Bernburg
Dessau
Köthen

Hochschule Anhalt
Anhalt University of Applied Sciences

emw
Fachbereich
Elektrotechnik, Maschinenbau
und Wirtschaftsingenieurwesen

Masterarbeit

zur Erlangung des akademischen Grades
Master of Engineering (M. Eng.)

Hannes Knothe

Vorname Nachname

Elektro- und Informationstechnik,
MEF 2014, 4051460

Studiengang, Matrikel, Matrikelnummer

Thema:

Entwicklung einer Ansteuerung über einen FPGA für eine LED Wand aus WS2812B LEDs

Prof. Dr. Michael Brutscheck

1. Prüfer

Prof. Dr. Steffen Strauß

2. Prüfer

15.01.2018

Abgabe am

Selbstständigkeitserklärung

Hiermit erkläre ich, dass die Arbeit selbständig verfasst, in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt wurde und keine anderen als die angegebenen Hilfsmittel und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet wurden.

Leipzig, 15.01.2018

Ort, Datum

Unterschrift des Studierenden

Angaben zur Fachhochschule

Logo der Fachhochschule	
Name der Fachhochschule	Hochschule Anhalt
Abteilung	Fachbereich 6 - Elektrotechnik, Maschinenbau und Wirtschaftsingenieurwesen
Name des Betreuers	Prof. Dr. Michael Brutscheck

Kontaktdaten

Anschrift des Standortes, an dem die Arbeit verfasst wurde	Hochschule Anhalt Bernburger Straße 57, 06366 Köthen,
E-Mail-Adresse des Betreuers	michael.brutscheck@hs-anhalt.de

Kurzfassung

Gegenstand der hier vorgestellten Masterarbeit ist die Entwicklung einer Ansteuerung mittels eines Field Programmable Gate Array (FPGA) für eine LED Wand. Der Fachbereich Elektrotechnik, Maschinenbau und Wirtschaftsingenieurwesen (EMW) der Hochschule Anhalt hat eine LED Wand aus insgesamt 18.000 LEDs entwickelt. Diese sind dabei in 36 Ketten zu je 500 Stück miteinander verschalten. Ziel dieser Arbeit soll es sein, eine Ansteuerungslogik zu entwerfen, welche es möglich macht, Bilder oder Videodateien auf dieser Wand darzustellen. Als Hardware wird hierfür das Development and Education Board DE2-115 der Firma Terasic verwendet. Dieses bietet neben dem Cyclone IV FPGA von Altera auch die benötigten Hardwarekomponenten wie einen SD Karten Steckplatz oder einen ADV7180 Videodecoder, welche für die Umsetzung des Projektes benötigt werden.

Inhaltsverzeichnis

1 Motivation und Zielsetzung

1.1 Einleitung in die Thematik

Der Fachbereich EMW der Hochschule Anhalt hat eine LED Wand mit einer Höhe von 3,0 Meter und einer Länge von 5,4 Meter aus 18.000 LEDs entwickelt und gebaut. Dafür wurden 180 Pixel in der Breite und 100 Pixel in der Länge vom LED Typ WS2812B auf einem Trägermaterial angeordnet. Ursprünglicher Zweck dieser Wand war es, als Hintergrundstrahler für beispielweise Filmaufnahmen zu fungieren. Für diese Anwendung leuchten alle LEDs der Wand in der gleichen Farbe und Intensität. In der Abbildung 1 ist ein Bild der LED Wand in der Funktion als Hintergrundstrahler dargestellt.

Abbildung 1: LED Wand als Hintergrundstrahler, einfarbig leuchtend [HK17]

Generell gibt es für LED Wände in dieser Form ein breites Anwendungsspektrum als Werbe- oder Anzeigetafeln. Dafür ist es jedoch notwendig, dass sich ständig wechselnde Inhalte mit einer gewissen Frequenz auf der LED Wand darstellen lassen.

1.2 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist es, eine Ansteuerungslogik über einen FPGA für die LED Wand zu entwickeln. Dabei soll es einem Anwender ermöglicht werden, Bild- oder Videomaterial auf der LED Wand darzustellen. Als Hardware für die Ansteuerung soll hierbei das DE2-115 FPGA Evaluierungsboard von der Firma Terasic genutzt werden. Dieses bietet neben einem Cyclone IV FPGA von Altera diverse Peripheriebausteine, wie einen SD Karten Slot oder einen ADV7180 SDTV Video Decoder, welche neben dem FPGA für die Realisierung des Projektes mit genutzt werden sollen. Ein schematischer Signalflussplan des Gesamtprojektes ist in der folgenden Abbildung 2 zu sehen.

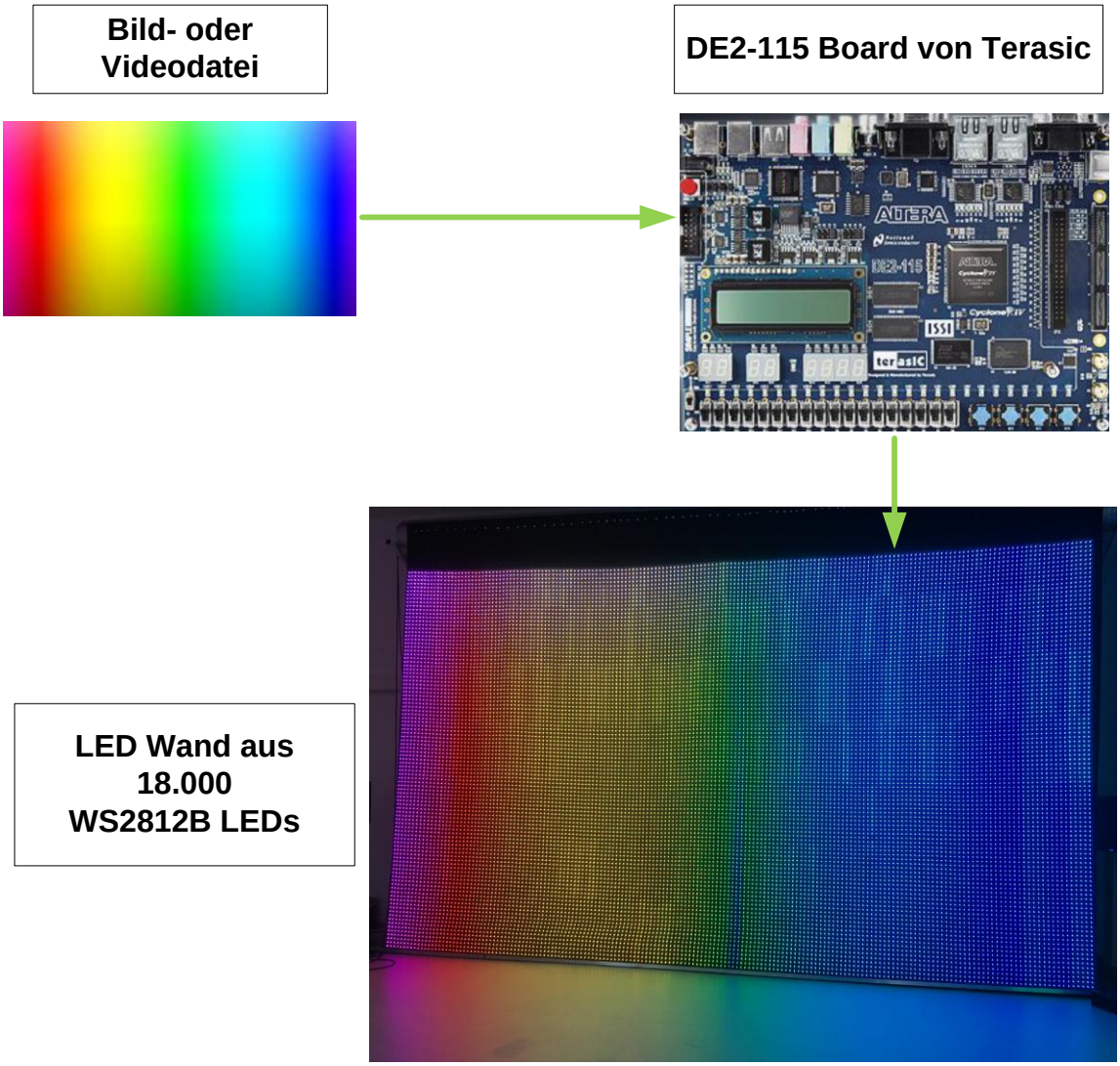


Abbildung 2: Schema des Gesamtprojektes [HK17]

2 Stand von Wissenschaft und Technik

In diesem Kapitel werden alle Komponenten und Themengebiete, welche die Voraussetzung zur Realisierung des Projektes bilden, abgehandelt. Begonnen wird dabei mit der einzelnen WS2812B LED. Es wird beschrieben, wie die Komponente aufgebaut ist, wie sie funktioniert und welcher Farbraum sich mit ihr darstellen lässt. Darauf aufbauend wird erläutert, wie eine Mehrzahl solcher Bauelemente verschaltet werden, um einen Strip zu bilden. Die Verschaltung mehrerer Strips wiederum bildet die Grundlage für die komplette LED Wand, für welche ebenfalls der Aufbau und die Funktionsweise erläutert wird.

Des Weiteren wird die verwendete Hardware für die Ansteuerungslogik beschrieben. Dabei wird zum einen auf das DE2-115 Entwicklungsboard von Terasic sowie auf den Cyclone IV FPGA von Altera näher eingegangen. Es werden ebenfalls die Entwicklungsumgebungen Quartus II und Eclipse, welche für die Programmierung und Konfiguration der Komponenten verwendet wurden, in einem gesonderten Kapitel beschrieben.

2.1 Der RGB Farbraum

Das RGB Farbmodell basiert auf den Lichtfarben Rot, Grün und Blau. Die Farben, welche in der Abbildung 3 zu sehen sind, werden additiv gemischt. Ausgehend von keinem Licht (Schwarz) entstehen mit zunehmender Intensität der Primärfarben hellere Töne, bis am Ende bei jeweils 100 Prozent Weiß entsteht. Grau entsteht bei identischer Intensität von Rot, Grün und Blau [MaWä17].

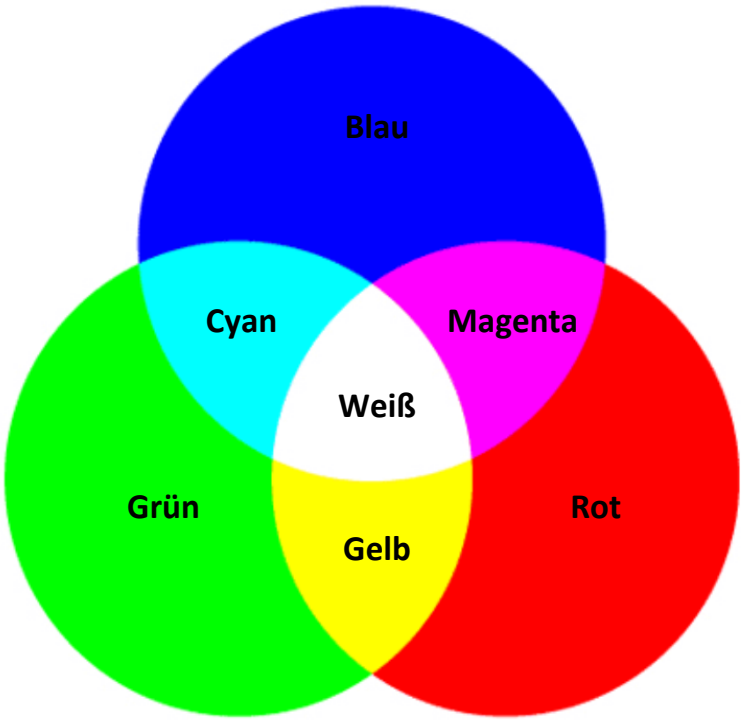


Abbildung 3: RGB Lichtfarben [HK17]

Es wird nun festgelegt, dass jeder der drei Farbkanäle acht Bit breit ist. Das heißt, es stehen 2^8 = 256 Abstufungen der Helligkeit für jede einzelne LED zur Verfügung. Somit hat ein Pixel, bestehend aus drei farbigen LEDs, eine Datentiefe von drei mal acht Bit, was 24 Bit pro Pixel entspricht. Darum lässt sich, wie in Gleichung 1.1 dargestellt, mit einem solchen Bildpunkt nur eine begrenzte Anzahl von Farben darstellen.

$$256\,Rotwerte \times 256\,Grünwerte \times 256\,Blauwerte = 16.777.216\,Farben \quad \textbf{Gleichung 1.1}$$

Dies hat zur Folge, dass der RGB Farbraum durch die begrenzte Datentiefe des Helligkeitswertes stark eingeschränkt ist. Mit anderen Worten, es lassen sich nicht alle sichtbaren Farben mit einem Pixel mit 24 Bit Datentiefe darstellen. Der verbleibende Farbraum, welcher auch der Standard für Computermonitore ist, nennt sich sRGB. Dieser ist in der Abbildung 4 dargestellt.

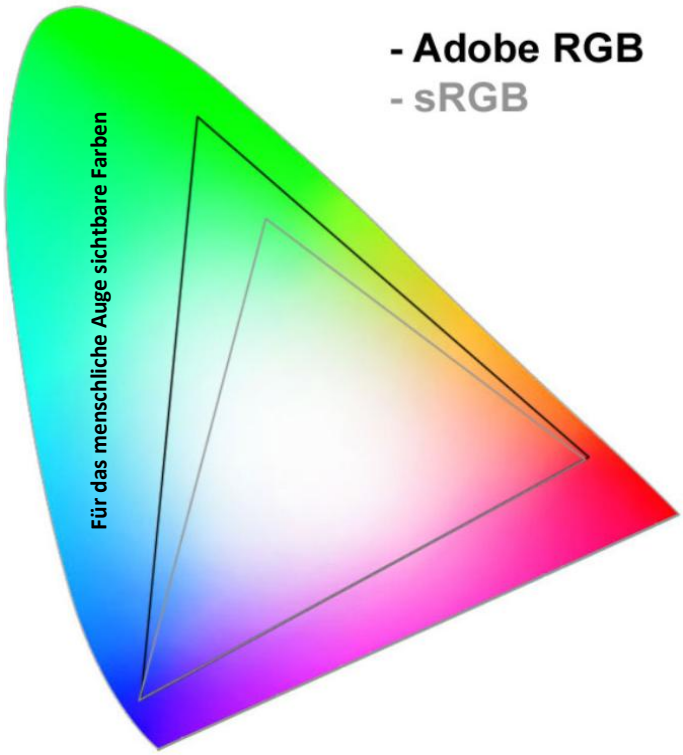


Abbildung 4: Schematische Darstellung des sRGB Farbraums [RGB17]

Die unterschiedlichen technischen Systeme der Farbdarstellung auf Monitoren sowie Geräteabweichungen führen zu unterschiedlicher Darstellung der gleichen Prozessfarbwerte [PeBü04]. Das heißt, es kann dazu kommen, dass Bilder, welche auf einer LED Leinwand dargestellt werden, im Gegensatz zum Monitor eine andere Farbdarstellung haben.

2.2 Das YCbCr Farbmodell

Das YCbCr Farbmodell wurde für das Digitalfernsehen nach der Norm PAL entwickelt. Da in dieser Masterarbeit auch eine Videodarstellung auf der LED Wand untersucht werden soll, bei der die Eingangsdaten im PAL Format vorliegen, wird dieses Modell kurz erläutert.

Der Name des Farbmodells besteht aus drei Abkürzungen, dabei steht „Y“ für die Grundhelligkeit, „Cb“ für die Blau-Gelb Komponente (Blue-Yellow Chrominance) und „Cr“ für die Rot-Grün Komponente (Red-Green Chrominance). Das Farbbild wird, wie in der Abbildung 5 dargestellt, in die einzelnen Komponenten Helligkeit, Gelb-Blau Komponente und Rot-Grün Komponente zerlegt. Damit zählt das Farbmodell zu den Helligkeit-Farbigkeit-Modellen, da ein Farbort nicht wie im RGB System durch drei Grundfarben angeben wird, sondern durch Helligkeit und Farbigkeit. Eine Umrechnung der Werte Y, Cb und Cr vom YCbCr Farbmodell in die den Rot-, Grün- und Blauwert vom RGB Farbmodell ist hierbei möglich.

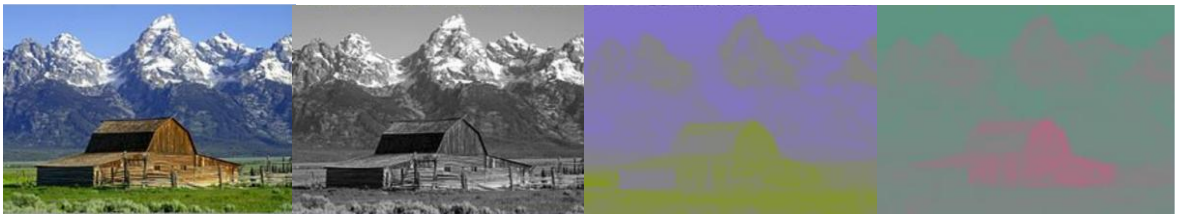


Abbildung 5: YCbCr Farbraum [WIKI17]

2.3 Die LED WS2812B

In diesem Kapitel wird die LED WS2812B genauer beschrieben. Dabei wird der Aufbau, die Funktionsweise sowie die Logik der Ansteuerung näher erläutert. Bei der LED vom Typ WS2812B handelt es sich um eine vollintegrierte RGB LED inklusive Ansteuerungslogik. Eine einzelne LED dieses Modells ist in der Abbildung 6 dargestellt, wobei die vier Anschlüsse "VSS" (Masse), "VDD" (Versorgungsspannung), "Din" (Datenleitung Eingang) und "Dout" (Datenleitung Ausgang) nach außen geführt sind.

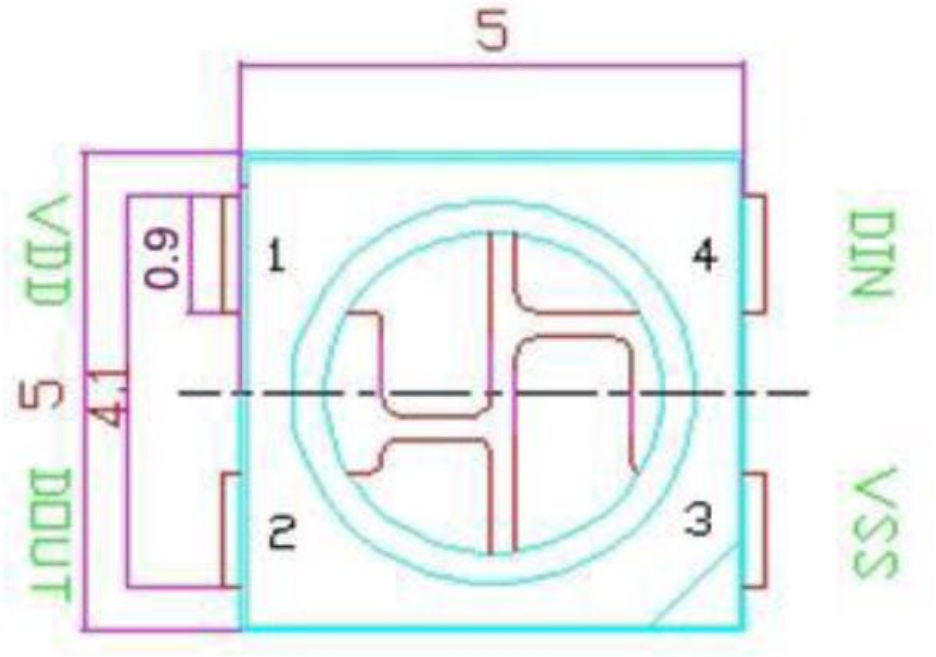


Abbildung 6: Einzelne WS2812B mit Anschlussbezeichnung [WS17]

Wie in der Abbildung 7 zu sehen, werden die einzelnen Pixel typischerweise auf einer Klebefolie mit integrierten Leiterbahnen verlötet, einem sogenannten Strip. Dies kann einzeln oder zu mehreren geschehen und wird in den folgenden Kapiteln näher erläutert.

Abbildung 7: Einzelne WS2812B LED auf einem Strip [HK17]

2.3.1 Aufbau der WS2812B

In der stark vergrößerten Darstellung der WS2812B in Abbildung 8 sind die einzelnen Komponenten gut zu erkennen. Der Controller I ist mit den einzelnen farbigen LEDs: II = grüne LED, III = rote LED und IV = blaue LED über gebondete Drähte verbunden. Aus diesem physischem Aufbau wird ersichtlich, warum die LEDs, wenn sie in Strips zu einer LED Wand verbaut werden, immer die gleiche Ausrichtung haben müssen. Eine veränderte Einbaulage würde später im Bild sichtbar sein, da die Leuchtquellen im Pixel eine andere Anordnung hätten. Um diesen Effekt zu verdeutlichen, ist im Anhang auf Seite v eine leuchtende WS2812B abgebildet. Dabei strahlen die einzelnen LEDs (Rot, Grün und Blau) zum Test mit einer Leistung von drei Prozent.

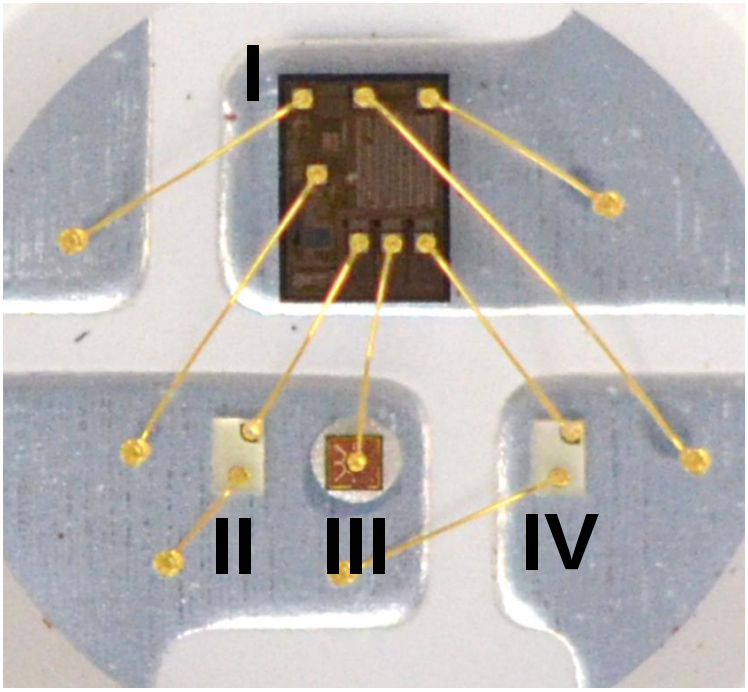


Abbildung 8: WS2812B vergrößert, [PS14]

2.3.2 Funktionsweise der WS2812B

Wie sich schon aus dem physischen Aufbau ableiten lässt, besteht eine WS2812B intern aus drei einzelnen farbigen LEDs. Jede dieser LEDs ist, wie in der Abbildung 9 zu sehen, an einem eigenen Analog-Digital Wandler angeschlossen. Dieser hat die Funktion, dass digitale acht Bit Helligkeitssignal mittels Pulsweitenmodulation (PWM) in die passende analoge Größe umzuwandeln. Hierbei wird die Helligkeit einer LED über den Strom geregelt. Je mehr Strom durch die Diode fließt, desto heller leuchtet sie.

