Inhalt

Einleitung

Das Regeln/die Regelung ist ein Vorgang bei dem eine Größe, die zu regelnde Größe (Regelgröße), fortlaufend erfasst, mit einer anderen Größe, der Führungsgröße verglichen und im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst. [Definition Regelung nach DIN 19226]

Diese Arbeit beschäftigt sich mit der Realisierung eines Messglied, bei denen geringe Totzeiten im Fokus stehen. Totzeiten können allgemein dafür sorgen, dass ein Regelkreis instabil oder träge wird und somit einer definierten Anforderung nicht mehr entsprechen könnte.

Die Messgröße soll die horizontale Position eines Objektes darstellen, die anhand seines Farbunterschied zum Rest einer Szene (Hintergrund) erkannt wird. Für die Erkennung eines solchen Objektes bietet sich eine Sensorlösung an, die Teile des elektromagnetischen Spektrums aufnimmt und diese in ein elektrisches Signal umwandelt.

Farbe ist, in dieser Arbeit durch die Auswahl des Sensors auf eine Kamera der Firma Sentech, durch einen Rot- Grün und Blaukanal (RGB) definiert. Diese drei Farbkanäle spiegeln die Grundfarben wider, welcher der Sensor dem natürlichen Licht entnimmt.

Um die Totzeiten im Messglied gering zu halten, wurde bei der Auswahl der Kamera auf eine hohe Anzahl an Bildern die Sekunde geachtet. Die Auswertung der Bilder für die Erkennung eines Objektes und dessen Position erfolgt in einem Field Programmalbe Gate Array (FPGA-Board). Das FPGA-Board bietet die Möglichkeit die Signale von der Kamera in Echtzeit zu verarbeiten und auszuwerten und hat aufgrund seines Aufbaus weniger Limitationen als ein herkömmlicher Mikroprozessor, was die Bearbeitungsgeschwindigkeit angeht.

Zu Bearbeitungsbeginn der Studienarbeit war es möglich, die Daten der Kamerabilder mittels Camera Link von der Kamera auf das FPGA-Board zu übertragen und diese dort zwischenzuspeichern. Die zwischengespeicherten Daten konnten über den VGA-Ausgang auf einem angeschlossenen Monitor ausgegeben werden. [Lukas Herbst, Bilddatenvorverarbeitung in einem FPGA] Zudem war es möglich anhand von Graustufen (Schwarzweißbild) und dessen Wechsel in der Intensität bis zu drei Objekte zu erkennen.

Auf Basis der zuvor beschreibenden Funktionen werden neue Funktionen implementiert. Die Funktionsweise der bisherigen Objekterkennung im Graustufenbereich soll für die Erkennung im Farbbereich als Vorlage verwendet werden. So sollte es einfacher sein ein farbiges Objekt von einem anders gefärbten Hintergrund zu unterscheiden und somit auch dessen Position zu erkennen.

Wichtig bei der Erfassung ist das Herausfinden der relevanten Parameter, die eine robuste Objekterkennung erst möglich machen. So spielen beispielsweise die Beleuchtungszeit, Umgebungslicht und die Farbwahl von Objekt und Hintergrund eine große Rolle.

…Bezug auf den Aufbau der Studienarbeit

Digitale Bildverarbeitung

Diese Studienarbeit stützt sich in vielen Abschnitten aus Erkenntnissen der vorangegangenen Studienarbeit „Bilddatenvorverarbeitung in einem FPGA, 25. Januar 2020“ von Lukas Herbst. Dessen Arbeit bezieht sich in vielen Teilen auf die Literatur von Rafael C. Gonzalez und Richard E. Woods, die zusammen das Buch „Digital Image Processing“ verfasst haben. Eine detaillierte Beschreibung zu diesem Abschnitt finden Sie in der Arbeit von Herrn Herbst. Folgend die Details die für diese Ausarbeitung am wichtigsten waren.

Funktionsweise Darstellung von Pixeln

Ein digitales Bild kann aus einer endlichen Anzahl aus Pixeln bestehen, so wird meist mit x die horizontale Breite und mit y die vertikale Höhe definiert. Die Gesamtzahl der Pixel in einem Bild ergibt sich aus dem Produkt von x und y.

So ergibt sich ein Raster aus Pixeln. Jeder Pixel kann mit der Funktion f(x,y) einzeln ausgelesen werden.

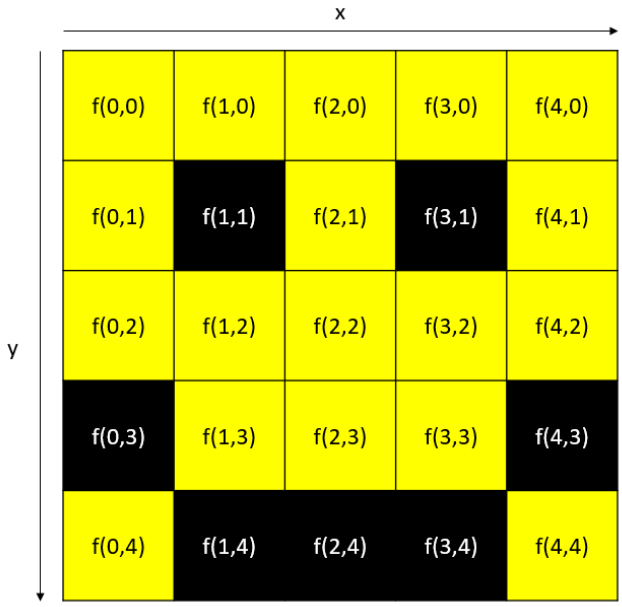


Abbildung 1: Digitales Bild Beispiel

Der Rückgabewert der Funktion f(x,y) ist abhängig von der Definition des Pixels. So wird beispielsweise für ein Bild im Graustufenbereich nur ein einziger Wert definiert oder wie in Abbildung Bilderverarbeitung 1 ein RGB Farbwert. Ein RGB Farbwert verfügt über drei Informationen, diese definieren wie stark ein rot, grün oder blau Anteil ausgeprägt ist.

Beispiel zu Abbildung Bildverarbeitung 1: {rot = 255, grün = 255, blau = 0} = f(0,0)
Das Beispiel ergibt einen starken Gelbton. Die Zahl 255 stellt hier den maximal Wert für die Darstellung mit 8-Bit pro Farbkanal dar und wird auch Farbtiefe genannt. Je mehr Bits für die Farbtiefe zur Verfügung stehen, desto feiner können Farbunterschiede realisiert werden. Für diese Arbeit wird von einer Farbtiefe von 8-Bit ausgegangen.

Farbunterschiede erkennen

In [2.3 Mathematische Operationen zur Bilddatenverarbeitung, Bilddatenvorverarbeitung in einem FPGA, Lukas Herbst] beschrieben, können Unterschiede von Pixeln in eine Reihe wie folgt erfasst werden.

$$1.\, Ableitung\ Definiton\ von\ Gonzalles\colon\ \frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Das Ergebnis der ersten Ableitung beschreibt den Unterschied von einem zum nächsten Pixel. Folgend eine Veranschaulichung zu einer Reihe die aus 5 Pixeln besteht. Jeder Pixel wird durch eine Graustufe repräsentiert.

