Adaptive Architekturen für Vorwärtsfehlerkorrektur

Sergei Sawitzki
saw@fh-wedel.de
Fachhochschule Wedel
Feldstraße 143, 22880 Wedel, BRD

Im vorliegenden Beitrag werden ausgehend von den vier am weitesten verbreiteten Familien fehlerkorrigierender Codes adaptive Implementierungen verschiedener Decoder für die Vorwärtsfehlerkorrektur *(forward error correction, FEC)* untersucht. Dabei wird deutlich, dass einerseits eine rekonfigurierbare Ausführung von Decodern für eine Code-Familie in der Regel mit lediglich 5 – 10 % Hardware-Mehraufwand verbunden ist und andererseits bei einer rekonfigurierbaren Implementierung von Decodern für verschiedene Code-Familien über 40 % der Chipfläche im Vergleich zur getrennten Umsetzung der gleichen Decoder eingespart werden kann, da Synergien sowohl auf der algorithmischen Ebene als auch bei feingranularer Betrachtung genutzt werden können.

1 Einleitung

In überwiegender Mehrheit der modernen digitalen Kommunikations- und Speichersysteme kommen einzeln oder in Kombination vier Familien fehlerkorrigierender Codes zum Einsatz: Faltungs-, REED-SOLOMON-, *low-density parity-check* (LDPC) und Turbo-Codes. Das Aufkommen und die zunehmende Verbreitung mobiler Endgeräte, die zudem gleichzeitig verschiedene Kommunikationsstandards unterstützen, bedingten ein steigendes Interesse der Industrie und der Forschungsgemeinschaft in adaptiven FEC-Architekturen, die verschiedene Ausprägungen einer Code-Familie oder Decoder für mehrere unterschiedliche Code-Familien unterstützen. Dabei wird die Zielstellung verfolgt, durch sinnvolle Nutzung gleichartiger Ressourcen Chipfläche und/oder Leistungsbedarf im Vergleich zur getrennten Implementierung einzelner Decoder zu reduzieren. Gerade aufgrund der Vielfalt möglicher Anwendungsszenarien (mehrere parallele gleichartig kodierte Datenströme, simultane bzw. zeitversetzte Verarbeitung unterschiedlich codierter Datenströme, Skalierbarkeit in Abhängigkeit von Durchsatzanforderungen usw.) innerhalb eines klar abgesetzten Anwendungsfeldes erscheint eine adaptive Ausführung der Decoder-Hardware attraktiv. Die Vielfalt diverser Architekturkonzepte reicht von ASIP *(application-specific instruction-set processor)* [1, 2] über Network-on-Chip (NoC) [3] bis hin zu mehr oder weniger flexiblen Speziallösungen [4, 5]. Reine Software-Implementierungen kommen dabei aufgrund der enormen Anforderungen an Rechenleistung nicht in Frage, da beispielsweise bei iterativen Dekodierungsverfahren über 1 500 Basisrechenoperationen pro dekodiertem Bit bei Durchsätzen von mehreren Hundert Mbit/s ausgeführt werden müssen.

Tabelle 1: Vergleich verschiedener VITERBI-Decoder-Architekturen

Architektur	Metrik	Referenz	Ergebnis
[6] $K = 3 \ldots 7$, g beliebig	Chipfläche	$K = 7$, g fest	+2.9 %
	Durchsatz	$K = 7$, g fest	$-1.5\,\%$
VITURBO [5]	Chipfläche	$K = 9$, g beliebig	+9 %
$K = 3 \ldots 9$, g beliebig	Maximale Taktfrequenz	$K = 9$, g beliebig	-30 %
[7] $K \in \{7,9\}$, g nach EDGE, CMDA2000 und WCDMA	Chipfläche	$K = 9$, g fest (CDMA2000)	Mehraufwand vernachlässigbar
	Kritischer Pfad	$K = 9$, g fest	$+4\,\%$

Tabelle 2: Vergleich verschiedener REED-SOLOMON Decoder-Architekturen basierend auf Daten aus [11]. n bezeichnet die Blockgröße, t maximale Anzahl der korrigierbaren Symbolfehler, $f(x)$ das Generatorpolynom.

Vergleichs-parameter	Decoder-Architektur				
	[8]	[9]	[10]	[11], Variante 1	[11], Variante 2
Codeparameter	fest	n und t variabel	Universaldecoder: n, t, m und $f(x)$ variabel		
m	8	8	1 ... 8	1 ... 10	1 ... 8
t	8	1 ... 8	1 ... 8	1 ... 8	1 ... 16
Auslöschungs-korrektur	keine	keine	keine	≤ 16	≤ 16
Technologie	$0,25\,\mu m$	$0,35\,\mu m$	$0,25\,\mu m$	$0,13\,\mu m$	$0,13\,\mu m$
Durchsatz (Gb/s)	1,6	0,8	0,048	2,2	2,4
Gatter-äquivalanten	21 000	34 000	44 000	75 000 + 35 Kbit RAM	39 000 + 15 Kbit RAM