Gesteuert werden die Analog-Digital Wandler dabei von dem internen Controller der WS2812B. Dieser benötigt als Eingangsinformation ein 24 Bit Datentelegramm mit den Farbinformationen für die drei einzelnen LEDs. Der genaue Aufbau des Telegramms wird im nächsten Kapitel näher erläutert.

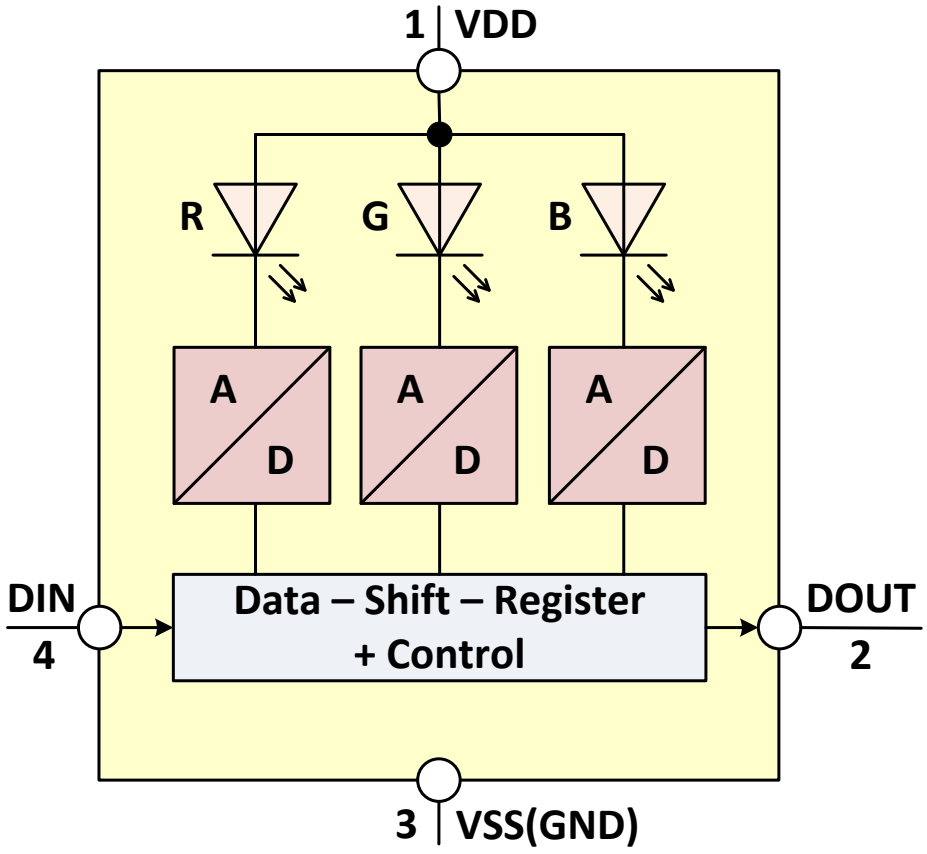


Abbildung 9: Interner Aufbau der WS2812B [HK17]

2.3.3 Aufbau des Datentelegramms zur Ansteuerung der WS2812B

Um die WS2812B zum Leuchten zu bringen, muss dem Bauteil ein 24 Bit Datentelegramm mit den Farbinformationen gesendet werden. Wie bereits im vorherigen Abschnitt 2.3.1 beschrieben, verfügt jede WS2812B intern über drei getrennte LEDs. Dabei stehen jedem der drei Farbkanäle Rot, Grün und Blau acht Bit für die Farbinformation zur Verfügung. Mit diesen acht Bit können insgesamt 255 verschieden Helligkeitsstufen für die drei Farben festgelegt werden, wobei ein Wert von 0 eine abgeschaltete LED und 255 eine LED mit 100 Prozent Leuchtleistung repräsentiert. Der sich nun daraus ergebende RGB Farbraum, welcher sich mit der WS2812B darstellen lässt, wurde bereits im Kapitel 2.1 näher beschrieben. Die drei acht Bit Werte der einzelnen Farben werden dem Pixel nun in einem 24 Bit Datenwort, wie in Abbildung 10 dargestellt, in einer bestimmten Reihenfolge gesendet. Dabei wird das höchstwertigste Bit des Grünkanals zuerst übertragen. Danach folgen der Rot- und Blaukanal.

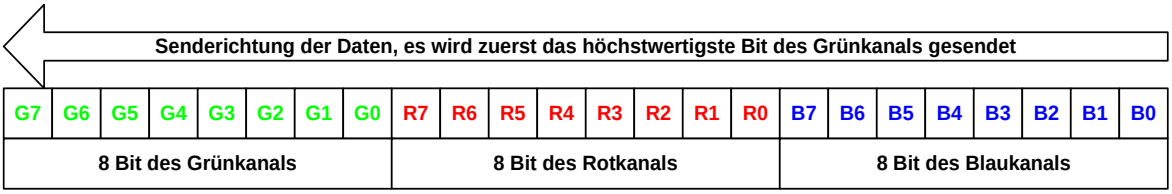


Abbildung 10: 24 Bit Datenwort mit Farbinformationen für WS2812B [HK17]

Ist die WS2812B am Spannungseingang VDD mit 5,0 Volt sowie am Eingang VSS mit einer Masse (0 Volt) beaufschlagt und werden ihr dabei Signale von der Steuerungslogik am Eingang „Din“ zugesendet, beginnt sie die Daten in ihrem internen Controller zu speichern. Die Werte werden dabei, wie in der Abbildung 9 zu sehen, in einem 24 Bit breiten Schieberegister gepuffert. Eine genaue Beschreibung des Timings und der benötigten Signalform erfolgt im Kapitel 3.1.

Nach der Übertragung des Datenwortes zur LED wird nun vom internen Controller auf Basis des Farbwertes von 0 bis 255 ein PWM Signal für die drei internen farbigen LEDs (Rot, Grün und Blau) der WS2812B generiert. Um diesen Vorgang auszulösen, benötigt die WS2812B ein Startsignal. Dies geschieht, wenn der Eingang „Din“ für mehr als 50 µs auf dem low Pegel gehalten wird. Dieses Startsignal wird im Kapitel 3.1.1 noch näher erläutert. Werden diese Vorgänge in der beschriebenen Reihenfolge durchgeführt, leuchtet das Pixel nun in der Farbe, die dem gesendeten 24 Bit Wert entspricht.

2.3.4 Zusammenschaltung von mehreren WS2812B zu einem Strip

Eine wichtige Eigenschaft einer WS2812B ist die Option, diese in Ketten von bis zu 2.000 Stück miteinander verschalten zu können. Dabei wird, wie in Abbildung 11 dargestellt, der Datenbusausgang "DO" einer LED mit dem Datenbuseingang "DIN" der nächsten LED verbunden. Im vorherigen Abschnitt wurde bereits das 24 Bit Datentelegramm zur Ansteuerung einer einzelnen Komponente beschrieben. Sollen nun mehrere zusammengeschaltete WS2812B über einen Datenbus angesteuert werden, müssen die Werte nacheinander von der Steuerung über die Datenleitung "DIN" in die erste LED geschrieben werden. Sobald eine LED das 25. Bit an ihrem Dateneingang erhalten hat, übergibt diese den Wert des ersten Bits über den Ausgang "D0" an den Eingang "DIN" der nächsten LED. Die Pufferspeicher der internen Controller, der zum Strip verschaltenen LEDs, verhalten sich hierbei wie ein Schieberegister. Gleichzeitig kommt es bei der Übergabe der Daten von einem Pixel zum anderen zu einem sogenannten "reshaping" der Codeform. Dieser Vorgang geschieht im internen Controller der Komponente und gewährleistet eine fehlerfreie Übergabe der Informationen von der ersten bis zur letzten LED im Strip.

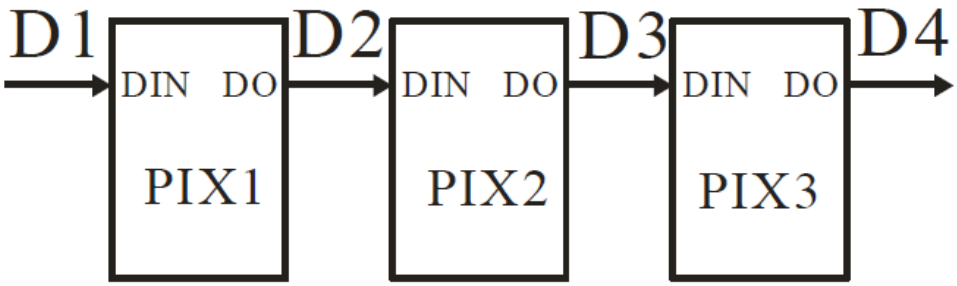


Abbildung 11: Zusammengeschaltete WS2812B LEDs, [WS17]

Somit werden für die Ansteuerung von beispielsweise drei LEDs, wie in der Abbildung 11 zu sehen, auf einem Strip gleich drei mal 24 Bit seriell in die erste LED geschrieben. Danach hat jede der drei LEDs einen mit 24 Bit codierten RGB Wert im Speicher des internen Controllers stehen. Wird nun für mehr als 50 µs ein low Pegel am Datenbus der ersten Komponente angelegt ("RES" Befehl), werden die gespeicherten Werte in die PWM Register des Controllers übertragen und die einzelnen Pixel beginnen in der gewünschten Farbe zu leuchten. Eine genaue Beschreibung der Datenübertragung von der Steuerung, welche mit einem Field Programmable Gate Array (FPGA) realisiert wurde, zur WS2812B, erfolgt in Kapitel 3.

2.4 Aufbau der LED Wand

Um eine Mehrzahl von einzelnen Strips als Hintergrundstrahler nutzen zu können, beziehungsweise um gegebenenfalls Bilder mit ihnen darstellen zu können, wurden diese in einer bestimmten Anordnung auf einem Moltongewebe aufgebracht. In der Abbildung 12 ist zu erkennen, wie dieser Prozess mit Hilfe einer Schablone durchgeführt wurde. Vorher wurden die Strips zu Ketten mit der benötigen Länge miteinander verbunden. Ein Strip besteht jeweils aus 100 einzelnen WS2812B LEDs. Insgesamt wurden 180 dieser Strips parallel auf dem Moltongewebe angeordnet.

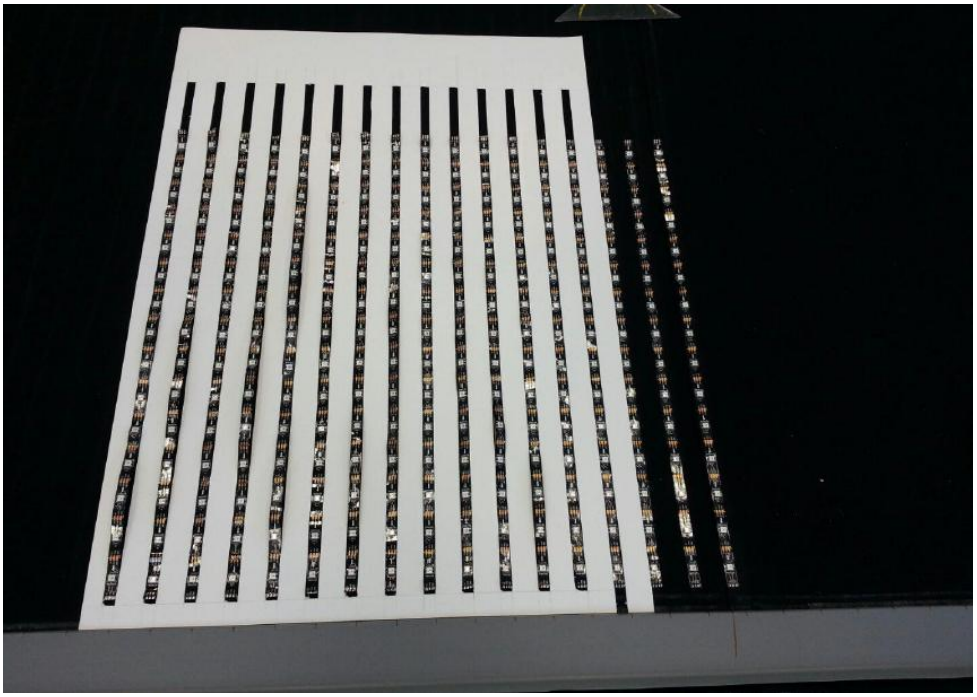


Abbildung 12: Aufbringung der Strips auf das Trägermaterial [CS14]

Anschließend wurden die Strips miteinander verschaltet. Im vorherigen Kapitel wurde bereits beschrieben, wie sich eine Vielzahl von WS2812B über eine einzige Datenleitung steuern lassen. Die tatsächliche Verschaltung des Datenbusses für einen Strip der LED Wand ist in der Abbildung 13 dargestellt. Es wurde dabei immer vom Ausgang "DO" der letzten WS2812B eines Strips mit 100 einzelnen LEDs ein schwarzes Kabel zum Eingang "DIN" der ersten LED des nächsten Strips gezogen. Diese dünne schwarze Leitung wurde mit auf dem Moltongewebe fixiert. Es wurden fünf Strips mit jeweils 100 WS2812B zu einem Strip zusammengefügt. Dieser Strip aus 500 einzelnen WS2812B kann später mit nur einer Datenleitung vom FPGA angesteuert werden. Der Aufbau, wie er für den Strip 1 in der Abbildung 13 dargestellt ist, wurde so für insgesamt 36 Strips wiederholt. Daraus ergibt sich eine 3,0 Meter mal 5,4 Meter große LED Wand aus 18.000 einzelnen LEDs.

Jeder einzelne Strip benötigt zusätzlich noch eine 5,0 Volt Spannungsversorgung. In der Entwicklungsphase der LED Wand wurde inklusive aller Reserven eine maximale Leistung von 0,3 Watt pro Pixel angenommen. Dies ergibt eine maximale Gesamtleistung von 5.400 Watt für die 18.000 Einzelkomponenten. In der Tabelle 1 sind die Leistungsdaten einer WS2812B aus dem Datenblatt des Herstellers dargestellt.

Farbe	Wellenlänge (nm)	Intensität (mcd)	Strom (mA)	Spannung (V)
Rot	620 - 630	550 - 700	16	1,8 - 2,2
Grün	515 - 530	1.100 - 1.400	16	2,8 - 3,1
Blau	465 - 475	200 - 400	16	2,9 - 3,2

Tabelle 1: Leistungsdaten der WS2812B [WS17]

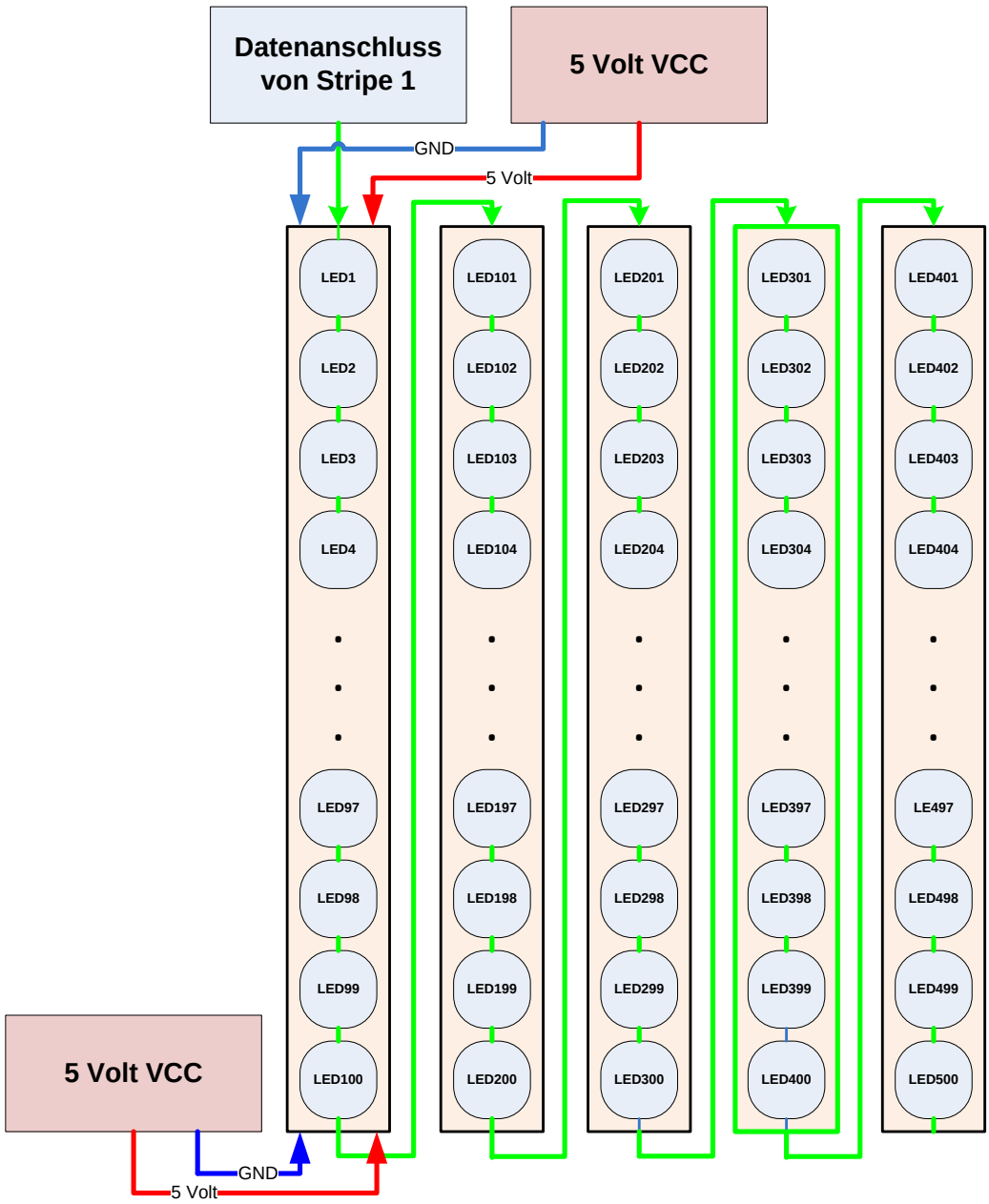


Abbildung 13: Verschaltung eines Strips der LED Wand [HK17]

2.4.1 Die Spannungsversorgung der LED Wand

In der Abbildung 13 ist die Spannungsversorgung exemplarisch für einen einzelnen Strip mit 100 Pixeln abgebildet. So wie dort dargestellt, wird jeder der 180 Strips mit der 5,0 Volt Spannung versorgt. Da bei einer einseitigen Einspeisung von oben oder unten der Spannungsabfall auf dem Strip zu groß wäre, um die letzte Komponente in der Anordnung noch zu versorgen, erfolgt die Einspeisung der Versorgungs-spannung beidseitig. Dafür wurden insgesamt 30 Spannungswandler mit einer Leistung von 200 Watt je Stück beim Bau der LED Wand verwendet. Dabei speisen jeweils 15 Wandler von oben und 15 Wandler von unten Spannung in die LED Wand ein. Ein Wandler ist hierbei jeweils parallel an 12 Strips angeschlossen. Zusätzlich dazu müssen sich alle Module ein gemeinsames Massepotential teilen.

Da es bei dem gleichzeitigen Zuschalten der 30 Spannungswandler zu einem sehr hohen Strom in der 230 Volt Zuleitung kommt, werden die Komponenten über eine Anlaufschaltung zeitlich versetzt eingeschaltet. Außerdem muss der obere und der untere Kreis über getrennte 230 Volt Zuführungen versorgt werden, da sonst der maximale Stromwert von 16 Ampere auf der Zuleitung überschritten werden würde.

2.5 Das DE2-115 Development and Education Board von Terasic

Das DE2-115 Board bietet die komplette Hardware, welche für die Umsetzung des Projektes benötigt wird. In der Abbildung 14 sind die einzelnen Komponenten des Boards dargestellt. Hauptbestandteil hierbei ist der Cyclone IV FPGA, welcher im nächsten Kapitel gesondert betrachtet wird. Für das Projekt wichtige Bestandteile sind neben dem FPGA, der Programmieranschluss via USB Blaster, der SD Card Socket, der 50 MHz Oszillator, der ADV7180 Videodecoder und der EPCS64 Speicherbaustein für die dauerhafte Speicherung der FPGA Parameter.

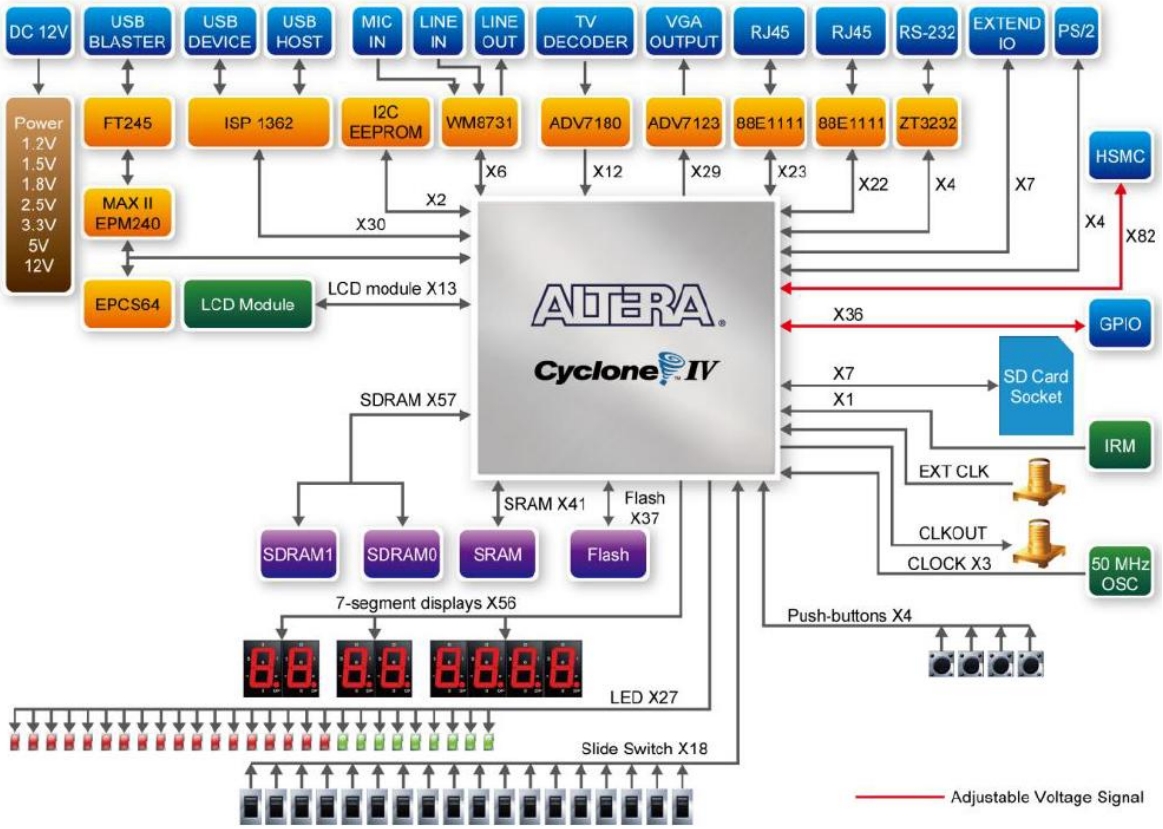


Abbildung 14: Blockschaltbild vom DE2-115 Board von Terasic [TE10]

Des Weiteren bietet das Board die Möglichkeit, die 36 Strips über eine einzelne Steckverbindung anzuschließen. Dies lässt sich über einen 40 poligen IDE Stecker realisieren. In der Abbildung 14 ist der Anschluss unter der Bezeichnung X36 (GPIO) dargestellt. An diesem liegen die Verbindungen zu 36 frei parametrierbaren Aus- oder Eingängen (GPIO[0] - GPIO[35]) des FPGAs an. Zusätzlich dazu gibt es noch einen weiteren 14 poligen IDE Stecker für Eingangs- und Ausgangspins auf dem Board. Hier werden noch einmal sieben zusätzliche I/O Pins (EX_IO[0] - EX_IO[6]) zur Verfügung gestellt. Auf diese "Extended_IO Pins" wird im Kapitel 3.3.2 noch näher eingegangen. Der Hersteller Terasic bietet zusätzlich zum DE2-115 eine fertige Pinkonfigurationsliste für die bereits am FPGA angeschlossenen Hardware-komponenten. Diese ist als Excel-Dokument mit auf der CD zur Masterarbeit unter dem Namen "DE2_115_pin_assignments" abgelegt.

