Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DIPLOMARBEIT

Erweiterung eines Video-Controllers zur Unterstützung eines DDR-Speichers

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

o. Univ.-Prof. Dr. Dietmar Dietrich

Dipl.-Ing. Ingo Wieser

am

Institut für Computertechnik (E384)

der Technischen Universität Wien

durch

Georg Csecsinovics

8826513

Anton Haglgasse 10, 3003 Gablitz

Gablitz, am 7.September 2008 _____

Kurzfassung

Im Rahmen dieser Arbeit wird ein maßgeschneiderter Video-Controller zur Ansteuerung eines Displays für einen Arbeitsplatz in der Flugsicherung modifiziert, um einen kostengünstigeren und leistungsfähigeren Videospeicher einsetzen zu können. Der bestehende Video-Controller verwendet als externen Videospeicher ein SRAM (Static Random Access Memory).

Um einen DDR-SDRAM-Speicher (Double Data Rate-Synchronous Dynamic RAM) einsetzen zu können, wurde ein neues Schnittstellenmodul entwickelt. Die Anforderungen an dieses Modul sind die Erfüllung der Funktion des Video-Controller trotz des nichtdeterministischen Zugriffs auf einen dynamischen Speicher über einen DDR-SDRAM-Controller sowie die Überbrückung von unterschiedlichen Taktfrequenzen und Wortbreiten zwischen Video-Controller und DDR-Controller. Um den Datenfluss an das Display sicherzustellen, werden Datenblöcke, vorzugsweise im Umfang einer Bildschirmzeile, vorgeladen und in einem Pufferspeicher verwaltet. Größe und Ladezeitpunkt können über Parameter flexibel eingestellt werden. Für die effiziente Durchführung aller anderen Schreib- und Leseoperationen stehen weitere Einheiten im neu entwickelten Schnittstellenmodul zur Verfügung.

Zur Verifikation des modifizierten Video-Controllers wurde ein komplettes Bildverarbeitungssystem in der Simulation nachgestellt, bei dem der Befehlssatz des Video-Controllers zur Generierung von Testbildern angewendet wurde. Um die von der Simulation des Displays erstellten Bitmap-Dateien einfach durch einen Dateivergleich überprüfen zu können, wurde eine Software zur Berechnung der Grafikoperationen erstellt.

Der Video-Controller konnte erfolgreich erweitert werden um ein DDR-SDRAM verwenden zu können. Trotz nichtdeterministischen Zugriffes auf den dynamischen Speicher werden die Bilddaten zeitgerecht an den Monitor ausgegeben und die Grafikbefehle korrekt ausgeführt. Um die Verarbeitungsleistung für Grafikoperationen mit zusätzlicher Logik zu steigern, wurden zusätzliche Verbesserungen durchgeführt bzw. angeregt.

Abstract

The scope of this project is to modify a customized video controller that triggers a work station display at a flight control centre so that a more economic and high performance video memory can be used. The existing video controller uses a SRAM (Static Random Access Memory) as an external video memory.

To be able to use a DDR-SDRAM-Memory (Double Data Rate-Synchronous Dynamic RAM), a new interface connection module has been developed. The requirements to this module are to ensure the functionality of the video controller despite a non-deterministic access to dynamic memory via a DDR-Controller as well as the bridging of different clock frequencies and bit width between the video controller and DDR-Controller. To make sure the data flow to the display is not interrupted, blocks of data, preferably of the size of a pixel line, are preloaded and managed in a buffer memory. The size and timing of the loading of the picture data can be managed flexibly by parameters. In the newly developed interface module, the efficient execution of all other write and read operations is guaranteed by additional subunits. Further logic had to be implemented that synchronizes the access to the memory for the visualisation on the monitor with the write and read accesses to the picture data for graphical operations.

To verify the modified video controller, a graphic system has been simulated that uses the instruction set of the video controller to generate test images. In order to simplify the verification of the bitmap files generated by the simulation display, software to calculate the graphic process was developed.

The video controller was modified successfully to use a DDR-SDRAM-Memory. Despite non-deterministic access to the dynamic memory, the monitor can be supplied with picture data on time and graphical operations can be executed correctly. To increase the speed of graphic processing, additional logic was developed and suggested.

Danksagung

Für die Unterstützung bei meiner Diplomarbeit möchte ich bei meinem Betreuer Dipl.-Ing. Ingo Wieser und Dipl.-Ing. Hannes Muhr von Oregano Systems herzlich bedanken.

Ein ganz besonderer Dank gilt meiner ganzen Familie für ihre Unterstützung während des Studiums.

Inhaltsverzeichnis

1. Einleitung 1
1.1 Bestehendes Videosystem 1
1.2 Aufgabenstellung 3
1.3 Lösungssuche 4

2. Bilddarstellung in Computersystemen 7

3. Video-Controller 10
3.1 Aufbau 10
3.2 Videospeicher 11
3.3 Befehlssatz 12
3.4 Einheiten des Video-Controllers 14
3.4.1 Registereinheit 14
3.4.2 Einheit Bildschirmaufbau 17
3.4.3 Einheit Zeichenoperationen 21
3.4.4 Einheit Signaltöne 24
3.4.5 Speicherschnittstelle 27

4. Halbleiterspeicher 30
4.1 SRAM 30
4.2 DRAM 32
4.3 DDR-SDRAM 37

5. DDR-SDRAM-Controller 41

6. Modifikation des Video-Controllers 44
6.1 Aufbau der modifizierten Schnittstelle 44
6.2 Komponente Bildschirm 45
6.2.1 Struktur der Komponente Bildschirm 48
6.2.2 Starten des Ladevorganges 49
6.2.3 Anfordern der Daten aus dem DDR-Speicher 50
6.2.4 Schreiben der Bilddaten in das Dualport-RAM 51
6.2.5 Datenausgabe an die Einheit Bildschirm 51
6.2.6 Dimensionierung des Dualport-RAMs 52
6.3 Komponente Lesen 54
6.3.1 Struktur der Komponente Lesen 54

6.3.2 Starten des Ladevorganges und Ausgabe der Daten ... 56
6.3.3 Laden der Daten aus dem DDR-SDRAM ... 57
6.3.4 Setzen der Haltesignale ... 58
6.4 Komponente Schreiben ... 58
6.4.1 Struktur der Komponente Schreiben ... 59
6.4.2 Zusammenstellung der Schreibbefehle ... 60
6.4.3 Schreiben der Daten in den DDR-Speicher ... 61
6.4.4 Dimensionierung des Dualport-RAMs ... 62
6.5 Komponente Arbiter ... 63
6.5.1 Struktur des Arbiters ... 63
6.5.2 Vergabe des Speicherzugriffs ... 66
6.5.3 Zuweisung der eintreffenden Lesedaten ... 66
6.5.4 Setzen der Haltesignale ... 67
6.6 Komponente Datenausgabe ... 67

7. Verifikation ... 69
7.1 Einführung ... 69
7.2 Verifikationsebenen ... 71
7.3 Simulationsaufbau ... 72
7.4 Testfälle ... 75
7.4.1 Testfall Verifikation des Bildaufbaues ... 75
7.4.2 Testfall Verifikation der Schreib- und Lesezugriffe ... 76
7.5 Verifikationssoftware ... 78
7.6 Verifikation der flimmerfreien Zeichenoperationen ... 81

8. Weiterführende Optimierungen ... 84
8.1 Engpass bei der Datenausgabe ... 84
8.2 Engpass Komponente Lesen ... 85

9. Zusammenfassung und Ausblick ... 89

Wissenschaftliche Literatur ... 92

Internetreferenzen ... 92

Abbildungsverzeichnis ... 93

Tabellenverzeichnis ... 96

Abkürzungen

BMP	Windows Bitmap
CAS	Column Access Strobe
CL	CAS Latency
CPU	Central Processing Unit
DAC	Digital Analog Converter
DDR	Double Data Rate
DRAM	Dynamic RAM
EDO	Extended Data Output
FPGA	Field Programmable Gate Array
FIFO	First In First Out
IP Core	Intellectual Property Core
JEDEC	Joint Electron Device Engineering Council
PLL	Phase Locked Loop
RAM	Random Access Memory
RAS	Row Address Strobe
RGB	Red Green Blue
SDRAM	Synchronous DRAM
SRAM	Static Random Access Memory
TFT	Thin Film Transistor
VESA	Video Electronics Standards Association
VHDL	VHSIC Hardware Description Language
VHSIC	Very Highspeed Integrated Circuit
WPLD	„WASP“ Programmable Logic Device

1. Einleitung

Ausgangspunkt dieser Diplomarbeit ist ein Video-Controller zur Ansteuerung eines Displays für einen Arbeitsplatz im Bereich der Flugsicherung. Er ist maßgeschneidert für eingebettete Systeme (Embedded Systems) mit reduziertem Funktionsumfang, unterstützt die Ausgabe der Bilddaten an den Monitor und führt befehlsgesteuerte Manipulation der Bilddaten durch.

1.1 Bestehendes Videosystem

Ein Video-Controller übernimmt in einem Computersystem (Abbildung 1) die Aufgaben der visuellen Datenausgabe zur Unterstützung des Benutzers. Weiters ist im Video-Controller ein Audio-Controller zur Ausgabe von Signalisierungstöne integriert.

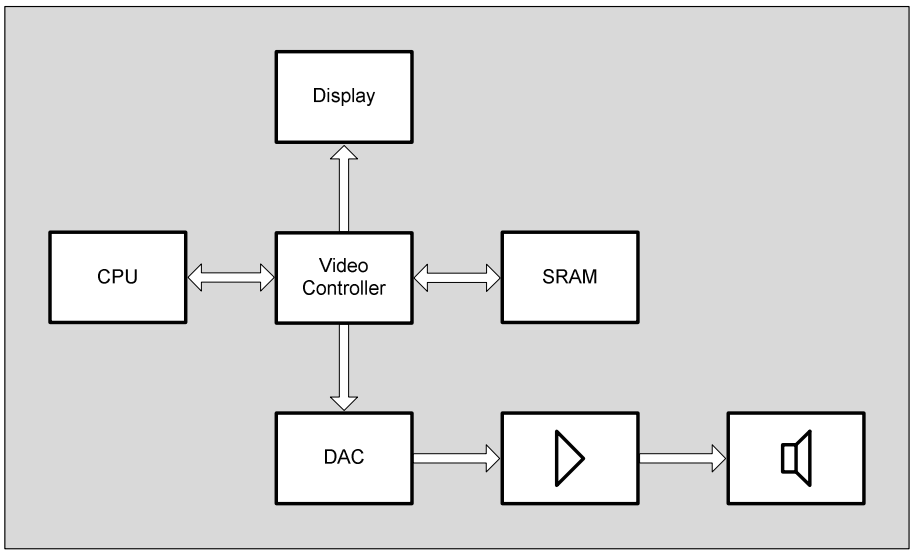

Abbildung 1: Komponenten des bestehenden Videosystems mit SRAM

Der Controller erhält Befehle und Daten von einer Central Processing Unit (CPU). Die Bilddaten werden im Videospeicher, in diesem Fall einem Static Random Access Memory (SRAM), gespeichert. Von dort werden sie vom Video-Controller kontinuierlich ausgelesen und pixelweise an den Monitor zur Darstellung ausgegeben. Der Video-Controller kann befehlsgesteuert den Bildspeicher initialisieren und Zeichenoperationen wie das Kopieren und Einfärben von Rechtecken in der Bildschirmseite ausführen.

Zusätzlich ist im Video-Controller ein Audiocontroller zur Ausgabe von Audiosignalen integriert. Signalisierungstöne aus dem Videospeicher werden mit einem Digital-Analog-Konverter (DAC) in ein analoges Audiosignal umgewandelt und an einen Lautsprecher verstärkt ausgegeben.

Das für den Videospeicher verwendete SRAM ist zwar ein schneller und einfach zu handhabender, aber auch ein teurer Speicher. Im Gegensatz dazu ist ein dynamischer Speicher wesentlich kostengünstiger und gut integrierbar. Da bei gleichen Kosten größere Speichermodule eingesetzt werden könnten, wäre der Speicher, der nicht für den Video-Controller benötigt wird, für weitere Aufgaben anderer Einheiten verfügbar. Dynamische Speicher waren in früheren Zeiten im Vergleich zum statischen Speicher relativ langsame Speicher. Die Adressierung erfolgt im Zeitmultiplex getrennt für Zeile und Spalte. Entsprechend dauerte ein Speicherzugriff etliche Takte verglichen mit einem statischen Speicher, bei dem die Datenausgabe nach einem Takt erfolgen kann (siehe hierfür Kapitel 4). Außerdem müssen bei dynamischen Speichern die Speicherinhalte periodisch aufgefrischt werden, während dieser Zeit kann der Speicher nicht für Schreib-/Lesezugriffe verwendet werden.

Im Laufe der Zeit wurden bei dynamischen Speichern allerdings Maßnahmen getroffen, durch die der Datendurchsatz gesteigert werden kann. So werden beispielsweise ganze Zeilen in Lesepuffern gespeichert, auf die wiederholt schnell zugegriffen werden kann. Allerdings ist bei dynamischen Speichern die erzielbare Geschwindigkeit von der Art der Zugriffe abhängig. Bei wahlfreiem Zugriff, bei dem häufig Zeilen- und Bankwechsel erfolgen, können die Optimierungen nicht genützt werden, bei passenden Zugriffen innerhalb der Zeile können jedoch Transferraten von einigen GB/s erzielt werden.

Gerade aber der Video-Controller liest die Speicherdaten linear aus einem zusammenhängenden Speicherbereich und gibt die Daten an einen Monitor aus. Aus diesem Grund liegt es nahe, diesen Speichertyp im Video-Controller verwenden zu wollen. Es wird ein DDR-SDRAM (Double Data Rate-Synchronous Dynamic RAM) verwendet. Bei diesem weiterentwickelten DRAM erreicht man die doppelte Datenrate, indem sowohl zur steigenden, als auch zur fallenden Taktflanke Daten gelesen oder geschrieben werden. Außerdem ist es einfacher zu handhaben, da Speicherbefehle synchron zum Takt abgesetzt werden und nicht die zeitlichen Beziehungen der Signale untereinander berücksichtigt werden müssen.

Wird bei einem Speicherzugriff auf mehr Daten zugegriffen, kann man die Anzahl an Zugriffen auf den Speicher reduzieren. Durch den Zugriff auf das DDR-SDRAM mit 128 Bit pro Takt können mehrere Datenwörter gleichzeitig aus dem Speicher ausgegeben werden. Da der bestehende Video-Controller auf das SRAM mit einer Datenwortbreite von 16 Bit zugreift, muss die Speicherschnittstelle entsprechend modifiziert werden. Es müssen Schreibbefehle zu 128-Bit-Blöcken gesammelt werden und die Daten bei Lesebefehlen aus den geladenen Blöcken ausgewählt werden.

Da im dynamischen Speicher die gespeicherte Information nach einiger Zeit verloren geht, ist ein zyklisches Auffrischen des Speicherinhaltes, der sogenannte Refresh, erforderlich. Der DDR-SDRAM-Controller frischt den Speicher auf und übersetzt die einfachen Schreib-/Lesebefehle auf den Speicher (Abbildung 2).

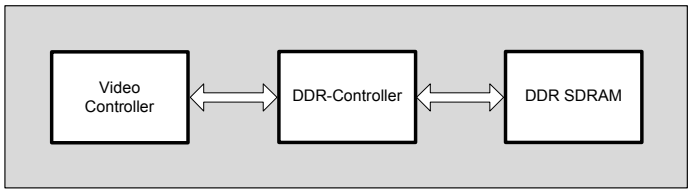

Abbildung 2: DDR-Controller beim Einsatz eines DDR-Speichers

Im Video-Controller gibt es unterschiedliche Arten des Zugriffs auf den Speicher, die sich durch ihre Dringlichkeit unterscheiden. Der Zugriff erfolgt nach einem Zeitscheibenverfahren, wobei dem Lesezugriff der Bildschirmdaten jeder vierte Takt zugeordnet wird, um den kontinuierlichen Datenstrom zum Monitor zu gewährleisten. Allen anderen Zugriffen stehen die drei übrigen Takte zur Verfügung, deren Auswahl nach festgelegten Prioritäten erfolgt.

Im System mit DDR-SDRAM kann nicht davon ausgegangen werden, dass der Speicher für einen Zugriff zur Verfügung steht. Aufgrund von Refresh-Vorgängen kann es sein, dass keine Speicherbefehle abgesetzt werden können. Daher werden die Bilddaten einer Zeile vorgeladen, um sie rechtzeitig für den Bildschirmaufbau bereitstellen zu können.

1.2 Aufgabenstellung

Der bestehende Video-Controller liegt in Form eines VHDL IP-Core zur Verwendung in einem FPGA, einem Field-Programmable Gate Array vor. IP bedeutet in diesem Zusammenhang Intellectual Property und man versteht darunter das geistige Eigentum an einer Einheit mit einer Funktionalität, die kommerziell oder unentgeltlich verwendet werden kann. Die Beschreibung erfolgt in diesem Fall in der Hardwarebeschreibungssprache VHDL. Diese Abkürzung bedeutet VHSIC (Very High Speed Integrated Circuit) Hardware Description Language.

Der Video-Controller soll dahingehend modifiziert werden, dass die Verwendung eines DDR-SDRAMs möglich wird. Zur Verwendung eines solchen Speichers ist ein DDR-SDRAM-Controller notwendig. Hierfür soll ein von Dipl.-Ing. Ingo Wieser für das Institut für Computertechnik in Zusammenarbeit mit Firma Oregano entwickelter DDR-Controller verwendet werden. Dabei ist zu beachten, dass der Video-Controller mit einer Taktfrequenz von 81 MHz und einer Datenwortbreite von 16 Bit auf den Speicher zugreift. Der DDR-Controller arbeitet mit einer Taktfrequenz von 133 MHz und bietet Anwendungen eine Schnittstelle mit einer Datenwortbreite von 128 Bit.

Die Änderungen sollen auf die Schnittstelle zum Speicher begrenzt bleiben und den modularen Aufbau des Video-Controllers unterstützen. Sie soll dem Video-Controller dieselben Ein- und Ausgänge bieten, so dass alle anderen Einheiten wiederverwendet werden können und keine Änderungen erforderlich sind.

1.3 Lösungssuche

Zu Beginn wurde nach bestehenden Video-Controllern gesucht, um möglicherweise existente Lösungen zu verwenden. Aus Kostengründen wurden kommerzielle Video-Controller IP-Cores nicht in Erwägung gezogen. Frei verfügbare IP-Cores sind auf der Plattform „opencores.org“ [1] im Internet erhältlich.

Da mit zunehmender Designgröße Entwicklungsdauer und Kosten stark steigen, kann durch die Verwendung von IP-Cores die Entwicklungszeit verkürzt werden. Bei kommerziellen IP-Cores können zum Teil sehr hohe Kosten anfallen und die Integration in das eigene Design ist schwierig, da der Sourcecode der Hardware in verschlüsselter Form vorliegt und deshalb keine Anpassungen an die eigenen Anforderungen möglich sind. Um vor allem kleinen Entwicklungsteams mit beschränkten Mitteln die Möglichkeit der Kooperation zu bieten, wurde diese Plattform für Open-Source-Hardware geschaffen. Sie wurde von Damjan Lampret gegründet und existiert seit 1999 und hat sich folgende Ziele gesetzt hat:

- frei verfügbare Open-Source-Hardware
- Setzen von Entwicklungsstandards für Open-Source-Hardware
- Entwicklung von Open-Source-Hardware
- Dokumentation von Open-Source-Hardware

Die auf „opencores.org“ angebotenen Video-Controller waren mehrheitlich noch nicht fertig gestellt, die bereits einsatzbereiten wurden auf ihre Eignung überprüft:

Monochrome Text-Mode VGA Video Display Adapter

Die Bildschirmauflösung des Video-Controllers (Abbildung 3) beträgt 640 x 480 Bildpunkte, entsprechend werden 80 x 40 Zeichen mit einer Größe von 8 x 12 Pixel dargestellt.

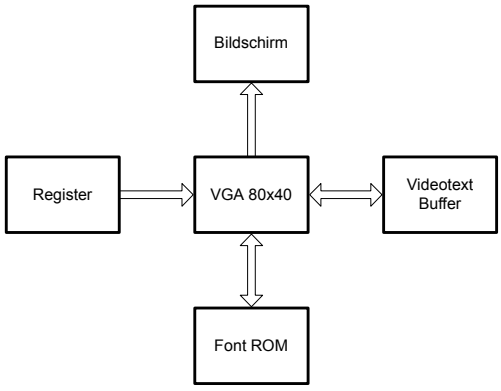

Abbildung 3: Monochromer Text-Mode-Video-Controller VGA 80x40

Die Zeichen sind im sogenannten Font ROM (Read Only Memory) gespeichert, in dem für jedes Pixel im Zeichen ein Bit vorgesehen ist. Bei der Ausgabe am Bildschirm werden dann für jede Bildschirmzei-

le die entsprechenden Zeilen der Zeichen ausgegeben. Es handelt sich, wie aus dem Namen ersichtlich, um einen monochromen Video-Controller. Er unterstützt zwar nur acht Farben, wobei allerdings nur eine als Farbe der Zeichendarstellung ausgewählt werden kann.

In der Registereinheit können die Betriebsparameter des Video-Controllers wie z. B. Textfarbe oder das Cursorverhalten eingestellt werden. Der Videotextbuffer ist eine Dualport-RAM (Random Access Memory), ein Speicher der mit einem Schreib- und einem Leseanschluss ausgeführt ist. Der Video-Controller liest lediglich aus dem Speicher Daten aus, er kümmert sich nicht um das Befüllen des Speichers mit Bilddaten, dies muss von zusätzlicher Logik im FPGA über den Schreibanschluss des Dualport-RAMs erfolgen.

VGA/LCD Controller

Dieser Video-Controller ist ein Grafik-Video-Controller der Farbdarstellung, 256 Graustufen sowie 8-Bit-Paletten Farbdarstellung unterstützt.

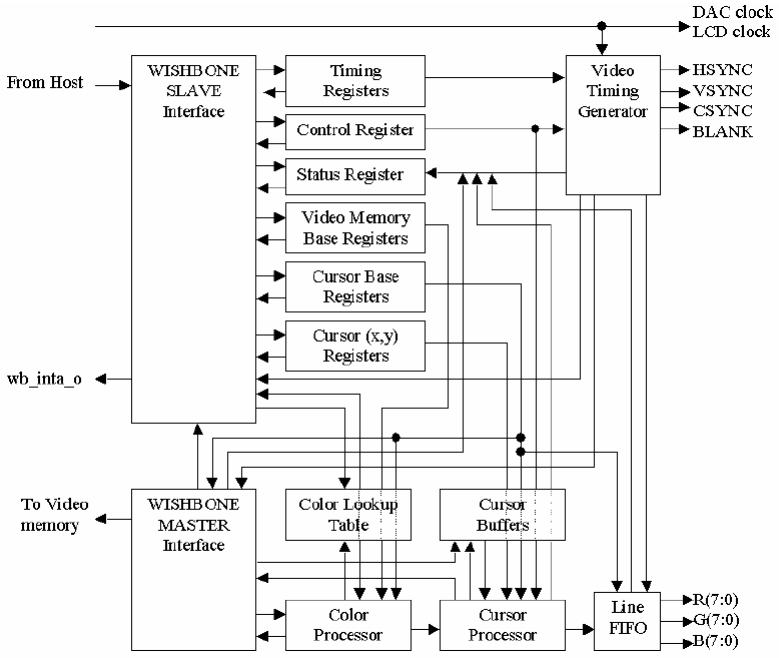

Abbildung 4: Struktur VGA/LCD-Controller [HER2003]

Verfügbare Auflösungen sind 320 x 240, 640 x 480 und 800 x 600 Bildpunkte, die Bildwiederholfrequenzen liegen bei 56, 60 oder 72 Hz. Die Daten können aus einem SRAM oder eine SDRAM übernommen werden, für SDRAMs muss der Controller nur unterschiedlich konfiguriert werden.

Wie in Abbildung 4 ersichtlich werden die Bilddaten über eine Wishbone-Schnittstelle aus dem Speicher angefordert. In der Einheit für die Farbverarbeitung werden die Bilddaten gegebenenfalls über eine Palettentabelle in Echtfarbendarstellung umgewandelt. In der folgenden Cursor-Einheit erfolgt die Umschaltung zwischen der Darstellung der Bildpunkte oder eines Cursors.

Abschließend werden die darzustellenden Bildpunkte über ein Zeilen-FIFO (First In First Out) an den Monitor ausgegeben. Weiters erzeugt der Video-Controller das Taktsignal und die Synchronisationssig-

nale für den Monitor. Über eine weitere Wishbone-Schnittstelle werden die Register des Video-Controllers über eine angebundene CPU konfiguriert.

Auch dieser Video-Controller liest den Bildspeicher lediglich aus, die Bilddaten müssen ebenfalls von einer zusätzlichen Logik generiert werden.

Wishbone Monitor Controller

Der Wishbone Monitor Controller ist ein Grafik-Video-Controller, bei dem ein SRAM als Videospeicher verwendet wird. Intern arbeitet der Video-Controller mit 8-Bit-Farbpalettendarstellung. Zur Darstellung am Monitor erfolgt nach der zentralen Video-Controller-Logik, in der aus dem Speicher die Farbpalettenwerte ausgelesen und Synchronisationssignale erzeugt werden, die Umwandlung der Farbpalettenwerte über eine Palettentabelle in RGB-Farbwerte. Die Kommunikation mit einer CPU erfolgt über eine Wishbone-Schnittstelle, ebenso ist die Schnittstelle zum Speicher über Wishbone ausgeführt.

Sämtlichen aufgelisteten Video-Controller ist gemeinsam, dass sie lediglich die Bilddaten aus dem Videospeicher auslesen, am Monitor ausgeben und die Synchronisationssignale erzeugen. Während der bestehende Video-Controller von Oregano Systems speziell für die Anforderungen eines Arbeitsplatzes in der Flugsicherung maßgeschneidert wurde. Zusätzlich zur reinen Bilddarstellung am Monitor werden auch Zeichenoperationen auf die Bildspeicherinhalte unterstützt und über einen Audio-Controller können Signalisierungstöne ausgegeben werden. Diese zusätzliche Logik muss bei der Verwendung eines alternativen Video-Controllers an anderer Stelle, außerhalb des Video-Controllers, hinzugefügt werden. Weiters ergäbe sich bei der Verwendung eines alternativen Video-Controllers, dass die Zeichenoperationen und die Bilddarstellung nicht mehr gekoppelt erfolgen könnten und sich somit negative Auswirkungen auf die Darstellung (Flimmerfreiheit) sowie am Speicherdatendurchsatz für die Bilddatenausgabe am Monitor ergeben könnten.

Aus diesen Gründen wurde entschieden, den bestehenden Video-Controller für den Zugriff auf ein DDR-SDRAM zu modifizieren. Dadurch sollen die Module des Video-Controllers übernommen werden, lediglich die Schnittstelle zum Speicher muss für die Verwendung eines DDR-SDRAMs modifiziert werden.

2. Bilddarstellung in Computersystemen

Um die Arbeitsweise eines Video-Controllers und seine Aufgaben zu verdeutlichen, wird in diesem Kapitel die Bilddarstellung in Computersystemen kurz beschrieben.

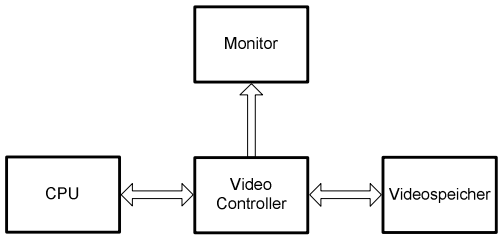

Abbildung 5: Bilddarstellung in Computersystemen

Das Prinzip der Bilddarstellung in Computersystemen ist in Abbildung 5 dargestellt. Die CPU (Central Processing Unit) erteilt dem Video-Controller oder Grafikadapter Grafikbefehle. Dieser manipuliert aufgrund dieser Grafikoperationen die im Videospeicher befindlichen Bilddaten. Der Video-Controller liest die Daten einer Bildschirmseite zyklisch aus dem Videospeicher und gibt diese pixelweise an einen Monitor aus.

Die Darstellung der Daten auf einem Bildschirm orientiert sich an der Arbeitsweise eines Monitors mit Kathodenstrahlröhre. Die prinzipielle Anordnung ist in Abbildung 6 dargestellt.

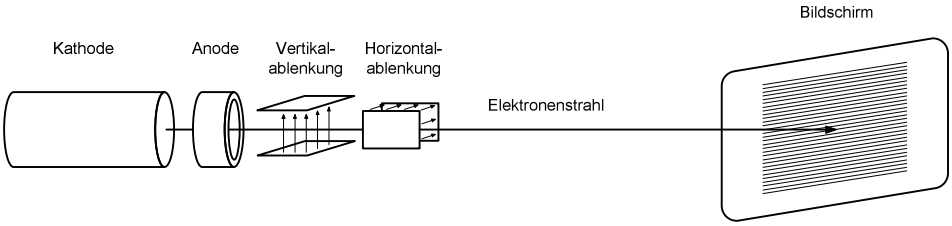

Abbildung 6: Prinzip der Kathodenstrahlröhre [HPM00]

Die aus der Kathode austretenden Elektronen werden vom elektrischen Feld der Anode beschleunigt und vom elektrischen Feld zwischen den Platten der Vertikal- und Horizontalablenkung gesteuert. Danach trifft der Elektronenstrahl auf einen mit fluoreszierendem Material beschichteten Bildschirm. Dort,

wo die Elektronen auftreffen, wird ein Aufleuchten verursacht, dessen Helligkeit von der Intensität des Elektronenstrahls abhängt.

Der Bildschirm ist in Zeilen unterteilt, die punktweise beschrieben werden. Am Ende einer Zeile, wie in Abbildung 7 dargestellt, bewirkt der Horizontalrücklauf die Positionierung des Elektronenstrahls an den Anfang der nächsten Zeile. Am Ende der letzten Zeile bewirkt der Vertikalrücklauf die Rückkehr an den Anfang der ersten Zeile und die Darstellung des Bildes beginnt von Neuem.

Durch ständige Wiederholung des Bildes, dem Nachleuchten der fluoreszierenden Schicht und der Trägheit des Auges entsteht am Monitor ein stehendes Bild.

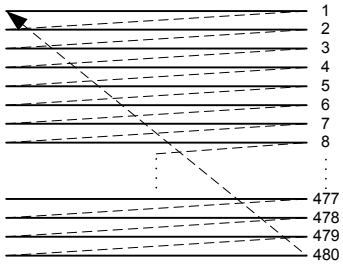

Abbildung 7: Zeilenweiser Bildschirmaufbau (Non-Interlaced)

Farbpunkte werden mit drei Elektronenstrahlen und drei unterschiedlichen fluoreszierenden Schichten gebildet. Je ein Strahl repräsentiert eine der Grundfarben Rot, Grün und Blau (RGB). Eine Farbe wird durch additive Farbmischung gebildet, somit besteht ein Pixel eigentlich aus drei Punkten. Es ergibt sich zum Beispiel ein weißer Bildpunkt, wenn alle drei Elektronenstrahlen mit gleicher Intensität am Bildschirm auftreffen.

Die Modulation des Elektronenstrahls und die Rücklaufsignale müssen synchronisiert werden, sodass der Horizontalrücklauf genau am Ende einer Zeile und der Vertikalrücklauf nach der Darstellung der letzten Zeile erfolgt. Diese Aufgaben übernimmt der Grafikadapter, er erzeugt die dafür notwendigen horizontalen und vertikalen Synchronisierungssignale HSYNC und VSYNC. In Abbildung 8 ist der Verlauf des HSYNC-Signals dargestellt. Das Signal HSYNC ist low-aktiv und wird nach dem Front-Porch-Bereich (FP) für die Dauer von SYNC-Takten aktiviert. Anschließend folgt der Back-Porch-Bereich (BP) und danach die Bilddatenausgabe an den Monitor im aktiven Bildbereich.

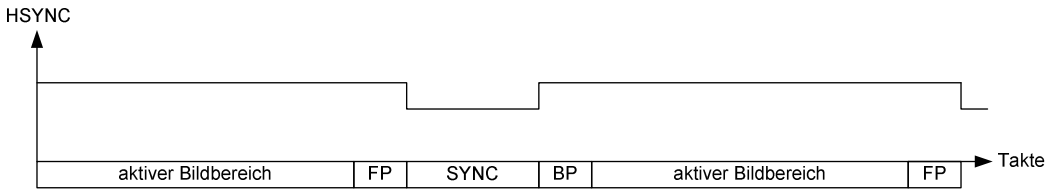

Abbildung 8: Zeilendarstellung

Das Signal VSYNC wird in ähnlicher Weise erzeugt, ebenfalls low-aktive wird es nach dem Ende der letzten Zeile aktiviert und vor Beginn der ersten Zeile deaktiviert.

Ab dem Beginn der Darstellung einer Zeile im aktiven Bereich werden mit jedem Pixeltakt die RGB-Farbwerte eines Bildpunktes an den Bildschirm ausgegeben. Nach diesem Bereich, in der sogenannten Austastlücke wird der RGB-Wert auf Schwarz (0-0-0) gesetzt. Die Zeit zwischen Anfang der Austastlücke und dem Beginn der Aktivierung des Synchronisationssignals wird als Front Porch, die Zeit nach dem HSYNC-Puls bis zum Ende der Austastlücke als Back Porch bezeichnet.

Tabelle 1:Timingparameter für 800 x 600 Bildpunkte und 60 Hz (VESA)

	Pixeltakt	40 MHZ
horizontal	Aktiver Bereich	800 Pixel
	Front Porch	40 Pixel
	Synchronisationsimpuls	128 Pixel
	Back Porch	88 Pixel
vertikal	Aktiver Bereich	600 Zeilen
	Front Porch	1 Zeile
	Synchronisationsimpuls	4 Zeilen
	Back Porch	23 Zeilen

Tabelle 1 zeigt die Timingparameter für eine Bildschirmauflösung von 800 x 600 Bildpunkten und einer Bildwiederholfrequenz von 60 Hz nach VESA (Video Electronics Standards Association).

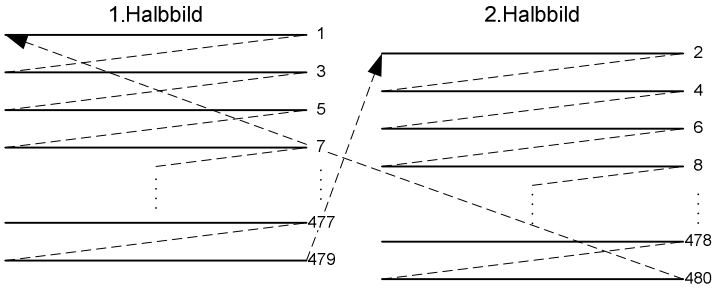

Abbildung 9: Interlaced-Darstellung

In diesem Zusammenhang ist auch das in Abbildung 9 dargestellte Zeilensprungverfahren, auch Interlaced-Modus genannt, zu erwähnen. Bei diesem Verfahren werden zuerst alle ungeraden, dann die geraden Bildschirmzeilen geschrieben. Die Bildwiederholfrequenz bleibt dabei gleich, es werden allerdings nur Halbbilder dargestellt. Da der Vertikalrücklauf mit derselben Frequenz arbeitet aber nur halb so viele Zeilen dargestellt werden, halbiert sich die notwendige Videobandbreite. Für eine gute Bildqualität ist es aber notwendig, dass beim nachfolgenden Bild der Elektronenstrahl exakt zwischen den im vorhergehenden Bild dargestellten Zeilen verläuft. Da diese Justierung schwer zu erreichen ist, ist die Bildqualität meist schlechter als im Non-Interlaced-Modus.

3. Video-Controller

In diesem Kapitel wird der in der Arbeit zu modifizierende Video-Controller analysiert. Es werden seine Verbindungen zu anderen Einheiten des Computersystems und die Funktionen, die er zu erfüllen hat, erläutert.

3.1 Aufbau

Die Aufgabe des Video-Controllers ist die Ausgabe der im Videospeicher enthaltenen Daten einer Bildschirmseite auf einem Monitor. In Abbildung 10 sieht man das Blockschaltbild des Video-Controllers und seine Verbindungen zu den externen Einheiten.

