Die RMW-Zyklusdauer t_{Zyklus} errechnet sich unter Beachtung der zusätzlichen externen Verzögerungen folgendermaßen:

$$t_{Zyklus} = \max(t_{HOLD}, t_{PRE}, t_{av} + t_{ADS}) + \max(t_{ACC}, t_{ADH}) + \max(t_{inc} + t_{SETUP}, t_{WRITE})$$

Die einzelnen Zeiten bedeuten:

t_{HOLD}: data hold time = Haltezeit, für die die Eingangsdaten des RAMs nach Ende des Schreibimpulses noch unverändert anstehen müssen

t_{PRE}: precharge time = Zeit, die vom RAM intern benötigt wird, um die Lese- und Schreibleitungen der Speichermatrix auf bestimmte Pegel voreinzustellen

t_{av}: address valid time = Zeit, die extern benötigt wird, um gültige Adressen zur Verfügung zu stellen; dies kann jedoch schon während des vorangehenden Schreibzyklus geschehen, da das RAM die Adresse zwischenspeichert

t_{ADS}: address setup time = Zeit, die die Adressen gültig sein müssen, bevor der Lesezugriff gestartet werden kann

t_{ACC}: access time = Zugriffszeit nach Übergabe gültiger Adressen

t_{inc}: incrementer time = Rechenzeit des Inkrementers bzw. Addierers, bis alle Ausgangsdaten gültig sind

t_{SETUP}: setup time = Zeit, die die Eingangsdaten des RAMs schon gültig sein müssen, bevor der Schreibimpuls endet

t_{WRITE}: write time = Minimale Dauer des Schreibimpulses

t_{ADH}: address hold time = Zeit, die die Adressen nach Start des Zugriffs noch gültig bleiben müssen

Für ein mit der ES2-Bibliothek ECPD10 [7] in 1µm-CMOS-Technologie hergestelltes 1K*8 RAM werden beispielsweise folgende Werte als typisch angegeben:

t_{PRE} = 5,3 ns
t_{ADS} = 0,0 ns
t_{ADH} = 2,0 ns
t_{ACC} = 8,9 ns + 0,2 ns je pF Last
t_{SETUP} = 3,3 ns
t_{WRITE} = 5,1 ns
t_{HOLD} = 0,0 ns

Die Maximalwerte (schlechtester Chip bei ungünstigsten Umgebungsbedingungen: Temperatur maximal und Versorgungsspannung minimal) liegen etwa um den Faktor 2 höher.

Die Formel für die Zykluszeit vereinfacht sich, wenn aufgrund der Maximumsbildung vernachlässigbare Zeiten weggelassen werden:

$$t_{Zyklus} = t_{PRE} + t_{ACC} + \max(t_{inc} + t_{SETUP}, t_{WRITE})$$

Damit ergibt sich allein aus diesen Zeiten eine minimale Zyklusdauer von ca. 20 ns, im schlechtesten Fall ca. 40 ns. Dazu kommt ein Teil der Zeit, die der Inkrementer bzw. Addierer benötigt und Verluste, die sich aus der Aufteilung der einzelnen Phasen des Read-Modify-Write-Zyklus auf mehrere Phasen des Systemtakts ergeben.

Erst nachdem der Histogrammteil implementiert war, stand fest, daß der Histogrammer mit einem asynchronen (statischen) RAM realisiert werden sollte, das am MAZ zur Verfügung stand.

Bei asynchronen RAMs entfällt die Precharge-Zeit und es ist kein Taktsignal notwendig. Die zwischen zwei RMW-Zugriffen benötigte Ruhepause ergibt sich dann aus der minimalen Adreß- bzw. Datenhaltezeit nach Deaktivierung des Schreib-Signals, je nachdem, welche Zeitspanne größer ist. Adreß- und Datenlatches müssen extern vorgesehen werden, wenn sie benötigt werden. Zusammen mit diesen Latches ergibt sich eine Ansteuerung analog zum untersuchten RAM.
Für das im Histogrammer verwendete RAM CRAM03A [4] gelten laut Herstellerunterlagen gemäß Abbildung 4.4 folgende Parameter als Maximalwerte:
t_{PRE} = 0 ns
t_{ADS} = -- ns
t_{ADH} = -- ns
t_{ACC} = 15 ns
t_{SETUP} = 4,5 ns
t_{WRITE} = 13 ns
t_{HOLD} = 1 ns

Bei beiden genannten RAMs können wahlweise gemeinsame oder getrennte Datenein- und Datenausgänge spezifiziert werden. Ein RAM mit getrennten Ein- und Ausgängen soll verwendet werden, um die Notwendigkeit eines Tristate-Busses zwischen RAM und Inkrementer zu vermeiden. Zwischen Datenausgang und -eingang kann eine ungetaktete Inkrementereinheit geschaltet werden.

4.3.3 Implementation des Read-Modify-Write-Zugriffs

Um den Read-Modify-Write-Zyklus optimal an die Erfordernisse des RAMs anzupassen, ist eine möglichst feine Zeitreferenz notwendig. Die feinste definierte Zeitunterteilung, die dem Histogrammer zur Verfügung steht, ist eine halbe Taktperiode, da der Takteingang (im Idealfall) im Abstand dieser Zeit toggelt. Allerdings ist die Länge einer halben Taktperiode im allgemeinen nicht so genau spezifiziert und nicht so stabil wie die Taktfrequenz und damit die Dauer von einem Takt bis zum Folgetakt. Die von gängigen Bausteinen wie dem 50 MHz 486-Prozessor [12] geforderte Taktstabilität liegt bei beispielsweise 0,1% gemessen an aufeinanderfolgenden Takten. Diese Anforderung wird von Quarzoszillatoren problemlos erfüllt. Die Anforderungen an das Tastverhältnis sind geringer, da die Zeit, in der ein negatives oder positives Signal am Takteingang erkannt wird, beeinflußt wird durch die Symmetrie des Quarzoszillators, die positive und negative Schaltschwelle des Takteingangs, den extern zur Verfügung stehenden Spannungshub und die Flankensteilheiten. Die meisten dieser Parameter sind ferner stark temperaturabhängig und unterliegen Exemplarstreuungen. Der Histogrammer soll mit Pixelfrequenzen bis zu ca. 25 MHz arbeiten können. Wird zur Ablaufsteuerung die doppelte Frequenz benötigt, so muß die Taktfrequenz im Bereich von ca. 30 bis 50 MHz liegen. Quarzoszillatoren bieten innerhalb dieses Frequenzbereiches eine Übereinstimmung des Tastverhältnisses mit dem idealen Tastverhältnis von 50% auf fünf Prozent, also schlechtestenfalls 45% oder 55% [27]. Innerhalb des Chips können allerdings zusätzlich noch geringfügige Verfälschungen durch unterschiedliche Anstiegs- und Abfallzeiten der Gatterausgänge entstehen. Um jedoch die benötigte Taktfrequenz des Chips nicht noch einmal verdoppeln zu müssen, ist es sinnvoll, mit halben Takten zu arbeiten.
Alternativ dazu wäre auch ein Trimmen von Signallaufzeiten durch Verzögerungselemente denkbar, um kurze Verzögerungen zu erzeugen. Dies erschwert jedoch das Design des Chip-Layouts erheblich, da die Signallaufzeiten wesentlich von den Leitungslängen abhängen und unter Umständen die einzelnen Signale eines Datenbusses auf dem Chip ganz unterschiedlich

lange Wege zwischen den Einheiten zurücklegen müssen. Ein solches Vorgehen wird von dem verwendeten Standardzellen-Entwurfssystem nicht unterstützt.

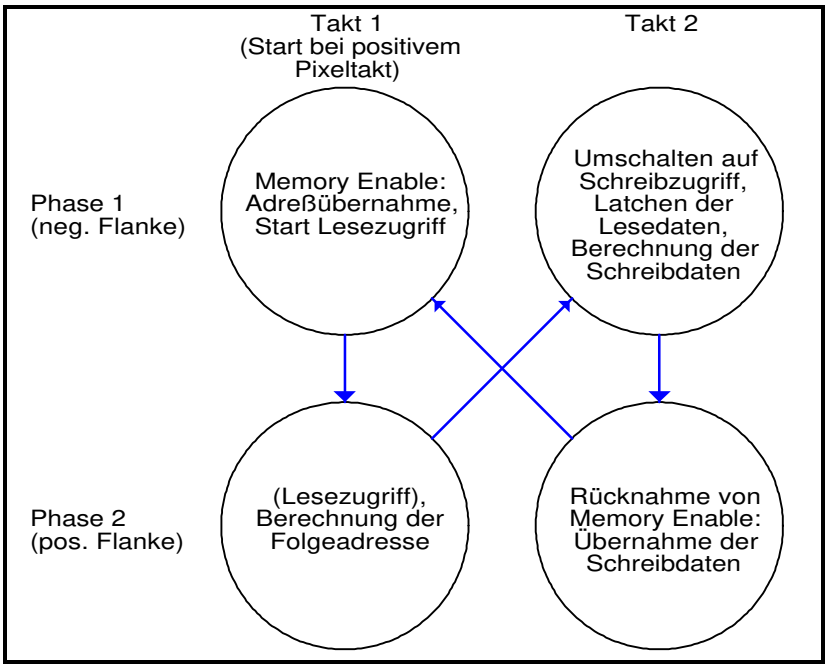

Abbildung 4.5: Aufteilung des RMW-Zugriff auf Taktphasen

Eine Realisierung des Read-Modify-Write-Zyklus erfordert minimal drei Taktphasen: Lesezugriff, Schreibzugriff und Precharge-Zeit. In der Precharge-Zeit erfolgt die Beendung des Schreibzugriffs und damit die Datenübernahme sowie die Vorbereitung auf den nächsten Zugriff. Da der Lesezugriff die längste Zeit in Anspruch nimmt, soll er auf zwei Taktphasen verlängert werden. Damit ist seine Dauer auch gemäß der hohen Stabilität der Taktfrequenz genau definiert. Die gewählte Aufteilung des RAM-Zugriffs auf vier Taktphasen, also zwei Takte, ist in Abbildung 4.6 gezeigt. Eine Aufteilung auf eine gerade Anzahl von Taktphasen ist auch deshalb notwendig, weil alle anderen getakteten Einheiten entweder positiv oder negativ flankengesteuert, also mit jedem Takt arbeiten.
Die vier Taktphasen können durch ein Schieberegister aus einem positiv und einem negativ flankengesteuerten Flip-Flop in einschrittiger Codierung erzeugt werden. Eingangssignal ist der auf einen Takt verkürzte Pixeltakt. So wird der Taktphasengenerator für jeden Pixeltakt gestartet. Damit ist die maximal mögliche Pixeltaktrate gleich der halben Taktfrequenz des Histogrammers.