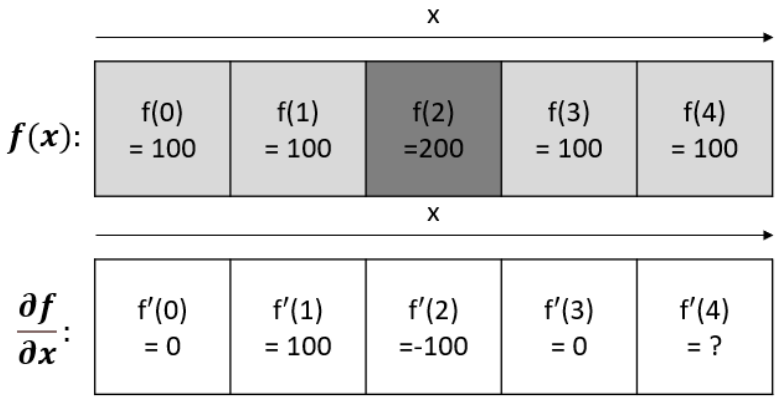


Abbildung 2: Pixelreihe f(x) und 1. Ableitung f(x), Graustufen in 8 Bit Darstellung

An der Stelle f(0) ergibt die erste Ableitung null, da der Nachbar Pixel keinen Unterschied aufweist. Ein Sprung ist an einem hohen Betrag der ersten Ableitung zu sehen, so zum Beispiel an f(1) oder f(2). Hier ändert sich der Wert sehr stark im Vergleich zum Nachbarn. Eine Besonderheit stellen die Ränder da, hier kann man mit der Formel der ersten Ableitung keinen Wert berechnen und kann daher vernachlässigt werden.

Übersicht Ist-Zustand

Eine kurze Übersicht bietet folgende Grafik und beschreibt den vorgefundenen Zustand, der bei Beginn des Projektes übernommen wurde.

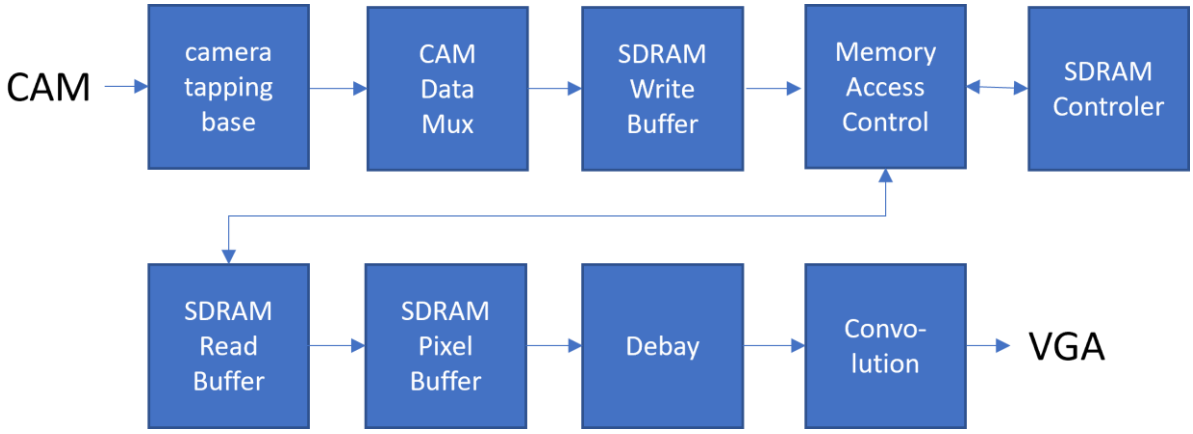


Abbildung 3: Übersicht der Module innerhalb des bisherigen Quartusprojekts

In der Ausarbeitung der Studienarbeit „Bilddatenvorverarbeitung in einem FPGA“ von Herrn Herbst, sind die Module im Detail beschrieben. Daher werden im Folgenden die Module in Ihrer groben Funktion beschrieben, um sich ein Gesamtbild des Projektes machen zu können.

Abbildung 3 verfasst die Aufteilung der Module innerhalb des FPGA Bordes. Der Aufbau der vorgefundenen Hardware ist in Abbildung 4 zu sehen.

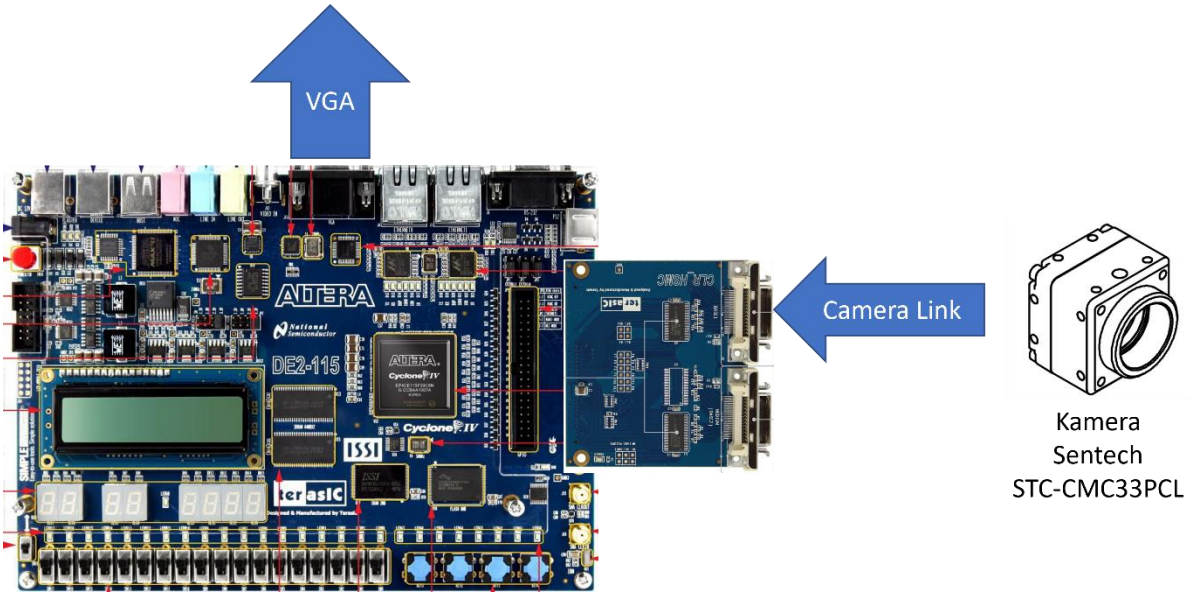


Abbildung 4: Übersicht verwendete Hardware

Modul: cameralink_tapping_base

Die Kamera „STC-CMC33PCL“ verfügt über verschiedene Modi für die Übertragungsweise der Kameradaten. Der erste Kontaktpunkt zwischen FPGA-Modul und Kamera stellt das Modul „cameralink_tapping_base“ dar. In dem Modul werden die Daten der Camera Link Schnittstelle nach der Spezifikation der Kameramodi vorverarbeitet.