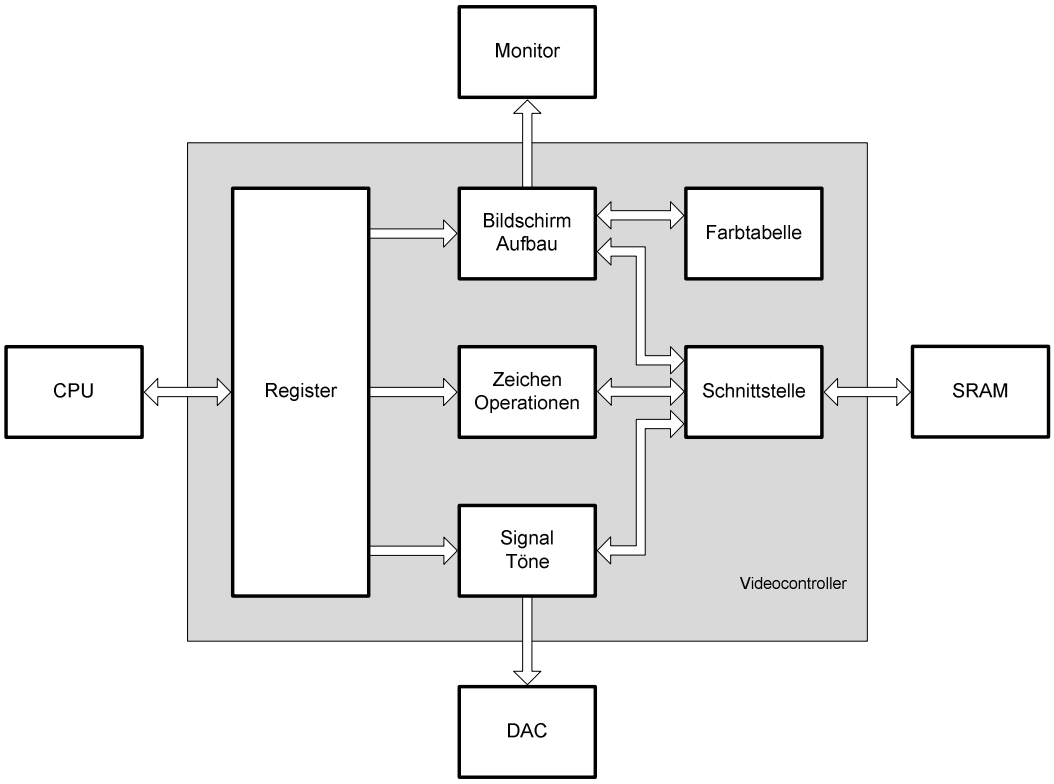

Abbildung 10: Struktur des Video-Controllers

Die Bilddaten sind in einem externen SRAM gespeichert, wobei die Farbinformation in Form einer Farbtabelle vorliegt. Auf die Bildschirmdaten können Zeichenoperationen, gesteuert von einer CPU, durchgeführt werden, wie das Kopieren und Einfärben von rechteckigen Bildausschnitten. Außerdem unterstützt der Video-Controller die Ausgabe von Signaltönen, die ebenfalls im Bildspeicher abgelegt sind.

Von der CPU können Grafikbefehle an den Video-Controller abgesetzt werden, indem das Steuerregister der Einheit Register geschrieben und Parameter in weiteren Registern gesetzt werden. Die Einheiten für den Bildschirmaufbau, die Zeichenoperationen und die Signaltöne erkennen die für sie bestimmten Anweisungen und reagieren darauf.

Das Modul Zeichenoperationen führt Zeichen- und Kopierbefehle aus. Dabei werden Bilddaten aus dem externen Videospeicher, einem SRAM, gelesen, manipuliert und zurück in den Speicher geschrieben.

Zur Darstellung der Bilder am Monitor liest der Video-Controller ständig die Daten einer Bildschirmseite aus dem Bildspeicher und gibt sie pixelweise am Monitor aus. Die Auflösung einer Bildschirmseite beträgt 800 mal 600 Bildpunkte. Es werden 256 Farben unterstützt, wobei eine Farbpalette mit entsprechend 256 Einträgen von 12 Bit verwendet wird. Daraus werden die 3 x 6-Bit-RGB (Red Green Blue) ermittelt. Im Bildspeicher wird nur der 8-Bit-Index der zugehörigen Farbe in der Farbtabelle gespeichert. Dies führt zur Reduktion des benötigten Speichers, weil nur eine geringere Anzahl an Farben eines größeren Farbraumes unterstützt wird.

Zusätzlich ist im Video-Controller ein Sound-Controller integriert, der ebenfalls den Videospeicher benützt. Nach einem Befehl zur Wiedergabe eines gespeicherten Signaltons wird der entsprechende Sound aus dem Speicher geladen und an einen Digital Analog Converter (DAC) seriell mit 8 kHz ausgegeben. Das Audio-Signal wird anschließend verstärkt und über einen Lautsprecher ausgegeben.

Die Speicherschnittstelle dient dazu, den Zugriff auf den Speicher durchzuführen, es müssen die Prioritäten beim Zugriff und entsprechend Haltesignale gesetzt werden. Weiters müssen die Daten an die lesenden Einheiten ausgegeben werden, bzw. beim Schreiben die Daten an den Speicher ausgegeben werden.

3.2 Videospeicher

Die Aufteilung des Videospeichers in verschiedene Bereiche ist in Abbildung 11 dargestellt. Es werden zwei Bildschirmseiten vom Video-Controller unterstützt, zwischen denen umgeschaltet werden kann. Die Basisadresse VGA_BASE_ADDRESS definiert den Anfang der Bilddaten der ersten Bildschirmseite. Die Konstante PIXEL in der Zeichnung ergibt sich aus Anzahl der Bildpunkte multipliziert mit 8 Bit:

$$PIXEL = 800 * 600 * 8\ Bit = 468{,}75\ kB$$

Die Anfangsadresse des zweiten Bildbereiches ergibt sich dann indem man zu der Basisadresse VGA_BASE_ADDRESS die Konstante PIXEL addiert.

Unterhalb der Bilddaten ab der Adresse VGA_BASE_ADDRESS befinden sich 16 Signalisierungstöne zu je 256 Byte im Videospeicher.

Der Bereich unterhalb der Daten der Signalisierungstöne, die sogenannte Scratch Pad Area, wird von den Bildoperationen dazu verwendet, die Daten für Bildtransfers aufzunehmen.

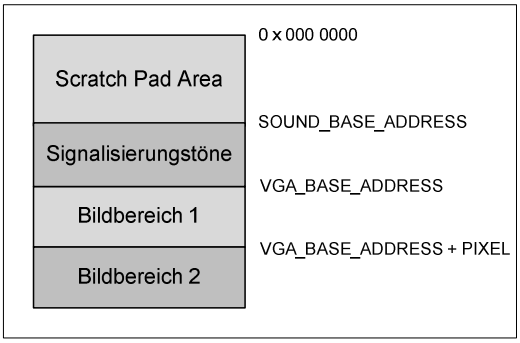

Abbildung 11: Speicheraufteilung des Videospeichers

3.3 Befehlssatz

Im Folgenden werden die einzelnen vom Video-Controller unterstützten Befehle beschrieben. Diese werden von der CPU an den Video-Controller abgesetzt, indem das Register VGA_CONTROL der Registereinheit entsprechend geschrieben wird (siehe dazu das Kapitel 3.4.1 Registereinheit).

Einige der nachfolgend beschriebenen Befehle können optional flimmerfrei durchgeführt werden. Bei der Manipulation der Bildschirmseiten im Speicher kann es vorkommen, dass die Zeichenoperationseinheit die Bildschirmausgabe überholt. Am Monitor erfolgt dann innerhalb einer Bildschirmseite der Wechsel vom neuen zum alten Bild. Dies kann durch ein Flimmern wahrgenommen werden. Um dies zu verhindern, wird bei der flimmerfreien Durchführung eines Befehls dessen Ausführung solange verzögert, bis eine Bildschirmseite fertig dargestellt wurde. Zu diesem Zeitpunkt startet die Einheit der Zeichenoperationen mit der Ausführung des Befehls. Da die Zeichenoperationen schneller als die Bildschirmausgabe durchgeführt werden, kann es zu keinem Flimmern kommen.

Bei sämtlichen Zeichenbefehlen wird der Bildausschnitt, auf den dieser Befehl ausgeführt werden soll, durch die Angabe des linken, oberen Punktes sowie Breite und Höhe definiert. Dabei wird der linke, obere Punkt als Index des Punktes in einem eindimensionalen Feld, in das das zweidimensionale Bild transformiert wird, gemäß $P(x, y) \mapsto (Breite\ der\ Zeile \times y\ + x)$ angegeben.

Bildumschaltfunktion

Das Umschalten zwischen den im Videospeicher gespeicherten Seiten erfolgt flimmerfrei am Ende der letzten Zeile des dargestellten Bildes in der Austastpause.

Kopieren von rechteckigen Bildausschnitten

Es werden rechteckige Bildausschnitte einer Bildschirmseite kopiert und an einer gewählten Position eingefügt. Das Kopieren erfolgt nicht flimmerfrei.

Zeichnen von Rechtecken

Unter Angabe der Zielposition und einer Farbe wird ein Rechteck gezeichnet. Linien ergeben sich als Grenzfälle der Rechtecke. Bei einer angegebenen Höhe von einem Pixel wird eine horizontale, bei einer Weite von einem Pixel wird eine vertikale Linie am Bildschirm gezeichnet. Auch hier erfolgt das Zeichnen ohne Verriegelung, d. h. nicht flimmerfrei.

Kopieren und Einfärben von rechteckigen Bildausschnitten

Durch Angabe der Quell- und Zielposition können rechteckige Bildbereiche kopiert und mit einer definierten Farbe eingefärbt werden. Dabei werden alle Bildpunkte, deren Farbe einer weiteren angegebenen Farbe entsprechen, als transparent betrachtet und nicht kopiert. Der Befehl erfolgt nicht flimmerfrei.

Flimmerfreies Kopieren von rechteckigen Bildausschnitten

Rechteckige Bildausschnitte werden flimmerfrei kopiert.

Lineares Kopieren

Rechteckige Bildausschnitte können in der Scatch-Pad-Area im unteren Bereich des Videospeichers linear, d. h. nicht zweidimensional, abgelegt werden. Die Parameter Breite und Höhe des rechteckigen Bildausschnittes gelten dann nur für den Zielbereich, aus dem Quellbereich wird fortlaufend gelesen. „Lineares Kopieren“ kann optional mit den Befehlen „Kopieren“, „Kopieren und Einfärben“ und „flimmerfreies Kopieren“ eingesetzt werden.

Kopieren von rechteckigen Bildausschnitten mit Transparenz

Beim Kopieren des rechteckigen Bildausschnittes werden nur jene Pixel kopiert, die nicht der angegebenen Farbe entsprechen. Der Kopiervorgang erfolgt nicht flimmerfrei.

Kopieren und Einfärben von rechteckigen Bildausschnitten mit definierter Farbe

Nur die Bildpunkte eines rechteckigen Bildschirmausschnittes die einer angegebenen Farbe entsprechen, werden kopiert und mit einer definierten Farbe eingefärbt. Alle anderen Pixel im Zielbereich bleiben unverändert. Entsprechen einander Quell- und Zielbereich, so können einzelne Farben umgefärbt werden.

3.4 Einheiten des Video-Controllers

Als nächstes folgt die Beschreibung der einzelnen Module des Video-Controllers.

3.4.1 Registereinheit

Die Registereinheit (Abbildung 12) stellt eine Vielzahl von Registern zur Verfügung, auf die die CPU zugreifen und so den Video-Controller konfigurieren kann.

Um diese Register zu schreiben, erhält der Video-Controller von der CPU Schreibanweisungen mit der entsprechenden Registeradresse. Einige Register können mit Leseanweisungen der CPU ausgelesen werden, unter anderem auch das Readback-Register, mit dessen Hilfe man den Inhalt des Videospeichers auslesen kann. Über die Registereinheit ist es der CPU auch möglich, die Farbtabelle zu schreiben.

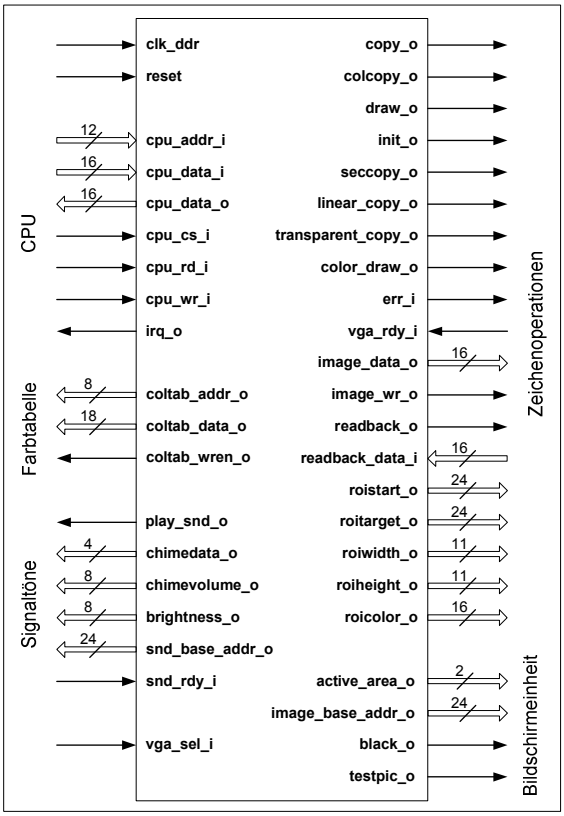

Abbildung 12: Ports der Registereinheit

Die Einheit der Signaltöne erhält von der Registereinheit die Information, wann (*play_snd_o*), welcher (*chime_data_o*) der Signalisierungstöne, in welcher Lautstärke (*chime_volume_o*) wiedergegeben werden soll. Außerdem legt die Basisadresse (*snd_base_addr_o*) der Signalisierungstöne jene Adresse fest, ab der die Sound-Daten im Videospeicher abgelegt sind. Mit dem Ausgabeport *brightness_o* wird die Helligkeit des Displays eingestellt.

Der Einheit Zeichenoperationen werden alle zur Durchführung der Befehle notwendigen Parameter übergeben. Über *readback_o* können 16-Bit-Daten aus dem Videospeicher angefordert werden.

Für die Bildschirmeinheit kann angegeben werden, welche der beiden Bildschirmseiten dargestellt werden soll (*active_area_o*), unter welcher Adresse die Bildschirmdaten im Videospeicher abgelegt sind (*image_base_addr_o*), ob ein Testbild (*testpic_o*) oder ein einfärbiger Bildschirm (*black_o*) ausgegeben werden soll.

Im Weiteren folgt eine Beschreibung von Funktion und Verwendung der wichtigsten Register.

VGA_CONTROL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	read back				color draw	active area		trans copy	lin copy	draw	sec copy	color copy	copy	init	black

Abbildung 13: Register VGA_Control

Mit dem Schreiben des Registers VGA_CONTROL durch die CPU können Befehle durch Setzen der einzelnen Bits erteilt werden. In

Abbildung 13 sind die Bits des Registers, darunter ihre Bedeutung angeführt. In Tabelle 2 folgt eine Beschreibung der Aktionen, die beim Setzen der jeweiligen Bits ausgelöst werden können.

Tabelle 2: Bedeutung der Registerstellen von CPU_CONTROL

Bit	Bedeutung
0	in Register ROICOLOR angegebene Farbe für alle Pixel ausgegeben
1	Transferadresse auf den Ausgangswert 0x000000 setzen
2	rechteckigen Bildausschnitt kopieren
3	rechteckigen Bildausschnitt kopieren und einfärben
4	Rechteck flimmerfrei kopieren
5	Rechteck zeichnen
6	Quellbereich ist linear und nicht zweidimensional
7	rechteckigen Bildausschnitt mit Transparenz kopieren
8 bis 9	aktiver Bildbereich, gibt an, welche Bildschirmseite aus dem Videospeicher dargestellt wird
10	Rechteck kopieren und mit definierter Farbe einfärben
14	Bildspeicher auslesen

VGA_STATUS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									vga sel		snd rdy			err	vga rdy

Abbildung 14: Register VGA_STATUS

Abbildung 14 gibt den aktuellen internen Zustand des Video-Controllers wieder. An diesem Register kann man z. B. erkennen, ob der Video-Controller für einen Befehl bereit ist oder noch mit der Ausführung des vorangegangenen Befehls beschäftigt ist. Alle weiteren Statusanzeigen sind aus Tabelle 3 zu entnehmen.

Tabelle 3: Bedeutung der Registerstellen von CPU_STATUS

Bit	Bedeutung
0	Aktiv gesetzt, wenn der Video-Controller wieder für neue Befehle bereit ist
1	Der Video-Controller im Error-Zustand
4	Wird aktiviert, wenn weniger als acht Sound Samples zur Wiedergabe vorhanden sind. Danach kann ein Interrupt ausgelöst werden, und die CPU hat 1 ms Zeit, einen Sound-Befehl abzusetzen, um einen kontinuierlichen Sound zu erreichen.
6	Gibt Auskunft über den angeschlossenen Displaytyp, wobei ‚0' ein herkömmliches, ‚1' ein High-Bright-Display kennzeichnet.

VGA_INTSTATUS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											snd rdy			err	vga rdy

Abbildung 15: Register VGA_INTSTATUS

Das Register in Abbildung 15 zeigt den Interrupt-Status des Video-Controllers an. Bit 4 zeigt z. B. nach einem Sound-Befehl an, dass der Video-Controller für einen neuen Audio-Befehl bereit ist. Bei der CPU kann damit ein Interrupt ausgelöst werden und ein neuer Befehl zur Ausgabe des nächsten Sounds kann erfolgen. Nach dem Auslesen wird das Interrupt-Register gelöscht. Die Bedeutung aller verwendeten Registerbits sind aus Tabelle 4 zu entnehmen.

Tabelle 4: Bedeutung der Registerstellen von VGA_INTSTATUS

Bit	Bedeutung
0	Video-Controller für neue Befehle bereit
1	Fehlersignal
4	Sound-Einheit für neue Befehle bereit

VGA_COLOR_TABLE

Dieses Register dient als Farbumsetzungstabelle mit 256 Einträgen zu je 12 Bit für die Umwandlung der 8-Bit-Farbdarstellung in 3 x 6 Bit-RGB-Farbdarstellung. Es werden jeweils nur die vier MSBs (Most Significant Bit) jedes Farbwertes aus der Tabelle entnommen. Die vier MSBs eines Tabelleneintrages werden für den Helligkeitswert der Farbe Rot (R-Wert) verwendet, die vier LSBs (Least Significant Bit) für die Helligkeit von Blau (B-Wert) und die vier Bits dazwischen für die Helligkeit von Grün (G-Wert). Für die zwei LSBs jedes Farbwertes werden die beiden MSBs verwendet. Für die Anwendung des Video-Controllers ist diese Auflösung ausreichend und es lässt sich Speicher für Register einsparen, es ist eine Reduzierung der Farbtabelle auf 256 x 12 Bit möglich.

Weitere verwendete Register

In Tabelle 5 sind weitere Register und ihre Verwendung aufgelistet die der Video-Controller zur Durchführung seiner Aufgabe benötigt. Register wie z. B. Beispiel das Register VGA_BASE_ADDRESS die 24 Bit breit sind, bestehen aus dem Register VGA_BASE_ADDRESS_L mit den 16 unteren Bits und aus VGA_BASE_ADDE_H mit den oberen 8 Bits.

Tabelle 5: Weitere vom Video-Controller verwendete Register

Register	Breite in Bit	Verwendung
VGA_BASE_ADDRESS	24	Basisadresse für den ersten Bildbereich im Speicher
VGA_DATA	16	Transferregister zum Beschreiben des Speichers
ALLROWS	16	Anzahl aller Zeilen pro Bild, sichtbare und unsichtbare
PREROWS	10	Anzahl der Zeilen vor der ersten sichtbaren Zeile
ROWS	10	Anzahl der aktiven Zeilen pro Bild
VWIDTH	8	Breite des VSYNC-Signals in Zeilen
ALLCOLUMNS	16	Anzahl aller Spalten pro Bild, sichtbare und unsichtbare
PRECOLUMNS	10	Anzahl der nicht sichtbaren Spalten vor der ersten sichtbaren Spalte
COLUMNS	11	Anzahl der aktiven Spalten pro Zeile
HWIDTH	8	Breite des HSYNC-Signals in Pixel
PIXEL	24	Anzahl der aktiven Bildpunkte pro Bildschirmseite
ROISTART	24	Linke obere Ecke des rechteckigen Quellbereichs
ROITARGET	24	Rechte obere Ecke des rechteckigen Zielbereichs
ROIWIDTH	11	Breite des rechteckigen Bildausschnittes
ROIHEIGHT	11	Breite des rechteckigen Bildausschnittes
BRIGHTNESS	8	Helligkeit des Display
SOUND_BASE_ADDRESS	24	Basisadresse der Signalisierungstöne im Videospeicher
CHIMEDATA	4	Auswahl eines der 16 Signalisierungstöne
CHIMEVOLUME	8	Lautstärke der Signaliierungstöne die über ein Potentiometer eingestellt wird
READBACK	16	Transferregister zum Auslesen des Bildspeichers
ROICOLOR	16	Farbregister mit SOURCECOLOR im High-Byte und TARGETCOLOR im Low-Byte

3.4.2 Einheit Bildschirmaufbau

Diese Einheit hat die Aufgabe, die auf dem Bildschirm darzustellenden Daten aus dem Videospeicher zu lesen, mittels Farbtabelle in die für das TFT-Display notwendige 3 x 6-Bit RGB Farbdarstellung umzuwandeln und am Bildschirm auszugeben. Außerdem hat diese Einheit die Aufgabe, die Synchronisierungssignale HSYNC und VSYNC für den Bildschirm zu erzeugen.

In Abbildung 16 sind die Ein- und Ausgänge zu den verbundenen Komponenten dargestellt. Das Register liefert die Information unter welcher Adresse die Bilddaten im Videospeicher gespeichert sind. Aus der Basisadresse (*base_addr_i*) und der Angabe, welches Bild (*active_area_i*) aus dem Speicher dargestellt werden soll, generiert diese Einheit die Adressen der Bilddaten. Außerdem kann ein Testbild (*testpic_i*) oder alle Pixel in einer Farbe (*black_i* und *roicolor_i*) ausgegeben werden.

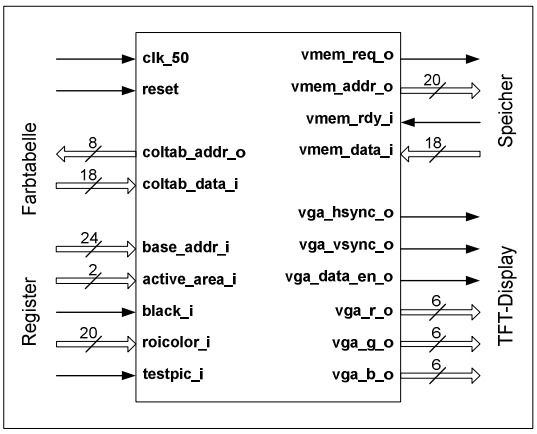

Abbildung 16: Entity für den Bildschirmaufbau

Aus dem Bildspeicher werden jeweils zwei Bildpunkte (16 Bit) gelesen und einzeln mit der Farbtabelle in das gewünschte Format umgewandelt. Der Farbwert eines Punktes adressiert eine Zeile der Farbtabelle unter der der Farbwert eingetragen ist. Danach folgt die Ausgabe des Bildpunktes an das TFT-Display im Pixeltakt, welcher der halben Taktfrequenz entspricht.

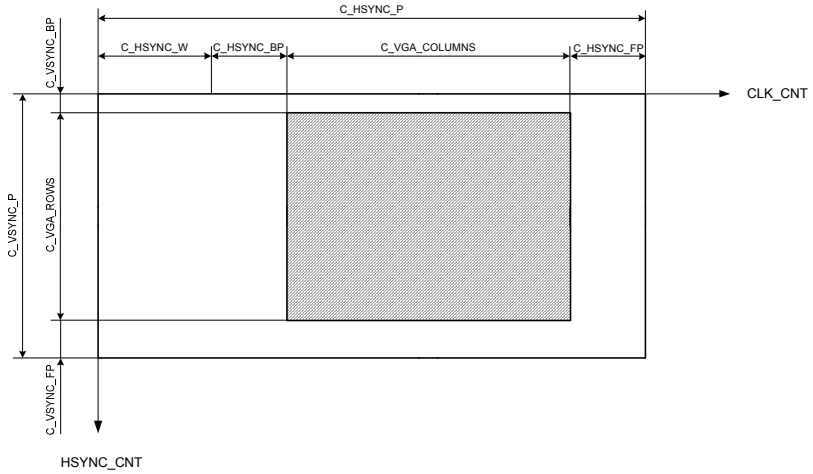

Abbildung 17: Aktiver Bildbereich

Der aktive Bildbereich ist in Abbildung 17 zu sehen, dies ist keine maßstabsgetreue Darstellung, weil die nichtaktiven Bereiche sehr klein gegenüber dem aktiven Bildbereich sind. Im schraffierten Bereich werden Bilddaten im Pixeltakt ausgegeben. Dieser Bereich wird durch zwei Zähler definiert. Horizontal der Zähler CLK_CNT, der im Pixeltakt inkrementiert wird. Aus seinem Zählerstand werden der aktive Bereich einer Zeile, in welchem Bilddaten an den Monitor ausgegeben werden, und das HSYNC-Signal abgeleitet.

In der vertikalen Richtung zählt der Zähler HSYNC_CNT die durchlaufenen Zeilen des Zählers CLK_CNT. Aus seinem Zählerstand wird ermittelt, in welchen Zeilen eine Datenausgabe erfolgt und wann das Synchronisationssignal VSYNC erzeugt wird.

Der aktive Bereich ergibt sich aus der Überschneidung des aktiven Bereiches einer Zeile und den Zeilen, in denen eine Datenausgabe erfolgt. Die zugehörigen Konstanten sind aus Tabelle 6 zu entnehmen.

Tabelle 6: Konstanten der Synchronisationssignale

Konstante	aktueller Wert	Beschreibung
C_HSYNC_W	128	Breite des HSYNC-Impulses
C_HSYNC_P	1024	Periode des HSYNC Signals
C_HSYNC_BP	32	Back Porch Weite
C_VSYNC_W	4	Breite des VSYNC-Impulses
C_VSYNC_P	640	Periode des VSYNC Signals
C_VSYNC_BP	34	VSYNC Back Porch Weite

In Abbildung 18 ist die Erzeugung des Signals HSYNC aus dem Zählerstand von CLK_CNT gezeigt.

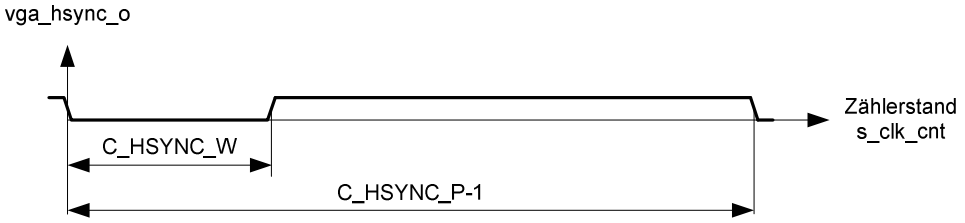

Abbildung 18: Bildschirmsynchronisationssignal HSYNC

Daraus ergibt sich für eine Zeile die Periodendauer von

$$T_{Zeile} = \frac{C_HSYNC_P}{fclk/2} = \frac{2*1024}{81{,}92\ MHz} = 25\ \mu s$$

In Abbildung 19 ist das Signal VSYNC das aus dem Zählerstand des Zählers HSYNC_CNT abgeleitet wird, veranschaulicht.

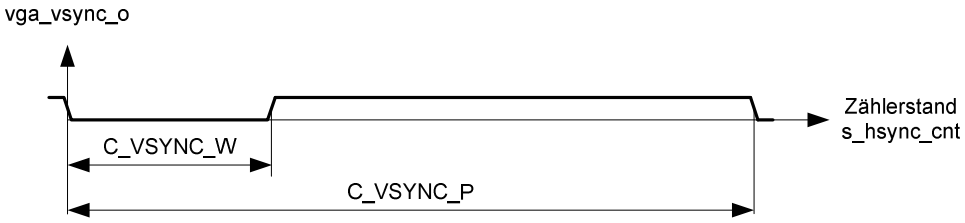

Abbildung 19: Bildschirmsynchronisationssignal VSYNC

Es ergibt sich somit die Dauer der Darstellung eines Bildes von

$$T = T_{Zeile} * C_VSYNC_P = 25\ \mu s * 640 = 16\ ms$$

Dies ergibt eine Bildwiederholfrequenz von

$$f = 1/T_{Bild} = 1/16\ ms = 62{,}5\ Hz$$

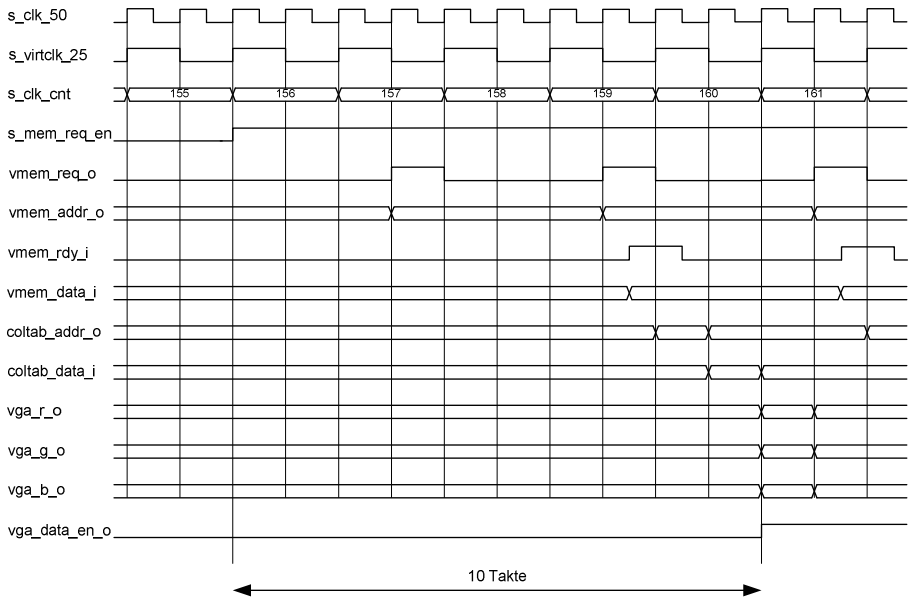

Abbildung 20: Timing-Diagramm für das Auslesen der Bildschirmdaten

In Abbildung 20 sind die Signalabläufe zum Lesen der Bildschirmdaten aus dem SRAM dargestellt. Aus dem Takt *s_clk_50* wird der Pixeltakt *s_virtclk_25* mit halber Frequenz erzeugt. Dies ist der Takt, mit dem die einzelnen Bildpunkte an das TFT-Display ausgegeben werden sollen. Da eine Leseoperation 16 Bit umfasst, werden bei einem Lesezugriff zwei Bildpunkte ausgelesen. Es ist also alle vier Takte bzw. zwei Pixeltakte eine Leseoperation notwendig.

Sobald mit *s_mem_req_en* die Lesebefehle aktiviert werden, wird zu jeder fallenden Taktflanke bei ungeraden *s_clk_cnt* ein Lesebefehl mit *vmem_req_o* abgesetzt. Die aus dem Speicher gelesenen Daten stehen dann fünf Takte später, signalisiert durch *vmem_rdy_i,* am Eingang *vmem_data,i* zur Verfügung. Danach werden zuerst die oberen 8 Bit als Eingangsadresse an das Dualport-RAM, das als Farbtabelle verwendet wird, angelegt. Nach einem weiteren Takt sind die Daten am Ausgang gültig und können an den Ausgängen *vga_r_o*, *vga_g_o* und *vga_b_o* weitergeleitet werden. Folglich ist das Aktivieren der Lesebefehle mit *s_mem_req_en* zehn Takte vor *vga_data_en_o* notwendig.

Da am Beginn aus dem Speicher dreimal hintereinander die erste Adresse der Zeile gelesen wird und nur der dritte Zugriff für den Bildschirm verwendet wird, erfolgt die Aktivierung um zusätzlich acht Takte früher, gesamt also 18 Takte. Die Adresse, von welcher gelesen wird, wird mit der Hilfe der Eingänge *base_addr_i* und *active_area_i* generiert.

für *active_area_i = „00“* gilt: $Startadresse = s_base_addr_i$

für *active_area_i = „01“* gilt: $Startadresse = s_base_addr_i + C_PIXEL/2$

Bei jedem erfolgreich abgesetzten Lesebefehl wird die Adresse um eins erhöht. Ist der Eingang *black_i* aktiv, so werden nicht die vom Speicher ausgelesenen Daten von der Farbtabelle umgewandelt, sondern die am Eingang *roicolor_i* in den unteren 8 Bit angegebene Farbe. Bei *testpic_i = ‚1‘* wird ein Farbmuster als Testbild ausgegeben.

3.4.3 Einheit Zeichenoperationen

Abbildung 21 zeigt die Einheit für die Zeichenoperationen mit ihren Ein- und Ausgängen. Aufgabe dieser Einheit ist, die von der Registereinheit initiierten Zeichenoperationen auszuführen. Dazu werden die benötigten Daten aus dem Speicher gelesen, manipuliert und wieder in den Speicher geschrieben.

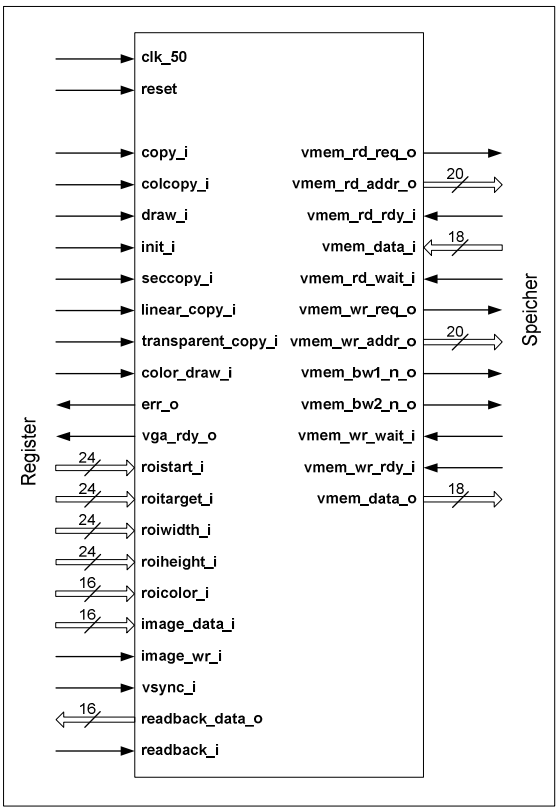

Abbildung 21: Ein- und Ausgabeports der Einheit Zeichenoperationen

Von der Registereinheit werden alle für die Zeichenoperationen notwendigen Parameter übernommen. Die Einheit erkennt, wenn die Registereinheit einen Befehl signalisiert. Danach werden die Daten des Quellbereichs aus dem Speicher gelesen, entsprechend dem Befehl und den Parametern manipuliert und schließlich wieder in den Speicher geschrieben. Wie die Grafikbefehle bearbeitet werden veranschaulicht

Abbildung 22. Tabelle 7: Befehle zum Wechsel in den Zustand CALCULATE POSITION

Befehl	**Signalisiert durch Eingang**
Rechteck zeichnen	draw_i
Flimmerfreies Kopieren	seccopy_i
Kopieren mit Transparenz	transparent_copy_i
Kopieren und Einfärben	colcopy_i
Lineares Kopieren	copy_i
Kopieren und Einfärben mit definierter Farbe	color_draw_i
Auslesen des Videospeichers	readback_i

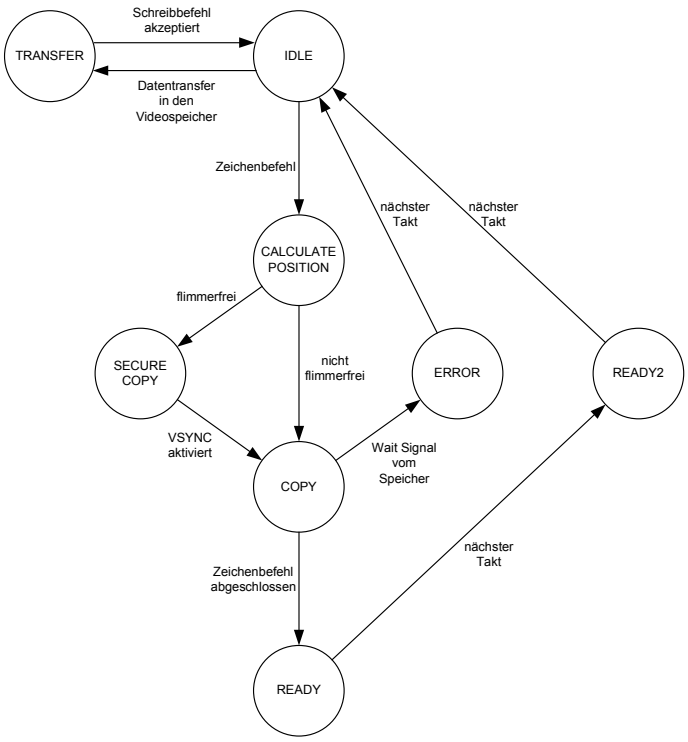

Abbildung 22: Zustandswechsel der Zeicheneinheit

Vom Ausgangszustand IDLE wird beim Aktivieren eines der in Tabelle 7 angeführten Befehle in den Zustand CALCULATE POSITION gewechselt. In diesem Zustand werden die Zeichenoperationen vorbereitet. Es werden Start- und Zieladresse der Operationen festgelegt. Die Startadresse ergibt sich aus dem am Eingang *roistart_i* anliegenden Wert. In der gleichen Weise wird die Zieladresse aus dem Eingang *roitarget_i* und die Länge einer Zeile aus *roiwidth_i* ermittelt. Der Eingang *roiheight_i* gibt die Anzahl der Zeilen für die Zeichenoperation an. In diesem Zustand werden auch die für die Zeichenoperationen benötigten Zähler auf Null gesetzt.

