Laufzeitoptimierung eines Software-Defined-Radio basierten DVB-T2-Messempfängers

Daniel Rother, Jörg Robert, Mariem Slimani, Jan Zöllner, Institut für Nachrichtentechnik, TU Braunschweig, Braunschweig, Deutschland, {rother, robert, slimani, zoellner}@ifn.ing.tu-bs.de

Kurzfassung

Software Defined Radio (SDR) ist ein Ansatz zur Implementierung von Übertragungssystemen, der gegenüber dem klassischen Hardwareentwurf zahlreiche Vorteile, wie eine erhöhte Flexibilität, eine kürzere Entwicklungszeit und geringere Kosten aufweist. Der limitierende Faktor in einer reinen Software-Implementierug, d. h. ohne den zusätzlichen Einsatz von Spezialhardware wie Field Programmable Gate Arrays (FPGAs) oder Grafikkarten für Teilbereiche der Decodierung, ist derzeit die Leistungsfähigkeit des Hauptprozessors, weshalb viele Implementierungen noch nicht echtzeitfähig sind. In dieser Arbeit soll am Beispiel eines am Institut für Nachrichtentechnik (IfN) der TU Braunschweig entwickelten SDR-DVB-T2-Messempfängers gezeigt werden, wie sich eine solche Implementierung optimieren lässt, so dass eine echtzeitfähige Decodierung auch für moderne, hochkomplexe Übertragungsstandards möglich wird.

1. Einleitung

Die zweite Generation des digitalen terrestrischen Fernsehens DVB-T2 [1] bietet eine Vielzahl wählbarer Parameter, woraus sehr unterschiedliche Einsatzmöglichkeiten resultieren. In den Ländern, die DVB-T2 bereits eingeführt haben, wird DVB-T2 primär für die Verbreitung von HD-Inhalten für stationären Empfang verwendet. Im Rahmen des Projektes „Modellversuch DVB-T2 Norddeutschland“ [2] sollte hingegen u.a. der mobile Empfang von DVB-T2 untersucht werden. Dafür war es notwendig, einen mobilen Messempfänger zu entwickeln, da die am Markt verfügbaren Receiver stets für den stationären Empfang optimiert sind.

Für diese Neuentwicklung wurde der Software-Defined-Radio- (SDR-) Ansatz gewählt. Bei diesem Ansatz ist die einzige Hardware-Komponente das Empfängerfrontend, welches das Empfangssignal abtastet und ins Basisband heruntermischt. Die restliche Decodierung erfolgt anschließend komplett in Software auf einem Standard x86-PC. Zusätzliche Spezialhardware wie FPGAs oder Grafikkarten zum Beschleunigen von Teilen der Decodierung werden nicht verwendet.

Dieser Ansatz hat gegenüber der klassischen Hardware-Realisierung den Vorteil, wesentlich schneller in der Entwicklung, flexibler und kostengünstiger zu sein. Ausführlich wurde dieser mobile Messempfänger in [3] und [4] vorgestellt.

Ein Nachteil des SDR-Ansatzes ist der im Vergleich zu einer Hardware-Implementierung deutlich geringere Durchsatz, weshalb der in [3] vorgestellte Empfänger nicht echtzeitfähig war und die Messdaten zunächst auf einer Festplatte zwischengespeichert werden mussten. Durch verbesserte CPU-Architekturen und geschickte Programmierung kann dieser Nachteil jedoch immer stärker minimiert werden.

Im Folgenden soll gezeigt werden, wie sich die vom Messempfänger verwendeten Algorithmen hinsichtlich der Laufzeit verbessern lassen, so dass Nutzdatenraten von rund 6 Mbit/s erreichbar sind. Bei vielen der in [2] vorgestellten DVB-T2-Parameterkombinationen wird diese Datenrate innerhalb einer Physical Layer Pipe (PLP) nicht überschritten, weshalb diese in Echtzeit decodiert werden kann.

Dieser Beitrag ist in vier Abschnitte gegliedert. In Abschnitt 2 werden zunächst kurz der verwendete DVB-T2-Empfänger und anschließend einige besonders zeitkritische Algorithmen vorgestellt. Danach wird in Abschnitt 3 beschrieben, wie sich diese Algorithmen hinsichtlich ihrer Laufzeit optimieren lassen. Die dabei erzielten Laufzeitverbesserungen werden daraufhin in Abschnitt 4 quantifiziert und abschließend werden die Ergebnisse in Abschnitt 5 noch einmal zusammengefasst.

2. Der DVB-T2-Messempfänger

Bei der Decodierung eines DVB-T2-Signals muss eine Vielzahl unterschiedlicher Algorithmen, z.B. zur QAM-Demodulation (Quadratur Amplituden Modulation) oder für die Decodierung des Fehlerschutzes (Forward Error Correction, FEC), nacheinander ausgeführt werden. Die Gesamt-Verarbeitungsdauer der Decodierung wird von diesen Algorithmen sehr unterschiedlich beeinflusst.

