OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

VGA-Grafikkarte auf einem FPGA

An der Fakultät für Informatik und Mathematik der
Ostbayerischen Technischen Hochschule Regensburg
im Studiengang
Technische Informatik

eingereichte

Projektarbeit in DAFP(Angewandte FPGA Programmierung)

Name:	Michael Braun
Matrikelnummer:	3113161
Name:	Korbinian Federholzner
Matrikelnummer:	3114621
Name:	Michael Schmidt
Matrikelnummer:	2907322
Erstgutachter:	Prof. Dr. Daniel Münch
Abgabedatum:	09.1.2020

Inhaltsverzeichnis

1. Einleitung

Die Grundaufgabenstellugn des eigenen Projektes war die Ausgabe einer Digitaluhr auf einem Bildschirm. Dazu sollte auf einem FPGA eine kleine Grafikkarte erstellt werde. Dazu wurde uns der Arty-7 zur Verfügung gestellt. Die Daten sollten nach den Berechnungen über eine VGA-Schnittstelle an eine Monitor geschickt werden. Die Auflösung sollte dabei 640x480 bei 60 Hz betragen. Dadurch wurden diverse Zeiten vorgegeben die im Kapitel Timing genauer betrachtet werden. Diese sind für die richtige Synchonisation zwischen Bildschirm und FPGA verantwortlich, denn beim Schreibvorgang wird bei der ersten Pixelreihe links oben begonnen und nach einer gewissen Zeit mit der nächsten fortgesetzt. Wird dabei die letzte Reihe erreicht, springt der „Schreibkopf" wieder zur ersten Reihe, wobei der Schreibvorgang nach einer festgelegten Zeitspanne mit einem neuen Bild begonnen wird.

2. Lösungsansätze

Unser Lösungsansatz für die Aufgabenstellung beeinhaltet eine Dreiteilung des Projekts, in Ausgabe, Speicher und Bilderstellung. Die Schnittstellen zwischen den drei Bereichen sind in zu sehen. Dabei beginnt der Ablauf dieser Grafikkarte in der Bilderstellungskomponente „IMG_CREATE". In dieser Komponente läuft die Uhr, deren Ausgabewerte durch „charmaps_ROM" in Bitmaps verändert werden. Diese werden danach in einzelne Bits aufgespalten und mit einer Farbkombination verknüpft und in den Speicher geschrieben.
Im Speicher...
In SYNC
Zusatzlich benötigt die Schnittstelle dabei noch die Komponente „Blank_Check". Diese verhindert dass außerhalb der fest vorgegebenen Schreibgrenzen Daten an den Bildschirm gesendet werden. Dazu überprüft die Komponente die Zeitintervalle und setzt wenn nötig den Ausgang auf „0".

3. Bilderstellung

3.1. Überblick

Der grundsätzliche Code für eine Uhr wurde zur Verfügung gestellt. Lediglich die Umformung in passende Zeichen, die in den Speicher geschrieben werden, wurde implementiert. Dabei werden die Zeichen in ASCII-Code von der Komponente „CLOCK_MACHINE" zur Verfügung gestellt. Diese werden dann basierend auf Zeichen und Zeile in eine Adresse umgewandelt, die an die Komponente „charmaps_ROM" übergeben werden. Dort wird dann das richtige Byte dazu ausgewählt. Dieses Byte zeigt durch „1" an das dieser Pixel gesetzt wird und bei „0" nicht. Dieses Byte wird dann in einzelne Bits/Pixel zerlegt und mit der Farbauswahl und einer Adresse, ebenfalls basierend auf Zeichen und Zeile an den Speicher übergeben.

