Tabelle 4.9: Zugriffsfolge bei Auslesen und Beschreiben der Fenster

RamConf/WinSel	00 (=Fenst.1)	01 (=Fenst. 2)	10 (=Fenst. 3)	11 (=Fenst. 4)
k6033	F1L (16) F1H (8)	F1H (8)	F3 (12)	F4 (12)
k6024	F1L (16) F1H (8)	F1H (8)	F3 (8)	F4 (16)
k5043	F1L (16) F1H (4)	F1H (4)	F3 (16)	F4 (12)
k4044	F1 (16)	-	F3 (16)	F4 (16)
k3333	F1 (12)	F2 (12)	F3 (12)	F4 (12)
k6060	F1L (16) F1H (8)	F1H (8)	F3L (16) F3H (8)	F3H (8)
k7050	F1L (16) F1H (12)	F1H (12)	F3L (16) F3H (4)	F3H (4)
kc000	F1L (16) F1M (16) F1H (16)	F1M (16) F1H (16)	F1H (16)	-

Die in Tabelle 4.9 gezeigten Zugriffsfolgen ergeben sich aus der Möglichkeit, über jeweils vier Multiplexer-Stellungen je Konfiguration sämtliche Speicherworte getrennt nach Fenstern über den 16 Bit breiten Datenbus zugreifen zu können. Ein zwei Bit breiter Zugriffszähler kann zyklisch diese vier Stellungen durchlaufen und bei Erreichen des jeweils letzten Teils eines Fensters auf den über WinSelect vorgegebenen Startwert zurückgesetzt werden.

Tabelle 4.10: PixelOutControl: Steuerung des Pixel-Ausganges

FReg. 21 Bit:	R/W	Bedeutung
0..5	W	PixOutPos - Wahl der untersten RAM-Bitstelle an PixelOut Erlaubte Werte: 0 (für Fenster 1 32*32), 2 (für Look Up Table 1 bzw. Fenster 1 64*64), 4,6,8,10,12 (Fenster 1 128*128 .. 2048*2048), 14 (Fenster 1 4096*4096 bzw. Fenster 2 64*64 bzw. LUT 2), 26,38 (LUT 3, LUT 4), 60: win_select 0..3 an Ausgangsbits 0..3, 62: PixelIn durchreichen (synchronisiert mit Clk)

Der Pixelausgang erlaubt es, wahlweise die Eingangspixel durchzureichen oder die internen Fensterkomparator-Signale auszugeben, um anzuzeigen, wann ein Fenster aktiv ist. Im Look-Up-Tabellen-Modus dient er dazu, mithilfe einer von maximal vier vom Host geladenen Tabellen (LUT) eine Umsetzung der Pixeldaten vorzunehmen oder eine durch den Histogrammer aus dem Histogramm von Fenster 1 oder Fenster 2 gebildete Egalisierungstabelle zur Egalisierung zu verwenden. Um den Aufwand für den Ausgangs-Multiplexer nicht zu hoch werden zu lassen, sind nicht alle möglichen Stellungen implementiert (s. Tabelle 4.10). Die Abstufung der Werte erlaubt die Nutzung der Egalisierungstabelle für alle Auflösungen von 32*32 bis 4096*4096 bei 10 oder 8 Bit Graustufenauflösung. (Es müssen dazu jeweils die 10 bzw. 8 MSB's der Egalisierungstabelle an den Ausgang durchgeschaltet werden können.) Über weitere Stellungen des Multiplexers könnten zum Test des Histogrammers beliebige interne Signale nach außen geführt werden.

Tabelle 4.11: Steuerregister für Intervallausgang und Schwellensummen

FSubAdr	R/W	Bedeutung
22	W	IntervMin - 10 Bit Intervallminimum (inklusive) für Intervallausgang
23	W	IntervMax - 10 Bit Intervallmaximum (inklusive) für Intervallausgang
24	W	Thrs1 - 10 Bit Schwellwert für Schwellensumme Fenster 1
25	W	Thrs2 - 10 Bit Schwellwert für Schwellensumme Fenster 2
26	W	Thrs3 - 10 Bit Schwellwert für Schwellensumme Fenster 3
27	W	Thrs4 - 10 Bit Schwellwert für Schwellensumme Fenster 4

Der Intervallausgang (Interval) liefert jederzeit die Information, ob ein eingespeistes Pixeldatum innerhalb des eingestellten Intervalls liegt:
Interval:=[IntervMin ≤ PixelDatum ≤ IntervMax]
Der Ausgang wird synchron zum Pixelausgang und Pixeltaktausgang gesetzt.

Tabelle 4.12: ClrStat - Löschregister für Statistikwerte

FReg. 28 Bit:	R/W	Bedeutung
0..3	W	ClrMinMax - Setzen um Minima und Maxima für Fenster 1..4 zu initialisieren (Minima auf 1023, Maxima auf 0)
4..7	W	ClrSums - Setzen um Helligkeitsummen und Quadratsummen für Fenster 1..4 zu löschen
8..11	W	ClrThrsSums - Setzen um Schwellensummen für Fenster 1..4 zu löschen

Über das Löschregister (FReg. 28, s. Tabelle 4.12) ist ein selektives Löschen der Statistikwerte für die einzelnen Fenster möglich. So kann beispielsweise über mehrere Bilder in einem Fenster das Minimum und Maximum gesucht werden, wobei die Helligkeitsummen für jedes Bild neu gebildet werden. Durch Beschreiben des Registers mit 0FFFh werden alle Werte gelöscht.

4.5.2.3 Statistikregister

Tabelle 4.13 zeigt die Adressen der Statistikregister. Sämtliche Statistikregister können nur gelesen werden.

Tabelle 4.13: Statistikregister

SSubAdr	R/W	Bedeutung
0,2,4,6	R	MinVal - 10 Bit Minimum Fenster 1..4
1,3,5,7	R	MaxVal - 10 Bit Maximum Fenster 1..4
8,9,10	R	Sum1 - 16,16,2 Bit Helligkeitssumme Fenster 1 (Lo,Mid,Hi)
11,12,13	R	Sum2 - 16,16,2 Bit Helligkeitssumme Fenster 2 (Lo,Mid,Hi)
14,15,16	R	Sum3 - 16,16,2 Bit Helligkeitssumme Fenster 3 (Lo,Mid,Hi)
17,18,19	R	Sum4 - 16,16,2 Bit Helligkeitssumme Fenster 4 (Lo,Mid,Hi)
20,21,22	R	SqrSum1 - 16,16,12 Bit Quadratsumme Fenster 1 (Lo,Mid,Hi)
23,24,25	R	SqrSum2 - 16,16,12 Bit Quadratsumme Fenster 2 (Lo,Mid,Hi)
26,27,28	R	SqrSum3 - 16,16,12 Bit Quadratsumme Fenster 3 (Lo,Mid,Hi)
29,30,31	R	SqrSum4 - 16,16,12 Bit Quadratsumme Fenster 4 (Lo,Mid,Hi)
32,33	R	ThrsSum1 - 16,8 Bit Schwellensumme Fenster 1 (Lo,Hi)
34,35	R	ThrsSum2 - 16,8 Bit Schwellensumme Fenster 2 (Lo,Hi)
36,37	R	ThrsSum3 - 16,8 Bit Schwellensumme Fenster 3 (Lo,Hi)
38,39	R	ThrsSum4 - 16,8 Bit Schwellensumme Fenster 4 (Lo,Hi)

Die Schwellensumme ist die Anzahl der Pixel, deren Grauwert größer oder gleich dem als Schwelle eingestellten Wert ist.
Die Breiten sämtlicher Register sind so gewählt, daß auch bei maximaler Auflösung und maximaler Fenstergröße kein Überlauf eintreten kann. (Einzige Ausnahme: Die Register für die Schwellensummen können bei voller Bildgröße auf Null überlaufen, wenn alle Pixel oberhalb des Schwellwertes liegen. Dieser Fall kann jedoch durch Betrachtung eines einzigen Pixels von dem Fall unterschieden werden, daß kein Pixel oberhalb der Schwelle lag.)
Die Aufteilung der bis zu 44 Bit breiten Werte auf bis zu drei Register findet analog der Aufteilung beim Speicherzugriff statt. Es werden zuerst die niederwertigen Bits gelesen, dann die verbleibenden höherwertigen Bits usw.. Ungenutzte Bits werden als 0 gelesen.
Sämtliche Werte werden während des Bildaufbaus gebildet, stehen also sofort nach Bildende zur Verfügung und können zu beliebiger Zeit ausgelesen werden. Alle Werte werden nur dann gebildet, wenn das Stopbit für das jeweils zugehörige Fenster gelöscht ist. Summe und Quadratsumme geben die Summe bzw. Quadratsumme aller Grauwerte innerhalb des jeweiligen Fensters an.

In Anhang A findet sich ein Flußdiagramm und ein Programmierbeispiel, um aufzuzeigen, wie der Histogrammer programmiert werden kann.

4.5.3 Realisierung des Businterface

Um den Histogrammer zu unterschiedlichen Bussystemen kompatibel zu machen, soll das Interface so ausgelegt werden, daß der Anschluß an besonders einfache gängige Bussysteme mit nur wenig zusätzlicher Hardware möglich ist.
Bei kleinen Computersystemen wie Einplatinenrechnern wird Peripherie oft direkt an den Prozessorbus angeschlossen. Beispiele für solche Bussysteme sind der Prozessorbus der Motorola 68000-Serie oder der Peripheriebus von 16-Bit Mikrocontrollern wie der Intel MCS-96 Serie.
Bei größeren und erweiterbaren Computersystemen wird ein vom Prozessortyp und vom Prozessorbus unabhängiges Peripherie-Bussystem vorgezogen. Ein solches Bussystem ist z.B. der AT- (ISA-) Bus. Schnellere Bussysteme wie der VME-Bus, EISA oder PCI verfügen meist über ein aufwendigeres Handshake, das von Vielzweck-Peripheriebausteinen nicht unterstützt wird. Hier ist aufgrund der ohnehin höheren Systemkosten der Einsatz von Zusatzhardware zur Anbindung an das Bussystem eher akzeptabel.

4.5.3.1 Charakteristika gängiger Bussysteme

Hier soll zunächst untersucht werden, welche Gemeinsamkeiten gängige 16 Bit-Bussysteme aufweisen. Diese Bussysteme verfügen alle über 16 Datenleitungen und 16 oder mehr Adreßleitungen.

