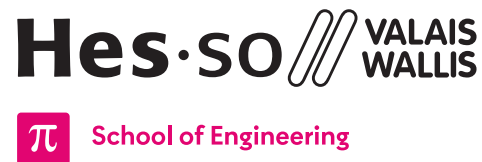


# Vorlesung Digital Design (DiD)



**Orientierung:** [Informatik und Kommunikationswissenschaften \(ISC\)](#)

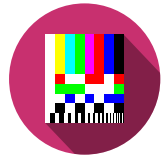
**Spezialisierung:** Data Engineering (DE)

**Kurs:** [Digital Design \(DiD\)](#)

**Autoren:** [Silvan Zahno](#), [Axel Amand](#)

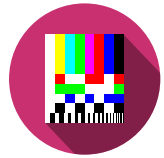
**Datum:** 04.12.2025

**Version:** v2.5



# Inhalt

1	Einführung .....	3
2	Spezifikation .....	4
2.1	Funktionen .....	4
2.2	Schaltung .....	4
2.3	<b>Video Graphics Array (VGA)</b> Timing (Beispiel) .....	6
2.4	HDL-Designer Projekt .....	8
3	Komponenten .....	9
3.1	<b>Field Programmable Gate Array (FPGA)</b> -Platine .....	9
3.2	Knöpfe und <b>Light Emitting Diode (LED)</b> .....	9
3.3	<b>Peripheral Module (PMod)</b> - <b>Digital Visual Interface (DVI)</b> Modul .....	10
4	Bewertung .....	11
5	Erste Schritte .....	12
5.1	Vorgehensweise .....	12
5.2	Tips .....	13
	Glossar .....	14



# 1 Einführung

Ziel des Projekts ist es, das erworbene Wissen am Ende des Semesters direkt mit Hilfe eines praktischen Beispiels anzuwenden. Es geht darum, ein Display über einen **VGA** Schnittstelle anzusteuern, um ein vordefiniertes Bild anzuzeigen. Dieses Displaysystem ist in der Abbildung [Abbildung 1](#) dargestellt.

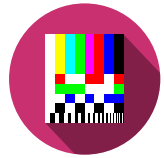


Abbildung 1 - Hardwareaufbau Display (EBS3)

Die Aufgabe besteht aus einer klar definierten minimalen [Spezifikationen 2](#), welche von der Entwicklungsgruppe mit zusätzlichen Funktionen optional erweitert werden kann, z. B. zwischen einem berechneten und einem vorab aufgenommenen Bild umzuschalten, den Bildschirm zu animieren ...



Mithilfe von Zusatzfunktionen können einige Extrapunkte erarbeitet werden.



## 2 Spezifikation

### 2.1 Funktionen

Die Basisfunktionen sind wie folgt definiert:

- Falls die Taste **start** gedrückt wird, wird ein Testbild auf dem Monitor angezeigt.
- Falls die Taste **stop** gedrückt wird, wird das Testbild entfernt und der Monitor wird schwarz.
- Das Testbild wird von der Schaltung generiert (*nicht vorab aufgenommen*) und besteht aus allen möglichen Farbkombinationen welche mit dem **3bit per pixel (bpp)** DVI Modul möglich sind. Die Abbildung [Abbildung 2](#) zeigt ein mögliches Testbild das angezeigt werden kann.
- Die verwendete Auflösung muss 640px x 480px @ 60 Hz betragen.



Abbildung 2 - Mögliches Testbild welches alle Farbkombinationen anzeigt

### 2.2 Schaltung

Die **FPGA** Entwicklungskarte bildet das Kernstück des Systems. Dort wird ein **LED** - Kapitel 3.2 angeschlossen sowie ein **DVI modul** - Kapitel 3.3. Am **High Definition Multimedia Interface (HDMI)** Ausgang des Modules wird der Bildschirm angeschlossen. Das Gesamtsystem ist schematisch in der Abbildung [Abbildung 3](#) ersichtlich.