Wird ein flimmerfreier Befehl (*seccopy_i = ‚1')* ausgeführt, so folgt als nächster Zustand SECURE COPY, in dem solange gewartet wird, bis man sich am Ende der Darstellung einer Bildschirmseite befindet *(vsync = ‚0')*. Sodann erfolgt der Wechsel in den Zustand COPY, in dem die eigentliche Zeichenoperation durchgeführt wird. Wird ein nicht flimmerfreier Befehl ausgeführt, so wird aus dem Zustand CALCULATE POSITION direkt in den Zustand COPY gewechselt.

Für den Bildausschnitt, auf den die Zeichenoperation durchgeführt wird, werden die Adressen generiert und die zugehörigen Daten aus dem Speicher gelesen. Nach der Manipulation der Daten werden diese in den Speicher geschrieben. Wurde die Operation abgeschlossen, wird in den Zustand READY gewechselt. Nach einem zusätzlich benötigten Takt, über den Zustand READY2, erreicht man wieder den Ausgangszustand IDLE.

Beim Zugriff auf den Speicher werden die Schreibzugriffe den Lesezugriffen vorgezogen, d. h. es kann theoretisch für diese kein Haltesignal vorkommen. Ist dies doch der Fall, so handelt es sich hierbei um einen Fehler und es wird der Zustand ERROR eingenommen. Danach wird über den Ausgang *err_o* der Registereinheit der Fehler gemeldet und anschließend der Ausgangszustand IDLE eingenommen. Bei

der Aktivierung von *readback_i* wird das mit *roistart_i* adressierte Datenwort aus dem Speicher gelesen und über *readback_data_o* an die Registereinheit ausgegeben.

Im Zustand IDLE kann mit *image_wr_i = ‚1'* in den Zustand TRANSFER gewechselt und der Videospeicher geschrieben werden. Bei der Initialisierung wird mit der Aktivierung von *init_i* die Transferadresse auf ‚0' rückgesetzt. Nach jedem Datentransfer in den Videospeicher wird diese Transferadresse um eins erhöht. So ist es möglich, den Videospeicher beim Systemstart zu initialisieren. Wird der Schreibbefehl im Zustand TRANSFER übernommen, wird in den Zustand IDLE zurückgekehrt.

Der Ablauf eines Lesebefehls ist in Abbildung 23 dargestellt. Ein Lesebefehl wird mit *vmem_rd_req_o = ‚1'* und der Adresse an *vmem_rd_addr_o* signalisiert. Sollte *vmem_rd_wait_i = ‚1'* sein, signalisiert die Speicherschnittstelle noch nicht bereit zu sein. Erst bei *vmem_rd_wait_i = ‚0'* kann der Befehl übernommen und die Leseanforderung zurückgezogen werden.

Die Gültigkeit der Daten am Eingang *vmem_data_i* wird mit *vmem_rd_rdy_i = ‚1'* angezeigt und von der Zeicheneinheit übernommen. Zusätzlich zu den 16 Bit Daten stehen zwei weitere Bits zur Verfügung, diese können als Paritätsbits verwendet werden.

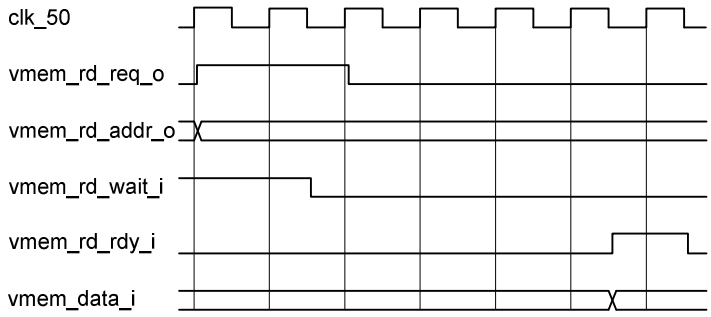

Abbildung 23: Timing-Diagramm des Lesebefehls

In Abbildung 24 ist ein Schreibzugriff dargestellt. Im Gegensatz zum Lesebefehl kommt im fehlerfreien Betrieb kein Haltesignal *vmem_wr_wait_i = ‚1'* vor. Der Befehl wird mit der nächsten steigenden Taktflanke von der Speicherschnittstelle übernommen. *vmem_wr_addr_i* gibt die Adresse an, auf die die Daten von *vmem_data_o* in den Speicher geschrieben werden sollen.

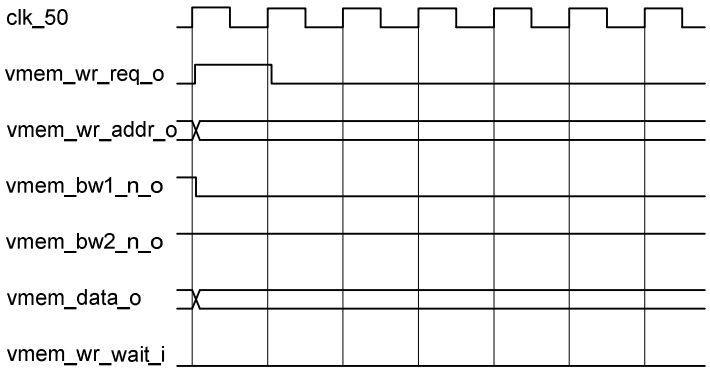

Abbildung 24: Timingdiagramm des Schreibbefehls

Die low-aktiven Ausgänge *vmem_bw1_n_o* und *vmem_bw2_n_o* signalisieren, welches Byte der Daten in den Speicher übernommen werden soll, wobei *vmem_bw1_n_o* Bit 0 bis 7 und *vmem_bw1_n_o* Bit 8 bis 15 repräsentiert. Bei den abgebildeten Signalen würde nur das untere Byte in den Speicher übernommen werden.

3.4.4 Einheit Signaltöne

Diese Einheit dient zur Ausgabe der Signaltöne an den DAC zur Sound-Wiedergabe. Ihre Ein- und Ausgänge sind in Abbildung 25 dargestellt. Die Signalisierungseinheit liest einen von 16 Signalisierungstöne mit je 256 Byte aus dem Videospeicher und führt sie dem DAC zu.

Von der Registereinheit wird signalisiert, wann (*play_snd_i*) welcher Sound (*chime_data_i*) in welcher Lautstärke (*chimevolume_i*) ausgegeben werden soll. Aufgrund der Basisadresse (*base_addr_i*) wird der entsprechende Signalton über die Speicherschnittstelle aus dem Speicher geladen.

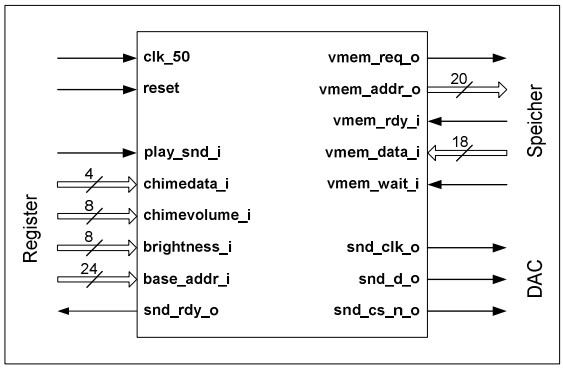

Abbildung 25: Ein- und Ausgangsports der Einheit Signaltöne

Die Audiodaten werden nach dem Laden seriell an einen DAC ausgegeben. Außerdem wird der DAC auch dazu benutzt, den Wert für die gewünschte Helligkeit des Displays (*brightness_i*) in einen Analogwert umzuwandeln.

Aus dem Takt *clk_50 = 81,92 MHz* werden mittels Zähler drei verschiedene Signale erzeugt, die zur Steuerung der Tonausgabe verwendet werden. Nach 40 Takten ist das Signal *s_dac_2M* aus Abbildung 26 für einen Takt auf eins.

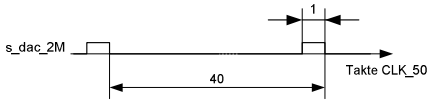

Abbildung 26: Signal s_dac_2M

$$fs_dac_2M = \frac{81{,}92\ MHz}{40} = 2{,}048\ MHz$$

Ein weiterer Zähler zählt die Durchläufe von *s_dac_2M,* nach 125 Durchläufen wird das Signal *s_dac_16k* für einen Takt auf eins gesetzt:

$$fs_dac_16k = \frac{2048kHz}{125} = 16{,}384kHz$$

Bei jedem zweiten Puls von *s_dac_16k* wird auch *s_dac_8k* für einen Takt eins gesetzt, daraus ergibt sich:

$$fs_dac_8k = \frac{fs_dac_16k}{2} = 8{,}192kHz$$

Die Ausgabe der vom DAC umzuwandelnden Werte sind in Abbildung 27 dargestellt.

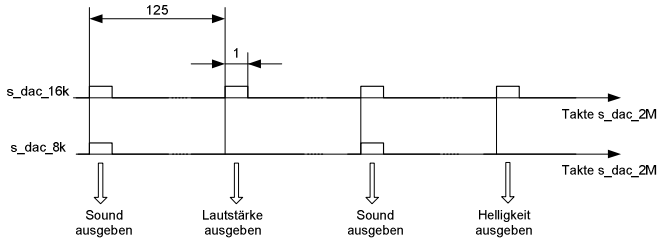

Abbildung 27: Ausgabe der Signaltöne

Die Ausgabe der Werte erfolgt nach einem Zeitschlitzverfahren. In einem Zyklus von vier Pulsen von *s_dac_16k*, wird zu jedem zweiten Puls, zu dem auch ein Puls von *s_dac_8k* vorliegt, je ein Byte der Sound-Daten an den DAC ausgegeben. An den beiden anderen Pulsen wird abwechselnd der Wert der Lautstärke und der Helligkeit des Bildschirms an den DAC ausgegeben. Während die Audiodaten aus dem Speicher gelesen werden, stammen Lautstärke und Helligkeit des Displays von der Registereinheit, es muss also nur einmal auf den Speicher zugegriffen werden. Die Ablaufsteuerung der Ausgabe der Signaltöne ist in Abbildung 28 zu sehen.

Ausgehend von *play_snd_i = ,1'* wird von der Registereinheit ein Sound-Befehl initiiert, wobei *chime-data_i* die Nummer des Sounds angibt, welcher an den DAC ausgegeben werden soll. Beim nächsten Puls von *s_dac_8k = ,1'* wird das erste der 128 16-Bit-Wörter aus dem Speicher angefordert.

Mit *vmem_rdy_i = ,1'* stehen die Daten am Eingang *vmem_data_i* zur Verfügung. Das High-Byte wird in das 13-stellige Schieberegister *s_shift_data* geladen und seriell über *snd_d_o* ausgegeben, wobei jedes Bit jeweils bei *s_dac_2M = ,1'* ausgegeben wird.

Beim nächsten Puls von *s_dac_16k,* bei dem kein Puls von *s_dac_8k* vorliegt, wird abwechselnd der Lautstärke- bzw Helligkeitswert in das Schieberegister geladen und an den DAC ausgegeben.

Beim nächsten Puls von *s_dac_8k* wird das Low-Byte in das Schieberegister geladen und bitweise zu Pulsen von *s_dac_2M* an den DAC über *snd_d_o* ausgegeben. Danach wird das zweite 16- Bit-Wort aus dem Speicher geladen und ausgegeben. Der Vorgang wiederholt sich so lange, bis alle 256 Bytes des Sounds ausgegeben worden sind.

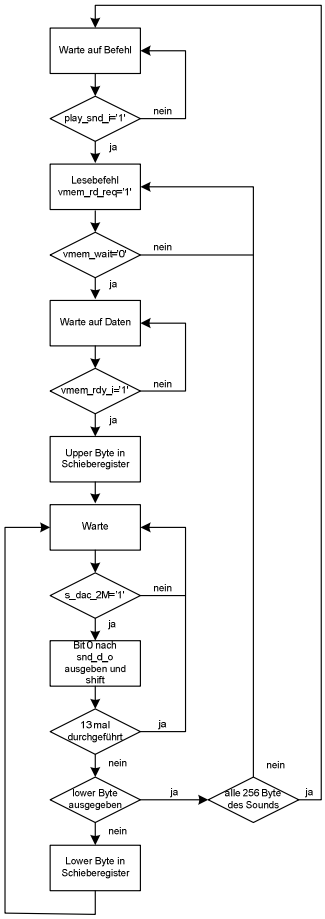

Abbildung 28: Ablaufsteuerung der Ausgabe der Signaltöne

In Abbildung 29 sieht man, wie das 13-Bit-Schieberegister zu laden ist, um dem DAC die gewünschten Befehle zu erteilen. Die oberen drei Bits („101“) signalisieren dem DAC das Schreiben von Daten, die nächsten beiden Stellen adressieren einen Kanal des DACs und die unteren 8-Bit sind entsprechend Tabelle 8 mit den entsprechenden Daten aus den Eingängen zu laden.

Sound	1	0	1	1	1	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
Volume	1	0	1	0	0	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0
Brightness	1	0	1	0	1	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

Abbildung 29: Befehlsformat für die Signaltonausgabe

Tabelle 8: Belegung der Schieberegisterstellen zur Signaltonausgabe

Registerstellen	Bedeutung
S7-S0	vmem_data_i High-Byte bzw. Low-Byte
V7-V0	chime_volume_i
B7-B0	brightness_i

3.4.5 Speicherschnittstelle

Die Speicherschnittstelle koordiniert den Zugriff auf den Videospeicher. Hier treffen die Speicherbefehle der Einheiten aufeinander, sie werden nach festen Prioritäten an den Speicher weitergeleitet und im Fall von Lesezugriffen, werden die gelesenen Daten werden an die aufrufende Einheit ausgegeben. In Abbildung 30 sind die Ein- und Ausgänge der Speicherschnittstelle dargestellt.

Von den drei Einheiten Bildschirm, Zeichenoperationen und Signaltöne werden Lesebefehle an den Speicher und von der Einheit Zeichenoperationen zusätzlich Schreibbefehle gestellt. Die Zugriffsversuche werden von der Speicherschnittstelle an den Speicher weitergeleitet, wobei der Zugriff nach festen Prioritäten (Tabelle 9) vergeben wird.

Tabelle 9: Zugriffsarten auf das Speicherinterface

Priorität	Speicherzugriff	Eingang
1	Schreibzugriffe der Zeicheneinheit	drw_wr_req_i
2	Lesezugriffe der Einheit für den Bildschirmaufbau	strm_req_i
3	Lesezugriffe der Einheit für Signaltöne	snd_req_i
4	Lesezugriffe der Zeicheneinheit	drw_rd_req_i

Oberste Priorität beim Zugriff auf den Speicher besitzen die Schreibzugriffe der Zeicheneinheit, danach folgen die Lesezugriffe für den Bildschirmaufbau. Da beide Zugriffe nie zur selben Zeit auftreten, wird die Bildschirmeinheit trotz niedrigerer Priorität nie verzögert und die rechtzeitige Ausgabe der Bilddaten gewährleistet. Warum diese Zugriffe nie gemeinsam auftreten, ist in der Verarbeitung der Daten in der Zeichenoperationseinheit begründet. In dieser Einheit erfolgt der Schreibzugriff nach der Manipulation der gelesenen Bilddaten, immer acht Takte nach dem zugehörigen Lesebefehl (Abbildung 31). Der Lesebefehl hat eine niedrigere Priorität als der Bildschirm und wird deshalb bei einer Kollision verzögert. Da aber die Lesezugriffe der Bildschirmeinheit alle vier Takte erfolgen, können diese nicht gleichzeitig mit einem Schreibzugriff auftreten. Die Zeicheneinheit kann beim Schreiben nicht angehalten werden und erhält deshalb die höchste Priorität. Um eine kontinuierliche Sound-Ausgabe zu erreichen, werden die Lesebefehle der Signaltöne denen der Zeichenoperationen vorgezogen. Durch die niedrigere Priorität der Lesezugriffe gegenüber der Schreibzugriffe der Zeichenoperation, wird eine Blockade der Einheit verhindert, weil manipulierte Daten erst geschrieben werden bevor neue Daten gelesen werden.

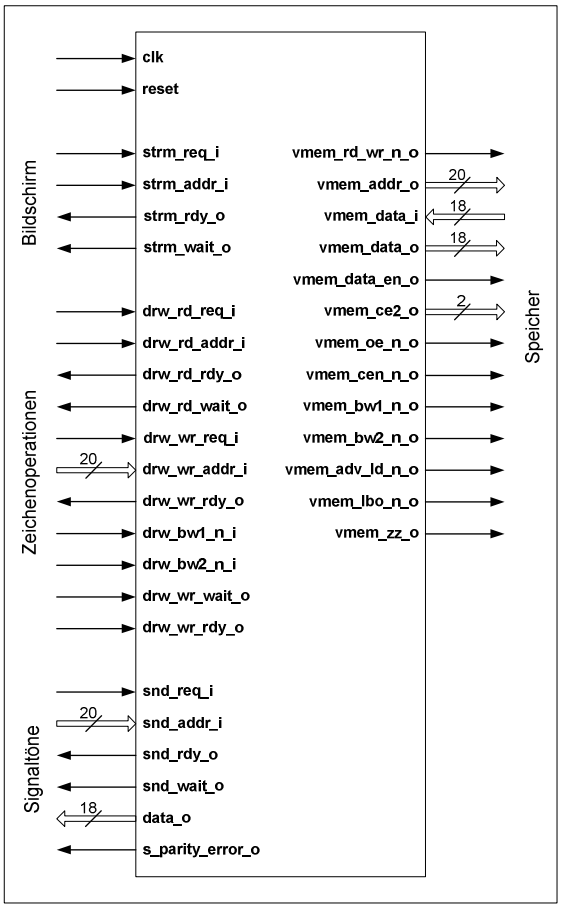

Abbildung 30: Ein- und Ausgangsports der Speicherschnittstelle

In Abbildung 31 kann man erkennen, wie die Schnittstelle Kollisionen beim Speicherzugriff behandelt. Die Einheit für den Bildschirmaufbau liest alle vier Takte ein 16-Bit-Datenwort aus dem Speicher, hier durch *strm_req_i* dargestellt. Um rechtzeitig Bildschirmdaten am LCD-Display ausgeben zu können, darf es zu keiner Verzögerung kommen. Gegenüber den Leseanforderungen der Sound-Einheit *snd_req_i* und der Zeicheneinheit *drw_rd_req_i* besitzt der Bildschirmaufbau Vorrang. Kollisionen mit einem Schreibbefehl der Zeicheneinheit (in Abbildung 31 Signal *drw_wr_req*) werden dadurch vermieden, dass der Schreibwunsch immer acht Takte nach dem Lesen (*drw_rd_req_i*) erfolgt und folglich nicht zeitgleich mit dem Bildschirmaufbau auftreten kann.

Das Eingangssignal mit der höheren Priorität wird mit der entsprechenden Adresse und beim Schreibzugriff mit den Daten an den Speicher weitergeleitet. Die nachrangigen Speicherzugriffe werden angehalten. Zu diesem Zweck wird zuvor mit der fallenden Taktflanke geprüft, welche Zugriffswünsche vorliegen und nach den obigen Prioritäten die Haltesignale gesetzt. Danach zieht nur jene Einheit den Zugriffswunsch zurück, deren Lese- bzw. Schreibbefehl akzeptiert worden ist, d. h. kein Haltesignal bei der steigenden Taktflanke vorgelegen hat.

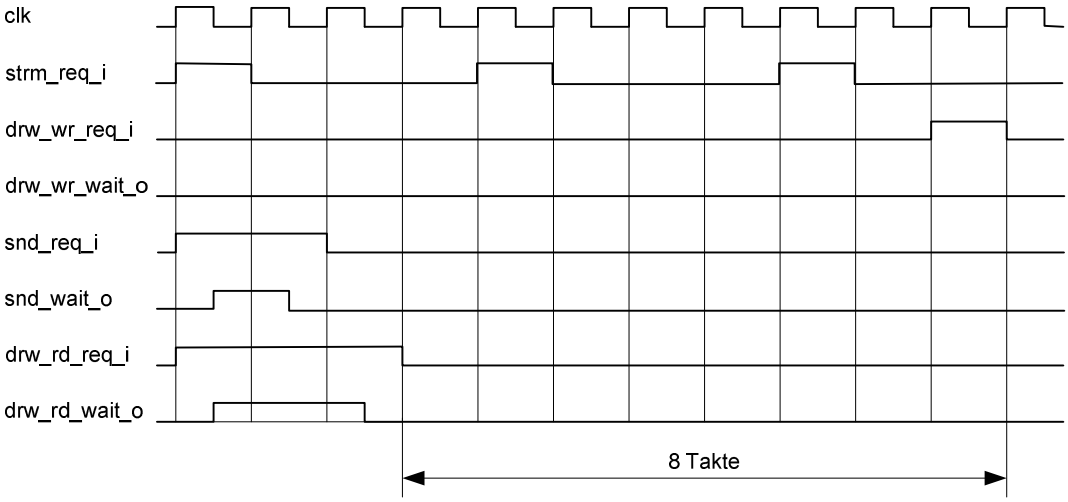

Abbildung 31: Priorisierung an der Schnittstelle zum Speicher

Die Daten für den Bildschirmaufbau sind zur vierten steigenden Taktflanke nach dem abgesetzten Befehl verfügbar (Abbildung 32).

Zur Fehlererkennung wird von der Schnittstelle beim Speichern die gerade Parität des unteren und des oberen Byte gebildet und mit den Daten ausgegeben. Beim Lesen von Daten wird ebenfalls die Parität gebildet und mit den gespeicherten Paritätsbits verglichen, bei einer Unterscheidung wird ein Fehler angezeigt.

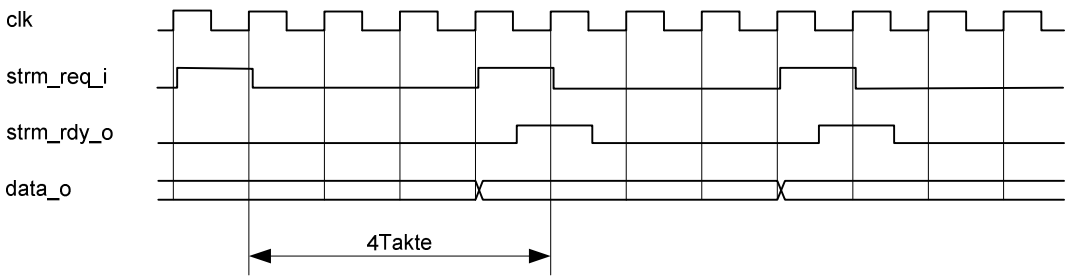

Abbildung 32: Timing-Diagramm Datenausgabe Speicherinterface

4. Halbleiterspeicher

RAMs (Random Access Memories) sind Schreib-/Lesespeicher mit wahlfreiem Zugriff, d. h. der Inhalt einer Speicherzelle ist durch Angabe einer Adresse direkt verfügbar [WOI07]. Je nach Art des Speicherprinzips unterscheidet man zwischen statischem RAM, kurz SRAM, und dynamischem RAM (DRAM). Ein statischer Speicher behält die in ihm gespeicherte Information bei aufrechter Versorgungsspannung. Beim dynamischen Speicher muss der Speicherinhalt durch Auffrischen des Speicherinhalts erhalten werden.

In diesem Kapitel soll auf die Unterschiede der Speichertypen eingegangen werden, um die notwendigen Änderungen in der Schnittstelle zum Speicher zu verdeutlichen.

4.1 SRAM

Die Zellen eines SRAM-Bausteins sind aus Flipflops aufgebaut. Das Speichern der Information erfolgt in bistabilen Kippschaltungen und bleibt solange erhalten, bis der Speicherinhalt durch einen neuerlichen Schreibvorgang überschrieben oder die Versorgungsspannung abgeschaltet wird.

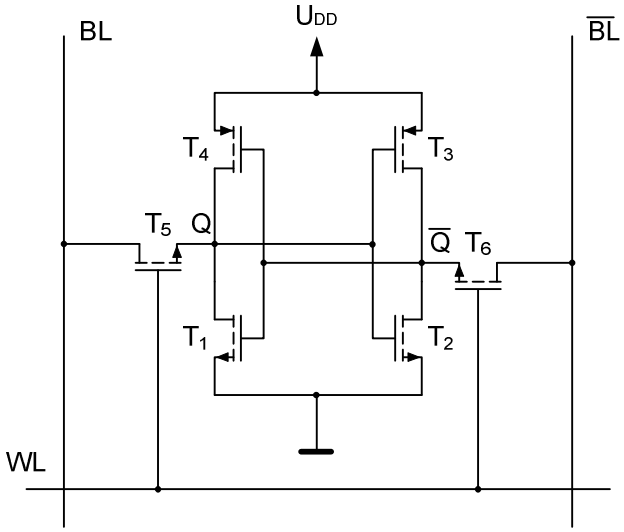

Abbildung 33: 6-Transistorzelle [SIE03 S. 266]

In Abbildung 33 ist der Aufbau einer SRAM-Speicherzelle in CMOS-Technologie dargestellt. Die beiden Transistoren T_5 und T_6 arbeiten als Auswahltransistoren und werden über die Wortleitung WL aktiviert. Die Transistoren T_1 bis T_4 bilden zwei kreuzgekoppelte Inverter und sorgen dafür, dass die Punkte Q bzw. Q# die Potentiale von BL und BL# da einnehmen. Solange die Betriebsspannung anliegt, bleibt die Information erhalten.

Beim Schreiben wird die Bitleitung BL an den gewünschten Logikpegel gelegt, an BL# der entsprechende komplementäre Pegel. Die Ausgangsknoten Q bzw. Q# nehmen dann beim Auswählen des Transistors diese Bitleitungspegel ein, beim Deaktivieren der Transistoren T_5 und T_6 bleibt dieser Zustand erhalten.

Um die gespeicherte Information auszulesen, werden die Bitleitungen BL und BL# auf das Potential von U_{DD} vorgeladen und anschließend die Speicherzelle mit T_5 und T_6 ausgewählt. Je nach Speicherzustand werden die Knoten Q bzw. Q# umgeladen und es ergibt sich eine Spannungsdifferenz, die je nach Speicherinformation positiv oder negativ ist. Diese Spannungsdifferenz wird mit einem Differenzverstärker ausgewertet und das digitale Ausgangssignal erzeugt [SIE03 S. 266].

Der Vorteil der SRAM-Zelle liegt darin, dass im Ruhezustand praktisch keine Querströme im CMOS-Inverter des Flip-Flops auftreten. Dadurch lassen sich SRAM-Speicher mit sehr niedriger Ruhestromaufnahme realisieren. Der Nachteil liegt im hohen schaltungstechnischen Aufwand von sechs Transistoren, der damit verbundenen großen Chipfläche und den Kosten.

Im SRAM-Baustein sind die Speicherzellen in einer Matrix angeordnet (Abbildung 34).

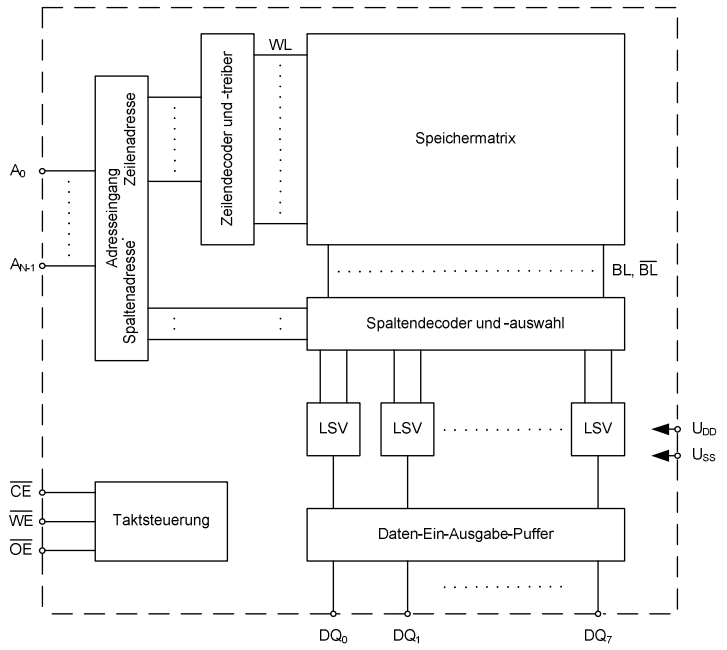

Abbildung 34: Struktur eines SRAM-Speichers [RHE92, S. 51]

Die Verwendung der Anschlüsse kann aus Tabelle 10 entnommen werden. Die Adresse A_i wird in einem Schritt vom Speicher übernommen und die Daten nach der Zugriffszeit an den Ausgängen DQ_0 bis DQ_7 geliefert.

Tabelle 10: Anschlüsse des SRAM-Speichers

Bezeichnung	Signalname
Ai	Adresssignale
DQj	bidirektionale Ein- und Ausgänge
UDD, USS	Betriebsspannungen
$\overline{CE}$	Chip-Enable
$\overline{WE}$	Write-Enable
$\overline{OE}$	Output-Enable
BL	Bitleitung
WL	Wortleitung
LSV	Lese-Schreib-Verstärker

Aus der Adresse wird mittels Zeilen- und Spaltendecoder eine Speicherzelle ausgewählt und die entsprechende Wortleitung WL und die Bitleitungen BL, BL# verbunden. Beim Schreiben (CE#= 'L', WE#= 'L') werden die Daten vom Daten-Ein-Ausgabe-Puffer durch einen Schreib-Lese-Verstärker verstärkt und in den Speicher eingeschrieben. Beim Lesen (CE# = 'L', WE# = 'H') wird die Differenzspannung zwischen dem Bitleitungspaar BL und BL# verstärkt und in den Daten-Ein-Ausgabe-Puffer geschrieben.

SRAM-Speicher sind schnell und können auch mit heutzutage üblichen Taktraten betrieben werden. Sie sind unkompliziert im Zugriff, haben aber einen höheren Platzbedarf als dynamische Speicher. Die Integration der SRAM-Speicherzelle ist nur mit höherem technischem Aufwand möglich und SRAMs weisen meist eine geringere Speicherkapazität als DRAMs auf. Sie sind teuer und werden hauptsächlich für Cache-Speicher verwendet [HPM00, S. 517].

SRAMs sind zwar schnelle aber auch teure Speicher, die in kleineren Einheiten verwendet werden als dynamische Speicher. Typisches Einsatzgebiet des statischen Speichers sind daher Cache-Speicher.

4.2 DRAM

Im Unterschied zum SRAM wird bei DRAMs die Information als elektrische Ladung in einem Kondensator gespeichert. Beim Lesevorgang kommt es zum Entladen des Kondensators (zerstörendes Lesen), wodurch anschließend der Speicherinhalt wieder in den Speicher geschrieben werden muss. Weiters ist ein periodisches Auffrischen des Speicherinhaltes (Refresh) notwendig, da die Leckströme die Kondensatoren kontinuierlich entladen. Diese Aufgabe wird von einer eigens dafür vorgesehenen Steuerlogik übernommen [BÄH02, S. 132].

In Abbildung 35 ist der einfache Aufbau einer DRAM-Speicherzelle aus nur einem Transistor und einem Kondensator zu sehen.

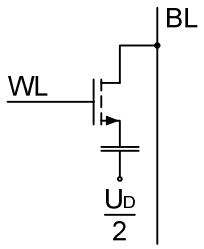

Abbildung 35: DRAM-Speicherzelle [SIE03, S. 268]

Um Information in die Speicherzelle zu schreiben, wird an die Bitleitung BL der entsprechende Pegel U_D oder 0V angelegt. Danach wird die Speicherzelle mittels der Wortleitung WL über den Auswahltransistor selektiert und der Kondensator aufgeladen. Die positive oder negative Spannung am Kondensator repräsentiert logisch ‚1' oder ‚0'.

Beim Lesevorgang wird die Bitleitung auf $U_D/2$ vorgeladen und entlädt sich je nach Speicherzustand bei der Aktivierung mit der Wortleitung. Die Auswertung dieser relativ kleinen Spannungsänderung an der Bitleitung erfolgt aufgrund der großen Kapazität der Bitleitung und der im Verhältnis kleinen Speicherkapazität mit einem differentiellen Leseverfahren [SIE03, S. 269].

DRAMs sind bedingt durch ihren einfacheren Aufbau mit dem Kondensator als Speicherelement 10-mal langsamer, aber auch 10-mal billiger als SRAM-Chips und benötigen nur ein Viertel bis ein Achtel der Fläche [BEI04, S. 192]. Der Zugriff auf den DRAM-Speicher ist aber bedingt durch die notwendigen Refresh-Zyklen aufwendiger und macht eine eigene Controller-Logik notwendig.

Ansteuerung des DRAMs

Der große Vorteil einer DRAM-Zelle sind der einfache Aufbau, die daraus resultierenden geringeren Kosten und der geringe Platzbedarf gegenüber der SRAM-Zelle. Es können um eine Größenordnung höhere Packungsdichten als bei SRAMs erreicht werden. Um trotz großer Speichertiefe kleine Speicherbauteile realisieren zu können, müssen die Anschlüsse reduziert werden. Die Adressen werden daher in Zeilen- und Spaltenadressen aufgeteilt und in zwei Stufen im Multiplexbetrieb an den Speicher angelegt. Die Zeilenadresse wird mit der fallenden Flanke des RAS-Signals (Row Address Strobe) übertragen in ein Pufferregister übernommen. Anschließend wird mit einem Decoder die adressierte Zeile des Speichers aktiviert. Danach werden alle Kondensatoren der Zeile über den Auswahltransistor mit den Bitleitungen verbunden, die Ladungen fließen auf die Bitleitungen und ihr Potential ändert sich geringfügig. Sinkt das Potential, war der Kondensator ungeladen, steigt es, so war der Kondensator geladen. Die DRAM-Steuerung aktiviert die Leseverstärker, die Potentialdifferenzen der Bitleitungen BL und BL# werden verstärkt und der Speicherinhalt der gesamten Zeile wird einem I/O-Gatter zugeführt.

Mit der fallenden Flanke des CAS-Signals (Column Address Strobe) wird der zweite Teil der Adresse übergeben. Die Spaltenadresse wird dekodiert und aus dem I/O-Gatter genau ein Speicherinhalt ausgewählt und an den Ausgang ausgegeben. Da die Auswahl einer Matrixzeile und das anschließende Auslesen des Speicherinhaltes über den Schreib-/Leseverstärker viel Zeit benötigt, kommt es trotz Multiplexbetrieb der Adresseingabe zu keiner Verzögerung [BÄH02, S. 171].

Am Ende des Lesevorganges sind BL und BL# auf dem Pegel entsprechend dem Speicherzustand der Zelle, d. h. die Zeile ist beim Lesen einem Auffrischen unterzogen. Anschließend werden Spaltendecoder, Zeilendecoder und Verstärker wieder deaktiviert und der Lesevorgang ist abgeschlossen.

Das Schreiben in den Speicher geschieht in ähnlicher Weise wie das Lesen. Mit der fallenden Flanke des RAS-Signals wird die Zeilenadresse übergeben und eine Zeile durch den Zeilendecoder ausgewählt. Gleichzeitig wird der DRAM-Steuerung mit dem WE-Signal mitgeteilt, dass ein Schreibvorgang folgt. Danach werden von der Logik die Eingangsdaten dem Eingangsdatenpuffer zugeführt. Durch die Auswahl der Zeile werden die gespeicherten Ladungen auf die Bitleitungen übertragen. Mit fallender Flanke des CAS-Signals wird die Spaltenadresse an den Spaltendecoder übergeben. Dieser decodiert die Spaltenadresse und schaltet ein Transfergatter durch. Die Eingangsdaten werden verstärkt und heben bzw. senken je nach Schreibdaten das Potential der Bitleitungen. Die nicht ausgewählten Spalten der Zeile frischen ihren Speicherinhalt dabei auf und anschließend werden Zeile und Spalte wieder deaktiviert [HPM00].

Adressierung des DRAMs

In Abbildung 36 ist die Adressierung eines DRAM-Bausteins dargestellt.

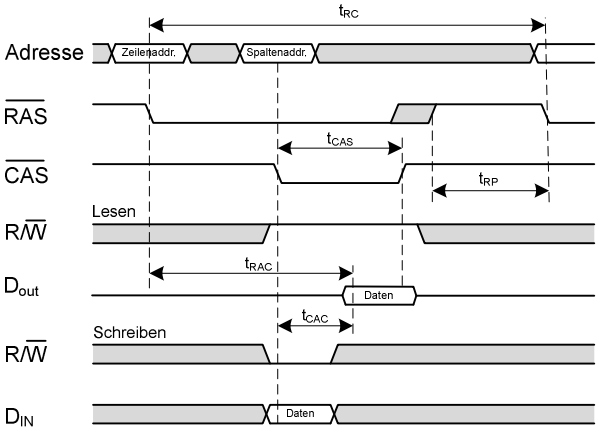

Abbildung 36: Adressierung eines DRAM-Bausteins [BÄH02, S. 172]

Beim Lesen werden die Daten ausgehend von der fallenden Flanke des RAS-Signals nach der RAS-Zugriffszeit t_{RAC} am Ausgang ausgegeben. Während dieser Zeit muss sich das R/W-Signal auf logisch ‚1' befinden. Die Daten sind am Ausgang bis zur steigenden Flanke von CAS stabil, danach wird der Ausgang wieder hochohmig (tristate).

Nach dem Speicherzugriff müssen die ausgelesenen Daten wieder in den Speicher geschrieben und die Bitleitung vorgeladen werden (Precharge), dies dauert die Zeit t_{RP}. Erst dann kann auf den Speicher erneut zugegriffen werden. Die minimale Zeit zwischen zwei Zugriffen, d. h. zwei fallenden Signalflanken von RAS ist die Zykluszeit t_{RC}.