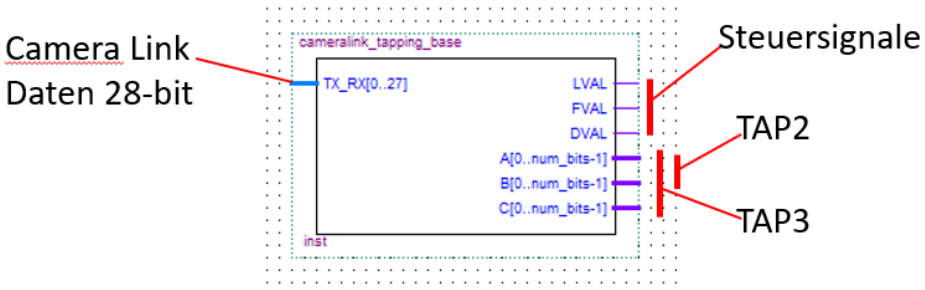


Abbildung 5: Blockschaltbild

Die Eingangsgröße „CLRRX_BASE[0..27]“ stellt die Daten dar, die von der Kamera an das FPGA Board übermittelt werden. Die Definition vom Eingang „TX_RX“ kann in der Spezifikation zu Camera Link nachgelesen werden und wird an dieser Stelle nicht genauer erläutert. Um die Ausgangsgrößen aus Abbildung 5 zu verstehen, benötigen wir ein Verständnis dafür welche Modi in der Kamera zur Verfügung stehen.

Kameramodi: Taps, horizontales und vertikales Timing

Ein Bild wird von der Kamera zeilenweise übermittelt, für die Übertragung werden die Steuersignale LVAL, FVAL und DVAL verwendet.

LVAL = Line Valid, HIGH definiert gültige Pixel
FVAL = Frame Valid, HIGH definiert gültige Zeile
DVAL = Data Valid, HIGH definiert Daten gültig

Jeder Pixel kann dabei aus einer Bittiefe von 8, 10 oder 12 Bits bestehen, dies ist eine weitere Einstellung der Kamera.

Horizontales Timing 2 Tap

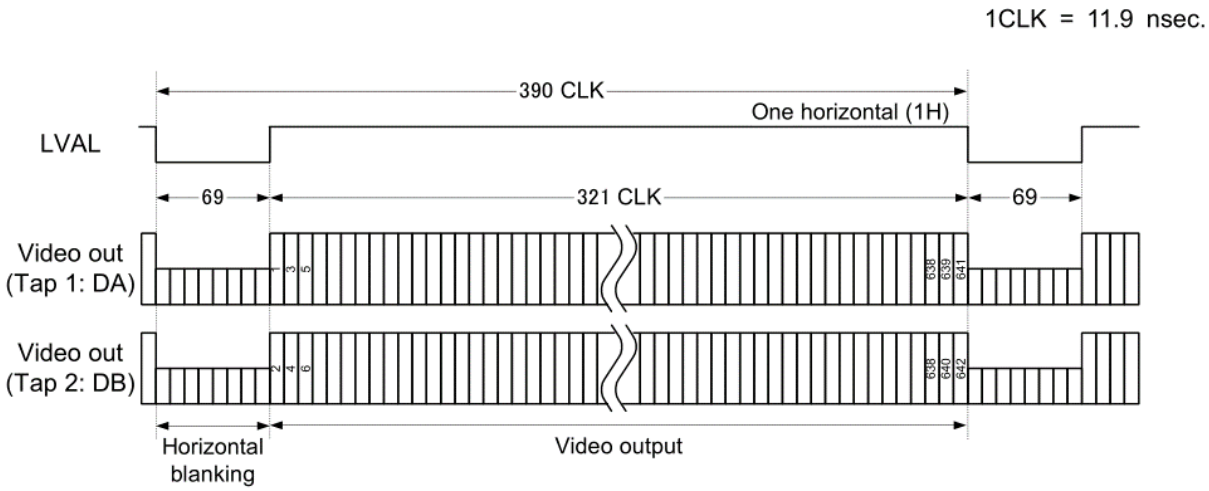


Abbildung 6: Übersicht Timing horizontal, Modi 2Tap, 642 Pixel horizontal

„2 Tap“ bedeutet das eine Bildzeile aufgeteilt über zwei Kanäle übertragen wird. In Abbildung 6 werden über das Signal „Tap 1: DA“, alle ungeraden Pixel aus den Gesamten 642 übertragen und in „Tap 2: DB“ alle geraden Pixel. Durch die parallele Übertragung wird der Datendurchsatz verdoppelt im Vergleich zur einfachen Datenübertragung bei gleichem Takt.

Diese Einstellung wird in der Kamera als „TAP Count“ eingestellt, oben beschrieben die Einstellung „TAP Count = 2Tap“. Darüber hinaus gibt es die Einstellung „3Tap“, welche die die Übertragung auf drei Kanäle aufteilt, jedoch nicht anders in der Funktionsweise zu „2Tap“ daher keine genauere Erläuterung dazu an dieser Stelle.

Das Signal „LVAL“ zeigt mit einer steigenden Flanke den Beginn der Pixel an.

Vertikales Timing

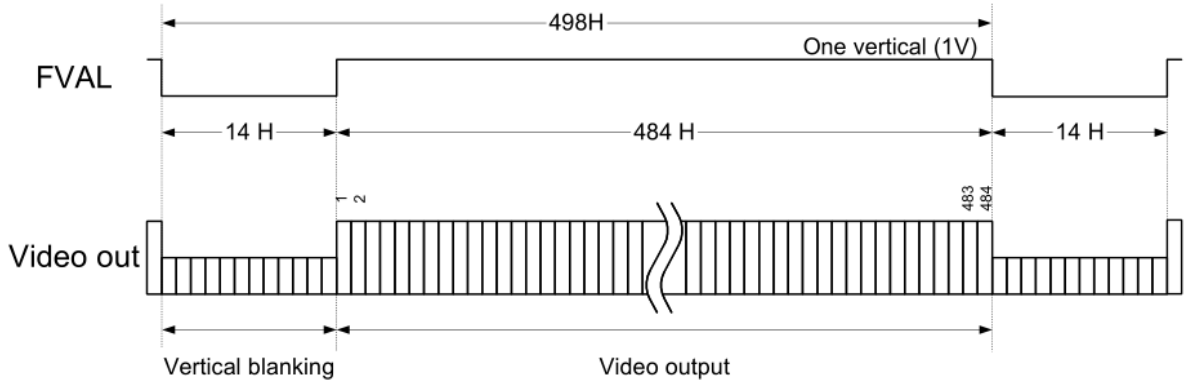


Abbildung 7: Übersicht Timing vertikal, 484 Zeilen vertikal

In Abbildung 7 wird das Timing der vertikalen Zeilen erläutert. Wichtig beim Lesen der Abbildung ist die Einheit der Zeit, anders als bei Abbildung 6 ist hier die der Takt (CLK) Ausschlag gebend, sondern die Anzahl der vergangenen Zeilen. So steht „498H“ für 498 Zeilen die Vergangen sind.
(H = horizontals).

Modul: camera_data_mux_gen

Das Modul vereint die Signale „TAP1“ und „TAP2“ aus Abbildung 5: Blockschaltbild in einem 4 Byte Shift Register. Eine detaillierte Erklärung des Moduls kann in der Ausarbeit „Seite 19 camera_data_mux_gen, Bilddatenvorverarbeitung in einem FPGA, Lukas Herbst“ gefunden werden.