3.2. Lösungsansätze

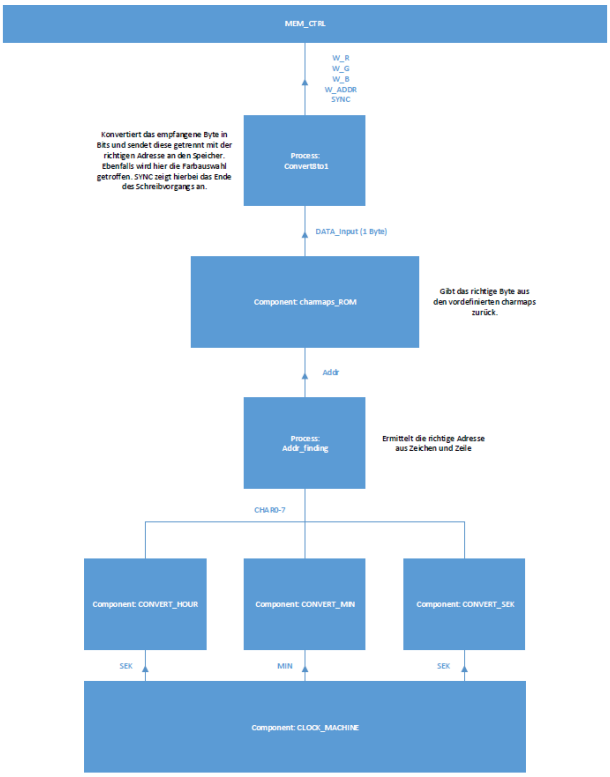

Abbildung 1: Schematische Darstellungder Komponente IMG_CREATE

Durch die Vorgabe der anderen Komponenten war bereits ein gewisser Weg vorgegeben. Einige Unterschiede könnten lediglich in der Programmierung entstehen. Die Trennung in zwei Prozesse ist hierbei sinnvoll da beide von unterschiedlichen

Takten beeinflusst werden. Außerdem haben beide einen anderen Kontext: „Addr_finding" das Ansprechen der Komponenten „charmaps_ROM" und „Convert8to1" das Weiterleiten der Farben mit der richtigen Adresse an den Speichercontroller.

3.3. Umgesetzte Lösung

Der grundsätzliche Aufbau wird in der folgenden Grafik 1 anhand eines Blockdiagramms dargestellt.

Dabei wird die gesamte Einheit unterhalb von „MEM_CTRL" als „IMG_Creation" beschrieben. Diese besteht aus den Komponenten „CLOCK_MACHINE" , „CONVERT" und „charmaps_ROM", sowie den Prozessen „Addr_finding" und „Convert8to1". Der Grundaufbau der Uhr beginnt dabei in der „CLOCK_MACHINE", welche die Zahlen der Uhr festlegt. Diese werden dann von der Komponente „CONVERT" in ASCII umgewandelt. Mit diesen Ausgangswerten beginnt der Prozess „Addr_finding". Dort werden die letzten sechs Zeichen der Bitfolge mit 16 multipliziert. Anschließend wird noch ein Zähler für die aktuelle Zeile hinzuaddiert. Dabei wird immer zuerst die erste Zeile des ersten Zeichens, dann die erste Zeile des zweiten Zeichens bearbeitet, da dies dem Lesevorgang aus dem Speicher heraus entspricht.

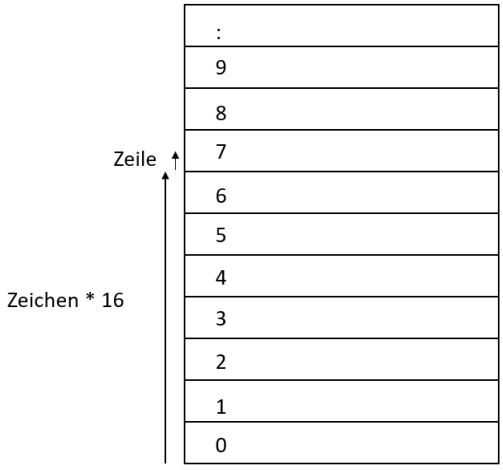

Abbildung 2: Charmap Aufteilung und Adresse

Dies ist dadurch möglich, da die letzten sechs Bit des ASCII-Codes genau der Adresse der Charmaps entsprechen. Die genaue Charmapaufteilung ist in der Grafik 2 zu sehen. Jedes Kästchen/Zeichen besteht dabei aus 16 Zeilen mit einer Länge von einem Byte. Nach einem kompletten Durchlauf der Zeichenfolge werden die Zähler zurückgesetzt. Da jedoch die Konvertierung im Prozess „Convert8to1" im Takt der

W_CLk erfolgt muss Hier der Takt acht mal langsamer sein. Dies wurde über einen weiteren Zähler realisiert. Dieser Zähler gibt außerdem den Takt für die Komponente „charmaps_ROM" vor, damit dem Process „Convert8to1" genügend Zeit bleibt das ankommende Byte auch zu verarbeiten. Um die durch die zeitlichen Diskrepanzen entstehenden Standardwerte der Komponente „charmaps_ROM" auszugleichen wird ein Standardwert gewählt der in keiner Bytekombination vorkommt, in diesem Fall „12". Diese wird dann im Prozess „Convert8to1" ausgefiltert. Dort wird dann das ankommende Byte von der Stelle 7 bis zur Stelle 0 durchgegangen und jeweils im normalen Takt eine Farbkombination in den Speicher geschrieben. Die Adresse wird dabei mit Konstanten verrechnet, die die Abstände zum Bildschirmrand festlegen, siehe 3. Die Adresse wird dabei nach folgender Gleichung berechnet:
Pixeladresse = (Vertikaler Abstand $*$ horizontaler Maximalwert) + (Zähler Zeilen $*$ horizontaler Maximalwert) + Horizontaler Abstand + Bit innerhalb des Zeichens (Zeichen $*$ 8)
Dadurch wird die richtige Speicherzelle mit den 12 Bit des Farbcodes gefüllt, vier Bit