Soll in den Speicher geschrieben werden, müssen gleichzeitig Spaltenadresse und Daten am Dateneingang D_{IN} anliegen. Diese Daten werden dann mit der fallenden Flanke von CAS in den Speicher übernommen. Während des Einschreibens muss sich das R/W-Signal auf logisch ‚0' befinden.

t_{RAC} ist die Zeit zwischen dem Anlegen der Zeilenadresse und der Ausgabe des ersten Datums einer Zeile. t_{CAC} ist die Zeit zwischen Übergabe der Spaltenadresse und der Gültigkeit der Daten am Ausgang. Sie ist charakteristisch für aufeinander folgende Datenausgaben innerhalb einer Zeile im Burst-Modus.

Auffrischen von DRAMs

Ein Nachteil der DRAM- gegenüber der SRAM-Zelle liegt in der Flüchtigkeit der Ladung des Speicherkondensators. Um zu verhindern, dass die Information im Speicher nicht verloren geht, muss die gespeicherte Information jeder Zelle in einem Intervall im Millisekundenbereich aufgefrischt, d. h. ein sogenannter Refresh durchgeführt werden. Dabei wird der Inhalt des gesamten Speichers Zeile für Zeile ausgelesen und erneut in den Speicher geschrieben. Dies kann auf verschiedene Arten erfolgen [WOI07]:

Burst-Refresh: Am Anfang der Auffrischperiode werden alle Zeilen nacheinander aufgefrischt.

Distributed Refresh: Die einzelnen Refresh-Zyklen werden gleichmäßig über die Auffrischperiode verteilt.

Hidden Refresh: Der Refresh wird nur von außen angestoßen, ein interner Zähler wählt die nächste Adresse.

DRAM-Betriebsmodi

Die Verarbeitungsgeschwindigkeiten bei Prozessoren erhöhten sich ständig, sodass die Leistungsfähigkeit herkömmlicher DRAM-Bausteine nicht mehr ausreichend war. Deshalb versuchte man mit verschiedenen Maßnahmen, die Datenraten der dynamischen Speicher zu erhöhen.

Page Mode

Beim Speicherzugriff auf das DRAM wird zuerst mit dem RAS-Signal die Zeilenadresse angegeben und danach mit dem CAS-Signal die Spaltenadresse. Betrifft der nachfolgende Speicherbefehl dieselbe Zeile, so ist es eigentlich nicht notwendig, die Zeile nochmals zu aktivieren. Beim Page Mode (Abbildung 37) bleibt die Zeilenadresse gleich, es wird nur die Spaltenadresse geändert. Dadurch kann man die Zeit für die nochmalige Angabe und Aktivierung der Zeilenadresse einsparen. Das RAS-Signal bleibt aktiv, nur das CAS-Signal wird kurzzeitig zurückgenommen, um danach eine neue Spaltenadresse zu decodieren. Zu beachten ist aber, dass das RAS-Signal aus Gründen der Stabilität der Daten nicht beliebig lang aktiv bleiben darf, nach etwa 200 Zugriffen muss es deaktiviert werden.

In diesem Modus kann aber nicht nur gelesen sondern auch geschrieben werden, es können auch Lese- und Schreibbefehle abwechselnd durchgeführt werden, sofern sie innerhalb der Zeile zugreifen.

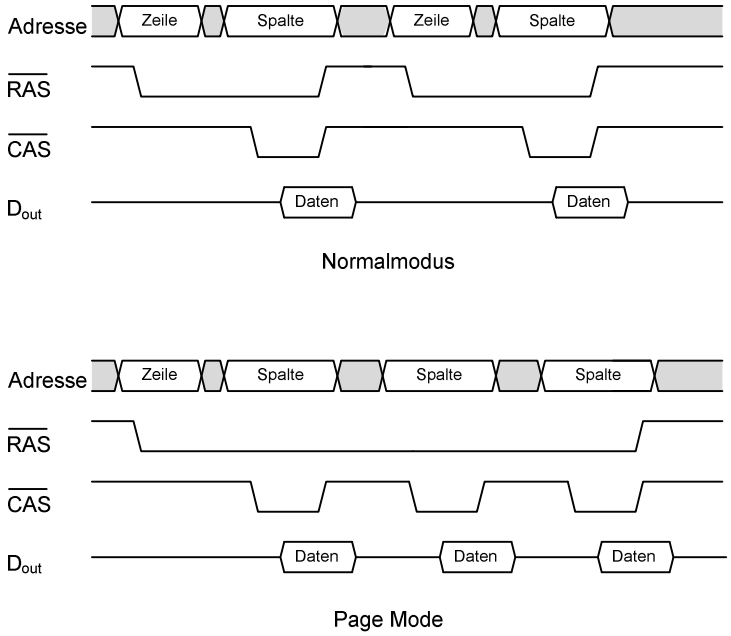

Abbildung 37: Page Mode [HPM00]

Hyper-Page Mode (EDO-Modus)

Beim vorhin beschriebenen Page Mode wird die Gültigkeit der Daten mit dem CAS-Signal angezeigt. Erst nach der steigenden Flanke und nach einer bestimmten Erholzeit kann das CAS-Signal die nächste Spaltenadresse mit der steigenden Flanke des CAS-Signals dem Speicher übermitteln. Beim Extended-Data-Output-DRAM (EDO-DRAM) zeigt nicht die fallende Flanke des CAS-Signals, sondern ein internes Signal die Gültigkeit der Daten an. Dadurch kann parallel zur Datenausgabe die nächste Spaltenadresse bereits mit dem CAS-Signal übergeben werden. Damit können Lesezugriffe innerhalb einer Seite beschleunigt werden [WOI07].

Burst-Hyper-Page Mode (BEDO-Modus)

Bei Speicherzugriffen innerhalb einer Zeile sind fortlaufende Adressen sehr wahrscheinlich. Beim BEDO Mode werden Daten mehrerer aufeinanderfolgender Adressen mit einem Lesebefehl aus dem Speicher gelesen, man spricht hier von einem Burst-Zugriff. Es wird zuerst dem Speicher mit dem RAS-Signal die Zeilenadresse und mit dem CAS-Signal die Spaltenadresse übermittelt. Anschließend fungiert das CAS-Signal als Zählimpuls für einen Spaltenzähler, z. B. werden bei einem Burst = 4 die Daten von vier aufeinanderfolgenden Spaltenadressen ausgegeben. Aufgrund der Pipelinestruktur kann während des letzten Impulses des CAS-Signals bereits die nächste Spaltenadresse übermittelt werden, sodass innerhalb einer Seite der nächste Burst gleich anschließen kann [WOI07].

Interleaving

Beim Interleaving wird der Speicher in Bänke unterteilt, die getrennt angesprochen werden können. So ist es möglich, während des Zugriffes auf eine Bank die nächste Bank bereits zu aktivieren und eine

Beschleunigung bei sequenziellen Speicherzugriffen zu erreichen. Beim 2-Wege-Interleaving überlagern sich RAS-Vorladezeit einer Bank mit der Zugriffszeit der anderen Bank und die Zugriffsrate kann sich verdoppeln. Beim 3- bzw. 4-Wege-Interleaving erhöht sich, bei entsprechender Adressierung, die Datenrate auf das Drei- und Vierfache.

In Abbildung 38 ist das Lesen auf ein DRAM dargestellt, der gleichzeitig Page-Modus und 2-Wege-Interleaving nutzt.

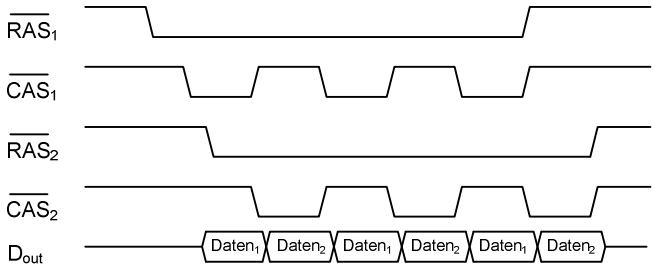

Abbildung 38: Lesezugriff im Paged/Interleaved-Modus

Bank 1 speichert dabei die ungeraden Adressen einer Page und Bank 2 die geraden, die Bänke sind also nicht wie beim einfachen Page-Modus wortweise sondern Page-weise verzahnt. Die Zugriffsrate kann mit der Kombination beider Verfahren nochmals erhöht werden, Voraussetzung für die Leistungssteigerung ist allerdings, dass die Speicherzugriffe innerhalb einer Page fortlaufend adressiert sind [HPM00].

Der in der Diplomarbeit verwendete DDR-Controller unterstützt den parallelen Bankzugriff. Vom Controller wird die Information gespeichert, welche Bank mit welcher Zeile geöffnet ist. Dadurch werden die Bänke bis zu einem Zugriff auf eine andere Zeile innerhalb der gleichen Bank oder einem Refresh offengehalten. Damit wird verhindert, dass Bänke geschlossen werden und beim nächsten Zugriff wieder aktiviert werden müssen [WIE07].

4.3 DDR-SDRAM

Beim SDRAM (Synchronous Dynamic Random Access Memory) hängt der Zugriff auf den Speicher nicht mehr von den Signalflanken der Signale RAS und CAS ab. Vielmehr werden die Zustände der Signale CS (Chip Select), RAS, CAS und WE (Write Enable) zur steigenden Taktflanke als Kommandos interpretiert. Dies hat den großen Vorteil, dass man synchron zum Takt Speicherbefehle erteilen kann und sich Probleme aufgrund zeitkritischer Signalflanken vermeiden lassen [WOI07].

DDR bedeutet Double Data Rate und man versteht unter einem DDR-SDRAM einen doppelt getakteten synchronen dynamischen Speicher, bei dem sowohl zur steigenden als auch zur fallenden Taktflanke Daten transferiert werden können.

Dieser Speicher bietet außerdem weitere Eigenschaften wie:

- Die Adressangaben, Daten- und Steuersignale werden bei diesem Typ in Register abgelegt und mit dem Bustakt synchronisiert.

- Eine interne Logik für den Refresh des Speichers mit Refresh-Zähler und Counter für die Adressierung ist vorhanden. Zusätzlich wird ein Power Down Mode unterstützt, eine Art Standby-Zustand, in dem der Speicherinhalt erhalten bleibt.
- Schreib- und Lesezugriffe werden in Burst-Form durchgeführt. Bei jedem Speicherbefehl wird eine programmierte Sequenz von 2, 4 oder 8 Datentransfers hintereinander durchgeführt.
- Durch das Pipelining der Kommandos ist es möglich, bereits vor Ende der Datenausgabe weitere Befehle zu aktivieren [WOI07].

Standardisiert wurde die Schnittstelle des SDRAMs vom Joint Electron Device Engineering Council, kurz JEDEC. Dieser Standard enthält die Minimalanforderungen an SDRAMs um nach JEDEC standardisiert genannt werden zu dürfen. Die Befehle für den Speicherzugriff werden durch eine Kombination der Signale CS, RAS, WE, CAS angegeben. Tabelle 11 enthält die möglichen Befehle und ihre Aktivierung nach dem JEDEC-Standard.

Tabelle 11: SDRAM-Speicherbefehle [JED03]

Befehl	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$
DESELECT	H	X	X	X
NO OPERATION	L	H	H	H
ACTIVE	L	L	H	H
READ	L	H	L	H
WRITE	L	H	L	L
BURST TERMINATE	L	H	H	L
PRECHARGE	L	L	H	L
AUTO OR SELF REFRESH MODE	L	L	L	H
MODE REGISTER SET	L	L	L	L

Zum Abschluss dieses Kapitels folgt eine kurze Beschreibung der einzelnen Befehle nach dem JEDEC-Standard [JED03]:

Deselect

Es werden keine neuen Befehle vom DDR-SDRAM angenommen, in Bearbeitung befindliche Befehle sind davon nicht betroffen.

No Operation

Durch diesen Befehl wird verhindert, dass unerwünschte Befehle während Wartezyklen oder dem Zustand Idle in die Register übernommen werden.

Active

Aktiviert eine Zeile einer Bank für den Speicherzugriff. Die Zeile bleibt anschließend geöffnet, bis sie durch einen Precharge-Befehl geschlossen wird. Soll in derselben Bank eine andere Zeile aktiviert wer-

den, muss zuerst die Zeile mit Precharge geschlossen werden, bevor die nächste Zeile mit Active geöffnet werden kann.

Read

Führt einen Lesezugriff im Burst-Modus auf die geöffnete Zeile durch. Ist Auto Precharge aktiviert, so wird die Zeile nach Abschluss des Lesezugriffs durch Precharge vorgeladen, anderenfalls bleibt sie für weitere Zugriffe geöffnet.

Write

Bei diesem Befehl wird ein Schreibzugriff im Burst-Modus auf eine geöffnete Zeile durchgeführt. Dabei befinden sich die Datenbytes an den Datenleitungen und die Maskierungsbits auf den ihnen zugehörigen Datamask-Leitungen. Es werden nur jene Datenbytes in den Speicher geschrieben, deren zugehörige Datamask-Bits nicht gesetzt sind.

Burst Terminate

Mit diesem Befehl kann eine Read-Burst-Sequenz abgebrochen werden.

Precharge

Um eine oder alle geöffneten Speicherzeilen zu schließen, wird der Befehl Precharge verwendet.

Auto- or Self Refresh Mode

Um einen Verlust des Speicherinhaltes zu verhindern, ist es notwendig, die Daten alle 64 ms aufzufrischen. Beim Auto-Refresh-Befehl wird eine Zeile aufgefrischt, wobei die Adressierung der Zeile durch den Speicher mit Hilfe eines internen Adresszählers erfolgt. Üblicherweise wird der Speicher periodisch aufgefrischt, mit einem entsprechenden Refresh-Intervall von 64 ms geteilt durch die Zeilenanzahl des Speichers. Die Refresh-Befehle können aber auch in Burst-Form durchgeführt werden, erlaubt sind hierbei maximal acht aufeinanderfolgende Refresh-Befehle.

Die zweite Variante, durch die die Speicherinhalte erhalten bleiben, ist der Self-Refresh-Modus. Dies ist ein Modus, bei dem die Speicherinhalte selbst vom Speicher aufgefrischt werden, auch wenn das übrige System inaktiv ist. In diesen Modus wird der Speicher durch Anlegen des Self-Refresh-Befehls versetzt, in diesem wird kein externes Taktsignal benötigt. Durch eine festgelegte Befehlssequenz kann der Self-Refresh-Modus verlassen und nach 200 Takten wieder aktiv auf den Speicher zugegriffen werden.

Mode Register Set

Dient dazu, das Extended-Mode-Register zu setzen, um Parameter wie z. B. Ausgangstreiberstärke zu konfigurieren.

Beispiel eines Lesebefehls

Als Beispiel sind in Abbildung 39 zwei Lesebefehle auf die gleiche Bank und Zeile mit der Burstlänge von 4 dargestellt. Ein Speicherzugriff auf die noch nicht geöffnete Bank beginnt mit einem Active-Befehl, die dabei angegebene Adresse wählt die anzusprechende Bank und die Zeile aus, um sie für den Zugriff zu öffnen. Als nächstes folgt dann die Anweisung Read oder Write. Die mit diesen Befehlen anliegende Adresse gibt die Startadresse des Speicherzugriffs an, ab welcher der burst-orientierte Speicherzugriff erfolgt.

Die Ausgangsdaten DQ werden mit dem dazugehörigen Strobe-Signal DQS als Zeitreferenz übertragen. Die CAS-Latenzzeit (CL = 2) ist die Anzahl der Takte zwischen Lesebefehl und Ausgabe des ersten Datenblockes. Durch den Zugriff auf die gleiche Bank und Zeile, erfolgt die Datenausgabe des zweiten Lesebefehls gleich im Anschluss an die Ausgabe der Lesedaten des ersten Befehls.

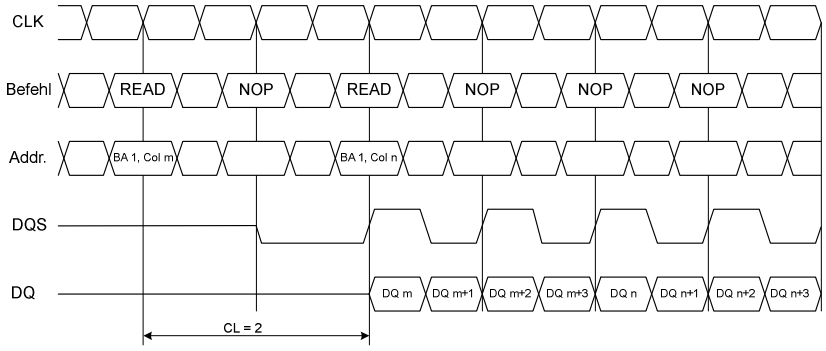

Abbildung 39: Read-Befehl mit Burstlänge von 4 [JED03]

5. DDR-SDRAM-Controller

Als DDR-Controller soll der von Dipl.-Ing. Ingo Wieser in seiner Diplomarbeit für das Institut für Computertechnik der TU-Wien in Zusammenarbeit mit der Firma Oregano Systems entworfene Controller verwendet werden. Dieser bietet dem Video-Controller folgende Anwendungsschnittstelle:

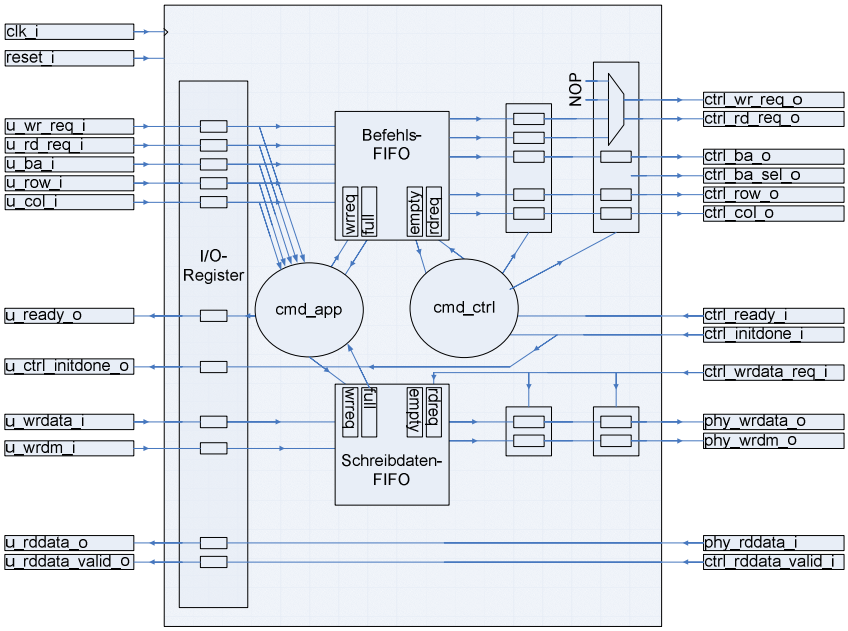

Abbildung 40: Anwendungsschnittstelle Blockschaltbild [WIE07]

Die Eingangsports sind über Register geführt, damit sich die Laufzeiten von Video- und DDR-Controller nicht addieren und ein definiertes Timing gewährleistet ist.

Auf der linken Seite von Abbildung 40 sind die für den Video-Controller relevanten Ports zu sehen. In Tabelle 12 sind die Eingangskombinationen von *u_wr_req_i* und *u_rd_req_i* und die drei durch sie dargestellten Befehle Schreiben, Lesen und NOP (No Operation) angeführt.

Tabelle 12: Befehlssignale an den Controller [WIE07]

Signal	Schreiben	Lesen	NOP	Ungültiger Befehl
u_wr_req_i	1	0	0	1
u_rd_req_i	0	1	0	1

Für das Adresssignal sind die Ports *u_ba_i*, *u_row_i* und *u_col_i* vorgesehen und repräsentieren Bank-, Zeilen- und Spaltenadresse. Sie sind zwecks Unterscheidbarkeit der Zuordnung getrennt geführt.

Die vom Video-Controller abgesetzten Befehle und Schreibdaten werden in ein FIFO (First In First Out) übernommen, wobei zwischen dem Befehls- und dem Schreibdaten-FIFO unterschieden wird. Das bedeutet für den Entwurf der modifizierten Speicherschnittstelle, dass die Reihenfolge der abgesetzten Lesebefehle gespeichert werden muss, um nach Erhalt der Daten diese korrekt weiterleiten zu können. Der Controller unterstützt Burstlängen von 2, 4 oder 8, d. h. bei einem Lese- bzw. Schreibbefehl können 2, 4 oder 8 x 64 Bit bewegt werden. Bei einem Schreibbefehl mit der Burstlänge = 8 wird ein Befehl in das Befehls-FIFO aufgenommen, sowie je ein Satz Schreibdaten zu vier aufeinander folgenden Taktzyklen.

Der Ausgang *u_ready_o* signalisiert, ob das Befehls-FIFO bereit ist, einen Befehl aufzunehmen. Erst bei *u_ready_o = ,1'* kann der Video-Controller einen Befehl absetzen, dabei wird auch die Burstlänge berücksichtigt, um gewährleisten zu können, dass der vollständige Befehl ins FIFO passt. Ist dies nicht der Fall, ist das FIFO voll und es muss gewartet werden. Der Befehl NOP wird allerdings von der Schnittstelle ignoriert und nicht im FIFO abgelegt.

u_ctrl_initdone_o = ,1' signalisiert den Abschluss der Initialisierung des Speichers und gibt an, dass der DDR-Controller bereit ist.

Bei einem Schreibbefehl liegen die Daten an *u_wrdata_i* und die Maskierungsbits an *wr_dm_i*, wobei je ein Maskierungsbit für jeweils ein Byte der Schreibdaten vorgesehen ist (Tabelle 13).

Tabelle 13: Maskierungsbits [WIE07]

Maskierungsbit	Schreibdatenbyte
0	Datenbyte wird geschrieben
1	Datenbyte wird ausmaskiert

Mit *rd_data_valid_o = ,1'* signalisiert der DDR-Controller, dass am Ausgang *rd_data_o* gültige Daten anliegen und von der Anwendung übernommen werden können.

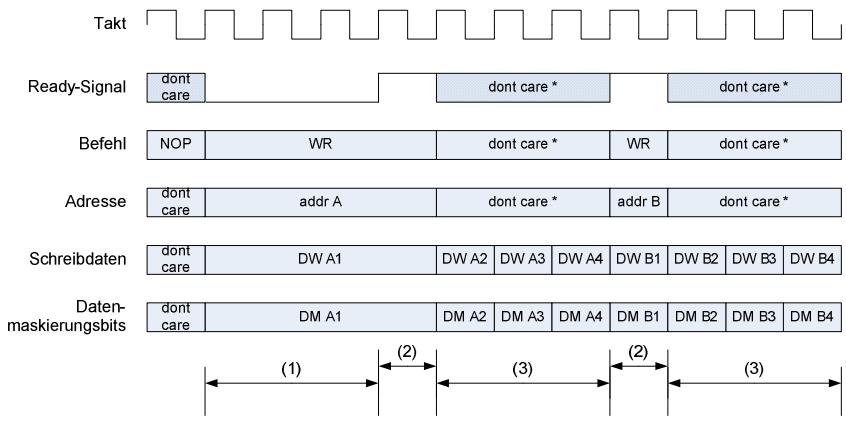

Abbildung 41: Kommunikation mit dem DDR-Controller – Schreiben [WIE07]

Abbildung 41 stellt die Kommunikation zwischen der Anwendung und dem DDR-Controller mit einer eingestellten Burstlänge = 8 dar. Zu Beginn ist der Controller noch nicht bereit, einen Befehl anzunehmen und signalisiert das, indem das Ready-Signal ‚0' ist (1). Erst ab (2) ist der DDR-Controller bereit und akzeptiert den Schreibbefehl. Danach folgen drei Takte(3) in denen weiter Daten und Maskierungsbits in die entsprechenden FIFOs übernommen werden.

In Abbildung 42 ist ein Lesebefehl an den DDR-Controller mit der Burstlänge = 8 dargestellt. Nach (1) ist der Controller wieder bereit, einen Befehl anzunehmen (2). Nach einer bestimmten Anzahl von Takten (3), hier nicht maßstäblich dargestellt, signalisiert der DDR-Controller mit Data Valid die Gültigkeit der Daten am Ausgang (4). Danach folgen drei weitere Datenausgaben.

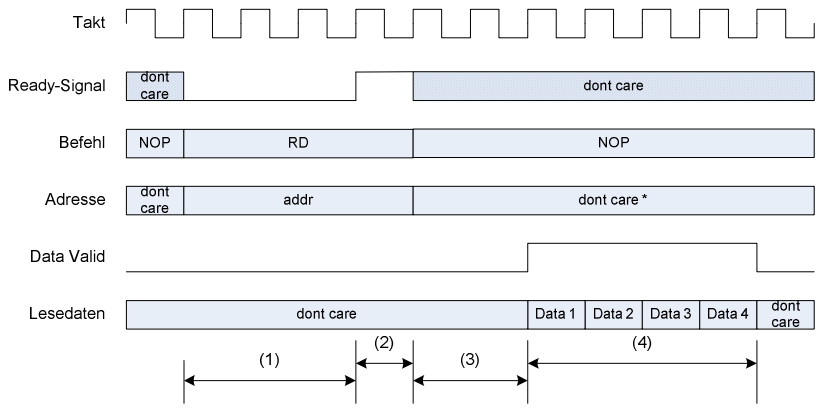

Abbildung 42: Kommunikation mit dem DDR-Controller - Lesen

6. Modifikation des Video-Controllers

Durch die unterschiedliche Art der Zugriffe auf die beiden Speichertypen muss die Schnittstelle zum Speicher geändert werden, die übrigen Einheiten des Video-Controllers sollen aber unverändert verwendet werden. Dieses Kapitel beschreibt die Modifikation der Schnittstelle des bestehenden Video-Controllers, um ein DDR-SDRAM statt des SRAMs als Bildspeicher verwenden zu können.

6.1 Aufbau der modifizierten Schnittstelle

Im bestehenden Video-Controller konnte durch die Verwendung eines SRAMs, durch definierte Antwortzeiten bei Lesebefehlen, ein Zeitschlitzverfahren angewandt werden. Bei DDR-SDRAMs hingegen ist nicht definiert, zu welcher Zeit ein Speicherbefehl ausgeführt wird, da möglicherweise gerade zu dieser Zeit ein Refresh-Zyklus ausgeführt wird oder ein Bank- oder Zeilenwechsel erfolgt.

Durch die Verwendung eines DDR-SDRAMs ist die Verwendung eines Speicher-Controllers notwendig, der die Schreib- und Lesebefehle des Video-Controllers auf die entsprechenden DRAM-Befehle übersetzt und im Hintergrund das Auffrischen der Speicherinhalte durchführt.

Die Arbiterlogik muss angepasst werden, da statt eines fixen Zeitschlitzverfahrens durch die unbestimmte Dauer des Zugriffs auf den DDR-Speicher, Daten vorgeladen werden müssen.

In den Folgeeinheiten muss sodann eine Umsetzung der breiten DDR-SDRAM Daten (128 Bit) auf die schmalen internen Datenwortbreiten (16 Bit) des Video-Controllers erfolgen. All diese Änderungen sollen aber so erfolgen, dass die Schnittstellen außer zum Speicher unverändert bleiben.

In Abbildung 43 sind die Module der modifizierten Speicherschnittstelle dargestellt. Lesezugriffe der zeitkritischen Daten für den Bildaufbau müssen rechtzeitig initiiert werden um eine zeitgerechte Ausgabe der Bildpunkte zu sichern. Deshalb wird in der modifizierten Schnittstelle ein eigenes Modul Bildschirm vorgesehen, um die Bilddaten für die Einheit Bildschirmaufbau aus dem DDR-Speicher anzufordern und bereitzuhalten.

Alle weiteren Lesezugriffe der Einheiten Signaltöne und Zeichenoperationen werden vom Modul Lesen bearbeitet. Dieses soll bei einem Lesebefehl jenen 128-Bit-Datenblock aus dem Speicher lesen, in dem das adressierte 16-Bit-Datenwort enthalten ist und die Daten an die entsprechende Einheit des Video-Controllers ausgeben. Zusätzlich müssen gleichzeitig auftretende Zugriffswünsche der Einheiten Signaltöne und Zeichenoperationen auf den Speicher koordiniert werden.

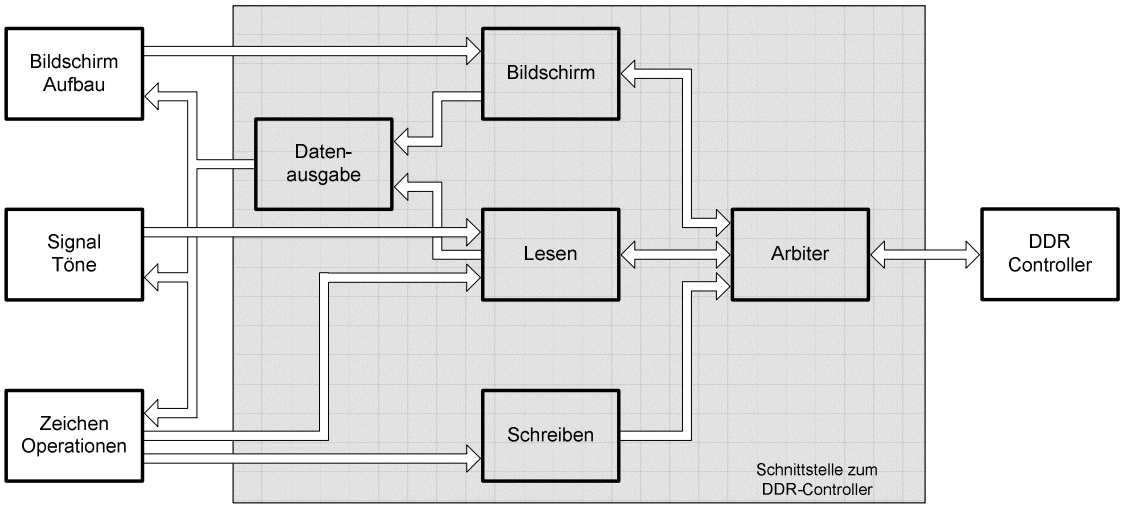

Abbildung 43: Struktur der modifizierten Schnittstelle

Das Modul Schreiben nimmt die Schreibbefehle der Einheit Zeichenoperationen mit 16-Bit-Daten entgegen. Um die Anzahl der Schreibbefehle zu reduzieren, werden bei gleichen DDR-Speicheradressen 128-Bit-Schreibbefehle zusammengesetzt. Da sich die Schreibbefehle der Einheit Zeichenoperationen des Video-Controllers nicht verzögern lassen, ist das Puffern von Schreibbefehlen, während nicht auf den Speicher zugegriffen werden kann, eine weitere Aufgabe dieser Einheit.

Die genannten Module greifen über den Arbiter auf den DDR-Controller zu. Er wählt bei gleichzeitigen Zugriffen eine Einheit nach festen Prioritäten aus, während die anderen Einheiten mittels Haltesignalen angehalten werden. Bei Lesezugriffen wird gespeichert, von welcher Einheit der Befehl stammt, um die eingehenden Lesedaten der aufrufenden Einheit korrekt zuordnen zu können.

Der ursprüngliche Video-Controller verwendet zur Ausgabe der Lesedaten ein gemeinsames Ausgabeport. Um dies beizubehalten, ist das Modul Datenausgabe vorgesehen. Bei gleichzeitig auftretenden Datenausgaben der Komponenten Bildschirm und Lesen gewährt dieses Modul den zeitkritischen Bildschirmdaten Vorrang gegenüber den anderen Lesedaten.

Im Weiteren werden die einzelnen Module im Detail behandelt.

6.2 Komponente Bildschirm

Die Komponente Bildschirm hat zwei Aufgaben zu erfüllen. Einerseits nimmt sie die 16-Bit-Lesezugriffe der Komponente Bildschirmaufbau des Video-Controllers entgegen, liest Datenblöcke von 128 Bit aus dem DDR-SDRAM und gibt die angeforderten 16-Bit-Daten an den Bildschirmaufbau aus. Andererseits müssen die Bilddaten rechtzeitig, vor ihrer Verwendung im Bildschirmaufbau, aus dem DDR-Speicher gelesen und zwischengespeichert werden, um eine Abkopplung zwischen den nichtdeterministischen Lesezugriffen auf den DDR-Speicher und den zeitkritischen Lesebefehlen des Bildschirmaufbaues zu erreichen.

Bei der Darstellung einer Bildschirmseite ist vorhersehbar, welche Daten aus dem Videospeicher benötigt werden. In der Komponente Bildschirm werden diese Daten im Vorhinein aus dem DDR-Speicher gelesen und zwischengespeichert, um rechtzeitig für den Bildschirmaufbau zur Verfügung zu stehen. Das Nachladen der nächsten Zeile erfolgt dann in der Austastpause, oder, bestimmt durch einen Parameter, schon während der Darstellung der aktuellen Zeile.

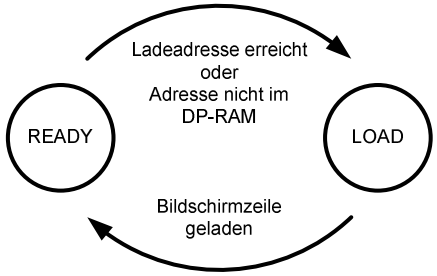

Abbildung 44: Zustandswechsel der Komponente Bildschirm

Am Ende einer Zeile, in der Austastpause, beginnt dieses Modul autonom die nächste Bildschirmzeile aus dem DDR-Speicher zu lesen. Ausgehend vom Zustand READY wird in den Zustand LOAD gewechselt (Abbildung 44) und es werden 50 Lesebefehle an das DDR-SDRAM für je 16 Pixel gestellt. Die vom Speicher ausgegebenen Daten werden in einem Dualport-RAM zwischengespeichert. Nach dem Eintreffen des letzten 128-Bit-Blockes ist der Lesevorgang abgeschlossen und das Modul kehrt in den Zustand READY zurück. Setzen die Lesezugriffe der Einheit Bildschirmaufbau wieder ein, können die Bilddaten aus dem Zwischenspeicher ausgegeben werden.

Der Zeitpunkt, wann das Nachladen der nächsten Zeile erfolgt, wird durch einen Parameter *g_reload_at* festgelegt. Dies kann am Ende einer Zeile erfolgen (*g_reload_at = 800*), aber auch schon eher, je nachdem, wie lange der DDR-Controller zum Nachladen einer Zeile benötigt. Zu diesem Zweck müssen zwei Speicherbereiche im Dualport-RAM verwaltet werden, der restliche Bereich der in Darstellung befindlichen Zeile und der Speicherbereich der nächsten Zeile. Die nachfolgende Abbildung 45 soll den Vorgang des Nachladens während der Darstellung einer Zeile verdeutlichen, als Beispiel wurde *g_reload_at = 768* gewählt. Wird also beim Auslesen der Daten einer Zeile der 768. Punkt aus dem Dualport-RAM angefordert, beginnt der Ladevorgang der nächsten Zeile.

In Abbildung 45 (a) ist die erste Zeile des Bildschirminhaltes geladen, die 800 Bildpunkte der Zeile sind in 50 Blöcken zu je 16 Pixel im Dualport-RAM gespeichert. Die Grenzen des geladenen Speicherbereiches sind in *first_loaded_addr1* und *latest_loaded_addr1* festgehalten.

Wird nun Punkt 768, dieser findet sich in BLOCK 48, gelesen, wird der Ladevorgang der nächsten Zeile gestartet. Der restliche Speicherinhalt der aktuellen Zeile bleibt im Dualport-RAM, der Bereich mit den schon gelesenen Bilddaten wird zum Überschreiben freigegeben. Dementsprechend wird die Anfangsadresse des Speicherbereiches 1 auf die aktuelle Leseposition BLOCK 48 gesetzt.

Mit jedem Block, der vom Speicher eintrifft, wird die Speichergrenze *latest_load_addr2* auf die nächste Speicheradresse gesetzt und die Daten zum Lesen freigegeben.

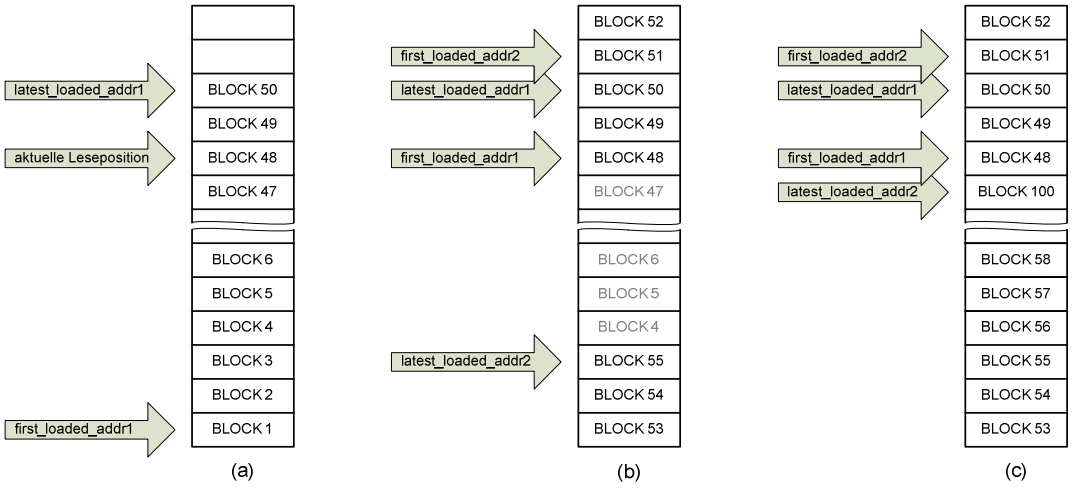

Abbildung 45: Nachladen des Dualport-RAMs

In Abbildung 45 (b) sind bereits fünf Datenblöcke vom Speicher eingelangt, in Abbildung 45 (c) sind alle angeforderten Datenblöcke vom Speicher eingetroffen und der Ladevorgang ist abgeschlossen. Im weiteren Verlauf wird bei jedem Lesezugriff überprüft, ob sich das adressierte 16-Bit-Wort im Dualport-RAM befindet, ist dies nicht der Fall, werden ab der angeforderten Adresse die Daten von einer Zeile mit 800 Pixel in das Dualport-RAM geladen, andernfalls erfolgt die Ausgabe der entsprechenden Daten.