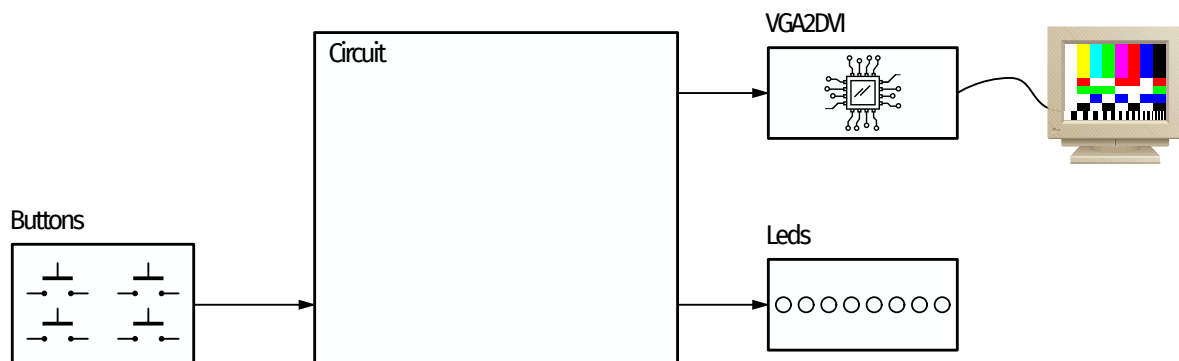
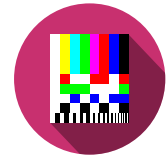


Abbildung 3 - Display Schaltung



Die komplette Schaltung funktioniert wie folgt:

- Vier Tasten werden zur Steuerung des Systems verwendet: **start**, **stop** sowie zwei frei verfügbare Knöpfe **button\_3** und **button\_4**. Diese können für optionale Funktionen benutzt werden. Ein '1' auf dem Signal bedeutet, dass die Taste gedrückt ist.
- Mit dem drücken des **start** Knopfes werden die Bildinformationen, die aus der Position der Pixel berechnet wird (z.B. farbige Balken), über die **VGA** Schnittstelle zum **PMod** -Module kontinuierlich übertragen.
- Durch das Drücken des **stop** Knopfes wird ein schwarzes Bild übertragen.
- Das **PMod** -Modul für die zuvor erwähnte **VGA** Video Schnittstelle besitzt die folgenden Signale: **vga\_dataEnable**, **vga\_pixelClock**, **vga\_hsync**, **vga\_vsync** sowie **vga\_rgb[2:0]**.
  - Das Modul konvertiert **VGA** -Video Signale zu **Transition Minimized Differential Signaling (TMDS)** Signalen und sendet diese über die **HDMI** Schnittstelle zum Monitor.
- Die **testOut**-Pins können verwendet werden, um zusätzliche Informationen über das System auszugeben, z. B. für Debuggingzwecke oder zur Kontrolle von **LED**.

Der leere Design-Toplevel zeigt alle Signale, die an die **FPGA** -Platine angeschlossen sind, siehe Abbildung **Abbildung 4**:

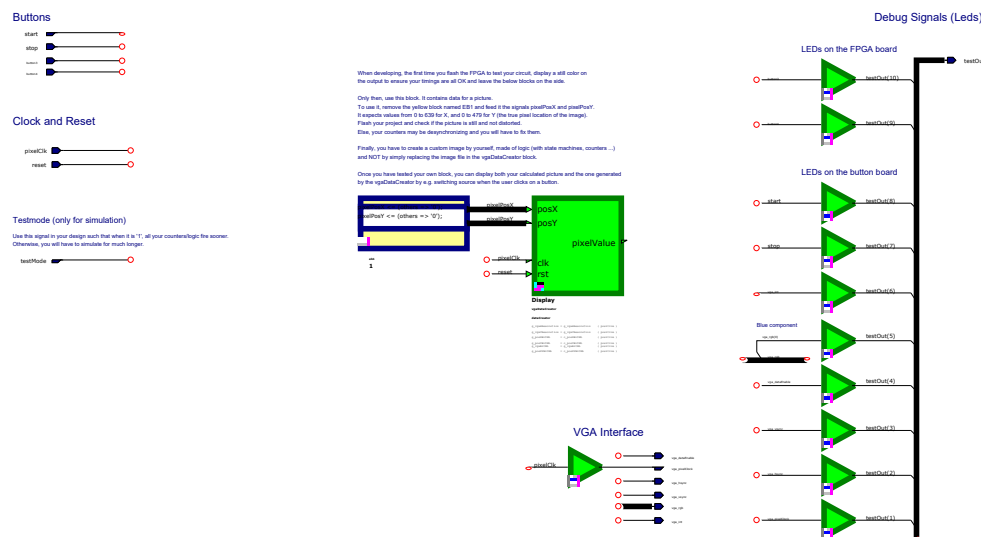


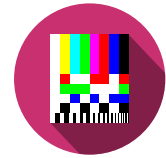
Abbildung 4 - Leere Toplevel Schaltung



Das Signal **testOut** ermöglicht es, die auf dem **Abschnitt 3.2** vorhandenen LEDs ein- und auszuschalten.



Der bereits vorhandene Block **vgaDataCreator** ermöglicht es, eine Pixelfarbe entsprechend der vorgegebenen X- und Y-Positionen (von 0 bis 639 für X, 0 bis 479 für Y) zu liefern, um ein Bild anzuzeigen. Er ermöglicht die grobe Erkennung eines Synchronisationsfehlers (d. h. das Bild sieht verzerrt, wellig, abgeschnitten ... aus). **Dieser Block wird zu Testzwecken zur Verfügung gestellt und ersetzt nicht die eigentliche Anforderung, ein benutzerdefiniertes Bild anzuzeigen.**



## 2.3 VGA Timing (Beispiel)

In den Abbildungen [Abbildung 6](#) und [Abbildung 5](#) werden die VGA Timings für die Auflösung 640px x 480px @ 60Hz dargestellt.

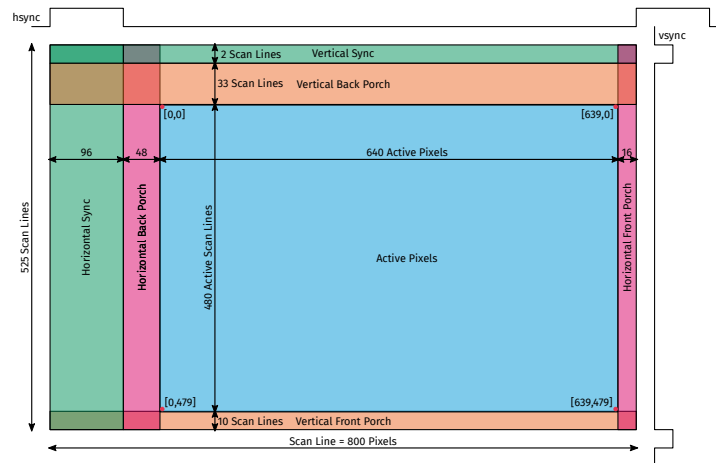


Abbildung 5 - VGA Video Dekodierung

Das Bild wird zeilenweise aufgebaut, beginnend in der oberen linken Ecke:

- **vga\_hsync** und **vga\_vsync** zeigen an, ob eine neue Zeile (**hsync**) oder eine neue Seite (**vsync**) beginnt und müssen für eine bestimmte Zeit gehalten werden.
- Die Zeit für ein Pixel ist ein Zyklus von **vga\_pixelClock**, dessen Frequenz bereits nach dem 640px x 480px @ 60Hz Standard eingestellt ist.
- **vga\_dataEnable** zeigt an, ob das Datensignal aktiv ist und gelesen werden kann.
- Der **vga\_rgb** Vektor enthält die Daten für ein Pixel, mit 1 Bit Tiefe pro Farbe.  
**vga\_int** kann parallel aktiviert werden, wodurch hellere Farben angezeigt werden.



