



Christian Oster <christian.oster95@gmail.com>

Fwd: Aktueller Stand Camera480Hz1 Nachricht

Prof. B. Gick <gick@hs-koblenz.de>
An: Christian Oster <choster@hs-koblenz.de>

22. April 2021 um 18:09

Viele Grüße,
Berthold Gick

Prof. Dr.-Ing. B. Gick
University of Applied Sciences
FB IW
Konrad-Zuse-Str. 1
D-56075 Koblenz
Tel. xx49-261-9528-384 (Büro)
-356 (Labor)
Fax xx49-261-9528-499
E-Mail gick@hs-koblenz.de
Büro Raum G140
Labor Raum G127

----- Weitergeleitete Nachricht -----

Betreff:Aktueller Stand Camera480Hz

Datum:Thu, 20 Jun 2019 11:47:12 +0200

Von:Josh Perske <jperske@hs-koblenz.de>

An:Prof.Dr.B.Gick (Prof.Dr.B.Gick) <gick@hs-koblenz.de>

Sehr geehrter Herr Gick,

Hier wie gewünscht der aktuelle Stand des Kamera Projekts.

Eine Übersicht was aktuell funktioniert und was ich geplant habe finden Sie in Bild 1 Finished.

Den Projektstand dazu finden Sie im Archiv 20190331_rw_buffer_fertig.qar.

Der Projektstand 20190620_mac_wip.qar ist noch unfertig, darin bin ich den Memory Access Controller am überarbeiten.

In den Projekten sind folgende Eingaben möglich:

- * Taster 0: Reset
- * Taster 1: Einzelbild Trigger
- * Switch 0: Wechsel zwischen Einzelbild (0) und Continuous (1) Modus
- * Switch 1: Wechsel zwischen internen (0) und externen (1) RAM

Der SRAM Read und SRAM Write Buffer sind als Ringpuffer realisiert. Die Anzahl der Zeilen die gepuffert werden, können über den Parameter LINE_BUFFER_N festgelegt werden. Außerdem werden über die Parameter die Datenbreite und die Bildgröße festgelegt. Der Read und der Write Buffer fordern jeweils über ein Request Bit Zugriff auf den RAM. Die Buffer sind erst aktiv, wenn auf den Request ein entsprechender Enable Eingang gesetzt wird. Im Projekt 20190331_rw_buffer_fertig.qar ist das noch über ein Schaltnetz im Schaltplan realisiert. Im Projekt 20190620_mac_wip.qar habe ich angefangen einen VHDL Memory Access Controller zu schreiben, der Anfragen von mehreren Puffern mit unterschiedlicher Priorität bearbeiten kann.

Auf Bild 1 sehen Sie außerdem meine Planung, welche Bausteine als nächstes ergänzt werden sollen. Ich habe dies in zwei Schritte unterteilt.

Im ersten Schritt soll ein weiterer Read Buffer hinzugefügt werden um das parallele Auslesen des RAM mit dem

Memory Access Controller zu testen. Als zusätzlichen Baustein soll dahinter ein weiterer Pixel Buffer hinzugefügt werden. Die Funktion des Pixel Buffer sehen sie auf Bild 2. Eingangsdaten sind die Daten aus dem Read Buffer, Ausgangsdaten sind ein 5x5 Pixel Array. Dieses 5x5 Array bietet bei der weiteren Auswertung der Bilddaten immense Vorteile.

Um dieses Pixel Array zur Verfügung zu stellen, werden 8 FIFO Puffer mit der Breite des Bildes benötigt. Mit Beginn eines neuen Frame werden zunächst die vier mit A bezeichneten Puffer mit den ersten vier Bildzeilen gefüllt. Danach kann die Ausgabe der Ausgangsdaten Taktsynchron erfolgen. Mit jedem Takt werden die Ausgangsdaten einen Pixel weiter geschoben und die nächsten fünf Pixel werden nachgeschoben. Die Pixel der unteren vier Zeilen werden aus Buffer A entnommen die obere Zeile direkt aus dem Read Buffer. Parallel wird Zeile 2 aus Buffer A in Zeile 1 von Buffer B geschoben, Zeile 3 aus A in Zeile 2 aus B usw. Nach einer Bildzeile werden nun Buffer A und Buffer B gewechselt und nun werden die nächsten Pixel aus B entnommen und eine Zeile versetzt in A zurück geschrieben. Mit diesem Baustein kann somit ein 5x5 Pixel Array zur Verfügung gestellt werden, welches mit jedem Takt um 1px verschoben wird und bei einem Zeilenwechsel latenzfrei zur nächsten Zeile springt.