Es gibt zwei Möglichkeiten, warum Daten bei einem Lesezugriff nicht im Zwischenspeicher sind. Erstens bei einem Wechsel auf eine andere im Speicher befindliche Bildschirmseite, oder zweitens zu Beginn beim ersten Laden einer Zeile.

Beim ersten Lesezugriff der Einheit Bildschirmaufbau des Video-Controllers wird in den Zustand LOAD gewechselt und die erste Zeile des Bildschirms in das Dualport-RAM geladen. Da die Bildschirmausgabe nicht angehalten werden kann, die Bilddaten aber noch nicht in den Zwischenspeicher geladen wurden, können die ersten Punkte der Zeile noch nicht dargestellt werden. Es wäre auch möglich gewesen, dieses Modul so zu gestalten, dass vor dem Beginn der Darstellung der ersten Zeile die Daten in das Dualport-RAM geladen werden. Dazu müsste man allerdings die Basisadresse der Bildschirmseite im Videospeicher aus der Registereinheit auslesen. Da aber die Ports der Schnittstelle zum Video-Controller bei der Modifikation unverändert bleiben sollten, wurde dieser Ansatz verworfen. Die Anzahl der Punkte, die im ersten Bild nicht dargestellt werden, hängt von der Geschwindigkeit des DDR-Controllers ab. Bei dem verwendeten Controller sind dies sechs Bildpunkte der ersten Zeile des ersten dargestellten Bildes. Im zweiten Bild ist dies nicht mehr der Fall, da nach der letzten Zeile die erste Zeile des nächsten Bildes vorgeladen wird und die Bilddaten rechtzeitig zur Verfügung stehen.

Sobald die ersten Daten aus dem Speicher eingetroffen sind, können diese an den Video-Controller ausgegeben werden, nach Eintreffen der letzten Daten der Zeile wird in den Zustand READY gewechselt.

6.2.1 Struktur der Komponente Bildschirm

In Abbildung 46 ist der strukturelle Aufbau der Komponente Bildschirm zu sehen.

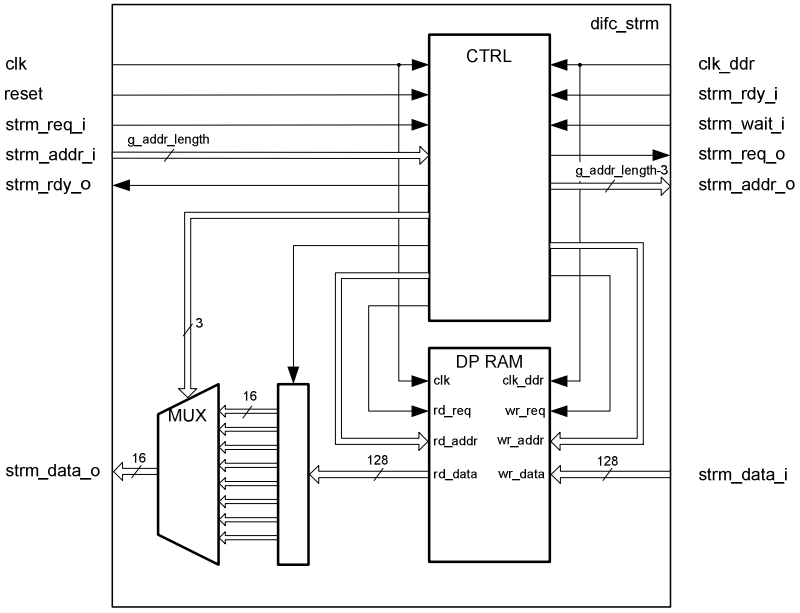

Abbildung 46: Struktur der Komponente Bildschirm

Die Steuerlogik (CTRL) wertet die von der Einheit Bildschirmaufbau eingehenden Lesebefehle aus, sorgt für das Laden der Bilddaten aus dem DDR-SDRAM in den Zwischenspeicher und gibt die angeforderten Daten aus. Sie wird durch vier Steuerprozesse gebildet:

Prozess LOAD_CTRL: Startet den Ladevorgang einer Bildschirmzeile.

Prozess LOAD_LINE: Generiert die Lesebefehle an den DDR-Speicher.

Prozess WRITE_DPRAM: Schreibt die aus dem DDR-SDRAM eingehenden Lesedaten in das Dualport-RAM.

Prozess READ_DPRAM: Liest die Bildschirmdaten aus dem Dualport-RAM und gibt sie an die Einheit Bildschirm aus.

Als Zwischenspeicher wird ein Dualport-RAM verwendet, ein Speicher mit zwei Zugangsports, die mit unterschiedlichen Frequenzen betrieben werden können. Das in diesem Modul verwendete Dualport-RAM weist in Richtung des Video-Controllers einen Ausgang auf, der mit 81 MHz getaktet wird und einen Dateneingang in Richtung des DDR-Speichers, der mit einer Taktfrequenz von 133 MHz betrieben wird. Auf diese Weise kann der Taktfrequenzübergang vom Video- zum DDR-Controller bewältigt werden.

Der adressierte 128-Bit-Block wird beim Lesezugriff vom Dualport-RAM in ein Register geladen, mit einem Multiplexer das adressierte 16-Bit-Datenwort ausgewählt und an den Bildschirmaufbau ausgegeben.

Die nachfolgende Tabelle 14 gibt die Ein- und Ausgänge und deren Verwendungszweck an.

Tabelle 14: Ein- und Ausgänge der Einheit Bildschirm

Port	Bedeutung
clk	81-MHz-Taktsignal des Video-Controllers
reset	Reset-Signal
strm_req_i	Signalisierung des Lesebefehls vom Video-Controller
strm_addr_i	Adresse des zu lesenden 16-Bit-Wortes
strm_rdy_o	Daten an strm_data_o gültig
strm_data_o	Datenausgang
clk_ddr	133-MHz-Taktsignal des DDR-Controllers
strm_rdy_i	Daten vom DDR-SDRAM an strm_data_i gültig
strm_wait_i	Haltesignal, keine Befehle absetzbar
strm_req_o	Signalisierung eines Lesebefehls an den DDR-Controller
strm_addr_o	Leseadresse des 128-Bit-Blockes aus dem DDR-SDRAM
strm_data_i	Dateneingang

6.2.2 Starten des Ladevorganges

Die Einheit für den Bildschirmaufbau des Video-Controllers signalisiert einen Lesebefehl über den Eingang *strm_req_i = ‚1'* zur steigenden Taktflanke. Der Prozess LOAD_CTRL der Steuerlogik prüft, ob sich die angeforderten Daten im Dualport-RAM befinden und leitet gegebenenfalls den Ladevorgang ein (Abbildung 47). Zu diesem Zweck wird die Adresse am Eingang *strm_addr_i* mit den in Registern gespeicherten Adressgrenzen, der im Dualport-RAM befindlichen Daten verglichen.

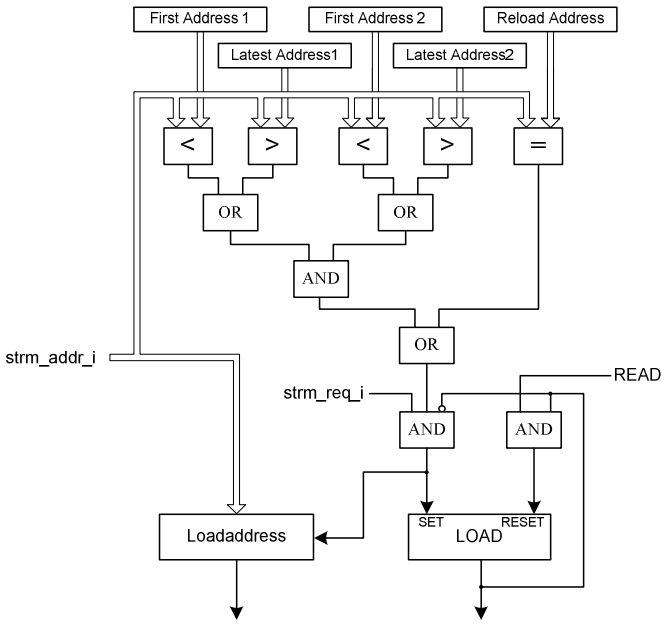

Abbildung 47: Steuerprozess LOAD_CTRL

Liegt der adressierte Speicherbereich außerhalb dieser Grenzen oder entspricht die Adresse jener, bei der nachgeladen werden soll, wird das Register LOAD = ‚1' gesetzt und der Ladevorgang ausgelöst.

Mit dem Signal *READY* signalisiert der Prozess RD_DPRAM, dass der Ladevorgang abgeschlossen ist und die Einheit in den Zustand READY zurückkehrt (Register LOAD = ‚0').

6.2.3 Anfordern der Daten aus dem DDR-Speicher

In Abbildung 48 ist das Prinzip, nach dem der Prozess LOAD_LINE die Daten aus dem DDR-Speicher anfordert, dargestellt. Der Ladevorgang wird durch Setzen des Registers LOAD, durch den vorher beschriebenen Prozess LOAD_CTRL ausgelöst. Zur nächsten steigenden Taktflanke des 133-MHz-Taktsignals erkennt der Prozess den Zustandswechsel und beginnt mit dem Laden des Zählers (CTR) mit der im Register LOADADDRESS befindlichen Startadresse. Diese Adresse wird gemeinsam mit dem Signal *strm_req_o = ‚1'* ausgegeben und so der erste Lesebefehl signalisiert. Zur nächsten steigenden Taktflanke signalisiert *strm_wait_i = ‚0'*, dass der Lesebefehl akzeptiert wurde und der nächste Befehl absetzbar ist.

Nach jedem erteilten Lesebefehl wird der Adresszähler um eins erhöht und der nächstfolgende Block angefordert. Dieser Vorgang wird solange wiederholt, bis die Adresse des letzten Lesebefehls der Zeile erreicht wurde, diese ist im Register STOP ADDRESS festgelegt. Nach dem letzten Lesebefehl wird das Signal *strm_req_o* zurückgesetzt.

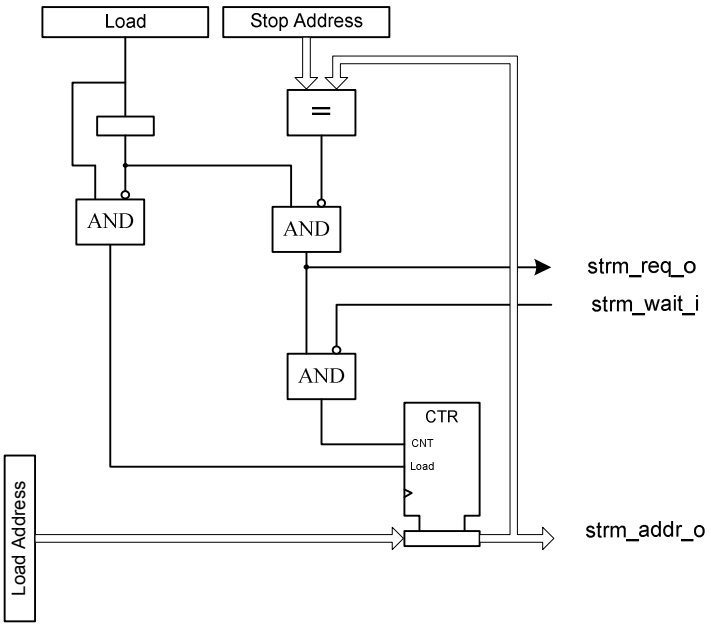

Abbildung 48: Anfordern der Daten aus dem DDR-Speicher

6.2.4 Schreiben der Bilddaten in das Dualport-RAM

Der Prozess READ_DPRAM (Abbildung 49) der Steuerlogik schreibt die eintreffenden Daten aus dem DDR-SDRAM in das Dualport-RAM. Der Eingang *strm_rdy_i = ,1'* signalisiert zur steigenden Taktflanke die Gültigkeit der Daten am Dateneingangsport *strm_data_i*. Die empfangenen Daten werden an die nächste freie Adresse im Dualport-RAM geschrieben. Die Adresse wird durch einen Zähler CTR 1 erzeugt, sodass der Speicher zyklisch beschrieben wird.

Der zweite Zähler CTR 2 hat die Aufgabe, die vom Speicher eingetroffenen Datenblöcke zu zählen. Er wird zu Beginn des Zustandes LOAD auf Null gesetzt. Nach jedem eingelangten Datenblock wird der Zähler um eins erhöht und mit dem im Register REQUESTED gespeicherten Wert verglichen, welcher der Zahl der abgesetzten Lesebefehle entspricht. Wurde die angeforderte Anzahl von Blöcken vom DDR-Speicher empfangen, wird mit dem Signal *Ready = ,1'* das Ende des Ladevorgangs signalisiert, um in den Zustand READY zurückzukehren.

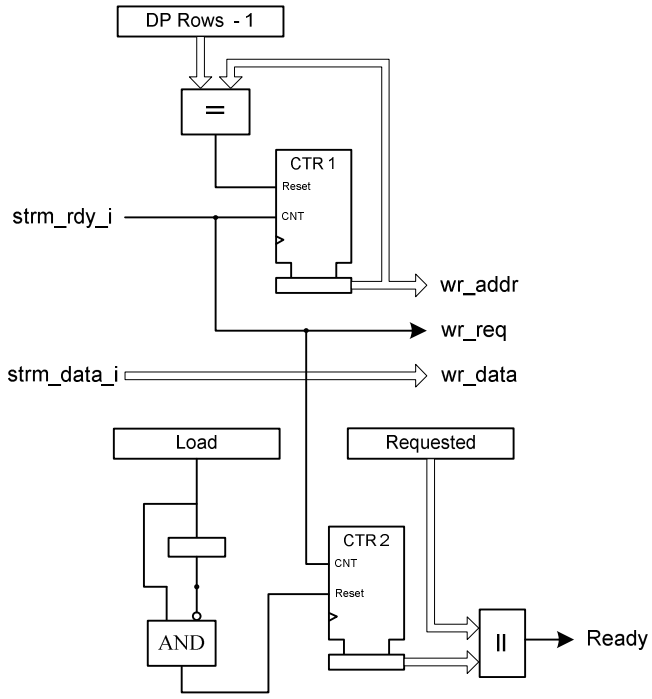

Abbildung 49: Schreiben des Dualport-RAMs durch den Prozess WRITE_DPRAM

6.2.5 Datenausgabe an die Einheit Bildschirm

Bei einem Lesebefehl (*strm_req_i = ,1'*) wird überprüft, ob die Daten bereits im Dualport-RAM geladen sind. Ist dies der Fall, können sie ausgelesen werden. Dabei muss vorher die richtige Adresse im Dualport-RAM berechnet werden. Der Steuerprozess READ_DPRAM bestimmt, wie in Abbildung 50 dargestellt, den Bereich, in dem die Daten enthalten sind. Der Multiplexer MUX 1 wählt die Beginnadresse dieses Bereiches aus und subtrahiert diese von der Adresse des Lesebefehls. Man erhält somit die relative Adresse bezüglich des Anfangs des im Dualport-RAM eingelagerten Speicherbereiches. Im nächsten Schritt wird dessen Startadresse im Dualport-RAM mit dem Multiplexer MUX 2 ausgewählt

(First DP-Add. 1 oder First DP-Add. 2) und addiert. Das Ergebnis ist die Dualport-RAM-Adresse, unter der die Daten zu finden sind.

Die so erzeugte Adresse wird an den Adresseingang *rd_addr* gemeinsam mit *rd_req = ‚1'* an das Dualport-RAM geführt, welches einen Takt später die Daten am Ausgang *rd_data* liefert. Die Daten werden anschließend, wie in Abbildung 46 bereits dargestellt, in ein Register übernommen und das adressierte 16-Bit Wort mit dem Multiplexer am Ausgang *strm_data_o* in das 128-Bit-Register ausgegeben.

Das Ready-Signal am Ausgangsport *strm_rdy_o* wird aus dem durch ein dreistufiges Schieberegister verzögerten Signal *strm_req_i* erzeugt.

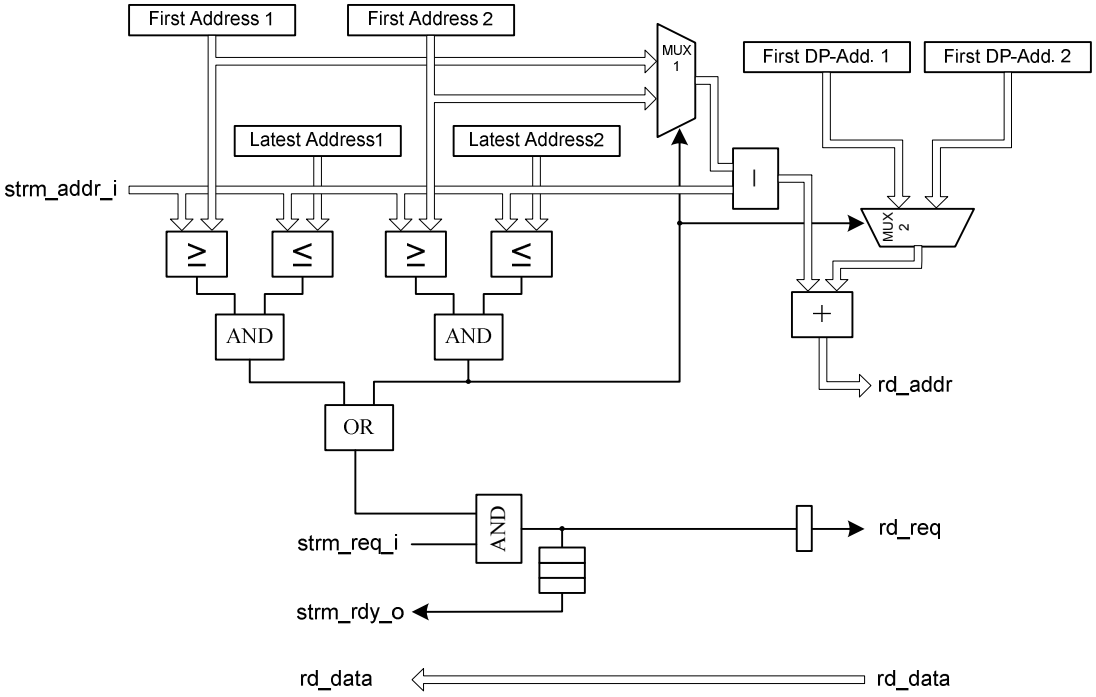

Abbildung 50: Datenausgabe durch den Prozess READ_DPRAM

6.2.6 Dimensionierung des Dualport-RAMs

Nun stellt sich die Frage nach der Größe des Dualport-RAMs. Sinnvoll ist jene Größe zu wählen, die benötigt wird, um eine Zeile mit 800 Bildpunkten zwischenzuspeichern. Nach der Darstellung einer Zeile kann man anschließend in der Austastpause die nächste Zeile in den Zwischenspeicher laden. Zunächst muss überprüft werden, ob die Dauer der Austastpause für das Laden einer Zeile ausreicht.

Die gesamte Länge einer Zeile besteht aus den 800 Bildpunkten und weiteren 224 Pixeltakten, zu denen keine Pixel am Bildschirm dargestellt werden, dies ist die Austastpause. Da der Pixeltakt mit halber Taktfrequenz arbeitet, beträgt die Austastpause 448 Takte. Somit ergibt sich für die Austastpause:

$$t_{Austastpause} = 448 * \frac{1}{81\,MHz} = 448 \times 12\,200\,ps = 5\,465\,600\,ps$$

Daraus ergeben sich bei der Taktfrequenz von 133 MHz des DDR-Controllers:

$$N_{133MHz} = 5\,465\,600\ ps * 133\ MHz = 726{,}92 \approx 726\,Takte$$

die für das Laden einer Bildschirmseite zur Verfügung stehen.

Im nächsten Schritt muss nun berechnet werden, wie lange der Ladevorgang der Daten vom DDR-SDRAM in das Dualport-RAM längstens dauert. Dazu ist vom ungünstigsten Ausgangszustand des DDR-Controllers auszugehen.

Es wird angenommen, dass das Befehls-FIFO des DDR-Controllers vollständig gefüllt ist, das Ready-Signal wurde zurückgenommen und somit alle Einheiten des Video-Controllers blockiert, sodass in jeder der Verarbeitungsstufen Befehle zwischengespeichert wurden. Zusätzlich soll angenommen werden, dass es sich bei den aufgenommenen Befehlen nur um jenen Befehl mit der längst möglichen Ausführungszeit handelt. Der am längsten dauernde Befehl an den DDR-Controller ist ein Schreibbefehl im wahlfreien Zugriff mit Zeilenwechsel innerhalb der gleichen Bank. Um eine Abschätzung der Dauer dieses Befehls vornehmen zu können, werden die Timingparameter aus Tabelle 15 angenommen. Diese Werte wurden auch in [WIE07] für die Abschätzung des Datendurchsatzes verwendet und beziehen sich auf den in der Simulation verwendeten Speicher.

Tabelle 15: Zeitkonstanten des DDR-Controllers [JED03]

Zeitkonstante	Bedeutung	Erklärung	Takte
t_{RCD}	Row Column Delay	Wartezeit zwischen der Aktivierung einer Zeile und dem ersten Schreib- bzw. Lesebefehl auf diese Zeile	3
t_{RP}	Row Precharge	Minimale Zeit zwischen Precharge- und Activate-Befehl	3
t_{RFC}	Refresh Cycle Time	Für einen Refresh-Zyklus benötigte Zeit	10
t_{WR}	Write Recovery Time	Notwendige Zeit zwischen letzten Daten eines Schreibbefehls und dem Precharge-Befehl	2
t_{RAS}	Row Access Strobe	Minimale Zeit zwischen Activate- und Precharge-Befehl	6
t_{RC}	Row Cycle Time	Minimale Zeit zwischen zwei Activate-Befehlen auf dieselbe Bank	9

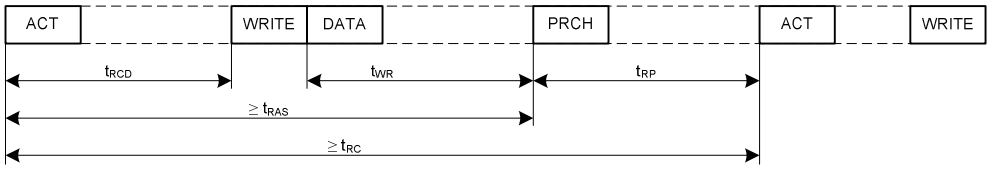

Abbildung 51: Timing-Bedingungen für Schreibbefehl mit Zeilenwechsel auf gleiche Bank

Mit den Zeitkonstanten und den Zeitbedingungen aus Abbildung 51 ist die Dauer des Schreibbefehls bestimmbar:

Aus $t_{RAS} = 6\,Takte \geq t_{RCD} + t_{WR} + 1 = 6\,Takte$

und $t_{RC} = 9\,Takte \geq t_{RAS} + t_{RP} = 9\,Takte$

folgt, dass ein Schreibbefehl im ungünstigsten Fall 9 Takte benötigt.

Nun stellt sich die Frage nach der Anzahl der Befehle, die auf ihre Abarbeitung im Befehls-FIFO des DDR-Controllers und in den einzelnen Stufen warten können.

Tabelle 16: Maximale Anzahl der wartenden Befehle

Stufe	Anzahl der Befehle
Befehl in Bearbeitung	1
FIFO	16
Pipelinestufen vor Control-Einheit	4
Bankauswahl	1
State Machine	1
State Machine geblocked	1
Gesamt	24

Bei den angenommenen Werten in Tabelle 16, für die Anzahl der Befehle die auf ihre Abarbeitung warten, handelt es sich beim DDR-Controller, um eine Einheit mit vielen Pipeline-Stufen. Insgesamt ergeben sich 24 gepufferte Befehle. Ein Befehl befindet sich im Eingangsregister des DDR-Controllers, maximal 16 Befehle können im Befehls-FIFO sein. Vom Befehls-FIFO zur Steuerlogik liegen vier Pipeline-Stufen vor und für einen Befehl wird geprüft, ob ein Bankwechsel vorliegt. Weiters wird ein Befehl in der zentralen Ablaufsteuerung des Speichercontrollers gepuffert und einer abgearbeitet.

Dazu sind noch zwei Registerstufen im Video-Controller zu berücksichtigen und unter der Annahme, dass noch ein Refresh-Zyklus, der in diesem Fall 10 Takte benötigt, anfallen könnte, ergeben sich:

$$N = 9\,Takte * 26\,Befehle + 10\,Takte = 244\,Takte$$

Die verfügbaren 726 Takte in der Austastpause sind also in jedem Fall ausreichend.

6.3 Komponente Lesen

Die Komponente Lesen koordiniert die Lesezugriffe der Einheiten Zeichenoperationen und Signaltöne des Video-Controllers auf den DDR-Speicher. Bei gleichzeitigen Speicherzugriffen werden die Lesebefehle der Signalisierungseinheit denen der Zeicheneinheit vorgezogen und diese mittels Haltesignal angehalten. Anschließend erfolgt die Ausgabe der 16 Bit Daten an die Einheit, von der der Lesebefehl ausging.

6.3.1 Struktur der Komponente Lesen

Der strukturelle Aufbau ist in Abbildung 52 dargestellt. Die Steuerlogik (CTRL) nimmt die 16-Bit Lesebefehle entgegen und lädt den adressierten 128-Bit-Block aus dem DDR-SDRAM in ein Register. Mit einem Multiplexer wird anschließend das adressierte 16-Bit-Wort ausgewählt und ausgegeben.

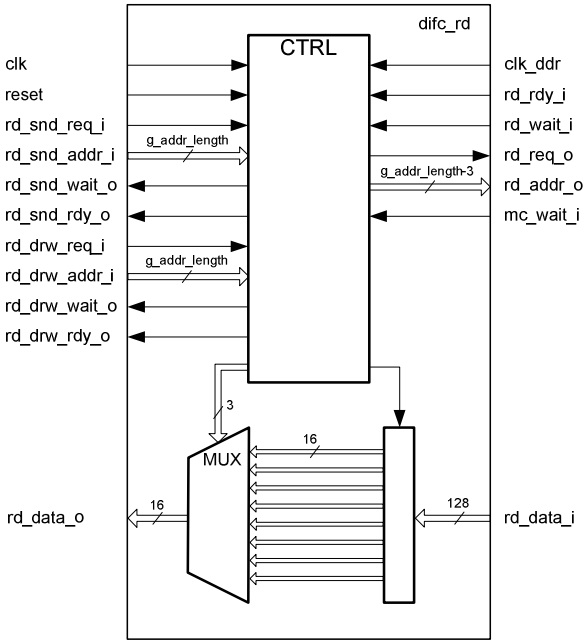

Abbildung 52: Struktur Komponente Lesen

Tabelle 17 zeigt die Anschlüsse der Einheit Lesen und deren Verwendung.

Tabelle 17: Ein- und Ausgänge der Einheit Lesen

Signal	**Verwendung**
clk	81-MHz-Taktsignal
reset	Reset-Signal
rd_snd_req_i	Lesebefehl der Signalisierungseinheit
rd_snd_addr_i	Leseadresse der Signalisierungseinheit
rd_snd_wait_o	Wait-Signal für die Signalisierungseinheit
rd_snd_rdy_o	Ready-Signal für die Signalisierungsein-heit
rd_drw_req_i	Lesebefehl der Zeicheneinheit
rd_drw_addr_i	Leseadresse der Zeicheneinheit
rd_drw_wait_o	Wait-Signal für die Zeicheneinheit
rd_drw_rdy_o	Ready-Signal für die Zeicheneinheit
rd_data_o	Datenausgabe
rd_req_o	Lesebefehl an den DDR-Speicher
rd_addr_o	Leseadresse an den DDR-Speicher
clk_ddr	133-MHz-Taktsignal
rd_rdy_i	Ready-Signal des DDR-Speichers
rd_wait_i	Wait-Signal des DDR-Speichers
mc_wait_i	Wait-Signal vom Arbiter
rd_data_i	Daten vom DDR-Speicher

Die Steuerlogik wird von folgenden Prozessen realisiert:

READ_WORD: Dieser Prozess nimmt Lesebefehle von den Einheiten Signaltöne und Zeichenoperationen entgegen und setzt die entsprechenden Prioritäten. Ist das adressier-

te Datenwort nicht im Register gespeichert, wird dies dem Prozess LOAD_REGISTER signalisiert und das Lesen der Daten veranlasst. Nach dem Eintreffen der Daten in das Register wird das adressierte 16-Bit-Datenwort ausgegeben.

LOAD_REGISTER: Benötigt der Prozess READ_WORD Daten aus dem DDR-SDRAM, stellt der Prozess LOAD_REGISTER den dafür erforderlichen Lesebefehl an den DDR-Speicher. Nach dem Eintreffen werden die Daten in das Register gespeichert und der Abschluss des Lesens dem Prozess READ_WORD signalisiert.

SET_WAIT: Erzeugt die Haltesignale für die Einheiten Zeichenoperationen und Signaltöne der Lesezugriffe des Video-Controllers.

6.3.2 Starten des Ladevorganges und Ausgabe der Daten

Der Steuerprozess READ_WORD ist in Abbildung 53 dargestellt. Die Einheiten Signaltöne und Zeichenoperationen signalisieren ihre Lesebefehle durch Setzen der Eingangsports *rd_snd_req_i* und *rd_drw_req_i = ‚1'*.

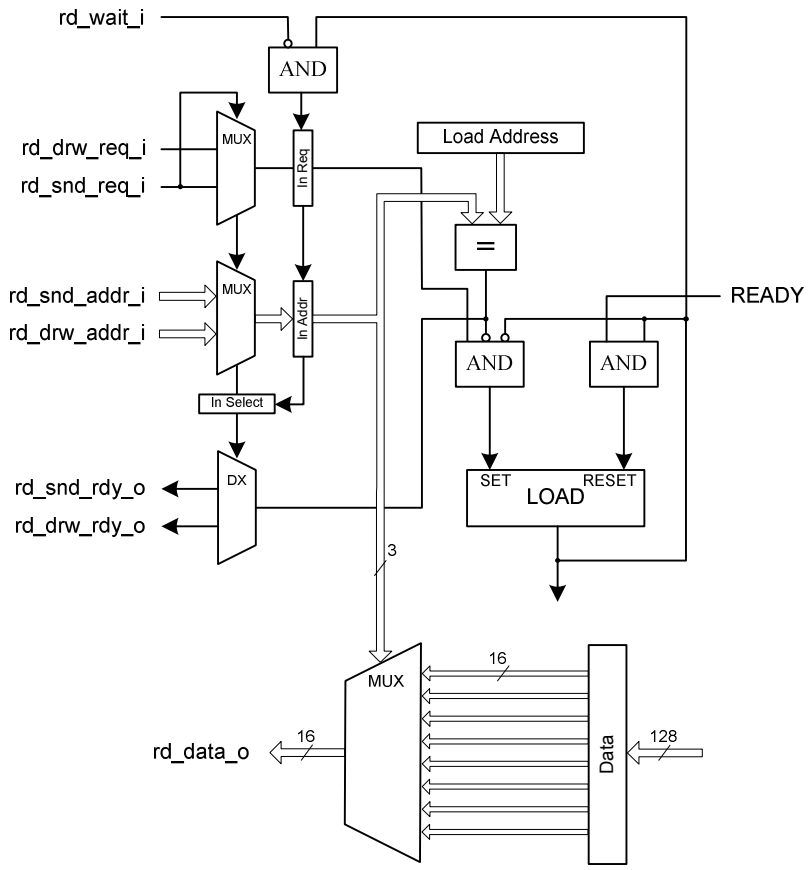

Abbildung 53: Steuerung des Ladevorganges und der Datenausgabe

Der Lesebefehl und dessen Adresse wird mit Multiplexer ausgewählt und in die Register IN_REQ und IN_ADDR übernommen. Bei gleichzeitigen Lesebefehlen wird der jener der Einheit Signaltöne ausgewählt. Anschließend wird die übernommene Adresse im Register IN ADDRESS mit der im Register LOAD ADDRESS, des aktuell gespeicherten 128-Bit-Blocks, verglichen. Stimmen beide Adressen überein, wird aus dem Register DATA das entsprechende 16-Bit-Wort ausgewählt und am Ausgang *rd_data_o* ausgegeben. Mit einem Demultiplexer erfolgt die Auswahl, welcher Einheit die Gültigkeit der Daten mit einem Ready-Signal bestätigt wird. Zu diesem Zweck wurde in Register IN SELECT gespeichert, von welcher Einheit der Befehl stammte.

Sind die angeforderten Lesedaten nicht gepuffert, muss der adressierte Block erst aus dem DDR-Speicher geladen werden. Der Prozess READ_WORD, mit 81 MHz getaktet, kann das Laden aufgrund der unterschiedlichen Taktfrequenz zum DDR-Controller nicht selbst durchführen. Er setzt deshalb das Register LOAD = ‚1' und signalisiert dem Prozess LOAD_REGISTER den Ladezustand. Dieser ist mit 133 MHz des DDR-Controllers getaktet und lädt die entsprechenden Daten aus dem DDR-SDRAM.

Mit dem Signal READY wird das erfolgreiche Laden der Daten von Prozess LOAD_REGISTER signalisiert. Der Prozess READ_WORD kehrt daraufhin mit LOAD = ‚0' wieder in den Zustand READY zurück und gibt die angeforderten Daten aus.

6.3.3 Laden der Daten aus dem DDR-SDRAM

Der Prozess LOAD_REGISTER (Abbildung 54) arbeitet mit 133 MHz und erkennt zur nächsten steigenden Taktflanke das Setzen des Registers LOAD. Er stellt an den DDR-Controller den Lesebefehl für den 128-Bit-Block, in dem die angeforderten Daten enthalten sind. Die Adresse dazu befindet sich im Register LOAD_ADDRESS.

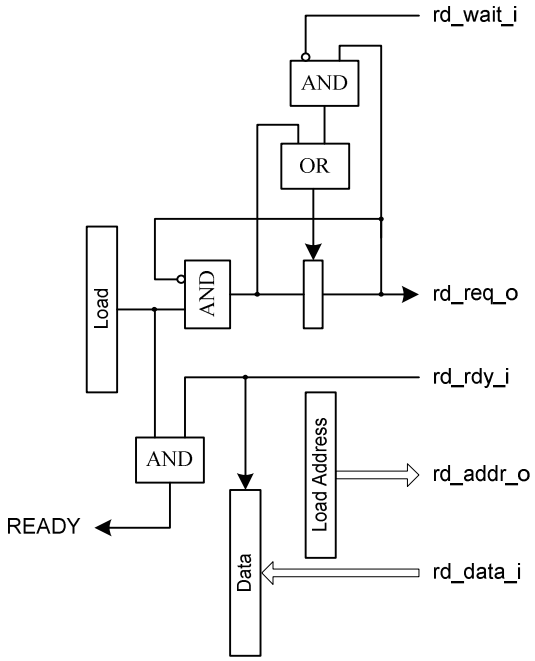

Abbildung 54: Laden des Registers Data

Der Ausgang *rd_req_o = ,1'* signalisiert einen Lesebefehl und wird nach einem Takt zurückgenommen, sofern das Haltesignal *rd_wait_i = '0'* ist. Nach dem Empfang der Daten aus dem Speicher, der vom DDR-Controller mit dem Signal *s_ready = ,1'* angezeigt wird, beschreibt der Prozess das Register DATA und signalisiert dem Prozess READ_WORD mit dem Signal *READY='1'* den Abschluss des Ladevorganges.

6.3.4 Setzen der Haltesignale

Vom Steuerprozess SET_WAIT wird zur fallenden Taktflanke überprüft, ob *rd_snd_req_i = ,1'* vorliegt. Ist dies der Fall, wird *rd_drw_wait_o = ,1'* gesetzt. Damit kann der Zeicheneinheit zur steigenden Taktflanke signalisiert werden, dass keine Lesebefehle angenommen werden können.

6.4 Komponente Schreiben

Die Komponente Schreiben nimmt die 16-Bit-Schreibbefehle der Zeichenoperationseinheit entgegen. Um die Anzahl der Schreibbefehle auf das DDR-SDRAM zu reduzieren, werden Schreibbefehle auf den gleichen DDR-Speicherblock in ein Register eingetragen und zu 128-Bit-Schreibzugriffen gesammelt. Bei unterschiedlicher DDR-Speicheradresse wird der Registerinhalt bestehend aus Adresse, Daten und Datamask-Bits in einen Pufferspeicher übernommen. Von dort werden die Daten hintereinander in den DDR-Speicher geschrieben, um zusätzliche Wartezeiten durch abwechselndes Lesen und Schreiben zu verhindern.