Die zeitlichen Bedingungen für die Signale **vga\_hsync** sowie **vga\_vsync** sind genau zu verstehen und zu beachten. Mehr Infos können in der [Video Electronics Standards Association](#) (VESA) Spezifikation [1] sowie auf der TinyVGA Webseite [2] sowie gefunden werden.

Eine zeitliche Ansicht der VGA Signale ist in [Abbildung 6](#) dargestellt:

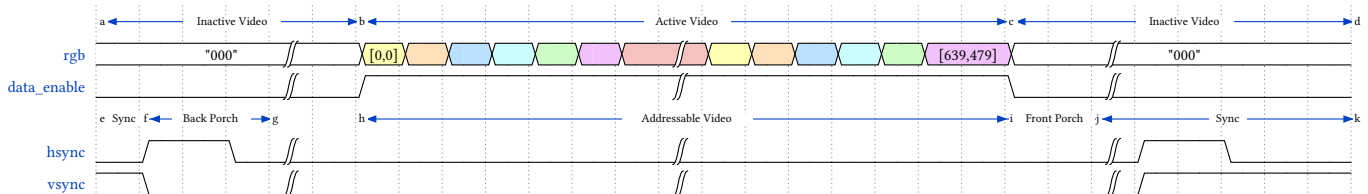
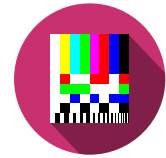


Abbildung 6 - Zeitliche Abfolge der VGA Signale (**hsync** und **vsync** vermischt)



Ausserhalb des Bereichs **Active Pixels** muss der RGB-Wert zu jeder Zeit 0 sein.



Bei einem Bild von 640px x 480px werden rein theoretisch 800px x 525px übertragen. Die zusätzlichen Pixel sind für das vertikale und horizontale **front porch** und **back porch** notwendig. Diese sind dazu da, einem alten CRT Monitor genug Zeit einzuräumen damit der Elektronenstrahl seine Position zwischen den Linien und Seiten neu positionieren kann.

Ein Beispiel für die Auflösung 1920px x 1440px @ 60Hz ist in der Listing [Tabelle 1](#) ersichtlich:

```

Name          1920x1440 @ 60Hz
Aspect Ratio   4:3
-----
Pixel Clock    234    MHz
Pixel Time     4.27   ns
Horizontal freq. 90    kHz
Line Time      11.11  µs
Vertical freq.  60    Hz
Frame Time     16.66  ms
-----

Horizontal Timings
-----
Visible Area    1920
Front Porch     128
Sync Width      208
Back Porch      344
Total (blanks)  672
Total (all)     2600
Sync Polarity   neg
-----

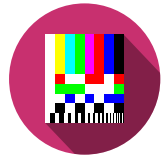
Vertical Timings
-----
Visible Area    1440
Front Porch      1
Sync Width       3
Back Porch       56
Total (blanks)   60
Total (all)     1500
Sync Polarity    pos
-----

Active Pixels
-----
Visible Area    2,764,800
  
```

Tabelle 1 - [VGA](#) Konfiguration für 1920px x 1440px @ 60Hz



Die obigen Abbildungen sind Beispiele, es liegt an den Studenten diese zu verstehen und an die geforderte Auflösung anzupassen.