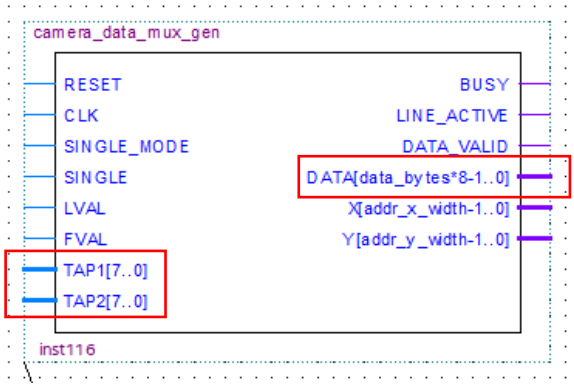


Abbildung 8: Blockschaltbild

Aufbau Intern: camera_data_mux_gen (Vereinfachte Version)

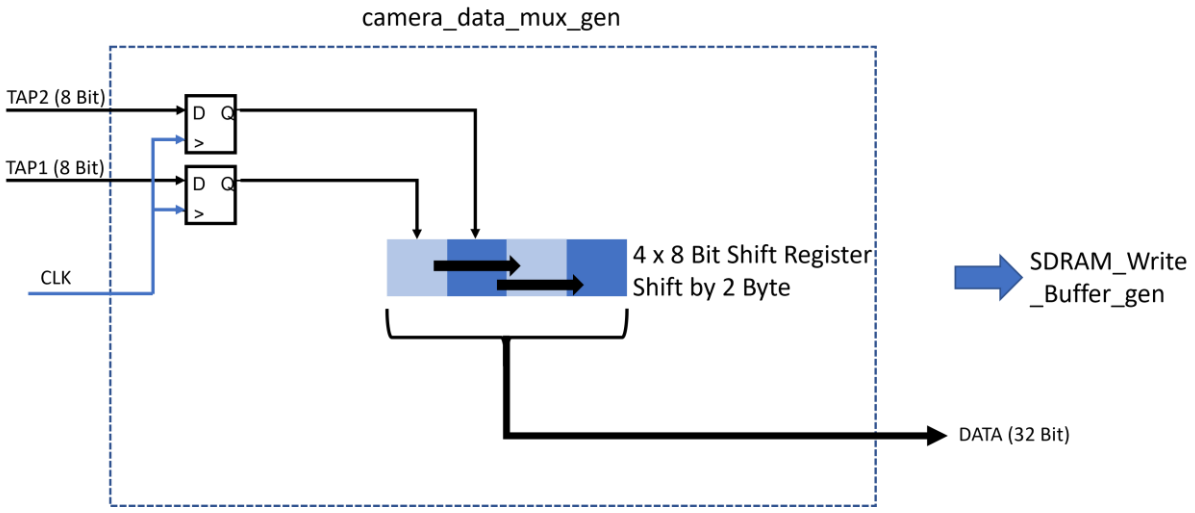


Abbildung 9: Vereinfachte Funktionsweise Modul „camera_data_mux_gen“

Modul: SDRAM_Write_Buffer_gen

Das Modul nimmt die vorgearbeiteten Kameradaten aus dem Modul „camera_data_mux_gen“ entgegen und schreibt diese mithilfe des Moduls „Memory Access Control“ in den SDRAM der sich auf dem FPGA Board befindet. Es ist die Schnittstelle zwischen dem Kameradatenbereich und den SDRAM-Bereich. Beide Bereiche sind unterschiedlich getaktet, daher benötigt das Modul aus beiden Bereichen den Takt. So werden die Eingangsdaten der Kamera mit dem Takt „cam_clk“ verarbeitet und die Synchronisation mit dem Modul „Memory Access Control“ über den Takt „sdram_clk“ realisiert.

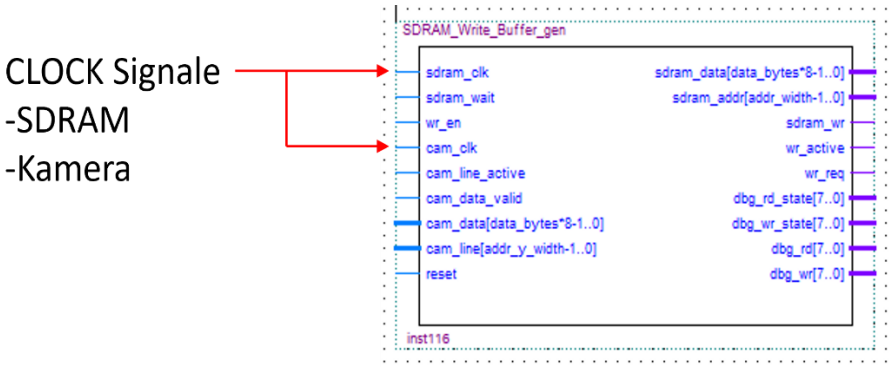


Abbildung 10: Blockschaltbild

Die Kameradaten werden in einem FIFO gepuffert, um dann bei freiem Schreibzugriff auf den SDRAM diese dort zu speichern. Der Ausgang „sdram_data“ sind die Kameradaten, die in den SDRAM geschrieben werden und „sdram_addr“ gibt die Position im SDRAM vor. In der aktuellen Form werden jeweils zwei Byte per Taktänderung aus dem FIFO in den SDRAM geschrieben.

Aufbau Intern: SDRAM_Write_Buffer_gen (Vereinfachte Version)

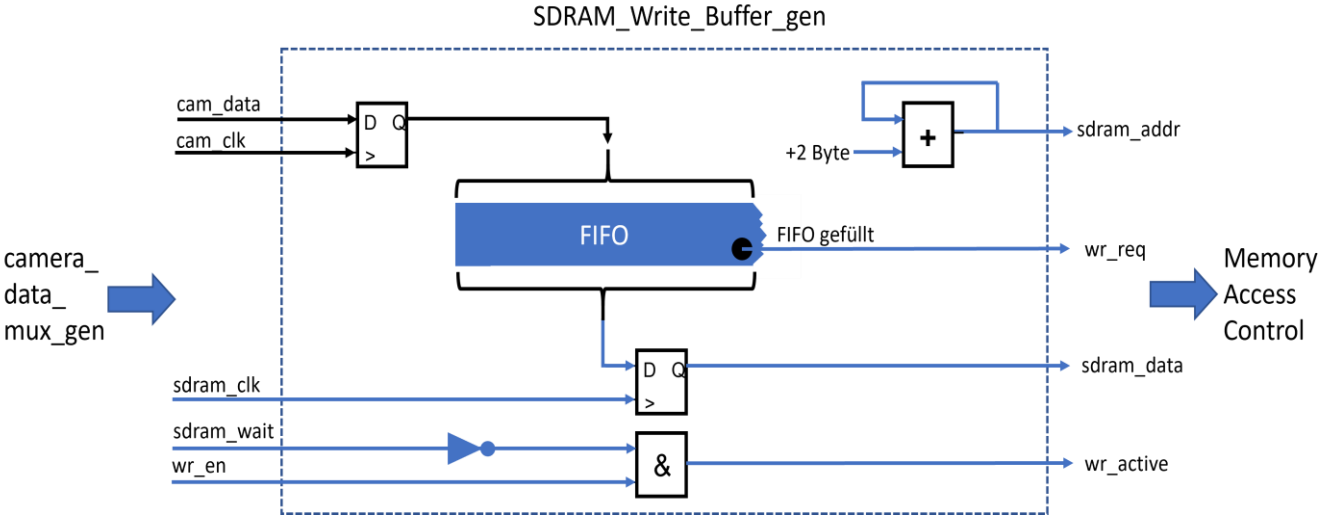
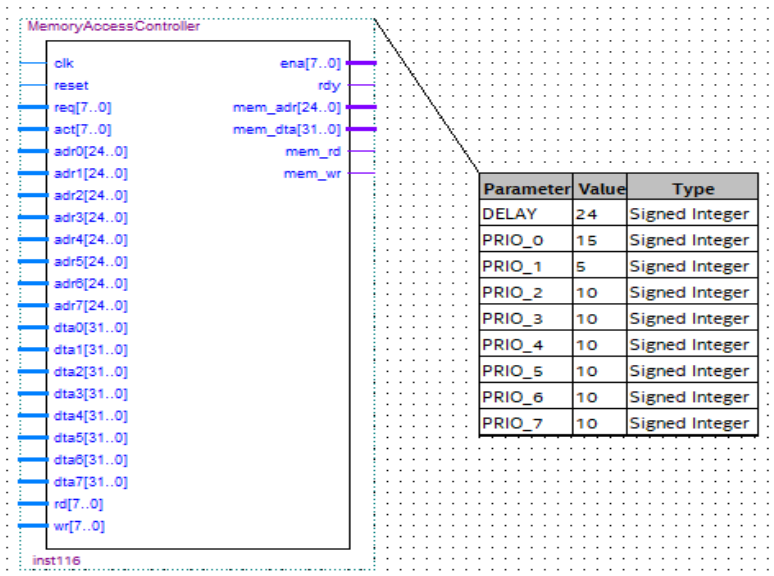