1. AT-Bus
Der AT-Bus [30] verfügt über zwei Paare getrennter Schreib- und Leseleitungen. Ein Paar dient zur Ansteuerung von als Speicher ansprechbaren (memory-mapped) Einheiten und ein Paar zur Ansteuerung von Ein-/Ausgabe- (I/O-) Einheiten. Würde der Histogrammer als Ein-/Ausgabe- Einheit angesprochen, könnten über I/O-Stringbefehle des Prozessors aus einem Register oder in ein Register mit einem Befehl nacheinander mehrere Worte geladen werden. Über die Autoinkrement-Funktion des Histogrammers könnte so einfach ein ganzes Histogramm gelesen werden.
I/O-Schreib- und Leseleitung sind negativ aktiv (active low) und werden erst aktiviert, nachdem die Adresse für mindestens 90 ns ansteht. Bei einem Schreibzugriff sind die Schreibdaten während der gesamten Zeit, die das Schreibsignal aktiv ist, gültig. Bei einem Lesezugriff werden die Lesedaten ca. 30 ns vor Deaktivierung des Lesesignals eingelesen. Schreib- und Lesesignal sind für mindestens 150 ns aktiv.
Memory-Schreib- und Lesezyklen laufen ähnlich ab, allerdings werden die Schreib- und Leseleitungen bereits 30 ns nach Anliegen einer gültigen Adresse aktiviert und gültige Schreibdaten stehen spätestens erst 45 ns nach dem Schreibsignal an. Die Zyklen können auf eine bis zu 90 ns lange Aktivierung von Schreib- und Lesesignal verkürzt werden.

2. 68000-Prozessor-Bus
Der 68000-Prozessor [16] verfügt über eine kombinierte Schreib-/Leseleitung und getrennte Daten-Strobes für die unteren und oberen acht Bit des 16-Bit Busses.
Bei Schreib- und Lesezugriffen sind zuerst die Adreßleitungen und die Schreib-/Leseleitung gültig. Die Daten-Strobes werden bei einem Lesezugriff einen halben Takt nach den Adressen aktiviert (Low) und einen halben Takt nach dem Einlesen der Daten deaktiviert. Bei einem Schreibzugriff werden die Daten-Strobes erst einen halben Takt nach dem Anlegen gültiger Schreibdaten aktiviert. Die Schreibdaten bleiben nach Deaktivierung der Strobes noch für einen halben Takt gültig.
Die Taktfrequenz beträgt dabei maximal 16 MHz, also ist ein halber Takt etwa 30 ns lang. Beim Lesezugriff sind die Datenstrobes 2½, beim Schreibzugriff 1½ Takte lang aktiv.

3. 8096-Mikrocontroller-Bus
Das Timing beim 8096-Mikrocontroller [13], einem Mitglied der MCS-96 Familie, ist ähnlich dem I/O-Timing des AT-Bus. Der 8096 hat allerdings einen gemultiplexten Adreß-/Datenbus, der über externe Adreß-Latches demultiplext werden muß. Beim Schreibzyklus kann per Software ausgewählt werden, ob der Schreibimpuls bereits gleichzeitig mit den Daten kommt, oder erst wenn die Daten bereits gültig sind.

4.5.3.2 Businterface-Timing des Histogrammers

Die untersuchten Businterfaces differieren in einigen Punkten. Beispielsweise unterscheidet sich die Ausprägung der Steuersignale. Geringfügige Verknüpfungen der Steuersignale können jedoch nahezu ohne zusätzlichen Hardwareaufwand vorgenommen werden, da in jedem Fall für den Histogrammer eine Adreßdecodierung innherhalb des Adreßraums des Hostrechners erfolgen wird. Hierfür wird oft ein PAL oder GAL (EEPLD) verwendet, das aus den Adreß- und Steuerleitungen ein Chip-Select-Signal generiert. Dieser Baustein kann auch verwendet werden, um einheitlich ein getrenntes Schreib- und Lesesignal zu generieren.
Nach Anpassung der Steuersignale bleibt als ein wesentlicher Unterschied zwischen den Bussystemen der zeitliche Bezug zwischen dem Anliegen gültiger Schreibdaten und dem Schreibsignal. Bei allen untersuchten Systemen sind die Schreibdaten während der den Schreibzyklus beendenden steigenden Flanke des Schreibsignals gültig.
Alle Bussysteme sind so konzipiert, daß relativ einfach Standard-Speicherbausteine angeschlossen werden können. Auch die meisten Speicherbausteine übernehmen Daten während der gesamten Zeit, für die das Schreibsignal aktiv ist (vgl. [22]) und speichern den Wert, der zum Zeitpunkt der steigenden Flanke des Schreibsignals anlag. Deshalb wird auch für den Histogrammer diese Konvention getroffen.

Anhand dieser Überlegungen können geeignete Buszyklen definiert werden. Zu beachten ist jedoch, daß der Histogrammer über einen eigenen Takt verfügt, über den die internen Abläufe gesteuert werden. Schreib- und Lesezugriffe auf Register müssen synchron zu diesem Takt erfolgen. Da auch der Hostrechner über seinen eigenen Takt verfügt, muß eine Synchronisation des Zugriffs mit dem Takt des Histogrammers innerhalb des Histogrammers stattfinden. So ist beispielsweise für Lesezugriffe auf Register wie den Bildzähler ein Latch einzufügen, da sich der Zähler während des Zugriffes verändern könnte. Das Register muß während des Lesezugriffes taktsynchron geladen werden. Eine Veränderung des Lesewertes zu einem ungünstigen Zeitpunkt könnte im Host inkonsistente Zustände oder Metastabilität hervorrufen.
Da während oder nach einem Zugriff noch Zeit benötigt wird, um Operationen wie das Löschen des Histogrammspeichers und die Erhöhung des Histogrammspeicher-Adreßzählers vorzunehmen, ergibt sich für einen Zugriff eine Dauer von mehreren Takten.
Um die dann noch mögliche Busbandbreite auch mit schnelleren Bussystemen optimal nutzen zu können, wird auf Wunsch des MAZ zusätzlich zu den Steuersignalen ein Data Acknowledge-Signal (/DTACK) eingeführt. Das DTACK-Signal zeigt mit negativem Pegel an, daß ein Schreib- oder Lesezugriff beendet werden darf. Bei Erkennung des Endes eines Zugriffs nimmt der Histogrammer das DTACK-Signal innerhalb eines Taktes wieder zurück.
Diese vom 68000-Bussystem übernommene Konvention ist zwar nicht mit allen Bussystemen direkt kompatibel, ermöglicht aber unter Verwendung externer Zusatzlogik eine optimale Anpassung.
Ein mit den untersuchten Bussystemen kompatibles Timing zeigen Abbildung 4.12 und Abbildung 4.13. Die Werte in den Tabellen (Tabelle 4.14, Tabelle 4.15) sind durch die Implementation einzuhaltende Mindestwerte. Die in Nanosekunden (ns) angegebenen Werte sind dabei großzügig gewählte Schätzwerte, deren exakter Wert durch die Verzögerung von Signalen auf dem Chip bestimmt werden wird. Das Zustandekommen der in Taktzyklen

angegebenen Werte wird in Kapitel 4.5.3.3 beschrieben. Bei nicht aufgeführten Werten gibt es keine Limitierungen. Der dargestellte Lesezugriff zeigt, wie das taktsynchron definierte DTACK-Signal liegt. Beim Schreibzugriff gilt das gleiche Timing für DTACK.