## 2.4 HDL-Designer Projekt

Ein vordefiniertes HDL-Designer Projekt kann im [Cyberlearn](#) oder [Git](#) heruntergeladen werden. Die Dateistruktur des Projektes sieht folgendermassen aus:

```
did_display
+--Board/          # Project and files for programming the FPGA
|  +--concat/      # Complete VHDL file including PIN-UCF file
|  +--ise/          # Xilinx ISE project
+--Display/         # Library for the components of the student solution
+--Display_test/    # Library for the simulation testbenches
+--doc/             # Folder with additional documents relevant to the project
|  +--Board/        # All schematics of the hardware boards
|  +--Components/   # All data sheets of hardware components
+--img/             # Pictures
+--Libs/            # External libraries which can be used e.g. gates, io, sequential
+--Prefs/           # HDL-Designer settings
+--Scripts/         # HDL-Designer scripts
+--Simulation/      # Modelsim simulation files
+--Tools/           # Specific tools, like a picture to BRAM translator
```

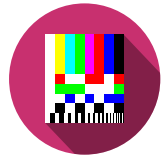


Der Pfad des Projektordners darf keine Leerzeichen enthalten.



Im Projektordner **doc/** können viele wichtige Informationen gefunden werden. Datenblätter, Projektbewertung sowie Hilfsdokumente für HDL-Designer um nur einige zu nennen.





## 3 Komponenten

Das System besteht aus drei verschiedenen Hardwareplatinen, die in der Abbildung [Abbildung 1](#) zu sehen sind.

- Ein Entwicklungsboard [FPGA](#), siehe Abbildung [Abbildung 7](#).
- Eine Steuerkarte mit 4 Tasten und 8 [LED](#), siehe Abbildung [Abbildung 8](#).
- Ein [PMod - DVI](#) Modul um den Bildschirm über eine [HDMI](#) Schnittstelle an das System anzuschliessen, siehe Abbildung [Abbildung 9](#).

### 3.1 FPGA -Platine

Auf der EBS3-Karte erzeugt der verwendete Oszillator ein Taktsignal (**clock**) mit einer Frequenz von  $f_{\text{clk}} = 100 \text{ MHz}$ . Dieser Takt wird **lcdClock** genannt und ist **nur** für den dedizierten LCD-Block der Platine [Knöpfe - LCD, Kapitel 3.2](#) vorgesehen (*derzeit nicht verwendet*).

Dieser Takt wird auch durch den PLL auf  $f_{\text{clk}} = 25 \text{ MHz}$  reduziert, genannt **pixelClk**. Dieser sollte für die Entwicklung Ihres Schaltkreises verwendet werden.



Abbildung 7 - EBS3 [FPGA](#) Platine [3]

### 3.2 Knöpfe und [LED](#)

Die Platine mit den Knöpfen und [LED](#) [4] wird an die [FPGA](#) Platine angeschlossen. Sie hat 4 Tasten und 8 [LED](#), die im Design verwendet werden können. Falls gewünscht kann diese Platine mit einer [Liquid Crystal Display \(LCD\)](#) Anzeige ausgestattet werden [5], [6] (auf Anfrage erhältlich).

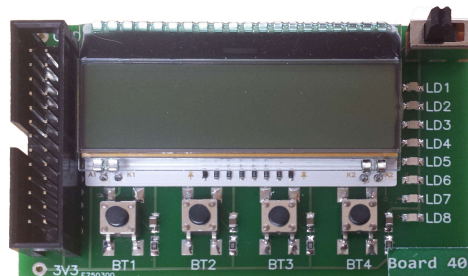
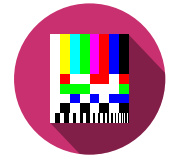


Abbildung 8 - Knöpfe-[LED](#) - [LCD](#) Platine [4]



### 3.3 PMod - DVI Modul

Das **PMod** modul **VGA** nach **DVI** Konvertiert die **VGA** Signale to **TMDS** Signalen. Diese erlaubt es einen **HDMI** Monitor an das System anzuschliessen.

