JPEG-Codec

(Encoder)

Dokumentation (V1.5)

Inhaltsverzeichnis

1 Eigenschaften

- JPEG-Kompression/Dekompression entsprechend „baseline process“ nach CCITT T.81 (ISO/IEC 10918-1).
- Der JPEG-Codec besteht aus einem separaten Encoder- und Decoderteil, die einzeln oder parallel betrieben werden können. Hohe Leistungsfähigkeit und Robustheit im praktischen Einsatz.
- Geringste Verzögerungszeiten zwischen Dateneingang und komprimierten Daten am Ausgang (2…8 µs).
- Dieser high-performance Codec ist für Einzelbilder hoher Qualität und/oder Motion-JPEG (MJPEG) geeignet.
- Geringer Platzverbrauch bei hoher Geschwindigkeit in XILINX FPGAs (Spartan-6 < 1200 Slices).
- Qualität und Komprimierung mit 4 oder mehr vorgegebenen oder benutzerdefinierten Quantisierungstabellen wählbar. Datenmenge des komprimierten Bildes zwischen 1 % und 33 % der Datenmenge des unkomprimierten Bildes.
- Voller Reset vor und nach jedem Bild, d.h. bei Motion-JPEG hat jedes Bild gleiche Anfangsbedingungen.
- Leichtes Einfügen in ein vorhandenes HDL-Programm bzw. Verbinden mit Programmmodulen durch definierte Schnittstellen.
- Programmierung des gesamten JPEG-Codecs wurde mit grafischer Oberfläche (Matlab/Simulink mit XILINX System Generator) als modularer Aufbau ausgeführt.
- Bildgröße kann beliebig sein (z.B. 64 k x 64 k).
- Optionale Module für Videokamerasteuerung, Bayer-Pattern-Interpolation, Farbraumkonvertierung, RAM-Anbindung usw. sind verfügbar. Individuelle Anpassung der IP-Cores möglich.
- Optional mit automatischer Komprimierung für limitierten Datendurchsatz. (Datenmenge des komprimierten Bildes wird der Bandbreite der Schnittstelle effektiv angepaßt).
- Development Kit für Codec-Test verfügbar.
- Core für XILINX FPGAs einsetzbar (Spartan-3-Familie, Spartan-6, Virtex-4, Virtex-5, Virtex-6, 7-er Familie Artix, Kintex, Virtex, ZYNQ).

2 Anwendungen

- Überwachungssysteme, intelligente Überwachungskameras
- Videokonferenzsysteme, digitale Foto- oder Videokameras
- Echtzeitfähige Videobearbeitungssysteme
- Intelligente Videoauswertung
- Latenzarme Automotive- oder Echtzeitsysteme

3 Beschreibung des Encoders

3.1 Platzverbrauch und Geschwindigkeit

Der **JPEG-Encoder** (Kernkomponenten 2D-DCT und VLC) ist in den folgenden XILINX FPGAs mit dem Platzverbrauch nach Place & Route beispielhaft aufgelistet:

FPGA	Slice Reg	Slice LUTs	Belegte Slices	DSP48 oder Emb. Mult	BlockRAM	BlockRAM /kbit
Spartan-3E	2234	3442	2769	1	5	80
Spartan-6	2240	2790	1192	1	3+2	56
Virtex-5	2199	2818	1173	1	1+3	80
Virtex-6	2205	2653	1098	1	1+3	80
Virtex-7	2201	2640	1142	1	1+3	80
Kintex-7	2184	2591	1080	1	1+3	80

Die möglichen maximalen Verarbeitungsgeschwindigkeiten sind im Folgenden für FPGAs mit entsprechendem Speedgrade für den maximalen Systemtakt und maximalen Pixeltakt bei 24 bit Datenbreite je Pixel für Encoder für höchste Qualitäten (Komprimierung auf 1…30 % des Originalbildes) aufgelistet:

FPGA	Speedgrade	t_{max} /ns	$f_{System\ max}$ /MHz	$f_{Pixel\ max}$ /MHz (24 bit/Pixel)
Spartan-3E	4	8,3	120	60 bzw. 15 (HQ)
Spartan-6	2	5,68	176	88 bzw. 22 (HQ)
Spartan-6	3	4,81	208	104 bzw. 26 (HQ)
Virtex-5	2	4,17	240	120 bzw. 30 (HQ)
Virtex-6	2	3,12	320	160 bzw. 40 (HQ)
Virtex-7	2	3,29	304	152 bzw. 38 (HQ)
Kintex-7	2	3,67	272	134 bzw. 34 (HQ)

Damit lassen sich bei Virtex-6 Speedgrade 2 Komprimierungen von 440 Bilder/s für VGA-Auflösung (640x480), 173 Bilder/s für XGA-Auflösung (1024x768), 64 Bilder/s HDTV (1920x1080) und 4,3 Bilder/s (0,23 s/Bild) für Bilder mit 6464x4832 Pixel erreichen. Für höhere Komprimierungen bzw. geringere Qualitäten kann bei dem Encoder für höhere Komprimierung (auf 1… 15 % des Originalbildes) der Pixeltakt gleich dem halben Systemtakt sein. Für high-quality (Komprimierung bis über 30 % des Originalbildes) werden geringere Pixeltakte genutzt.Damit lassen sich bei Virtex-6 Speedgrade 2 Komprimierungen von 130 Bilder/s für VGA-Auflösung (640x480), 51 Bilder/s für XGA-Auflösung (1024x768), 19 Bilder/s HDTV (1920x1080) und 1,28 Bilder/s (0,78 s/Bild) für Bilder mit 6464x4832 Pixel erreichen.