2 VITERBI-Decoder

VITERBI-Algorithmus ist das am häufigsten zum Einsatz kommende Dekodierungsverfahren für Faltungscodes. Der größte Teil des Rechenaufwandes bei der Decodierung entfällt dabei auf sogenannte *Add-compare-select*-Operation (ACS), die schaltungstechnisch zwei Addierern, einem Komparator bzw. Subtrahierer und einem Multiplexer entspricht. Die Anzahl der ACS-Operationen pro dekodiertem Symbol wächst exponentiell (2^{K-1}) mit der Rückgriffstiefe K des Codes, ebenfalls müssen bei einer Multistandard-Umsetzung des Decoders die Verbindungen zwischen einzelnen ACS-Einheiten und deren Eingangsbeschaltung (letztere in Abhängigkeit von den Generatorpolynomen des Codes g) rekonfigurierbar gestaltet werden. Wie aus der Tabelle 1 ersichtlich, ist dieser Mehraufwand im Vergleich zu einer nicht rekonfigurierbaren Implementierung jedoch vertretbar (10 % Steigerung bei der Chipfläche im schlimmsten Fall).

3 REED-SOLOMON-Decoder

REED-SOLOMON-Codes sind lineare zyklische Blockcodes, die über einem Erweiterungskörper $GF(q^m)$ definiert sind. Bei Binärdarstellung werden pro Codesymbol m Bit benötigt. Sowohl die Kodierung als auch die Dekodierung setzen schaltungstechnische Umsetzung von Addition und Multiplikation in $GF(q^m)$ voraus. Bei RS-Decodern, die auf Dekodierung eines einzigen Codes ausgelegt sind, können viele GF-Multiplizierer mit einem konstanten Eingang ausgeführt werden, was die Schaltungskomplexität massiv reduziert. Die Tabelle 2 fasst die Eckdaten verschiedener Decoder-Architekturen zusammen. Daraus ist ersichtlich, dass der Hardware-Aufwand mit Flexi-

Tabelle 3: Rekonfigurierbare Multi-Standard FEC-Decoder vs. einfache bzw. getrennt implementierte Decoder. FFU *(flexible functional unit)* ist eine Basis-Funktionseinheit, mit der ein Datenpfad für sowohl Turbo- als auch LDPC-Dekodierung aufgebaut werden kann

Architektur	Metrik	Referenz	Ergebnis
VITERBI-Turbo-Decoder			
VITURBO [5]	Chipfläche	VITERBI-Decoder ($K = 3 \ldots 9$)	+5 % Logik +25 % Speicher
[13] (vollparallell)	Chipfläche	Getrennte Decoder	$-14,5\,\%$
	Durchsatz	Getrennte Decoder	gleich
[13] (zeitmultiplex)	Chipfläche	Getrennte Decoder	$-28,1\,\%$
	Durchsatz	Getrennte Decoder	VITERBI gleich, Turbo 1/2
Turbo-LDPC-Decoder			
[14]	Chipfläche	Getrennte Decoder	$-10\,\%$
[12]	Chipfläche	Turbo-Decoder	$+10 \ldots +20\,\%$
		LDPC-Decoder	$+15 \ldots +20\,\%$
		FFU vs. Einzeldatenpfade	$-39 \ldots -42\,\%$

bilitätsgrad ansteigt. So resultiert eine konfigurierbare Einstellung des Fehlerkorrekturvermögens bei gleicher Symbolbreite in ca. 61 % mehr Siliziumfläche. Ein Decoder mit flexibler Einstellung aller Parameter benötigt mehr als die doppelte Chipfläche bei stark reduziertem Durchsatz.

4 Kombinierte Decoder für unterschiedliche Code-Familien

Bei Turbo- und LDPC-Codes erfolgt die Decodierung iterativ, wobei schrittweise die Wahrscheinlichkeitsmetriken der einzelnen Symbole präzisiert (Turbo) bzw. Teillösungen großer Systeme von Paritätsgleichungen zwischen gleichartigen Rechnerstrukturen ausgetauscht werden. Bei detaillierter Betrachtung der Datenpfade einzelner Decoder-Implementierungen fallen insbesondere Ähnlichkeiten zwischen VITERBI- und Turbo-Decodern auf, so dass bei entsprechender Modifikation eine gemeinsame Nutzung gleicher Hardware-Strukturen möglich ist. Alternativ kann die Basiskomponente eines Turbo-Decoders (SISO-Modul) derart modifiziert werden, dass sie für die Dekodierung eines LDPC-Codes einsetzbar wird [12]. In beiden Fällen kann im Vergleich zur getrennten Implementierung der Decoder Chipfläche eingespart werden, wie die Tabelle 3 zeigt.