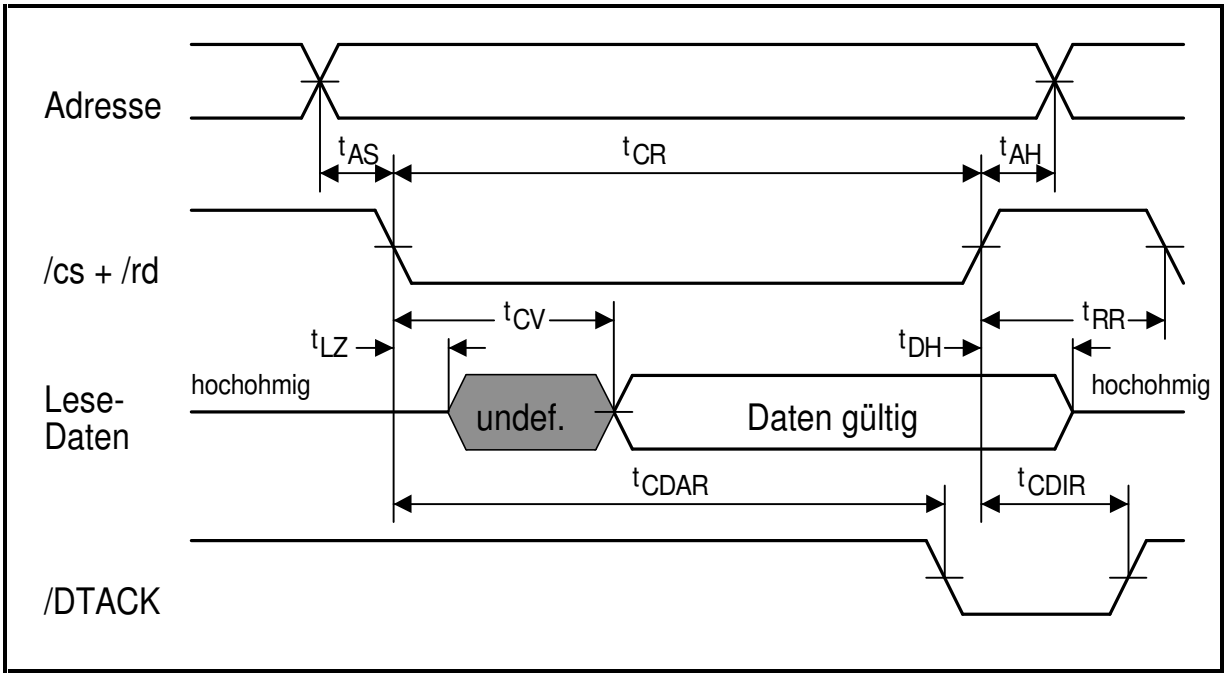

Abbildung 4.12: Lesezugriff auf den Histogrammer

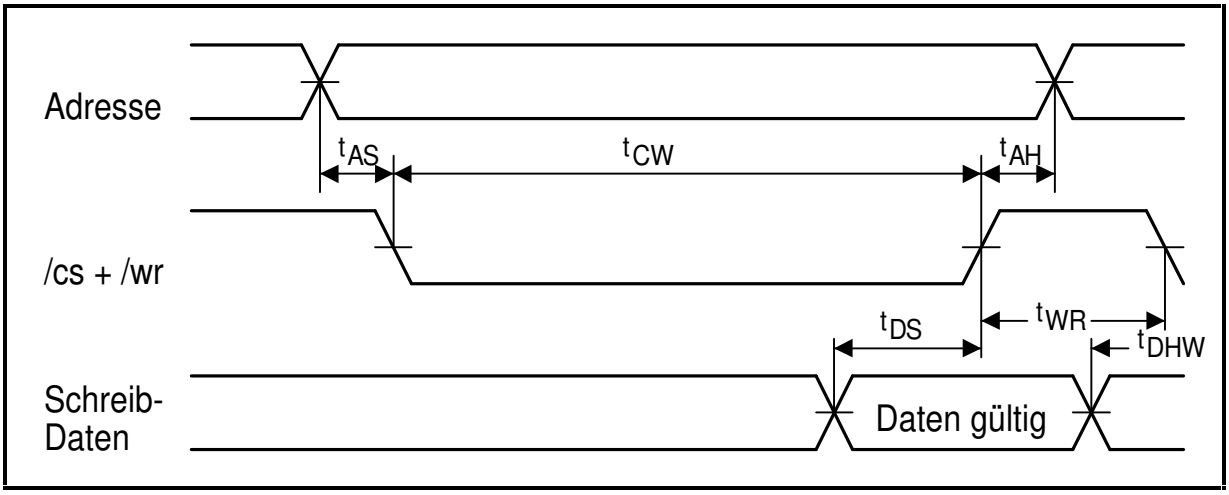

Abbildung 4.13: Schreibzugriff auf den Histogrammer

Tabelle 4.14: Anforderungen des Histogrammers an das Bustiming des Hostrechners

Parameter	Symbol	min.	max.
Chip Select time for READ cycle	t_{CR}	3 t_{CLK}	-
Address setup time	t_{AS}	10 ns	-
Address hold time	t_{AH}	10 ns	-
Chip Select to data valid time	t_{CV}	-	3 t_{CLK}
Chip Select to output in Low-Z	t_{LZ}	0 ns	-
Data hold time	t_{DH}	0 ns	20 ns
Read cycle recovery time	t_{RR}	1 t_{CLK}	-
Chip Select time for WRITE cycle	t_{CW}	2 t_{CLK}	-
Data setup time	t_{DS}	20 ns	-
Data hold time from end of write	t_{DHW}	10 ns	-
Write cycle recovery time	t_{WR}	2 t_{CLK}	-

Bemerkungen zu Tabelle 4.14:
t_{CLK} bezeichnet die Taktdauer des Histogrammers (typisch 25 ns)
Die angegebenen Zeiten beziehen sich auf einen durch Chip Select kontrollierten Zugriff, d.h. Read bzw. Write sind während des gesamten Zugriffs aktiv. Die gleichen Werte sind auch für durch Read bzw. Write gesteuerte Zugriffe gültig, d.h. CS ist während des gesamten Zugriffs aktiv.

Verwendung des DTACK-Signales
Eine Auswertung des Acknowledge-Signals DTACK macht eine separate und vom Takt des Histogrammers und des Hostbussystems abhängige Waitstate-Erzeugung unnötig. Die Auswertung von DTACK ist zu empfehlen, wenn Schreib- oder Lesezugriffe des Host-Bussystems kürzer sein können als die angegebenen Mindestwerte. Da im allgemeinen die Taktfrequenz des Chips höher sein wird als die Taktfrequenz des Host-Busses, können die Zugriffszeiten bestimmt durch die Phasenlage von Histogrammer- und Hosttakt zueinander von Zugriff zu Zugriff um die Dauer eines Hosttaktes differieren. Während ein mit Hosttakt gesteuerter Waitstate-Generator dabei von der maximalen Zugriffsdauer ausgehen muß, kann durch Auswertung des DTACK-Signales die maximale Daten-Transferrate erreicht werden. Die Abhängigkeit der Zugriffsdauer vom Startzeitpunkt des Zugriffs spiegelt sich in der Tabelle zum DTACK-Timing wieder.

Tabelle 4.15: DTACK Timing des Histogrammers

Parameter	Symbol	min.	max.
CS active to DTACK active for read cycle	t_{CDAR}	2,5* t_{CLK}	3,5* t_{CLK} + 10 ns
CS active to DTACK active for write cycle	t_{CDAW}	1,5* t_{CLK}	2,5* t_{CLK} + 10 ns
CS inactive to DTACK inactive for read cycle	t_{CDIR}	0,5 t_{CLK}	1,5 t_{CLK} + 10 ns
CS inactive to DTACK inactive for write cycle	t_{CDIW}	0,5 t_{CLK}	1,5 t_{CLK} + 10 ns

Bemerkungen zu Tabelle 4.15:
t_{CLK} bezeichnet die Taktdauer des Histogrammers (typisch 25 ns)
Halbe Taktzyklen ergeben sich aufgrund einer Synchronisation der Eingangssignale mit der negativen Taktflanke und der Ausgabe sämtlicher Signale mit der positiven Taktflanke.
Die angegebenen Zeiten beziehen sich auf einen durch Chip Select kontrollierten Zugriff, d.h. Read bzw. Write sind während des gesamten Zugriffs aktiv. Die gleichen Werte sind auch für durch Read bzw. Write gesteuerte Zugriffe gültig, d.h. CS ist während des gesamten Zugriffs aktiv.
*) Bei einem direkt auf einen Schreibzugriff folgenden Zugriff erhöht sich die Zeit um einen Takt.

4.5.3.3 Interne Realisierung des Buszugriffs

Die oben angegebenen Zeiten für die Buszyklen des Histogrammers hängen eng mit der Notwendigkeit zusammen, intern innerhalb eines Lese- oder Schreibzugriffes auf den Histogrammspeicher eine Reihe von aufeinanderfolgenden Operationen zu veranlassen. Der Zugriff auf die als Flip-Flops realisierten Registerbänke ist dagegen sehr einfach. Zum Verständnis, warum sich die angegebene Taktzyklenanzahl für einen Zugriff ergibt, soll im folgenden der interne Ablauf der Zugriffe gezeigt werden. Ein wesentlicher Grundsatz bei der Implementation des Speicherzugriffs war, keinesfalls einen die maximale Taktfrequenz des Chips begrenzenden Teil zu schaffen. Deshalb werden weitaus großzügigere Zugriffszeiten vorgesehen als bei der Histogrammaufzeichnung.

Aktionsreihenfolge beim Lesezugriff auf den Histogrammspeicher:
1. **Adressierung**: Anlegen der Speicheradresse an das RAM
2. **Lesezugriff**: RAM-Lesezugriff über Memory Enable starten und RAM-Adresse latchen
3. **Lösch-(Schreib-)zugriff**: Inkrement der Speicheradresse, Übergabe der Lesedaten an den Datenbus (Lesedaten latchen) Bereitstellung der Schreibdaten für einen ggf. automatisch folgenden Löschzugriff, RAM auf Schreiben umstellen
4. Beenden des Schreibzugriffs und Übernahme der Schreibdaten
5. Inkrement des Fenstersubadreßzählers (darf mit folgendem Zugriffs-Zyklus überlappen)

Aktionsreihenfolge beim Schreibzugriff auf den Histogrammspeicher:
1. **Adressierung**: Anlegen der Speicheradresse an das RAM
2. **Lesezugriff**: RAM-Lesezugriff über Memory Enable starten und RAM-Adresse latchen
3. **Schreibzugriff**: Inkrement der Speicheradresse und Anlegen des über die Schreibdaten modifizierten Speicherwortes an das RAM, RAM auf Schreiben umstellen
4. Beenden des Schreibzugriffs und Übernahme der Schreibdaten
5. Inkrement des Fenstersubadreßzählers

Es werden also sowohl beim Lesezugriff als auch beim Schreibzugriff Read-Modify-Write-Zugriffe auf das RAM vorgenommen. Beim Lesezugriff, um den Speicherinhalt selektiv nach dem Auslesen zu löschen, beim Schreibzugriff, weil über den Bus immer nur ein 4 bis 16 Bit

breiter Teil des Speicherwortes verändert wird und der Rest unverändert bleiben soll. Die Zusammenarbeit der genannten internen Aktionen mit der Ablaufsteuerung des externen Protokolls wird im Zustandsdiagramm gezeigt (Abbildung 4.14).