Eine erste Analyse mit Hilfe des Intel VTune Amplifier XE Profilers [5] ergab, dass einige wenige Algorithmen zusammen für rund 90% der Verarbeitungsdauer verantwortlich sind (siehe auch Bild 1). Dementsprechend müssen zum Erreichen der Echtzeitfähigkeit vor allem diese Algorithmen optimiert werden.

Diese kritischen Algorithmen sowie das zur Realisierung des SDR-DVB-T2-Empfängers am IfN entwickelte Toolkit werden daher im Folgenden kurz vorgestellt.

2.1. Das SDR-Toolkit

Die unterschiedlichen Algorithmen wurden in dem SDR-Toolkit in einzelne Funktionsblöcke gekapselt (siehe auch Bild 2). Diese Struktur ist beispielsweise von Programmen wie Matlab Simulink und GNU Radio bekannt und erhöht

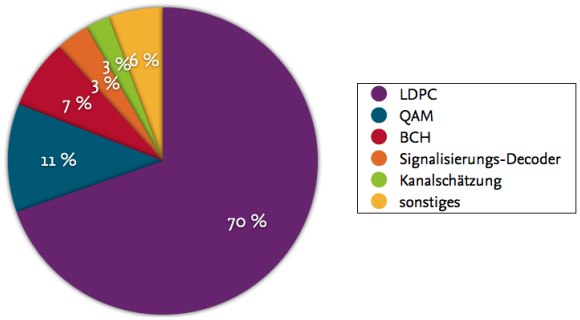

Bild 1 Verarbeitungsdauer der unterschiedlichen Algorithmen für die DVB-T2-Parameter 256 QAM, CR 3/5, 16k FFT

sowohl die Übersichtlichkeit als auch die Wiederverwendbarkeit der einzelnen Blöcke.
Um die Fähigkeiten moderner Mehrkern-Prozessoren optimal ausnutzen zu können, wurde das Toolkit so entworfen, dass jeder Funktionsblock aus mindestens einem Thread besteht. Einzelne, sehr langsame, Algorithmen können dadurch nicht die gesamte Verarbeitungskette blockieren, sondern es können parallel andere Verarbeitungsschritte ausgeführt werden.
Untereinander kommunizieren die einzelnen Blöcke über Verbindungen, die gleichzeitig auch die Synchronisation der einzelnen Threads gewährleisten. Über diese Verbindungen tauschen die Blöcke Pakete in einem definierten Format miteinander aus.
Die Kapselung in diese drei Grundelemente hat den Vorteil, dass nach der einmaligen Entwicklung der Inter-Block-Kommunikation der Entwicklungsfokus voll und ganz auf den verwendeten Algorithmen liegen kann. Außerdem können einzelne Verarbeitungsschritte leicht auf Spezialhardware (z.B. Grafikkarten oder FPGAs) oder auf per Netzwerk angebundene zusätzliche Rechner ausgelagert werden.
Ein weiterer Vorteil ist, dass die Datenströme an jedem beliebigen Messpunkt abgegriffen und typische Messgrößen wie die Bitfehlerrate (Bit Error Rate, BER) oder das Signal-zu-Rausch-Verhältnis (Signal to Noise Ratio, SNR) berechnet werden können. Der SDR-Ansatz erlaubt es darüber hinaus auch, neue Messgrößen wie die Äquivokation [6] einfach in das bestehende Decoder-Design zu integrieren.
Um eine hohe Performance zu erzielen und um die Plattformunabhängigkeit zu gewährleisten, wurde das SDR-Toolkit in Standard C++ geschrieben. Als einzige externe Bibliothek wird das Boost-Framework verwendet [7].
Zusätzlich werden an den Performance-kritischen Stellen Single-Instruction-Multiple-Data- (SIMD-)Befehle verwendet. Die Grundidee von SIMD ist, dass dieselbe Operation gleichzeitig auf unterschiedlichen Daten durchgeführt wird. Dadurch erhöht sich die Parallelität und gleichzeitig verringert sich die Ausführungszeit der Operation um einen entsprechenden Faktor.