5 Zusammenfassung

Ausgehend von den im vorliegenden Beitrag angeführten Daten kann geschlussfolgert werden, dass eine adaptive Umsetzung von Verfahren zur Vorwärtsfehlerkorrektur in Hardware sowohl innerhalb einer Code-Familie als auch als Kombination verschiedener Code-Familien vielversprechend ist. So kostet eine flexible Ausführung eines VITERBI-Decoders für die Rückgriffstiefen von $3-7$ mit einstellbaren Generatorpolynomen weniger als 3 % zusätzliche Chipfläche bei nahezu gleichem Durchsatz verglichen mit einer Standardlösung für $K = 7$. Bei der Kombination eines LDPC- mit einem Turbo-Decoder kann durch gemeinsame Nutzung gleicher Ressourcen sogar über 40 % der Chipfläche beim Datenpfad eingespart werden. Es ist jedoch zu beachten, dass das Nutzen einer adaptiven Lösung stark vom Anwendungsszenario abhängt. So können die adaptiven Decoder in der Regel nicht mehrere Datenströme gleichzeitig dekodieren bzw. haben

einen im Vergleich zur Einzellösung reduzierten Durchsatz. Weiterhin kann beispielsweise bei LDPC abhängig vom eingesetzten Code die auf den Speicher entfallene Chipfläche im Vergleich zur Fläche des Datenpfades dominant sein. Da eine gemeinsame Nutzung des Speichers aufgrund stark unterschiedlicher Zugriffs(band)breiten und Kapazitäten meistens nicht die gleiche Effizienz hat wie eine gemeinsame Nutzung des Datenpfades, fallen die Vorteile einer adaptiven Decoder-Implementierung in diesem Fall wesentlich bescheidener aus [14].

Literatur

[1] T. Vogt, N. Wehn. A Reconfigurable ASIP for Convolutional and Turbo Decoding in an SDR Environment. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(10): 1309 – 1320, October 2008.

[2] P. M. Velayuthan. *Towards Optimized Flexible Multi-ASIP Architectures for LDPC/Turbo Decoding*. Ph.D. thesis, Université de Bretagne-Sud, 17 December 2012.

[3] M. Scarpellino, A. Singh, E. Boutillon, G. Masera. Reconfigurable Architecture for LDPC and Turbo decoding: A NoC Case Study. *10th International Symposium on Spread Spectrum Techniques and Applications*, pp. 671 – 676, IEEE, August 2008.

[4] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, R.-H. Yan. A Unified Turbo/Viterbi Channel Decoder for 3GPP Mobile Wireless in 0.18-µm CMOS. *IEEE Journal of Solid-State Circuits*, 37(11): 1555 – 1564, November 2002.

[5] J. R. Cavallaro, M. Vaya. VITURBO: A Reconfigurable Architecture for Viterbi and Turbo Decoding. *International Conference on Acoustics, Speech, and Singal Processing*, pp. 497 – 500, IEEE, April 2003.

[6] K. Chadha, J. R. Cavallaro. A Reconfigurable Viterbi Decoder Architecture. *Conference Record of the 35th Asilomar Conference on Signals, Systems & Computers*, pp. 66 – 71, IEEE, November 2001.

[7] T. Vogt, N. Wehn, P. Alves. A Multi-Standard Channel-Decoder for Base-Station Applications. *17th Symposium on Integrated Circuits and Systems Design*, pp. 192 – 197, IEEE, September 2004.

[8] A. G. M. Strollo, N. Petra, D. De Caro, E. Napoli. An Area-Efficient High-Speed Reed-Solomon Decoder in 0.25 µm CMOS. *30th European Solid State Circuits Conference*, pp. 479 – 482, IEEE, September 2004.

[9] H.-Y. Hsu, A.-Y. Wu. VLSI Design of a Reconfigurable Multi-mode Reed-Solomon Codec for High-speed Communication Systems. *Asia-Pacific Conference on ASIC*, pp. 359 – 362, IEEE, August 2002.

[10] J.-C. Huang, C.-M. Wu, M.-D. Shieh. C.-H. Wu. An Area-Efficient Versatile Reed-Solomon Decoder for ADSL. *International Symposium on Circuit and Systems*, pp. 517 – 520, IEEE, May 1999.

[11] F.-K. Chang, C.-C. Lin, H.-C. Chang, C.-Y. Lee. Universal Architectures for Reed-Solomon Error-and-Erasure Decoder. *Asian Solid-State Circuits Conference*, pp. 229 – 232, IEEE, November 2005.

[12] Y. Sun, J. R. Cavallaro. A Flexible LDPC/Turbo Decoder Architecture. *Journal of Signal Processing Systems*, 64(1): 1 – 16, Springer, July 2011.

[13] G. Krishnaiah, N. Engin, S. Sawitzki. Scalable Reconfigurable Channel Decoder Architecture for Future Wireless Handsets. *Design, Automation and Test in Europe Conference and Exhibition*, pp. 1563 – 1568, EDAA, April 2007.

[14] J. T. M. H. Dielissen, N. Engin, S. Sawitzki, C. H. van Berkel. Multi-Standard FEC Decoders for Wireless Devices: Scalability Issues and Multi-standard Capabilities. In A. Tasić, W. A. Serdijn, L. E. Larson, G. Setti (editors): *Circuits and Systems for Future Generations of Wireless Communications*, Series on Integrated Circuits and Systems, pp. 271 – 297, Springer, Berlin, 2009.