Abbildung 11: Vereinfachte Funktionsweise Modul „SDRAM_Write_Buffer_gen“

Modul: Memory Access Control

Das Modul regelt den Zugriff von mehreren Quellen auf den gleichen SDRAM.



Parameter	Value	Type
DELAY	24	Signed Integer
PRIO_0	15	Signed Integer
PRIO_1	5	Signed Integer
PRIO_2	10	Signed Integer
PRIO_3	10	Signed Integer
PRIO_4	10	Signed Integer
PRIO_5	10	Signed Integer
PRIO_6	10	Signed Integer
PRIO_7	10	Signed Integer

Modul: SDRAM_Pixelbuffer

Durch vorherige Schritte liegen Bilddaten im SDRAM vor. Die einzelnen Pixel werden jeweils als Byte dargestellt und sind einzeln aus dem SDRAM lesbar. Für die weitere Bildverarbeitung und die Ausgabe auf einer VGA Schnittstelle werden immer 5x5 Pixelblöcke im Modul „SDRAM_Pixelbuffer“ gebuffert.

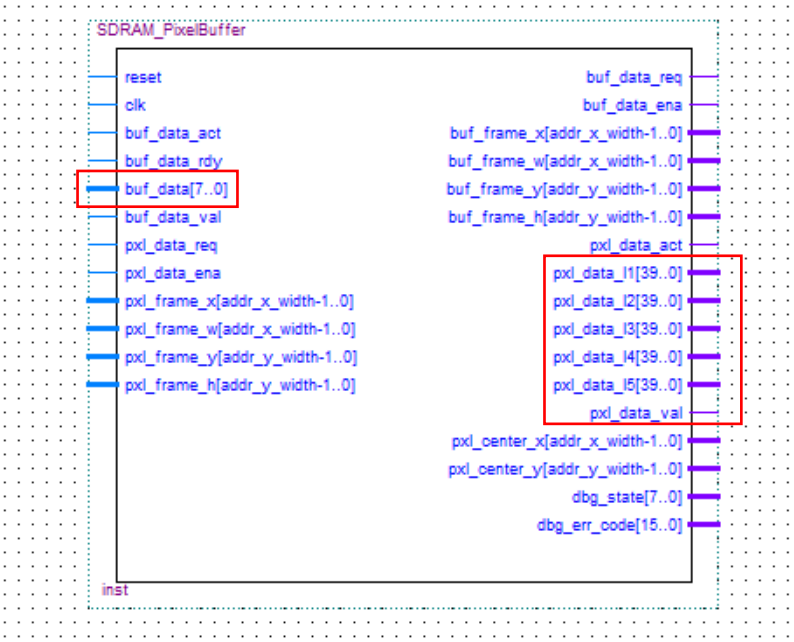


Abbildung 12: Blockschaltbild, rot markiert die wichtigsten IOs

Der 5x5 Byte Block ist zeilenweise auf den Ausgängen „pxl_data_l1[39 .. 0]“ bis „pxl_data_l5[39 .. 0]“ lesbar. Die Ausgänge dürfen erst gelesen werden, wenn der Ausgang „pxl_datal_val“ einen logischen High-Pegel vorweist.

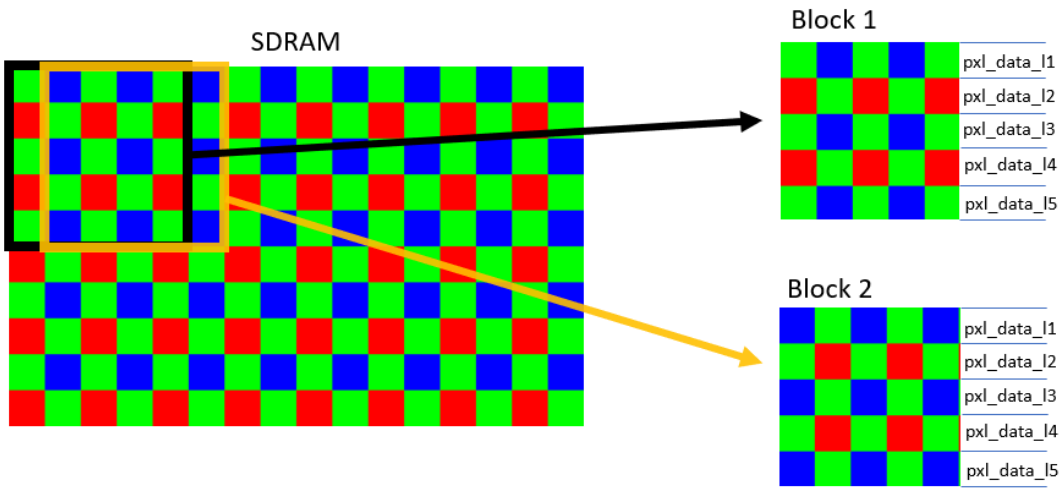


Abbildung 13: Schaubild Funktionsweise Pixelbuffer

Die fünf Ausgänge, für jeweils eine Zeile, sind auf den Eingang „pxl_datal1 ... pxl_datal5“ des Modul „debay“ gelegt für die Auswertung des Bayer Pattern.

Wenn ein Block komplett erfasst und bearbeitet wurde, wird der Block um eine Pixelspalte im Bild verschoben und der nächste Block wird gepuffert. Ist das Ende der aktuellen Spalte erreicht, dann wird der Buffer mit dem nächsten Block, am Spaltenbeginn um eine Zeile versetzt, befüllt. Siehe dafür Abbildung 14.