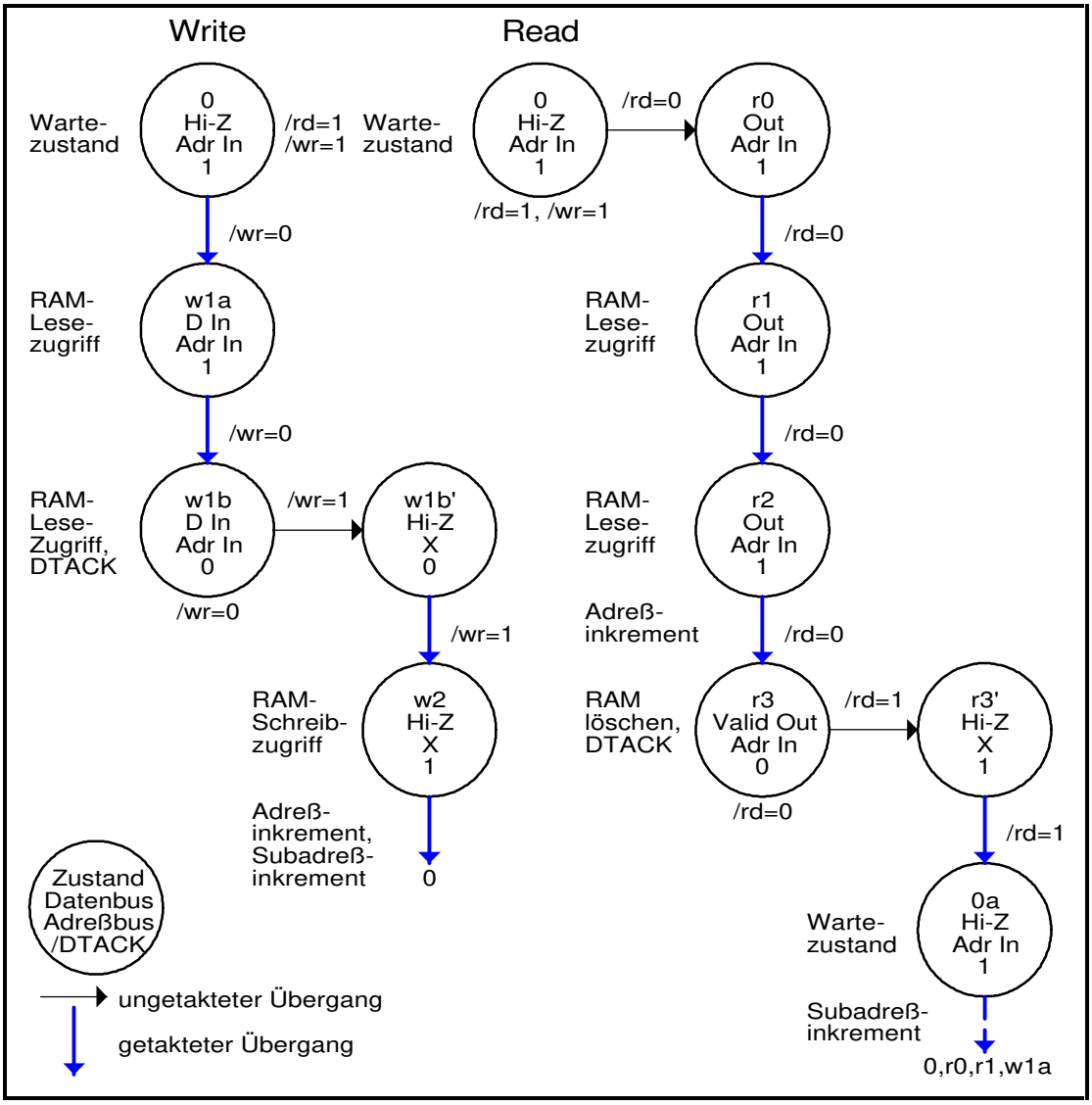

Abbildung 4.14: Zustandsdiagramm der Bus-Zugriffssteuerung

Bei Lese- und Schreibzugriff sind zwei Zustände für den Lesezugriff auf das RAM vorgesehen. Der folgende Modifikations-Schreibzugriff wird in einem Takt erledigt. Durch verzögerte Reaktion des Hostrechners auf das DTACK-Signal kann der RAM-Lesezugriff beim Bus-Schreibzugriff verlängert werden, während beim Bus-Lesezugriff dadurch der RAM-Schreibzugriff verlängert wird. Der erste Takt des Wartezustandes nach einem Lesezugriff ist explizit noch einmal ins Zustandsdiagramm mit aufgenommen, da erst nach diesem Takt das Inkrement der Subadresse vorgenommen wird. Im Zustandsdiagramm des Schreibzugriffs wird sichtbar, daß nach Rücknahme des DTACK-Signals noch ein interner Zustand folgt, in dem der eigentliche Schreibzugriff auf das RAM erfolgt. Da innerhalb der Dauer dieses Zustands bereits der nächste Buszugriff beginnen kann, wird der Folgezugriff in diesem Fall um einen Takt verlängert.
Das Zustandsdiagramm zeigt die asynchrone Adreß- und Datenübernahme und die asynchrone Steuerung des Tristate-Busses beim Lesezugriff. Ein Abbruch beider Zugriffe durch vorzeitige

Rücknahme der Bus-Steuersignale ist jederzeit möglich, kann jedoch zu inkonsistenten Änderungen des RAMs und der Zähler führen. Beim Lesezugriff ist ein vorzeitiger Abbruch der Aktion intern vorgesehen, damit es auch im Fehlerfall keine Kurzschlüsse auf dem Datenbus geben kann. Ebenso ist der Histogrammer vor Kurzschlüssen auf dem internen Datenbus geschützt, auch wenn zugleich das Schreib- und das Lesesignal aktiviert wird. Diese nicht spezifizierten Übergänge sind allerdings nicht in den Zustandsgraphen aufgenommen.

4.5.4 Realisierung der Bilddatenschnittstelle

Aufgabe der Bilddatenschnittstelle des Histogrammers ist es, Daten von der Kamera bzw. einer zwischen Kamera und Bildverarbeitungseinheit geschalteten Datenaufbereitungseinheit entgegenzunehmen. Diese Schnittstelle kann sehr einfach gestaltet werden, da sie unidirektional arbeitet. Ein bidirektionales Handshake ist hier nicht üblich.

Auf Bildaufnehmerröhren basierende Kameras tasten mit einem Elektronenstrahl eine lichtempfindliche Anode ab, auf die das Bild abgebildet wird. Der Strahl wird auf jeweils ein Pixel gerichtet und fährt zeilenweise das Bild ab. Nach der Abtastung jeweils einer Zeile entsteht eine kurze Pause, die für den Strahlrücklauf benötigt wird. Auf CCD-Technik (Charge Coupled Device) basierende Halbleiterkameras haben als Bildaufnehmer ein auf einem Halbleiterchip integriertes DRAM-ähnliches Array lichtempfindlicher Zellen, das ebenfalls seriell zeilenweise ausgelesen wird. Bei CCD-Sensoren ist keine Pause für einen Zeilenrücklauf notwendig. Bei praktisch allen Systemen wird nach dem Auslesen eines Bildes jedoch eine längere Pause gemacht, bevor das nächste Bild ausgelesen wird.
Für digitale Datenverarbeitung ausgelegte Kameras sind mit integriertem Bildspeicher für ein oder mehrere Bilder, externer Steuerung von Bildstart und Verschlußzeit, extern einspeisbarem Pixeltakt und unterschiedlichen Typen von Schnittstellen erhältlich (z.B. [20]).
Gemeinsam ist allen Kameras die zeilenorientierte pixelweise serielle Ausgabe der Bilddaten.
Übliche Pixelfrequenzen liegen im Bereich von 10 bis 25 MHz bei Bildfrequenzen zwischen 25 und 100 Hz. Die Auflösungen liegen dabei im Bereich von einigen Zehntausend bis zu einer Million Pixeln.
Die Übergabe der Bilddaten findet im allgemeinen mit einem festen der Pixelfrequenz entsprechenden Takt statt. Für jedes Bild wird ein Bild-Synchronimpuls geliefert. Zeilenimpulse finden nur Verwendung, wenn Bildschirmgeräte angeschlossen werden können sollen.

Das MAZ wünschte folgendes Schnittstellenverhalten:
Die Bilddatenschnittstelle des Histogrammers (im folgenden als Pixelschnittstelle bezeichnet) soll neben einem 10 Bit breiten Datenbus (PixelIn) für den Grauwert über einen Bildstart-Eingang (Start) und einen Eingang für den Pixeltakt (PClkIn) verfügen. Solange das Startsignal aktiv ist wird mit jedem Pixeltakt ein Pixeldatum auf dem Pixeldatenbus übertragen. Das Startsignal zeigt mit einer positiven Flanke den Beginn eines Bildes an und bleibt mindestens bis zur Übertragung des letzten Pixels aktiv. Der positive Pegel des Pixeltaktes zeigt das Anliegen eines gültigen Pixeldatums an.

Abbildung 4.16 zeigt das Protokoll der Pixelschnittstelle. Da der Pixeltakt nicht mit dem Takt des Histogrammers gekoppelt sein muß, ist es notwendig, den Pixeltakteingang innerhalb des Histogrammers über ein Flip-Flop zu synchronisieren. Der Takt kann jedoch nur korrekt erkannt werden, wenn abwechselnd ein High-Pegel und ein Low-Pegel erkannt wird. Jeder Pegel müßte also für mindestens einen Takt des Histogrammers anstehen, damit das Flip-Flop ihn weitergibt. Um diese Einschränkung zu entschärfen, wurde eine Schaltung zur Synchronisation des Pixeltaktes entworfen, die Taktimpulse schon dann erkennt, wenn sie lang

genug sind, um ein Flip-Flop über den Takteingang zu toggeln (Abbildung 4.15: Synchronisation des Pixeltaktes). Gleichzeitig verkürzt diese Schaltung die Dauer der positiven Pegel des regenerierten Pixeltaktes auf genau eine interne Taktperiode. Parallel zum Pixeltakt werden die Pixeldaten eingelesen. Wie bei allen asynchronen Systemen können Metastabilitäten der zur Synchronisation verwendeten Flip-Flops nicht ausgeschlossen werden. Für diese Flip-Flops sollten daher Typen mit niedriger Dauer metastabiler Zustände gewählt werden.