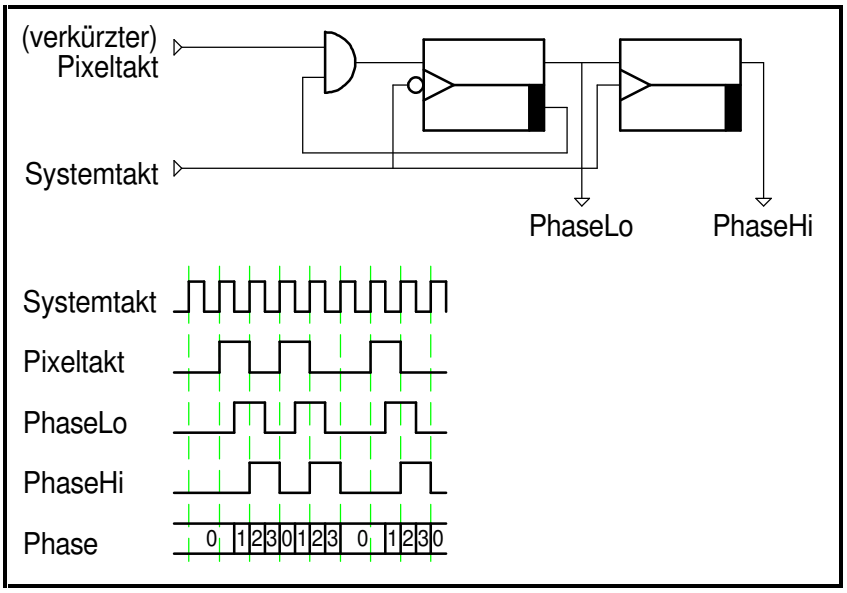

Abbildung 4.6: Erzeugung der Taktphasen

Für die Berechnung der Schreibdaten aus den Lesedaten (und den Pixeldaten) steht die Dauer von drei Taktphasen abzüglich der Lesezugriffszeit und der Setup-Zeit für die Daten beim Schreibzugriff zur Verfügung, also etwa eine halbe Taktperiode.

Für den Inkrementer bzw. Addierer zur Verfügung stehende Zeit:
$t_{inc} = 3*(t_{CLK}/2) - t_{ACC} - t_{SETUP}$

Wenn es bei Verwendung eines asynchronen RAMs gelingt, die Datenhaltezeit nach einem Schreibzugriff auch ohne zusätzlichen Zustand für die Precharge-Zeit zu garantieren, steht eine halbe Taktperiode mehr Zeit für den Schreibzugriff und damit auch für den Inkrementer zur Verfügung. Dies kann jedoch erst bei Simulation auf Gatterebene ermittelt werden.

4.3.4 Realisierung der ALU

Die ALU übernimmt alle Berechnungen auf dem Histogrammspeicher: Bei Histogrammerstellung inkrementiert die ALU den Speicherinhalt in Abhängigkeit von der Speicheraufteilung und den Fenstereinstellungen, addiert im Kachelmodus den aktuellen Grauwert zum Speicherinhalt und akkumuliert den Speicherinhalt bei Berechnung der Summenhäufigkeiten.

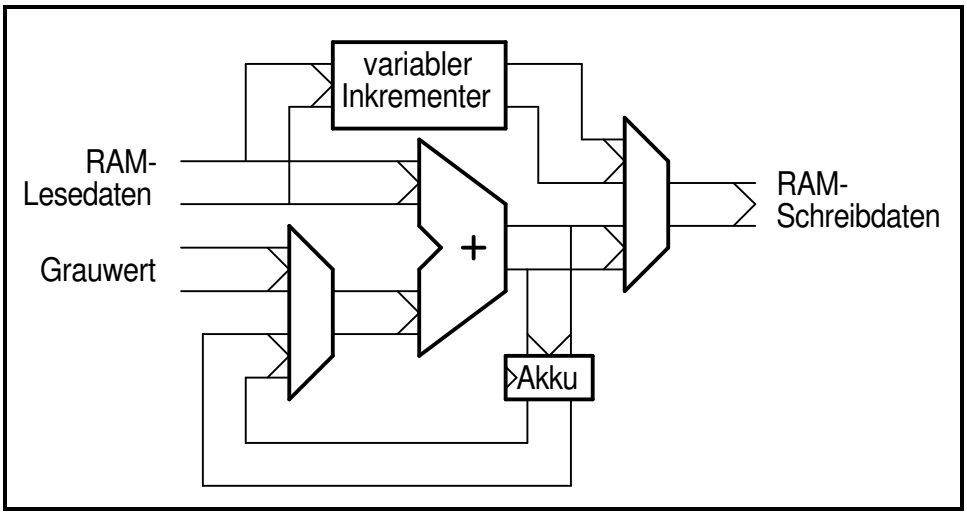

Abbildung 4.7: Blockschaltbild der ALU

Zur Berechnung der Kachelsummen ist ein schneller Addierer mit einer Breite von 24 Bit erforderlich, da 24 Bit des Speichers für Kachelsummen genutzt werden sollen. Das Argument ist der aktuelle Grauwert mit einer Breite von 10 Bit.
Zur Berechnung der Summenhäufigkeiten muß das Histogramm des betroffenen Fensters von Speicherstelle 0 anfangend aufsummiert werden. Die Berechnung dieser Summen erfordert ein zusätzliches Hilfsregister. Dieser Akkumulator wird zu Anfang der Berechnung gelöscht. In jedem Schritt wird der Inhalt der jeweiligen Histogrammspeicherstelle zum Wert im Akkumulator addiert und anschließend in den Akkumulator und die Histogrammspeicherstelle übertragen.
Der Addierer zur Berechnung der Kachelsummen kann auch zur Bildung der Summenhäufigkeiten verwendet werden, indem er über einen Multiplexer steuerbar wahlweise den Grauwert des aktuellen Pixels oder den Inhalt des Akkumulators zum Speicherinhalt addiert. Da jeweils nur eine der beiden Funktionen benötigt wird, kann auf diese Weise ein Addierer eingespart werden. Der Addierer muß über eine Breite von 24 Bit für beide Argumente verfügen. Dann ist es möglich, die Summenhäufigkeiten für ein bis zu 24 Bit breites Fenster zu bilden. Abbildung 4.7 zeigt den Aufbau der ALU.
Eine Berechnung von Summenhäufigkeiten für jedes Histogramm eines Bildes erscheint nicht als sinnvoll, da zu jedem Zeitpunkt nur anhand einer Summenhäufigkeitstabelle egalisiert werden kann. Die Aufteilung des Speichers auf die Fenster soll deshalb so gewählt werden, daß das Fenster, dem der größte Teil des Speichers zugeordnet werden kann, jeweils an der gleichen Position in jedem Speicherwort beginnt. Zweckmäßig ist die Wahl von Bit 0 als LSB des größten Fensters. So kann die Summenhäufigkeit über dieses Fenster in jeder Konfiguration gebildet werden, ohne daß ein weiterer Multiplexer zwischen RAM-Datenausgang und Eingang des Addierers benötigt wird.

Der Addierer hat im Gegensatz zum Inkrementer eine feste Breite und ist nicht von der Speicherkonfiguration abhängig (s. Kapitel 3.2.6). Damit kann er vom Entwurfssystem anhand von vorzugebenden Laufzeitanforderungen aus Gattern konstruiert werden oder als (parametrisierbare) Standardzelle generiert werden.
Der Histogramminkrementer muß in Abhängigkeit von der Speicheraufteilung auf die unterschiedlichen Fenster in ein bis vier Inkrementer mit variabler Breite aufgeteilt werden können. Ein solcher Inkrementer ist nicht als Standardzelle verfügbar.