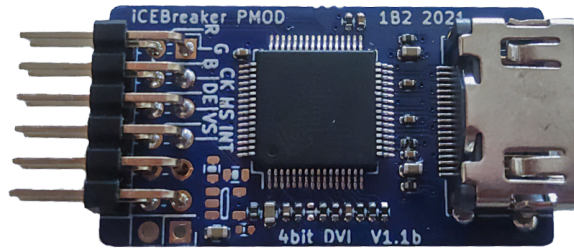


Abbildung 9 - PMod - DVI Modul

Das Blockschaltbild des Chips Texas Instrument TFP410 [7] kann in der Abbildung **Abbildung 10** entnommen werden.



Studiert das Datasheet [7] sowie das Schema des **PMod** -Moduls genau [8].

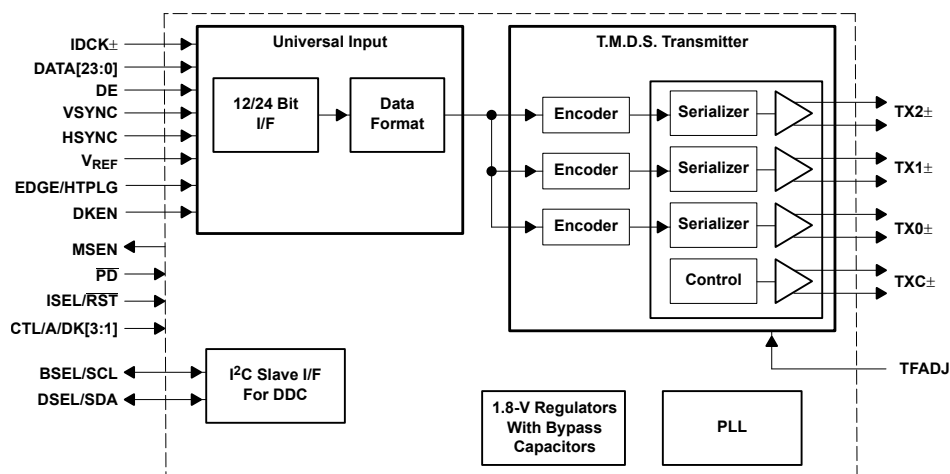
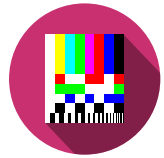


Abbildung 10 - Blockschaltbild des **PMod** -Chips TI TFP410 [7]



## 4 | Bewertung

Im Ordner **doc/** zeigt die Datei **evaluation-bewertung-riscv.pdf** das detaillierte Bewertungsschema von [Tabelle 2](#).

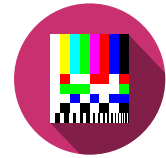
Die Schlussnote beinhaltet den Bericht, den Code sowie eine Präsentation eurerseits des Systems.

Evaluierte Aspekte	Punkte
<b>Bericht</b>	<b>55</b>
Einleitung	3
Spezifikation	5
Entwurf	20
Verifizierung und Validation	10
Integration	9
Schlussfolgerung	3
Formale Aspekte des Berichtes	5
<b>Funktionalität der Schaltung</b>	<b>30</b>
<b>Qualität der Lösung</b>	<b>10</b>
<b>Präsentation</b>	<b>10</b>
<b>Total</b>	<b>105</b>

Tabelle 2 - Bewertungsraster



Das Bewertungsraster gibt bereits Hinweise über die Struktur des Berichtes. Für einen guten Bericht konsultieren Sie das Dokument **Wie verfasst man einen Projektbericht?** [\[9\]](#).



## 5 | Erste Schritte

Um mit dem Projekt zu beginnen, kann folgendermassen vorgehen werden:

- Lest die obigen Spezifikationen und Informationen genau durch.
- Stöbert durch die Dokumente im Ordner **doc/** eures Projektes.
- Analysiert im Detail die Blöcke welche bereits vorhanden bzw. vorgegeben sind.
- Entwickelt ein detailliertes Blockdiagramm. Die Signale und deren Funktionen solltet Ihr erklären können.
- Implementierung und Simulation der verschieden Blöcken.
- Bestätigt die Funktionsweise der gegebenen Hardware mit dem vorinstallierten Programm.
- Testen der Lösung in der Simulation und finden etwaiger Fehler 🐛.