Der Vorteil des 5x5 Pixel Array wird auf Bild 3 deutlich. Dort ist der Image Pre-Processing Baustein grob beschrieben. Aus 5x5 Pixel Rohdaten können über simple Schaltnetze drei 3x3 Pixel RGB Daten generiert werden. Die Dann zur Verfügung stehenden 3x3 Pixel können via Laplace-Filterung auf Kanten untersucht werden, auch das ist dann ohne Buffer mit einem simplen Schaltnetz möglich. Ergebnis sind dann die RGB Kantendaten des Pixel in der Mitte des 5x5 Feldes. Diese Pixeldaten können dann vom Image Processing Controller verwendet werden um Korrelationen mit einem Referenzbild zu berechnen.

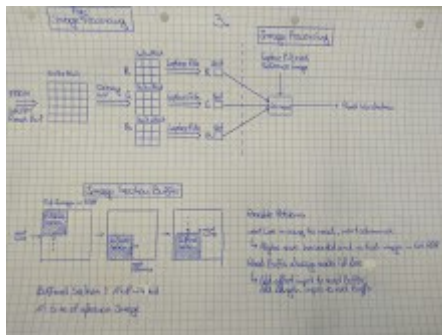
Zuletzt noch ein paar Worte zur Planung des Image Processing. Hinter dem Pre-Processing stehen jetzt Pixeldaten zu Verfügung und es kann mit jedem Takt der nächste Pixel der Zeile angefordert werden. Die Idee ist jetzt durch den Image Processing Controller einen Bildausschnitt zu laden, der dann mit einem Referenzbild verglichen werden kann (siehe Bild 1 Step 2). Dazu könnten zwei Image Section Buffer angelegt werden. Der eine enthält das Referenzbild, der Andere einen Ausschnitt aus dem Kamerabild. Die Korrelation der beiden Bilder kann mittels Schaltnetzen ausgerechnet und zusammen mit den Koordinaten des Bildausschnitts abgespeichert werden. Der Image Processing Controller verschiebt dann den Bildausschnitt um einen Pixel weiter und lädt die nächsten Pixel nach. Das Ergebnis wäre ein Korrelationswert für jede Koordinate des Bildes (außer an den Randbereichen), siehe dazu Bild 3 Image Section Buffer.

Um den Image Section Buffer möglichst schnell mit der nächsten Bildzeile oder -spalte zu füllen, gibt es zwei starke Optimierungen. Das Eine wäre, mit zwei Write Buffern das Bild einmal normal und einmal 90° gedreht auf dem Ram abzulegen. Das Andere wäre eine Optimierung am Read Buffer. Wenn man diesem direkt einen Zeilen- und eine Spaltenoffset und eine Zeilenlänge übergibt, können direkt nur die als nächstes benötigten Bilddaten gepuffert werden.

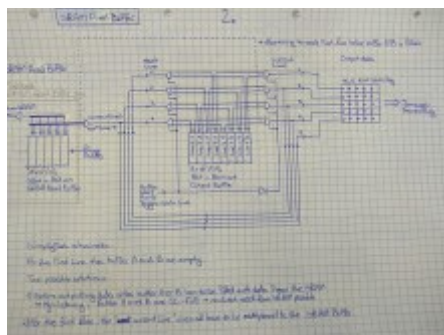
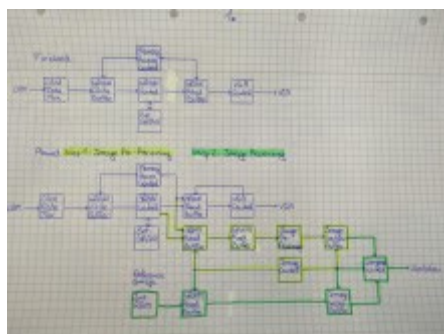


Ich hoffe das ist einigermaßen verständlich, gerade um die Funktionsweise des Pixelbuffer habe ich mir schon ziemlich viele Gedanken gemacht. Bei Fragen stehe ich natürlich gerne zur Verfügung.

Mit freundlichen Grüßen
Josh Perske

5 Anhänge



3.jpg
1881K

**2.jpg**
2128K**1.jpg**
1686K **20190331_rw_buffer_fertig.qar**
1910K **20190620_mac_wip.qar**
93K