Bei einem Schreibbefehl der Einheit der Zeichenoperationen des Video-Controllers, wird das 16-Bit-Datenwort in das 128-Bit-Blockregister eingetragen. Mit den Daten werden auch die DDR-Speicheradresse und die Maskierungsbits der Eingänge *wr_bw1_i* und *wr_bw2_i* in ein Register eingetragen. Die Maskierungsbits signalisieren, ob das untere bzw. das obere Byte in den Speicher geschrieben werden soll. Bei einem nachfolgenden Schreibbefehl wird dessen DDR-Speicheradresse mit der im Register gespeicherten Adresse verglichen. Bei einer Übereinstimmung werden die Daten und die Maskierungsbits in das Register eingetragen. Unterscheiden sich die Adressen, wird der Inhalt des Registers in das Dualport-RAM geschrieben und das Register zurückgesetzt. Danach erfolgt wiederum der Eintrag ins Blockregister wie oben beschrieben.

Nach einer bestimmten Anzahl der im Dualport-RAM beschriebenen Zeilen wird der gesamte Inhalt in den DDR-Speicher übertragen. Die Anzahl der beschriebenen Zeilen, nach der dies erfolgt, ist mit einer Konstanten festgelegt.

Wie bereits erwähnt, steht dem ursprünglichen Video-Controller das SRAM in jedem Fall zur Verfügung und es treten keine Verzögerungen beim Schreiben der Daten auf. Dies wird auch vom modifizierten Video-Controller verlangt. Daher werden sämtliche Schreibbefehle der Zeicheneinheit in dieser Einheit gepuffert und die Zeit, in der kein Zugang zum DDR-Speicher möglich ist, überbrückt. Es müssen also genügend Zeilen im Dualport-RAM, das als Pufferspeicher eingesetzt wird, vorhanden sein.

Nach jedem Schreibbefehl wird eine, durch eine Konstante bestimmte Anzahl von Takten abgewartet, ob noch ein weiterer Schreibbefehl folgt, ist dies nicht der Fall, so wird das Blockregister entleert und

der Inhalt in das Dualport-RAM geschrieben. Anschließend wird der gesamte Inhalt des Dualport-RAMs in den DDR-Speicher geschrieben.

6.4.1 Struktur der Komponente Schreiben

In Abbildung 55 ist die Struktur der Komponente Schreiben dargestellt. Die Steuerlogik CTRL nimmt die 16-Bit-Schreibbefehle der Einheit Zeichenoperationen des Video-Controllers entgegen und trägt sie in ein Blockregister ein. Die Auswahl der Position, an der diese Eintragungen vorgenommen werden, erfolgt durch Multiplexer unter Verwendung der drei untersten Adressbits von *wr_addr_ i*.

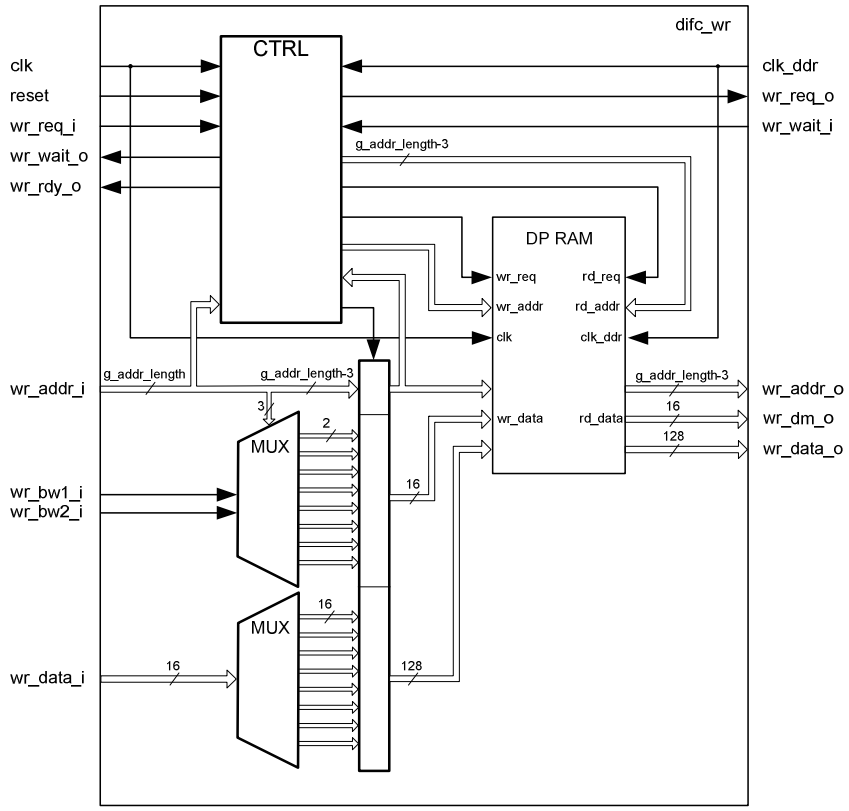

Abbildung 55: Struktur Komponente Schreiben

Bei unterschiedlicher DDR-Adresse veranlasst die Steuerlogik die Übernahme des Registerinhaltes in ein Dualport-RAM und entscheidet, wann die Daten in das DDR-SDRAM geschrieben werden.

Die Steuerlogik wird durch zwei Prozesse gebildet:

CTRL_BLOCKREGISTER: Führt die Einträge der 16-Bit-Schreibbefehle in das 128-Bit-Blockregister durch und schreibt den Registerinhalt in das Dualport-RAM. Weiters soll dieser Prozess das Ende einer Schreibsequenz erkennen und die Übertragung der Daten in den DDR-Speicher veranlassen.

WRITE_DDR: Dieser Prozess veranlasst das Schreiben der Daten vom Dualport-RAM in den DDR-Speicher.

Die folgende Tabelle 18 gibt die Ein- und Ausgänge der Einheit und deren Verwendungszweck an.

Tabelle 18: Ein- und Ausgänge der Einheit Schreiben

Port	Bedeutung
clk	81-MHz-Taktsignal
reset	Reset-Signal
wr_req_i	Write-Request-Signal vom Video-Controller
wr_addr_i	16-Bit-Wortadresse vom Video-Controller
wr_wait_o	Wait-Signal an die Zeicheneinheit
wr_rdy_o	Ready-Signal an die Zeicheneinheit
wr_bw1_i	Byte-Enable-Signal des Lower Byte
wr_bw2_i	Byte-Enable-Signal des Upper Byte
wr_data_i	16-Bit-Eingangsdaten vom Video-Controller
wr_wait_i	Wait-Signal vom DDR-Speicher
clk_ddr	Takt-Signal 133 MHz
wr_req_o	Schreib-Request
wr_addr_o	Blockadresse
wr_dm_o	Datamask-Bits
wr_data_o	128-Bit-Schreibdaten für den DDR-Speicher

6.4.2 Zusammenstellung der Schreibbefehle

In Abbildung 56 ist der Prozess CTR_BLOCKREGISTER dargestellt, mit der Aufgabe eingehenden Schreibbefehle in das Blockregister einzutragen und dessen Inhalt des Blockregisters in weiterer Folge in das Dualport-RAM zu schreiben. Außerdem erkennt dieser Prozess das Ende einer Folge von Schreibbefehlen und sorgt für die vollständige Übernahme der Schreibbefehle in den DDR-Speicher.

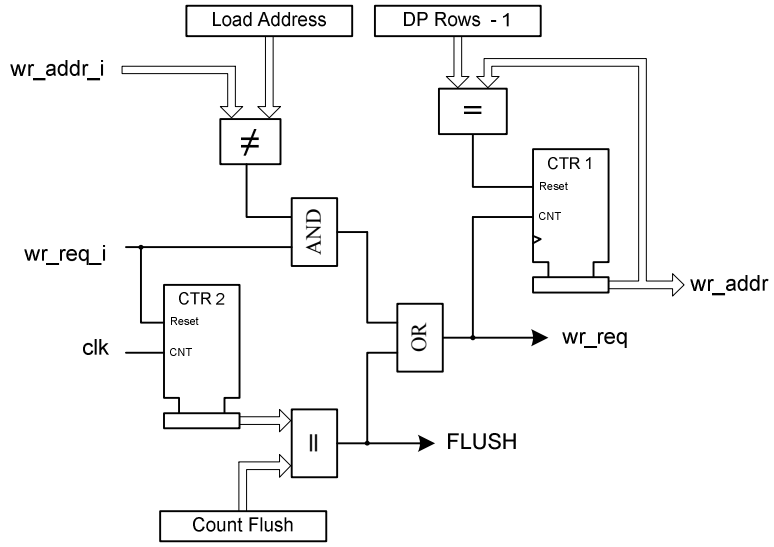

Abbildung 56: Übernahme des Registerinhalts in das Dualport-RAM

Die Zeicheneinheit signalisiert Schreibbefehle durch *wr_req_i = ‚1'* und es wird die Schreibadresse *wr_addr_i* mit der im Register LOAD_ADDRESS eingetragenen DDR-Speicheradresse verglichen. Unterscheiden sich die Adressen, wird der Inhalt des Blockregisters mit *wr_req = ‚1'* vom Dualport-

RAM übernommen. Die Adresse dafür wird von einem zyklischen Zähler CNT 1 erzeugt, der nach jedem Eintrag inkrementiert wird.

Um das Ende der Schreibbefehle erkennen zu können, wird der Zähler CNT 2 als Timer verwendet. Mit jeder steigenden Taktflanke des 81-MHz-Taktsignals wird der Zähler erhöht, bei jedem weiteren Schreibbefehl zurückgesetzt. Erreicht der Zähler den Wert im Register COUNT_FLUSH, wird dies als das Ende einer Schreibsequenz interpretiert. Daraufhin wird der Inhalt des Blockregisters in das Dualport-RAM geschrieben und anschließend mit dem Signal FLUSH dem Prozess WRITE_DDR signalisiert, die gesammelten 128-Bit-Schreibbefehle an den DDR-Speicher abzusetzen.

6.4.3 Schreiben der Daten in den DDR-Speicher

Der zweite Prozess WRITE_DDR der Steuerlogik ist in

Abbildung 57: Speicherung des Inhalts des Dualport-RAMs in den DDR-Speicher

veranschaulicht. Dieser Prozess hat die Aufgabe, die im Dualport-RAM gespeicherten 128-Bit-Schreibbefehle an das DDR-SDRAM abzusetzen. Dabei bestehen zwei Möglichkeiten wie dies ausgelöst wird, erstens durch das Signal FLUSH des Prozesses CTR_BLOCKREGISTER am Ende einer Schreibsequenz und zweitens durch Erreichen einer maximalen Zahl beschriebener Zeilen im Dualport-RAM.

Um zu gewährleisten, dass in Zeiten, bei denen kein DDR-Speicherzugriff möglich ist, alle Schreibbefehle gepuffert werden können, müssen genügend freie Zeilen im Dualport-RAM vorhanden sein. Die Anzahl der beschriebenen Zeilen, bei der der Übertragungsvorgang begonnen werden soll, kann als Parameter im Register DP_ROWS_WR angegeben werden. Um die Anzahl der beschriebenen Zeilen zu ermitteln, bildet der Steuerprozess laufend die Differenz zwischen der nächstfolgenden Schreib- und Leseposition im Dualport-RAM. Wird dabei der Wert von Register DP_ROWS_WR erreicht, werden die im Zwischenspeicher gespeicherten Schreibbefehle an den DDR-Speicher gestellt.

Für $vga_addr > ddr_addr$ gilt:

$$DPROWS - ddr_addr + vga_addr + 1 \leq DP_ROWS_WR$$

und für $vga_addr \leq ddr_addr_i$

$$vga_addr - ddr_addr \leq DP_ROWS_WR$$

wobei *vga_addr* die Schreibadresse des Dualport RAMs zum Register und *ddr_addr* die Leseadresse zum DDR-Controller ist.

Welche Berechnung schlussendlich vom Multiplexer der ausgewählt wird, hängt davon ab, welche Adresse größer ist.

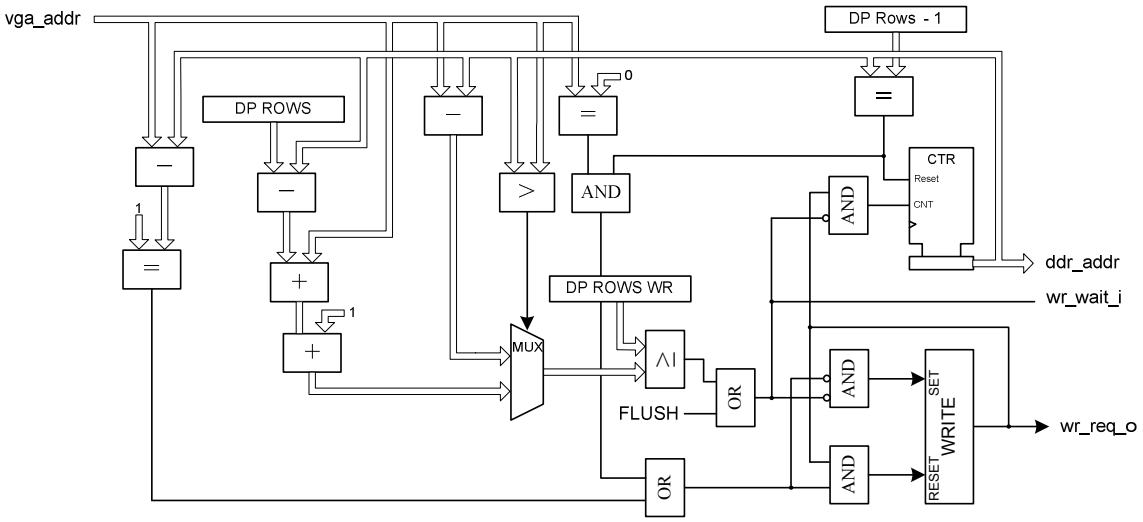

Abbildung 57: Speicherung des Inhalts des Dualport-RAMs in den DDR-Speicher

Ist eine der Bedingungen erfüllt, wird das Register WRITE gesetzt und mit *wr_req_o = '1'* ein Schreibbefehl signalisiert. Die zweite Möglichkeit, das Register WRITE zu setzen, ist, wenn der Prozess CTR_BLOCKREGISTER mit dem Signal FLUSH die Ausgabe der gespeicherten Schreibbefehle anordnet.

Der Zähler CTR erzeugt die Leseadresse für das Dualport-RAM. Dabei werden die Adressen zyklisch fortgesetzt. Immer wenn ein Lesebefehl akzeptiert worden ist (*wr_wait_i = ,0'* bei *wr_req_o = ,1')* zur steigenden Taktflanke, wird der Zähler inkrementiert.

Der Übertragungsvorgang ist abgeschlossen, wenn alle gepufferten Befehle aus dem Dualport-RAM an das DDR-SDRAM gestellt wurden, also die Leseadresse *ddr_addr* eine Adresse unter der nächsten Schreibposition *ddr_addr* des Dualport-RAMs ist:

für *vga_addr = 0* ist dies bei

$ddr_addr = DP_ROWS - 1$ erreicht,

anderenfalls bei

$vga_addr - ddr_addr = 1.$

Ist eine der beiden Bedingungen erfüllt, wird das Register WRITE gelöscht und der Schreibvorgang ist beendet.

6.4.4 Dimensionierung des Dualport-RAMs

In den Zeiten, in denen kein Speicherzugriff auf das DDR-SDRAM möglich ist, müssen eingehende 16-Bit-Schreibbefehle gepuffert werden. Nun sollen die erforderlichen Zeilen des Dualport-RAMs ermittelt werden.

Die Zeicheneinheit bearbeitet die Grafikoperationen immer in derselben Weise, egal ob Bilddaten kopiert werden oder ob gezeichnet werden soll. Vor jedem Schreibbefehl werden Daten von der Speicher-

adresse ausgelesen, manipuliert und wieder in den Speicher geschrieben. Aus Gründen der Einfachheit werden die Daten auch bei einem Zeichenbefehl gelesen, obwohl der Speicherinhalt in diesem Fall nicht relevant ist. Das bedeutet aber für Schreibbefehle, dass sie erst abgesetzt werden, nachdem die entsprechenden Daten vom Speicher eingetroffen sind. Daraus folgt aber weiter, dass die Obergrenze der Zeilen im Dualport-RAM, die man als Pufferspeicher benötigt, die Anzahl der bereits vor der Unterbrechung des Speicherzugangs abgesetzten Lesebefehle ist.

Das Maximum der abgesetzten Lesebefehle ergibt sich unter der Annahme, eines mit Lesebefehlen gefüllten Befehls-FIFO im DDR-Controller, den in den Registerstufen davor befindlichen Befehlen und den in den Registerstufen danach befindlichen Daten der bereits bearbeiteten Befehle. Die Anzahl der Zeilen entspricht dem Maximum der abgesetzten Befehle. Neue Lesebefehle müssen nicht berücksichtigt werden, da diese von den Schreibbefehlen durch ihre niedrigere Priorität blockiert werden und warten müssen.

In der konkreten Konfiguration können lt. Tabelle 16 im DDR-Controller 24 Lesebefehle aufgenommen worden sein. Dazu kommen zwei Registerstufen vom DDR-Controller bis zur Leseeinheit und zwei zusätzliche Zeilen für die aktuelle Bearbeitung in den Einheiten Lesen und Schreiben. In Summe sind also 28 Zeilen unbedingt erforderlich.

Gewählt wurde ein Dualport-RAM mit 32 Zeilen, also mit fünf Adressleitungen. In jeder Zeile werden für die Schreibdaten 128 Bit, für die Datamask-Bits 16 Bit und die Anzahl der Adressbits, in diesem Fall 17 Bit, der DDR-Speicheradresse benötigt.

6.5 Komponente Arbiter

Der Arbiter ermöglicht den Einheiten Bildschirm, Schreiben und Lesen der modifizierten Schnittstelle den Zugriff auf das DDR-SDRAM. Er wählt bei gleichzeitigen Zugriffen eine Einheit nach festen Prioritäten aus, während die anderen Einheiten mittels Haltesignalen angehalten werden. Außerdem wird bei Lesezugriffen gespeichert, von welcher Einheit dieser Befehl entgegengenommen wurde. Dadurch können die korrespondierenden, vom DDR-Speicher erhaltenen Lesedaten der aufrufenden Einheit zugeordnet werden.

6.5.1 Struktur des Arbiters

In Abbildung 58 ist die Struktur des Arbiters zum DDR-Controller zu sehen. Die Bedeutung der Ein- und Ausgänge sind am Ende dieses Kapitels in Tabelle 19 angeführt.

Die Steuerlogik nimmt die Lesebefehle der Einheiten Bildschirm und Lesen, sowie die Schreibbefehle der Einheit Schreiben entgegen und erteilt, im Falle einer Kollision, den DDR-Speicherzugriff nach folgenden Prioritäten:

1. Lesezugriff der Komponente Bildschirm
2. Schreibzugriff der Komponente Schreiben
3. Lesezugriff der Komponente Lesen

Der Komponente mit dem Speicherbefehl mit der höchsten Priorität wird der Zugriff auf den DDR-Speicher erteilt, den anderen werden Haltesignale ausgegeben. Für den ausgewählten Speicherzugriff wird ein DDR-Speicherbefehl generiert und die zugehörige Adresse mit einem Multiplexer mit dem Steuersignal M0 bis M1 ausgewählt, die Ausgabe der Schreibdaten erfolgt bei einem Schreibbefehl direkt.

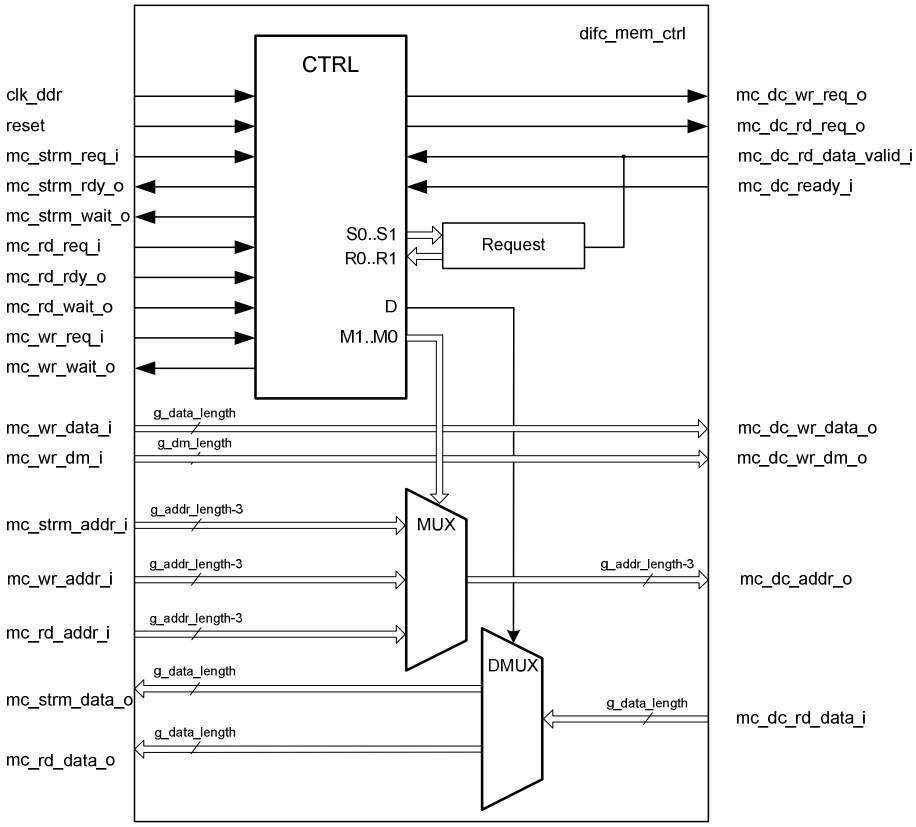

Abbildung 58: Arbiter für den Zugriff auf den DDR-Controller

In Abbildung 59 ist ein Beispiel dargestellt wie gleichzeitig gewünschte Speicherzugriffe von der Komponente Arbiter aufgelöst werden. Alle drei Einheiten Bildschirm, Lesen und Schreiben signalisieren zu Beginn einen Speicherzugriff. Gemäß den oben festgelegten Proritäten setzt sich der Lesezugriff des Moduls Bildschirm (*mc_strm_req_i*) durch und am Ausgang signalisiert *mc_dc_rd_req_o* dem DDR_Controller einen Lesebefehl. Den beiden anderen Einheiten wird zur fallenden Taktflanke ein Haltesignal gesetzt. Nachdem die Bildschirmeinheit ihren Befehl absetzen konnte (*mc_strm_wait_o = ‚0'*) wird der Lesebefehl zurückgezogen. Als nächstes setzt sich die Einheit Schreiben (*mc_wr_req_i*) aufgrund ihrer höheren Priorität gegenüber der Leseeinheit (*mc_rd_req_i*) durch und der Arbiter stellt einen Schreibbefehl an den Speicher. Erst wenn alle Zugriffswünsche der anderen Einheiten erfüllt sind, kann die Einheit Lesen ihren Befehl absetzen.

Wurde ein Lesebefehl erteilt, wird in einem zweistelligen Schieberegister REQUEST über S0 und S1 die Quelle des Lesebefehls eingetragen. Bei *S0 = ‚1'* wird der Lesebefehl der Komponente Bildschirm, bei *S1 = ‚1'* der Einheit Lesen zugeordnet. Werden nun die Lesedaten als Konsequenz der Lesebefehle

erhalten, so werden aufgrund des ältesten Eintrags im Schieberegister die Daten mit einem Demultiplexer über das Steuersignal D korrekt zugeordnet.

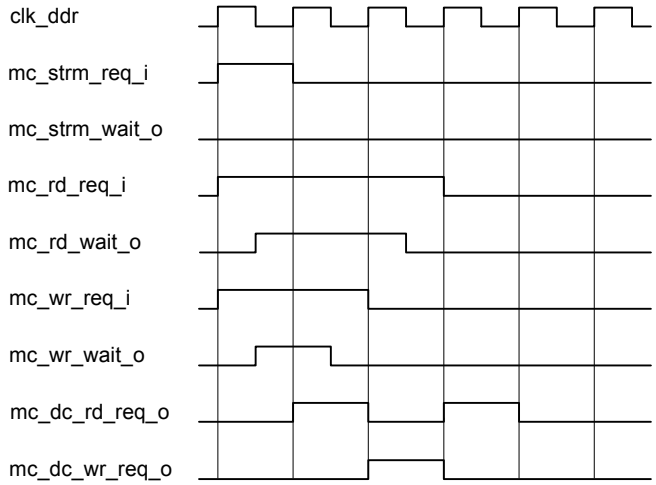

Abbildung 59: Priorisierung der Speicherzugriffe

Tabelle 19: Ein- und Ausgänge der Einheit Arbiter

Port	**Bedeutung**
clk_ddr	133-MHz-Taktsignal des DDR-Controllers
reset	Reset-Signal
mc_strm_req_i	Lesebefehl von der Komponente Bildschirm
mc_strm_rdy_o	Lesedaten an mc_strm_data_o für die Komponente Bildschirm gültig
mc_strm_wait_o	Komponente Bildschirm anhalten
mc_rd_req_i	Lesebefehl von der Komponente Lesen
mc_rd_rdy_o	Lesedaten an mc_rd_data_o für die Komponente Lesen gültig
mc_rd_wait_o	Komponente Lesen anhalten
mc_wr_req_i	Schreibbefehl von der Komponente Schreiben
mc_wr_dm_i	Datamask-Bits für den Schreibbefehl der Komponente Schreiben
mc_strm_addr_i	Adresse für den Lesebefehl der Einheit Bildschirmaufbau
mc_wr_addr_i	Adresse für den Schreibbefehl der Einheit Schreiben
mc_rd_addr_i	Adresse für den Lesebefehl der Einheit Lesen
mc_strm_data_o	Gelesene Daten für die Einheit Bildschirm
mc_rd_data_o	Gelesene Daten für die Einheit Lesen
mc_dc_wr_req_o	Schreibbefehl an den DDR-Controller
mc_dc_rd_req_o	Lesebefehl an den DDR-Controller
mc_rd_data_valid_i	Lesedaten an mc_dc_rd_data_i vom DDR-Controller gültig
mc_dc_rdy_i	DDR-Controller ist bereit, Befehle anzunehmen
mc_dc_wr_data_o	In den Speicher zu schreibende Daten
mc_dc_wr_dm_o	Datamask-Bits für die zu schreibenden Daten
mc_dc_addr_o	DDR-Adresse
mc_dc_rd_data_i	aus dem DDR gelesene Daten

Die Steuerlogik besteht aus den drei folgenden Steuerprozessen:

Prozess ARBIT: Vergibt den Speicherzugriff nach den Prioritäten und erzeugt die Speicherbefehle für den DDR-Controller.

Prozess RDDATA_CTRL: Leitet die eintreffenden Lesedaten an die aufrufende Einheit weiter.

Prozess WAIT_CTRL: Erzeugt die Haltesignale für die Einheiten des Video-Controllers.

6.5.2 Vergabe des Speicherzugriffs

Der Prozess ARBIT der Steuerlogik (Abbildung 60) entscheidet über die Zuteilung der eingehenden Befehle zum DDR-Controller, indem die Requests unter Berücksichtigung der Wait-Signale ausgewertet werden.

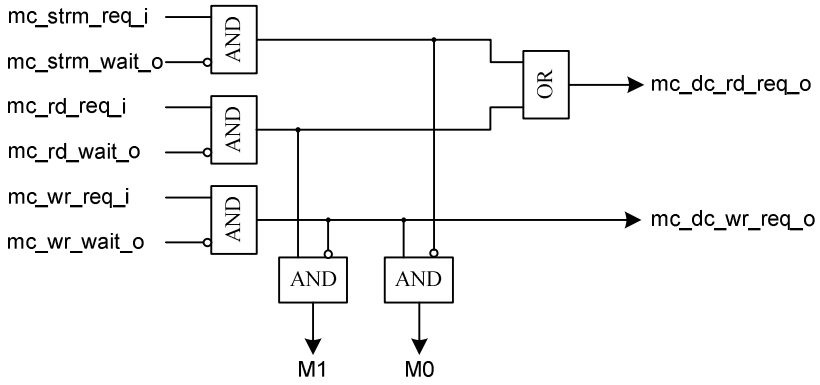

Abbildung 60: Prozess ARBIT

Es wird aus den gewünschten Speicherzugriffen, anhand der oben verteilten Prioritäten, der entsprechende Schreib- oder Lesebefehl an den DDR-Speicher signalisiert. Zusätzlich werden die Signale M1 und M0 generiert, mit Hilfe derer der Adressmultiplexer die Adressen für den Ausgang *mc_dc_addr_o* auswählt. Welche Adressen ausgewählt werden, ist in Tabelle 20 ersichtlich.

Tabelle 20: Auswahl des Adressmultiplexers

M1	M0	ausgewählte Adresse
0	0	mc_strm_addr_i
0	1	mc_wr_addr_i
1	0	mc_rd_addr_i
1	1	-

6.5.3 Zuweisung der eintreffenden Lesedaten

Wurde ein Lesebefehl abgesetzt, so erkennt der Steuerprozess READDATA_CTRL mit *mc_rd_data_valid_i = ,1'*, dass die Daten vom DDR-Controller am Dateneingang *mc_dc_rd_data_i* gültig sind und übernommen werden können. Der Registerstand des Schieberegisters REQUEST bestimmt, welcher Einheit die Daten zuzuordnen sind. Bei *R0 = ,1'* werden mit dem Demultiplexer die

Daten an den Ausgang *mc_strm_data_o*, bei *R1 = ,1'* an den Ausgang *mc_rd_data_o* geschaltet. Mit *mc_rd_data_valid_i = ,1'* wird auch das Schieberegister um eine Stelle weitergeschoben.

6.5.4 Setzen der Haltesignale

Vom Prozess WAIT_CTRL (Abbildung 61) werden die Haltesignale an die drei Einheiten Bildschirm, Lesen und Schreiben des Video-Controllers generiert. Einerseits kann der DDR-Controller nicht zur Übernahme von Befehlen bereit sein, andererseits können die Zugriffe der Einheiten gleichzeitig erfolgen. Dies wird zur fallenden Taktflanke überprüft und entsprechend den Prioritäten die Haltesignale für die Einheiten des Video-Controllers gesetzt.

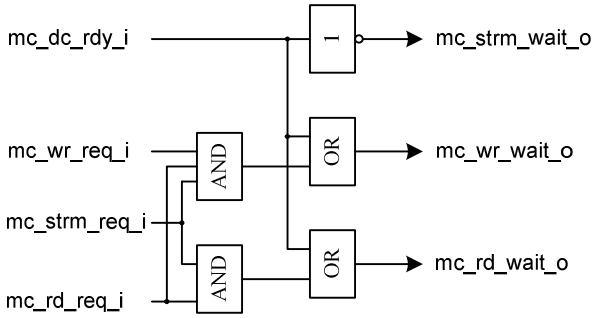

Abbildung 61: Erzeugung der Haltesignale

6.6 Komponente Datenausgabe

Der ursprüngliche Video-Controller verwendet zur Ausgabe Lesedaten ein gemeinsames Ausgabeport. Dies soll auch beim modifizierten Video-Controller beibehalten werden. Da es zu gleichzeitig auftretenden Datenausgaben der Komponenten Bildschirm und Lesen kommen kann, benötigt man die Komponente Datenausgabe. Im Fall einer Kollision wird der Einheit Bildschirm aufgrund der zeitkritischen Bilddaten der Vorrang eingeräumt und die Einheit Lesen angehalten.

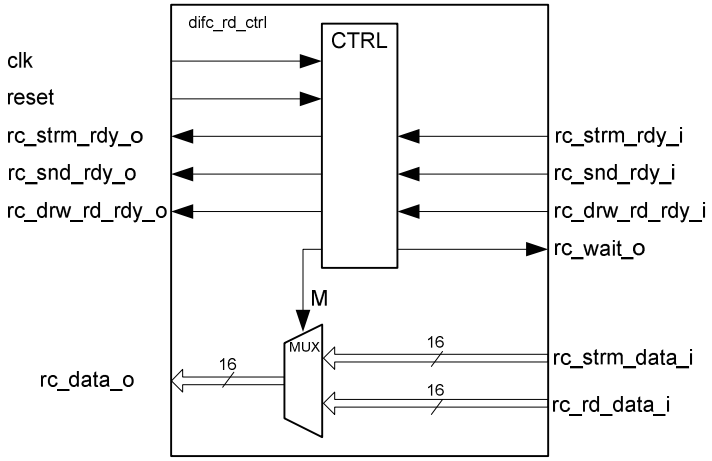

Abbildung 62: Ausgabe der Lesedaten

Die Struktur dieser Komponente ist aus Abbildung 62 ersichtlich. Die Steuerlogik wertet die vom DDR-Controller kommenden, im Bild an der rechten Seite befindlichen, Datenausgaben aus und leitet die Daten mit einem Multiplexer an den Datenausgang. Gleichzeitig wird dem Empfänger die Gültigkeit der Lesedaten signalisiert. Bei gleichzeitig gewünschter Datenausgabe wird mit *rc_wait_o = ,1'* die Einheit Lesen angehalten. Der Grund, warum hier drei Ready-Signale, aber nur zwei Dateneingänge vorhanden sind, liegt darin, zwischen den Empfängern Einheit Zeichenoperationen und Signaltöne unterscheiden zu müssen.

In Tabelle 21 sind die Ports der Komponente Datenausgabe und ihre Verwendung angeführt.

Tabelle 21: Ports der Komponente Datenausgabe

Port	**Bedeutung**
clk	81-MHz-Taktsignal
reset	Reset-Signal
rc_strm_rdy_o	Ready-Signal für die Komponente Bildschirmaufbau
rc_snd_rdy_o	Ready-Signal für die Komponente Signaltöne
rc_drw_rd_rdy_o	Ready-Signal für die Komponente Zeichenoperationen
rc_data_o	Datenausgabe
rc_strm_rdy_i	Daten von Komponente Bildschirm gültig
rc_snd_rdy_i	Sound-Daten von Komponente Lesen gültig
rc_drw_rd_rdy_o	Daten für Zeichenoperationen von Komponente Lesen gültig
rc_wait_o	Wartesignal für Komponente Lesen
rc_strm_data_i	Daten von der Einheit Bildschirmaufbau
rc_rd_data_i	Daten von Komponente Lesen

Die Steuerlogik wird vom Prozess ARBIT, in Abbildung 63 dargestellt, realisiert. Er steuert die Datenausgabe mit dem Multiplexer über das Signal M. Kommt es zu einer Kollision bei der Datenausgabe von Bildschirmdaten *rc_strm_rdy_i = ,1'* und Daten der Komponente Lesen (*rc_snd_rdy_i* oder *rc_drw_rdy_i*) wird die Einheit Lesen mit *rc_wait_o = ,1'* zur fallenden Taktflanke angehalten. Die Ready-Signale werden wie in Abbildung 63 erzeugt.

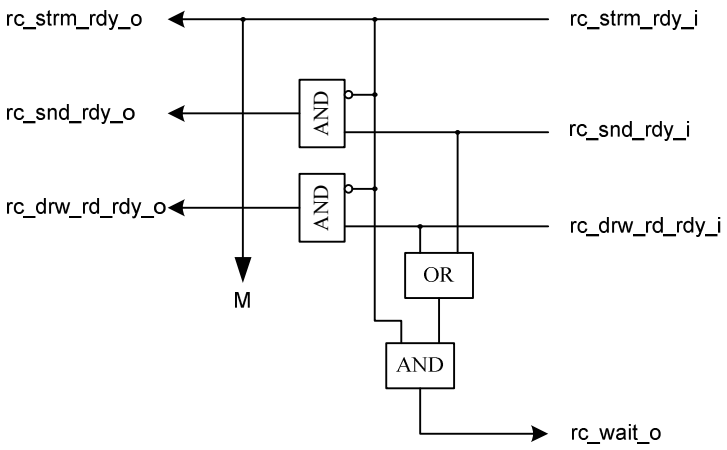

Abbildung 63: Steuerprozess ARBIT

7. Verifikation

Nach der Realisierung des Design-Entwurfs bzw. auch während der Realisierung, wenn die Entwicklung der einzelnen Einheiten abgeschlossen ist, folgte die Verifizierung. Die Aufgabe der Verifikation ist ganz allgemein die Überprüfung der Korrektheit eines Schaltungsentwurfs in Bezug auf die funktionalen und zeitlichen Anforderungen. In diesem Kapitel wird auf die Methodik der Verifikation im Allgemeinen, sowie die konkrete Durchführung der Verifikation im Video-Controller-Design eingegangen.

Um die funktionalen Anforderungen zu überprüfen, wird der Entwurf z. B. simuliert oder in einem FPGA-Prototypen realisiert. Das korrekte Zeitverhalten kann mit der Methode der statischen Timing-Analyse überprüft werden [SIE03].