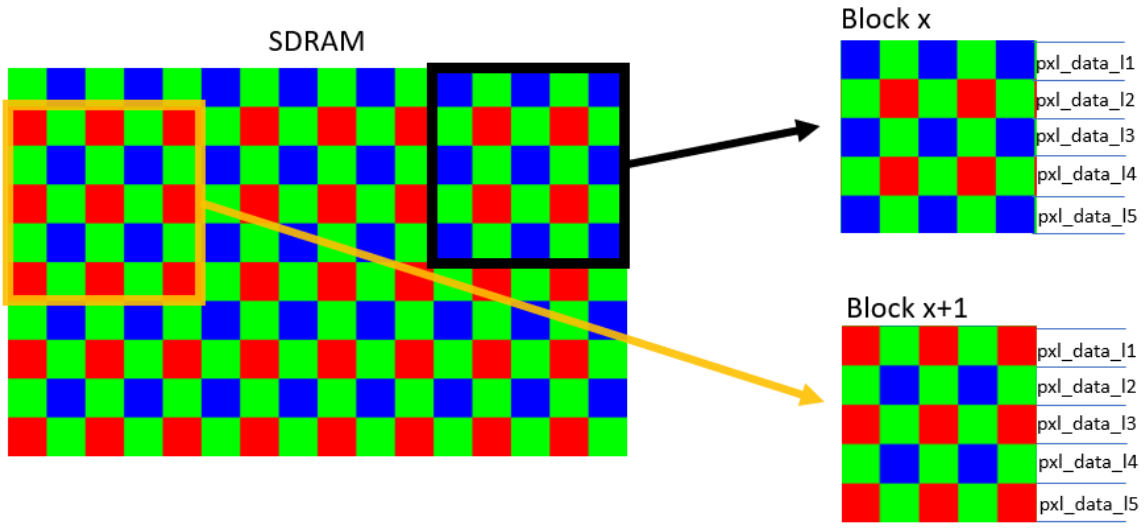


Abbildung 14 Schaubild Pixelbuffer Zeilenende

Aufbau Intern: SDRAM_Pixelbuffer(Vereinfachte Version)

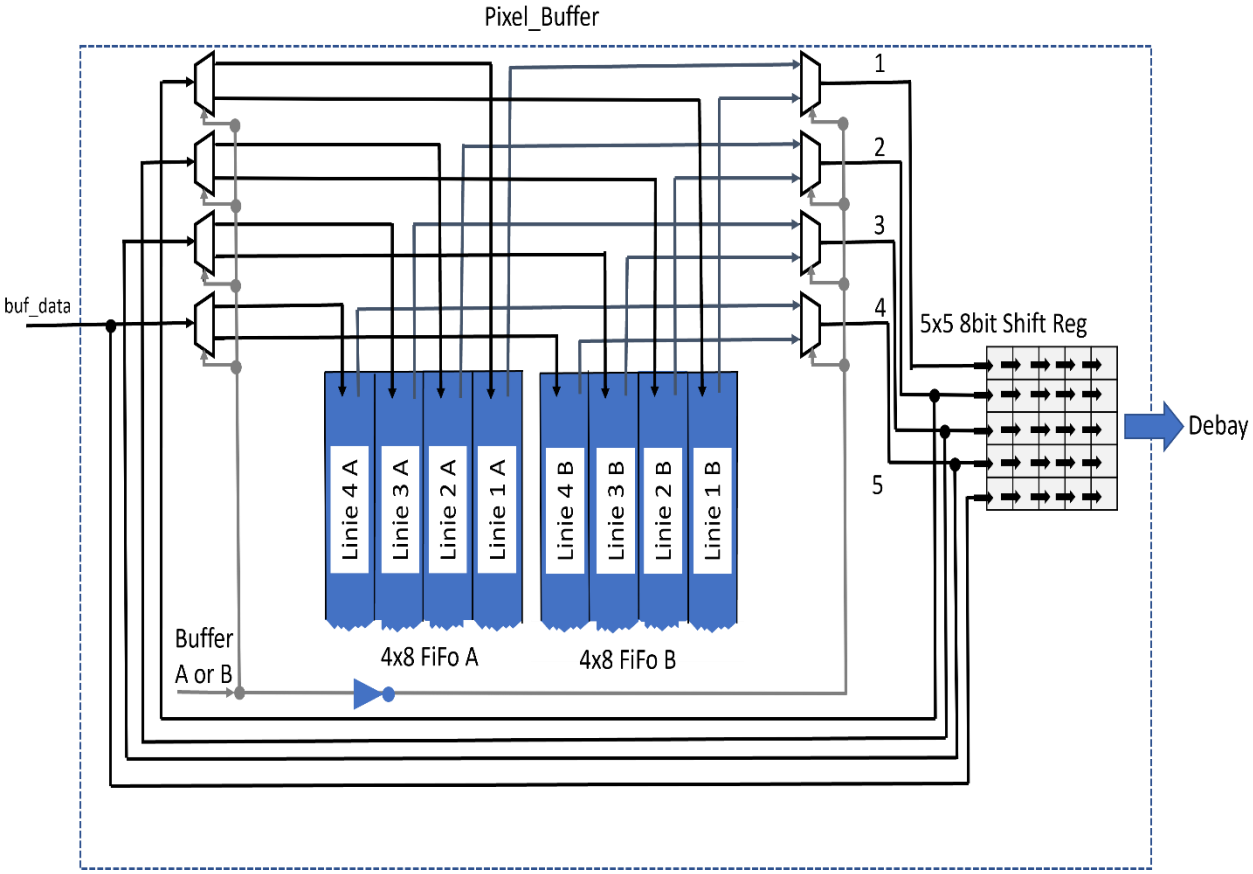


Abbildung 15 Schaubild interner Aufbau SDRAM_Pixelbuffer

Modul: Debay

Die Pixeldaten (5x5 Byte Block) des Modul „sdram_pixelbuffer“ liegen an den Eingängen „pxl_data_l1[39 .. 0]“ bis „pxl_data_l5[39 .. 0]“ an. Jeder Eingang steht für eine Zeile im Block.

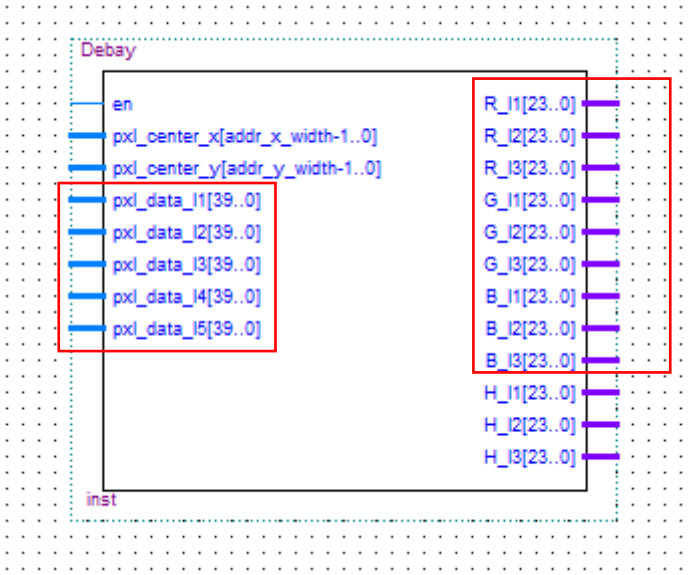


Abbildung 16 Blockschaltbild

Der 5x5 Byte Block ist nach dem Bayer Muster formatiert. Das Bayer Muster reduziert die Anzahl der Farbkanäle pro Pixel auf einen einzigen Farbkanal. Für die Ausgabe des Bildes über eine VGA-Schnittstelle sollen drei Kanäle pro Pixel zur Verfügung stehen. Im Modul „Debay“ wird aus dem 5x5 Byte Block für jeden der drei Farbkanäle (R, G, B), ein 3x3 Byte Block gewonnen. Um jeweils 3x3 Bytes pro Kanal zu bekommen, wird der 5x5 Byte Block wie in Abbildung 17 aufgeteilt.