Der zugehörige **JPEG-Dekompressions-Core** (Decoder) arbeitet ebenfalls mit dem halben Systemtakt als Pixeltakt und erreicht gleich hohe Geschwindigkeiten. In praktischen Einsätzen vollständiger Applikationen konnte mit einem Spartan-3E (Speedgrade 4) der Decoder 495 Bilder/s QVGA (320x240) bzw. 120 Bilder VGA bei 96 MHz Systemtakt dekomprimieren.

3.2 Ein- und Ausgänge

Der JPEG-Codec besteht aus einem Encoder- und Decoderteil, **die auch parallel betrieben werden können**. Damit ist es zu Testzwecken möglich, die Ausgangssignale des Encoders dem Decoder zuzuführen und die Eingangs- und Ausgangsdaten des Codecs miteinander zu vergleichen. Die Beschreibung der Ein- und Ausgänge wird für den Encoder und Decoder getrennt behandelt.

Die folgende Beschreibung bezieht sich auf die Kernkomponenten der JPEG-Module (2D-DCT und VLC). Vorgelagerte Konvertierungen und Speicherblöcke sind in der Beschreibung der internen Einzelmodule ab Abschnitt 3.3 erläutert. Die Datenformate (Bitbreiten etc.) beziehen sich jedoch auf die folgenden allgemeinen Beschreibungen.

Der **JPEG-Encoder** besitzt 6 Eingänge und 5 Ausgänge (Bild 1), von denen ein Eingangssignal *DIN* die Komponentensignale des Bilddaten-/Videostromes mit der zeitlichen Abfolge 4:2:2 für *YUYV* bzw. *YCbYCr* erwartet. Mit den eingangsseitigen Steuerungssignalen *EN1* und *ENZ1* wird die gültige Zuordnung zu den Bild-/Videodaten festgelegt (Bild 2). Sind diese Signale „0“, so können die Bild-/Videodaten beliebige Werte annehmen, ohne Einfluß auf die Komprimierung zu haben.

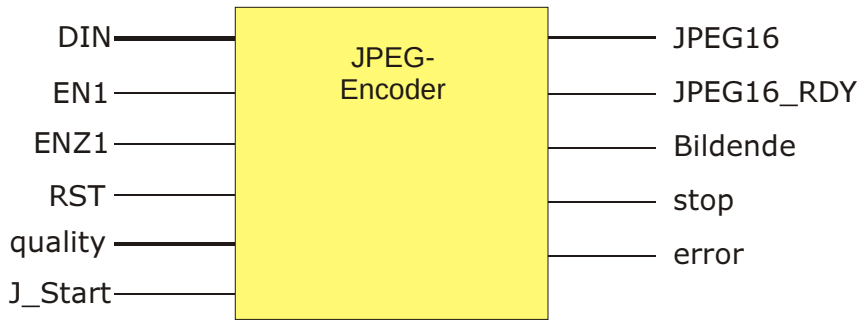

Bild 1: Symbol des JPEG-Encoders mit Ein- und Ausgängen

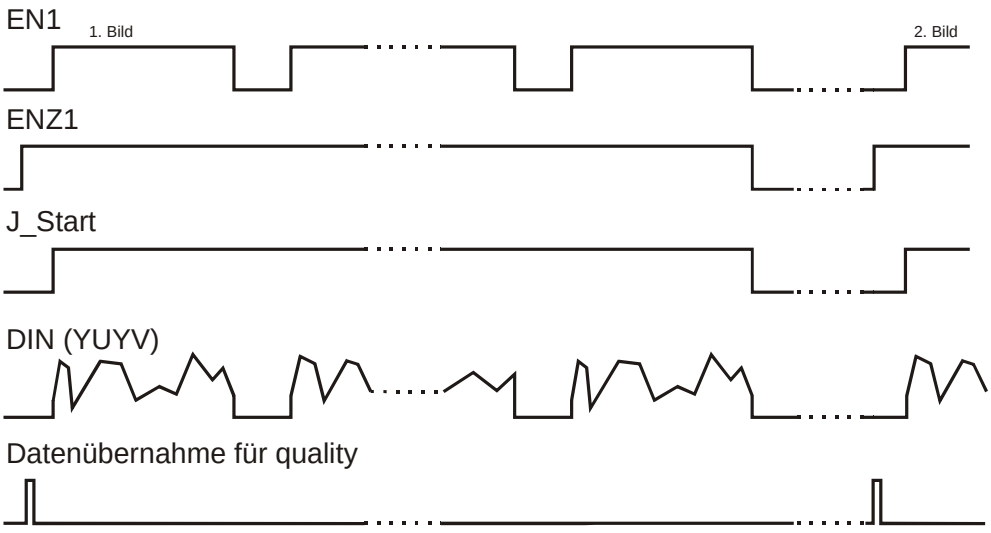

Bild 2: Beispielhafte Darstellung der Eingangssignale zur Steuerung des JPEG-Encoders