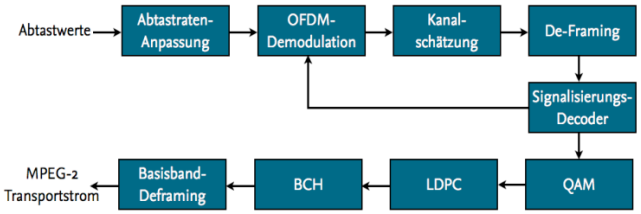

Bild 2 Vereinfachtes Blockschaltbild des SDR-DVB-T2-Empfängers

Ein weiterer Vorteil dieser Befehle ist, dass sie beim Kompilieren häufig direkt in einen einzigen Maschinenbefehl übersetzt werden, womit erneute Laufzeitgewinne verbunden sind. Für x86-Prozessoren sind diese Befehle als die Befehlssatzerweiterungen Streaming SIMD Extensions (SSE) und Advanced Vector Extensions (AVX) verfügbar.
Für eine detailliertere Beschreibung des DVB-T2-Empfängers und des SDR-Toolkits sei auf [3], [4] und [8] verwiesen.

2.2. Vorwärtsfehlerschutz

In der digitalen Signalübertragung wird häufig ein Vorwärtsfehlerschutz eingesetzt, um Übertragungsfehler korrigieren zu können. Dazu existiert eine Vielzahl unterschiedlicher Verfahren, von denen im Folgenden die beiden von DVB-T2 verwendeten Verfahren Low-Density-Parity-Check- (LDPC-) Code und die Bose-Chaudhuri-Hocquenghem- (BCH-) Code vorgestellt werden.

2.2.1. Low-Density-Parity-Check-Codes

Die LDPC-Codes wurden von Robert Gray Gallager bereits 1962 im Rahmen seiner Dissertation erfunden [9]. Ihre Performance liegt nahe der theoretisch möglichen Kanalkapazität („Shannon-Limit“); Sie waren aufgrund ihrer Komplexität jedoch jahrzehntelang nicht rechenbar und wurden daher nicht verwendet. Durch die enormen Fortschritte in der Halbleiter-Industrie konnte dieses Problem jedoch beseitigt werden und LDPC-Codes werden aufgrund ihrer Performance seitdem in vielen modernen Übertragungsstandards wie z. B. in der DVBx2-Familie, bei IEEE 802.11n oder bei IEEE 802.16 eingesetzt.
Häufig wird ein LDPC-Code als sogenannter Tanner-Graph dargestellt, der das zur Decodierung verwendete iterative Belief-Propagation-Verfahren sehr gut veranschaulicht.
Dieser Graph setzt sich aus den Bitknoten (Bit Nodes, BN) und den Checkknoten (Check Nodes, CN) zusammen. Die Spalten der LDPC-Paritätsmatrix entsprechen dabei den Bit-, die Zeilen den Checkknoten und eine gesetzte Eins entspricht einer Verbindung zwischen den entsprechenden Knoten. Beispielsweise entspricht die Paritätsmatrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \quad (1)$$

dem in Bild 3 dargestellten Graphen.

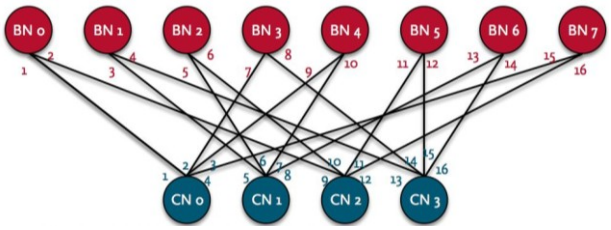

Bild 3 Tanner-Graph eines einfachen LDPC-Codes

Bei einem gültigen Codewort sind alle Paritätsprüfungen erfüllt, d. h. das XOR über alle Eingangskanten eines Checknodes ist Null.
Zur Decodierung eines Codewortes werden die Knoten jeweils aktualisiert und ihr neuer Wert wird über die Kanten an die benachbarten Knoten übertragen. Der aktualisierte Wert einer Bitknotenkante wird dabei mit der Formel

$$v_{n\to m} = u_n + \sum_{m'\in M(n)\setminus m} w_{m'\to n} \qquad (2)$$

berechnet. Die Variable $v_{n->m}$ ist dabei der neue Wert für die Kante von Bitknoten n zu Checkknoten m, u_n ist der alte Wert des Bitknotens n, $w_{m'->n}$ ist der alte Wert der Kante von Checkknoten m' zu Bitknoten n sowie $M(n)$ ist die Menge aller Eingangskanten an Bitknoten n. Um Fehlerfortpflanzungen zu vermeiden, wird beim Aktualisieren einer Kante ihr vorheriger Wert nicht verwendet.
Für das Aktualisieren der Checkknoten-Kanten kann die Formel

$$T_f = \prod_{n'\in N(m)\setminus n} -\tanh\left(\frac{v_{n'\to m}}{2}\right)$$
$$w_{m\to n} = \log\left(\frac{1-T_f}{1+T_f}\right) \qquad (3)$$

verwendet werden. Dabei ist $w_{m->n}$ der neue Wert für die Kante von Checkknoten m zu Bitknoten n, $v_{n'->m}$ ist der alte Wert der Kante von Bitknoten n' zu Checkknoten m und $N(m)$ ist die Menge aller Eingangskanten an Checkknoten m. Eine Näherungslösung dieser Formel ist

$$w_{m\to n} = \prod_{n'\in N(m)\setminus n} \operatorname{sign}(v_{n'\to m}) \min|v_{n'\to m}| . \qquad (4)$$

Diese MinSum genannte Variante ist wesentlich schneller zu berechnen, sie besitzt jedoch auch eine um ca. 0,5 dB geringere Performance.
Die Knoten-Aktualisierungen werden solange wiederholt, bis entweder sämtliche Paritätsprüfungen erfolgreich sind, oder bis die maximale Anzahl an Iterationen erreicht ist.