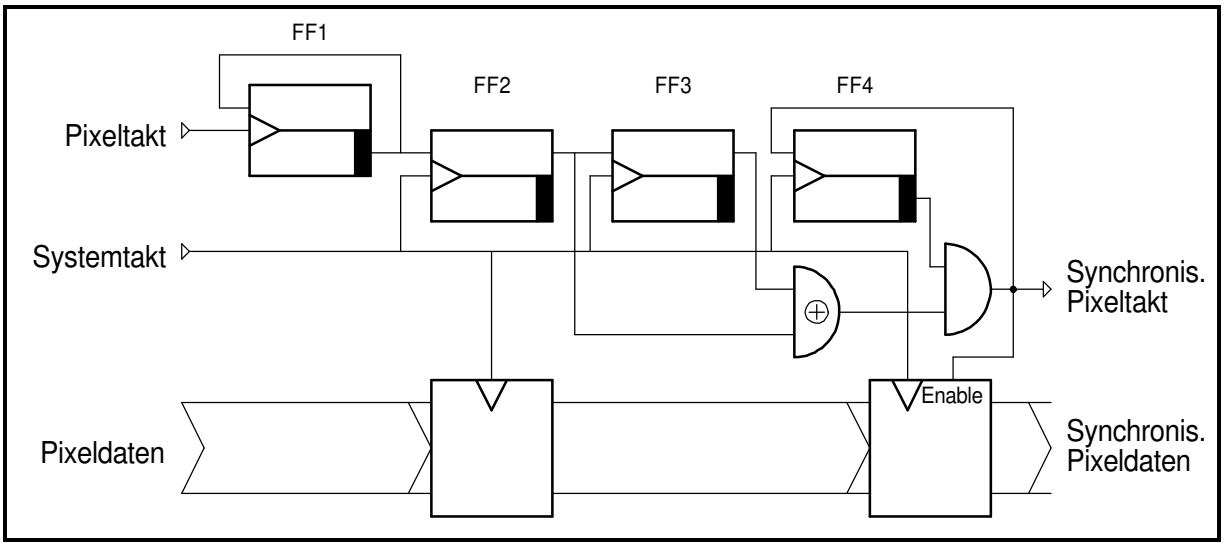

Abbildung 4.15: Synchronisation des Pixeltaktes

Beschreibung der Schaltung:
Flip-Flop FF1 nimmt eine Frequenzhalbierung des Pixeltaktes vor. Am Ausgang von FF1 steht also ein Signal an, das mit jeder steigenden Flanke des Pixeltaktes seinen Pegel wechselt. Dieses Signal wird über FF2 mit dem Systemtakt synchronisiert. FF3 speichert den jeweils vorigen Zustand des Signales. Über ein Exklusiv-Oder-Gatter werden Änderungen des frequenzhalbierten und synchronisierten Pixeltaktes erkannt und damit findet wieder eine Frequenzverdoppelung auf die ursprüngliche Frequenz statt. Für jeden Pixeltakt steht einen Systemtakt lang ein positiver Pegel am Ausgang des Exklusiv-Oder-Gatters an. Dieser Schaltungsteil arbeitet also für Pixeltaktraten bis zur vollen Frequenz des Systemtaktes. Für korrekte Funktion des Histogrammers wird jedoch vorausgesetzt, daß die Pixelfrequenz höchstens halb so hoch ist wie die Systemfrequenz. Um zu verhindern, daß Verletzungen dieser Bedingung zu undefinierten Reaktionen der folgenden Schaltungsteile führen, wird über FF4 und ein Und-Gatter sichergestellt, daß in jedem Fall auf einen High-Pegel ein Low-Pegel folgt.
Der untere Schaltungsteil zeigt die Verwendung des synchronisierten Pixeltaktes zur Synchronisation der Pixeldaten.
In Abbildung 4.18 findet sich ein Beispiel für den Signalverlauf an den Ein- und Ausgängen der Schaltung.

Solange die Pixelfrequenz deutlich niedriger ist als die halbe Taktfrequenz des Histogrammers, kann der Pixeltakt vollkommen asynchron zum Takt des Histogrammers liegen. Liegt die Pixelfrequenz jedoch so dicht an der halben Taktfrequenz, daß die Einhaltung der Dauer zweier Systemtakte je Pixeltakt nicht garantiert werden kann, ist nur eine synchrone Datenübertragung möglich. Synchron bedeutet, daß der Systemtakt des Histogrammers und der Pixeltakt durch Frequenzteilung aus einer gemeinsamen Referenzfrequenz gewonnen werden. Da der Histogrammer sämtliche zu synchronisierenden Eingangssignale jeweils mit der

positiven Taktflanke übernimmt, sollte der Pixeltakt jeweils kurz nach der positiven Flanke des Systemtaktes oder exakt mit der negativen Flanke getoggelt werden. Ein neues Datum kann bereits mit der negativen Flanke des Pixeltaktes zur Verfügung gestellt werden. Es ist darauf zu achten, daß bei den verwendeten hohen Frequenzen Signallaufzeiten auf der Platine und Laufzeiten von SSI-Gattern (TTL-, HC-Logikbausteine) nicht vernachlässigbar sind.
Eine praktische Grenze für den Übergang zu einer synchronen Taktung liegt je nach Höhe der Störsignale auf den Taktleitungen, die zu Verschiebung der Flanken führen können (Jitter), bei einer Pixeltaktfrequenz oberhalb von etwa 45% der Systemtaktfrequenz.
Tabelle 4.16 zeigt das Timing der Pixelschnittstelle. Die in Nanosekunden (ns) angegebenen Werte sind dabei großzügig gewählte Schätzwerte, deren exakter Wert durch die Verzögerung von Signalen auf dem Chip bestimmt werden wird.

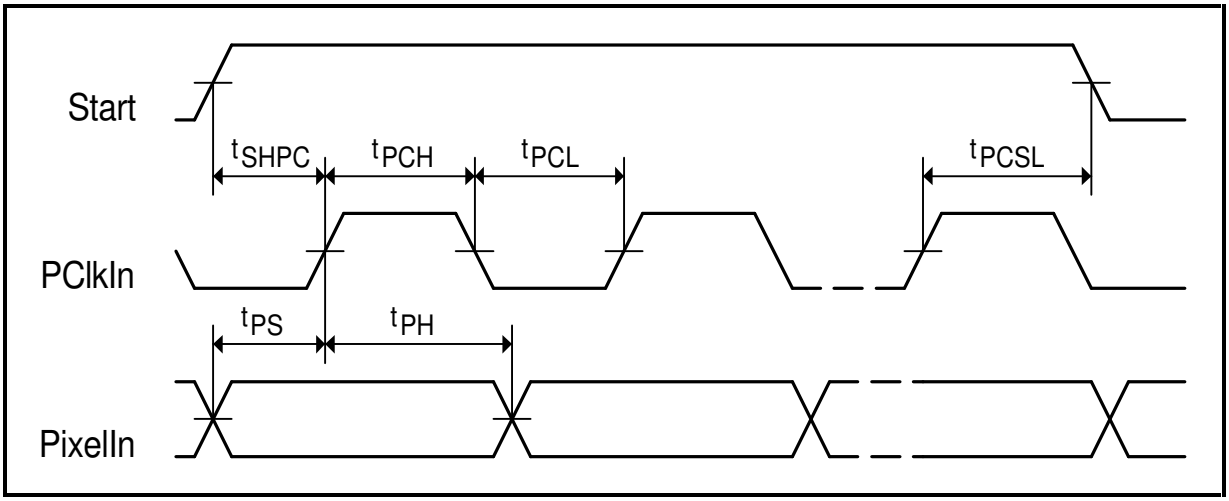

Abbildung 4.16: Timing der Pixelschnittstelle

Tabelle 4.16: Timing der Pixelschnittstelle

Parameter	Symbol	min.	max.
Pixel Clock time	$t_{PCH}+t_{PCL}$	$2 \cdot t_{CLK}$	-
Pixel Clock high time	t_{PCH}	5 ns	-
Pixel Clock low time	t_{PCL}	5 ns	-
Start high to Pixel Clock high time for first pixel	t_{SHPC}	5 ns	t_{PCLK} - 5 ns
Start hold time after Pixel Clock high for last pixel	t_{PCSL}	5 ns	t_{PCLK} - 5 ns *
Pixel Data setup time	t_{PS}	5 ns	-
Pixel Data hold time	t_{PH}	t_{CLK} + 5 ns	-

Bemerkungen zu Tabelle 4.16:
t_{CLK} bezeichnet die Taktdauer des Histogrammers, $t_{PCLK}=t_{PCH}+t_{PCL}$ die Dauer eines Pixeltaktes
*) Beschränkung nur bei externer Vorgabe des Bildendes, bei Beendung aufgrund der intern eingestellten Bildgröße muß das Startsignal spätestens kurz vor dem erneuten Setzen des Stopbits zurückgenommen werden.
Für die Leitungen zur externen Fenstersteuerung (WinEn) gilt das gleiche Timing wie für PixelIn.

Es ist nicht notwendig, daß der Pixeltakt kontinuierlich am Histogrammer anliegt.
In einer ersten Implementation wurde jedoch für korrekten Start und Abschluß der Bildverarbeitung innerhalb des Histogrammers aufgrund des internen Pipelinings ein weiterer

Takt nach Übertragung des letzten Pixels benötigt. Diese Notwendigkeit konnte durch Einführung eines zusätzlichen verzögerten internen Pixeltakt-Signals beseitigt werden.

Zur Ausgabe eines egalisierten Bildes soll der Histogrammer ebenfalls über einen Pixelausgang verfügen. An diesem Ausgang müssen neben den Pixeldaten auch das interne Startsignal und der interne Pixeltakt zur Verfügung stehen, da diese Signale innerhalb des Histogrammers synchronisiert und um einige Takte verzögert werden. Wenn nach Auswahl eines Gehäuses für den Histogrammer noch genügend Anschlüsse des Gehäuses unbelegt bleiben, sollen auch die internen Vergleichsergebnisse der Fensterkomparatoren parallel zu den Pixeldaten ausgegeben werden. Das Timing des Pixelausganges soll dem Timing des Pixeleinganges entsprechen. Genaue Timingwerte stehen jedoch erst nach einer abschließenden Simulation des Histogrammers zur Verfügung.

4.6 Timing-Konzept des Histogrammers

Nach der Untersuchung einzelner Teile des Histogrammers und einzelner Aspekte des Timings in den vorangehenden Kapiteln ist es notwendig, das Timing aller Teile aufeinander abzustimmen. Dieses Problem wird im Histogrammer wesentlich durch den Datenfluß zum RAM während der Bildübertragung bestimmt, da hier alle Eingabedaten zeitlich passend zum RAM-Timing eintreffen müssen, damit der Zeit-Spielraum für die einzelnen Komponenten möglichst unkritisch ist. Bei dieser Untersuchung ist auch zu entscheiden, ob ein Pipelining der Daten innerhalb der Komponenten, die Daten und Adressen für das RAM liefern, den maximalen Durchsatz erhöhen kann. Die Implementation einer Pipeline würde beim Histogrammer, abgesehen von dem Aufwand für Halteregister, keine größeren Probleme mit sich bringen. Dies liegt daran, daß der Histogrammer, im Gegensatz zu beispielsweise Mikroprozessoren, sehr lange Datenströme sequentiell verarbeitet, bevor ein Punkt erreicht wird, an dem die Pipeline wieder geleert werden muß. Beim Histogrammer muß die Pipeline nämlich nur am Bildende geleert werden. So wird weder die Steuerung der Pipeline besonders kompliziert, noch der mittlere (verzögerte) Durchsatz (sustained performance) gegenüber dem Spitzendurchsatz (peak performance) merklich niedriger sein. Wie allerdings bereits festgestellt wurde, ist innerhalb des sehr zeitkritischen Histogrammteils mit der gewählten Implementation kein Pipelining möglich (s. Kapitel 4.3). Da die maximale Geschwindigkeit durch den Histogrammteil bereits beschränkt wird, soll versucht werden, mit einer möglichst geringen Anzahl von Pipelinestufen auszukommen, um das Design übersichtlich zu halten.