Mit den zwei bzw. drei bit (je nach Version) des Eingangssignals „*quality*“ kann eine der vier bzw. sechs definierten Quantisierungstabellen gewählt werden. Diese Einstellung hat entscheidenden Einfluß auf die Komprimierung bzw. Bildqualität des komprimierten Bildes, aber auch auf den Datendurchsatz. Die Qualitäts- bzw. Komprimierungseinstellungen (quality-Signal) können zu beliebigen Zeiten geändert werden. Die Übernahme der Einstellungen erfolgt mit dem Beginn des folgenden Bildes (Bild 2). Die Qualitätsstufe jedes Bildes erlaubt eine eindeutige Zuordnung des JPEG-Headers (inkl. der Quantisierungstabellen) zu jedem Bild. Damit können in einem Videostrom (MJPEG) auch unterschiedliche Qualitätsstufen für einzelne Bilder gesetzt werden.

Durch das Signal „J_Start“ wird der Datenstrom freigegeben (*J_Start* = „1“ bis *Bildende*). Damit kann die Ausgabe der komprimierten Daten nach der Ausgabe des Headers gestartet.

Signal	**Zahlenformat**	**Beschreibung des Einganges**
DIN	8 bit unsigned	Dateneingang unkomprimierter Bildpixel Helligkeits- und Farbdifferenzsignale des Bildes (YUYV)
EN1	boolean	horizontales Synchronisationssignal (Enable) der Bild- bzw. Videodaten (s. Bild 2); High aktiv; jedes gültige Bildpixel wird mit einem zugehörigem EN1=1 quittiert.
ENZ1	boolean	vertikales Synchronisationssignal (Enable) der Bild- bzw. Videodaten (s. Bild 2); High aktiv; während allen Bildpixeln eines Bildes (unabhängig von Blankings oder Datenlücken) wird mit einem zugehörigem ENZ1=1 bis Bildende quittiert.
RST	boolean	Resetsignal für den gesamten Encoder; High aktiv; EN1 und ENZ1 sollten bei aktivem RST 0 sein und erst bei einem neuen gültigen Bildanfang wieder aktiviert werden. Während RST=1 erfolgt keine Komprimierung.
quality	2 bit unsigned	Qualität und Kompressionsgrad der Videobilder (Einstellung einer der vier bzw. sechs definierten Quantisierungstabellen); Wert wird mit der steigenden Flanke von ENZ1 übernommen (Bild 2)
J_Start	boolean	Kontrollsignal zum Freigegeben des Datenstromes (*J_Start* = „1“ bis *Bildende*). Start der komprimierten Daten nach der Ausgabe des Headers.

Tabelle 1: Auflistung der JPEG-Encoder Eingänge

Das Ausgangssignal „*JPEG16*“ liefert kürzestens alle 2 Systemtakte einen 16 bit Wert der komprimierten Bilddaten. Bei hoher Komprimierung bzw. geringen Änderungen der Bildinhalte reduziert sich diese Datenrate. **Optional** wird ein FIFO-Modul angeboten, mit dem die Pufferung der Ausgangsdaten bis zum Auslesen auf Anforderung erfolgt.

Die Datenausgabe erfolgt mit einem Headermodul (separates Modul entsprechend Kundenwunsch), der alle Informationen zur Decodierung enthält. Für die universelle Verwendung der Komprimierung in Systemen mit geringer Übertragungsrate beinhalten die Ausgangsdaten optional einen verkürzten Dateiheader. Dieser kann in Abhängigkeit von der Qualitätsstufe und Ereigniseinstellungen später angefügt werden oder durch die direkte Übertragung zum Decoder nur die notwendigsten Informationen beinhalten. Die geringere Datenmenge schafft Stabilität bzw. höhere mögliche Bildqualitäten.

Die Gültigkeit des 16 bit JPEG-Datenwertes wird mit „*JPEG16_RDY* = 1“ signalisiert. Die Gültigkeit wird mit der Dauer eines Systemtaktes zur gleichzeitigen Datenübernahme eines *JPEG16*-Wertes ausgegeben (Bild 3). Das *JPEG16_RDY*-Signal geht – wie die JPEG-Daten - kürzestens alle 2 Systemtakte einmal auf „1“. Bei hoher Komprmierung bzw. geringen Änderungen der Bildinhalte vergrößert sich der zeitliche Abstand.

Bild 3: Beispiel des zeitlichen Verhaltens der JPEG-Encoder Ausgänge

Das Signal „*Bildende*“ = 1 wird das Ende des aktuell ausgegebenen Bildes definiert. Dieses Signal wird nach dem letzten gültigen Datenwert (*JPEG16_RDY*=1) gesetzt.