2.2.2. Bose-Chaudhuri-Hocquenghem-Codes

LDPC-Codes neigen dazu, auch bei einem hohen SNR ein paar Restfehler nicht korrigieren zu können, woraus ein sogenannter Error Floor resultiert. Um auch diese letzten Fehler korrigieren zu können, werden zusätzlich BCH-Codes eingesetzt [10].

2.3. Quadratur Amplituden Modulation

Die QAM ist eine in der digitalen Übertragungstechnik häufig eingesetzte Modulationsart. Bei dieser werden die einzelnen Sendesymbole auf unterschiedliche Konstellationspunkte abgebildet.
Auf Empfängerseite muss diese Abbildung rückgängig gemacht werden. Aufgrund von Störungen weichen die Empfangssymbole jedoch von ihrer idealen Position ab. Bei der sogenannten Hard-Decision werden die Empfangssymbole daher immer auf den nächstgelegenen Konstellationspunkt abgebildet.
Für die Gesamtperformance des Übertragungssystems ist die Soft-Decision jedoch deutlich besser geeignet. Bei dieser wird die Wahrscheinlichkeit (Log-Likelihood-Ratio, LLR) für eine gesendete Eins bzw. Null mit Hilfe der Formel

$$\mathrm{LLR}(b_i) = \ln\left(\frac{P(b_i=1)}{P(b_i=0)}\right) \qquad (5)$$

wobei die Variable b_i dabei das Eingangsbit und P die *a posteriori* Wahrscheinlichkeit dieses Bits ist bzw. im Fall der QAM mit

$$\mathrm{LLR}(b_i) = \ln\frac{\sum_{x\in C_i^1}\exp\left(-\frac{(I-\rho_I I_x)^2+(Q-\rho_Q Q_x)^2}{2\sigma^2}\right)}{\sum_{x\in C_i^0}\exp\left(-\frac{(I-\rho_I I_x)^2+(Q-\rho_Q Q_x)^2}{2\sigma^2}\right)} \qquad (6)$$

bestimmt. Die Variable x ist das gesendete Bit, $C_i^{0/1}$ ist die Menge aller Konstellationspunkte mit einer 0 bzw. 1 an Stelle i, I_x/Q_x sind die gesendeten In-Phase- bzw. Quadratur-Koordinaten, I/Q sind die tatsächlich empfangenen Koordinaten, $\rho_{I/Q}$ sind die Ampiltuden-Fading-Koeffizienten und σ^2 ist die Varianz des Kanalrauschens [10].
Diese LLR-Werte werden anschließend an den LDPC-Decoder übergeben, der diese bei der Fehlerkorrektur als Startwerte der Bitknoten berücksichtigt und dadurch deutlich leistungsfähiger wird.

2.4. Signalisierungs-Decoder

Die DVB-T2-Signalisierungsinformationen enthalten u.a. Informationen über das eingesetzte Framing, d. h. beispielsweise auf welchen Orthogonal-Frequency-Division-Multiplex- (OFDM)-Subträgern eine PLP übertragen wird. Diese Informationen müssen demnach der OFDM-Demodulation verfügbar gemacht werden, um die eigentlichen Nutzdaten zu decodieren.
Um eine fehlerfreie Übertragung der Signalisierungsinformationen zu gewährleisten, sind diese ebenfalls LDPC-codiert. Dabei wird stets die Coderate 1/2 bei einer Blocklänge von 16200 Bits verwendet.

3. Laufzeitoptimierung der Algorithmen

In diesem Abschnitt wird zunächst analysiert, weshalb die in Abschnitt 2 beschriebenen Blöcke für rund 90% der Verarbeitungsdauer des DVB-T2-Empfängers verantwort-

lich sind. Der Grund dafür ist unterschiedlich und es werden daher anschließend für jeden Block die jeweils eingesetzten Optimierungen vorgestellt. Neben theoretischen Überlegungen stützt sich diese Analyse auch auf ein Profiling des Quellcodes mit Hilfe des Intel VTune Amplifier XE [5].

3.1. Optimierungen des LDPC-Decoders

Der LDPC-Decoder verwendet die von dem QAM-Demapper berechneten LLR-Werte, die als *Float*-Datentyp gespeichert werden.