Zur Ermittlung eines geeigneten Timings wurde der für die Histogramm-Erstellung wesentliche Teil des Histogrammers, bestehend aus Pixeleingang, Bildzähler und Fensterkomparatoren und der RAM-Inkrementer-Schleife sowie die Timing-Diagramme zum RAM RMW-Zyklus (Abbildung 4.4) untersucht. Unter Beachtung der Taktung der einzelnen Einheiten wurde der Signalverlauf bei unterschiedlichen Lagen des Pixeltaktes relativ zum Takt des Histogrammers betrachtet (Abbildung 4.18: Timing bei Histogrammerstellung und Kachelmodus). Der Klarheit halber sind die externen Eingangssignale des Histogrammers gegenüber dem Takt versetzt dargestellt. Die einzige Anforderung an die Lage der Pixeldaten relativ zum externen Pixeltakt ist, daß die Pixeldaten bei Erkennung der steigenden Pixeltaktflanke durch den Histogrammer gültig sein müssen. Die Nummern in den Datenbussen zeigen, zu welchem Pixel die jeweiligen Datenworte gehören. Weiterhin wurde das Timing-Diagramm geeignet um die Signale zur RAM-Ansteuerung (aus Kapitel 4.3.2) ergänzt. Aus dem Timing-Diagramm geht hervor, zu welchem Zeitpunkt Adressen und Daten zur RAM-Ansteuerung für jedes Pixel gültig sein müssen. Anhand dessen wurden die notwendigen Adreß- und Datenverbindungen zum RAM und Histogramminkrementer vorgenommen (s. Abbildung 4.17).

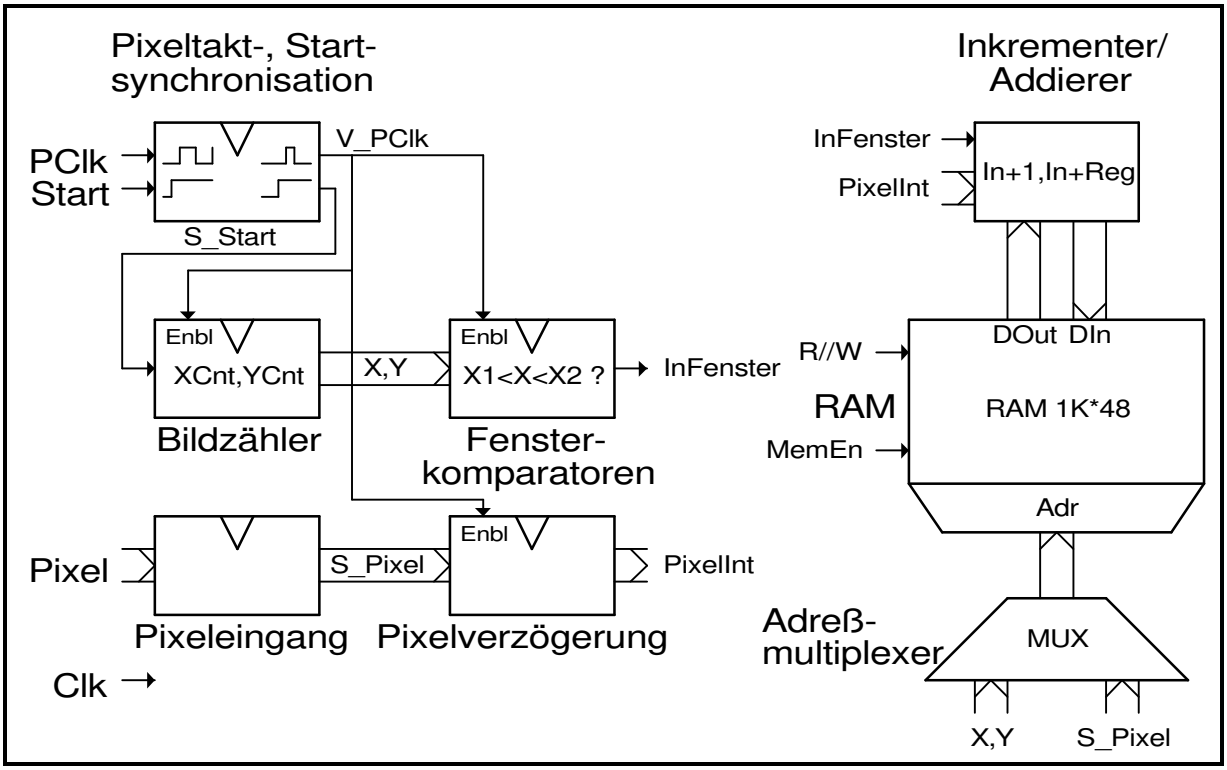

Abbildung 4.17: Ausschnitt aus Blockschaltbild

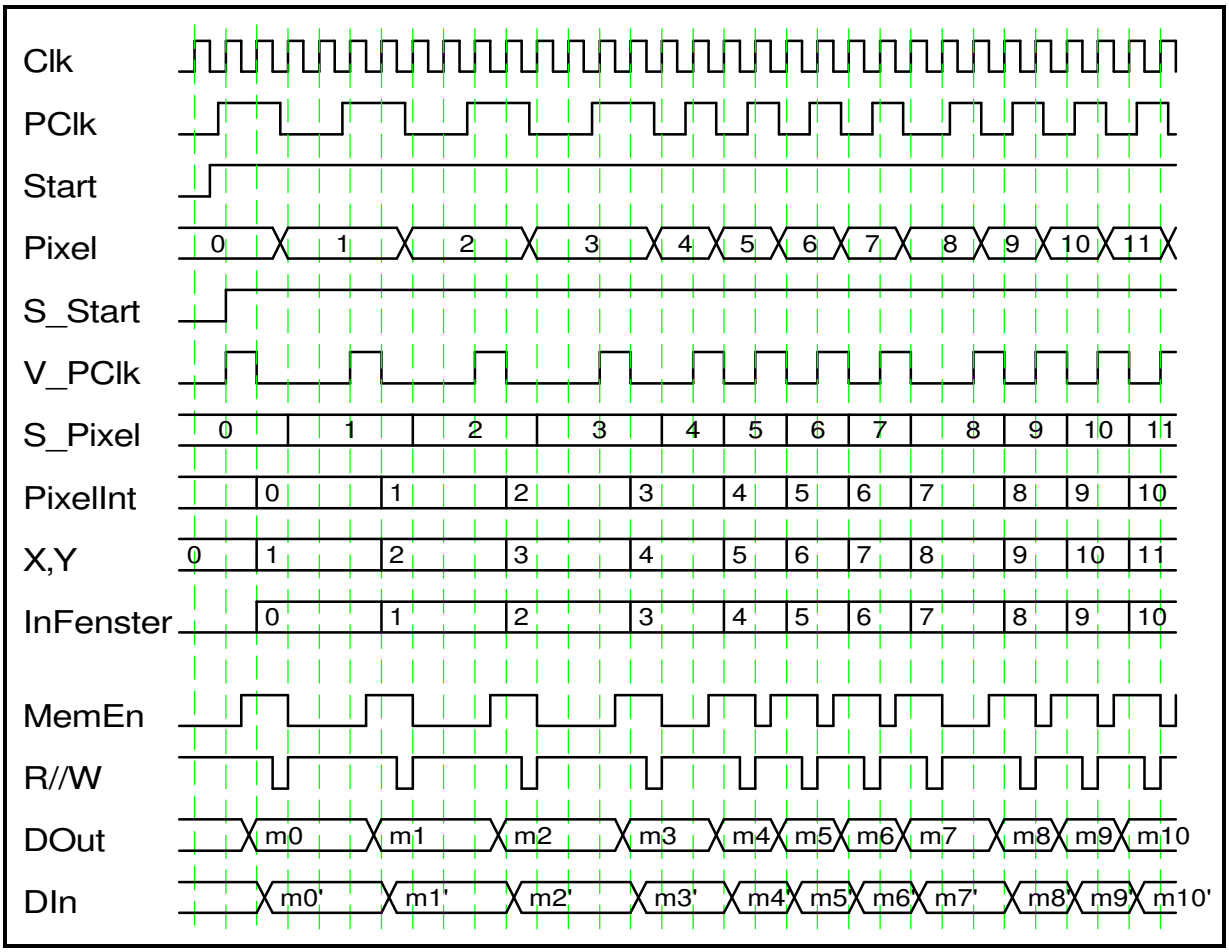

Abbildung 4.18: Timing bei Histogrammerstellung und Kachelmodus

Die Abbildungen (Abbildung 4.17, Abbildung 4.18) zeigen, daß die Eingangsdaten für den Inkrementer/ Addierer verschoben gegenüber den RAM-Adressen geliefert werden können. Da als RAM-Adresse wahlweise die Pixeldaten (im Histogrammmodus) oder die Bildposition (im Kachelmodus) dienen, müssen beide zum gleichen Zeitpunkt bezüglich einer Pixelübertragung gültig sein. Erst einen Pixeltakt später müssen die Pixeldaten für den Kachelmodus gleichzeitig mit der Fensterkomparatorinformation gültig sein. So steht sowohl für Bildzähler als auch für den Fensterkomparator ein ganzer Pixeltakt zur Berechnung der Ausgangssignale zur Verfügung. Damit ist also ein zweistufiges Pipelining, wie im Blockschaltbild dargestellt, naheliegend. Alle Einheiten werden nur bei Erkennung einer positiven Pixeltaktflanke einmal getaktet.