2.6 Der Cyclone IV FPGA von Altera

Beim dem FPGA, welcher zur Umsetzung des Projektes genutzt wurde, handelt es sich um den EP4CE115F29C7N der Cyclone IV Familie von Altera. Der FPGA hat derzeitig einen Einzelpreis von circa 360,00 Euro. Die wichtigsten Kenndaten dieses Chips sind in der folgenden Tabelle 2 abgebildet. Um zu wissen, welche Vertreter dieser Chipgeneration noch verfügbar sind und welche Eigenschaften diese haben können, ist eine Aufschlüsselung des Namenscodes in der Abbildung 15 dargestellt.

Ressource	EP4CE115F29C7N
Logic elements (LEs)	114480
Embedded memory (kbits)	3888
Embedded 18 × 18 multipliers	266
General-purpose PLLs	4
Global Clock Networks	20
User I/O Banks	8
Maximum user I/O	528

Tabelle 2: Parameter des EP4CE115F29C7N FPGA von Altera [ALT16]

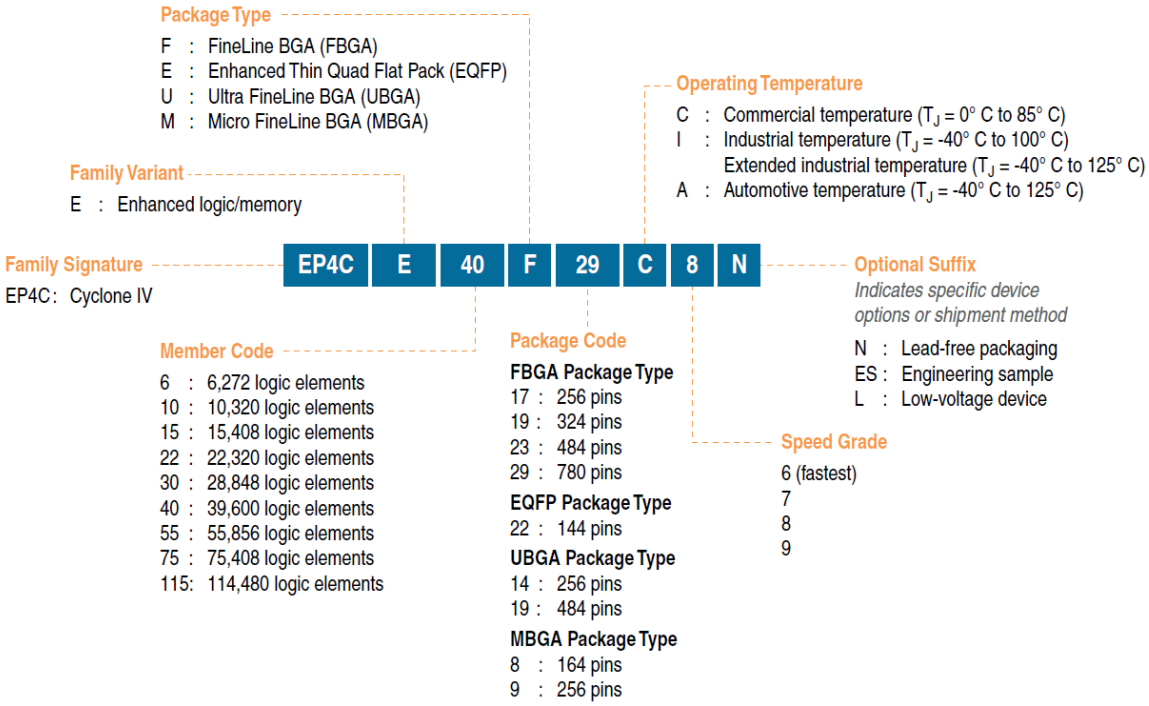


Abbildung 15: Schlüssel für Namensgebung des Cyclone IV FPGA [ALT16]

Die Ein- und Ausgangspins des Cyclone IV FPGAs lassen sich laut den Input/Output (I/O) Spezifikationen auf verschiedene Logiken einstellen. Der Ausgangspuffer jedes I/O Pins verfügt über eine parametrierbare Steuerung. Dabei lässt sich jeder einzelne Pin auf einen der in Tabelle 3 aufgeführten I/O Standards einstellen.

3,3 V LVTTL	3,3 V LVCMOS	3,0 V LVTTL	3,0 V LVCMOS
2,5 V LVTTL/LVCMOS	PCI	PCI-X	

Tabelle 3: Input/Output Standards des Cyclone IV FPGA

2.7 Die Pegelwandlung für die WS2812B Ansteuerung

Wie sich aus der Tabelle 3 erkennen lässt, liegt die maximale Ausgangsspannung für einen I/O Pin des Cyclone IV FPGAs bei 3,3 Volt. Für die Ansteuerung einer WS2812B hingegen werden laut Datenblatt Rechtecksignale mit einer Spannungsamplitude von 5,0 Volt benötigt. Die Spezifikation des Datentelegramms der WS2812B wird im Kapitel 3.1 noch genauer erläutert. Was sich jedoch jetzt schon aus diesen Informationen schließen lässt, ist die Tatsache, dass für die korrekte Ansteuerung einer LED mittels des Cyclone IV FPGAs eine Hardware benötigt wird, welche eine Pegelwandlung der Ausgangssignale von 3,3 Volt auf 5,0 Volt realisiert.

Im Rahmen dieser Masterarbeit wird hierfür der SN74HCT04 eingesetzt, welcher in der Abbildung 16 dargestellt ist. Dabei handelt es sich um einen Inverterbaustein von Texas Instruments. Dieser bietet insgesamt sechs Inverterschaltungen integriert in einem Bauteil. Er invertiert dabei logisch die Eingangspegel der Signale von den Anschlüssen 1A - 6A und gibt die invertierten Signale an den Anschlüssen 1Y - 6Y wieder aus. Liegt ein high Pegel von 3,3 Volt an einem Eingang an, so wird dieser am Ausgang mit 0 Volt GND Spannung repräsentiert. Liegt hingegen ein low Pegel von 0 Volt an einem Eingang an, so wird dieser mit der Spannung, die an V_{CC} anliegt, am Ausgang ausgegeben. Hierfür werden 5,0 Volt Versorgungsspannung (V_{CC}) für den SN74HCT04 gewählt. Somit lässt sich mit insgesamt sechs dieser Bauelemente die Pegelwandlung für die 36 Strips der LED Wand realisieren. Es muss lediglich bei der Treiberentwicklung darauf geachtet werden, dass die Signale, die der FPGA ausgibt, vorher intern einmal invertiert werden, um nach der Pegelwandlung der richtigen Logik zu entsprechen.

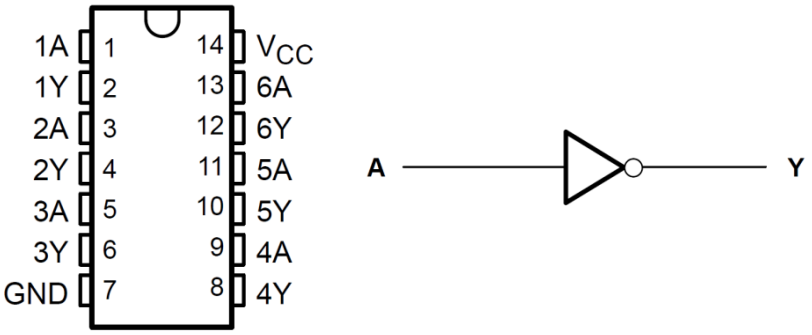


Abbildung 16: Der SN74HCT04 Inverterbaustein [TI16]

Intern besteht der Inverterbaustein aus einer Verstärkerschaltung. Die Steig- und Fallzeit des Ausgangspegels in Abhängigkeit vom Eingangspegel beträgt laut Datenblatt 500 Nanosekunden. Dies ist für die Anwendung des SN74HCT04 als Pegelwandler absolut ausreichend. Beim Bau der LED Wand wurde für die insgesamt sechs verbauten Inverterbausteine eine Schaltung entwurfen. Eine Abbildung davon befindet sich im Anhang auf der Seite vi.

2.8 Der NIOS-II Soft Core Prozessor

Der NIOS-II Prozessor ist eine synthetische CPU, die in einen Altera FPGA implementiert werden kann. Dabei handelt es sich um eine umfangreiche Form einer Designanweisung, die in einer Hardwarebeschreibungssprache, wie VHDL oder Verilog, eine elektrische Schaltung beschreibt. Diese Schaltung wirkt dann nach außen wie ein standardmäßiger Mikrocontroller, der sich beschreiben und konfigurieren lässt.

Eine reine Hardwarelösung via FPGA ist für diverse Anwendungen nicht immer unbedingt die beste Wahl. Ein Projekt kann unter Umständen sehr umfangreich werden, wenn versucht wird, eine Anwendung, wie beispielsweise die Daten von einer SD Karte auszulesen, als reine Hardwarelösung zu entwickeln. Genau für solche Anwendungen lohnt sich der Einsatz eines NIOS-II Prozessors.

Eine solche Implementierung eines Mikrocontrollers auf einem FPGA bietet eine Reihe von Vorteilen. Zum einen lässt sich der Controller genau auf die Anwendung hin anpassen. Es können beispielsweise Hardwarekomponenten, wie Ein- und Ausgänge, Speicher und Timer frei hinzugefügt oder angepasst werden. Zum anderen kann der Mikrocontroller direkt in Verbindung mit anderen Schaltungsteilen, die ebenfalls auf dem selben FPGA laufen, zusammenarbeiten. In der Abbildung 17 ist beispielhaft dargestellt, aus welchen einzelnen Komponenten ein NIOS-II bestehen kann. Die Konfiguration der CPU erfolgt dabei über das Tool „QSys“, welches von Altera für den FPGA mit bereit gestellt wird.

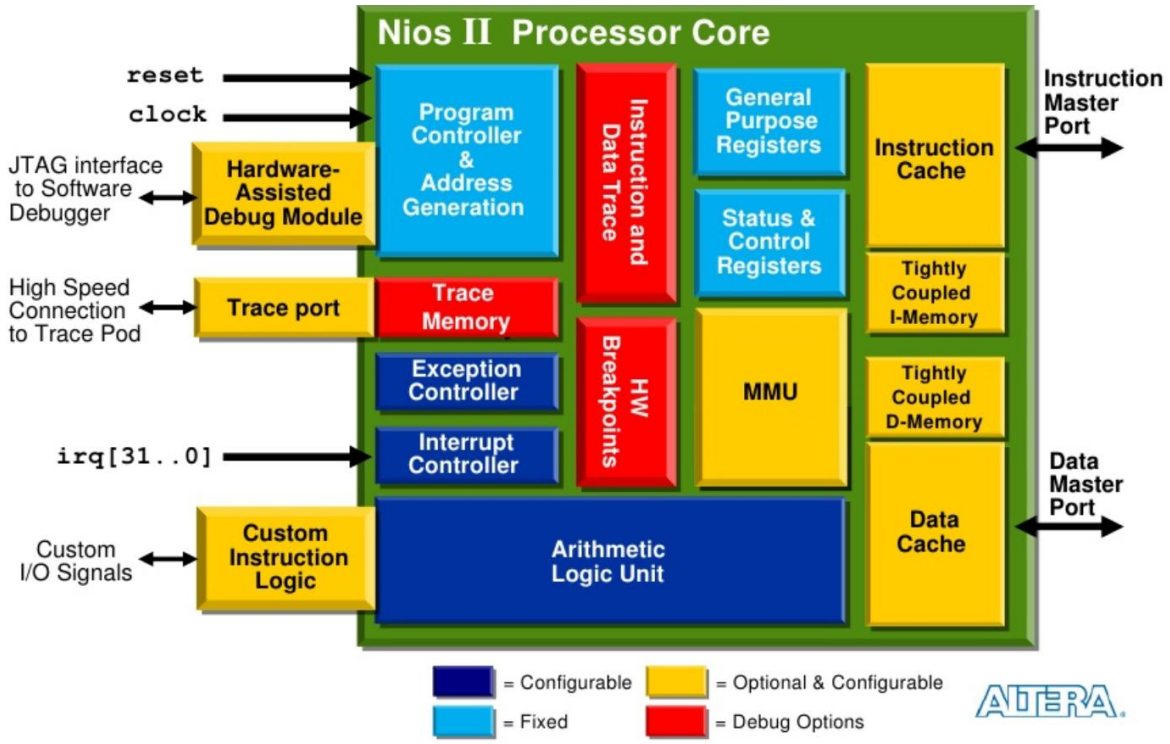


Abbildung 17: NIOS-II Soft Core Prozessor [ALT10]

2.9 Die Entwicklungsumgebungen Quartus II und Eclipse

Der Boardhersteller Terasic bietet dem Nutzer neben dem DE2-115 Board auch eine umfangreiche Toolchain von Altera. Dabei handelt es sich zum einen um die Software Quartus II. Sie ist eine Entwicklungsumgebung für die Konfiguration von Altera FPGAs. Dabei werden innerhalb der Software eine Vielzahl von unterschiedlichen Prozessen abgearbeitet, um aus dem Quellcode einer Hardwarebeschreibungssprache eine elektronische Schaltung zu synthetisieren. Die einzelnen Schritte vom Entwurf bis zur fertigen Schaltung sind in der Abbildung 18 aufgeführt.

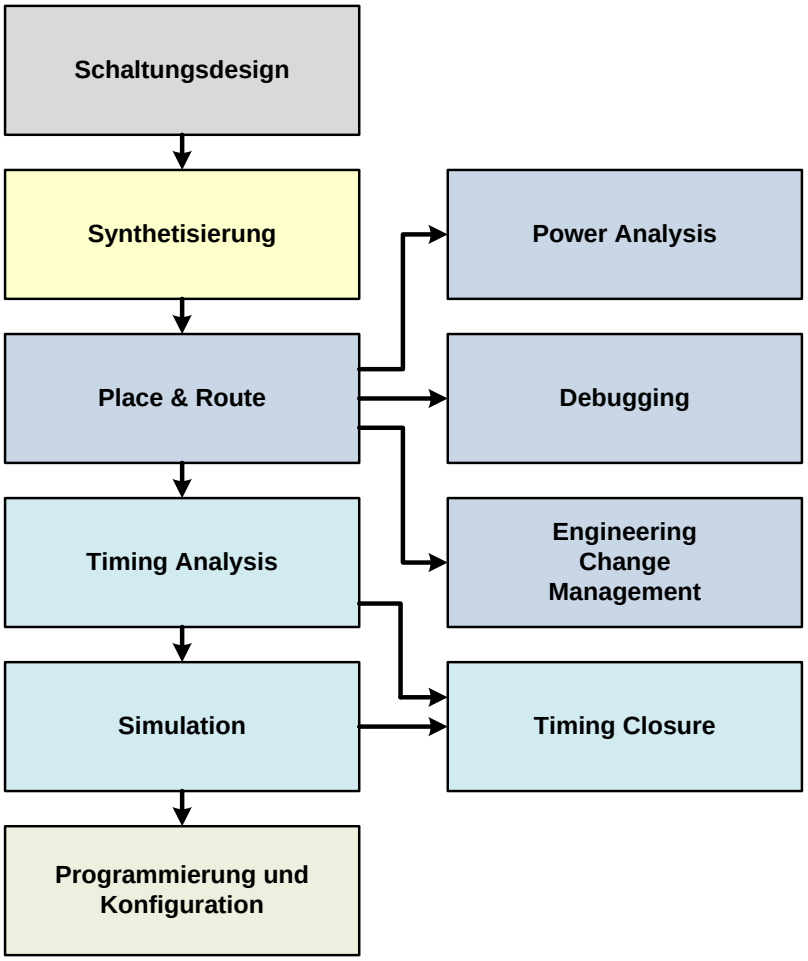


Abbildung 18: Design flow im Quartus II [PR17]

Neben der Synthetisierung des Quellcodes finden auch eine Reihe von Analysen und Optimierungen im Verlauf des Übersetzungsprozesses im Quartus II statt. Diese haben das Ziel, die Schaltung so energiesparend und zeitoptimiert wie möglich auf dem gewählten Bauteil unter zu bringen. Des Weiteren wird geprüft, ob beispielsweise die Logikelemente vom FPGA für die errechnete Schaltung ausreichen. Zudem lässt sich mit Hilfe des mitgelieferten Simulationstools der Schaltungsentwurf simulieren bevor er auf den FPGA konfiguriert wird. Nach Abschluss des Übersetzungsprozesses erhält der Nutzer eine Übersicht über die Auslastung der Ressourcen. Ein Auszug daraus ist in der Abbildung 28 auf der Seite 41, des im Rahmen der Masterarbeit erstellten Treibers, dargestellt.

Im Zusammenhang mit dem NIOS-II Mikrocontroller, welcher im Kapitel 2.8 beschrieben wurde, benötigt der Anwender neben Quartus II eine weitere Toolchain für die Realisierung eines Projektes. Für das Erstellen von Programmen für den Softcore Prozessor bietet Altera hier die Software Eclipse. Damit lassen sich Programme in der Programmiersprache C erstellen und auf den Mikrocontroller übertragen. Zusätzlich zu den mitgelieferten Softwarepaketen befinden sich auch eine Reihe von Tutorials und Beispielprojekten mit in dem Lieferumfang des DE2-115 Boards. Auf eines dieser Projekte, den SD Karten Treiber, wird im Rahmen dieser Masterarbeit noch näher eingegangen.

3 Aufbau des VHDL Treibers zur Ansteuerung der WS2812B

In diesem Kapitel wird der im Rahmen der Masterarbeit entwickelte VHDL Code für die Ansteuerung der WS2812B detailliert beschrieben. Dabei wird anfangs erläutert, wie überhaupt Farbinformationen zu einer WS2812B übertragen werden müssen und wie die LED dann tatsächlich zum Leuchten gebracht wird. Danach wird dargestellt, welche Schritte notwendig sind, um statt einer LED einen ganzen Strip von 500 LEDs über einen Datenbus zu steuern.

Darauf aufbauend wird erläutert, wie sich ganze Bilder auf der LED Wand aus 36 Strips mit je 500 einzelnen LEDs pro Strip darstellen lassen. Die Bilder werden hierbei von einer SD Karte eingelesen. Dieser Prozess und die Schnittstelle zwischen dem SD Karten Treiber und dem LED Strip Treiber werden in einem gesonderten Kapitel abgehandelt. Abschließend werden die nötigen Arbeitsschritte beschrieben, um die in dieser Masterarbeit entwickelte Ansteuerungslogik in Betrieb zu nehmen und damit Inhalte auf der LED Wand darstellen zu können.

3.1 Die Übertragung der Farbinformationen vom FPGA zur WS2812B

Im Kapitel 2.3.3 wurde bereits der Aufbau des 24 Bit Datentelegramms mit den Farbinformationen für eine einzelne LED beschrieben. Nun folgt eine detaillierte Beschreibung der Bit für Bit Datenübertragung vom FPGA zur WS2812B.

Um ein Bit an den internen Controller der LED zu übertragen, muss eine definierte Pegelfolge eingehalten werden. Dabei erfolgt die Informationsübertragung von einem Bit über die Änderung der Spannung am Datenbus. Es wird zwischen zwei verschiedenen Pegeln unterschieden. Zum einen der Highpegel (es liegen 5,0 Volt Spannung am Datenbus an) und zum anderen der low Pegel (es liegen 0 Volt Spannung am Datenbus an). Der Verlauf des Pegels für die Kodierung der Information von einem Bit für eine Eins oder eine Null ist in der Abbildung 19 dargestellt. Des Weiteren sind die dazugehörigen Zeiten der einzelnen Phasen (T0H, T0L, T1H, T1L) in der Tabelle 4 zu sehen.

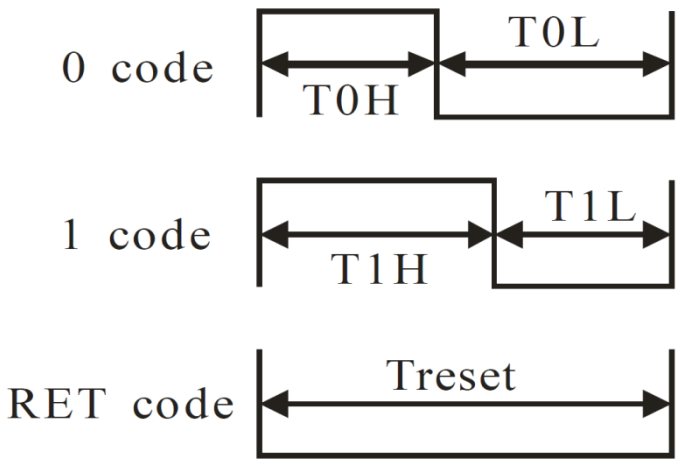


Abbildung 19: Vereinfachtes Zeitdiagramm der Übertragungslogik, [WS17]

Data Transfer Time (TH + TL = 1,25 µs ± 0,6 µs)		
T0H	0 code, high voltage time	0,40 µs + 0,15 µs
T1H	1 code, high voltage time	0,80 µs + 0,15 µs
T0L	0 code, low voltage time	0,85 µs + 0,15 µs
T1L	1 code, low voltage time	0,45 µs + 0,15 µs
RES	low voltage time	> 50 µs

Tabelle 4: Timing für Bitübertragung [WS17]

Der in VHDL entwickelte Programmablauf für die Übertragung eines Bits, dessen Wert in der Variable "Temp_Wert_Strip_1" gespeichert wird, ist in der Abbildung 20 zu sehen. Dies ist eine vereinfachte Darstellung vom Programmablauf des VHDL Codes aus der Tabelle 5, welcher im Rahmen der Masterarbeit für die Ansteuerung der WS2812B entwickelt wurde.