### 5.1 Vorgehensweise

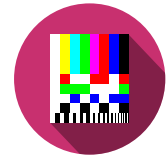
Um die Anzahl der Fehler, die gleichzeitig auftreten, zu minimieren, wird empfohlen, wie folgt vorzugehen:

1. Entwickeln Sie ein Blockdiagramm, das alle Ihre Eingänge, Ausgänge und Funktionen berücksichtigt, die Sie implementieren möchten, ohne diese Blöcke zu implementieren (lassen Sie die Signale auf '0'). Lesen Sie die folgenden Punkte, bevor Sie beginnen.
2. Implementieren Sie das strikte Minimum, um eine einzige Farbe auf dem Bildschirm anzuzeigen, ohne die Tasten und/oder die Position des angezeigten Pixels zu berücksichtigen. Denken Sie daran, dass die Farbe nur im Bereich **Active Pixels** aktiv sein sollte, ansonsten schwarz ("000").  
*Lassen Sie die Blöcke **vgaDataCreator** und **eb1** beiseite.*  
Dies ermöglicht es, die Timings der Signale **vga\_pixelClock**, **vga\_hsync**, **vga\_vsync** und **vga\_dataEnable** zu überprüfen.
3. Verwenden Sie nun den Block **vgaDataCreator**, um ein vorab aufgenommenes Bild auf dem Bildschirm anzuzeigen. Entfernen Sie den gelben Block **eb1** (dieser hält die Signale **pixelPosX** und **pixelPosY** auf 0). Geben Sie ihm zur richtigen Zeit die Position des Pixels auf dem Bild, wobei zu beachten ist, dass er einen Takt benötigt, um die Farbdaten des angezeigten Pixels auszugeben.
  - **pixelPosX** ist ein **unsigned** von **10 Bits**, das Eingabewerte von 0 bis 639 akzeptiert.
  - **pixelPosY** ist ein **unsigned** von **9 Bits**, das Eingabewerte von 0 bis 479 akzeptiert.
  - Das Referenzbild [Abbildung 11](#) ist unter **doc/image\_in\_memory.bmp** verfügbar, um zu überprüfen, ob die Anzeige korrekt ist (seltsame Pixel aufgrund der JPEG-Kompression und der 1BPP-Konvertierung):



Abbildung 11 - Bild auf dem Bildschirm aus vgaDatacreator

Das Anzeigen eines Bildes ermöglicht es, Fehler in der Synchronisation, Drift, Genauigkeit



der Positionszähler ... zu erkennen, je nachdem, ob das Bild verzerrt, wellig, abgeschnitten, in Bewegung ist ...

4. Implementieren Sie dann einen Block, der ein Bild berechnet, das je nach Position des Pixels auf dem Bildschirm berechnet wird. Das Bild sollte alle möglichen Farbkombinationen enthalten, die angezeigt werden können, siehe [Abbildung 2](#).
5. Fügen Sie schliesslich die Tasten und die zugehörigen Funktionen hinzu. Mindestens sollte es möglich sein, die Anzeige zu stoppen (schwarzes Bild) und wieder zu starten (Testbild).

Zu jedem Zeitpunkt können bestimmte Signale auf den **LED** angezeigt werden, um das Debuggen zu erleichtern. Verwenden Sie dazu das Signal **testOut**, das mit dem **LED** -Modul verbunden ist. Es ist möglich, diese mit Oszilloskopen oder Logikanalysatoren zu messen, um bei der Fehlererkennung zu helfen.