In Abschnitt 2.2.1 wurde gezeigt, dass aus Fehlerfortpflanzungsgründen der alte Wert einer Kante nicht in die Berechnung ihres neuen Wertes eingeht. Dies hat jedoch zur Folge, dass die LLR-Werte aller Kanten und nicht nur der Knoten gespeichert werden müssen. Bei DVB-T2 sind dies rund 240.000 Kanten, die pro Iterationsschritt jeweils zweimal gelesen und geschrieben werden müssen (einmal aus BN- und einmal aus CN-Sicht). Dies sind rund 2 MByte an Daten. Zusätzlich ist ein LDPC-Code informationstheoretisch umso besser, je zufälliger die Anordnung seiner Kanten ist.

Für eine Softwareimplementierung resultieren daraus sehr viele wahlfreie Speicherzugriffe. Prinzipiell wird jedoch immer angestrebt, die zu verarbeitenden Daten möglichst linear im Speicher vorzuhalten. Das Einlesen und Verarbeiten der Daten erfolgt dann sehr schnell, wohingegen jeglicher wahlfreie Zugriff um Größenordnungen langsamer ist. Durch komplexe Caching-Mechanismen des Hauptprozessors kann dieser Unterschied abgemildert werden. Die Größe dieser Caches und somit auch ihre Wirkung ist jedoch begrenzt (z. B. 256 kByte Level 2 Cache bei einem Intel Core i7-3770K Prozessor). Aus diesem Grund scheidet auch die Implementierung auf einer Grafikkarte aus .

Die Knoten-Aktualisierungen mit den Formeln (2) und (4) können schnell berechnet werden und tragen nur wenig zur Verarbeitungsdauer bei. Die hohe Verarbeitungsdauer des LDPC-Decoders resultiert demnach primär aus den Speicherzugriffen.

Eine Beschleunigung kann zum einen durch eine Reduzierung der Bitauflösung der LLR-Werte und zum anderen durch eine Linearisierung des Speichers erreicht werden. Hardwareimplementierungen haben gezeigt, dass bereits mit drei bis vier Bit quantisierte LLR-Werte für eine gute Performance ausreichend sind [11]. Zusätzlich können statt des *Float*-Datentyps auch *Integer*-Datentypen verwendet werden.

Die Hardwareimplementierungen erreichen ihre hohe Geschwindigkeit außerdem durch die massiv parallele Verarbeitung der einzelnen Knoten. Dies ist sehr einfach möglich, da eine Knoten-Aktualisierung immer nur von seinen Eingangskanten und nicht von anderen Knoten abhängig ist. Eine Parallelisierung wird daher auch für die Optimierung unter Verwendung der in Abschnitt 2.1 vorgestellten SSE-Befehle eingesetzt.

Für die Speicherlinearisierung sollte im Speicher immer die jeweils n-te Kanten der parallel bearbeiteten Knoten linear im Speicher angeordnet werden, damit diese gemeinsam geladen werden können. Wenn beispielsweise vier Bitknoten parallel bearbeitet werden sollen, müssen die BN-Kanten für den in Bild 2 dargestellten LDPC-Code in der Reihenfolge [1 3 5 7 | 2 4 6 8 | 9 11 13 15 | 10 12 14 16] gespeichert sein.

Der kleinste mit SSE verarbeitbare Datentyp ist acht Bit Integer (*int8*). Er wird daher für die Quantisierung der LLR-Werte verwendet. Aus der SSE-Registerbreite von 128 Bit ergibt sich damit ein Parallelfaktor von 16.

3.2. Optimierungen des QAM-Demappers

Im Gegensatz zum LDPC-Decoder sind in diesem Block nicht die Speicherzugriffe ausschlaggebend für die Verarbeitungsdauer, sondern die aufwendige und damit langsame Berechnung der Formel (6), insbesondere der darin enthaltenen Exponential- und Logarithmus-Funktion.

Für die Exponentialfunktion kann statt der genauen Berechnung jedoch auch eine vorberechnete Look-Up-Table (LUT) verwendet werden.

Zusätzlich geht in die Berechnung der LLR-Werte mit Formel 6 immer nur der Wert des jeweiligen Empfangssymbols ein. Andere Symbole haben darauf keinen Einfluss und die Berechnung lässt sich damit sehr gut parallelisieren.

Dafür, und für zusätzliche Laufzeitgewinne, werden wiederum die SSE-Befehle eingesetzt. Diese enthalten jedoch keine Funktionen zum Berechnen des Logarithmus und es wird daher zusätzlich die Bibliothek AMDLibM [12] eingebunden, in der diese und weitere Funktionen implementiert sind[1].