Die sich aus seiner Position zwischen RAM-Datenausgang und RAM-Dateneingang und der maximalen Laufzeit von einer halben Taktperiode ergebenden Anforderungen an den Inkrementer lassen nur ein ungetaktetes Design mit einer geringen Anzahl von sequentiell durchlaufenen Gatterstufen zu. Weiterhin wichtig sind schnelle Steuerbarkeit (Umschaltung zwischen +1 oder +0) und eine einfache Aufteilbarkeit in mehrere separate Inkrementer (für die einzelnen Fenster).
Für den Fall, daß mehr Grauwerte auf eine Histogrammspeicherstelle entfallen, als mit der gewählten Bitanzahl darstellbar ist, soll ein Überlauf in jedem der einzelnen Teilinkrementer verhindert werden. Ein solcher Fall kann auftreten, wenn alle Punkte eines Fensters der Größe 2^n * 2^n den gleichen Grauwert haben, da das Ergebnis für den entsprechenden Grauwert dann 2^(2*n) ist, also 1 mehr, als mit 2*n Bit darstellbar. Ohne Überlaufschutz würde sich in einer 2*n Bit breiten Histogrammspeicherstelle der Wert 0 ergeben. Durch den Überlaufschutz wird der Inhalt der Speicherstelle auf den maximal darstellbaren Wert begrenzt.

Eine Möglichkeit, den Histogramminkrementer zu realisieren, ist eine separate Realisierung aller Inkrementer für jede vorkommende Breite und eine Auswahl der benötigten Inkrementer durch Multiplexer. Die Gesamt-Breite von 48 Bit und die Anzahl von acht unterschiedlichen Fensterkombinationen läßt diese Realisierung jedoch als zu aufwendig erscheinen. Deshalb soll im folgenden eine variabel zusammenschaltbare Inkrementer-Struktur untersucht werden.

Zuerst wird untersucht, wie ein einfacher Inkrementer zu realisieren ist und wie eine reguläre Struktur aussehen kann, die eine Binärzahl inkrementiert.
Ein binärer Inkrementer liefert an seinem Ausgang den um Eins erhöhten Eingangswert.

Tabelle 4.1: 3-Bit Inkrementer mit Überlaufschutz

Eingang	Ausgang
000	001
001	010
010	011
011	100
100	101
101	110
110	111
111	111

Aus der Tabelle kann man erkennen, daß ein Bit des Arguments vom Inkrementer genau dann invertiert werden muß, wenn alle rechten Nachbarn dieses Bits den Wert 1 haben. Dies kann mit einer einfachen Struktur aus Und-Gattern und Exklusiv-Oder-Gattern realisiert werden (Abbildung 4.8: 4-Bit Parallel-Inkrementer). Ein solcher Inkrementer arbeitet zweistufig, solange nicht bei den höherwertigen Bits die maximale Eingangsanzahl der Und-Gatter überschritten wird. Größere Und-Gatter muß der Silicon-Compiler durch einen Baum aus Und-Gattern ersetzen und es ergibt sich eine dreistufige Struktur.
Der gesamte 48 Bit breite Inkrementer kann aus einer Kette solcher Inkrementer zusammengesetzt werden, die nach Bedarf zu je einem Teilinkrementer pro Fenster kaskadiert werden. Der Überlaufschutz schaltet den Teilinkrementer auf Addition von Null um, wenn alle seine Eingänge auf 1 sind. Damit das Carry-Bit nicht durch die gesamte Inkrementerkette pro Fenster laufen muß, kann für weiter links befindliche Stufen ein paralleles Carry aus den Überlaufschutzausgängen (Clip) aller rechten Einzelinkrementer gebildet werden (Abbildung 4.9: 12-Bit Inkrementer aus 3 Einzelinkrementern).

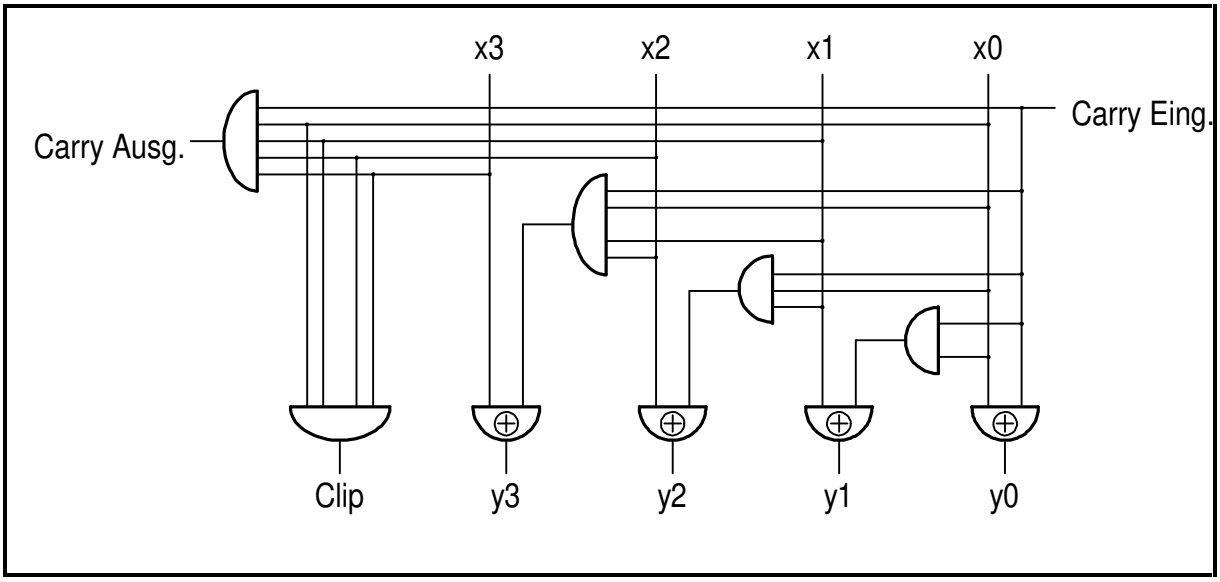

Abbildung 4.8: 4-Bit Parallel-Inkrementer

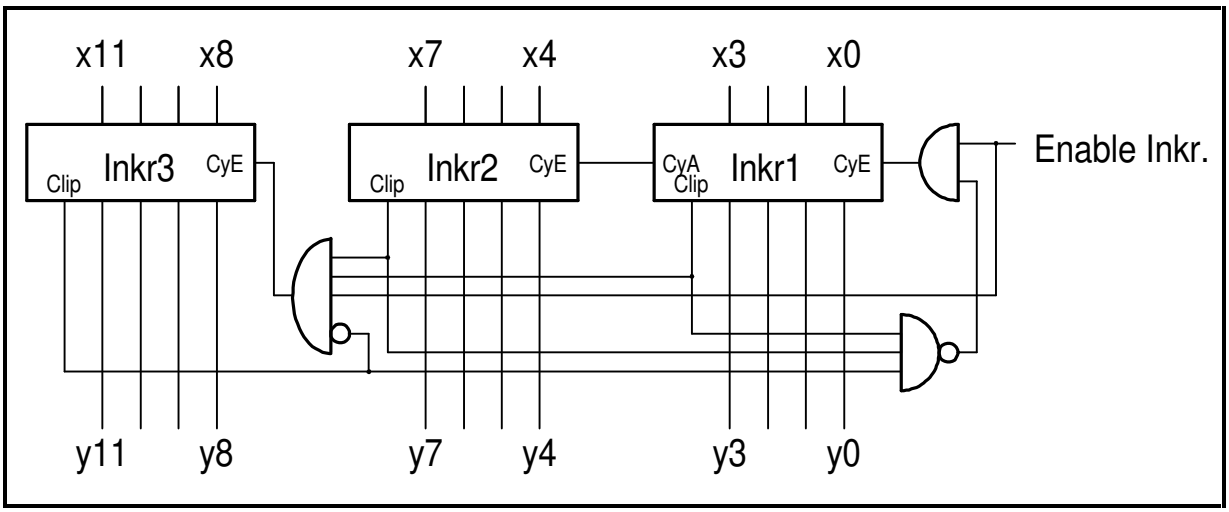

Abbildung 4.9: 12-Bit Inkrementer aus 3 Einzelinkrementern

4.4 Statistikteil

Der Statistikteil soll parallel über jedes der vier einstellbaren Fenster die fünf Größen Minimum, Maximum, Schwellensumme, Grauwertsumme und Quadratsumme direkt während der Bildübertragung an den Histogrammer bilden können.

Die Größen Grauwert-Minimum und -Maximum müssen mit einer Breite von 10 Bit entsprechend der Grauwertauflösung gebildet werden, indem für jeden Punkt innerhalb des Fensters geprüft wird, ob er die bis dahin gefundenen Extremwerte über- bzw. unterschreitet. Vor Bildung der Extrema muß das Minimum auf den größten darstellbaren Grauwert (1023) initialisiert werden, das Maximum auf den kleinstmöglichen (0).

Die maximale Größe der Schwellensumme entspricht der Anzahl der Punkte innerhalb des Fensters.

Da der Histogrammer nicht in jeder Speicherkonfiguration über alle vier Fenster ein Histogramm bilden kann und soll, stehen die jeweils ungenutzten Fenster allein für statistische Berechnungen zur Verfügung. Deshalb ist es sinnvoll, sämtliche Statistikregister für die einzelnen Fenster mit voller Breite auszulegen, so daß sich keine Einschränkungen für die Fenstergrößen ergeben wie bei der Histogrammerstellung. Die Register für die Schwellensumme müssen also 24 Bit breit sein, um im größtmöglichen Fenster (4096^2) sämtliche Punkte zählen zu können.

Die maximale Größe der Grauwertsumme und der Quadratsumme hängt von Grauwertauflösung und maximaler Fenstergröße ab. Der höchstmögliche Wert der Grauwertsumme ist 4096*4096*1023, also mit 34 Bit darstellbar, und die Grauwert-Quadratsumme analog dazu mit 44 Bit.
Die Bildung dieser Summen kann über ein Register geschehen, das für jedes Pixel innerhalb des Fensters getaktet wird. Das Register wird über einen Addierer rückgekoppelt. Da der Summand eine geringere Breite hat als die Summe, kann der Addierer in einen Addierer für die unteren und einen Inkrementer für die oberen Bits der Summe aufgeteilt werden (Abbildung 4.10: Bildung der Grauwertsummen).