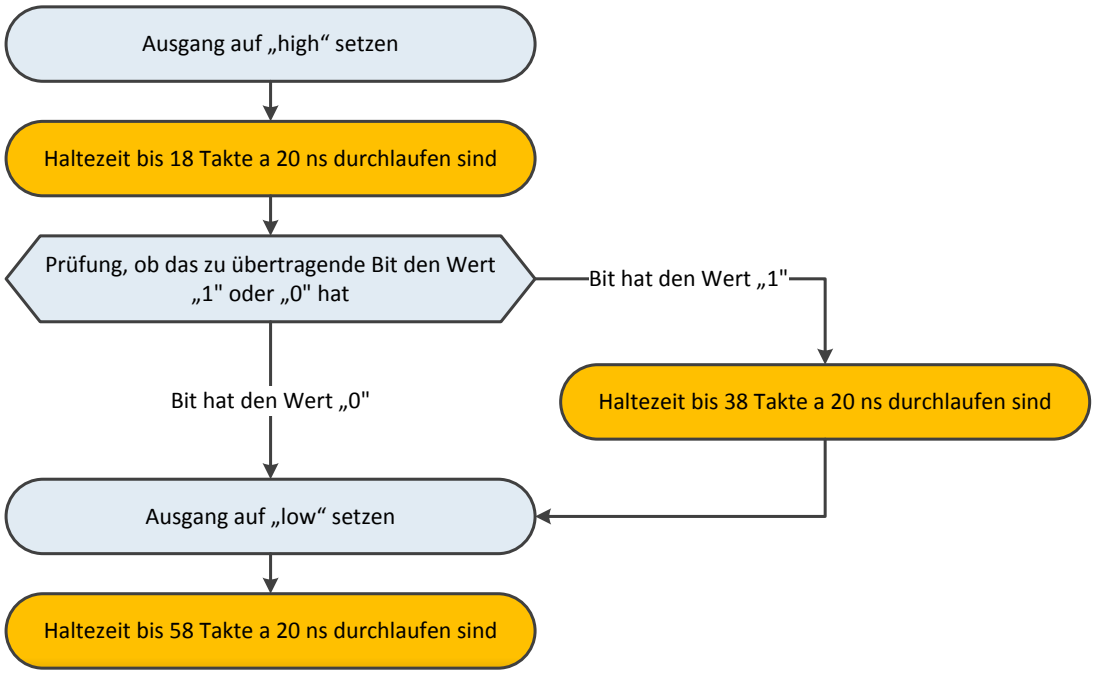


Abbildung 20: Programmablauf zur Übertragung eines Bits [HK17]

```
process(clk)   --Treiber zur Übertragung eines Bits
begin
if(rising_edge(clk)) then
        --Ausgang auf high setzen
        if(Taktzaehler_1_Strip_1 = "000000") then
        countEnable_Strip_1 <= '1';
        end if;

        --nach 18 Takten a 20ns prüfen ob eine 0 übertragen wird
        if((Temp_Wert_Strip_1 ='0')AND(Taktzaehler_1_Strip_1 = "010010")) then
        countEnable_Strip_1 <= '0';
        end if;

        --Ausgang nach 38 Takten a 20ns auf 0 setzen bei einer 1 Übertragung
        if(Taktzaehler_1_Strip_1 = "100110") then
        countEnable_Strip_1 <= '0';
        end if;

        --nach insgesamt 58 Takten a 20ns alles zurücksetzen
        if(Taktzaehler_1_Strip_1 = "111010") then
        Taktzaehler_1_Strip_1 <= (others => '0');
        end if;
end if;
end process;
```

Tabelle 5: VHDL Code für die Übertragung eines Bits [HK17]

In den beiden Ausdrucken des Oszilloskops in Abbildung 21 sind die Spannungsverläufe der Bitübertragung vom Datenbus, welche im Rahmen der Entwicklung ausgelesen worden, dargestellt. Dabei lassen sich die laut Protokoll geforderten Pegelverläufe der WS2812B gut erkennen. Bei dem linken Bild handelt es sich um die Übertragung eines Bits mit dem Wert Null und bei dem rechten Bild um die Übertragung eines Bits mit dem Wert Eins.

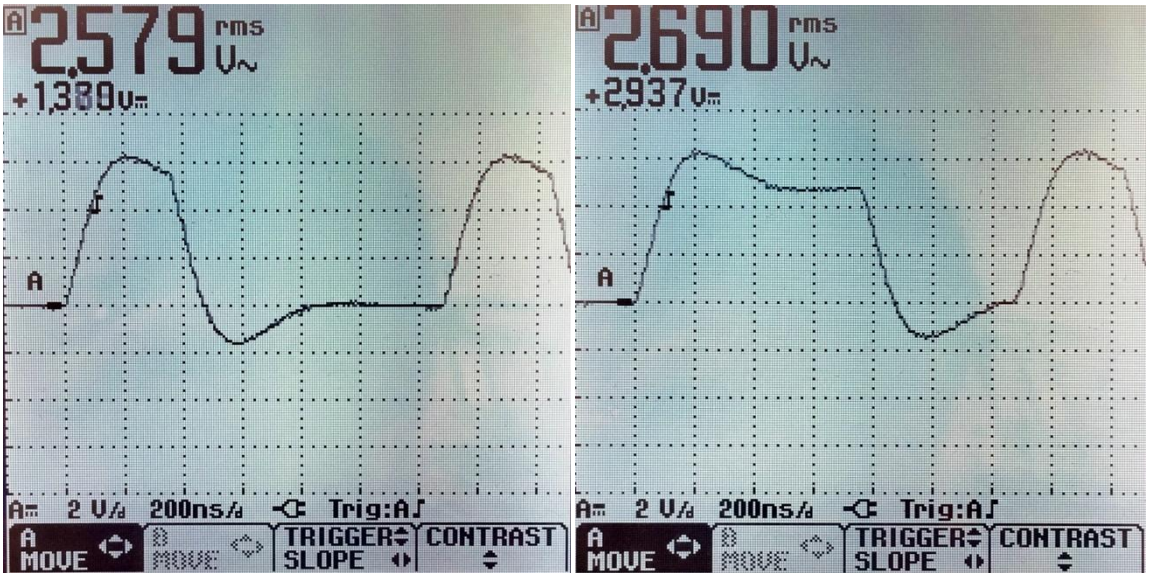


Abbildung 21: Spannungsverläufe am Datenbus bei der Bitübertragung [HK17]

Da der Cyclone IV FPGA seine Ausgänge nur mit maximal 3,3 Volt treibt, ist anhand des maximalen Spannungspegels zu erkennen, dass die Messung nach dem Inverterbaustein gemacht wurde. Dabei hat das leichte Überschwingen der Pegel auf temporäre Werte von über 6,0 Volt beziehungsweise -1,0 Volt keine Auswirkungen auf die Übertragung. Die Ursache dafür liegt in den Impedanzen der Übertragungsstrecke vom FPGA zum LED Controller. Hierbei kommt es teilweise zu Reflexionen der einlaufenden Signalwellen, was ein gedämpftes Einschwingen (englisch: ringing) des Signals auf seinen DC Wert zur Folge hat.

Auch die Anstiegs- und Abfallzeiten scheinen für den Controller der WS2812B keinen negativen Einfluss zu haben. Des Weiteren ist zu erkennen, dass die Periodendauer bei beiden Übertragungsdiagrammen einen identischen Wert von 1.280 ns hat.

3.1.1 Das Startsignal um die WS2812B zum Leuchten zu bringen

Nach Abschluss der Übertragung von 24 Bit mit den Farbinformationen für eine LED, benötigt diese nun ein Signal, um die empfangenen Werte in ihr internes PWM Register zu übertragen. Dies erfolgt über den "RES" Befehl auf dem Datenbus. Der Pegel sowie das Timing für diesen Befehl sind ebenfalls in Abbildung 19 und Tabelle 4 auf Seite 27 dargestellt. Um die benötigten 50 µs Haltezeit zu realisieren, wird der Pegel am Datenbus für 2.500 Takte à 20 ns auf "low" gehalten. Der Programmablauf eines Testprogramms aus der Entwicklungsphase des Treibers ist in Abbildung 22 zu sehen.

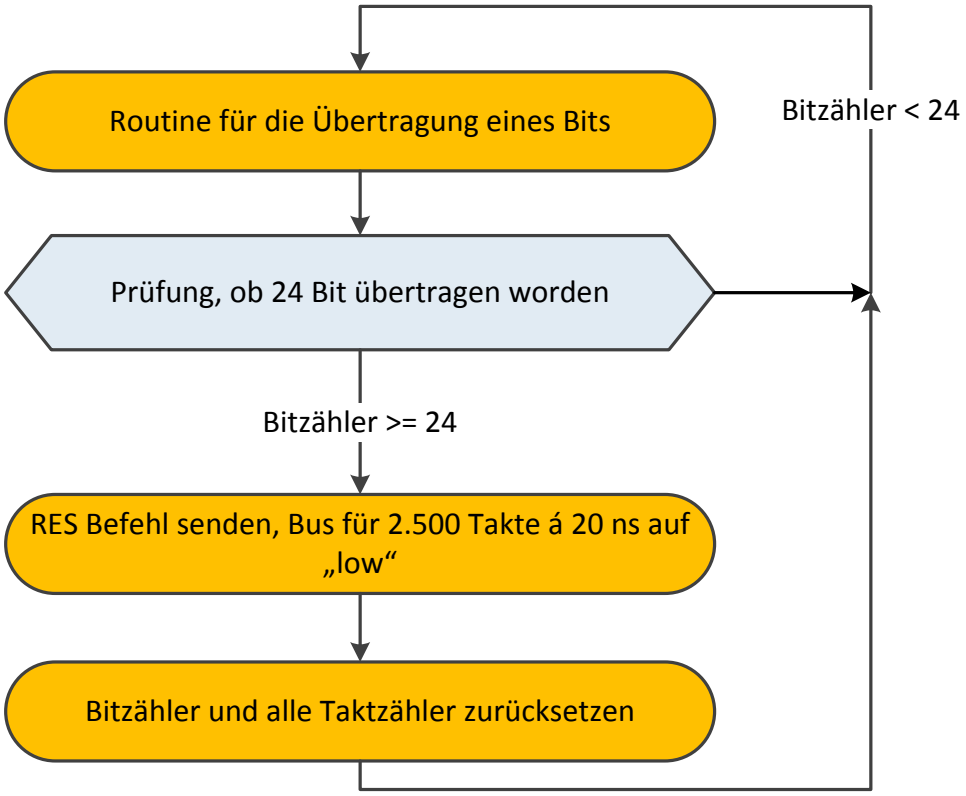


Abbildung 22: Routine für die Ansteuerung einer WS2812B [HK17]

In diesem Testprogramm wird die WS2812B immer wieder in einer Dauerschleife mit Daten beschrieben und erhält nach dem Empfang von 24 Bit den "RES" Befehl und der Datenbus wird für 50 µs auf dem low Pegel gehalten. Danach beginnen die drei internen farbigen LEDs (Rot, Grün und Blau) in der geforderten Helligkeit zu leuchten. Die Überlagerung der drei Farben ergibt laut dem RGB Farbsystem den gewünschten Farbton, in welchem die WS2812B dann leuchtet.

In der folgenden Tabelle 6 ist der VHDL Code für die Generierung des "RES" Befehls dargestellt. Dieser wird, wie schon beschrieben, nach der Übertragung der Farbinformationen an die LED gesendet. Das benötigte Zeitfenster für den geforderten Pegel wird dabei über einen Taktzähler erzeugt, welcher 2.500 Takte mit einer Länge von 20 ns mitzählt. Während dieser Zeit liegt der benötigte low Pegel am Dateneingang des Strips an.

```
process(clk)   --Auszug aus dem Treiber der WS2812 für Strip_1
begin
if(rising_edge(clk)) then
  (...)     --die Datenleitung wurde vorher bereits auf low Pegel gesetzt
            --wenn das 24bit Telegramm an die LED übertragen ist, wird die 50 us Pause aktiviert
            if(unsigned(Bitzaehler) >= 24) then
                        --Taktzähler der 20ns Takte für die 50µs Pause inkrementieren
                        Taktzaehler_3_Strip_1 <= std_logic_vector(unsigned(Taktzaehler_3_Strip_1) + 1);
                        Wartezeit_Strip_1 <= '1';

                        --nach 2500 Takten a 20 ns wird zurückgesetzt, das entspricht 50µs
                        if(Taktzaehler_3_Strip_1>="100111000100")then
                       --Zähler der 24bit Telegrammdurchläufe, Taktzaehler und Sperrvar. werden zurückgesetzt
                        Wartezeit_Strip_1 <= '0';
                        Taktzaehler_3_Strip_1 <= (others => '0');
                         Bitzaehler                 <= (others => '0');
                        end if;
end if;    end if;
end process;
```

Tabelle 6: Auszug aus dem VHDL Code vom "RES" Befehl [HK17]

3.2 Die Ansteuerung eines Strips mit 500 LEDs

Im Kapitel 2.3.4 wurde bereits beschrieben, in welcher Form die WS2812B physisch verschaltet werden müssen, um als Strip über einen einzelnen Datenbus angesteuert zu werden. Es wird nun detailliert erläutert, welche Arbeitsschritte seitens des FPGAs nötig sind, um den kompletten Strip zu steuern.

In der Abbildung 23 ist der Programmablauf für die Übertragung der Informationen an 500 Pixel auf einen Strip dargestellt. Dabei werden die Bilddaten testweise direkt als Variable mit in den VHDL Quellcode eingefügt. Ein Auszug der Zuweisung von fünf Festwerten an die Variable „Speicher_fuer_Bildpunkte(1) - (5)“ ist in der folgenden Tabelle 7 zu sehen.

```
type Bildpunkte_Typ is array  ( integer range 1 to ( Bildpunkte_horizontal *
Bildpunkte_vertikal ) ) of std_logic_vector ( 23 downto 0 ) ;

signal Speicher_fuer_Bildpunkte : Bildpunkte_Typ;


      Speicher_fuer_Bildpunkte(1)       <= "111100000000000011110000";
        --vom Strip 1 die LED 1 bekommt den Wert einer Farbe zugewiesen

      Speicher_fuer_Bildpunkte(2)       <= "111100001111000000000000";
        --vom Strip 1 die LED 2 bekommt den Wert einer Farbe zugewiesen

      Speicher_fuer_Bildpunkte(3)       <= "111100000000000011110000";
        --vom Strip 1 die LED 3 bekommt den Wert einer Farbe zugewiesen

      Speicher_fuer_Bildpunkte(4)       <= "111100001111000000000000";
        --vom Strip 1 die LED 4 bekommt den Wert einer Farbe zugewiesen

      Speicher_fuer_Bildpunkte(5)       <= "111100000000000011110000";
        --vom Strip 1 die LED 5 bekommt den Wert einer Farbe zugewiesen
     (...)
```

Tabelle 7: Feste Zuweisung der Bilddaten im VHDL Quellcode [HK17]

Dabei ist zu erkennen, dass die Farbwerte der LEDs als Signal vom Typ "std_logic_vector" mit einer Breite von 24 Bit gespeichert werden. Die einzelnen Werte codieren einen RGB Farbwert nach der Protokolldefinition der WS2812B. Dies wurde bereits im Kapitel 2.3.3 erläutert. In dem Testprogramm wird der Strip nun permanent mit Daten beschrieben. Wenn alle 500 Datenworte übertragen sind, bekommt der Strip den "RES" Befehl und die LEDs leuchten. Danach beginnt die Routine wieder von vorn. Der Programmablauf dafür ist auf der nächsten Seite in der Abbildung 23 dargestellt.

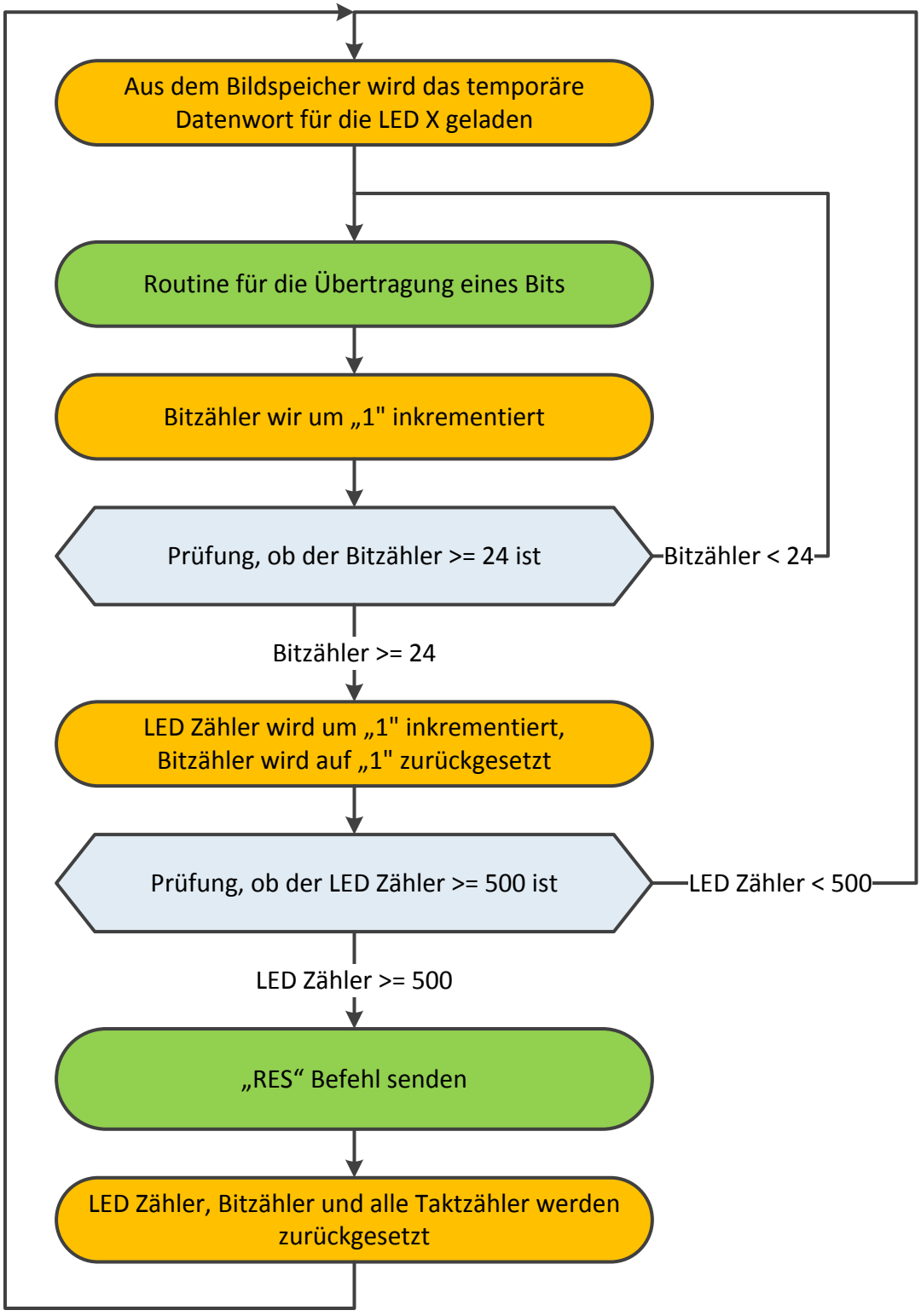


Abbildung 23: Programmablauf für die Ansteuerung eines Strips [HK17]

Der VHDL Quellcode für dieses Ablaufdiagramm ist auf der folgenden Seite in der Tabelle 8 zu sehen.

```
process(clk)   --Treiber WS2812B_Strip_1
begin
if(rising_edge(clk)) then
        if((unsigned(Taktzaehler_2_Strip_1) < LEDs_pro_Strip)AND(Wartezeit_Strip_1='0')) then

                if(Taktzaehler_1_Strip_1 = "000000") then
                countEnable_Strip_1 <= '1';          --Ausgang auf high setzen
                end if;

                if((Temp_Wert_Strip_1 ='0')AND(Taktzaehler_1_Strip_1 = "010010")) then
        --nach 18 Takten a 20ns prüfen ob eine 0 übertragen wird
                countEnable_Strip_1 <= '0';
                end if;

                if(Taktzaehler_1_Strip_1 = "100110") then
        --Ausgang nach 38 Takten a 20ns auf 0 setzen bei einer 1 Übertragung
                countEnable_Strip_1 <= '0';
                end if;

                if(Taktzaehler_1_Strip_1 = "111010") then
        --nach insgesamt 58 Takten a 20ns alles zurücksetzen
                Taktzaehler_1_Strip_1 <= (others => '0');
                Durchlaufzaehler_Strip_1 := Durchlaufzaehler_Strip_1+1;
                end if;

                if(Durchlaufzaehler_Strip_1 =24) then
                Durchlaufzaehler_Strip_1 := 0;                          --Durchlaufzähler zurücksetzen
                Taktzaehler_2_Strip_1 <= std_logic_vector(unsigned(Taktzaehler_2_Strip_1) + 1);
                LED_Zaehler_Strip_1 := LED_Zaehler_Strip_1 + 1          --LEDs werden mitgezählt
                end if;

        Taktzaehler_1_Strip_1 <= std_logic_vector(unsigned(Taktzaehler_1_Strip_1) + 1);
        --Taktzähler für 20ns Takte inkrementieren
        end if;

        if(unsigned(Taktzaehler_2_Strip_1) >= LEDs_pro_Strip) then
        --wenn die 24bit Telegramme an alle 500 LEDs übertragen sind wird die 50us Pause aktiviert

        Taktzaehler_3_Strip_1 <= std_logic_vector(unsigned(Taktzaehler_3_Strip_1) + 1);
        --Taktzähler der 20ns Takte für die 50us Pause inkrementieren
        Wartezeit_Strip_1 <= '1';

                if(Taktzaehler_3_Strip_1>="100111000100")then
                --nach 2500 Takten a 20ns wird zurückgesetzt

                --Zähler der 24bit Telegrammdurchläufe wird zurückgesetzt
                LED_Zaehler_Strip_1 := 1;
                Wartezeit_Strip_1 <= '0';
                Taktzaehler_3_Strip_1 <= (others => '0');
end if;     end if;    end if;
end process;
```

Tabelle 8: Auszug aus dem VDHL Code für die Ansteuerung eines Strips [HK17]

Mit diesem Treiber kann ein Strip gesteuert werden. Der Treiber wurde so zum Test für alle 36 Strips auf den FPGA konfiguriert. Dabei wurde, wie auf der Seite 32 beschrieben, ein komplettes Testbild als Variable mit in den Quellcode eingefügt. Wobei jeder einzelne Striptreiber aus dieser Variable, den für ihn zugewiesenen Bereich ausgelesen und an die angeschlossenen LEDs weitergegeben hat. Dies wurde auch praktisch an der LED Wand getestet und funktionierte einwandfrei.