3.3. Optimierungen des BCH-Decoders

Ähnlich wie beim LDPC-Decoder sind auch hier im Wesentlichen die Speicherzugriffe für die lange Verarbeitungsdauer dieses Blockes verantwortlich. Beispielsweise müssen in einem Array die Werte um jeweils eine Position verschoben werden. Wenn dafür statt des Befehls `array[i] = array[i+1]` ein `memcpy(array, array + 1, M)` verwendet wird, reduziert sich die Laufzeit bereits deutlich. Als zusätzliche Optimierung werden auch in diesem Block wiederum SSE-Befehle verwendet.

3.4. Optimierung des Signalisierungs-Decoders

Wie in Abschnitt 2.4 beschrieben wurde, sind die Signalisierungsinformationen LDPC-codiert und werden zusätzlich in die OFDM-Demodulation zurückgekoppelt (siehe auch Bild 2). Dauert diese LDPC-Decodierung zu lange, wartet die OFDM-Demodulation auf Daten und in-

[1] Intels Math Kernel Library (MKL) ist überraschender Weise auch auf Intel-Prozessoren in diesem Fall langsamer. Zudem fallen bei der MKL Lizenzkosten an und sie wird daher nicht verwendet.

folgedessen läuft die gesamte weitere Signalverarbeitungskette leer und wartet.
Im Falle von statischen Signalisierungsinformationen müssen diese nur einmalig decodiert werden und dieses Problem tritt dadurch nicht auf. Ansonsten können die in Abschnitt 3.1 vorgestellten LDPC-Optimierungen verwendet werden. Da der verwendete LDPC-Code an dieser Stelle bekannt ist, kann für den Signalisierungs-Decoder eine speziell angepasste, nicht generische LDPC-Implementierung verwendet werden, woraus zusätzliche Laufzeitgewinne resultieren.

4. Ergebnisse der Optimierungen

In diesem Abschnitt werden die in Abschnitt 3 vorgestellten Optimierungen anhand verschiedener Szenarien quantifiziert. Dazu werden zum einen reale Messdaten aus dem Projekt „Modellversuch DVB-T2 Norddeutschland“ sowie aus einem Projekt in Singapur und zum anderen synthetisch erzeugte Daten verwendet. Die Ergebnisse sind dadurch einerseits leicht reproduzier- und vergleichbar, andererseits aber auch realistisch.
Der für die Messungen verwendete PC enthielt als Prozessor einen Intel Core i7-3770K mit 3,5 GHz sowie 8 logischen Kernen, 16 GByte RAM. Als Betriebssystem wurde Ubuntu 12.04 eingesetzt. In den Messungen wurde jeweils ein 8 MHz breiter Kanal mit 10 MSamples/s komplex abgetastet und mit jeweils 16 Bit quantisiert. Die Eingangsdatenrate beträgt bei damit 320 Mbit/s.
In dem Datenstrom sind mehrere PLPs enthalten, die sich hinsichtlich der eingesetzten Coderate (CR), Modulation und auch in der Datenrate unterscheiden. Die Datenrate pro PLP ist zum einen von der eingesetzten Mod.-CR-Kombination, zum anderen von der gesamten PLP-Anzahl und schließlich auch vom Scheduling, d. h. wie oft eine PLP übertragen wird, abhängig. Der Empfänger muss immer nur die gerade betrachtete PLP decodieren. Die in den unterschiedlichen Szenarien verwendeten Werte sind in Tabelle 1 zusammengefasst. Zusätzlich ist dort die resultierende Nutzdatenrate, d. h. die Datenrate des MPEG2-Transportstroms, angegeben. Die Szenarien 3 bis 5 wurden dabei so gewählt, dass sie der in [2] vorgestellten DVB-T2-Parameterkonfiguration für Deutschland möglichst nah kommen.
Für die synthetischen Daten werden dieselben Parameter wie in den realen Messungen verwendet, allerdings können hier die Kanaleigenschaften und die Nutzdatenrate problemlos variiert werden.

Szenario Nr.	Messort	Modulation	CR	FFT-Größe	Nutzdatenrate
1	Singapur	256-QAM	3/5	16k ext.	12,5 Mbit/s
2	Singapur	64-QAM	2/3	16k ext.	6 Mbit/s
3	Norddeutschland	16-QAM	2/3	16k ext.	2,3 Mbit/s
4	Norddeutschland	64-QAM	2/3	16k ext.	2,3 Mbit/s
5	Norddeutschland	64-QAM	2/3	32k ext.	2,3 Mbit/s