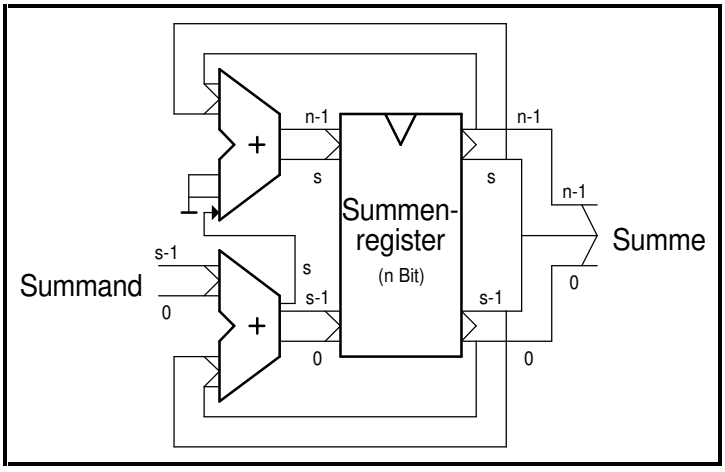

Abbildung 4.10: Bildung der Grauwertsummen

Die Geschwindigkeitsanforderung an die Addierer wird durch die Höhe des Pixeltaktes bestimmt. Solange mehrere zehn Nanosekunden für jede Addition zur Verfügung stehen, erscheint das Design unkritisch, da es dann sogar möglich ist, den Addierer als Ripple-Carry-Addierer oder aus kaskadierten 2-Bit-Carry-Look-Ahead-Addierern aufzubauen. Die Durchlaufzeit eines Ripple-Carry-Addierers liegt in der geplanten Zieltechnologie bei etwa 0,75 ns pro Bit. Sollte jedoch die maximale Taktfrequenz deutlich erhöht werden, wie dies bei Verwendung eines Dual-Port-RAMs im Histogrammteil möglich würde, stehen weniger Gatterlaufzeiten je Pixeltakt zur Verfügung. Die Addierer würden dann sehr groß, da die dann notwendigen parallelen Strukturen wesentlich mehr Gatter pro Bit benötigen als ein Ripple-Carry-Addierer. In diesem Fall sollte ein Pipelining eingeführt werden, um die Addierer aus mehreren kleinen Addierern zusammensetzen zu können. Da jedoch für die Berechnung jedes neuen Wertes bereits das vorige Ergebnis benötigt wird, bietet sich Pipelining nur für den Inkrementer-Teil der Addierer an. Eine einfache Möglichkeit dazu wäre der Aufbau des Inkrementer-Teils als einfacher Zähler aus kaskadierten Toggle-Flip-Flops. So müßten nur 10 bzw. 20 Bit breite Addierer für den unteren Teil der Summe eine Durchlaufzeit von unterhalb einer Pixeltakt-Periode aufweisen. Über das oberste Bit der Summe könnte der Zähler für die höherwertigen Bits getaktet werden.

4.4.1 Einsparmöglichkeiten beim Statistikteil

Der Statistikteil ist alleine durch die Anzahl von insgesamt 488 Registerbits sehr aufwendig. Auch wenn nur wenige zehn Gatteräquivalente zur Berechnung, Speicherung und Ausgabe jedes Bits notwendig sind, kommen insgesamt über 10000 Gatter zustande. Es ist somit abzusehen, daß der Statistikteil den größten Anteil von flächenintensiver „Random-Logik“ auf dem Chip stellen wird. Deshalb sollen hier Einsparmöglichkeiten untersucht werden.

Die Grauwertsumme und Quadratsumme liegt auch bei kleinen Fenstern leicht im Bereich einiger Millionen. Bei Statistikwerten dieser Größenordnungen ist keine absolute Genauigkeit notwendig. Eine deutliche Reduktion der Genauigkeit wäre akzeptabel. Berechnung mit geringerer Genauigkeit könnte den notwendigen Aufwand für Addierer und Register wesentlich reduzieren.

Eine Einsparung in der notwendigen Breite der Register und der Addierer wäre möglich, indem bei hoher Grauwertauflösung einige der niederwertigen Bits des Grauwertes ignoriert werden, oder bei großen Fenstern nur ein Teil der Pixel betrachtet wird.
Beide Ansätze sind jedoch problematisch, da unter Umständen auch bei hoher Grauwertauflösung Bildbereiche mit geringen Kontrast untersucht werden sollen und hier eine Reduktion der Grauwertauflösung wiederum inakzeptabel wäre. Das Auslassen von Pixeln bei großen Fenstern erfordert eine Kenntnis der Fenstergröße, was bei Aktivierung der Fenster durch externe Signale jedoch nicht im Voraus gegeben ist. Weiterhin kann es bei der Untersuchung regelmäßiger Muster zu total falschen oder bei Verschiebung des Objektes im Bild zu stark schwankenden Werten führen, wenn nur eine beschränkte Anzahl von regelmäßig über das Fenster verteilten Bildpunkten untersucht wird. Dieser Fall tritt beispielsweise auf, wenn beim Bild eines Schachbrettes alle untersuchten Bildpunkte in eine Kategorie von Feldern fallen.

Während die Berechnung der Grauwertsumme den 10 Bit breiten Grauwert als Argument erfordert, sind für die Quadratsumme 20 Bit und damit ein relativ breiter Addierer erforderlich. Da die Quadrate des Grauwertes für steigende Grauwerte jedoch schnell sehr groß werden, wäre eine Einsparung in der Genauigkeit hier am ehesten zu verkraften. Deshalb soll untersucht werden, wie groß der Fehler ist, wenn mehrere Bits des Grauwertquadrates bei der Aufsummierung eingespart werden.

1. Unterscheidbarkeit der Quadrate benachbarter Grauwerte:
   Nur wenn dicht benachbarte Grauwerte unterschieden werden können, ergibt sich auch bei geringem Kontrast ein verwertbares Ergebnis aus der Quadratsumme:
   Distanz D der Quadrate der direkt benachbarten Grauwerte g und g-1:
   $D = g^2 - (g-1)^2 = g^2 - g^2 + 2g - 1 = 2g - 1$
   Bei einer Einsparung von n Bits des Summanden ist die Distanz benachbarter Grauwertquadrate 2^n. Dann sind alle Grauwerte ab
   $$g = \frac{D+1}{2} = \frac{2^n + 1}{2} \approx 2^{n-1}$$
   unterscheidbar.
   Beispielsweise sind bei einer Reduktion der 20 Bit breiten Summanden um vier Bit die Quadrate aller Grauwerte oberhalb von acht unterschiedlich.

2. Relativer Fehler:
   Bei Abschneiden der niederwertigen n Bit der Summanden und Ergänzung der Summe um entsprechend viele 0-Bits ist der absolute Fehler je Wert maximal 2^n-1. Bei vorangehender

Rundung der Quadrate anhand der abzuschneidenden Stellen ist der maximale Absolutbetrag des Fehlers 2^{n-1}. Der relative Fehler hängt vom Grauwert ab.
Der maximale relative Fehler R des Quadrates bei Grauwert g und Einsparung von n Bits ist mit Rundung:

$$R = \frac{2^{n-1}}{g^2}.$$

Beipielsweise ist bei einer Reduktion der Summanden um vier Bit und Grauwerten im Bereich von 128 der relative Fehler unterhalb von 0,0005. Ungenauigkeiten von mehr als einem Prozent entstehen erst bei Grauwerten unterhalb von 28.

Die Komplexität eines schnellen Addierers, wie er zur Aufsummierung der unteren 20 Bit der Grauwertquadrate benötigt wird, steigt bei etwa konstanter Durchlaufzeit im Extremfall mit dem Quadrat der Bitanzahl. Deshalb würde eine Reduktion der Genauigkeit um einige Bit hier bereits eine deutliche Flächeneinsparung ermöglichen. Die angenommene Reduktion um vier Bit verringert den Aufwand für die Addierer also um 20 bis 36 Prozent. Wie gezeigt wurde, ist der dadurch entstehende Fehler gering, solange nicht die meisten Grauwerte innerhalb des Fensters sehr niedrig sind.

Einsparungen an der Genauigkeit der Berechnung führen in Extremfällen zu unbrauchbaren Ergebnissen. Eine Anpassung der Rechengenauigkeit an Gegebenheiten wie Grauwertauflösung und Fenstergröße würde einen nicht unerheblichen Aufwand bedeuten und die Programmierung und Spezifikation des Histogrammers verkomplizieren. Die bei geringer Reduktion der Genauigkeit erreichten Einsparungen sind im Verhältnis zur Gesamtgröße des Statistikteils gering. Nach Rücksprache mit dem MAZ ist die Reduktion der Genauigkeit nicht vertretbar.
Deshalb soll auf eine Reduktion der Rechengenauigkeit des Statistikteils verzichtet werden.