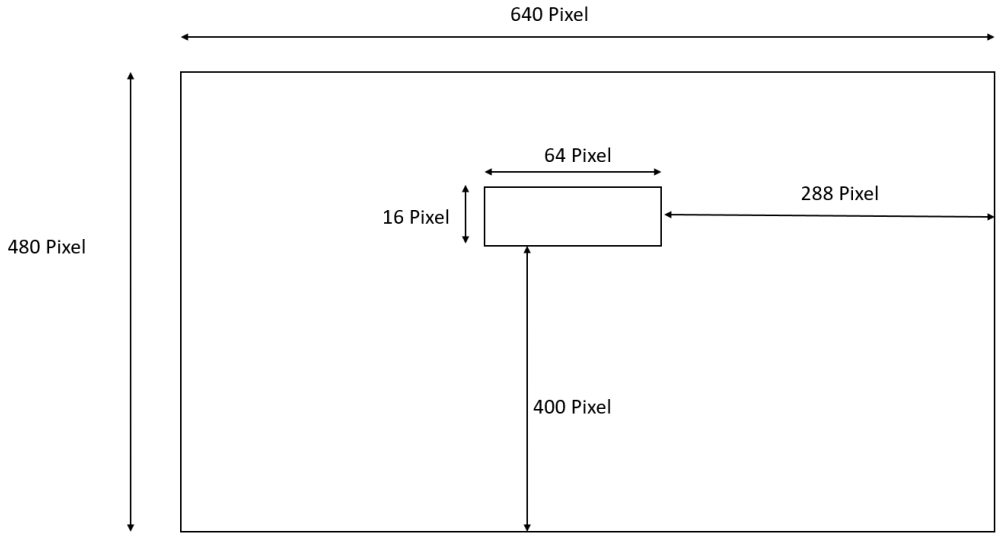

Abbildung 3: Aufteilung Bildschirm bzw. Adressraum

für jede Farbe. Dieser reicht daher pro Farbe von 0 bis 255. Beim Schreibvorgang wird somit nur der Bereich beschrieben in dem die Uhr auch zu sehen ist. Daher werden auch nur 16 Zeilen $*$8 Zeichen $*$8 Byte = 1024 Speicherzellen direkt angesprochen, wie in 3 zu sehen ist. Dabei wurde auch noch eine Funktion implementiert die die Farben verändert. So wird immer eine Farbe von dem Wert „0100" bis zum Maximalwert erhöht. Der Grundwert muss dabei ausreichend hoch angesetzt werden, damit dennoch eine Farbe zu sehen ist.

3.4. Mögliche Erweiterungen

Die Anzeige könnte um eine Anzeige in Millisekunden erweitert werden. Dies ist ebenfalls in die andere Richtung möglich (Tage/Wochen/etc.). Dazu müsste allerdings die Komponente „CLOCK_MACHINE" um die richtigen Zähler erweitert werden. Ebenfalls verändert werden müsste der Prozess „Addr_finding" und zwar um zwei weitere Zustände des Zählers „Count_Char". Die Adresse würde sich dabei nicht unterscheiden, da die gleichen Zeichen aus „charmaps_ROM" verwendet werden.
Eine weitere Erweiterung wäre die Ausgabe der Uhr in einem bewegten Modus, wie ältere Bildschirmschoner. Dazu müsste ein weitere Zähler eingefügt werden, der nach einem vollständigen Schreibvorgang weiter zählt und dabei die Adresse verändert. Dies bewirkt allerdings nur eine Verschiebung in horizontaler Richtung. Zusätzlich könnte auch noch ein zweiter Zähler eingeführt werden, der in vertikaler Richtung weiter zählt, nach dem Modus:
Vertikaler Zähler $*$ horizontaler Maximalwert = Vertikale Verschiebung
Dadurch würde dann eine diagonale Verschiebung erfolgen. Außerdem sollten noch Grenzen festgelegt werden, die verhindern, dass die Uhr nicht mehr zu sehen ist, da sie in einem Bereich angezeigt wird, der duch die Komponente „Blank_Check" nicht an den Bildschirm gesendet wird. Da die Grenzen der Uhr fest vorgegeben sind muss dazu lediglich eine Abfrage erfolgen. Bei einer sich bewegenden Uhr müssen allerdings alle nicht beschriebenen Pixel wieder auf „0" gesetzt werden, da ansonsten die Überbleibsel des vorherigen Schreibvorgangs noch zu sehen sind.

3.5. Zusammenfassung

4. Speicherverwaltung

4.1. Überblick

5. Timing und Blank_Check

5.1. Überblick

5.2.

6. Fazit und Ausblick

Anhang

A. Abkürzungsverzeichnis

B. Abbildungsverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Inhalt des Datenträgers

- ...
- ...