Tabelle 1 Eingesetzte Parameterkonfigurationen

Als Metrik für das Quantifizieren der Optimierungen dient die eingesparte Verarbeitungsdauer. Diese wird zum einen mit Hilfe des VTune-Profilers, zum anderen mit einer programminternen Zeitmessung bestimmt. Erstere hat den Vorteil, wesentlich genauer zu sein, allerdings entsteht durch das Profiling zusätzlicher Overhead.
VTune bestimmt neben der zwischen dem Start und dem Beenden des Programmes real vergangenen Zeit zusätzlich auch diejenige Zeit, die das betrachtete Programm tatsächlich auf dem Hauptprozessor gerechnet wurde. Auf Prozessoren mit n Kernen kann diese CPU-Zeit daher im Extremfall n-mal so groß wie die real vergangene Zeit sein. Für alle Laufzeitmessungen wird ein ca. 26 Sekunden langer IQ-Datenstrom verwendet.
Bild 4 zeigt exemplarisch die durch die verschiedenen Optimierungsschritte erzielten Laufzeitgewinne anhand von Szenario 1 und 2. Die CPU-Zeit ist in beiden Fällen deutlich von 830s auf 260s bzw. von 220s auf 95s zurückgegangen. Die LDPC-Verarbeitungsdauer konnte dabei um rund 70% reduziert werden, sie macht aber vor allem in Szenario 1 noch immer den größten Anteil aus.
Aus der Abbildung ist zusätzlich erkennbar, dass die übrigen Blöcke nach ihrer Optimierung nur noch zu einem geringen Teil zu der Gesamt-Verarbeitungsdauer beitragen.
In allen Szenarien und in allen Blöcken konnte die Verarbeitungsdauer deutlich reduziert werden. Der BCH-Decoder benötigt beispielsweise nur noch 5% seiner ursprünglichen Verarbeitungsdauer und kann somit beim Betrachten der Gesamt-Verarbeitungsdauer nahezu vernachlässigt werden. Diese wurde je nach Szenario um 70% bis 80% reduziert und die Optimierungen können somit als erfolgreich angesehen werden. Mit dem optimierten DVB-T2-Empfänger war es zudem möglich, die aufgezeichneten

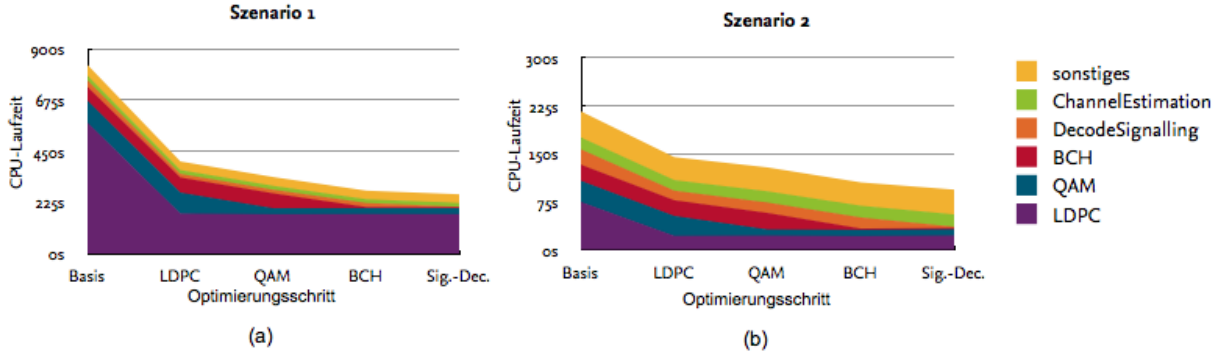

Bild 4 Laufzeitgewinne in den Szenarien 1 und 2

Daten der Szenarien 2 bis 5 in Echtzeit zu decodieren.
Mit den synthetisch erzeugten Daten war die Echtzeitdecodierung zunächst nicht möglich. Im Gegensatz zu den Messdaten wurde hierbei zunächst nur eine PLP verwendet. Daraus resultierte selbst im einfachsten Szenario 3 eine Nutzdatenrate von 14,8 Mbit/s[2].
In einem solchen Szenario ist erneut der LDPC-Decoder aufgrund seiner Speicherzugriffe das limitierende Element. Diese Limitierung resultiert zum einen aus der höheren Datenrate, zum anderen kann sie jedoch auch aus sich verschlechternden Kanaleigenschaften resultieren. In letzterem Fall müssen mehr Fehler korrigiert werden, der LDPC-Decoder benötigt dafür zusätzliche Iterationen mit somit auch mehr Speicherzugriffen.
Es wurde daher anschließend untersucht, welche die maximal mögliche Datenrate ist, bis zu der die LDPC-Decodierung echtzeitfähig ist. Die Anzahl der LDPC-Iterationen wurde dabei auf zehn festgelegt, damit die so bestimmte Datenrate auch in einem realen fehlerbehafteten Kanal, und nicht nur in einem quasi-idealen Kanal, erreichbar ist.
Das Ergebnis dieser Untersuchung ist, dass Datenraten von 6 Mbit/s erreichbar sind. Diese Datenrate gilt dabei für alle Szenarien, d. h. für die verschiedenen Modulationen und Coderaten. Der limitierende Faktor sind letztlich immer die Speicherzugriffe in dem LDPC-Decoder.
Die Parameterkonfiguration aus Szenario 4 ist nach [2] für den Empfang von 6 bis 7 Programmen gedacht. Pro Programm stünde dabei eine Datenrate von 3 bis 4 Mbit/s zur Verfügung[2]. Würde jedes Programm in einer eigenen PLP übertragen, wäre der optimierte DVB-T2-Empfänger für eine echtzeitfähige Decodierung somit ausreichend schnell.