Das Signal „*error*“ kennzeichnet interne Fehlfunktionen (z.B. Pufferüberlauf, Zählerüberschreitung) und gibt im System die Möglichkeit der Verifikation und Fehlerkontrolle. Weiterhin kann mit diesem Signal das komprimierte Bild eines Videostromes verworfen werden. Der JPEG-Encoder muß danach mit „*RST*“ rückgesetzt werden.

Das Signal „*stop*“ wird ausgegeben (=1), wenn interne Puffer und Zähler kurz vor dem Überlauf stehen. Bei „*stop*“ = 0 können die Bilddaten wieder in den Core eingeschrieben werden. Für gepufferte Bildsysteme (ohne festes Videotiming), bei

denendie Bilddaten aus einem RAM geliefert werden, kann so die Geschwindigkeit maximiert werden. Das Signal stellt den Handshake für vorgelagerte Datenpuffer dar.

Eine Übersicht zu den Ausgängen ist in der Tabelle 2 aufgelistet.

Signal	Zahlenformat	Beschreibung
JPEG16	16 bit unsigned	Komprimierte JPEG-Daten
JPEG16_RDY	boolean	Gültigkeitssignal der JPEG-Ausgangsdaten für einen Systemtakt; High aktiv (s. Bild 3)
Bildende	boolean	Ende des aktuell ausgegebenen Bildes; High aktiv (s. Bild 3)
error	boolean	Signal für einen internen Fehler, d.h. ungültiges Bild; High aktiv
Stop	boolean	Interne Puffer kurz vor Überlauf, Dateneingabe anhalten; High aktiv

Tabelle 2: Auflistung der JPEG-Encoder Ausgänge

3.3 Beschreibung der Encoder-Module

Der vollständige JPEG-Encoder besteht aus den unten aufgeführten und in Bild 4 dargestellten Modulen, die auch einzeln und in verschiedenen Varianten verfügbar sind. Diese Varianten werden in den einzelnen Abschnitten beschrieben.

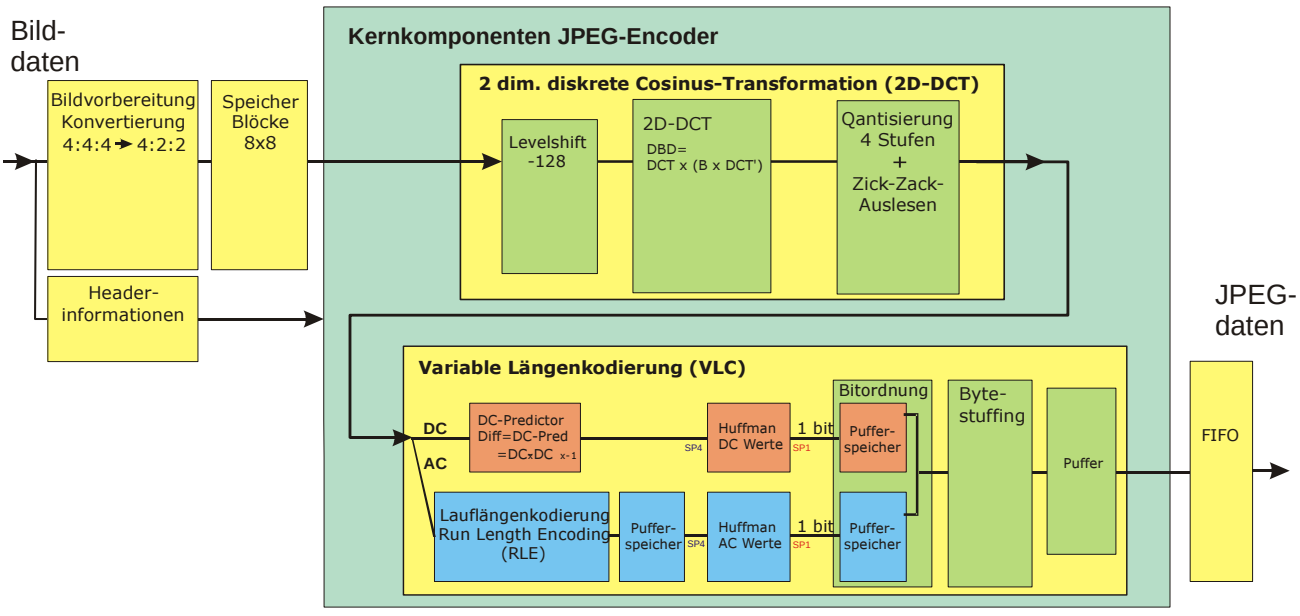

Bild 4: Vollständige Darstellung der JPEG-Encoder Module

Der Encoder besteht aus folgenden Modulen:

- Bildvorbereitung, Aufbereitung der YUV-Eingangsdaten
- Bilden der 8 x 8 Pixel Blöcke für die Dateneingabe zur 2D-DCT in internem oder externem RAM je nach Anzahl der Pixel je Zeile des Bildes (Speicherung von 16 Zeilen: Bsp. VGA 16 Zeilen x 640 Pixel = 16 Zeilen x 640 Pixel x 16 bit = 160 kbit)
- Headerinformationen zu Beginn des komprimierten Bildes in den Ausgangs-FIFO schreiben
- **Diskrete Cosinus-Transformation** (2D-DCT, Kernkomponente) mit
  - Pegelverschiebung (Level shift)
  - 2-dimensionale diskrete Cosinus-Transformation
  - Quantisierung mit verschiedenen Quantisierungstabellen
  - Zick-Zack-Auslesen (Zig-Zag-Scanning)
- **Variable Längenkodierung** (VLC, Kernkomponente) mit
  - Aufteilung der DC- und AC-Anteile
  - DC-Predictor
  - AC-Lauflängenkodierung
  - Huffmankodierung der DC- und AC-Anteile
  - Bitordnung der Blockreihenfolge
  - Bytestuffing als Zusammensetzen der Bytefolge
  - 32-bit Ausgangspuffer
- FIFO zum Puffern der JPEG-Ausgangsdaten (Größe je nach Kundenwunsch)

3.3.1 Bildvorbereitung, Aufbereitung der YUV-Eingangsdaten

Die Eingänge des Moduls zur Bildvorbereitung entsprechen den Eingängen des gesamten Encoders und bedürfen hier keiner weiteren Erläuterung. In diesem Modul werden die YUV- bzw. YCbCr- Eingangsdaten mit der zeitlichen Reihenfolge 4:4:4 (4Y:4U:4V) in das Format 4:2:2 (4Y:2U:2V) konvertiert. Hierbei werden die Farbdifferenzsignale (Chrominanz) U, V bzw. Cb, Cr unterabgetastet. Man macht sich dabei zunutze, daß das menschliche Auge für Helligkeitsunterschiede deutlich empfindlicher ist, als für Farbunterschiede. Es ist somit nicht notwendig, beim YUV-Format alle Farbwerte zu speichern. Damit der Verlust an Bildqualität gering bleibt, durchlaufen die Farbsignale vor der Unterabtastung einen Tiefpaß. Die ursprüngliche Datenmenge von 24 bit wird damit auf 16 bit mit minimalen Qualitätseinbußen reduziert. Die Reihenfolge der weitergeleiteten Daten ist in Tabelle 3 dargestellt.

Helligkeit	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	usw.
Farbsignal	U_{12}	V_{12}	U_{34}	V_{34}	U_{56}	V_{56}	usw.

Tabelle 3: Darstellung der Ausgangsbilddaten nach der Konvertierung

Des Weiteren werden in diesem Modul die Zuordnung der Dateiheader bzw. der „control“- und „evnt“-Signale zu den Bildanfängen synchronisiert.

3.3.2 Bilden der 8 x 8 Pixel Blöcke für die Dateneingabe zur 2D-DCT

Die Bild-/Videodaten sind bei der Eingabe zeilenweise orientiert, müssen jedoch für die Kodierung in Blöcken von 8 x 8 Pixel und in der normkonformen Blockreihenfolge der Komponenten (Helligkeit und Farbsignale) an die DCT weitergegeben werden. Deshalb sollte die Anzahl der Pixel/Zeile und die Zeilenanzahl ein mehrfaches von 8 sein. Die Struktur dieser Blöcke ist in Tabelle 4 dargestellt (Y_{ab} mit a… Zeile, b… Spalte).

Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}	Y_{17}	Y_{18}
Y_{21}	Y_{22}	Y_{23}	Y_{24}	Y_{25}	Y_{26}	Y_{27}	Y_{28}
Y_{31}	Y_{32}	Y_{33}	Y_{34}	Y_{35}	Y_{36}	Y_{37}	Y_{38}
Y_{41}	Y_{42}	Y_{43}	Y_{44}	Y_{45}	Y_{46}	Y_{47}	Y_{48}
Y_{51}	Y_{52}	Y_{53}	Y_{54}	Y_{55}	Y_{56}	Y_{57}	Y_{58}
Y_{61}	Y_{62}	Y_{63}	Y_{64}	Y_{65}	Y_{66}	Y_{67}	Y_{68}
Y_{71}	Y_{72}	Y_{73}	Y_{74}	Y_{75}	Y_{76}	Y_{77}	Y_{78}
Y_{81}	Y_{82}	Y_{83}	Y_{84}	Y_{85}	Y_{86}	Y_{87}	Y_{88}

Y_{19}	Y_{110}	Y_{111}	Y_{112}	Y_{113}	Y_{114}	Y_{115}	Y_{116}
Y_{29}	Y_{210}	Y_{211}	Y_{212}	Y_{213}	Y_{214}	Y_{215}	Y_{216}
Y_{39}	Y_{310}	Y_{311}	Y_{312}	Y_{313}	Y_{314}	Y_{315}	Y_{316}
Y_{49}	Y_{410}	Y_{411}	Y_{412}	Y_{413}	Y_{414}	Y_{415}	Y_{416}
Y_{59}	Y_{510}	Y_{511}	Y_{512}	Y_{513}	Y_{514}	Y_{515}	Y_{516}
Y_{69}	Y_{610}	Y_{611}	Y_{612}	Y_{613}	Y_{614}	Y_{615}	Y_{616}
Y_{79}	Y_{710}	Y_{711}	Y_{712}	Y_{713}	Y_{714}	Y_{715}	Y_{716}
Y_{89}	Y_{810}	Y_{811}	Y_{812}	Y_{813}	Y_{814}	Y_{815}	Y_{816}