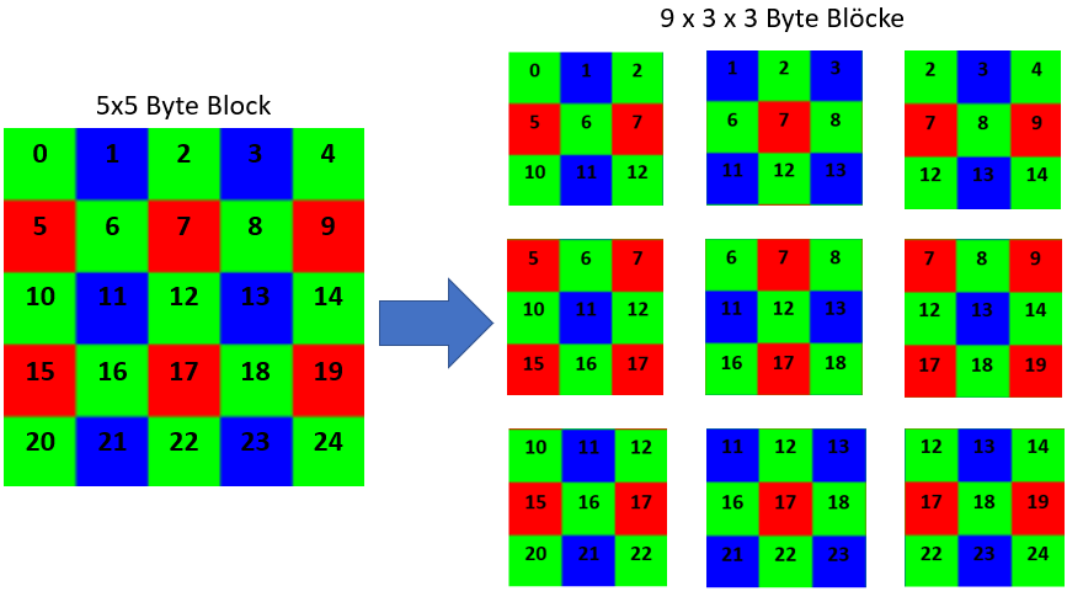


Abbildung 17 Aufteilung 5x5 Byte Block in kleinere 3x3 Blöcke

Durch die Aufteilung erhalten wir neun 3x3 Byte Blöcke. Aus jedem Block wird für alle drei Farbkanäle ein Byte extrahiert. Wie in Abbildung Debay 2 zu sehen, beinhaltet ein 3x3 Block mehr als nur eine Farbinformation für einen Kanal, so ist beispielsweise im ersten Block an Stelle 5 und 7 die Farbinformation für rot doppelt. Um aus zwei Byte ein zubekommen, nehmen wir das arithmetische Mittel der zwei Farbinformationen aus diesem Block. Zum besseren Verständnis dieser Vorgehensweise siehe Abbildung 18.

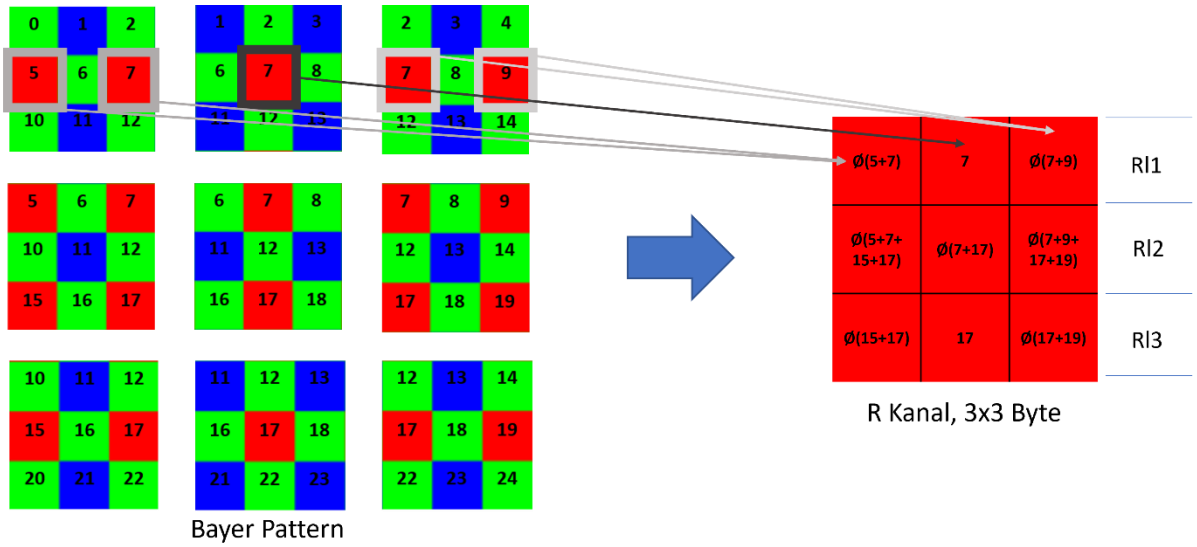
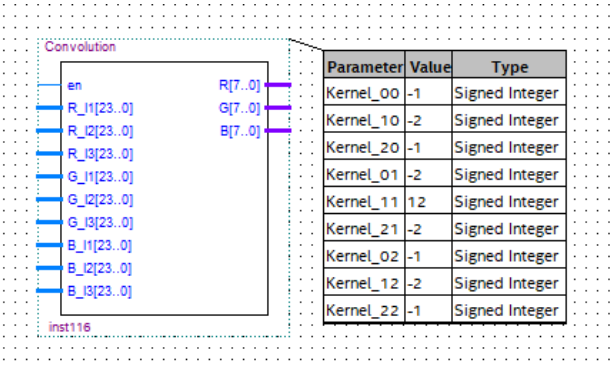


Abbildung 18: Zusammensetzung 3x3, Beispiel am Rotkanal, Rl1 = Output der ersten Zeile des Rotkanals

Nach dem Auswerten der kleineren Blöcke, werden die Informationen der 3x3 Blöcke pro Farbkanal, auf den, in Abbildung 16 rot markierten, Ausgängen gelegt.

Für die reine Ausgabe der Pixel über eine VGA Schnittstelle, würde ein Overhead bei der Anzahl der gelesenen Pixel entstehen, da 3x3 große Blöcke als Eingabe für das Decodieren des Bayer Pattern reichen würde. Jedoch ist die Größe von 5x5 als Eingabe gewünscht, da die zusätzlichen Pixel ein Maß an extra Information über Farbänderung und Farbbewegung liefert.

Modul: Convolution



Parameter	Value	Type
Kernel_00	-1	Signed Integer
Kernel_10	-2	Signed Integer
Kernel_20	-1	Signed Integer
Kernel_01	-2	Signed Integer
Kernel_11	12	Signed Integer
Kernel_21	-2	Signed Integer
Kernel_02	-1	Signed Integer
Kernel_12	-2	Signed Integer
Kernel_22	-1	Signed Integer

Abbildung 19: Blockschaltbild

„Convolution“ das bedeutet Faltung und ist der Hauptbestandteil der inneren Funktionsweise des Moduls in Abbildung 19. Die Eingangsgröße des Blocks ist die Ausgangsgröße zu sehen in Abbildung 18, es handelt sich um jeweils 3x3 Byte Blöcke für jeden Farbkanal (RGB). Aus diesem Byte Blöcken wird jeweils nur ein Farbwert gewonnen und auf die Ausgänge „R[7..0], G[7..0] und B[7..0]“ gelegt.