Dieses Konzept war ursprünglich ein Lösungsansatz für die Entwicklung der Ansteuerungslogik. Das heißt, es sollte ein Bildspeicher in Form einer Variable für die Farbinformationen für alle 18.000 WS2812B der LED Wand auf dem FPGA konfiguriert werden. Dieser würde eine Größe von 18.000 mal 24 Bit haben. Aus diesem Speicher sollten nun alle 36 Striptreiber zyklisch die Daten ihres Bereiches auslesen. Zusätzlich dazu hätte der Bildspeicher über eine Schnittstelle zyklisch neu beschrieben werden können. Hierbei hätte als Datenquelle beispielsweise der Ausgang des ADV7180 Videodecoders dienen können. Bei der Umsetzung dieses Konzeptes hat sich jedoch herausgestellt, dass der Cyclone IV FPGA (EP4CE115F29C7N) für einen Bildspeicher dieser Größe über zu wenige Logikelemente verfügt. Die Synthetisierung des Projektes lief bis zum Punkt "Fitter (place and route)" durch und brach dann ab.

Somit musste ein anderes Konzept entwickelt werden, um Logikelemente bei der Schaltung einzusparen. Um zunächst erst einmal die Darstellung unterschiedlicher Bilder zu ermöglichen, wurden folgende Änderungen vorgenommen. Statt 36 einzelner Striptreiber, welche jeweils einen eigenen Strip treiben, wurde das Programm auf einen einzelnen Striptreiber reduziert. Dieser schickt nun die Bilddaten Strip für Strip an die LED Wand. Das heißt, nachdem ein Strip mit 500 LEDs beschrieben wurde, wird der Treiberausgang zum nächsten Strip weitergeschaltet. Der Treiber greift dabei auf einen wesentlich kleineren Bildspeicher von 500 mal 24 Bit zu, wo jeweils die Informationen für einen Strip gespeichert werden. Dieser Bildspeicher bekommt zyklisch neue Bilddaten für den Strip, der gerade beschrieben werden soll. Somit baut sich das Bild an der LED Wand Strip für Strip innerhalb von Sekunden auf. Als Datenquelle wurde hier eine SD Karte gewählt. Um diese auslesen zu können wird ein extra Treiber benötigt. Auch für die Übergabe der gelesenen Daten an den Bildspeicher muss eine extra Schnittstelle entwickelt werden. Diese Arbeiten werden in dem Kapitel 3.3 näher beschrieben.

3.2.1 Die maximale Bildwiederholfrequenz eines Strips

In der Abbildung 24 ist zu erkennen, wie lange die Übertragung von insgesamt 18 Datenpaketen mit jeweils 24 Bit pro Paket dauert. Dabei handelt es sich um eine Messung von einem Test während der Entwicklungsphase. Dafür wurde der Spannungsverlauf am Datenbus der ersten LED eines Strips mit dem Oszilloskop ausgemessen. Es ist gut zu sehen, dass nachdem die Daten übertragen wurden, der Bus für 50 µs für den "RES" Befehl auf dem low Pegel gehalten wird.

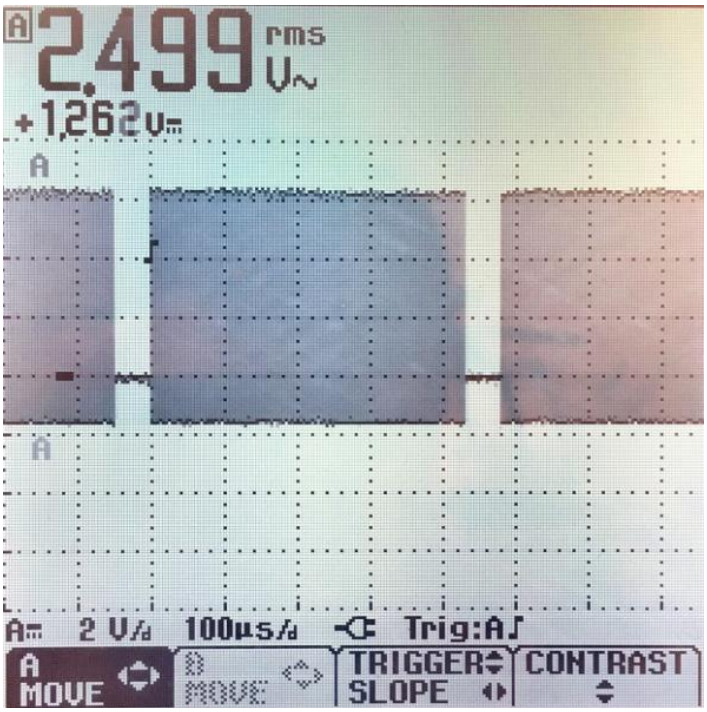


Abbildung 24: Ausdruck von der Übertragung der Datenpakete für 18 LEDs [HK17]

Aus dieser Zeitanalyse lässt sich ableiten, mit welcher maximalen Bildwiederholfrequenz ein Strip mit 500 Elementen maximal betrieben werden kann. Die Funktion zur Übertragung eines Bits, welche auf der Seite 28 beschrieben wurde, läuft in einem 20 ns Raster. Die vorgegebenen Haltezeiten für die einzelnen Phasen der Übertragung eines Bits aus der Abbildung 19, sind bei der Entwicklung des VHDL Codes, wie in der folgenden Tabelle 9 umgesetzt worden.

TOH	18 Takte à 20 ns	gesamt=theoretisch 1,16 µs, praktisch (gemessen) **1,28 µs**
TOL	40 Takte à 20 ns	
T1H	38 Takte à 20 ns	gesamt=theoretisch 1,16 µs, praktisch (gemessen) **1,28 µs**
T1L	20 Takte à 20 ns	

Tabelle 9: Taktzeiten für Pegel zur Bitübertragung [HK17]

Daraus folgt, dass die Übertragung eines 24 Bit Datenpakets, wie in der Gleichung 2.1 dargestellt, insgesamt 30,72 µs dauert.

$$24 \times 1{,}28\,\mu s = 30{,}72\,\mu s$$ **Gleichung 2.1**

Für 500 der 24 Bit Datenpakete ergibt sich eine Übertragungsdauer von 15,36 ms. Dies ist in der folgenden Gleichung 2.2 dargestellt.

$$500 \times 30{,}72\,\mu s = 15.360\,\mu s = 15{,}36\,ms$$ **Gleichung 2.2**

Werden nun noch die 50 µs Haltezeit für den "RES" Befehl addiert, ergibt sich daraus, wie in Gleichung 2.3 dargestellt, eine Zykluszeit von 15,41 ms.

$$15.360\ \mu s + 50\,\mu s = 15.410\,\mu s = 15{,}41\,ms$$ **Gleichung 2.3**

Daraus lässt sich errechnen, dass ein Strip mit 500 WS2812B theoretisch mit einer Bildwiederholfrequenz von 64,9 fps betrieben werden kann. Dies ist in der Gleichung 2.4 dargestellt.

$$\frac{1}{0{,}01541s} = \underline{\underline{64{,}9\,fps}}$$ **Gleichung 2.4**

3.3 Die Darstellung von gespeicherten Bildern

Um auf der LED Wand eine Mehrzahl von selbst gewählten Bildern darstellen zu können, muss es neben dem festen Speichern der Bilddaten im Quellcode eine weitere Möglichkeit geben, die Daten einzuspielen. Hierfür wird im Rahmen dieser Masterarbeit, das durch Altera mit dem DE2-115 Board mitgelieferte Verilog SD-Card-Reader Modul genutzt. Dieses befindet sich als komplettes Projekt auf der von dem Hersteller mitgelieferten CD im Ordner „DE2_115_demonstrations/ DE2_115_SD_CARD“. Dabei handelt es sich um einen in Verilog programmierten NIOS-II Controller, welcher in Verbindung mit dem zum Projekt zugehörigem Programm auf einer SD Karte gespeicherte Daten einließt.

Standardmäßig wird dabei der Inhalt einer Textdatei, welche unter einem festgelegten Namen auf der SD Karte gespeichert ist, eingelesen. Im Rahmen der Masterarbeit wurde dieses Template-Projekt so erweitert, dass es in einer Textdatei gespeicherte Bildinformationen einließt und diese dann über eine selbst entwickelte Schnittstelle an den in VHDL programmierten Treiber zur Steuerung der WS2812B übergibt. Der schematische Aufbau des Projektes ist in der Abbildung 25 abgebildet.

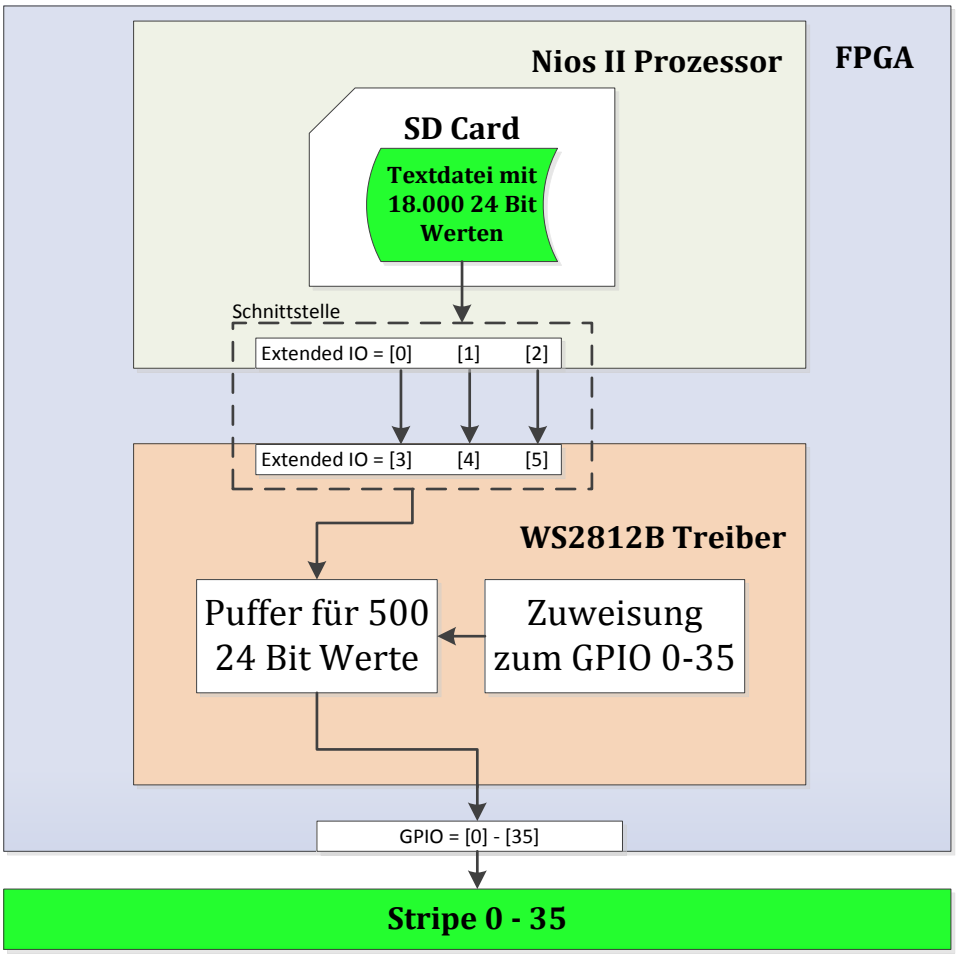


Abbildung 25: Schema der internen Struktur vom FPGA [HK17]

3.3.1 Der Softcore zum Einlesen der Bilddaten von einer SD Karte

Das in C-Code geschriebene SD Karten Programm für den NIOS-II Controller lässt diesen einzelne Zeichen von einer auf der SD Karte gespeicherten Textdatei einlesen. Im folgenden Schema ist der originale Programmablauf dargestellt:

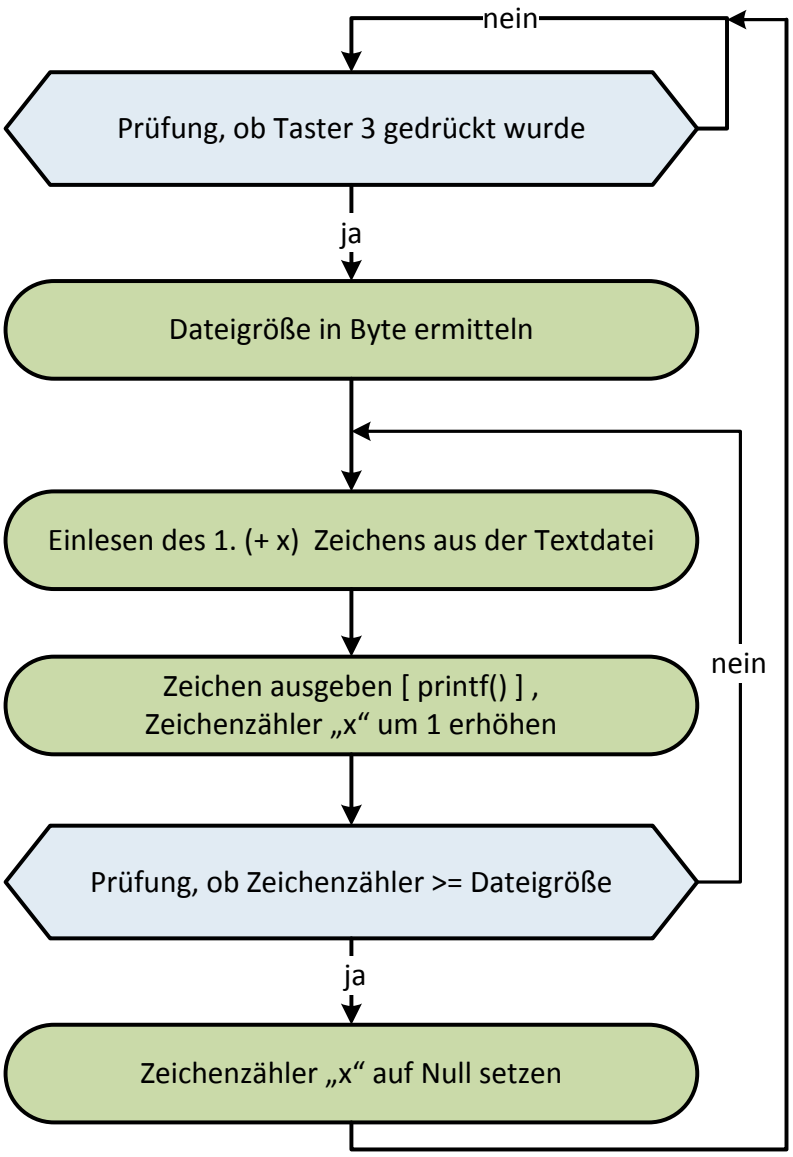


Abbildung 26: Programmablauf des original SD-Card C-Programms für den NIOS-II [HK17]

Dieses Programm wurde im Rahmen der Masterarbeit so angepasst, dass es die in einer Textdatei gespeicherten Bildinformationen ausließt und diese über eine Schnittstelle an den WS2812B Treiber übergibt. Für diese Schnittstelle wurden die „Extended IO“ Anschlüsse der DE2-115 Boards genutzt. Sie wurden zum einen als Ausgang (EX_IO[0], -[1], -[2]) des NIOS-II und zum anderem als Eingang (EX_IO[3], -[4], -[5]) des WS2812B Treibers genutzt. Diese Schnittstelle ist in der Abbildung 25 dargestellt.

Im folgenden Schema ist der geänderte Programmablauf vom SD Karten Programm dargestellt. Nach Betätigung der Taste drei werden die Daten des ersten Bildes ausgelesen und an den WS2812B Treiber übergeben. Wenn 500 Datenworte von der SD Karte gelesen worden, wird eine Pause von 50 µs eingelegt. Dieses Zeitfenster wird benötigt, um die 500, im WS2812B Treiber gepufferten, Datenworte an die dazugehörigen LEDs des Strips zu senden.

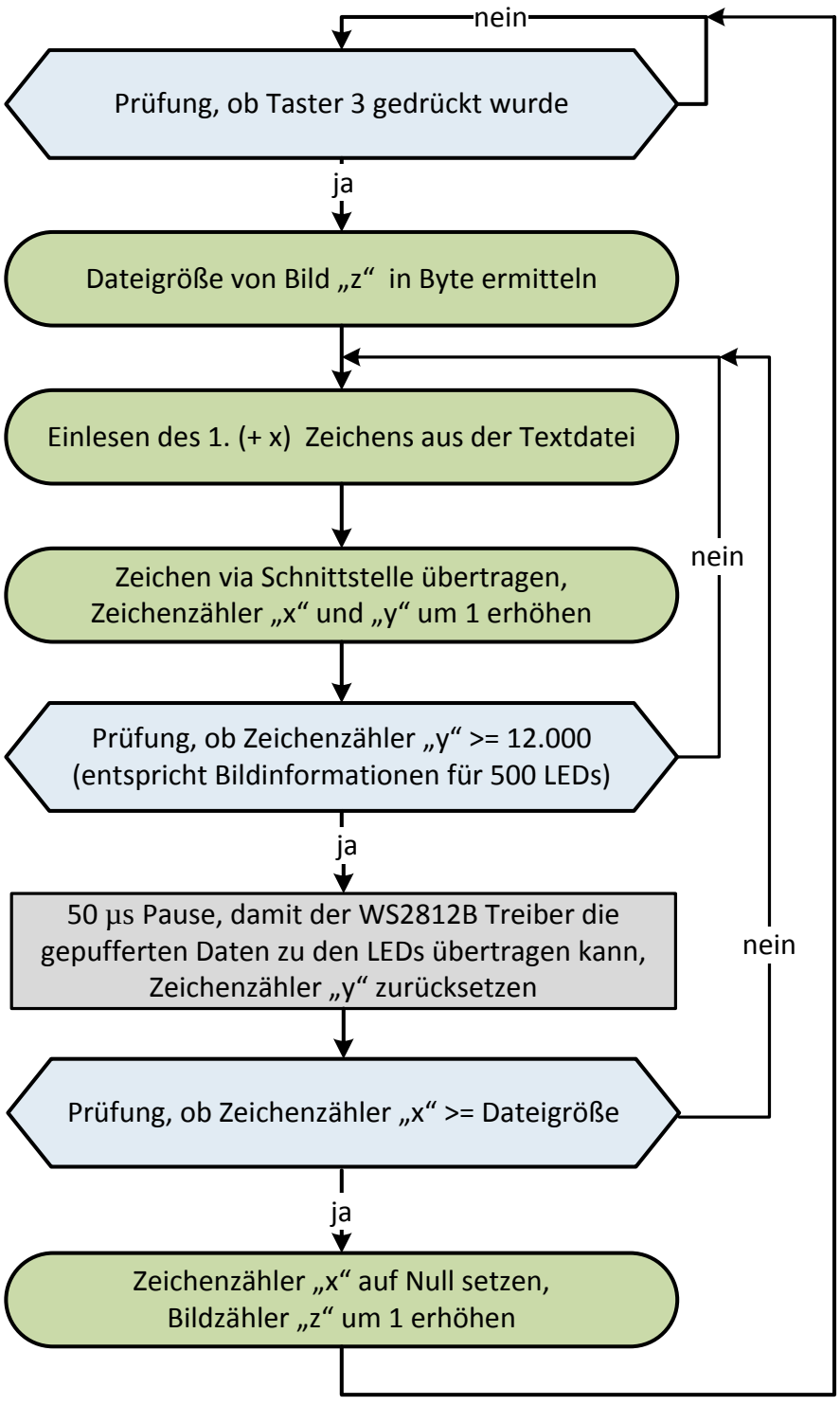


Abbildung 27: Geänderter Programmablauf des SD-Card C-Programms für den NIOS-II [HK17]

Somit baut sich das Bild Strip für Strip innerhalb von wenigen Sekunden an der LED Wand auf. Wurden alle Zeichen einer Datei übertragen, wird der Bildzähler um eins inkrementiert. Nach erneutem Drücken auf die Taste drei werden die Bildinformationen der nächsten Datei übertragen. Im Rahmen dieser Masterarbeit wurde der Treiber so geschrieben, dass die Bilddaten aus den fünf Platzhalterdateien "test_1" - "test_5" ausgelesen werden können. Sind die Informationen aus der letzten Datei ausgelesen, startet das Programm wieder mit der Datei "test_1". Dies kann im Rahmen einer Erweiterung beliebig angepasst werden. Das angepasste C-Projekt ist komplett auf der beigefügten CD im Anhang mit aufgeführt.

In der folgenden Abbildung 28 ist das Übersetzungsprotokoll des VHDL Gesamtprojektes für die Bilddarstellung auf der LED Wand dargestellt. Es ist zu erkennen, dass für die Synthetisierung der beiden Teilprojekte WS2812B Treiber und SD Karten Treiber 34 Prozent der Logikelemente des FPGAs benötigt werden.

	Task	Time
✓	Compile Design	00:10:28
✓	Analysis & Synthesis	00:03:10
✓	Fitter (Place & Route)	00:06:35
✓	Assembler (Generate programming files)	00:00:11
✓	TimeQuest Timing Analysis	00:00:32
	EDA Netlist Writer	
	Program Device (Open Programmer)	

Flow Summary	
Flow Status	Successful - Sat Oct 14 12:09:35 2017
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	DE2_115_SD_CARD
Top-level Entity Name	DE2_115_SD_CARD
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	39,296 / 114,480 (34 %)
Total combinational functions	37,984 / 114,480 (33 %)
Dedicated logic registers	16,369 / 114,480 (14 %)
Total registers	16419
Total pins	521 / 529 (98 %)
Total virtual pins	0
Total memory bits	2,188,160 / 3,981,312 (55 %)
Embedded Multiplier 9-bit elements	4 / 532 (< 1 %)
Total PLLs	1 / 4 (25 %)

Abbildung 28: Übersetzungsprotokoll aus Quartus II für das Gesamtprojekt [HK17]

3.3.2 Die Schnittstelle für die Übergabe der Bilddaten

Um die gelesenen Daten von der SD Karte an den WS2812B Treiber zu übergeben, bedarf es einer Schnittstelle zwischen den beiden Modulen. Dafür wurde der 14 polige Zusatzstecker JP4 vom DE2-115 Board genutzt, um eine serielle Verbindung aus drei Anschlüssen zu realisieren. Der Stecker JP4 bietet sieben zusätzliche GPIO Pins. Dabei fungiert das SD Karten Modul als Sender und das Striptreibermodul als Empfänger. Die Ausgänge zur Schnittstelle des SD Karten Moduls sind auf die Pins EX_IO[0], EX_IO[1] und EX_IO[2] geschaltet. Diese sind mit den Eingängen des Striptreibermoduls verbunden, welches mit den Pins EX_IO[3], EX_IO[4] und EX_IO[5] verschaltet ist. Die elektrische Verbindung wurde, wie in Abbildung 29 dargestellt, über drei Leitungsverbindungen (Blau, Grün und Gelb) realisiert.