4.5 Spezifikation der Schnittstellen des Histogrammers

Aus der jeweiligen Sicht eines Hardwaredesigners oder eines Programmierers wird ein Baustein in erster Linie durch seine Schnittstellen charakterisiert. Die Spezifikation des Bausteins hat deshalb in erster Linie die Aufgabe, diese Schnittstellen zu definieren. In diesem Kapitel soll untersucht werden, wie die Schnittstellen des Histogrammers sinnvoll aussehen können. Dabei soll die Spezifikation des Histogrammers im Detail festgelegt werden.
Meist ist es unumgänglich, daß der Hardwaredesigner auch die Software-Schnittstelle des Bausteins kennt, damit der Chip in der Applikation optimal genutzt werden kann. Für den Programmierer wiederum ist eine gewisse Kenntnis der Umgebung des Chips und seiner Verknüpfung mit dieser wichtig, um das gesamte System verstehen zu können.
Die gewählte Realisierung der Schnittstellen soll daher in den folgenden Ausführungen in einer Form beschrieben werden, die gleichermaßen die Betrachtungsweise von Hard- und Softwaredesigner berücksichtigt.

4.5.1 Interface des Histogrammers zum Auswertungsrechner

4.5.1.1 Spezifikationsvorschlag: Sequentielle Ansteuerung

Vorgeschlagen wurde vom MAZ eine Funktionsauswahl über fünf Steuerleitungen, von denen zwei den Modus festlegen (Initialisierung, Histogrammerstellung, Histogrammdatenübergabe, Statistikdatenübergabe) und die übrigen drei die auszuführende Operation. Die Aktionen sollten dann nach einem festen sequentiellen Schema erfolgen. Dieses Konzept ermöglicht eine sehr kompakte Spezifikation und ist z.B. bei dem kommerziellen Histogrammer HSP48410 von Harris [10] realisiert.

Die in diesem Kapitel vorgestellte Spezifikation folgt jedoch diesem Konzept nicht.
Aus folgenden Gründen wurde von dem Konzept abgewichen:

- Der festgelegte Funktionsablauf läßt dem Programmierer kaum Freiheiten.
- Es ist keine Statusabfrage während eines Funktionsablaufes möglich.
- Operationen können nicht unterbrochen werden, um zwischendurch weitere Werte abzufragen oder zu setzen.
- Die Übergabe der gesamten Daten findet in festgelegter Reihenfolge statt. Auch wenn nur wenige Werte des Histogramms benötigt werden, muß das gesamte Histogramm sequentiell ausgelesen werden.
- Ein zusätzlicher Portbaustein wird zur Ansteuerung des Funktionscodes benötigt.
- Der Anschluß an gängige Bussysteme ist problematisch.
- Einzelne Operationen sind schlecht testbar, da nur wenige Eingriffsmöglicheiten bestehen.

4.5.1.2 Realisierung eines registerorientierten Hostinterfaces

Um den Histogrammer in ähnlicher Weise programmierbar zu machen, wie viele gängige Computer-Peripherieschaltungen, wurde deren Ansteuerung und Programmierung bei der Definition des Interfaces zugrunde gelegt.
Als Vorbild für registerorientiert steuerbare Bausteine dienen Peripheriebausteine, die eine ähnliche Komplexität aufweisen wie der Histogrammer. Beispiele dafür sind der Rockwell 6522 (Seriell- und Parallelinterface IC), National Semiconductor 16550 (Serieller Schnittstellenbaustein mit FIFO) oder der Motorola 68901 (Seriell- und Parallelinterface und Timer). Weitere Bausteine mit ähnlichem Aufgabengebiet wie der Histogrammer sind z.B. der

Histogramm-/ Hough-Transformations Prozessor L64250 von LSI Logic [21] und Farbpaletten DAC's, wie sie auf PC-Grafikkarten eingesetzt werden.
Diese Bausteine werden über eine Vielzahl interner Register gesteuert. Sie verfügen über ein einfaches Businterface, das neben Adreß- und Datenleitungen zwei oder drei Steuerleitungen aufweist: Schreib- (und) Leseleitung (/RD, /WR oder R//W) und die Auswahlleitung Chipselect (/CS). Damit ist der Anschluß an praktisch jedes Standard-Bussystem möglich. Die Bausteine können von der CPU wie ein Speicher beschrieben und ausgelesen werden.

Bei der Definition des registerorientierten Hostinterfaces wurde einfache Programmierbarkeit des Histogrammers bei hoher Flexibilität angestrebt.

Der Histogrammer verfügt über eine große Anzahl von Funktionen (s. Kapitel 3). Sämtliche Funktionen werden über Register konfiguriert und aktiviert. Es gibt statisch einstellbare Parameter wie z.B. die Bildgröße und dynamische Funktionen, die eine einmalige Aktivität auslösen, wie beispielsweise den Start einer Histogrammerstellung. Während Parameter wie Grauwerte oder Bildkoordinaten jeweils zehn oder zwölf Bit umfassen, werden für Steueraufgaben meist nur wenige Bits benötigt. Um die Registeranzahl nicht zu groß werden zu lassen, können Steuerbits für mehrere Funktionen in jeweils einem Register zusammengefaßt werden. Nach Möglichkeit sollten nur eng zusammengehörige Steuerbits in einem Register untergebracht werden oder aber Steuerbits für Funktionen, die einander ausschließen. Dies ist sinnvoll, da einerseits eng zusammengehörige Bits meist gemeinsam umprogrammiert werden müssen und es andererseits bei einander ausschließenden Funktionen nicht notwendig ist, die Einstellungen der jeweils nicht genutzten Funktion zu sichern. Weiterhin können durch Gruppencodierung unsinnige Kombinationen ausgeschlossen werden, wie beispielsweise Histogramm- und Kachelmodus. Bei entsprechender Anordnung der Bits ist es nicht notwendig, vor dem Schreiben eines Registers den vorherigen Inhalt auszulesen und mit den neuen Werten zu verknüpfen. Damit müssen die Register nicht auslesbar sein. Der zusätzliche Hardwareaufwand für rücklesbare Register kann eingespart werden, und es müssen auch keine Schattenregister im Programm angelegt werden.

Das Auslesen der Histogrammdaten muß aufgrund der großen Datenmenge von bis zu sechs Kilobyte (1K*48 Bit) pro Bild und der relativ kurzen Zeit, die zwischen zwei Bildübertragungen zur Verfügung steht, schnell möglich sein, d.h. mit minimalem Handshake.
Bei digitalen Kameras für Bildverarbeitung kann die Wartezeit zwischen der Aufnahme von Bildern meist vom Anwender gesteuert werden. Bei CCIR-kompatiblen Kameras (Fernsehsystem [19]) entspricht die Pause zwischen zwei Bildern einer Anzahl von 11 Zeilen, die mit einer Zeilenfrequenz von 15,625 KHz dargestellt werden, also ca. 0,7 ms. Die Übertragungsgeschwindigkeit für die Histogrammdaten muß bis zu 8 MB/s betragen, um in dieser Zeit den gesamten Histogrammspeicher auslesen zu können.
Alle oben genannten Bausteine werden über einen 8 Bit breiten Datenbus (9 Bit beim L64250) programmiert und ausgelesen. Um über einen 8 Bit breiten Datenbus eine Transferrate von 8 MB/s zu erreichen, ist jedoch eine Geschwindigkeit von 8 Millionen Zugriffen pro Sekunde notwendig. Diese Geschwindigkeit wird von vielen Bussystemen nicht erreicht. Insbesondere wird 8 Bit breite Peripherie oft mit nur ein bis zwei Millionen Zugriffen pro Sekunde bedient. Deshalb sollte der Datenbus des Histogrammers breiter ausgelegt werden. Ursprünglich wurde vom MAZ ein 32 oder 64 Bit breiter Bus favorisiert. Der Aufwand für einen derart breiten Busanschluß ist jedoch nicht notwendig. Da die vom Histogrammer zu verarbeitenden Datentypen Pixel mit einer Grauwertauflösung von 10 Bit und Koordinaten mit einer Auflösung von 12 Bit sind, sowie 8 bis 24 Bit breite Histogrammtabellen, wurde ein 16 Bit breiter Datenbus vorgesehen. Da praktisch alle für Bildverarbeitung genügend leistungsfähigen Prozessorsysteme über 16- oder 32-Bit Datenbusse verfügen, ist eine Breite von 16 Bit ein guter Kompromiß zwischen hoher Leistungsfähigkeit und geringer Anschlußanzahl.

Alternativ zu einer hohen Datenübertragungsgeschwindigkeit und einer ausreichend großen Breite der Schnittstelle denkbar wäre auch eine (optionelle) Reduktion der Datenmenge vor der Übergabe an den Hostrechner. Für die Histogrammauswertung ist in den meisten Fällen auch ein Histogramm mit deutlich reduzierter Genauigkeit der Werte ausreichend. Beispielsweise könnte ein bis zu 32 Bit breites Histogrammdatum zur Übergabe auf einen 8 Bit-Wert reduziert werden, indem nicht der exakte Wert, sondern sein Logarithmus Dualis (codiert mit fünf Bit) und die führenden drei Bits übergeben werden. Der dabei auftretende Fehler beträgt maximal 12.5 Prozent des Wertes. Ein so codiertes Histogramm könnte auch in einen ein Kilobyte großen Zwischenspeicher innerhalb des Histogrammers übertragen werden und vom Host zu einem beliebigen Zeitpunkt ausgelesen werden.