Das Ziel des Blocks ist es für jeden neuen Block einen RGB Pixelwert zu berechnen. Die Blöcke stammen aus dem ursprünglichen Bild und werden jeweils um **eine** Zeile in Ihrer Auswahl verschoben. Damit schlussendlich ein Bild mit der gleichen Anzahl an Pixel in Höhe und Breite zur Verfügung steht.

Für die Gewichtung der Farbanteile aus dem Byte Block stehen die Kernel Parameter 00-22 zur Verfügung.

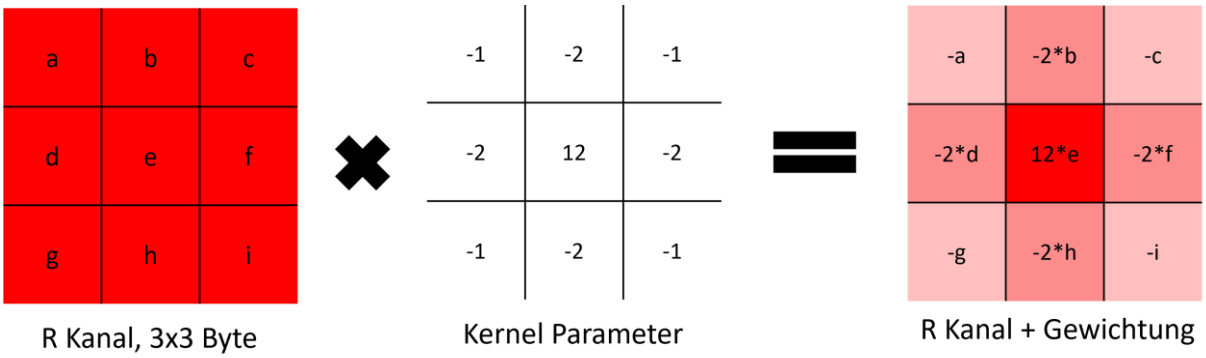


Abbildung 20: Gewichtung des 3x3 Rot Kanal für Berechnung Rot Kanal

Die Gewichtung führt dazu das in Abbildung 20 der Wert „e“ am stärksten betont wird, wobei die Nachbarwerte weniger gewichtet werden. Abbildung 21 zeigt, wie aus dem gewichteten R Kanal Block ein einziger Farbwert wird.

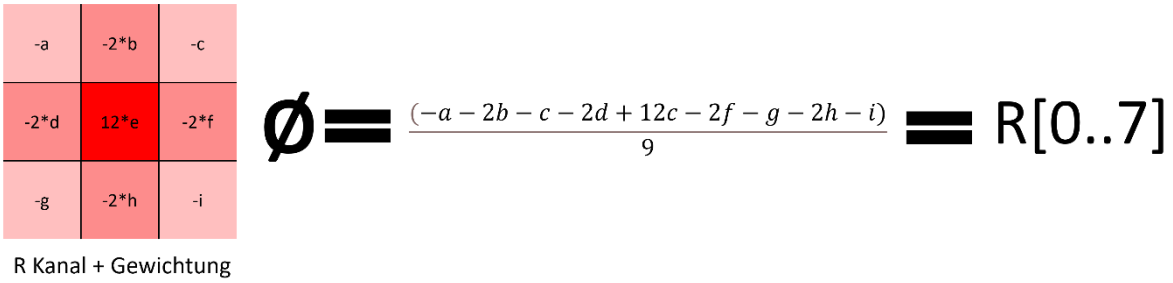


Abbildung 21: Berechnung Farbwert R

Übersicht Zustand (nach Studienarbeit Oster)

Während der Bearbeitung wurde die Erkennung eines farbigen Objektes in seiner X- und Y-Achse angestrebt. Dies wurde mit gewissen Einschränkungen erreicht, diese Einschränkungen werden im Kapitel „Beinflussende Paramater“ erläutert. Als Beispiel für die Erkennung diente der Aufbau in Abbildung 22.

Abbildung 22: Testaufbau Erkennung rotes Rechteck auf Bildschirm

Die Kamera ist auf das rote Rechteck gerichtet. Innerhalb des Kamerablinkwinkel soll die Position des Pixels gefunden werden, welches im Zentrum des Rechtecks sitzt. Die Position wird in Abhängigkeit der horizontalen und vertikalen Pixel dargestellt die maximal zur Verfügung stehen. Wenn das Objekt genau mittig im Sichtfeld der Kamera steht, so wird für die X-Achse eine Position von 320 und für die Y-Achse 240 ermittelt. Da, die verwendete Kamera, Bilder in einer maximalen Gesamtauflösung von 640x480 Pixeln aufgenommen werden können.

Die bisherige Ausgabe auf einem VGA-Monitor wurde um ein Zielkreuz (in Grün) erweitert, um die bisher erkannte Position auszugeben, dies zu sehen in Abbildung 18.

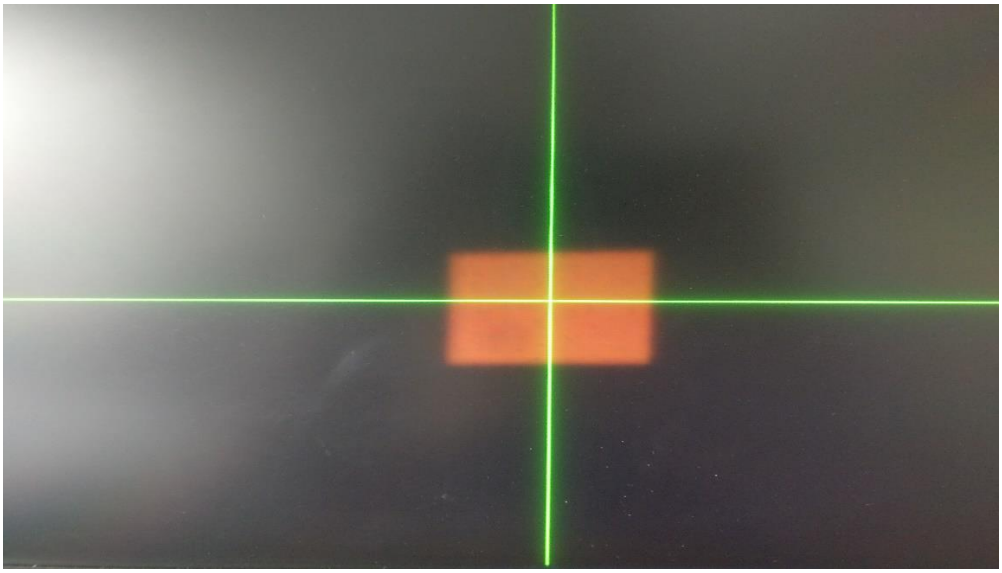


Abbildung 23: Zielkreuz auf erkannter Position

Funktionsweise/Anknüpfung an Vorheriges

Modul: LINE_DETECTION_CONV

Modul: DEB_OBJ_DETECTION