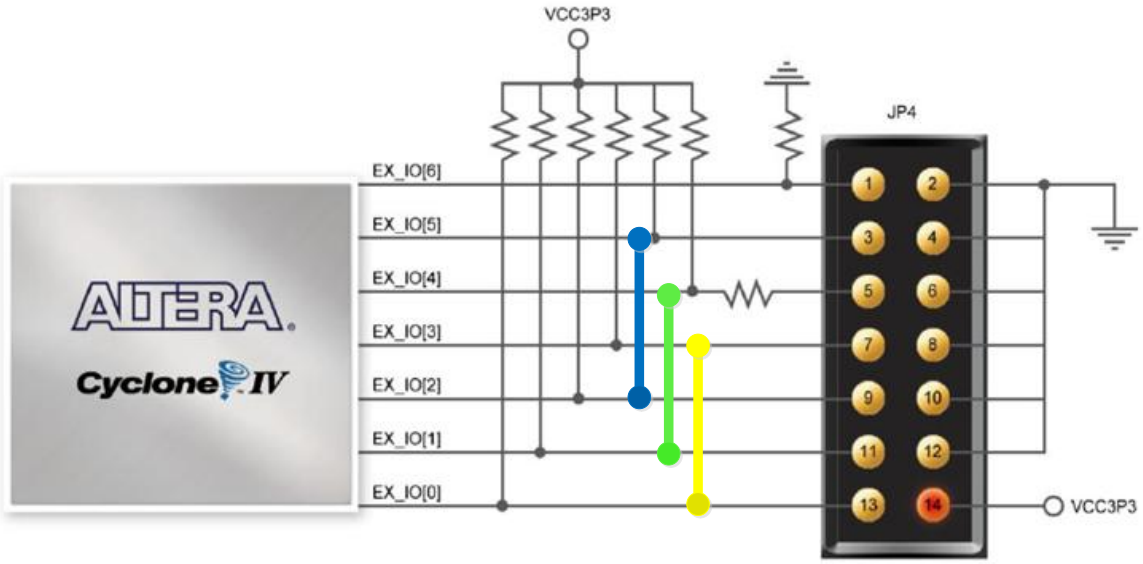


Abbildung 29: 14 poliger expansion header JP4 vom DE2-115 Board [WS17]

Die drei Kanäle der Schnittstelle (Blau, Grün und Gelb) haben folgende Funktionen. Der gelbe Kanal signalisiert den Start der Datenübertragung. Wechselt dieser seinen Pegel von low (0 Volt) auf high (3,3 Volt), beginnt die Datenübertragung. Der blaue Kanal signalisiert den Wert des zu übertragenden Bits. Bei einem low Pegel wird eine Null gesendet, bei einem high Pegel eine Eins. Der grüne Kanal fungiert als Bitzähler. Toggelt dieser seinen Pegel, signalisiert dies die Übertragung des nächsten Bits. Ein Schema des Übertragungsvorgangs ist in der folgenden Abbildung 30 dargestellt.

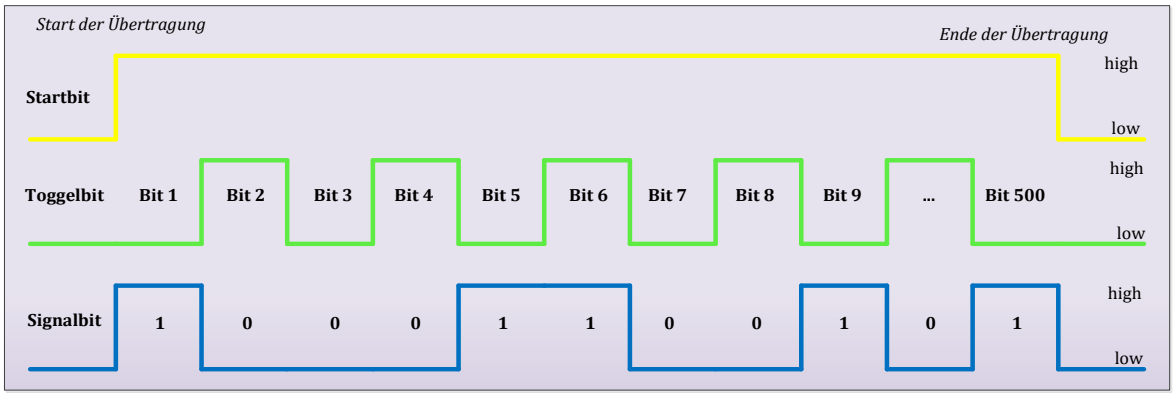


Abbildung 30: Serielle Schnittstelle für die Übertragung der Bilddaten [HK17]

Im Striptreibermodul des FPGAs werden die empfangenen Daten zwischengespeichert. Dabei fungiert der Pegel des Startbits als Eventauslöser. Soll ein neues Bild eingelesen werden, wird dieser Vorgang durch das Drücken der Taste drei (KEY3) im SD Karten Treiber gestartet. Das Startbit wird durch den SD Karten Treiber auf den "high" Pegel gesetzt und signalisiert dem Striptreiber den Beginn der Datenübertragung. Es werden jeweils 12.000 Bits, was 500 Datenworte mit 24 Bit entspricht, von der SD Karte gelesen und zum Striptreiber übertragen.

Der Striptreiber speichert nach jedem Toggelvorgang des Toggelbits den Wert, welcher am Signalbit anliegt, in den internen 12.000 Bit großen Bildspeicher. Zusätzlich werden die Bits auf beiden Seiten der Schnittstelle mitgezählt. Wurden insgesamt 12.000 Bits übertragen, geht der Pegel am Startbit auf low. Dies signalisiert den Abschluss des Übertragungsvorgangs der Daten für einen Strip. Nun sendet der Striptreiber die gepufferten Daten zum ersten Strip. In dieser Zeit werden für ein fest definiertes Zeitfenster keine neuen Daten von der SD Karte gelesen. Ist dieses Zeitfenster abgelaufen, werden die nächsten 12.000 Bits des Bildes von der SD Karte eingelesen, zwischengespeichert und zum nächsten Strip übertragen. Nach insgesamt 36 dieser Vorgänge wurde das Bild von der SD Karte gelesen und komplett zur LED Wand übertragen. Durch das erneute Drücken der Taste drei, kann der Vorgang für das nächste Bild gestartet werden.

3.4 Vorbereitung der Bilddaten zur Übertragung von der SD Karte an den LED-Strip Treiber

Wie bereits beschrieben, nutzt der FPGA in Verbindung mit dem SD Karten Treiber Modul den Inhalt von Textdateien als Datenquelle. Das heißt, bevor auf diese Weise Bilddaten eingelesen werden können, ist eine Aufbereitung der Quelldaten notwendig. Hierfür sind bestimmte Voraussetzungen zu erfüllen. Zum einen muss es sich bei der Bilddatei, welche später angezeigt werden soll, um ein Bild im 24 Bit Bitmap (BMP) Format handeln. Zum anderen sollte dessen Auflösung 180 mal 100 Bildpunkte betragen. Die Auflösung beliebiger BMP Bilder lässt sich relativ einfach mit dem Windows Standard Programm Paint anpassen. Einige Beispielbilder hierfür befinden sich auf der beigefügten CD im Ordner "Beispielbilder". Liegt die Quelldatei im richtigen Format sowie in der richtigen Auflösung vor, können im nächsten Schritt die Bildinformationen aus der Datei extrahiert werden.

Die kompletten Umwandlungs- und Anpassungsvorgänge wurden im Rahmen der Masterarbeit mittels Software, wie beispielsweise Matlab, sowie mit selbst programmierten Umwandlungsfunktionen realisiert. Im Kapitel 4 "Zusammenfassung und Ausblick" wird noch näher darauf eingegangen was hier an Programmierarbeit notwendig ist, um die Anwendung des Gesamtprojektes nutzerfreundlicher zu gestalten. Um an die Farbwerte für jeden der 18.000 einzelnen Bildpunkte zu gelangen, wird die BMP-Datei mit der Auflösung von 180 mal 100 Bildpunkten als Variable im Matlab angelegt. Dies geschieht, indem sie per "drag and drop" in das geöffnete Matlabfenster gezogen wird. Danach können mit Hilfe der Software die drei Farbtabellen für die einzelnen Farben Rot, Grün und Blau angezeigt werden.

In den Farbtabellen sind die Farbwerte als dezimale Größen dargestellt. Das Format für die Farbwerte, welches die WS2812B voraussetzt, wurde bereits im Kapitel 2.3.3 ausführlich beschrieben. Bleibt nun die Umformung der dezimalen Farbtabellen hin zu den geforderten Binärwerten in der richtigen Reihenfolge. Hierfür wurde im Rahmen der Masterarbeit eine Excel-Datei programmiert. Diese formt die dezimalen Farbwerte in binäre Werte um und ordnet sie in der benötigten Reihenfolge an. Sie ist, neben fünf fertig aufbereiteten Beispielbildern, ebenfalls auf der beigefügten CD mit hinterlegt. Aus der Excel-Datei können die Bildinformationen herauskopiert und in eine Textdatei eingefügt werden. Diese Textdatei wird abschließend auf einer SD Karte unter einem vorgegebenen Namen gespeichert. Eine detaillierte Schritt für Schritt Anleitung, wie hierbei vorzugehen ist, befindet sich im nächsten Kapitel.

3.5 Anleitung für den Betrieb

Es wird nun Schritt für Schritt beschrieben, wie vorzugehen ist, um ein neues Bild auf der LED Wand mittels des FPGA darstellen zu können.

I. Anlegen einer Variable im Matlab mit den Bilddaten der Bitmapdatei

Das 24 Bit BMP Bild in der Auflösung von 180 mal 100 Bildpunkten muss im Matlab als Variable angelegt werden. Dies geschieht, indem es per "drag and drop" in den Workspace gezogen wird. Der Vorgang muss mit dem Drücken der Taste "Finish" bestätigt werden. Dies ist in den beiden folgenden Abbildungen 31 und 32 dargestellt.

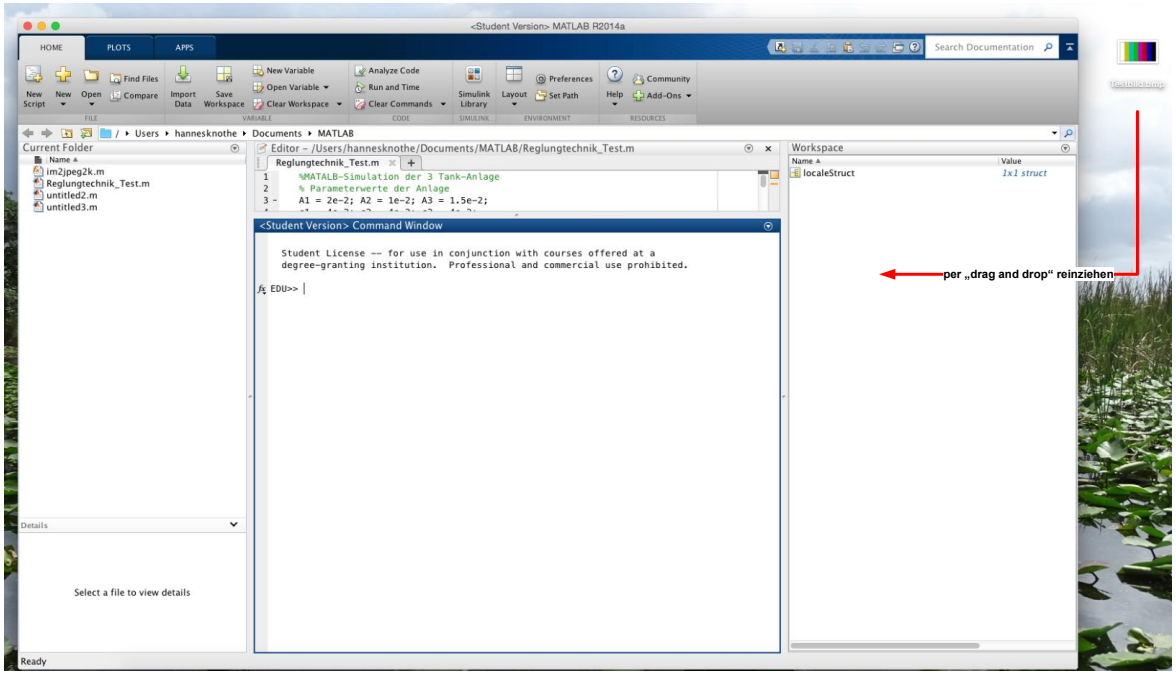


Abbildung 31: Anlegen einer Variable mit den Bilddaten im Matlab [HK17]

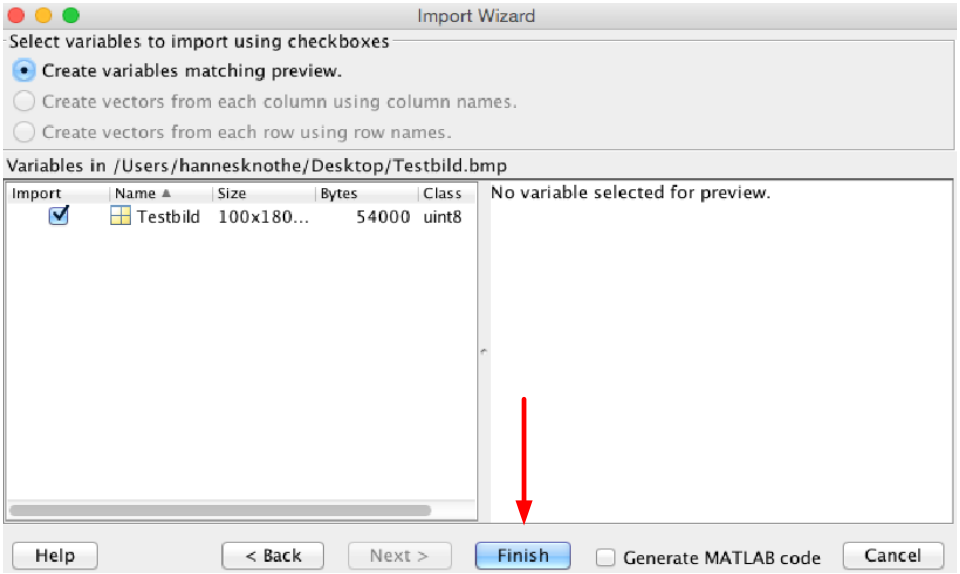


Abbildung 32: Bestätigung eines Vorgangs im Matlab [HK17]

II. Anzeigen des Inhaltes einer Variable im Matlab

Anschließend soll der Inhalt der angelegten Variable angezeigt werden. Dies wird über die Eingabe des Variablennamens im "Command Window" realisiert. Als Ergebnis dieser Eingabe werden die Farbwerte der einzelnen Bildpunkte in Form von drei Tabellen ausgegeben. Diese Werte müssen mit dem Windows Befehl "STR + A" markiert und in die Zwischenablage kopiert werden, um in eine neu anzulegende Textdatei eingefügt werden zu können. Beide Vorgänge sind in den folgenden beiden Abbildungen 33 und 34 zu sehen.

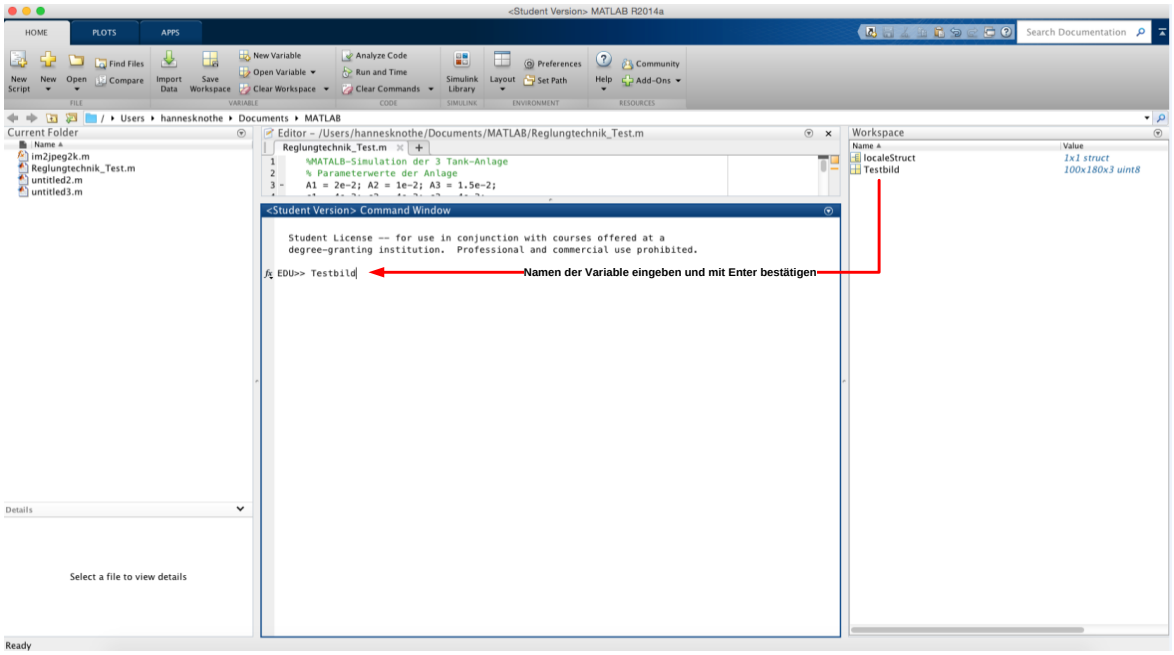


Abbildung 33: Eingabe des Variablennamens im "Command Window" [HK17]

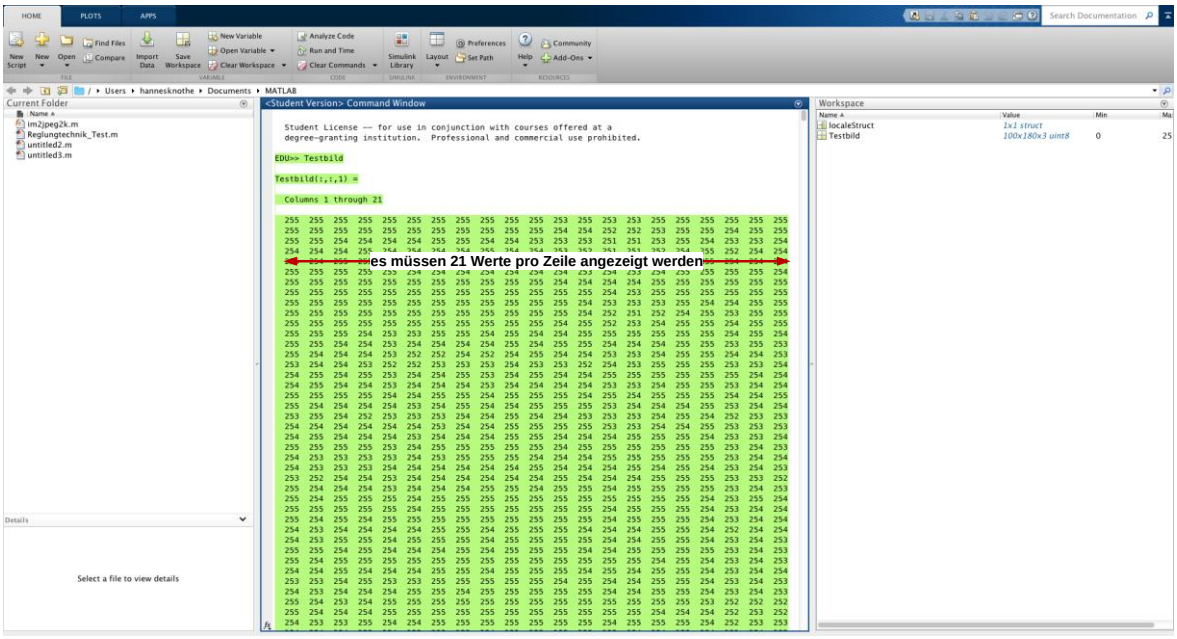


Abbildung 34: Markierte Farbwerte im Matlab [HK17]

Dabei ist es wichtig, dass das Fenster ("Command Window") in der Breite so eingestellt ist, damit insgesamt 21 Werte pro Zeile angezeigt werden. Dies ist für das folgende Umkopieren der Daten wichtig, da sie im Folgenden weiter mit der Software Excel bearbeitet werden und die Vorlage dieses Format verlangt.

III. Erstellung einer Textdatei mit dem Inhalt der Farbwerttabellen

Die Farbwerte werden nun übergangsweise, wie in der Abbildung 35 darestellt, in eine Textdatei zwischengespeichert.

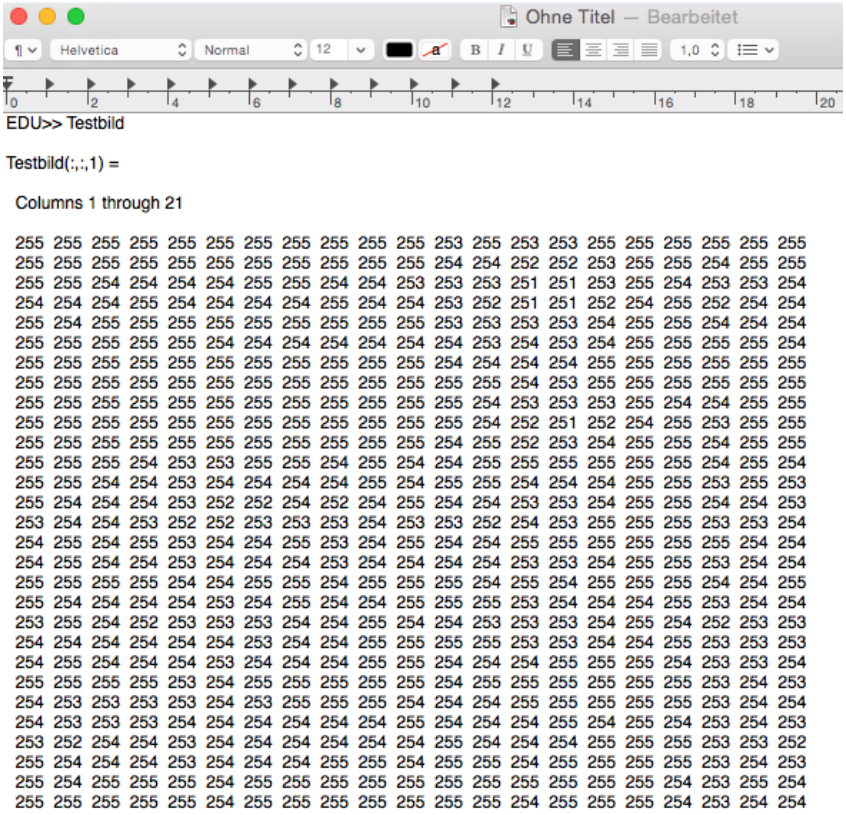


Abbildung 35: Textdatei mit dezimalen Farbwerten [HK17]

IV. Umwandlung der Farbwerte in das für die WS2812B benötigte Format

Anschließend müssen die Farbwerte in ein Format umgewandelt werden, welches der VHDL Treiber benötigt, um später die WS2812B anzusteuern. Dies geschieht mit Hilfe eines vorprogrammierten Excel-Dokuments, welches unter dem Namen "Formatumwandlung" mit auf der beigefügten CD abgespeichert ist. Dort werden die zwischengespeicherten Werte aus der Textdatei eingefügt. In dem Excel-Dokument werden dann die dezimalen Farbwerte in binäre Werte umgewandelt. Zusätzlich erfolgt die Farb- sowie Bitanordnung laut WS2812B Protokoll, wie in Kapitel 2.3.3 beschrieben.

Bevor das geschieht, müssen die Daten aus der Textdatei im richtigen Format in ein leeres Excel-Dokument eingefügt werden. Dies wird über die Excel-Funktion "Daten aus Textdatei einfügen" realisiert. Es ist notwendig, im Textkonvertierungs-Assistent feste Spaltenbreiten für die dezimalen Werte festzulegen. Dieser Vorgang ist in der folgenden Abbildung 36 zu sehen.