4.6.1 Ablaufsteuerung

Wie in den vorangehenden Kapiteln deutlich wurde, gibt es innerhalb des Histogrammers voneinander unabhängige Funktionsabläufe. Dazu gehören die Buszugriffsteuerung und die Steuerung des Speicherzugriffs. So ist es beispielsweise möglich, während einer Histogrammerstellung über den Prozessorbus auf Register des Histogrammers zuzugreifen. Dennoch muß eine Koordination beider Abläufe miteinander möglich sein, da auch über den Bus der Zugriff auf den Histogrammspeicher möglich sein soll.
Während der Speicherzugriff im Zeitraster halber Takte gesteuert wird, können die anderen Operationen im Zeitraster ganzer Takte gesteuert werden.
Eine Steuerung aller Einheiten des Histogrammers über ein globales Schaltwerk erscheint als nicht sinnvoll, da dieses sehr komplex würde und nicht einheitlich mit einer Flanke des Taktes

getaktet werden kann. Stattdessen werden alle Abläufe soweit möglich lokal innerhalb der jeweiligen Einheiten gesteuert. So kann eine weitgehende Unabhängigkeit der Einheiten voneinander erreicht werden, so daß kleine Änderungen der Spezifikation meist nur eine Einheit beeinflussen. Da der Histogrammspeicher bei einer Reihe von Aktivitäten exklusiv benötigt wird, geschieht nur die Steuerung des Speicherzugriffs und die Steuerung der am Speicherzugriff direkt beteiligten Einheiten über eine zentrale Logik. Da die Steuersignale für den Speicher hazardfrei erzeugt werden müssen, sind die Zähler zur Steuerung der Abläufe einschrittig codiert (vgl. Kapitel 4.3.3).

4.6.2 Taktung

Bei Realisierung eines Bausteins, der sehr viele Register enthält, ist zu entscheiden, ob die Register nur bedingt bei Bedarf getaktet werden sollen (gated clock) oder alle Register mit jedem Takt synchron geladen werden sollen.
Eine bedingte Taktung bringt den Vorteil einer deutlichen Reduktion der Leistungsaufnahme eines Bausteins, da Register nur dann getaktet werden müssen, wenn wirklich ein neuer Wert übernommen werden soll. Dazu wird das Tatksignal jedes Registers über ein Und-Gatter nur bedingt *enabled*. Bei hochintegrierten Bausteinen für batteriebetriebene Geräte und modernen, mit sehr hoher Taktfrequenz betriebenen Mikroprozessoren wird diese Technik bereits angewandt.
In einem synchron getakteten Design dagegen werden alle Register mit dem selben Taktsignal versorgt und über Eingangsmultiplexer wahlweise ein neuer Wert oder erneut der vorherige Inhalt des Registers übernommen. Dadurch, daß alle Register mit jedem Systemtakt getaktet werden, ergibt sich eine hohe Stromaufnahme auch dann, wenn nur wenige Register mit neuen Werten geladen werden.
Sollen Einheiten bedingt getaktet werden, muß sichergestellt werden, daß Verzögerungszeiten in unterschiedlichen Signalpfaden zueinander passen. Trotz der Verknüpfung des Taktes eines Registers mit einem weiteren Signal ist beispielsweise sicherzustellen, daß mehrere in der Art eines Schieberegisters hintereinandergeschaltete Register mit minimalem Versatz getaktet werden, um einen korrekten Datenfluß zu ermöglichen. Weiterhin sind Hazards auf der *Enable*-Leitung in mindestens einer Taktphase unbedingt zu vermeiden.

Für den Histogrammer wurde ein in sich synchron getaktetes Design gewählt, da eine Verwendung bedingter Taktung besondere Ansprüche an das Entwurfssystem und den Designer stellen. Die Technik der bedingten Taktung findet erst in den letzten Jahren vermehrt Anwendung in der Industrie.

4.6.3 Übernahme externer Signale

Bei der Verarbeitung asynchroner externer Signale wie dem Pixeltakt, dem Startsignal und den Bussteuersignalen, ist besondere Vorsicht geboten. Werden solche Signale direkt in einem Schaltwerk ausgewertet, beeinflussen sie oft den Zustand mehrerer Flip-Flops. Abbildung 4.19 zeigt einen solchen Fall. Durch Laufzeitunterschiede zwischen den einzelnen Pfaden zu den Flip-Flops wird der Übergang des Schaltwerks in einen unvorhergesehenen Zustand sehr wahrscheinlich. Ändert sich das Signal nämlich kurz vor der Taktung des Registers, erhält ein Flip-Flop innerhalb des Registers, zu dem das Signal nur wenige Gatter durchlaufen muß, bereits einen aus dem geänderten Zustand berechneten Wert, während ein Flip-Flop, zu dem das Signal mehr Gatter durchlaufen muß, noch gemäß dem alten Zustand des Signals geladen wird.

Eine Möglichkeit, solche Situationen zu vermeiden, ist die einschrittige Codierung des Schaltwerkszustandes. Dies kann jedoch unter Umständen zu einer sehr aufwendigen Lösung führen.
Eine weitere Möglichkeit ist es, asynchrone Signale über ein Flip-Flop zu synchronisieren, bevor sie Eingang in ein Schaltwerk finden. Dies ist unter minimaler Verzögerung der Reaktionszeit der Schaltung schon durch Einfügung eines Latches möglich (s. Abbildung 4.19). Das Latch übernimmt das Signal nur während des positiven Taktpegels und hält den Wert während des negativen Taktpegels, also vor der steigenden Flanke, konstant. Auf diese Weise wird das Eingangssignal verglichen mit direkter Auswertung um maximal eine halbe Taktperiode zusätzlich verzögert. Das Latch kann wahlweise auch durch ein negativ flankengetriggertes D-Flip-Flop ersetzt werden.

Unabhängig davon, ob das Schaltwerk einschrittig codiert wurde oder ein Flip-Flop zur Synchronisation vorgeschaltet wurde, können asynchrone Eingangssignale zum Einspeichern eines undefinierten Pegels in ein Flip-Flop führen. Ändert sich das Eingangssignal eines Flip-Flops nämlich zu einem ungünstigen Zeitpunkt bezüglich der aktiven Taktflanke, kann das Flip-Flop einen Pegel einspeichern, der zwischen den Logikpegeln liegt. Diese Situation führt dazu, daß der Ausgang des Flip-Flops für eine unbestimmte Zeit auf undefiniertem Potential bleibt. Das Flip-Flop befindet sich währenddessen in einem sogenannten metastabilen Zustand. Im allgemeinen ist der eingespeicherte Pegel jedoch einem der Logikpegel näher, so daß eine höhere Tendenz besteht, diesen Pegel einzunehmen. Rauschen trägt dazu bei, diesen Vorgang zu beschleunigen. Innerhalb dieser Zeit ist es unbestimmt, wie eine nachfolgende Logik das Ausgangssignal interpretiert.
Für die Auswertung asynchroner Signale sollten spezielle Synchronisations-Flip-Flops gewählt werden. Synchronisations-Flip-Flops sind intern so aufgebaut, daß metastabile Zustände eine möglichst geringe Dauer haben. Dazu müssen Synchronisations-Flip-Flops über eine hohe interne Verstärkung verfügen. Zusätzlich muß verhindert werden, daß ein metastabiler Zustand zu Schwingungen des Flip-Flops führt. Dazu müssen diese Flip-Flops über eine Frequenzkompensation verfügen, die dafür sorgt, daß die Schleifenverstärkung des Speicherelements im Flip-Flop bei Phasendrehungen über 180 Grad deutlich unterhalb von Eins liegt (vgl. [18]). Aufgrund dieser Anforderungen sind Synchronisations-Flip-Flops komplizierter aufgebaut als normale Flip-Flops und werden wegen des erhöhten Flächenbedarfs auf dem Chip nur an gefährdeten Stellen eingesetzt.
Die Wahrscheinlichkeit, daß sich ein Synchronisations-Flip-Flop eine gewisse Zeit nach der Taktflanke noch in einem metastabilen Zustand befindet, nimmt negativ exponentiell mit dem zeitlichen Abstand zur Taktflanke ab [31,17]. Deshalb ist es sinnvoll, innerhalb der Schaltung möglichst viel Zeit für den Einschwingvorgang des Flip-Flops vorzusehen. Hier ist abzuwägen zwischen Betriebssicherheit der Schaltung und Zeitverlust durch Synchronisation. Prinzipiell gibt es keine Möglichkeit, metastabile Zustände aufgrund asynchroner Signale vollkommen auszuschließen. Es ist jedoch möglich, einen Fehler durch Metastabilität äußerst unwahrscheinlich zu machen.

Eine Möglichkeit, das Problem asynchroner Eingangssignale des Histogrammers zu umgehen, ist der Aufbau eines vollkommen synchron arbeitenden Bildverarbeitungssystems. Dies kann jedoch nur verwirklicht werden, wenn das gesamte System inklusive Auswertungsrechner komplett entworfen wird. Bei modular veränderbaren Systemen ist es schwierig, alle Komponenten synchron zu takten.

Oft kann nichteinmal durch Simulation einer Schaltung geprüft werden, ob Eingangssignale fehlerhaft synchronisiert werden. Eine Schaltung mit fehlerhafter Synchronisation, wie die in Abbildung 4.19 gezeigte, kann aufgrund des Simulationsergebnisses oft nicht von einer Schaltung mit korrekter Synchronisation unterschieden werden, da im kritischen Bereich der

Verschiebung zwischen Taktsignal und Eingangssignal in jedem Fall Verletzungen der Setup-Zeit am Dateneingang von Flip-Flops entstehen. Sie veranlassen den Simulator, „Undefiniert“ im Flip-Flop einzuspeichern, da nicht bekannt ist, welchen Pegel das Flip-Flop einspeichern würde. Eine weitere Simulation ist dann meist unsinnig, da sich die undefinierten Pegel in Kürze in der gesamten Schaltung ausbreiten können. Der Simulator muß sich so verhalten, da er annehmen muß, daß der Zeitpunkt, zu dem das Eingangssignal erkannt wird, für die Funktion der Schaltung relevant ist. Für den Anwender ist es dagegen oft unbedeutend, ob das Signal sofort oder erst im Folgetakt übernommen wird.