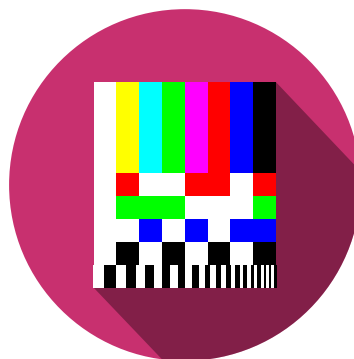
## 5.2 Tips

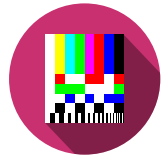
Anbei noch einige zusätzlichen Tips um Probleme und Zeitverlust zu vermeiden:

- Teilen Sie das Problem in verschiedene Blöcke auf, benutzt hierzu das leere Toplevel Dokument. Es ist ein ausgeglichener Mix zwischen Anzahl Komponenten und Komponentengrösse empfohlen.
- Analysieren Sie die verschiedenen Ein- und Ausgangssignale, ihre Arten, ihre Grössen ... Sollte man sich teilweise auf die Datenblätter und die Dokumentation beziehen.
- Beachten Sie bei der Erstellung des Systems das DiD Kapitel „Methodologie für die Entwicklung von digitalen Schaltungen (MET)“ [\[10\]](#)
- Halten Sie sich an die vorgeschlagene inkrementelle Vorgehensweise. Benutzen Sie die Tests so früh wie möglich.
- Speichern und dokumentieren Sie Ihre Zwischenschritte. Nicht funktionierende Architekturen, grundlegende Codes zum Testen der Architektur ... sind allesamt Material, das dem Bericht hinzugefügt werden kann.



Vergessen Sie nicht Spass zu haben.





# Glossar

*bpp* – bit per pixel [4](#)

*DVI* – Digital Visual Interface [2](#), [4](#), [9](#), [10](#)

*FPGA* – Field Programmable Gate Array [2](#), [4](#), [5](#), [9](#)

*HDMI* – High Definition Multimedia Interface [4](#), [5](#), [9](#), [10](#)

*LCD* – Liquid Crystal Display [9](#)

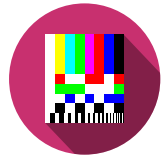
*LED* – Light Emitting Diode [2](#), [5](#), [9](#), [13](#)

*PMod* – Peripheral Module [2](#), [5](#), [9](#), [10](#)

*TMDS* – Transition Minimized Differential Signaling [5](#), [10](#)

*VESA* – Video Electronics Standards Association} [6](#)

*VGA* – Video Graphics Array [2](#), [3](#), [5](#), [6](#), [7](#), [10](#)



# Literatur

- [1] VESA, „VESA and Industry Standards and Guidelines for Computer Display Monitor Timing (DMT)“. [Online]. Verfügbar unter: <https://glenwing.github.io/docs/VESA-DMT-1.13.pdf>
- [2] TinyVGA, „VGA Signal Timing“. Zugegriffen: 7. Juli 2022. [Online]. Verfügbar unter: <http://www.tinyvga.com/vga-timing>
- [3] A. Amand und S. Zahno, „FPGA-EBS3 Electornic Technical Documentation“. 2022.
- [4] Silvan Zahno, „Schematic: Parallelport HEB LCD V2“. 2014.
- [5] Sitronix, „Datasheet Sitronix ST7565R 65x1232 Dot Matrix LCD Controller/Driver“. 2006.
- [6] Electronic Assembly, „Datasheet: DOGM Graphics Series 132x32 Dots“. 2005.
- [7] T. Instrument, „Datasheet Digital Transmitter Texas Insturment TFP410“. 2014.
- [8] E. Piotr, „Schematic: iCEBreaker PMOD 4bit DVI“. 2018.
- [9] Christophe Bianchi, François Corthay, und Silvan Zahno, „Comment Rédiger Un Rapport de Projet?“. 2021.
- [10] François Corthay, Silvan Zahno, und Christophe Bianchi, „Methodologie Für Die Entwicklung von Digitalen Schaltungen“. 2021.