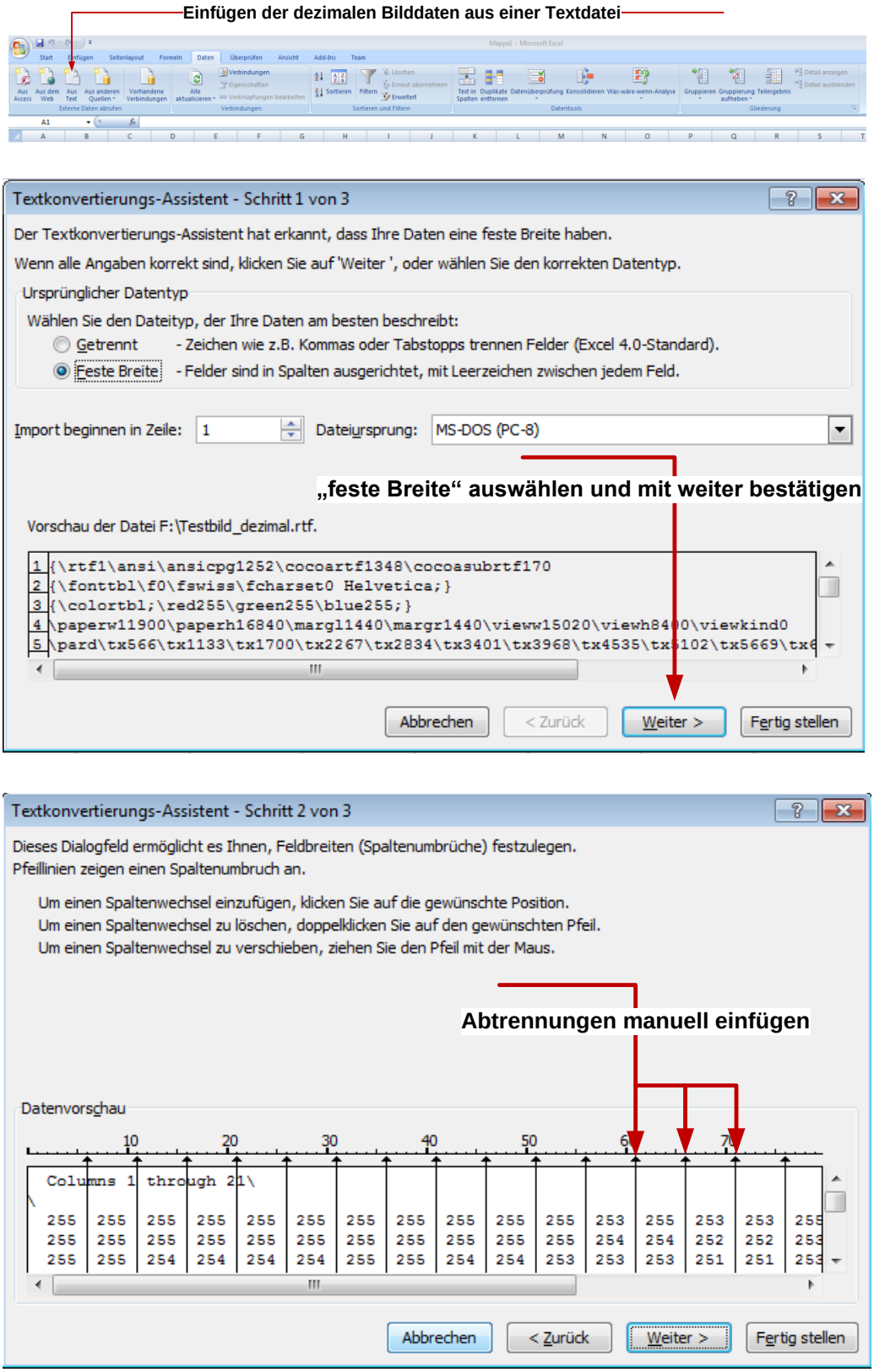


Abbildung 36: Einfügen der Daten einer Textdatei im Excel [HK17]

Nachdem die Daten nun in dem Excel-Dokument einfügt wurden, müssen noch die Querstriche (/) entfernt werden. Dies wird über die Funktion "Suchen und Ersetzen" (Tastenkombination: " STR + H") realisiert. Die Querstriche werden dabei durch Leerzeichen ersetzt.

Die so generierte Tabelle aus 21 mal 2.784 Feldern kann nun in das Excel-Dokument "Formatumwandlung" umkopiert werden. Die Daten werden, wie in Abbildung 37 zu sehen, auf der ersten Seite "Eingabe" eingefügt.

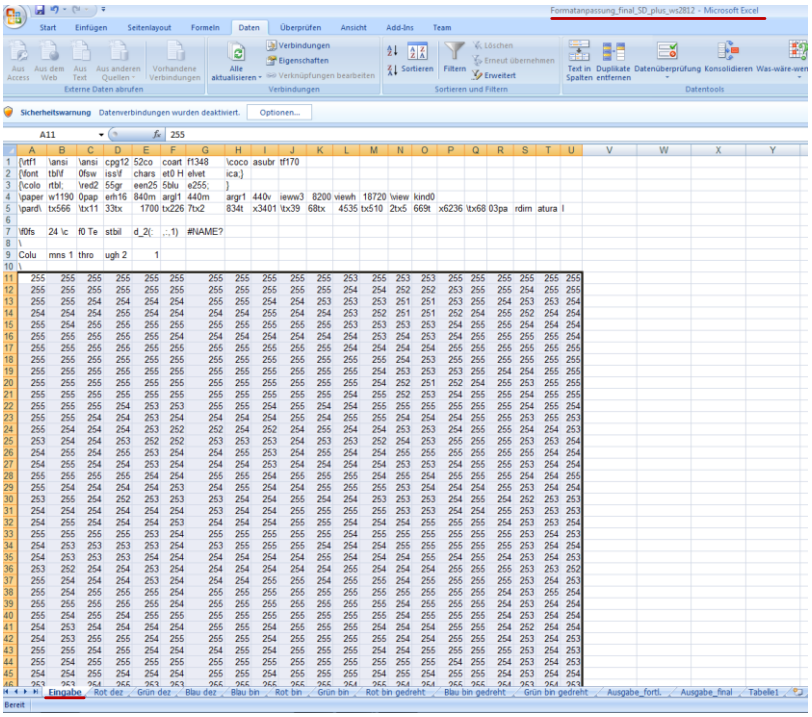


Abbildung 37: Einfügen der Daten im Konvertierungsdokument [HK17]

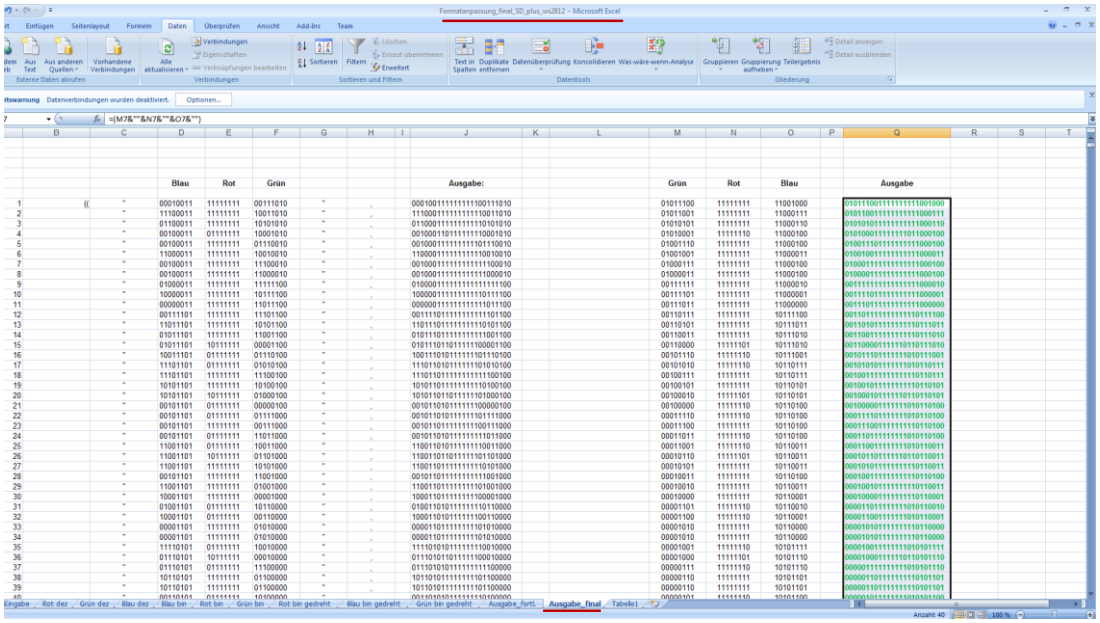


Abbildung 38: Fertige Bilddaten [HK17]

V. Einfügen der Bilddaten in eine Textdatei zur Speicherung auf der SD Karte

Abschließend müssen die Bilddaten (grüne Werte aus der Abbildung 38, insgesamt 18.000 24 Bit Werte) in eine Textdatei umkopiert werden. Dieses Dokument wird dann unter dem Namen test_1, test_2, test_3, test_4, oder test_5" auf der SD Karte gespeichert.

4 Zusammenfassung und Ausblick

Ziel dieser Masterarbeit war es, eine Ansteuerungslogik mittels FPGA für eine LED Wand zu entwickeln, wobei sich mit dieser Ansteuerung Bild- und Videomaterial auf der LED Wand darstellen lassen sollte.

Um die LEDs zum Leuchten zu bringen, wurde zu Beginn der Arbeit ein Treiber für den FPGA entwickelt, mit welchem sich Testdaten Bit für Bit in die Controller der LEDs schreiben ließen. Dieser Treiber stellt die Grundvoraussetzung für jede Art von Ansteuerung einer WS2812B dar. Die in der Masterarbeit dafür entwickelte Logik, die auch im Kapitel 3.1 ausführlich beschrieben wurde, lässt sich dabei unabhängig von der Eingangsquelle der Daten ebenso für zukünftige Weiterentwicklungen als Basis verwenden.

Im nächsten Schritt wurde dieser Treiber so angepasst und vervielfältigt, dass jeder einzelne der insgesamt 36 Strips seine eigene unabhängige Ansteuerung bekam. Weiterhin wurde zusätzlich ein Bildspeicher auf dem FPGA entwickelt, aus welchem jeder Striptreiber seine Daten beziehen sollte. Der Speicher sollte insgesamt 432 KBit, was die Daten für ein komplettes Bild darstellt, puffern können. In dieser Phase der Entwicklung ließen sich dann komplette Testbilder als Variable mit in den VHDL Quellcode integrieren. Damit konnte der FPGA dann durch Optimierung bei der Übersetzung den Speicherbedarf des Bildes so reduzieren, dass sich das Projekt nach relativ langer Rechenzeit, von bis zu über vier Stunden, synthetisieren ließ. Somit konnte mit der entwickelten Ansteuerungslogik ein einzelnes Bild auf der LED Wand dargestellt werden.

Weiterhin wurde zu diesem Zeitpunkt erkannt, dass der Cyclone IV FPGA zwar in der Lage ist Testbilder intern zu puffern, jedoch die Logikelemente nicht ausreichten, um einen extern beschreibbaren Bildspeicher zu integrieren. Obwohl es sich bei dem verwendeten FPGA Modell um den Vertreter mit den meisten Logikelementen seiner Generation handelt (siehe Kapitel 2.6), konnte im Quartus II ein beschreibbarer Speicher dieser Größe zwar synthetisiert, jedoch nicht auf dem FPGA platziert werden. Um das Projekt in dieser Form umzusetzen, wären ungefähr das Siebenfache der Logikelemente des verwendeten FPGAs nötig gewesen. Für die weitere Entwicklung gab es nun zwei Möglichkeiten. Entweder den Bildspeicher so anzupassen, dass nur Teile des Bildes gepuffert werden oder den Bildspeicher auf einen externen RAM Baustein auszulagern. Für die Darstellung von Festbildern auf der LED Wand fiel die Entscheidung hierbei auf die erste Variante. Dafür wurde der Treiber so angepasst, dass ein einziges Treibermodul jeweils alle 36 Strips sequentiell beschreibt. Die Daten dafür wurden in einem wesentlich kleineren Bildspeicher von 12 KBit gepuffert, was den Bilddaten für einen Strip mit 500 LEDs entspricht (500 mal 24 Bit = 12 KBit).

Der nächste Schritt machte es nötig, ein Modul zu entwickeln, aus dem der Bildspeicher seine Daten einliest. Da die komplette Eigenentwicklung eines Moduls, welches Daten von einer SD Karte liest, sehr aufwendig gewesen wäre, wurde hier teilweise auf ein mit dem Board mitgeliefertes Projekt zurückgegriffen. Der Hersteller Terasic liefert dem Anwender mit dem DE2-115 Board eine Reihe von Beispielprojekten mit. Unter anderem ist ein Modul, um Daten von einer SD Karte zu

lesen, mit im Lieferumfang enthalten. Dieses in Verilog programmierte Projekt wurde in der Masterarbeit so angepasst, dass es die ausgelesenen Bilddaten über eine selbst entwickelte Schnittstelle an das LED Treibermodul übergibt. Somit lassen sich in der derzeitigen Konfiguration bis zu fünf Bilder auf der LED Wand darstellen. Dabei baut sich das Bild innerhalb weniger Sekunden Strip für Strip auf der Wand auf. Mittels eines Tasters vom Board werden die einzelnen Bilder durch geschaltet. Die Anzahl der Bilder lässt sich theoretisch bis zur maximalen Speicherkapazität der SD Karte beliebig erweitern. Diese Ausbaustufe stellt den letzten Stand der Entwicklung innerhalb der Masterarbeit dar und es lässt sich festhalten, dass das Ziel der Bilddarstellung mit FPGA auf der LED Wand erreicht wurde.

Einlesen von Videodaten über den ADV7180

Neben dem Treiber zum Anzeigen von Bildern, wurde innerhalb der Masterarbeit auch versucht, einen Treiber für die Darstellung von Videosignalen auf der LED Wand zu entwickeln. Das DE2-115 Board bietet hierfür hardwareseitig eine "VIDEO IN" Schnittstelle, welche als FBAS Stecker ausgeführt und an einen ADV7180 Videodecoder Baustein angeschlossen ist. Damit lassen sich Videodaten beispielsweise im PAL Format einlesen und decodieren. Im Gegensatz zur Bilddarstellung ist das Anzeigen von Videosignalen auf der LED Wand wesentlich komplexer. Ausgehend von einem PAL Signal, welches aus zwei Halbbildern mit einer Auflösung von 720 mal 288 Bildpunkten besteht, was einem Vollbild von 720 mal 576 Bildpunkten entspricht, müssen zusätzlich zur Pufferung umfangreiche Umformungen und Anpassungen des Bildsignals in Echtzeit durchgeführt werden, um dieses auf der LED Wand darzustellen.

Bei der Recherche zu dieser Thematik wurde auf dem Forum GITHUB ein ähnliches Projekt gefunden. Unter der URL: "https://github.com/AntonZero/CCD_Cam/" ist dieses Projekt offen abgelegt. Zusätzlich ist es auf der beigefügten CD im Ordner "Videodarstellung/CCD_Cam_original" im Original enthalten.

Aufbau des Projektes CCD_Cam

Innerhalb des in VHDL programmierten CCD_Cam Projektes werden Videodaten im PAL Format über den ADV7180 Baustein eingelesen und über die VGA Schnittstelle wieder ausgegeben. Somit lässt sich das Videosignal auf einem Monitor darstellen. Intern besteht das Projekt aus einer Vielzahl von Einzelmodulen. Zum einen wird das Bildsignal von einer YCbCr Codierung in ein RGB Signal umcodiert. Zum anderen werden die Bilddaten aufgrund der unterschiedlichen Lese- und Schreibfrequenz der einzelnen Module auf einem externen SDRAM Baustein zwischengepuffert. Innerhalb des Projektes wird dann für die Videodarstellung nur ein Farbkanal ausgewertet. Das bedeutet, es wird lediglich ein schwarzweiß Videosignal ausgegeben. Auch der SDRAM Treiber ist nicht optimal programmiert, was ein leichtes Flackern des Bildes zur Folge hat.

Anpassung des CCD_Cam Moduls

Da dieses Modul ursprünglich für das DE1-SoC Board von Terasic entwickelt wurde, mussten einige Anpassungen an diesem vorgenommen werden, um auf dem DE2-115 Board lauffähig zu sein. Es erfolgte unter anderem eine Anpassung des FPGA Typs sowie eine Überarbeitung der Timergenerierung im QSys. Damit funktionierte das CCD_Cam Modul auf dem DE2-115 Board. Das angepasste Projekt ist im Ordner "Videodarstellung/CCD_Cam_angepasst_DE2-115" auf der beigefügten CD mit hinterlegt.

Nutzung des CCD_Cam Moduls für die Videodarstellung auf der LED Wand

Der Grundgedanke bei der Nutzung des CCD_Cam Projektes war es, dieses so anzupassen, dass das in RGB Form umgewandelte Videosignal genutzt wird, um es nach einer Formatanpassung parallel zur VGA Schnittstelle auf der LED Wand auszugeben. Für die Umsetzung dieser Idee wurden weitere Anpassungen am CCD_Cam Modul durchgeführt. Als erstes wurde im Umwandlungsmodul vom YCbCr Signal auf RGB Signal alle drei Farbkanäle für die Auswertung aktiviert, um später Farbbilder darstellen zu können. Leider hat der SDRAM Baustein des DE2-115 Boards lediglich eine Speicherbreite von 16 Bit. Für eine Pufferung eines kompletten RGB Farbwertes wären jedoch 24 Bit nötig. Somit müsste hier für die Pufferung der komplette SDRAM Treiber umgeschrieben werden. Da eine Entwicklung einer intelligenten RAM Steuerung im Rahmen der Masterarbeit äußert aufwendig geworden wäre, wurde die Weiterentwicklung in dieser Phase abgebrochen. Abgesehen davon müsste noch ein Modul zur Formatanpassung vom PAL Format auf die 180 mal 100 Pixel der LED Wand entwickelt werden. Ein möglicher Lösungsansatz hierfür befindet sich in einem gesonderten Kapitel in Anhang auf der Seite ix.

Eine Version des CCD_Cam Moduls mit dem letzten Entwicklungsstand befindet sich ebenfalls auf der beigefügten CD im Ordner "Videodarstellung/ CCD_Cam_final_DE2-115". Bei dieser Version werden alle drei Farbkanäle (RGB) aus dem YCbCr Signal gewonnen, allerdings werden nur der Rot- und der Grünkanal im SDRAM gepuffert. Auf der VGA Schnittstelle wird somit ein Bild aus diesen beiden Farben ausgegeben.

In der Entwicklungsphase wurden auch immer wieder Funktionstests der einzelnen Komponenten durchgeführt. Als Signalquelle diente hierfür eine einfache Digitalkamera von Samsung (Modell PL50). Es erfolgte auch schon testweise eine Einbindung, des aus dem für die Bilddarstellung entwickelten LED Strip Treibers, in das CCD_Cam Projekt. Zum Test wurden dann ohne Formatanpassung einzelne Bilddaten von selbst erstellten einfarbigen Testbildern abgegriffen und auf den Teststrips dargestellt. Die Testbilder waren dabei auf einer SD Karte innerhalb der Kamera gespeichert und konnten über diese als Ausgabesignal aktiviert werden.

Abschließend lässt sich festhalten, dass das Ziel der Videodarstellung auf der LED Wand mittel FPGA nicht erreicht wurde. Die gewonnen Erkenntnisse aus den Vorüberlegungen bilden jedoch eine Basis für die zukünftige Weiterentwicklung der Ansteuerungslogik.

Anhang

A. Auszug aus dem Datenblatt der WS2812B

Worldsemi

WS2812B

Intelligent control LED integrated light source

Features and Benefits

- Intelligent reverse connect protection, the power supply reverse connection does not damage the IC.
- The control circuit and the LED share the only power source.
- Control circuit and RGB chip are integrated in a package of 5050 components, form a complete control of pixel point.
- Built-in signal reshaping circuit, after wave reshaping to the next driver, ensure wave-form distortion not accumulate.
- Built-in electric reset circuit and power lost reset circuit.
- Each pixel of the three primary color can achieve 256 brightness display, completed 16777216 color full color display, and scan frequency not less than 400Hz/s.
- Cascading port transmission signal by single line.
- Any two point the distance more than 5m transmission signal without any increase circuit.
- When the refresh rate is 30fps, cascade number are not less than1024 points.
- Send data at speeds of 800Kbps.
- The color of the light were highly consistent, cost-effective..

Applications

- Full-color module, Full color soft lights a lamp strip.
- LED decorative lighting, Indoor/outdoor LED video irregular screen.

General description

WS2812B is a intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. It internal include intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a 12V voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.

The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel，the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.

LED with low driving voltage, environmental protection and energy saving, high brightness, scattering angle is large, good consistency, low power, long life and other advantages. The control chip integrated in LED above becoming more simple circuit, small volume, convenient installation.