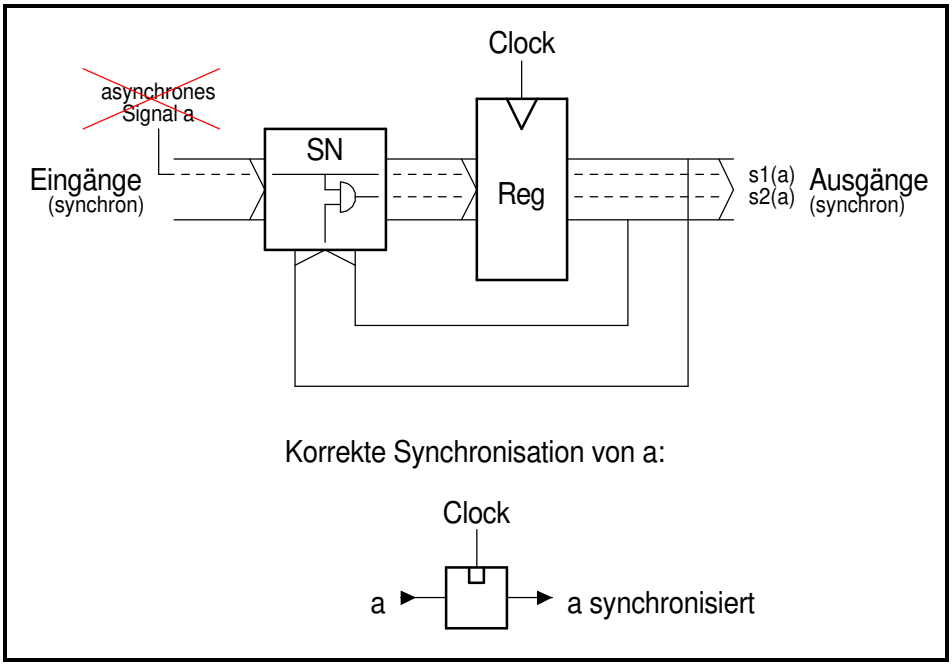

Abbildung 4.19: Übernahme asynchoner Signale

4.7 Überleitung

Kapitel 1 gibt neben einer Einleitung einige Argumente für die Realisierung eines Histogrammers als Chip. In Kapitel 2 wird der Aufgabenbereich eines Histogrammers innerhalb eines Bildverarbeitungssystems aufgezeigt. Kapitel 3 listet das Pflichtenheft des hier zu realisierenden besonders universellen Histogrammers auf und zeigt seinen grundsätzlichen Aufbau. Ein Konzept zur Realisierung aller wesentlichen Einheiten wird in Kapitel 4 entwickelt. Gleichzeitig werden Schnittstellen und Verhalten des Histogrammers spezifiziert. Damit ist der Entwurf des Histogrammers abgeschlossen.
Es folgt die Implementation des Histogrammers. Dazu wird zunächst entsprechend dem Konzept ein Verhaltensmodell in der Hardware-Beschreibungssprache VHDL erstellt. Danach wird das Verhaltensmodell in ein synthesefähiges Modell umgesetzt. Dann folgt die Synthese mit einem Standardzell-Entwurfssystem.

5 Realisierung des Histogrammers mit VHDL

5.1 Top-Down-Entwurf mit VHDL

Es ist unmöglich, eine integrierte Schaltung ausgehend von einer Problemstellung direkt in einem Schritt zu entwickeln.
Beim Top-Down-Design geschieht der Entwurf in mehreren Stufen, die systematisch von einer sehr abstrakten Beschreibung zu einer konkreten Struktur führen. Da die Realisierung aber auf die verfügbaren technischen Möglichkeiten Rücksicht nehmen muß, spielen auch sogenannte Bottom-Up-Einflüsse eine wesentliche Rolle. Im Bereich der Halbleitertechnologie und bei der Realisierung darauf basierender Grundschaltungen wird daher Bottom-Up entwickelt. Der Histogrammer wird mit einem Standardzell-Entwurfssystem als ASIC (Application Specific Integrated Circuit) realisiert. Der Anteil der Bottom-Up-Entwicklung ist dabei schon vom Halbleiterhersteller erbracht, so daß der Anwender des Standardzell-Entwurfssystems seine Schaltung hierarchisch von einer hohen Abstraktionsebene her anfangend nach der Top-Down-Strategie entwickeln kann.

Die Notwendigkeit, eine integrierte Schaltung zu entwerfen, ergibt sich meist im Verlauf eines System-Entwurfes (s. Abbildung 5.1). Ist die grundsätzliche Idee zu einem System entstanden, beginnt die Systemplanung. Dabei werden alle Informationen zusammengestellt, die für das System und sein Umfeld spezifisch sind. Begonnen wird damit, Ein- und Ausgabeverhalten des Systems festzulegen. Das System wird dazu vorerst als *black box* betrachtet. Anschließend wird das System selbst in funktionale Blöcke aufgespalten (Partitionierung). Bis zu diesem Punkt kann auf einem sehr abstrakten Niveau vorgegangen werden. Nach der Partitionierung muß überprüft werden, ob die einzelnen Blöcke technisch realisierbar sind oder ob bereits geeignete Bausteine kommerziell erhältlich sind. Ist ein geeignetes Konzept für alle Blöcke gefunden, müssen deren Schnittstellen definiert werden. Die so entstandene Systemspezifikation ist Grundlage für die Entwicklung der einzelnen Komponenten.
Als erster Schritt der Entwicklung einer solchen Komponente muß ihre Spezifikation detailliert aufgearbeitet werden. Anhand dieser Spezifikation wird im Falle einer integrierten Schaltung oft ein Simulationsmodell (Verhaltensmodell) erstellt, um die Funktionalität im Detail festzulegen und zu überprüfen. Anhand des Simulationsmodells wird eine Beschreibung auf Registertransfer-Ebene entwickelt und durch das Entwurfssystem automatisch auf Gatterebene umgesetzt. Durch eine erneute Simulation wird die Funktion geprüft. Bei dieser Simulation werden die Gatter durch Modelle nachgebildet. Aus der Beschreibung auf Gatterebene kann interaktiv bzw. automatisch ein Layout für den Chip generiert werden. Nach diesem Schritt erfolgt eine detaillierte Simulation des Chips unter Beachtung des physikalischen Layouts. Das Layout wird dann zusammen mit Testdaten an den Halbleiterhersteller übergeben.

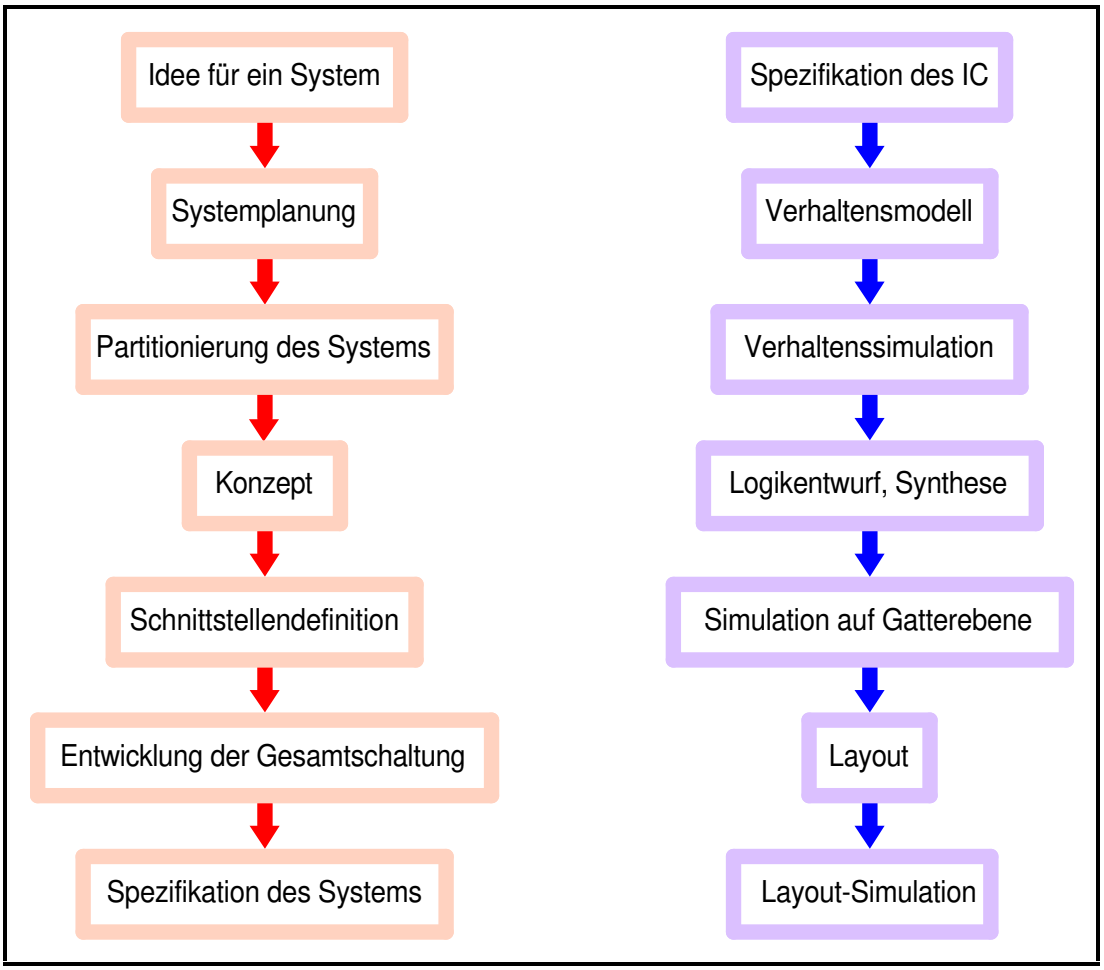

Abbildung 5.1: Systementwurf, Top-Down-Design

Die klare Trennung zwischen Systementwurf und Entwurf der einzelnen Blöcke und die zeitliche Abfolge der Schritte des System-Entwurfes werden in der Praxis oft nicht eingehalten. Gründe hierfür können beispielsweise zeitliche Zwänge sein. Auch bei Beginn des Entwurfes des Histogrammers war noch unklar, wie seine Schnittstellen aussehen sollten und in welcher Umgebung er eingesetzt werden sollte. Aus diesem Grund entstand die Spezifikation seiner Schnittstellen parallel und in Wechselwirkung mit der Entwicklung des Histogrammers.

Zur Entwicklung des Histogrammers wurde die Hardware-Beschreibungssprache VHDL ausgewählt. VHDL [11] ist eine prozedurale Programmiersprache. VHDL unterstützende Simulations- und Synthese-Werkzeuge ermöglichen es, einen Großteil der Schritte von der Spezifikation bis zum Layout eines Chips nahezu nahtlos aneinanderzufügen. In VHDL kann sowohl losgelöst von einer konkreten Struktur Verhalten beschrieben werden, als auch eine Struktur detailliert beschrieben werden.

Die folgende Übersicht (Tabelle 5.1) zeigt detailliert am Beispiel des Histogrammers, wie das Vorgehen beim Entwurfsablauf einer integrierten Schaltung mit einem Standard-zellentwurfssystem unter Benutzung von VHDL aussehen kann.