4.5.2 Registerbelegung und Programmierung

Der Histogrammer wird im Gegensatz zu vielen anderen Bausteinen innerhalb eines Rechnersystems zu jedem Zeitpunkt von nur einem speziellen Programm angesteuert werden. Da der Programmierer also jederzeit alle Inhalte der Schreib-Register kennt, bzw. eine Kopie von Registern im Speicher vorsehen kann, ist es nicht notwendig, die vom Programm zu setzenden Register auch auslesen zu können. Die meisten Register des Histogrammchips sind deshalb nicht rücklesbar ausgelegt, um eine Vielzahl von Tristate-Puffern einzusparen. Folglich kann Registern bei Schreib- und Lesezugriffen eine unterschiedliche Bedeutung zugewiesen werden. Bei den Steuerregistern, die den Operationsmodus des Chips festlegen, ist jedoch eine eventuelle Implementation von Rücklesbarkeit vorgesehen, falls diese später gewünscht wird.
Der Histogrammer soll das Big-Endian-Format für Daten verwenden. Dieses Format wird von den meisten (neueren) Prozessoren verwendet bzw. unterstützt. Die niedrigstwertigen Bits eines Wortes finden sich dabei jeweils in den untersten Bitpositionen. Die niedrigstwertigen Teile von breiteren Werten als die Busbreite finden sich analog dazu in den Speicherstellen mit der niedrigeren Adresse. Soll der Chip im Little-Endian-Format betrieben werden (z.B. bei der Motorola 68000-Serie verwendet), sind die Datenleitungen 0 bis 7 und 8 bis 15 zu vertauschen und die Datenübertragungen auf 16 Bit Zugriffe zu beschränken.

Um die Bezugnahme auf die Register und Steuerbits des Histogrammers in einem Programm zu vereinheitlichen, werden bereits englische Kurznamen für Register und Steuerbits vergeben.

In einer ersten Version der Registerstruktur wurde eine direkte Zugreifbarkeit aller Steuerregister („Funktionsregister“) und Ergebnisregister des Statistikteils („Statistikregister“) über den Adreß- und Datenbus implementiert. Der Histogrammspeicherinhalt konnte indirekt über einen Zähler adressiert über vier Speicherstellen innerhalb des Registeradreßraumes ausgelesen und beschrieben werden, wobei die Fenster je nach Konfiguration unterschiedlich separat auf diese Speicherstellen abgebildet wurden. Dabei hat sich eine Anzahl von insgesamt weniger als 64 beschreibbaren und 64 lesbaren Registern ergeben.

Auf Wunsch des MAZ wird jedoch gemäß einer später erschienenen Erweiterung des Konzeptes [25] eine indirekte Adressierung sämtlicher Register vorgenommen. Mit der indirekten Adressierung soll eine Verringerung der Anzahl der Adreßleitungen auf drei und Ladbarkeit aller Funktionsregister bzw. Lesbarkeit aller Statistikregister über eine einzige Adresse bewirkt werden. Dabei wurde eine Aufteilung der Register in drei Klassen vorgenommen:

1. **Funktionsregister**: Sämtliche Register zur Steuerung der internen Einheiten
2. **Histogrammspeicher**
3. **Statistikregister**: Ergebniswerte des Statistikteils

In einer übergeordneten Registerebene („**Oberste Registerebene**“) steht für jede dieser Registerklassen ein eigener Adreßzähler zur Verfügung, der über jeweils eine vom Host direkt adressierbare Speicherstelle gesetzt werden kann und bei Datenzugriffen automatisch inkrementiert wird. So kann die CPU weiterhin nacheinander sämtliche Register setzen bzw. lesen, ohne für jeden Wert eine neue Adresse in das zusätzliche Adreßregister schreiben zu müssen.
Auch bei späteren Erweiterungen des Funktionsumfanges und damit der Registeranzahl wird keine weitere Adreßleitung benötigt. Abbildung 4.11 zeigt die Organsisation der Registerebenen.

Das automatische Adreßinkrement ermöglicht die Datenübertragung im sogenannten Burst-Modus, wie er von neueren Mikroprozessoren und Bussystemen unterstützt wird. Dabei wird nur eine Startadresse an den Peripheriebaustein geliefert und mit jedem folgenden Zugriff auf sequentielle Adressen zugegriffen. Diese Form der Datenübertragung wird insbesondere auf gemultiplexten Adreß-/ Datenbussen (z.B. beim PCI-Bus [26]) genutzt, da sie hier eine Geschwindigkeitssteigerung um den Faktor zwei oder mehr ermöglicht.
Die von neueren Bussystemen unterstützten Burst-Zugriffe können vom Histogrammer jedoch nicht sinnvoll bedient werden, da sie einerseits meist 32 Bit breite Werte erfordern, andererseits nur für Zugriffsfrequenzen oberhalb von 16 Millionen Zugriffen pro Sekunde einen Vorteil bringen (z.B. beim PCI-Bus). Die nicht auf ein bestimmtes Bussystem spezialisierte 16 Bit-Schnittstelle des Histogrammers unterstützt keinen derart hohen Datendurchsatz.

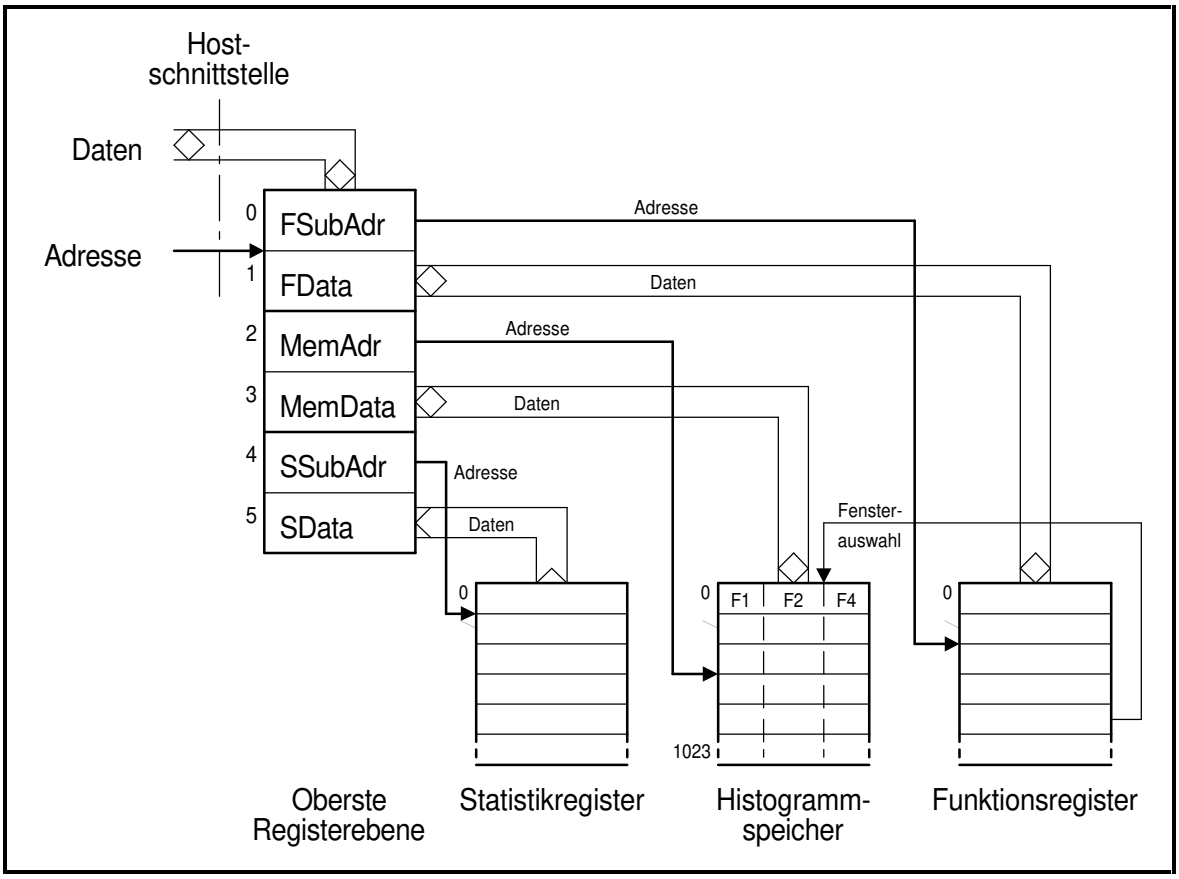

Abbildung 4.11: Registerebenen

4.5.2.1 Oberste Registerebene

In der obersten Registerebene (gemäß [25]) finden sich die Adreßregister für die einzelnen Registerklassen und jeweils eine Speicherstelle für die Datenübertragung an die zugeordneten Register. Folgende Struktur von sechs Registern wird für den Host sichtbar (s. Tabelle 4.2):

Tabelle 4.2: Oberste Registerebene

RegAdr	R/W	Bedeutung
0	W	FSubAdr - 5 Bit Subadresse für Funktionsregister * Autoinkrement bei Schreibzugriff auf FData
1	R/W	FData - Schreib-/Lesedaten für Funktionsregister
2	W	MemAdr - 10 Bit Adresse für Histogrammspeicher * Autoinkrement nach Auslesen / Schreiben aller zum selektierten Fenster gehörigen Datenworte * Beschreiben dieses Registers löscht den Zugriffszähler für MData
3	R/W	MData - Schreib-/Lesedaten für Histogrammspeicher * Über einen Zugriffszähler werden nacheinander alle zu einer Speicherstelle eines Fensters gehörigen Teile adressiert (je nach Fensterbreite 1-3 Datenworte)
4	W	SSubAdr - 6 Bit Subadresse für Statistikregister * Autoinkrement bei Lesezugriff auf SData
5	R	SData - Lesedaten aus Statistikregistern