5. Zusammenfassung und Ausblick

Software Definded Radio bietet u.a. aufgrund seiner Flexibilität insbesondere für Forschung und Entwicklung viele Vorteile. Limitierender Faktor hierbei ist vor allem die Leistungsfähigkeit der Prozessoren. Eine echtzeitfähige Verarbeitung war daher bislang häufig nicht möglich. In diesem Beitrag konnte jedoch gezeigt werden, dass durch Auswahl geeigneter Decodier-Algorithmen, die Verwendung moderner Programmiertechniken, geschicktem Speicherlayout und insbesondere durch den Einsatz von SIMD-Befehlen die Verarbeitungsdauer solcher Implementierungen deutlich reduziert werden kann. Im vorgestellten DVB-T2-Empfänger beträgt diese nur noch 20% bis 30% ihres ursprünglichen Wertes, wodurch Nutzdatenraten auf MPEG-Transportstrom-Ebene von 6 Mbit/s erreichbar sind.
Die maximal erreichbare Datenrate ist im wesentlichen von der Verarbeitungsdauer des LDPC-Decoders abhängig, der auch nach der Optimierung noch den größten Anteil an der Gesamt-Verarbeitungsdauer benötigt.

[2] Die im Vergleich zu Tabelle 1 unterschiedliche Datenrate resultiert aus der Unterschiedlichen PLP-Anzahl.

Durch die weiter voranschreitende Prozessorentwicklung wird die Verarbeitungszeit weiter zurück gehen. So bietet beispielsweise die für das Frühjahr 2013 erwartete neue Intel-Hauptprozessorarchitektur Haswell mit der Befehlssatzerweiterung AVX2 endlich auch Integerbefehle für die gegenüber SSE doppelt so großen AVX-Register und ermöglicht zudem das Laden aus wahlfreien Speicherbereichen mit einem einzigen Befehl. Beides dürfte die hier vorgestellte Implementierung noch einmal deutlich beschleunigen.

Danksagung

Die Autoren danken ihren Kollegen des Instituts für Nachrichtentechnik der Technischen Universität Braunschweig, insbesondere Prof. Dr.-Ing. Ulrich Reimers, für die hilfreichen Kommentare und Anmerkungen.

6. Literatur

[1] Digital Video Broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2), ETSI EN 302 755, V1.1.1, October 2008

[2] Terrestrik der Zukunft: Zukunft der Terrestrik, Projektbericht DVB-T2 Norddeutschland, Shaker-Verlag, Aachen, 2012

[3] Slimani, M; Robert, J.; Zoellner, J.: A Software-Based Mobile DVB-T2 Measurement Receiver, IEEE International Symposium on Broadband and Multimedia Systems (BMSB), 2012

[4] Slimani, M.; Robert, J.; Zoellner, J.: Softwarebasierter Messempfänger für DVB-T2, Fernseh- und Kinotechnik (FKT), Heft 3/2012, S. 84-87, März 2012

[5] Intel: Intel VTune Amplifier XE (online), 2013, Verfügbar: http://software.intel.com/en-us/intel-vtune-amplifier-xe (07.02.2013)

[6] Robert, J: Äquivokation: Eine „neue“ Messgröße für die digitale Datenübertragung. In: FKTG-Tagung 2012, Wiesbaden

[7] Boost-Library (online), 2013, Verfügbar: http://www.boost.org/ (07.02.2013)

[8] Zoellner, J; Robert, J; Rother, D; Slimani, M: Ein Toolkit für den Entwurf softwarebasierter Empfangssysteme. In: 15. ITG-Fachtagung für elektronische Medien, 2013

[9] Gallager, R. G., Low Density Parity Check Codes, Monograph, M.I.T. Press, 1963

[10] Digital Video Broadcasting (DVB); Implementation Guidelines for a second generation digital terrestrial television broadcasting system (DVB-T2), ETSI TS 102 831, V1.2.1, (08/2012)

[11] Falcao, G.; Sousa, L.; Silva, V.: “Massively LDPC decoding on multicore architectures”, IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 2, Feb. 2011

[12] AMD: AMDLibM (online), 2013, Verfügbar: http://developer.amd.com/tools/cpu-development/libm/