Literatur**


[1] S.G. Smith, P. Denyer, "Serial-data computation", Kluwer Academic Publishers, 1988

[2] K.K. Parhi, C-Y. Wang, A.P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures", IEEE Journal of Solid-State Circuits, Vol. 27, Nr. 1, S. 29-43, Januar 1992

[3] M.D. Ercegovac, T. Lang, "On-line arithmetic: a design methodology and applications in digital signal processing", VLSI Signal Processing, III, IEEE Press 1988, S. 252-263, 1988

[4] N. Takagi, H. Yasuura, S. Yajima, „High-Speed VLSI Multiplication Algorithmen with a Redundant Binary Addition Tree“, IEEE Transaction on Computers., Vol. 9, S. 789-796, 1985

[5] M.D. Ercegovac, "On-line arithmetic: an overview", Proc. SPIE conference, Vol. 495, Real time signal processing VII, S. 86-93, 1984

[6] A. Scherbyna, „On the minimal delay of on-line computations“, Ecole Normale Superieure de Lyon, Research Report 94-48, Dezember 1994

[7] M.D. Ercegovac, T. Lang, "On-the-fly conversion of redundant into conventional representations," IEEE Transactions on Computers, Vol. 36, Nr. 7, S. 895-897, Juli 1987

[8] A. Wassatsch, „VHDL-Entwurf und Synthese von Digit-On-Line Architekturen für verschiedene Funktionen“, Diplomarbeit, Universität Rostock, 1996