U_{11}	U_{12}	U_{13}	U_{14}	U_{15}	U_{16}	U_{17}	U_{18}
U_{21}	U_{22}	U_{23}	U_{24}	U_{25}	U_{26}	U_{27}	U_{28}
U_{31}	U_{32}	U_{33}	U_{34}	U_{35}	U_{36}	U_{37}	U_{38}
U_{41}	U_{42}	U_{43}	U_{44}	U_{45}	U_{46}	U_{47}	U_{48}
U_{51}	U_{52}	U_{53}	U_{54}	U_{55}	U_{56}	U_{57}	U_{58}
U_{61}	U_{62}	U_{63}	U_{64}	U_{65}	U_{66}	U_{67}	U_{68}
U_{71}	U_{72}	U_{73}	U_{74}	U_{75}	U_{76}	U_{77}	U_{78}
U_{81}	U_{82}	U_{83}	U_{84}	U_{85}	U_{86}	U_{87}	U_{88}

V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}
V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}	V_{27}	V_{28}
V_{31}	V_{32}	V_{33}	V_{34}	V_{35}	V_{36}	V_{37}	V_{38}
V_{41}	V_{42}	V_{43}	V_{44}	V_{45}	V_{46}	V_{47}	V_{48}
V_{51}	V_{52}	V_{53}	V_{54}	V_{55}	V_{56}	V_{57}	V_{58}
V_{61}	V_{62}	V_{63}	V_{64}	V_{65}	V_{66}	V_{67}	V_{68}
V_{71}	V_{72}	V_{73}	V_{74}	V_{75}	V_{76}	V_{77}	V_{78}
V_{81}	V_{82}	V_{83}	V_{84}	V_{85}	V_{86}	V_{87}	V_{88}

Tabelle 4: Darstellung der vier ersten Blöcke mit den Signalen für Y, U (Cb) und V (Cr)

Für die Zusammenstellung dieser Blöcke müssen 8 Zeilen vollständig geladen werden, bevor mit der Blockausgabe begonnen werden kann. Fast immer kommen die Daten als kontinuierlicher Bilddatenstrom, weshalb im Speicher 16 Zeilen abgelegt werden müssen. Die erste Zeile des Bildes für den letzten 8 x 8 Block wird dabei noch gelesen, während dem die 16. Zeile bereits einspeichert wird. Daraus ergeben sich für die 16 bit/Pixel notwendige Speichergrößen, wie sie in Tabelle 5 aufgelistet sind.

Tabelle 5 zeigt deutlich, daß nicht jeder mögliche Speicherbedarf auch innerhalb des FPGA mit BlockRAM gedeckt werden kann. Für größere Bildauflösungen wird deshalb die Verwendung von externem Speicher empfohlen. Für den JPEG-Encoder kann auf Kundenwunsch das Speichermodul an die Bildgröße angepaßt bzw. ersetzt werden. Die Videosignale nach Tabelle 5 werden vom Speichermodul seriell (Blockpixel für Blockpixel und Block für Block) ausgegeben.

Auflösung		Pixel/Zeile	Speichergröße
VGA	640 x 480	640	160 kbit
SVGA	800 x 600	800	200 kbit
XGA	1024 x 768	1024	256 kbit
SXGA	1280 x 1024	1280	320 kbit
HDTV	1920 x 1080	1920	480 kbit
3,1 MPx	2048 x 1536	2048	512 kbit
5 MPx	2592 x 1944	2592	648 kbit
8 MPx	3264 x 2448	3264	816 kbit
32 MPx	6500 x 4875	6500	1625 kbit

Tabelle 5: Speicherplatzbedarf für die Bildung der 8 x 8 Blöcke (1 kbit = 1024 bit)

3.3.3 Headerinformationen

Zu Beginn des komprimierten Bildes werden Headerinformationen in den Ausgangs-FIFO geschrieben, die Informationen der Bildgröße, -auflösung, -qualität und Art der VLC beinhalten. Dieser Header ist standardkonform und kann als Kommentare auch Zusatzinformationen (Bildauswertung, Vorverarbeitung,…) besitzen. An den Header schließt sich das komprimierte Bild an. Damit keine Bildinformationen vor oder während des Headers in den FIFO geschrieben werden, kann mit dem Signal *J_Start* die Ausgabe der komprimierten Bilddaten definiert gesteuert werden, so dass dieses Signal erst nach beendetem Header auf high gesetzt wird.

3.3.4 Diskrete Cosinus Transformation (DCT)

Pegelverschiebung (Level shift)

Bei der Pegelverschiebung werden die Komponenten der Bilddaten von ihren verschiedenen Zahlenformaten bereinigt und besitzen danach das vorzeichen-behaftete 8 bit Format. Werden die Daten im Vorfeld der DCT-Module schon in der genannten Weise aufbereitet, kann die Pegelverschiebung optional entfallen.