Bei dieser Registerorganisation wird also eine zusätzliche Ebene zwischen dem Hostrechner und den Funktionsregistern eingefügt. Bei wahlfreien Zugriffen auf die Register wird vor jedem Zugriff ein zusätzlicher Adreß-Schreibzugriff erforderlich. Dies ist insbesondere bei Read-Modify-Write-Zugriffen auf den Histogrammspeicher aufwendig, da dann jeweils über ein Funktionsregister der zugreifbare Teil des Histogrammspeichers vorgewählt werden muß und jeweils alle Bits eines Fensters gelesen oder geschrieben werden müssen (insbesondere bei Fenstern mit mehr als 16 Bit Breite), bevor das Inkrement zu nächsten Adresse stattfindet. Soll der Speicherinhalt durch den Host modifiziert werden, empfiehlt es sich deshalb, vor den Schreibzugriffen zuerst alle Lesezugriffe durchzuführen und die Daten im Host zwischenzuspeichern. Um das Polling von Statusbits und Read-Modify-Write-Zugriffe auf Funktionsregister zu erleichtern, findet beim Lesezugriff auf Funktionsregister kein Autoinkrement des Adreßzählers (FSubAdr) statt.
32 Bit-Zugriffe des Prozessors auf 16 Bit breite Peripherie werden von den meisten Bussystemen automatisch auf zwei 16 Bit Zugriffe umgesetzt. Ein 32 Bit-Schreibbefehl kann dann benutzt werden, um mit den unteren 16 Bit des 32 Bit-Wortes die Adresse des zu beschreibenden Registers zu setzen und mit den oberen 16 Bit das zugehörige Datenregister zu beschreiben. Zu beachten ist, daß die gewählte zweistufige Registerorganisation einen Lesezugriff auf den Histogrammer über 32 Bit-Prozessorbefehle nicht ermöglicht. Eine spätere softwarekompatible Erweiterung des Chips auf einen 32 Bit breiten Bus für Zugriffe auf den Histogrammspeicher ist deshalb bei der gewählten Organisation nicht mehr möglich. Diese Problematik ist von VGA-Grafik-Karten (Standard bei IBM Personal-Computern) her bekannt. Da jedoch die Softwarebasis für den Histogrammer klein und damit flexibler sein wird, sollte vollständige Software-Kompatibilität späterer erweiterter Versionen des Histogrammers kein wesentliches Argument sein.

4.5.2.2 Funktionsregister

Im Folgenden findet sich eine Auflistung aller Funktionsregister mit ihren Subadressen und eine Beschreibung ihrer Funktion. Alle gemäß Kapitel 3 einstellbaren Operationen des Histogrammers können über diese Register konfiguriert werden. Die Anordnung der Register wurde frei gewählt.

Tabelle 4.3: Register für Fensterkoordinaten und Bildauflösung

FSubAdr	R/W	Bedeutung
0,1,2,3	W	Win1 - 12 Bit Position Fenster 1: X1,Y1,X2,Y2
4,5,6,7	W	Win2 - 12 Bit Position Fenster 2: X1,Y1,X2,Y2
8,9,10,11	W	Win3 - 12 Bit Position Fenster 3: X1,Y1,X2,Y2
12,13,14,15	W	Win4 - 12 Bit Position Fenster 4: X1,Y1,X2,Y2
16	W	Width - 12 Bit Bildbreite-1 (Hauptfenster X)
17	W	Height - 12 Bit Bildhöhe-1 (Hauptfenster Y)
17	R	ActualLine - 12 Bit aktuelle Zeile (Y-Koordinate)

Über sechzehn Funktionsregister werden Bildgröße und Fensterpositionen festgelegt (s. Tabelle 4.3). Die Bildgröße kann in X- und Y-Richtung bis zu 4096 betragen. Um sämtliche Bildgrößen über 12 Bit-Register abzudecken, werden die Pixelkoordinaten intern von Null an bis zu den eingestellten Werten gezählt.
Das das gesamte Bild umfassende virtuelle Fenster soll als Hauptfenster bezeichnet werden.
Die Eckkoordinaten der Fenster werden über obere linke Ecke (X1,Y1) und untere rechte Ecke (X2,Y2) definiert. Es muß immer gelten X1<X2 und Y1<Y2. Die eingetragenen Koordinaten gehören mit zum Fenster.
Optional kann über jeweils einen Eingangspin pro Fenster definiert werden, in welchen Fenstern jedes Pixel liegt. Für externe Steuerung der Fenster muß das entsprechende Bit in WinControl (s. Tabelle 4.4) gesetzt werden. Dann bestimmt der jeweilige Eingangspin, wo ein Fenster aktiv ist. Die eingestellten Fensterkoordinaten sind dann belanglos. Ein Bildende kann jederzeit durch vorzeitige Rücknahme des Bildstart-Signals erzwungen werden. Bei externer Vorgabe des Bildendes muß die intern eingestellte Pixelanzahl (Höhe*Breite) größer oder gleich der maximal gewünschten Pixelanzahl sein, damit nicht bereits vor Rücknahme des Bildstart-Signals intern die Bildende-Bedingung erreicht wird. Die aktuelle Y-Koordinate kann für Testzwecke oder aber um den Fortschritt der Histogrammerstellung abzufragen, ausgelesen werden.

Tabelle 4.4: WinControl - Fensterstatus- und Steuerung

FReg. 18 Bit:	R/W	Bedeutung
0,1,2,3	W	ExternalControl - externe Fenstersteuerung für Fenster 1,2,3,4 aktivieren
4,5,6,7	R	WinStop - Stopbit Fenster 1,2,3,4 (gesetzt: Fensterdaten fertig) Bit 4 ist bei fester Fenstergröße für Fenster 1 immer gelöscht (0)
4,5,6,7	W	ClrWinStop - Setzen um entsprechendes Stopbit rückzusetzen Über Setzen von Bit 8 (W) können die Stopbits auch gesetzt werden
8	R	ActiveState - Gesetzt: Hauptfenster ist aktiv
8	W	SetWinStop - Setzen um Fensterstopbits 1-4 zu setzen statt zu löschen
9	R	Stop - Gesamt-Stopbit (gesetzt: Bildende, Histogramme lesbar)
9	W	Start - Setzen um Gesamt-Stopbit rückzusetzen

Da auf den Histogrammspeicher während der Bilddatenübertragung an den Histogrammer für jedes Pixel ein Zugriff durchgeführt werden muß, kann während dieser Zeit nicht vom Hostrechner her auf den Histogrammspeicher zugegriffen werden. Der Histogrammspeicher kann nur zwischen der Übertragung zweier Bilder ausgelesen werden. Es soll jedoch sichergestellt werden, daß der Hostrechner die Histogramme vollständig auslesen kann, auch wenn er die Pause zwischen zwei Bildübertragungen verpaßt oder aber wenn die Übertragung der Histogrammdaten an den Hostrechner nicht bis zum nächsten Bildstart beendet wird.
Um den ungestörten Zugriff des Hostrechners und die Übertragung konsistenter Daten sicherzustellen sind Stopbits eingeführt worden, über die gesteuert wird, wann der Histogrammer ein Bild auswertet und wann der Hostrechner auf Ergebnisse zugreifen kann (s. Tabelle 4.4).

Das Stopbit für das Hauptfenster (Gesamt-Stopbit) wird automatisch gesetzt, sobald eine Bildübertragung beendet wurde. Bei gesetztem Gesamt-Stopbit kann der Hostrechner auf den Histogrammspeicher zugreifen. Nach Beendigung des Zugriffs muß er das Stopbit löschen, damit der Histogrammaufbau beim nächsten Bildstart wieder beginnt.

Zusätzliche Stopbits für die einzelnen Fenster ermöglichen es, festzustellen, wann ein Fenster komplett an den Histogrammer übertragen wurde. Das Stopbit für ein Fenster wird gesetzt, nachdem das letzte Pixel des Fensters ausgewertet wurde. Voraussetzung für die Aufzeichnung von Daten für ein Fenster ist, daß das Stopbit rechtzeitig vor Übertragung des ersten Pixels gelöscht wurde. Nur dann, wenn ein Fenster nach Löschung des Stopbits komplett erkannt wird, findet eine Berechnung der Daten dieses Fensters statt. So können beispielsweise die Statistikdaten eines Fensters bereits ausgelesen werden, wenn das Stopbit anzeigt, daß das Bild bis zur letzten Zeile des Fensters übertragen wurde. Anschließend kann noch innerhalb der laufenden Bildübertragung dieses Fenster neu programmiert werden, um einen weiter unten liegenden Bildausschnitt zu bearbeiten. Diese Möglichkeit kann allerdings nur für die Statistikwerte genutzt werden, da keine Möglichkeit besteht, während der Bildübertragung auf den Histogrammspeicher zuzugreifen.
Der Hostrechner kann die Daten des zu einem Fenster gehörigen Teils des Histogrammspeichers z.B. zur späteren Nutzung als Egalisierungstabelle schützen, indem das Stopbit für das gewünschte Fenster nicht wieder gelöscht wird. Dann findet keine Modifikation des Speicherinhaltes bei Histogrammerstellung statt.
Bei externer Fenstersteuerung haben die Stopbits während der Bildübertragung keine Funktion, da der Chip nicht entscheiden kann, wann ein Fenster fertig ist. Da sie jedoch auch über Löschung der Fenster entscheiden (s.u.), können die Stopbits auch vom Host gesetzt werden.
Um einen uneingeschränkten Zugriff auf den Histogrammer noch vor Übertragung des ersten Bildes zu gewährleisten, sind nach einem Reset alle Stopbits gesetzt.