7.1 Einführung

Um einen Systemsausfall zu verhindern, muss man zeigen, dass das System keine Fehler enthält, fehlertolerant agiert oder zumindest einen sicheren Zustand einnimmt (fail safe). Man unterscheidet grundsätzlich drei Arten von Hardwarefehlern [KRO99]:

- Design-Fehler
- Fabrikationsfehler
- Fehler durch Gebrauch

Fehler durch Gebrauch äußern sich erst nach einer gewissen Verwendungsdauer, sie entstehen durch Abnützung, Alterung, unzureichende Wartung, Fehlbedienung usw. Fabrikationsfehler entstehen im Fabrikationsprozess und sind durch Testen feststellbar. Dieses Kapitel beschäftigt sich mit Design-Fehlern und deren Erkennung, sie basieren auf der fehlerhaften Transformation der Spezifikation in die Schaltkreisbeschreibung [KRO99].

Im Hardware-Design ist es besonders wichtig, Schwachstellen aufzufinden, da diese nachträglich schwieriger zu korrigieren sind als Software-Fehler. Daher gilt im Hardwaredesign [KRO99]:

- Fabrikationskosten sind bei Hardware wesentlich höher als bei Software
- Nach der Auslieferung sind keine Updates mit Bug-Fixes möglich
- Qualitätsansprüche sind bei Hardware höher, da keine Updates möglich sind

- Die Verifikation hat im Hardware-Design einen sehr hohen Anteil an der Entwicklungszeit des Produktes. Entsprechend bestimmt auch die Dauer der Verifikation die Time-to-Market-Zeit und nimmt Einfluss auf den Ertrag des Produktes. Trotzdem muss die Verifikation gut genug für eine fehlerfreie Hardware sein.

Für die Verifikation sind prinzipiell mehrere Ansätze möglich.

Funktionale Verifikation

Bei der funktionalen Verifikation werden spezielle Schaltungszustände überprüft. Der Schaltungsentwickler legt dabei fest, welche Signale am Eingang der Schaltung zu jenen Signalen am Ausgang der Schaltung führen, die das Verhalten der Schaltung zeigen. Dabei bestehen zwei Möglichkeiten:

- Das gesamte System wird simuliert und das Verhalten in der Simulation betrachtet und mit dem erwarteten Verhalten verglichen. Die Simulation ist sehr zeitaufwendig und dadurch sind keine Langzeittests möglich. Es werden daher kürzere Tests mit Spezialfällen durchgeführt.
- Der Entwurf kann auch in einem FPGA-Prototypen realisiert werden, wodurch ein Test in realer Hardware durchgeführt wird. Hier sind Langzeittests möglich. Das Testen erfolgt in Echtzeit, allerdings kann nicht sichergestellt werden, dass alle Spezialfälle abgedeckt werden.

Bei der funktionalen Verifikation wird eine sogenannte Testbench verwendet, deren Konzept in Abbildung 64 dargestellt ist.

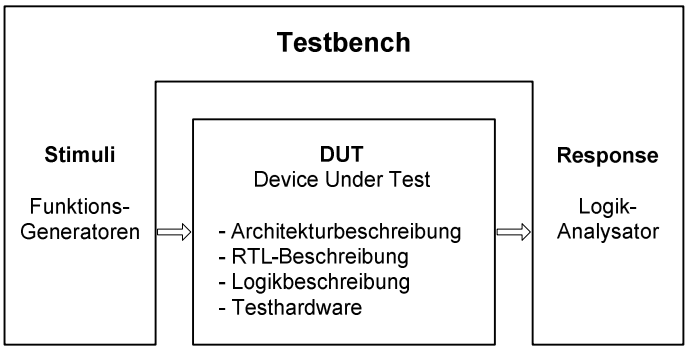

Abbildung 64: Testbench [SIE03]

Das zu prüfende Objekt wird als DUT, als Device Under Test, bezeichnet. Durch die Testbench werden repräsentative Eingangssignalmuster, die Stimuli, an die Eingänge der Schaltung gelegt und die Reaktion (Response) an den Ausgängen beobachtet. Zeigt die Response nicht die erwarteten Werte, konnte ein Fehlverhalten nachgewiesen werden. Da es beim Testen meist unmöglich ist, alle möglichen Stimuli zu untersuchen, kann das korrekte Verhalten der Schaltung nur für die Stimuli bewiesen werden.

Statische Timinganalyse

Die Analyse für die zeitlichen Anforderungen an das Design erfolgt mit Hilfe der statischen Timinganalyse, durch die das Zeitverhalten im FPGA bezüglich Laufzeiten und Verzögerungen in den Logikele-

menten berechnet wird. Daraus wird die maximal mögliche Taktfrequenz, bei der die Schaltung betrieben werden kann, ermittelt.

Formale Verifikation

Durch formale Verifikation wird die vollständige Korrektheit einer Schaltung bewiesen, d. h. es wird der Beweis erbracht, dass sich die Schaltung entsprechend ihrer Spezifikation verhält. Ausgangspunkt einer formalen Verifikation ist eine mathematische Beschreibung des Systems und der nachzuweisenden Eigenschaften.

Eine Möglichkeit der formalen Verifikation ist z. B. die Modellprüfung mit temporalen Logiken (model checking). Hier wird aus der Implementierungsbeschreibung ein Modell für die Logik extrahiert und anschließend geprüft, ob dieses Modell für Spezifikation gültig ist [FRO].

7.2 Verifikationsebenen

Die Verifikation eines Designs kann auf verschiedenen Ebenen durchgeführt werden [**BER06**].

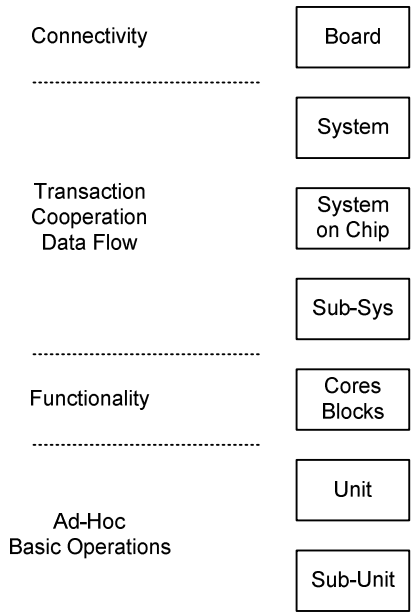

Abbildung 65: Ebenen der Verifikation [BER06]

In Abbildung 65 sind die verschiedenen Ebenen der Verifikation dargestellt. Auf der untersten Ebene sind die Einheiten (Units), die einzelnen Module wie FIFOs oder Zustandsautomaten. Da es für sie keine eigene unabhängige Spezifikation gibt, überprüft der Designer in der RTL-Beschreibung (Register Transfer Logik) den Code auf Syntaxfehler und Funktionalität.

Die einzelnen Module realisieren die Funktionsblöcke, sie sind die kleinsten unabhängig zu verifizierenden Funktionseinheiten. Dazu zählen auch wiederverwendbare Einheiten mit eigener Spezifikation (Reusable Cores), die in verschiedenen Designs unverändert verwendet werden können. Ziel der Verifi-

kation auf Blockebene ist, die Funktionalität der Blöcke zu gewährleisten. Verifikation auf Systemebene bedeutet die Verifikation des Zusammenschlusses der verschieden unabhängigen Blöcke. Im Vordergrund stehen hier die Kommunikation und das Zusammenspiel der Einzelteile des Systems. Bei der Verifikation auf der Ebene des Boards verwendet man ein Board Design Tool, das zusätzlich zum Design das Verhalten des Boards wie z. B. Signallaufzeiten einbringt [BER06].

Die modifizierte Speicherschnittstelle wird aus einzelnen Modulen zusammengesetzt, daher bietet sich für die Verifikation die Simulation auf Blockebene an. Nach Fertigstellung der Module wurden diese in eigens dafür erstellten Testbenches überprüft. Diese erzeugten typische Eingangssignalmuster für die Module, die Reaktionen der Ausgangssignale wurden mit der Simulationsumgebung Modelsim® von Mentorgraphics graphisch dargestellt und auf ihre Richtigkeit überprüft. Mit der Simulation der einzelnen Module konnte nachgewiesen werden, dass sich diese entsprechend ihrer Spezifikation verhalten und die erforderliche Funktion erbringen.

Als nächstes wurde auf Systemebene verifiziert. Dazu wurde die Testbench des bestehenden Video-Controllers verwendet, zusätzlich wurden DDR-Controller und externe Speicher als Simulation eingefügt. Der komplette Simulationsaufbau ist im folgenden Kapitel beschrieben.

7.3 Simulationsaufbau

Um die Funktionsweise des Video-Controllers zu verifizieren, muss das gesamte System in der Simulation verwendet werden. Es müssen einerseits die im FPGA eingesetzten Einheiten simuliert und andererseits alle relevanten externen Einheiten in Simulationsmodellen nachgebildet werden. Die benötigten Einheiten sind in Abbildung 66 dargestellt.

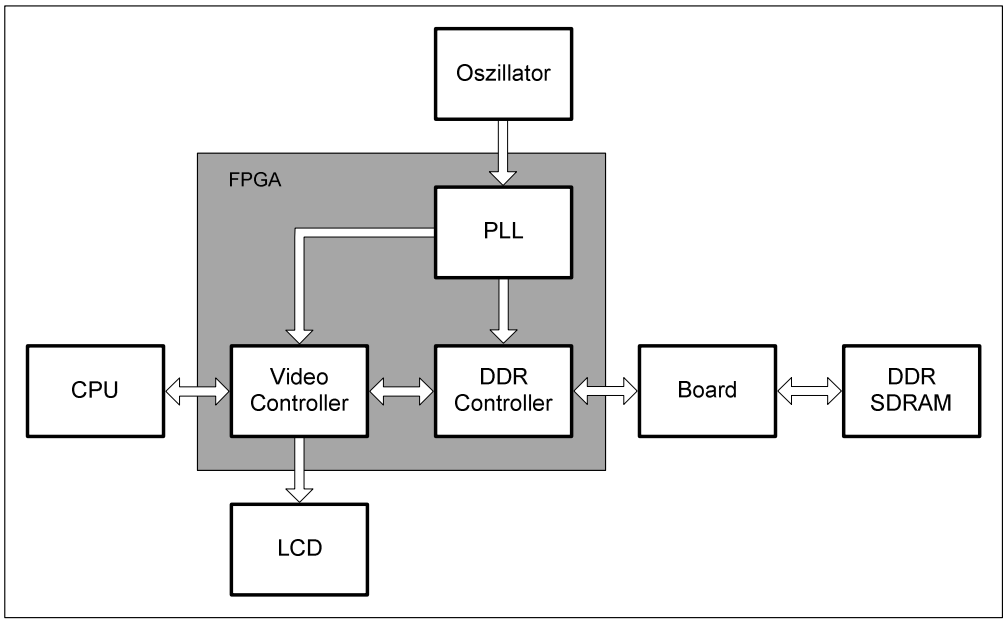

Abbildung 66: Aufbau der Simulation

Dazu zählen der Video-Controller, der DDR-Controller und ein PLL-Baustein (Phase Locked Loop), der die für Video-Controller und DDR-Controller notwendigen Taktsignale und verschiedenen Phasenlage erzeugt. Der DDR-Controller wird mit einer Taktfrequenz von 133 MHz, der Video-Controller mit einem 81-MHz-Taktsignal versorgt. Als Taktgeber wird ein simulierter Oszillator verwendet. Die Befehle an den Speicher werden vom DDR-Controller übernommen.

Für die Central Processing Unit wird ein Simulationsmodell einer ELAN-CPU SC520 (Abbildung 67) eingesetzt. Das Modell liest die auszuführende Befehlssequenz aus einer Textdatei und legt diese über Eingangssignale an den Video-Controller an.

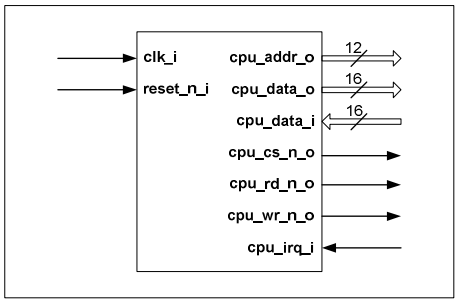

Abbildung 67: Ports der modellierten ELAN CPU

Die CPU (Abbildung 67) verfügt über einen 16-Bit-Datenbus, mittels Chipselect-, Read- bzw. Write-Signal werden Befehle an den Video-Controller abgesetzt. Zusätzlich ist ein Interrupt-Eingang vorhanden.

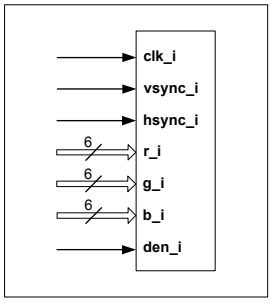

Abbildung 68: Entity der LCD-Simulation

Die Bildpunkte werden vom Video-Controller an eine Simulation eines LCDs (Liquid Crystal Display) ausgegeben (Abbildung 68). Diese Einheit wertet fortlaufend die Synchronisationssignale VSYNC und HSYNC, sowie die Bilddaten in 3 x 6-Bit-RGB-Darstellung aus und gibt die daraus resultierenden Bildschirmseiten in Bilddateien aus. Als Format wird hier Windows Bitmap (BMP) mit einer Auflösung entsprechend der Video-Controller-Auflösung von 800 x 600 Pixel verwendet.

Da der DDR-Speicher nicht im FPGA integriert ist, sondern extern an einem Board angebunden ist, werden die Signallaufzeiten zu und vom Speicher in der Simulation entsprechend verzögert (Abbildung 69).

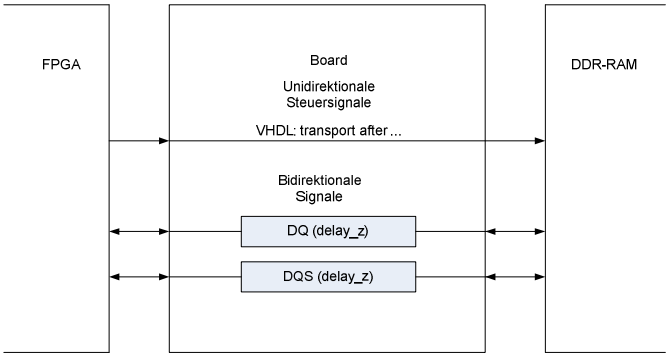

Abbildung 69: Simulationsmodell für die Laufzeitverzögerungen auf dem Board [WIE07]

Die Werte der Laufzeitverzögerungen wurden gemäß den Erfahrungswerten früherer Projekte auf dem WPLD-Board (WASP Programmable Logic Device) übernommen und nach Tabelle 22 modelliert.

Tabelle 22: Modellierte Signallaufzeiten [WIE07]

Signal am Board	Laufzeit
RAM_CLK	1,380 ns
CKE	2,760 ns
ADDR	2,760 ns
BA	2,760 ns
CSn	2,760 ns
RASn	2,760 ns
CASn	2,760 ns
WEn	2,760 ns
DM	2,760 ns
DQ	1,510 ns
DQS	1,510 ns

Als Speichermodell wurde das von der Firma Micron zum Download angebotene, freie VHDL-Modell eines MT46v32m8-Speicherchips übernommen [3]. Es handelt sich hierbei um einen 256-MBit-Speicherchip mit 8-Bit-Datenwortbreite. Da vom Hersteller keine RAM-Module sondern lediglich Modelle der Speicherchips angeboten werden, mussten mehrere Instanzen des Speicherchips entsprechend zusammengesetzt werden.

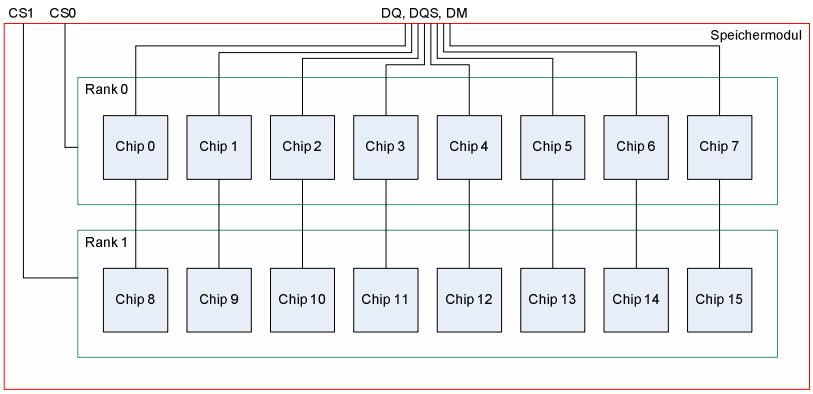

Abbildung 70: Aufbau des verwendeten Speichermodells [WIE07]

Das zusammengesetzte Speichermodul mit einer Datenwortbreite von 64 Bit und 512 MB ist in Abbildung 70 dargestellt Um der Realität zu entsprechen, ist das Modell parametrisierbar. Jeder Chip verfügt über 13 Adressleitungen und 2 Bankleitungen. Tabelle 23 zeigt die aktuelle Konfiguration der Parameter zur Simulation des Speichermodells.

Tabelle 23: Generische Variablen des Speichermodells

Generische Variable	Beschreibung	Typ	Standardwert
tCK	Clock	time	7.5 ns
tCH	Clock High	time	3.375 ns
tCL	Clock Low	time	3.375 ns
tDH	Input Hold Time für die DQ- und DQS-Leitungen	time	0.500 ns
tDS	Input Setup Time für die DQ- und DQS-Leitungen	time	0.500 ns
tIH	Input Hold Time für die Adress- und Steuerleitungen	time	0.900 ns
tIS	Input Setup Time für Adress- und Steuerleitungen	time	0.900 ns
tMRD	Mode Register Delay	time	15 ns
tRAS	Row Access Strobe	time	40 ns
tRAP	Row Auto Precharge	time	20 ns
tRC	Row Cycle Time	time	65 ns
tRFC	Refresh Cycle Time	time	75 ns
tRCD	Row Column Delay	time	20 ns
tRP	Row Precharge Time	time	20 ns
tRRD	Row Row Delay	time	15 ns
tWR	Write Recovery Time	time	15 ns
addr_bits	Adressbits	integer	13
data_bits	Datenbits	integer	8
cols_bits	Spaltenbits	integer	10

7.4 Testfälle

Nach dem Aufbau der Simulation wurde mit repräsentativen Testfällen die korrekte Arbeitsweise des Video-Controllers überprüft und mit den Erwartungen verglichen. Die Überprüfung der Funktion des Video-Controllers erfolgte zum einen graphisch in Modelsim, zum anderen wurden die vom Simulationsmodell des LCDs erzeugten Bilddateien analysiert. Weiters wurden im Video-Controller an verschiedenen Stellen Monitore eingesetzt, die den Datenverkehr im Video-Controller protokollieren und in Textdateien ausgeben und auch die Kommunikation fortlaufend auf ihre Korrektheit überprüfen.

7.4.1 Testfall Verifikation des Bildaufbaues

Um den Aufbau einer Bildschirmseite zu verifizieren, wurde über die simulierte ELAN-CPU eine komplette Bildschirmseite mit aufsteigend durchnummerierten Datenwerten in den Bildspeicher geladen und danach die Ausgabe der Bilddaten an der Ausgangsseite der modifizierten Speicherschnittstelle zur Bildschirmaufbaueinheit betrachtet. Hierfür wurde ein Monitor (*check_strm*), wie in Abbildung 71 dargestellt, eingesetzt.

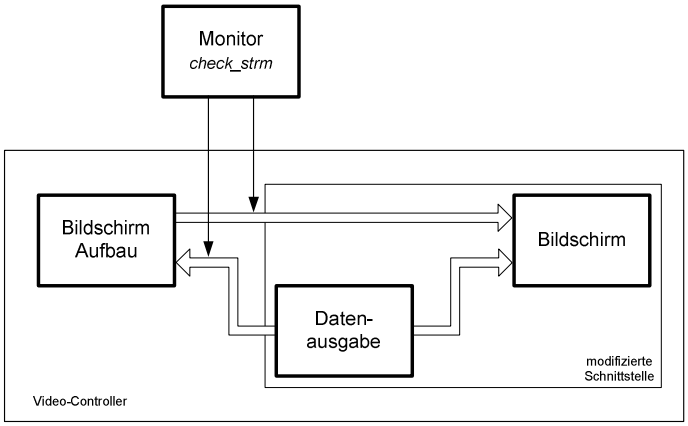

Abbildung 71: Monitor check_strm

Dieser beobachtet die Speicheranforderungen des Moduls für den Bildschirmaufbau des Video-Controllers und die aus dem DDR-SDRAM gelieferten Daten. Der Monitor meldet wenn:

- die angeforderten Daten nicht oder falsch übermittelt wurden
- Daten ohne Anforderung erhalten wurden
- Daten zum falschen Zeitpunkt eintreffen
- trotz Lesebefehle keine Daten eintreffen
- alle Daten der Bildschirmseite korrekt eingetroffen sind.

Die Meldungen werden in eine Textdatei geschrieben, sodass nach Ablauf der Simulation einer Bildschirmseite überprüft werden kann, ob der Datenverkehr korrekt stattgefunden hat.

Mit diesem Testfall konnte gezeigt werden, dass die angeforderten Bilddaten in der korrekten Reihenfolge und vollständig an die Einheit Bildschirmaufbau ausgegeben wurden.

7.4.2 Testfall Verifikation der Schreib- und Lesezugriffe

Um die korrekte Funktionsweise des Video-Controllers nachzuweisen, wurden verschiedene Befehlsfolgen über die CPU an den Video-Controller abgesetzt und es wurde überprüft, ob der Inhalt der Bilddatei mit den erwarteten Ergebnissen übereinstimmt. Als Beispiel soll hier eine anschauliche Folge von Zeichen- und Kopierbefehlen gezeigt werden.

Als erstes wird ein Rechteck 16 x 16 Pixel in der Farbe Rot beginnend an der Position (16,0) gezeichnet. Bei den Koordinaten bedeutet die erste Zahl die horizontale Koordinate und die zweite Zahl die vertikale Koordinate, wobei sich der Ursprung des Koordinatensystems in der linken oberen Ecke befindet und von links nach rechts und oben nach unten gezählt wird. Es gilt zu beachten, dass die Startposition nicht in Koordinaten, sondern als Nummer des Bildpunktes bei linearer Adressierung angegeben wird, also ein Wert von 0 bis 479 999. Die Farbdarstellung erfolgt in 8-Bit-Farbpalette, d. h. es kann aus 256 Farben ausgewählt werden.

In Abbildung 72 ist auf der linken Seite die Befehlsfolge der Zeichenoperation in der Eingabetextdatei für die Simulation des ELAN-Prozessors angegeben, auf der rechten Seite jeweils der Bildausschnitt mit dem Ergebnis der Zeichenoperationen.

```
Write ROITARGET_L    0x0010
Write ROITARGET_H    0x0000
Write ROIWIDTH       0x0010
Write ROIHEIGHT      0x0010
Write ROICOLOR       0x00F3
Write VGA_CONTROL    0x0020
```

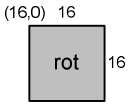

Abbildung 72: Rechteck zeichnen

Danach folgt ein Kopierbefehl *„Kopieren und mit definierter Farbe einfärben“,* der das rote Quadrat kopiert und auf Position (32,0) im Bild einfügt und mit der Farbe Grün (08) einfärbt (Abbildung 73).

```
Write ROISTART_L     0x0010
Write ROISTART_H     0x0000
Write ROITARGET_L    0x0020
Write ROITARGET_H    0x0000
Write ROIWIDTH       0x0010
Write ROIHEIGHT      0x0010
Write ROICOLOR       0xF308
Write VGA_CONTROL    0x0400
```

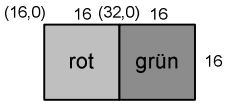

Abbildung 73: Kopieren und mit definierter Farbe einfärben

Der nächste Befehl *„Kopieren mit Transparenz“* kopiert aus dem Rechteck mit rotem und grünem Quadrat nur jene Bildpunkte, deren Farbe nicht Rot (F3) ist und fügt die Pixel ab der Position (0,16) ein (Abbildung 74).

```
Write ROISTART_L     0x0010
Write ROISTART_H     0x0000
Write ROITARGET_L    0x3200
Write ROITARGET_H    0x0000
Write ROIWIDTH       0x0020
Write ROIHEIGHT      0x0010
Write ROICOLOR       0xF300
Write VGA_CONTROL    0x0080
```

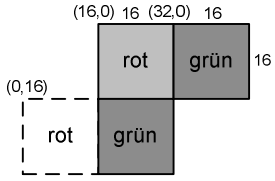

Abbildung 74: Kopieren mit Transparenz

Nun folgt *„Kopieren und Einfärben von rechteckigen Bildausschnitten mit definierter Farbe“*. Bei diesem Befehl wird der quadratische 32 x 32-Bildausschnitt mit den bisher drei gezeichneten Quadraten kopiert und an Position (32,16) eingefügt. Dabei werden nur die Bildpunkte mit der Farbe Rot kopiert und eingefügt (Abbildung 75).

```
Write ROISTART_L    0x0010
Write ROISTART_H    0x0000
Write ROITARGET_L   0x3220
Write ROITARGET_H   0x0000
Write ROIWIDTH      0x0020
Write ROIHEIGHT     0x0020
Write ROICOLOR      0x08F3
Write VGA_CONTROL   0x0008
```

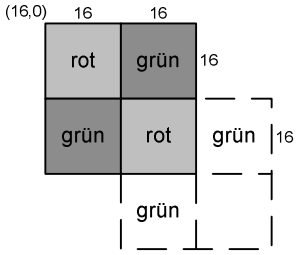

Abbildung 75: Kopieren und Einfärben mit definierter Farbe

Letzter Befehl dieser Sequenz ist *„Kopieren von rechteckigen Bildausschnitten“*, es werden die vier bisher gezeichneten Quadrate kopiert und an der Position (16,32) eingefügt (Abbildung 76).

```
Write ROISTART_L    0x0010
Write ROISTART_H    0x0000
Write ROITARGET_L   0x6410
Write ROITARGET_H   0x0000
Write ROIWIDTH      0x0020
Write ROIHEIGHT     0x0020
Write VGA_CONTROL   0x0004
```

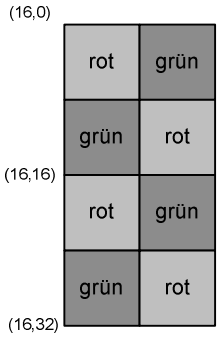

Abbildung 76: Kopieren von rechteckigen Bildausschnitten

Die Befehlsfolge wurde in der Testbench simuliert und nach Ausgabe von 800 x 600 Pixel an die Simulationseinheit des LCDs, eine Bitmapdatei einer Bildschirmseite, ausgegeben.

Um das Ergebnis zu überprüfen, muss für das erzeugte Bild untersucht werden, ob die Pixel entsprechend den Befehlen an den richtigen Stellen und vollständig eingefügt wurden. Die manuelle Prüfung ist sehr aufwendig und unübersichtlich, denn man muss die erzeugten Bilder Pixel für Pixel überprüfen.

Um diese Aufgabe zu automatisieren und damit zu vereinfachen, wurde eine Verifikationssoftware in der Programmiersprache C++ geschrieben, die die durchgeführten Zeichenoperationen des Video-Controllers in Software berechnet. Die damit erstellten Bilddateien können als Referenzdaten herangezogen werden und die Verifikation des Bildes kann durch einen einfachen Dateivergleich erfolgen.

7.5 Verifikationssoftware

Die Software (Abbildung 77) verwaltet ein Abbild des Bildspeichers in der Form eines Arrays in der die Farbindizes jedes Bildpunktes gespeichert werden. Im Palettenspeicher erfolgt die Zuordnung der Farben zu den Indizes. Von der Software werden die Zeichenbefehle, die von der CPU an den Video-Controller abgesetzt werden und in einer Befehlstextdatei enthalten sind, ausgegeben. Die Zeichenoperationen werden durch die einzelnen Funktionen des Programms ausgeführt und die entsprechenden

Bildpunkte im Bildspeicher-Array manipuliert. Die Inhalte des Bildspeichers können jederzeit in Bitmap-Dateien ausgegeben werden.

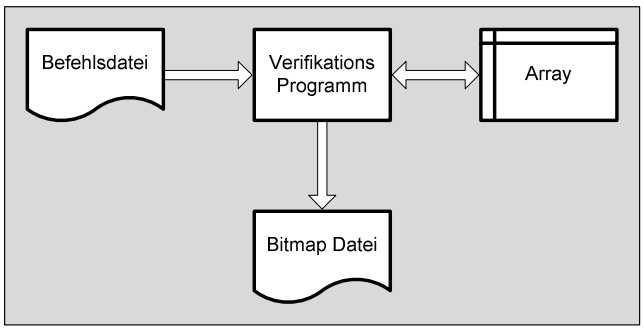

Abbildung 77: Verifikationssoftware

Automatisierter Dateivergleich

Die Berechnung der beiden Bitmap-Dateien erfolgt auf unterschiedliche Arten (Abbildung 78). In der Simulation auf Taktebene und mit Befehlen an den Speicher, in der Software hingegen auf funktionaler Ebene und mit direkt indizierten Zugriffen auf das Bildspeicher-Array. Da es unwahrscheinlich ist, dass bei unterschiedlichen Arten der Bilderstellung der gleiche Fehler gemacht wird, kann man bei einer Übereinstimmung davon ausgehen, dass der Video-Controller korrekt seine Funktionen erfüllt.

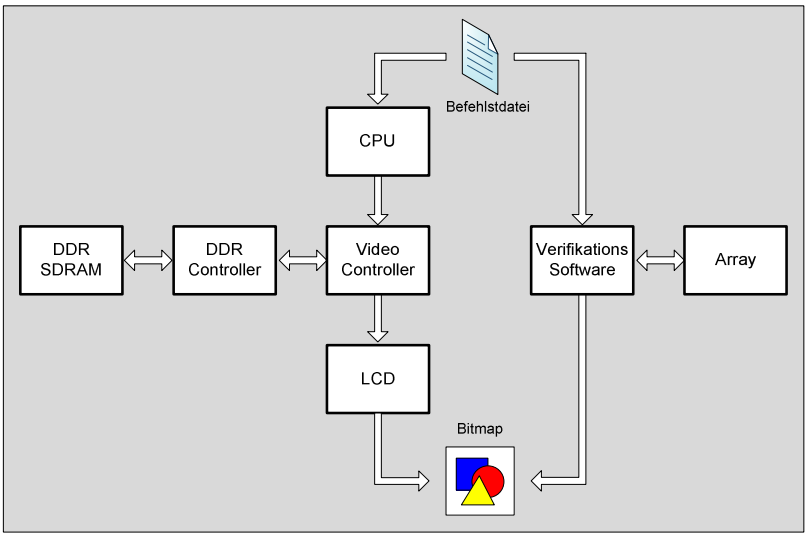

Abbildung 78: Automatisierter Dateivergleich

Die erzeugten Bitmap-Dateien können im Weiteren als Referenzobjekt dienen. Solche Referenzdateien werden als „Golden Files“ [BER06] bezeichnet. Immer dann, wenn die Funktionalität des Designs erweitert wird oder Änderungen am Video-Controller durchgeführt werden, kann man die „Golden Files“ heranziehen, um zu überprüfen, ob der Video-Controller auch nach den Änderungen weiterhin korrekt arbeitet, d. h. sich die erzeugten Bilder nicht verändert haben.

Erstellung einer Bitmap-Datei

Für die Generierung der Bitmap-Dateien in der Software musste das Bitmap-Format analysiert werden.

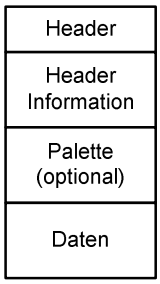

Abbildung 79: Aufbau einer Bitmap-Datei

Eine Bitmap-Datei besteht grundsätzlich aus vier Datenblöcken (Abbildung 79). Der Header und die Header-Information enthalten die für die Darstellung notwendigen Angaben wie z. B. Bilddimensionen oder Farbtiefe. Optional folgt die Palette zur Farbdarstellung und erst anschließend die eigentlichen Daten der Bildpunkte. Bei Palettenfarbdarstellung sind die Bilddaten als Palettenindizes gespeichert, während bei echter Farbdarstellung die Bilddaten direkt als Farbwerte gespeichert werden. Die Bildpunkte werden in der Reihenfolge von unten nach oben und links nach rechts in der Bitmap-Datei gespeichert.

Wie bereits erwähnt, erstellt die Software für die Abbilder des Bildspeichers Bitmap-Dateien, wobei diese gemäß Tabelle 24 erzeugt werden.

Tabelle 24: Einträge im Header und Header-Information der vom LCD erzeugten Bitmaps

Byteadresse	**Inhalt**	**Eintrag**	**Bedeutung**
00 bis 01	Dateisignatur	42 4D	„BM“ für Bitmap
02 bis 05	Dateigröße	36 57 07 00	481.078 Byte
06 bis 09	Null	00 00 00 00	
0A bis 0D	Position der Bilddaten	36 04 00 00	Ab Byte-Adresse 436
0E bis 11	Länge des Headers ab hier	28 00 00 00	40 Byte
12 bis 15	Bildbreite in Pixel	20 03 00 00	800
16 bis 19	Bildhöhe in Pixel	58 02 00 00	600
1A bis 1B	Ebenenanzahl	01 00	1
1C bis 1D	Farbtiefe in Bit/Pixel	08 00	8 Bit
1E bis 21	Kompressionsmethode	00 00 00 00	keine Kompression
22 bis 25	Datenmenge ohne Header	00 53 07 00	480.000 Byte
26 bis 29	Horizontale Auflösung	20 1C 00 00	7.200 Pixel pro Meter
2A bis 2D	Vertikale Auflösung	20 1C 00 00	7.200 Pixel pro Meter
2E bis 31	Anzahl benutzter Farben	00 01 00 00	256 Farben
32 bis 35	Anzahl wichtiger Farben	00 01 00 00	256 Farben
36 bis 435	Farbpalette		256 Einträge zu je 4 Byte
436 bis 75735	Bilddaten		480.000 Byte

7.6 Verifikation der flimmerfreien Zeichenoperationen

Es stellt sich die Frage, ob der Zeichenbefehl des flimmerfreien Kopierens auch nach der Erweiterung des Video-Controllers auf einen DDR-Speicher korrekt ausgeführt werden kann. Bei Verwendung des SRAMs war dies gewährleistet, da mit dem Start des Kopiervorganges auf die Aktivierung des vertikalen Synchronisationssignals *vsync* gewartet wurde. Der Bildschirmaufbau begann erst später mit dem Auslesen der Daten aus dem Speicher. Während des Auslesens einer Bildschirmzeile war nur ein Zugriff auf den Speicher innerhalb von vier Takten notwendig, die drei übrigen Takte konnten für das Lesen bzw. Schreiben des Kopierbefehls und für das Anfordern der Sound-Daten genutzt werden.

Im schlimmsten Fall wurden innerhalb der vier Takte einmal Sound-Daten gelesen, es blieben aber immer noch zwei Takte für den Kopierbefehl. Damit war gewährleistet, dass der Bildschirmaufbau nie eher Daten aus dem Speicher anfordern konnte, als der Kopierbefehl die manipulierten Daten in den Speicher schrieb. Somit tritt kein Bildschirmflimmern auf, d. h. innerhalb der Darstellung des Bildes trat kein Wechsel von den ursprünglichen auf die neuen Bildschirmdaten auf. Es soll nun untersucht werden, ob dies auch beim modifizierten Video-Controller erfüllt ist.

Anders als beim SRAM-Zugriff erfolgt die Ausgabe der Daten bei DDR-SDRAMs nichtdeterministisch, man kann also nicht konkret angeben, wann die Daten am Ausgang des DDR-Controllers ausgegeben werden. Deshalb wird mittels Simulation eines Worst-Case-Szenarios versucht, abzuschätzen, ob das flimmerfreie Kopieren möglich ist.

Es wird das Kopieren einer ganzen 800 x 600-Bildpunkte-Bildschirmseite betrachtet. Zusätzlich zum Kopierbefehl soll an der Schnittstelle zum DDR-Controller maximaler Datenverkehr herrschen. Zu diesem Zweck werden Lesebefehle für die Einheit der Signaltöne abgesetzt und so eine dauernde Sound-Ausgabe simuliert.

Ziel der Simulation ist zu sehen, ob das Kopieren vom Auslesen der Daten für den Bildschirmaufbau eingeholt, bzw. überholt wird. Sollte also die Bildschirmeinheit Datenblöcke der Bildschirmseite anfordern, die von der Zeicheneinheit noch nicht in den Speicher geschrieben wurden, könnte die Flimmerfreiheit für den Kopierbefehl nicht gewährleistet werden.

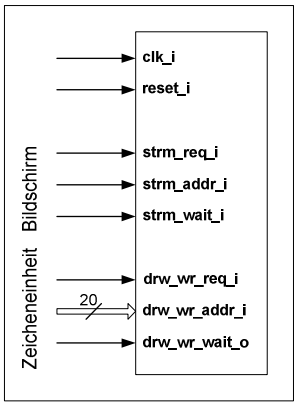

Abbildung 80: Monitor zur Beobachtung der Schnittstelle

Um beobachten zu können, welche Datenblöcke von der Einheit für den Bildschirmaufbau angefordert werden und welche von der Zeicheneinheit bereits in den Speicher geschrieben wurden, wird ein Monitor (Abbildung 80) an der Schnittstelle zum DDR-Controller eingesetzt.