2-dimensionale diskrete Cosinus-Transformation (2D-DCT)

JPEG, MJPEG und MPEG verwenden die zweidimensionale diskrete Cosinus Transformation zur Kodierung der Bild-/Videodaten. Bei dieser Transformation werden die Bilddaten vom Zeit- in den Frequenzbereich umgerechnet. Die DCT rechnet dabei einen zweidimensionalen Bilddatenbereich (Block) in einen zweidimensionalen Frequenz-bereich gleicher Größe um. In der o.g. Norm des JPEG-Standards werden immer Blöcke der Größe 8 x 8 Pixel transformiert. Die zweidimensionale DCT läßt sich aus der Durchführung der eindimensionalen DCT über die Zeilen und über die Spalten des Blocks kombinieren.

Ohne näher auf die einschlägige und umfangreiche Literatur zur Cosinus Transformation einzugehen, sei die Bestimmungsgleichung in Matrizenform dargestellt:

$$\mathbf{DBD = DCT \cdot (B \cdot DCT^{t})}$$

Dabei ist **B** die durch die 8 x 8 Blöcke gebildete Eingangsmatrize, **DCT** die Matrize der Koeffizienten der Cosinus Transformation, **DCT** ^{t} die transponierte Matrix von DCT und **DBD** die aus der Gleichung als Ergebnis erhältliche transformierte Matrix. Nach der Berechnung der DCT erhält man eine Matrix von 8 x 8 Frequenzen. Dabei steht die niedrigste Frequenz (f = 0) in der ersten Zelle (Zeile1, Spalte1). Diese Zelle wird Gleichanteil oder DC-Wert genannt. Die anderen Zellen enthalten die Amplituden der höheren Frequenzen (AC-Werte) die bis zur letzten Zelle (Zeile 8, Spalte 8) zunehmen. Diese Koeffizienten repräsentieren mit steigendem Abstand zum DC-Wert höhere Frequenzen, wobei die höheren vertikalen Frequenzen durch höhere Zeilenindizes repräsentiert werden und die höheren Horizontalfrequenzen durch größere Spaltenindizes. Die Transformation der Blöcke bzw. Matrizen durch die DCT führt nicht zu einem Qualitätsverlust, von Rundungsfehlern abgesehen. Die Ausgangssignale sind vorzeichenbehaftete 11 bit Daten mit der gleichen Datentaktrate wie die Eingangssignale.

Quantisierung mit verschiedenen Quantisierungstabellen

Bei der Quantisierung werden die 2-dimensionalen Eingangsmatrizen **DBD** mit der 2-dimensionalen Quantisierungsmatrize **Q** dividiert. Die Quantisierungsmatrize wird als Tabelle implementiert. Insgesamt stehen 6 vordefinierte Tabellen zur Auswahl, die über das Eingangssignal „quality“ gewählt werden können. Sollen andere Quantisierungstabellen zum Einsatz kommen, so ist dies optional möglich. Die Matrizengleichung der Quantisierung ist:

$$\mathbf{DBDQ = Runden(DBD/Q)}$$

Die entstehende Matrize **DBDQ** ist der auf Integerwerte gerundete Ergebnisblock. Mit der Quantisierungstabelle wird festgelegt, welche Bildqualität bzw. Kompressionsrate erreicht werden soll. Jede Zelle der Quantisierungstabelle kann Werte zwischen 1 und 255 annehmen. In der Norm vorgeschlagene Quantisierungsmatrizen sind in Tabelle 6 (quality = 3) dargestellt.

Bei der Quantisierung sollen Werte zu 0 gesetzt werden, die auf die Qualität des komprimierten Bildes kaum Auswirkungen haben. Da das menschliche Auge hohe Frequenzen nur gering beachten kann, werden diese hohen Frequenzanteile durch besonders hohe Werte geteilt, so daß das Ergebnis 0 ist. Damit kann eine hohe Datenkomprimierung erreicht werden. Bei der JPEG-Komprimierung ist die Quantisierung das eigentliche verlustbehaftete Verfahren.

quality = 3							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

quality = 2							
8	6	5	8	12	20	26	31
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	28
7	9	11	15	26	44	40	31
9	11	19	28	34	55	52	39
12	18	28	32	41	52	57	46
25	32	39	44	52	61	60	51
36	46	48	49	56	50	52	50

Tabelle 6: Vorgeschlagene Quantisierungstabellen (CCITT IT.81)

Es können für die Komponenten der Bildsignale (Y bzw. U,V) unterschiedliche Quantisierungstabellen verwendet werden. In diesem JPEG-Encoder wird die gleiche Quantisierungstabelle für alle Komponenten benutzt. Sie weicht geringfügig vom Normenbeispiel ab. Die Eingangs- und Ausgangsdaten haben die gleiche Taktrate.

Zick-Zack-Auslesen (Zig-Zag-Scanning)

Das Auslesen der Ergebnisblöcke erfolgt nicht linear, sondern in einer „Zick-Zack“ Reihenfolge durch die Matrize. Die durchnumerierte Reihenfolge des Auslesens ist in Tabelle 7 dargestellt.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Tabelle 7: Reihenfolge beim Auslesen der quantisierten Matrizen

Das Ziel dieses Auslesens ist es, die niedrigen Frequenzanteile zu Beginn des Datenstromes zu konzentrieren und die hohen Frequenzanteile, deren Amplituden sehr klein bzw. 0 sein sollten, an das Ende des Datenstromes eines Blockes zu stellen. Dadurch wird bei der Lauflängenkodierung eine große Anzahl aufeinander folgender Nullen erzielt, um die Komprimierung der Datenmengen zu erhöhen. Die Eingangsdaten **DBDQ** und Ausgangsdaten **DBDQZZ** haben die gleiche Taktrate, die danach an die variable Längenkodierung weitergeleitet werden.