Tabelle 4.5: ModeControl - Histogramm- und Auslese-Modus

FReg. 19 Bit:	R/W	Bedeutung
0,1	W	HistMode - 00: Histogramme aufnehmen 01: über (max.) 1024 gleichgrosse Kacheln summieren 10: Histogrammspeicher als Look-Up-Tabelle
5	W	RAMClearStart - Setzen um Löschung des Speichers (ab MAdr) zu starten (alle Fenster mit gelöschtem Stopbit werden gelöscht)
5	R	RAMClearState - Gesetzt: RAM-Löschung läuft (währenddessen ist kein Speicherzugriff möglich)
6	W	AccumulationStart - Setzen um Aufsummierung der Histogrammwerte (ab MAdr) zu starten (Aufsummierung in Fenster 1 (bis zu 24 Bit) und bei 12,12,12,12 Bit auch Fenster 2)
6	R	AccumulationState - Gesetzt: Erstellung der Egalisierungstabelle läuft (währenddessen ist kein Speicherzugriff möglich)
7	W	NoReadClear - 0=Histogrammspeicher wird beim Auslesen (Lesezugriff) gelöscht (alle Fenster mit gelöschtem Stopbit werden gelöscht) 1=Lesezugriff verändert Histogrammspeicher nicht
8,9	W	WinSelect - Auswahl des über MData zugreifbaren Fensters 00: Fenster 1, 01: Fenster 2, 10: Fenster 3, 11: Fenster 4

Die Steuerbits „HistMode“ (Tabelle 4.5, FReg.19, Bits 0,1) entscheiden über die Funktion des Chips und sollten nur bei gesetztem Gesamt-Stopbit beschrieben werden.
Wird der Histogrammspeicher als Look-Up-Tabelle verwendet, dann erscheint am Pixelausgang der über den Ausgangsmultiplexer ausgewählte Bereich der durch den Pixeleingang adressierten Speicherstelle. Dieser Modus wird genau wie der Histogrammodus bedient, mit dem Unterschied, daß intern keine Schreibzugriffe auf den Speicher stattfinden.
Auch der Kachelmodus unterscheidet sich nur durch die bei der Bildübertragung stattfindende Berechnung vom Histogrammodus. Weitere Erläuterungen zum Kachelmodus folgen weiter unten.

Beim Speicherzugriff des Hostrechners kann wahlweise jede Speicheradresse über ein Register (MAdr, s. Tabelle 4.2) vorgegeben werden, oder es wird eine Startadresse vorgegeben, und die Speicheradresse wird nach jedem Speicherzugriff erhöht. Diese Organisation wurde in erster Linie gewählt, um nicht einen Adreßbereich von über 6 Kilobyte im Hostrechner belegen zu müssen.
Fensterweises Auslesen nimmt dem Host die Aufgabe ab, die ausgelesenen Daten über Schiebebefehle und Maskierungen fensterweise aufzuteilen. Über WinSelect (FReg.19, Bits 8,9) wird ausgewählt, welches Fenster gelesen werden soll. Je nach Breite des Fensters (s. Tabelle 4.7) findet ein Inkrement der Histogrammspeicheradresse bei jedem, jedem zweiten oder nach drei Zugriffen statt. Es wird nacheinander wortweise zuerst auf die niederwertigsten Bits einer Speicherstelle des Fensters zugegriffen (L=Low), dann ggf. auf die verbleibenden höherwertigen Bits (M=Mid, H=High) usw. (s. Tabelle 4.9). Die ungenutzen Bits werden als Nullen zurückgelesen. So kann der Host die gelesenen Worte direkt nacheinander in den Speicher kopieren und erhält damit eine komplette Tabelle aller Werte des Fensters. Es ist auch möglich, jeweils nur die oberen Bitstellen eines Fensters mit einer Breite von über 16 Bit auszulesen, indem die um 1 inkrementierte Fensternummer (in WinSelect) angewählt wird.

Zum Speichertest eignet sich der Modus („kc000“), da in dem Modus Fenster 1 die gesamte Speicherbreite enthält und der Speicher so direkt als 1024*3 16 Bit Worte ausgelesen oder beschrieben werden kann.
Die Löschung des Speichers kann entweder direkt beim Auslesen der Histogramme geschehen oder als interne Operation separat gestartet werden. Die Stopbits der einzelnen Fenster entscheiden über die Löschung des Speichers, so daß Daten innerhalb der Fenster erhalten bleiben, wenn die Histogramme nicht neu aufgebaut werden sollen.
Die Startadresse für die Operationen RAM löschen und akkumulieren (Bildung der Summenhäufigkeitsverteilung) wird ebenfalls über das Register MAdr gesetzt. Die gewählte Operation läuft dann von der Startadresse bis zum Speicherende.

Die Organisation des Speicherzugriffs mit Autoinkrement der Adresse erlaubt die Datenübertragung jeweils eines Fensters über einen DMA-Kanal.

Tabelle 4.6: MemoryConfiguration - Kachelgröße und Speicheraufteilung

FReg. 20 Bit:	R/W	Bedeutung
0,1,2	W	TileWidth - Selektor für Kachelgröße: Seitenlängen=2^Selektor erlaubte Werte: 0 (für Test!), 3, 4, 5, 6, 7
8,9,10	W	RamConf - Speicherkonfiguration: Aufteilung des 48-Bit Speichers auf die Fenster, s. Zusatztabelle
11	W	Win1Var - 0=Fenster 1 ist fest auf Bildgröße und wird über das Gesamt-Stopbit gesteuert 1=Eckpunkte von Fenster 1 sind variabel

Die acht im Histogrammodus möglichen Speicherkonfigurationen (s. Tabelle 4.7) wurden zur einfacheren Benennung mit Kurznamen der Form k1234 versehen. Die einzelnen Dezimalen (hexadezimal) geben die Anzahl der Nibbles (1 Nibble=4 Bit) pro Fenster an. Im Beispiel hätte also Fenster 1 vier Bit, Fenster 2 acht Bits usw..
Im Kachelmodus kann der Benutzer das gesamte Bild (also das Hauptfenster) in quadratische Kacheln der Größe 8*8 bis 128*128 einteilen. Bedingt durch die Speichergröße sind maximal 1024 Kacheln möglich. Dabei werden die unteren 24 Bit jeder Speicherstelle des Histogrammspeichers zur Summierung aller Helligkeitswerte in der jeweiligen Kachel verwendet. Auch bei Einstellung des Kachelmodus entscheidet weiterhin die eingestellte Speicherkonfiguration über die Aufteilung des Speichers. Da für den Kachelmodus die unteren 24 Bit des Speichers genutzt werden, sollte eine Konfiguration eingestellt werden, in der Fenster 1 24 Bit breit ist. Die Kachelsummen können dann genau wie Histogramme in Fenster 1 gelesen und gelöscht werden.

Tabelle 4.7: Werte für RamConf

RamConf	Fenster 1	Fenster 2	Fenster 3	Fenster 4
000: k6033	24	-	12	12
001: k6024	24	-	8	16
010: k5043	20	-	16	12
011: k4044	16	-	16	16
100: k3333	12	12	12	12
101: k6060	24	-	24	-
110: k7050	28	-	20	-
111: kc000	48	-	-	-

Der Histogrammer unterstützt die acht gezeigten Speicheraufteilungen (Tabelle 4.7). Intern wird der Speicher vom LSB zum MSB in aufsteigender Reihenfolge mit den Fenstern eins bis vier jeweils vom LSB bis zum MSB belegt. Bei der Aufteilung des Histogrammspeichers auf die Fenster wurde berücksichtigt, daß das Fenster mit der größten Anzahl von Bits je Speicherwort jeweils bei Bit 0 des Hauptspeichers beginnt (vgl. Kapitel 3.2.2). Weiterhin wurden die Fenster so auf das Speicherwort verteilt, daß möglichst wenige Multiplexer notwendig sind, um die einzelnen Teile des Speicherwortes getrennt nach Fenstern mit dem Datenbus des Histogrammers zu verbinden. Die Kombinationen unterschiedlich breiter Speicherbereiche je Konfiguration entsprechen den in [24] vorgegebenen.
Da die Speicherbelegung bei Nutzung des Speichers als Look-Up-Tabelle für den Programmierer sichtbar wird, soll sie anhand eines Beispiels (Tabelle 4.8) verdeutlicht werden.

Tabelle 4.8: Beispiel für interne Speicheraufteilung

Fenster bei k6033	Fenster 1	Fenster 3	Fenster 4
Bitstellen der Fenster	0 .. 23	0 .. 11	0 .. 11
Bitstellen des Speichers	0 .. 23	24 .. 35	36 .. 47

Die Konfigurationen mit 28 bzw. 48 Bit für Fenster 1 sind nur eingeschränkt sinnvoll, da eine Bildauflösung von über 4096*4096 Pixeln nicht unterstützt wird. Eine höhere Bildauflösung würde breitere Zähler und Komparatoren für die Koordinaten, breitere Addierer für den Kachelmodus und die Erstellung der Egalisierungstabelle, sowie breitere Register für die Statistikwerte erfordern. Beide Modi können jedoch verwendet werden, um mehrere Histogramme aufzusummieren und so ein Histogramm über eine Bildfolge zu erstellen. Im 48 Bit-Modus kann diese Bildfolge bei einer Pixeltaktfrequenz von 25 MHz über 100 Tage lang dauern, ohne daß ein Überlauf eintreten kann.