Ausgangspunkt für die Betrachtungen ist das Ende der Ausgabe eines Bildes zum LCD. Dies wird durch den Wechsel des Signals VSYNC auf ‚0' signalisiert. Im Anschluss daran wird der Kopiervorgang gestartet. Die erste Zeile der neuen Bildschirmseite wurde bereits beim Erreichen des Endes der letzten Zeile der vorhergehenden Bildschirmseite geladen. Das bedeutet aber, dass die erste Zeile die Daten vor dem Kopiervorgang enthält und es nach der Darstellung der ersten Zeile zum Wechsel von den alten zu den neuen Daten kommt. Um dies zu verhindern, könnte man ein Neuladen der ersten Zeile veranlassen. Diese Änderung wurde aber als zu aufwändig bewertet, da sich die erste Zeile im Randbereich des Bildes befindet.

Das System befindet sich jetzt noch immer nach der letzten Zeile in der Austastpause, nach der die erste Zeile bereits vorgeladen wurde. Die Dauer der Austastpause (siehe Kapitel 3.4.2) ist relativ lange, wodurch in dieser Zeit 3 080 Blöcke im Speicher durch die Zeichenoperationseinheit manipuliert werden, während in den folgenden Austastpausen desselben Bildes, also dem Intervall des Ladens zweier Zeilen innerhalb eines Bildes lediglich 79 bis 84 Blöcke manipuliert werden können.

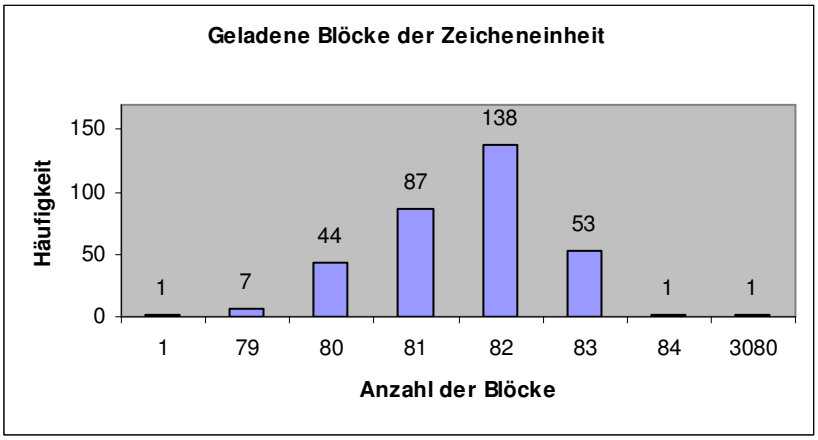

Abbildung 81: Verteilung der Zahl der kopierten Blöcke zwischen zwei Zeilen

In Abbildung 81 sind die in der Simulation auftretenden Häufigkeiten der Anzahl der kopierten Blöcke, für die Zeichenoperationen, zwischen dem Laden zweier Zeilen für den Bildaufbau, dargestellt. Der Wert 3 080 kommt im ersten Intervall dadurch zu Stande, dass die erste Zeile bereits geladen ist und eine längere Zeit zwischen dem Signal *vsync* und dem Laden der zweiten Zeile zur Verfügung steht. Der Wert 1 ist der verbleibende Block im letzten Intervall. Scheidet man diese beiden Werte aus, so werden im Mittel 81,58 Blöcke in einem Intervall geladen. Im Vergleich dazu werden für eine Bildschirmzeile in der Auflösung 800 x 600 Bildpunkte pro Zeile 50 128-Bit-Blöcke aus dem Speicher gelesen.

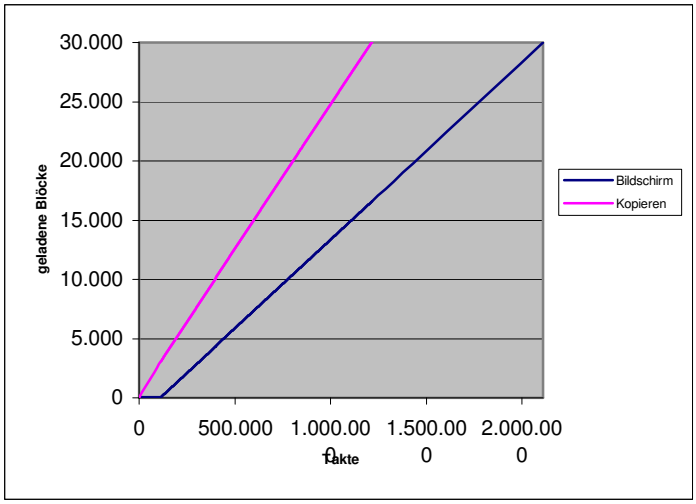

Abbildung 82: Vergleich Laden einer Bildschirmseite mit Schreibbefehlen von Secure Copy

Abbildung 82 zeigt, dass der Kopierbefehl schneller als das Laden der Daten für den Bildschirm abläuft, es kommt also innerhalb der Darstellung einer Bildschirmseite nicht zum Wechsel vom neuen zum vorherigen Bild. Schließlich ist noch zu betrachten, wie sich das Auffrischen des Speichers auswirken kann.

Der DDR-Controller führt den Autorefresh-Befehl zyklisch durch, wobei darauf geachtet wird, dass der gerade in Bearbeitung befindliche Befehl zum Abschluss gebracht werden kann.

Der gesamte Speicher muss alle 64 ms aufgefrischt werden, der DDR-Controller ist daher so konfiguriert, dass er alle 1 000 Takte einen Autorefresh-Zyklus einfügt. Zwischen dem Laden zweier Zeilen vergehen rund 3 330 Takte, d. h. es sind drei oder vier Refresh-Zyklen in dieser Zeitspanne enthalten. Da der DDR-Controller für einen Refresh-Zyklus für das Lesen 16, beim Schreiben 19 Takte zusätzlich benötigt, ist der Mehraufwand eines zusätzlichen Refresh-Zyklus verschwindend klein. Somit kann der Refresh-Zyklus, egal zu welchem Zeitpunkt er auftritt, in keinem Fall die Zeichenoperationen so sehr aufhalten, dass die Bildausgaben die Zeichenoperationen überholen. Wodurch trotz Refreshs eine flimmerfreie Darstellung des Bildes möglich ist.

8. Weiterführende Optimierungen

In Kapitel 7 wurde gezeigt, dass der Video-Controller seine Grundfunktion erfüllt, die Datenausgabe der Bilddaten aus dem Videospeicher an den Bildschirm erfolgt trotz Einsatzes eines SDRAMs rechtzeitig, um den Bildschirmaufbau zu gewährleisten. Durch Analyse der Struktur des Designs zeigte sich, dass im Bereich der Zeichen- und Kopierfunktionen die Möglichkeit der Beschleunigung der Verarbeitung besteht.

Der Video-Controller soll nun bezüglich Engpässen der Geschwindigkeit in der Datenverarbeitung betrachtet werden und mit verschiedenen Maßnahmen versucht werden, die Verarbeitungsleistung des Video-Controllers zu steigern.

8.1 Engpass bei der Datenausgabe

Eines der Designkriterien war von Anfang an, dass die Schnittstelle des Video-Controllers trotz Modifikation für den DDR-Speicher zu den Modulen unverändert bleiben sollte. Dabei existiert nur ein Datenausgabe-Port für die Einheiten Bildschirm, Zeichenoperationen und Signaltöne. Dies führt aber zwangsweise zu Kollisionen zwischen der Datenausgabe an die Bildschirmeinheit und an die beiden anderen Einheiten Signaltöne und Zeichenoperationen. Verwendet man für die beiden Einheiten Bildschirm und Lesen der modifizierten Schnittstelle statt eines gemeinsamen Datenports für jede der beiden Einheiten einen eigenen Datenport, so können keine Kollisionen auftreten und der Datendurchsatz kann gesteigert werden, da es zu keinen Verzögerungen mehr kommen kann (Abbildung 83).

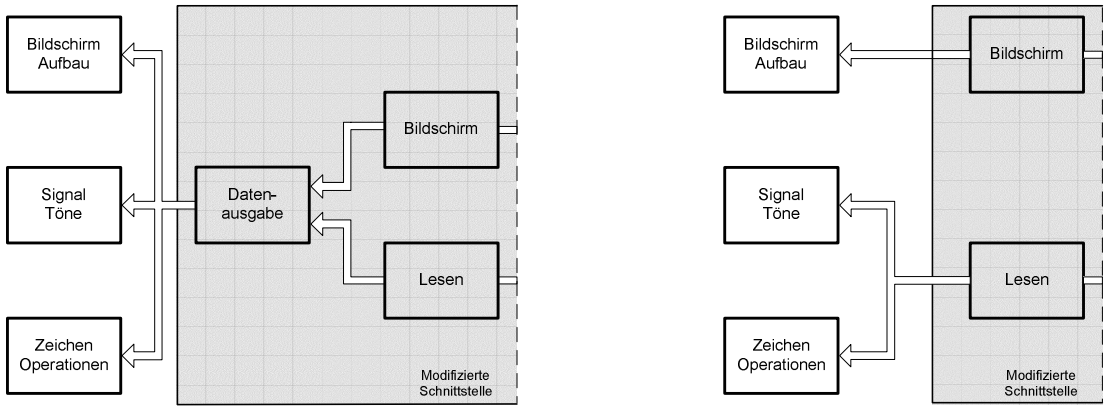

Abbildung 83: Erweiterung der modifizierten Schnittstelle auf zwei Ausgabeports

Um den Geschwindigkeitszuwachs dieser Maßnahme zu bestimmen, wird als Beispiel das Kopieren einer Bildschirmzeile simuliert. Es wird der ungünstigste Fall für die modifizierte Schnittstelle mit Datenausgabe-Einheit angenommen. Bilddaten werden über die Einheit Zeichenoperationen kopiert, während von der Einheit Bildschirm eine Zeile ausgegeben wird. Dadurch kommt es an der Einheit Datenausgabe oft zu Kollisionen. Nun wird verglichen, wie sich der Datendurchsatz durch die Verwendung der Variante mit getrennten Datenausgängen verbessert.

Tabelle 25: Beschleunigte Datenausgabe durch Vermeidung von Kollisionen

Datenausgabe über	Zeitdauer	Takte	Prozent
gemeinsames Ausgabe-Port	15176800 ps	1244	100,00
getrennte Ausgabe-Ports	12176400 ps	1162	93,41

In Tabelle 25 sind die Zeitdauern der beiden Varianten aufgelistet, man sieht, dass die Ausgabe über getrennte Datenausgabe-Ports eine Datendurchsatzsteigerung von 6,59 Prozent erreichen kann.

8.2 Engpass Komponente Lesen

Schwachpunkt der modifizierten Schnittstelle ist die Einheit Lesen. In dieser Einheit wird nur jeweils ein 128-Bit-Block aus dem Speicher geladen und die adressierten Daten innerhalb dieses Blocks an die anfordernden Einheiten Signaltöne und Zeichenoperationen ausgegeben. Erst wenn Daten aus einem anderen Datenblock angefordert werden, wird der entsprechende Block aus dem Speicher geladen.

Während der Ladezeit wartet die Einheit Zeichenoperationen und kann die Ausführung des gerade aktiven Befehls erst wieder nach dem Eintreffen des nächsten benötigten Datenblockes fortsetzen. Für Zeichenoperationen, die beim Einlesen rechteckiger Bildausschnitte benachbarte Datenblöcke benötigen, wäre es von Vorteil, neben der Verwendung des Datenblockes bereits den nächsten benötigten Datenblock zu laden.

Daher wird nun versucht, die Einheit Lesen der modifizierten Schnittstelle so zu erweitern, dass neben dem adressierten 128-Bit-Datenblock auch der Folgeblock geladen wird. Dadurch kann im Regelfall der linearen Zugriffe nahtlos ohne Unterbrechungen auf die Daten zugegriffen werden.

Das Zustandsdiagramm in Abbildung 84 stellt das Prinzip, nach dem die modifizierte Variante der Einheit Lesen arbeitet, dar. Ausgehend vom Zustand Start wird bei einem Lesebefehl der adressierte Datenblock aus dem Speicher angefordert und im Zustand Load1 auf das Eintreffen dieser Daten gewartet.

Mit dem Ready-Signal werden die eingetroffenen Daten in das erste Datenregister geschrieben, der Datenblock mit der nächsten Adresse angefordert und in den Zustand Load2 gewechselt. In diesem Zustand erfolgt bereits die Datenausgabe für Lesebefehle auf den ersten Datenblock. Wird vor dem Eintreffen des zweiten Datenblockes versucht, auf Daten eines anderen Blockes zuzugreifen, wird der Lesebefehl in einem Register gesichert und in den Zustand Waiting übergegangen, um auf das Eintreffen der Daten zu warten. Lesebefehle im Zustand Load2, die trotz gleichzeitigem Eintreffen des nachfolgenden Datenblockes nicht erfüllen werden können, werden gesichert. Während die Annahme weite-

rer Lesebefehle verweigert wird, wird über den Zustand Missed der gesicherte Befehl abgearbeitet, der nachfolgende Datenblock angefordert und anschließend der Zustand Load2 eingenommen.

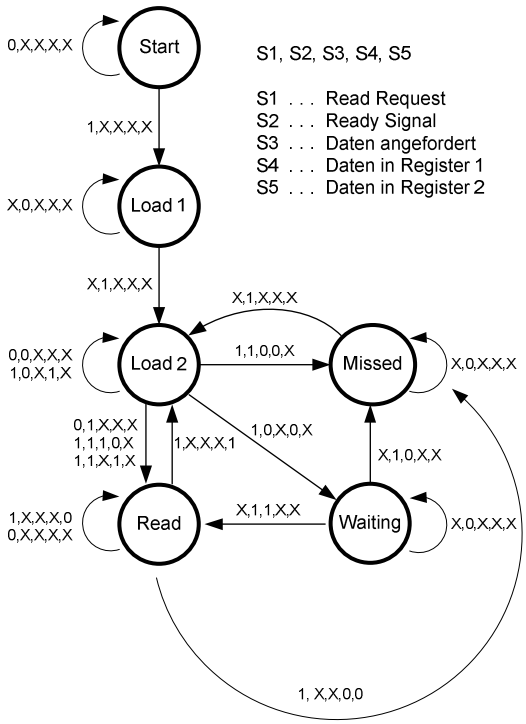

Abbildung 84: Zustandsdiagramm der Einheit Lesen mit zwei Datenregister

Im Zustand Read wird aus dem ersten 128-Bit-Blockregister gelesen. Sobald ein Zugriff auf den zweiten Datenblock erfolgt, wird der nachfolgende Block geladen und in den Zustand Load2, in dem auf das Eintreffen des nachfolgenden Datums gewartet wird, gewechselt.

Das Blockschaltbild dieser Alternative ist in Abbildung 85 dargestellt, die Steuereinheit (CTRL) nimmt die Lesebefehle entgegen und fordert die benötigten Daten aus dem DDR-Speicher an. Beim Eintreffen der Daten werden diese in den Registern für den aktuellen oder den Folgeblock gespeichert und je nachdem, in welchem Block sich die Ausgabedaten befinden, per Multiplexer ausgegeben.

Die Geschwindigkeitsgewinne dieser Erweiterung sind abhängig von der Abfolge der Adressen der Lesebefehle. Werden vorwiegend Zeichenoperationen mit rechteckigen Bildbereichen, die breit sind, verwendet, so ist der Einsatz der Einheit Lesen mit zwei 128-Bit-Datenblöcken sinnvoll. Ist dies nicht der Fall und werden vorwiegend schmale Bildbereiche verwendet, so ist der Einsatz der einfacheren Einheit Lesen mit nur einem Datenblock sinnvoller. Bei einer vertikalen Linie z. B. müsste man anderenfalls für jeden einzelnen Bildpunkt zwei Datenblöcke aus dem Speicher lesen, wodurch in diesem ungünstigen Fall sogar eine langsamere Verarbeitung als in der Variante mit einem Register erreicht werden kann. Welche Ausführung der Komponente Lesen verwendet werden sollte, hängt von der Charakteristik der verwendeten Befehle der speziellen Anwendung ab.

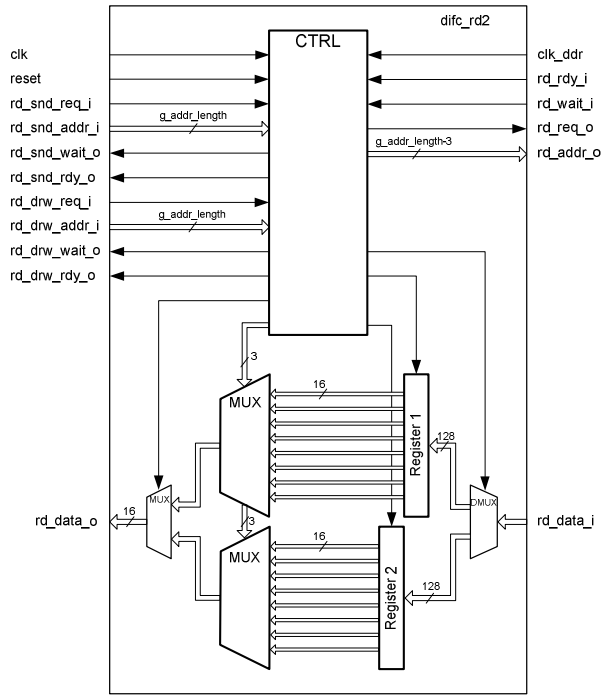

Abbildung 85: Blockschaltbild der Variante der Einheit Lesen mit zwei Datenregistern

Um die beiden Varianten für unterschiedliche Kopierbefehle mit verschiedener Breite der zu kopierenden Rechtecke vergleichen zu können, wurde jeweils ein Kopierbefehl mit einer identen Anzahl von Bildpunkten durchgeführt und die Anzahl der nötigen Takte zur Durchführung bestimmt (Tabelle 26). Dabei wurde die erste Lösung mit nur einem geladenen Block mit der Variante mit zwei Blöcken verglichen, und die Änderung der Verarbeitungsgeschwindigkeit in Prozent angegeben.

Tabelle 26: Bildausschnitt mit 800 Bildpunkten

Breite Bildpunkte	Höhe Bildpunkte	Block 1 Takte	Block 2 Takte	Änderung der Zeitdauer in %
800	1	1181	814	-31,08
400	2	1192	838	-29,70
200	4	1193	889	-25,48
100	8	1260	1003	-20,40
50	16	1405	1260	-10,32
25	32	1628	1471	-9,64
20	40	1724	1820	5,57
16	50	2079	1614	-22,37
10	80	1745	2381	36,45
5	160	3162	4745	50,06
2	400	7543	11823	56,74
1	800	14454	23822	64,81

Unter einer Breite von 20 bis 25 Bildpunkten verliert die Variante ihren Vorteil, denn es wird nur einer der zwei geladenen 128-Bit-Blöcke verwendet. Auf den zweiten muss zwar gewartet werden, die nach-

folgende Adresse befindet sich aber in der nächsten Zeile, es wird in den Zustand Missed gewechselt, in dem der Ladevorgang erst begonnen werden muss. Hier erweist sich das Vorladen des Blockes mit der nachfolgenden Adresse als Nachteil.

Auffällig ist der Bildausschnitt mit den Größen 16 mal 50 Bildpunkten. Trotz geringerer Breite ist die Abarbeitung des Kopierbefehls schneller als der Bildausschnitt 20 mal 40 Bildpunkte. Das ist damit zu begründen, dass 16 Bildpunkte genau in einem 128-Bit-Block Platz haben, man für 20 Bildpunkte aber zwei Blöcke benötigt.

Welche der beiden Varianten Geschwindigkeitsvorteile bringt, hängt, wie schon oben erwähnt, vom Format der am häufigsten verwendeten Bildausschnitte ab. Um die Geschwindigkeit weiter zu erhöhen, könnte man eine zusätzliche Logik implementieren, die treffsichere Prognosen liefern könnte, welcher Block als nächstes verwendet werden wird. Da es sich bei den Bildbereichen um rechteckige Ausschnitte handelt, ist nach der ersten Zeile erkennbar, wie breit ein Bildausschnitt ist und wann der Zeilenwechsel in der nächsten Zeile erfolgen wird. Diese Grenzen könnten mittels Zähler, der bei fortlaufender Adressierung erhöht wird, ermittelt werden. Damit könnte erreicht werden, dass am Ende einer Zeile eines Bildausschnitts bereits der Anfang der nächsten Zeile geladen wird, ohne den Zustand Missed auflösen zu müssen. Aus zeitlichen Gründen konnte diese Logik allerdings nicht realisiert werden.

9. Zusammenfassung und Ausblick

Es ist gelungen, den Video-Controller IP-Core zu modifizieren, um ein DDR-SDRAM mit dem DDR-Controller des Instituts für Computertechnik verwenden zu können. Die Speicherschnittstelle wurde dahingehend modifiziert, dass der Speicherzugriff für die Module des Video-Controllers nicht nach einem Zeitschlitzverfahren zugeteilt, sondern nach festgelegten Prioritäten vergeben wird. Damit und mit dem Vorladen von Bildschirmzeilen konnte auch für den nichtdeterministischen Zugriff auf einen dynamischen Speicher die unterbrechungsfreie Ausgabe der Daten für den Bildschirmaufbau sichergestellt werden.

Für die Verifikation des modifizierten Video-Controllers wurde das Gesamtsystem, bestehend aus dem Video-Controller sowie einer CPU, dem DDR-Controller und einem DDR-SDRAM simuliert. Um die Auswertung der Simulationsergebnisse zu automatisieren, wurde eine Software erstellt, die anhand der erteilten CPU-Befehle die zu erwartenden Simulationsergebnisse berechnet. Dadurch konnte die korrekte Arbeitsweise durch einen einfachen Vergleich der Inhalte der Dateien festgestellt werden.

Während die Funktion des Video-Controllers erfüllt wurde, konnte die Zieltaktfrequenz des Video-Controllers nicht erreicht werden. Sollte der Video-Controller implementiert werden, müssen am Design Änderungen vorgenommen werden. Die Berechnungen, die vom Video-Controller innerhalb eines Taktes durchgeführt werden müssen, sind in einigen Pfaden im Design relativ aufwändig. Diese müssten somit, um eine höhere Taktfrequenz erreichen zu können, auf mehrere Takte aufgeteilt werden, da sich hierdurch für die sich ergebenden Teilberechnungen eine längere Taktperiode ergeben würde. Somit müsste die Logik im FPGA derart geändert werden, dass die Berechnungen an den kritischen Stellen gepipelined erfolgen bzw. zusätzliche Registerstrufen eingeführt werden.

Als Engpass in der Ausführungsgeschwindigkeit der Zeichenoperationen erwies sich das Einlesen der zu manipulierenden Bilddaten. Durch Erweiterung auf zwei Ausgabeports lassen sich Kollisionen bei der Datenausgabe vermeiden und eine Steigerung der Verarbeitungsgeschwindigkeit erzielen. Weiters wurde ein Ausbau der Einheit Lesen vorgenommen. Durch Vorladen der nächst benötigten, zu manipulierenden Daten können diese früher bereitgestellt werden. Der Erfolg dieser Maßnahme hängt allerdings von der Charakteristik der zu bearbeitenden Bildausschnitte ab. Ob der zusätzliche Hardware-Aufwand gerechtfertigt ist, muss im jeweiligen Anwendungsfall entschieden werden. Zukünftig könnte eine Erweiterung der Logik auch treffsicherere Prognosen liefern, welcher Datenblock als nächstes benötigt wird. Dadurch wäre eine weitere Steigerung der Ausführungsgeschwindigkeit der Zeichenoperationen auf Kosten zusätzlicher Logik im FPGA erreichbar.

Um die Qualität des Video-Controllers zu verbessern, könnte die Bildauflösung erhöht werden. Die nächst höheren üblichen Auflösungen sind 1024 x 768 und 1280 x 1024 Bildpunkte. In Tabelle 27 sind die Timingparameter zur Bildschirmsynchronisation für beide Auflösungen und eine Bildwiederholfrequenz von 60 Hz angegeben.

Tabelle 27: Timingparameter der Synchronisationsimpulse [2]

Auflösung	**1024 x 768**	**1280 x 1024**
Sichtbarer Bereich	1024	1280
Front Porch	24	48
HSYNC	136	112
Back Porch	160	248
Gesamtpixel	1344	1688
Sichtbare Zeilen	768	1024
Front Porch	3	1
VSYNC	6	3
Back Porch	29	38
Gesamtzeilen	806	1066
Pixeltakt	65 MHz	108 MHz

Um eine Bildwiederholfrequenz von 60 Hz zu erreichen, ist z. B. bei der Auflösung 1024 x 768 ein Pixeltakt von 65 MHz notwendig. Dies wird durch eine Erhöhung der Taktfrequenz des Video-Controllers auf 130 MHz erreicht.

Tabelle 28 gibt darüber Auskunft, wie viele Takte zum Laden einer Zeile benötigt werden. Für das Beispiel aus Kapitel 6.2.6 mit 244 Takten maximaler Wartezeit bei gefülltem Befehls-FIFO und unter Berücksichtigung des Taktverhältnisses zwischen Video-Controller und DDR-Controller, benötigt man zum Laden einer Bildschirmzeile im ersten Fall etwa die Dauer der Austastpause, im zweiten Fall reicht die Austastpause nicht mehr aus. Durch Vorladen der nächsten Bildzeile während der Darstellung einer Zeile können die Bilddaten aber in jedem Fall rechtzeitig in den Zwischenspeicher geladen werden.

Tabelle 28: Takte zum Laden einer Zeile in den Zwischenspeicher der Einheit Bildschirmaufbau

Auflösung	**1024 x 768**	**1280 x 1024**
Austastpause	320	408
128-Bit Blöcke pro Zeile	64	80
erforderliche Taktfrequenz des Video-Controllers	130 MHz	216 MHz
Frequenzverhältnis DDR- zu Video-Controller	1:1	2:1
Takte für das Laden einer Zeile	308	648

Eine weitere Möglichkeit, den Video-Controller zu verbessern, wäre die Farbtiefe zu erhöhen. Da mit dem DDR-SDRAM auch mehr Speicher als beim ursprünglichen Video-Controller mit SRAM zur Verfügung steht, ist die Einsparung von Speicherplatz kein vorrangiges Ziel mehr. Für die Darstellung ei-

nes Bildpunktes im Speicher könnten statt 8 beispielsweise 16 Bit verwendet werden. Verwendet man je 5 Bit für jede der Farben Rot, Grün und Blau, so könnten 32 768 statt lediglich 256 verschiedenen Farben dargestellt werden.

Notwendig dafür wäre allerdings ein Umbau des Video-Controllers, denn es müssten:

- Bilddaten zu jedem zweiten anstatt zu jedem vierten Takt aus dem Speicher gelesen werden
- Zeichenoperationen auf 16-Bit-Wörter anstatt auf 8-Bit-Wörter durchgeführt werden
- Farbregister von 8 Bit auf 15 Bit pro Farbe erweitert werden

Außerdem müsste für eine Erhöhung der Farbtiefe betrachtet werden, welchen Einfluss die höhere Anzahl der Speicherzugriffe (bei Erhöhung auf 16 Bit eine Verdoppelung) auf die Zeichenoperationen hat – eventuell wäre es möglich, dass bei entsprechenden Auflösungen keine Flimmerfreiheit mehr gewährleistet wäre, da die Zeichenoperationen gegebenenfalls nicht mehr innerhalb der Darstellung einer Bildschirmseite abgeschlossen werden könnten.

Schlussendlich wäre zu überlegen, für den Video-Controller weitere aktuell sowie zukünftig übliche Speichertechnologien wie DDR2 und DDR3 einzusetzen. Die hierfür nötigen Änderungen wären wohl mit überschaubarem Aufwand möglich, da es sich bei diesen Speichertypen ebenfalls um dynamischen Speicher handelt und somit das Verhalten entsprechend ähnlich wäre und die modulare Struktur des modifizierten Video-Controllers ermöglichen würde, dass sich sämtliche Änderungen auf die Speicherschnittstelle beschränken.

Wissenschaftliche Literatur

[HER2003] Richard Herveille: „VGA/LCD CORE v 2.0 Spezifications", 2003

[TIE90] U. Tietze, Ch. Schenk: *Halbleiterschaltungstechnik*, Springer-Verlag 1990

[BÄH02] Helmut Bähring: *Mikrorechnertechnik II,* Springer-Verlag 2002

[SIE03] Siemers, Sikora: *Taschenbuch der Digitaltechnik,* Fachbuchverlag Leipzig 2003

[HPM00] Hans-Peter Mesmer: *PC Hardwarebuch*, Addison-Wesley Verlag 2000

[BEI04] Beierlein, Hagenbruch: *Mikroprozessortechnik*, Fachbuchverlag Leipzig 2004

[RHE92] Rhein, Freitag: *Mikroelektronische Speicher*, Springer-Verlag 1992

[KRO99] Thomas Kropf: *Introduction to Formal Hardware Verification*, Springer-Verlag 1999

[BER06] Janick Bergeron: *Writing Testbenches Using System Verilog*, Springer-Verlag 2006

[JED03] JEDEC STANDARD: *DDR SDRAM Specification*, März 2003

[WOI07] Woitowitz, Urbanski: *Digitaltechnik*, Springer-Verlag 2007

[WIE07] Ingo Wieser: *Ein Controller für doppelt getakteten dynamischen Speicher*, Institut für Computertechnik, TU Wien, 2007

[FRO] Frößl, Kropf, Schneider: *Bewertung Temporaler Logiken für die Hardware-Verifikation*, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe

Internetreferenzen

[1] http://www.opencores.org/

[2] http://www.tinyvga.com/

[3] http://www.micron.com

Abbildungsverzeichnis

Abbildung 1: Komponenten des bestehenden Videosystems mit SRAM 1
Abbildung 2: DDR-Controller beim Einsatz eines DDR-Speichers 3
Abbildung 3: Monochromer Text-Mode-Video-Controller VGA 80x40 4
Abbildung 4: Struktur VGA/LCD-Controller [HER2003] 5
Abbildung 5: Bilddarstellung in Computersystemen 7
Abbildung 6: Prinzip der Kathodenstrahlröhre [HPM00] 7
Abbildung 7: Zeilenweiser Bildschirmaufbau (Non-Interlaced) 8
Abbildung 8: Zeilendarstellung 8
Abbildung 9: Interlaced-Darstellung 9
Abbildung 10: Struktur des Video-Controllers 10
Abbildung 11: Speicheraufteilung des Videospeichers 12
Abbildung 12: Ports der Registereinheit 14
Abbildung 13: Register VGA_Control 15
Abbildung 14: Register VGA_STATUS 15
Abbildung 15: Register VGA_INTSTATUS 16
Abbildung 16: Entity für den Bildschirmaufbau 18
Abbildung 17: Aktiver Bildbereich 18
Abbildung 18: Bildschirmsynchronisationssignal HSYNC 19
Abbildung 19: Bildschirmsynchronisationssignal VSYNC 19
Abbildung 20: Timing-Diagramm für das Auslesen der Bildschirmdaten 20
Abbildung 21: Ein- und Ausgabeports der Einheit Zeichenoperationen 21
Abbildung 22: Zustandswechsel der Zeicheneinheit 22
Abbildung 23: Timing-Diagramm des Lesebefehls 23
Abbildung 24: Timingdiagramm des Schreibbefehls 23

Abbildung 25: Ein- und Ausgangsports der Einheit Signaltöne 24
Abbildung 26: Signal s_dac_2M 24
Abbildung 27: Ausgabe der Signaltöne 25
Abbildung 28: Ablaufsteuerung der Ausgabe der Signaltöne 26
Abbildung 29: Befehlsformat für die Signaltonausgabe 26
Abbildung 30: Ein- und Ausgangsports der Speicherschnittstelle 28
Abbildung 31: Priorisierung an der Schnittstelle zum Speicher 29
Abbildung 32: Timing-Diagramm Datenausgabe Speicherinterface 29
Abbildung 33: 6-Transistorzelle [SIE03 S. 266] 30
Abbildung 34: Struktur eines SRAM-Speichers [RHE92, S. 51] 31
Abbildung 35: DRAM-Speicherzelle [SIE03, S. 268] 33
Abbildung 36: Adressierung eines DRAM-Bausteins [BÄH02, S. 172] 34
Abbildung 37: Page Mode [HPM00] 36
Abbildung 38: Lesezugriff im Paged/Interleaved-Modus 37
Abbildung 39: Read-Befehl mit Burstlänge von 4 [JED03] 40
Abbildung 40: Anwendungsschnittstelle Blockschaltbild [WIE07] 41
Abbildung 41: Kommunikation mit dem DDR-Controller – Schreiben [WIE07] 42
Abbildung 42: Kommunikation mit dem DDR-Controller - Lesen 43
Abbildung 43: Struktur der modifizierten Schnittstelle 45
Abbildung 44: Zustandswechsel der Komponente Bildschirm 46
Abbildung 45: Nachladen des Dualport-RAMs 47
Abbildung 46: Struktur der Komponente Bildschirm 48
Abbildung 47: Steuerprozess LOAD_CTRL 49
Abbildung 48: Anfordern der Daten aus dem DDR-Speicher 50
Abbildung 49: Schreiben des Dualport-RAMs durch den Prozess WRITE_DPRAM 51
Abbildung 50: Datenausgabe durch den Prozess READ_DPRAM 52
Abbildung 51: Timing-Bedingungen für Schreibbefehl mit Zeilenwechsel auf gleiche Bank 53
Abbildung 52: Struktur Komponente Lesen 55
Abbildung 53: Steuerung des Ladevorganges und der Datenausgabe 56
Abbildung 54: Laden des Registers Data 57
Abbildung 55: Struktur Komponente Schreiben 59

Abbildung 56: Übernahme des Registerinhalts in das Dualport-RAM 60
Abbildung 57: Speicherung des Inhalts des Dualport-RAMs in den DDR-Speicher 62
Abbildung 58: Arbiter für den Zugriff auf den DDR-Controller 64
Abbildung 59: Priorisierung der Speicherzugriffe 65
Abbildung 60: Prozess ARBIT 66
Abbildung 61: Erzeugung der Haltesignale 67
Abbildung 62: Ausgabe der Lesedaten 67
Abbildung 63: Steuerprozess ARBIT 68
Abbildung 64: Testbench [SIE03] 70
Abbildung 65: Ebenen der Verifikation [BER06] 71
Abbildung 66: Aufbau der Simulation 72
Abbildung 67: Ports der modellierten ELAN CPU 73
Abbildung 68: Entity der LCD-Simulation 73
Abbildung 69: Simulationsmodell für die Laufzeitverzögerungen auf dem Board [WIE07] 74
Abbildung 70: Aufbau des verwendeten Speichermodells [WIE07] 74
Abbildung 71: Monitor check_strm 76
Abbildung 72: Rechteck zeichnen 77
Abbildung 73: Kopieren und mit definierter Farbe einfärben 77
Abbildung 74: Kopieren mit Transparenz 77
Abbildung 75: Kopieren und Einfärben mit definierter Farbe 78
Abbildung 76: Kopieren von rechteckigen Bildausschnitten 78
Abbildung 77: Verifikationssoftware 79
Abbildung 78: Automatisierter Dateivergleich 79
Abbildung 79: Aufbau einer Bitmap-Datei 80
Abbildung 80: Monitor zur Beobachtung der Schnittstelle 81
Abbildung 81: Verteilung der Zahl der kopierten Blöcke zwischen zwei Zeilen 82
Abbildung 82: Vergleich Laden einer Bildschirmseite mit Schreibbefehlen von Secure Copy 83
Abbildung 83: Erweiterung der modifizierten Schnittstelle auf zwei Ausgabeports 84
Abbildung 84: Zustandsdiagramm der Einheit Lesen mit zwei Datenregister 86
Abbildung 85: Blockschaltbild der Variante der Einheit Lesen mit zwei Datenregistern 87

Tabellenverzeichnis

Tabelle 1:Timingparameter für 800 x 600 Bildpunkte und 60 Hz (VESA) 9
Tabelle 2: Bedeutung der Registerstellen von CPU_CONTROL 15
Tabelle 3: Bedeutung der Registerstellen von CPU_STATUS 16
Tabelle 4: Bedeutung der Registerstellen von VGA_INTSTATUS 16
Tabelle 5: Weitere vom Video-Controller verwendete Register 17
Tabelle 6: Konstanten der Synchronisationssignale 19
Tabelle 8: Belegung der Schieberegisterstellen zur Signaltonausgabe 27
Tabelle 9: Zugriffsarten auf das Speicherinterface 27
Tabelle 10: Anschlüsse des SRAM-Speichers 32
Tabelle 11: SDRAM-Speicherbefehle [JED03] 38
Tabelle 12: Befehlssignale an den Controller [WIE07] 41
Tabelle 13: Maskierungsbits [WIE07] 42
Tabelle 14: Ein- und Ausgänge der Einheit Bildschirm 49
Tabelle 15: Zeitkonstanten des DDR-Controllers [JED03] 53
Tabelle 16: Maximale Anzahl der wartenden Befehle 54
Tabelle 18: Ein- und Ausgänge der Einheit Schreiben 60
Tabelle 19: Ein- und Ausgänge der Einheit Arbiter 65
Tabelle 20: Auswahl des Adressmultiplexers 66
Tabelle 21: Ports der Komponente Datenausgabe 68
Tabelle 22: Modellierte Signallaufzeiten [WIE07] 74
Tabelle 23: Generische Variablen des Speichermodells 75
Tabelle 24: Einträge im Header und Header-Information der vom LCD erzeugten Bitmaps 80
Tabelle 25: Beschleunigte Datenausgabe durch Vermeidung von Kollisionen 85
Tabelle 26: Bildausschnitt mit 800 Bildpunkten 87