3.3.5 Variable Längenkodierung (variable length code - VLC)

Aufteilung der DC- und AC-Anteile

Bei der weiteren Bearbeitung des Datenstromes wird der Gleichanteil (DC-Anteil) von den Anteilen höherer Frequenz (AC-Anteile) getrennt und separat weiterverarbeitet.

DC-Predictor

Der Datenpfad des **DC**-Anteils erhält je Block einen Datenwert. Von diesem Datenwert wird der DC-Anteil des vorangehenden Blockes abgezogen.

DIFF = DC - Predictor

Dieser vorangehende Anteil wird **Predictor** genannt. Die Differenz **DIFF** wird der weiteren Entropiekodierung (Huffman) zugeführt. In den häufigsten Fällen ist diese Differenz deutlich kleiner als der DC-Wert selbst, so daß bei der Kodierung eine geringere Datenmenge entsteht.

AC-Lauflängenkodierung

Die AC-Lauflängenkodierung umfaßt das Zählen der Nullen zwischen den Frequenzwerten, die Vorzeichenbestimmung und die Größenbestimmung des Wertes als Betrag und Code. Diese Lauflängenkodierung wird in der Norm Run Length Encoding (RLE) genannt.

Huffmankodierung der DC- und AC-Anteile

Die Entropiekodierung mittels des Huffman-Algorithmus wird für die DC- und AC-Werte getrennt durchgeführt. Beim Huffman-Verfahren kodiert man häufig vorkommende Werte mit wenigen und seltene Werte mit mehr Bits. Zur Festlegung der jeweiligen Bit-Kodierungen sind die Zeichenhäufigkeiten zu ermitteln. Für die AC-Kodierung gehen neben den Werten zusätzlich die Anzahl der Nullen zwischen den Werten mit ein.

Die Häufigkeit der Wertegrößen und Lauflängen ist in Tabellen zusammengefaßt, die in der Norm spezifiziert sind. So wird bei der Huffman-Kodierung gefordert, daß eine oder mehrere Huffmantabellen spezifiziert werden. Diese Huffmantabellen werden dann zur Kodierung als auch zur Dekodierung verwendet. Damit sind im Header neben der Quantisierungstabelle auch die Huffmantabellen mit enthalten.

Die Ausgangsdaten haben teilweise eine Taktrate (Faktor 4 für Encoder höherer Qualität, Faktor 1 für Encoder niedrigerer Qualität) als die Eingangsdaten, da die Ausgangsdaten als serieller Bitstrom vorliegen.

Bitordnung der Blockreihenfolge

Bei der Bitordnung muß die Reihenfolge aus separat kodierten DC- und AC-Werten wieder in der Reihenfolge der Blöcke zu einem Bitstrom zusammengesetzt werden. Dabei müssen auch gepufferte Daten, die noch im AC-Pufferspeicher liegen, mit berücksichtigt werden.

Bytestuffing als Zusammensetzen der Bytefolge

Im Modul Bytestuffing wird der einkommende zeitlich unregelmäßige Bitstrom zu Bytes zusammengesetzt. Diese Bytes werden getestet und danach zu 2 Byte-Werten kombiniert. Die Bildanfangs- (FF D8) und Bildendemarkierungen (FF D9) werden ebenfalls in das komprimierte Bild eingefügt.

16-bit Ausgangspuffer

Für die definierte Datenausgabe von 16 bit ist ein Ausgangsspeicher als Datenpuffer der Bytewerte das abschließende Modul des JPEG-Encoders.

3.3.6 FIFO-Ausgangspuffer

Optional ist ein Ausgangs-FIFO verfügbar, mit dem die Taktrate der Ausgangsdaten auf den Durchschnittswert bei maximaler Qualität des Bildes von 16 Systemtakten je *JPEG16*-Wert erreicht wird. Es können aber auch die externen Programmteile das Auslesen in Abhängigkeit des Füllstandes des FIFO steuern.

4. Revision History

Revisionsnummer	Änderung	Änderungsdatum
1.0	Originaldokument, Ausgabe als Kurzdokumentation	20.06.2004
1.1	vollständige Beschreibung des Encoders nach Baseline process	10.07.2007
1.2	Änderung 1. und 4. im Text, Vorbereitung zur Einbeziehung des Decoders in die Dokumentation	24.02.2008
1.3	Einbeziehung des Decoders in die Dokumentation	12.09.2008
1.4	Platzverbrauch und Geschwindigkeit für Decoder aktualisiert	30.01.2009
1.5	• aktuelle Überarbeitung; • 3.1 Platzverbrauch und Geschwindigkeit; • Decoder in separates Dokument (IP_JPEG_Decoder_de01_05).	23.01.2014