Worldsemi

WS2812B

Intelligent control LED integrated light source

Prameter	Smybol	conditions	Min	Tpy	Max	Unit
Input current	I_I	V_I=V_{DD}/V_{SS}	——	——	±1	μA
Input voltage level	V_{IH}	D_{IN}, SET	0.7V_{DD}	——	——	V
	V_{IL}	D_{IN}, SET	——	——	0.3 V_{DD}	V
Hysteresis voltage	V_H	D_{IN}, SET	——	0.35	——	V

Switching characteristics（T_A=-20～+70°C, V_{DD}=4.5～5.5V,V_{SS}=0V,unless otherwise specified）

Prameter	Symbol	Condition	Min	Tpy	Max	Unit
Transmission delay time	t_{PLZ}	CL=15pF,DIN→ DOUT,RL=10KΩ	——	——	300	ns
Fall time	t_{THZ}	CL=300pF,OUTR/OUTG/OUTB	——	——	120	μs
Input capcity	C_I	——	——	——	15	pF

RGB IC characteristic parameter

Emitting color	Model	Wavelength(nm)	Luminous intensity(mcd)	Voltage(V)
Red	13CBAUP	620-625	390-420	2.0-2.2
Green	13CGAUP	522-525	660-720	3.0-3.4
Blue	10R1MUX	465-467	180-200	3.0-3.4

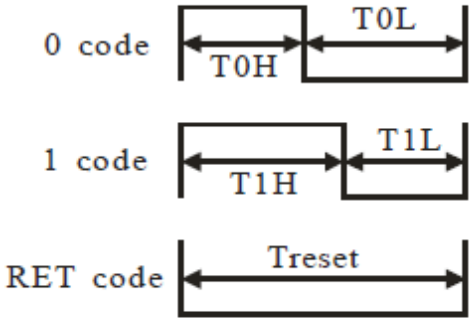
Data transfer time(TH+TL=1.25μs±600ns)

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.8us	±150ns
T0L	0 code , low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.45us	±150ns
RES	low voltage time	Above 50μs	

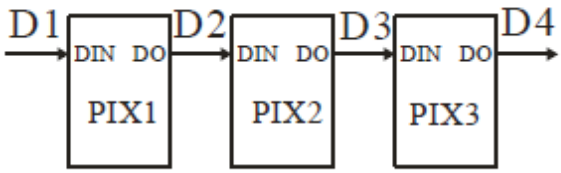
Worldsemi

WS2812B

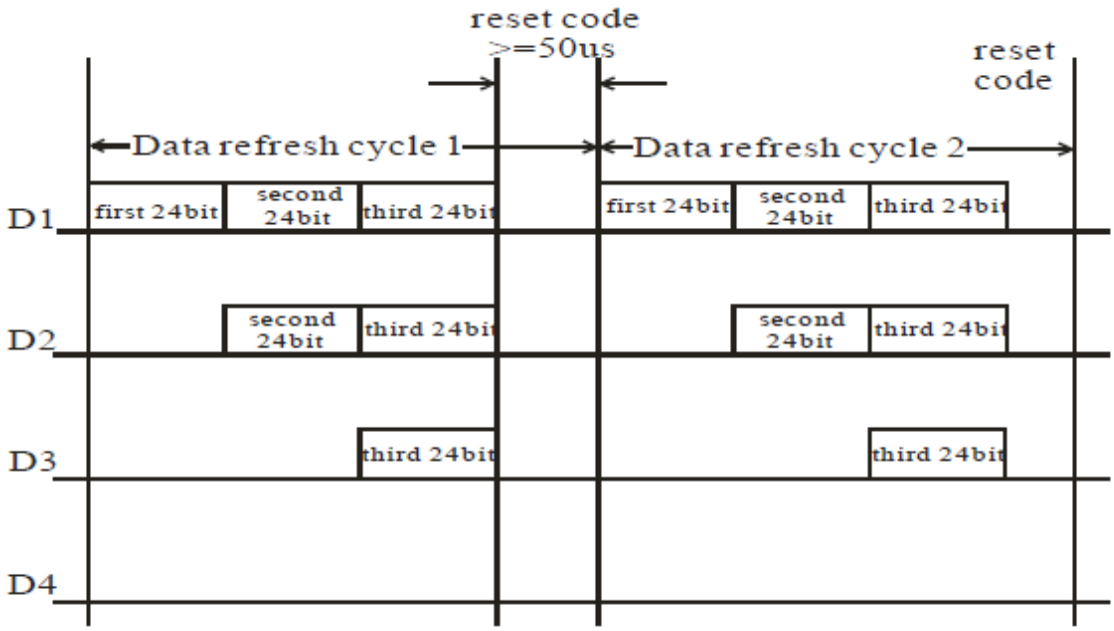
Intelligent control LED integrated light source



Cascade method:



Data transmission method:



Worldsemi

WS2812B

Intelligent control LED integrated light source

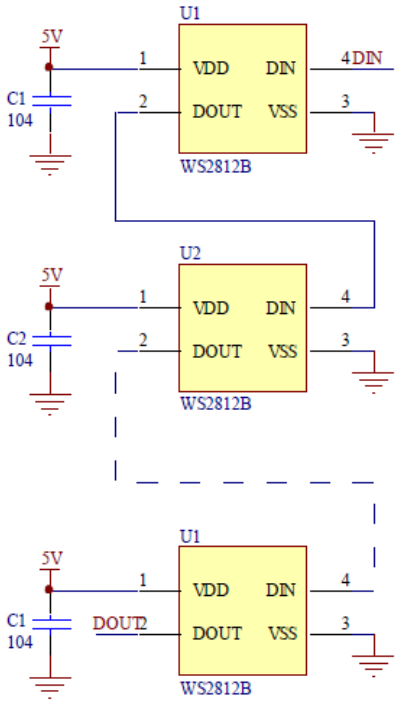
Note: The data of D1 is send by MCU,and D2, D3, D4 through pixel internal reshaping amplification to transmit.

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

Typical application circuit:



B. Zusätzliche Bilder des Projektes

Abbildung 39: Mit drei Prozent Leistung pro LED leuchtende WS2812B [HK17]

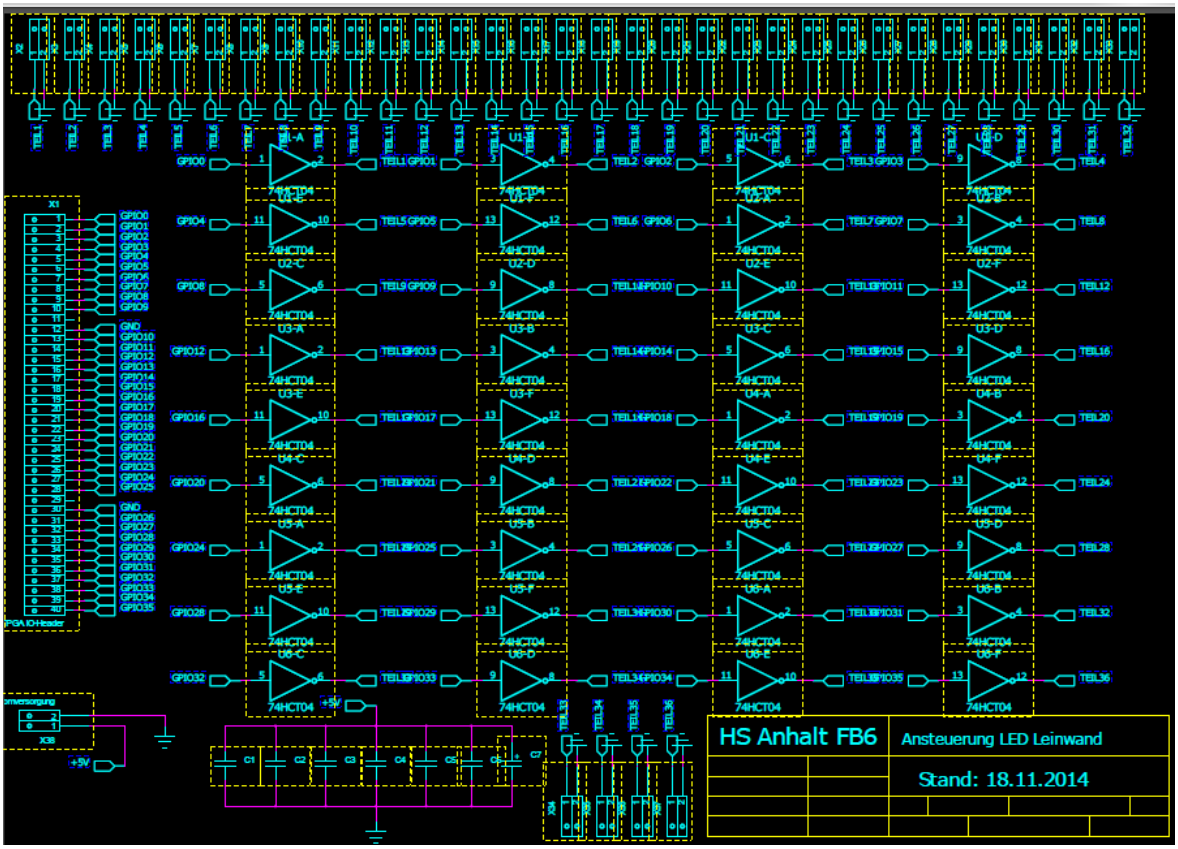


Abbildung 40: Schaltplan der Inverterschaltung für die Pegelwandlung von 3,3 V auf 5,0 V [CS14]

C. Spektrum einer WS2812B

Das in der Abbildung 41 dargestellte Lichtspektrum einer WS2812B wurde mit dem Spectro-Radiometer "specbos 1211" aufgenommen. Dabei wurde das Messgerät auf die LED Wand gerichtet, während alle 18.000 Pixel mit 100 Prozent Leuchtkraft in der Farbe Weiß leuchteten.

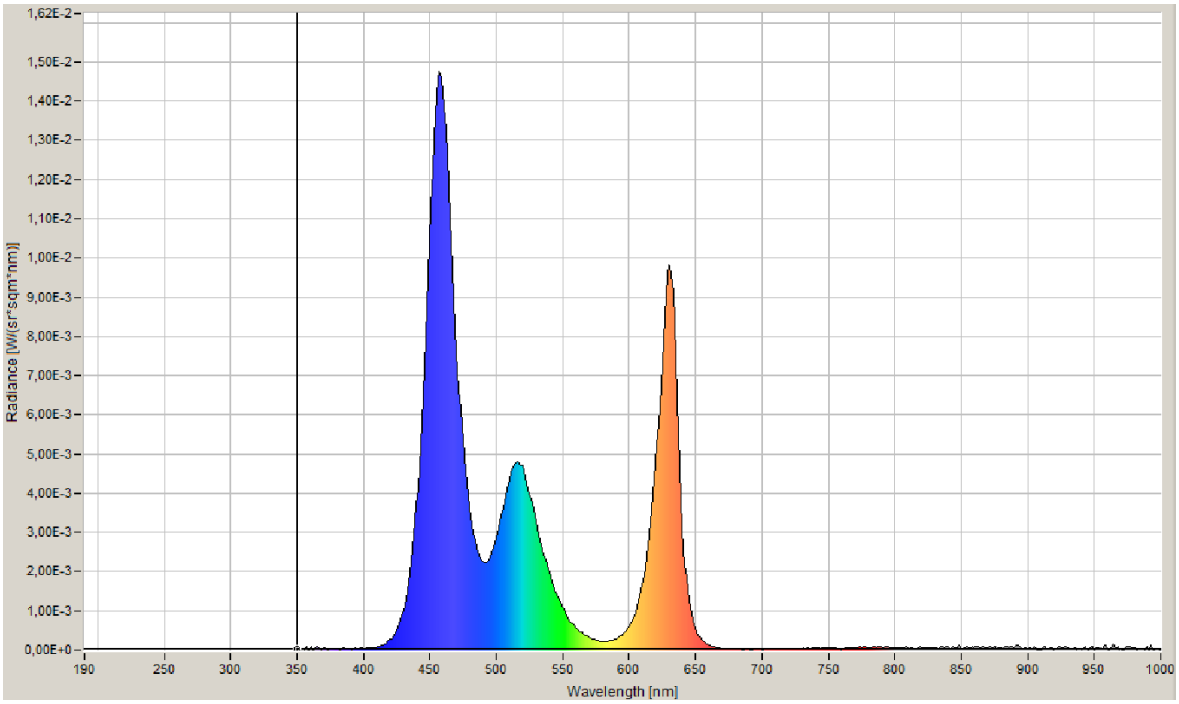


Abbildung 41: Messergebnis der Spektralanalyse [HK17]

Color	Wavelength(nm)	Intensity(mcd)	Operating Current(mA)	Operating Voltage(V)
RED	620-630	550-700	16	1.8-2.2
GREEN	515-530	1100-1400	16	2.8-3.1
BLUE	465-475	200-400	16	2.9-3.2

Tabelle 10: Parameter der einzelnen LEDs einer WS2812B [WS17]

D. Ansätze für Weiterentwicklungen

Abschließend soll hier noch einmal erläutert werden, wo sich Potential für eventuelle Weiterentwicklungen auf Basis dieser Masterarbeit eröffnet.

Optimierung der Benutzerfreundlichkeit bei der Bilddarstellung

Im momentanen Entwicklungsstand sind noch immer einige Vorarbeiten seitens des Anwenders nötig, um mit dem System Bilder auf der LED Wand darstellen zu können. Hierbei handelt es sich zum einen um die Vorbereitung des Bildmaterials. Da dieses momentan noch aufwendig in Format und Dateityp angepasst werden muss sowie die Bilddaten, wie in Kapitel 3.4 beschrieben, umgeformt werden müssen, gibt es hier großen Spielraum für Optimierungen. Ein Lösungsansatz hierbei ist die Entwicklung einer Software, welche alle diese Aufgaben realisiert. Dies könnte zum einen als klassische Windows Anwendung oder beispielsweise als Anwenderskript im Matlab programmiert werden. Eine Funktionsübersicht ist in der folgenden Abbildung 42 dargestellt. Damit würde die Vorbereitung der Bilder stark vereinfacht werden.

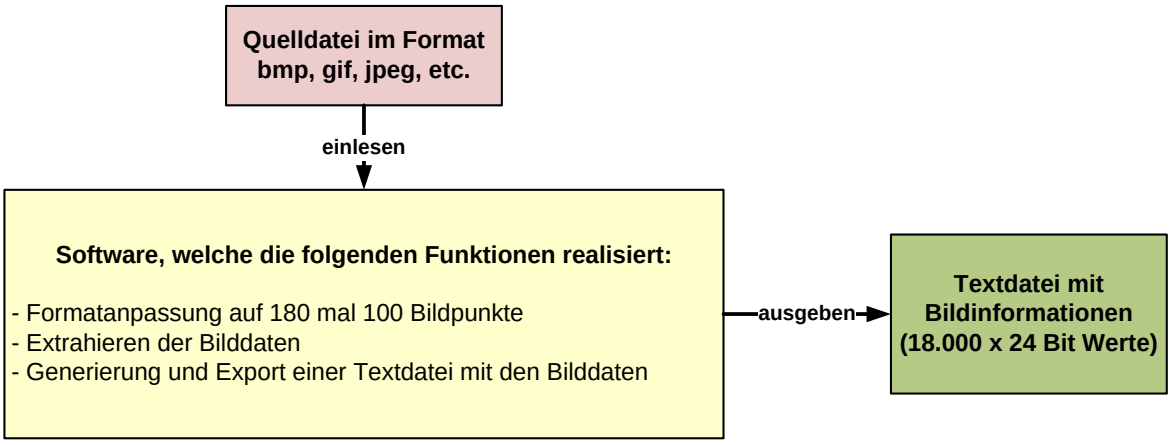


Abbildung 42: Software für die Bildvorbereitung [HK17]

Des Weiteren muss momentan nach jedem Neustart des DE2-115 Boards das Programm für den NIOS-II Prozessor vom Anwender neu aufgespielt werden. Theoretisch lässt sich dieses Programm in die FPGA Konfiguration für den NIOS-II Controller fest rein programmieren. Für eine nachträgliche Änderung der Controller Konfiguration über die Software QSys wird jedoch die *.sopcinfo Datei benötigt. Sie enthält alle Informationen für die Konfiguration des NIOS-II Controllers. Da diese Datei vom Hersteller leider nicht mit geliefert wurde, ist eine nachträgliche Änderung der NIOS-II Konfiguration nicht mehr möglich. Ein möglicher Lösungsansatz hierfür wäre, den kompletten NIOS-II Controller im QSys neu aufzusetzen und dabei das C-Programm fest mit im Speicher einzubinden. Dafür muss vom C-Programm in der Software Eclipse eine hex-Datei erzeugt werden. Diese wird dann bei dem Entwurf des NIOS-II Prozessors im QSys mit eingebunden und automatisch in die FPGA Konfiguration aufgenommen. Somit wäre nach dem Neustart des Boards keine Programmierung mehr erforderlich.

Entwicklung eines SDRAM Treibers als Puffer für den Bildspeicher

Eine Möglichkeit, um die Ansteuerungslogik der LED Wand videofähig zu gestalten, ist die Entwicklung eines Treibers für die SDRAM Bausteine des DE2-115 Boards. In Verbindung mit dem Treiber sollten die RAM Bausteine die 24 Bit Datentelegramme in Echtzeit puffern können. Die bereits vorhandenen Module ADV7180 Ansteuerung, YCbCr auf RGB Umwandlungsmodul sowie Teile des LED Strip Treibers können dafür innerhalb einer Weiterentwicklung der Ansteuerungslogik verwendet werden. Dabei würde der ADV7180 Chip die Videodaten eines PAL Signals einlesen und an den FPGA übergeben. Auf diesen würde dann zum einen die Formatumwandlung der Daten vom YCbCr Format in das RGB Format realisiert werden und zum anderen eine Anpassung der Auflösung von PAL auf die der LED Wand stattfinden. Die aufbereiteten Daten würden dann im SDRAM gepuffert werden, um von dort vom LED Strip Treiber zur Darstellung auf der LED Wand abgerufen werden zu können.

Eine Möglichkeit die PAL Auflösung anzupassen besteht darin, das Videosignal, welches Bilder mit einer Auflösung von 720 mal 576 Pixeln liefert, jeweils oben und unten um 88 Zeilen zu beschneiden. Dann würde es genau passen, jeweils vier mal vier Pixel des PAL Bildes zu einem Pixel für die Darstellung auf der LED Wand zusammenzufassen (720 / 4 = 180, 400 / 4 = 100). Eine Übersicht von dieser Umformung ist in der folgenden Abbildung 43 dargestellt.

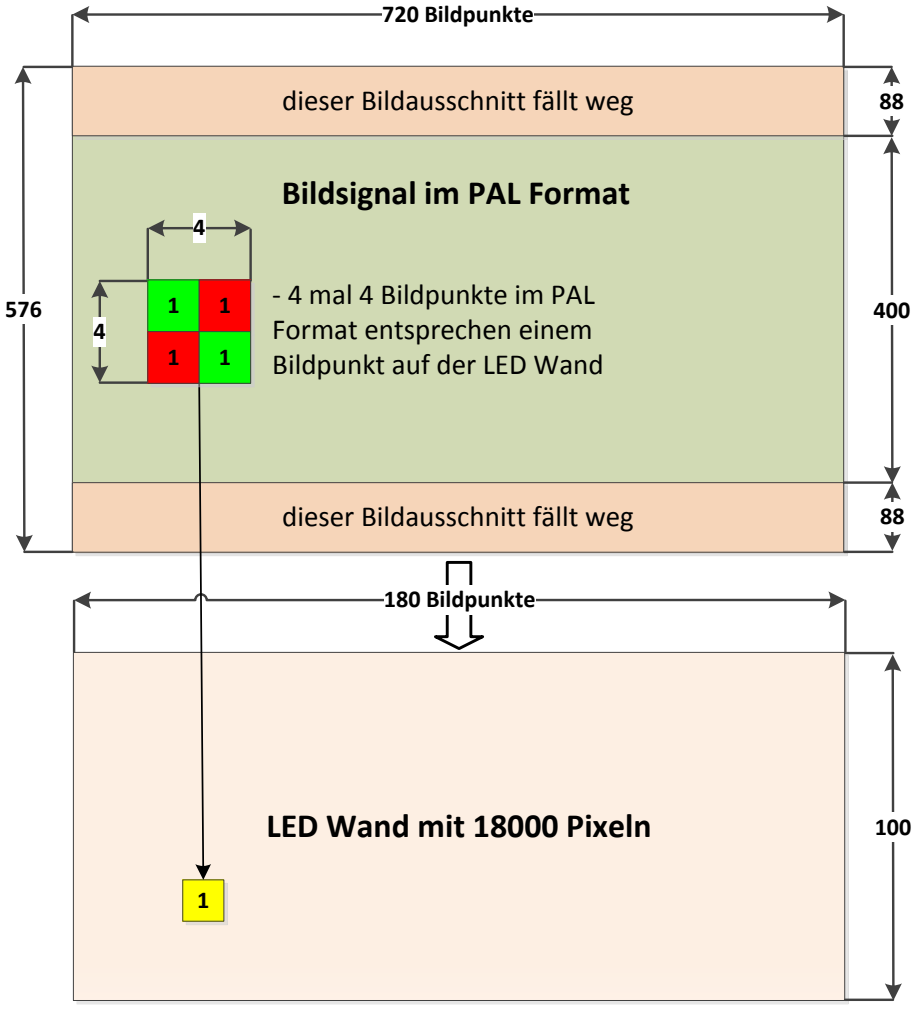


Abbildung 43: Anpassung der PAL Auflösung [HK17]

Symbol- und Abkürzungsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bezeichnung
BMP	Windows Bitmap (Dateiformat für Bilder)
CMOS	Complementary Metal Oxide Semiconductor (sich ergänzender Metalloxid-Halbleiter)
DC	direct current (Gleichstrom)
EMW	Elektrotechnik, Maschinenbau und Wirtschaftsingenieurwesen
FPGA	Field Programmable Gate Array
fps	frames per second, Bildwiederholfrequenz
I/O	Input/Output
kbit	Kilobit
LED	lichtemittierende Diode
LVCMOS	Low Voltage Complementary Metal Oxide Semiconductor
LVTTL	Low Voltage Transistor Transistor Logik
mA	Milliampere
mcd	Millicandela
Mhz	Megaherz
ms	Millisekunde
nm	Nanometer
ns	Nanosekunde
PCI	Peripheral Component Interconnect
PCI-X	Peripheral Component Interconnect eXtended
PWM	Pulsweitenmodulation
TTL	Transistor Transistor Logik
V	Volt
µs	Mikrosekunde

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur- und Quellenverzeichnis

[ALT10] Altera Corporation
www.altera.com
Bild vom NIOS-II Soft Core Prozessor

[ALT16] Altera Corporation
Cyclone IV Device Handbook
Kalifornien USA, 2016

[BP04] Bühler, Peter
MediaFarbe analog & digital
Springer, Berlin, 2004

[CPP11] Chu, Pong P.
Embedded SoPC Design with NIOS-II Processors and VHDL Examples
John Wiley & Sons, New Jersey, 2011

[CS14] Christian Schulz
Bau einer LED Hintergrundbeleuchtung für das Labor Medientechnik
Hausarbeit, HS Anhalt, Köthen, 2014

[HK17] Hannes Knothe
selbst erstelltes Bild
Masterarbeit, Leipzig, 2017

[PR17] Dave Stevenson
http://www.primrosebank.net/computers/mtx/projects/mtxplus/cpu/cpld/mtxplus_cpld_7128_prog.htm
Bild vom Prozessablauf in der Software Quartus II

[PS14] Paul Stoffregen, 2014
https://community.arm.com/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-19-89/4786.led_5F00_macro.png
Bild von einer vergrößerten WS2812B

[RGB17] Wolfgang Scheidle, Tussenhausen, Germany
http://www.cilab.de/rgb.shtml
Bild vom RGB Farbraum

[TE10] Terasic Technologies
Terasic DE2-115 user manual
Taiwan, 2010

[TI13] Texas Instruments Incorporated
SN74HCT04 HEX INVERTER user manual
Dallas, Texas USA, 2013

[WIKI17] Wikidedia, 2017
https://de.wikipedia.org/wiki/YCbCr-Farbmodell#/media/File:Barns_grand_tetons_YCbCr_separation.jpg
Bild vom YCbCr Farbraum

[WM15] Wäger, Markus
Das ABC der Farbe
Rheinwerk Design, Bonn, 2015

[WS17] WORLDSEMI CO., LIMITED
Datenblatt der WS2812B
WS2812B Specifications, 2017

Lebenslauf

Angaben zur Person

Name	Hannes Knothe
Anschrift	Leonhard-Frank-Straße 34, 04318 Leipzig
Geburtsdatum	19.02.1984
Geburtsort	Leipzig
Staatsangehörigkeit	deutsch
Familienstand	verheiratet

Ausbildung und berufliche Tätigkeiten

1990 – 1994	118. Grundschule in Leipzig
1994 – 2002	Felix-Klein Gymnasium Leipzig
2002 – 2003	Grundwehrdienst
2003 – 2006	Ausbildung zum Mechatroniker bei STILL Leipzig
2006 – 2009	Angestellter Techniker im Außendienst bei STILL Leipzig
2009 – 2013	Bachelorstudium der Elektrotechnik an der HS Anhalt in Köthen
2014 – 2018	Masterstudium der Elektro- und Informationstechnik an der HS-Anhalt in Köthen