OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Computer Architektur

Sommersemester 2025

Projektzielsetzung

Version 1.0

Teammitglieder:

Fabian Becker _____

Jendrik Jürgens _____

Nicolas Koch _____

Franz Krempl _____

Daniel Sowada _____

Michael Specht _____

Professor:
Dr. Daniel Münch

Abgabedatum:
30.04.2025 11:30 Uhr

26. April 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Projektzielsetzung

Projekttitel:
Integriertes Demosystem für Sensor- und Displayansteuerung auf FPGA-Basis
Teammitglieder:
Fabian Becker, Jendrik Jürgens, Nicolas Koch, Franz Krempl, Daniel Sowada, Michael Specht
Projektbeschreibung:
Ziel des Projekts ist die Entwicklung und Integration eines funktionsfähigen Demosystems, das zwei getrennte Komponenten – den Sonarsensor (PMOD MAXSONAR) und das LCD-Display (PMOD CLP) – auf einer gemeinsamen FPGA-Plattform (Digilent Arty A7-100) vereint. Die Messwerte des Sensors sollen in Echtzeit auf dem Display ausgegeben werden. Dazu werden eigene IP-Cores für beide Komponenten entworfen, implementiert, getestet und in die Hardwareplattform integriert.

Geplante Arbeitspakete und Zuständigkeiten:

1. Systemintegration bestehender Demosysteme
   - Integration der Software- und Hardware-Demoprojekte zu einem Gesamtsystem (HW & SW)

     Zuständig: Alle Teammitglieder
2. Entwicklung Custom IP für Sensor (UART / MAXSONAR)
   - Konzept (Blockdiagramm, Registermapping) auf Basis der `tut_int` Vorlage
   - Reduzierte Funktionalität orientiert an Xilinx UART Lite IP

     Zuständig: Fabian Becker, Nicolas Koch
3. Entwicklung Custom IP für Display (PMOD CLP, Nibble-Mode)
   - Umsetzung von Timinganforderungen in Hardware
   - Anzeige von Daten auf LCD über FSM und spezifizierte Steuerregister

     Zuständig: Jendrik Jürgens, Michael Specht
4. Erstellung und Test der IP-Testbenches (Core und AXI)
   - Entwicklung mit Fokus auf Polling (Interrupt optional bei Zeitreserve)
   - Simulation und Validierung des Verhaltens

     Zuständig: Alle Teammitglieder
5. Treibersoftware
   - Schreiben von Treibern für die beiden IPs unter Verwendung der SW-Templates von `tut_int`
   - SW-basierte Initialisierung und Datentransfer

Zuständig: Franz Krempl, Daniel Sowada

6. Integration in vollständiges Demosystem
   - Zusammenführung aller Komponenten zu einem lauffähigen System
   - Validierung auf der realen Hardwareplattform

   Zuständig: Alle Teammitglieder

Systemblockbild:

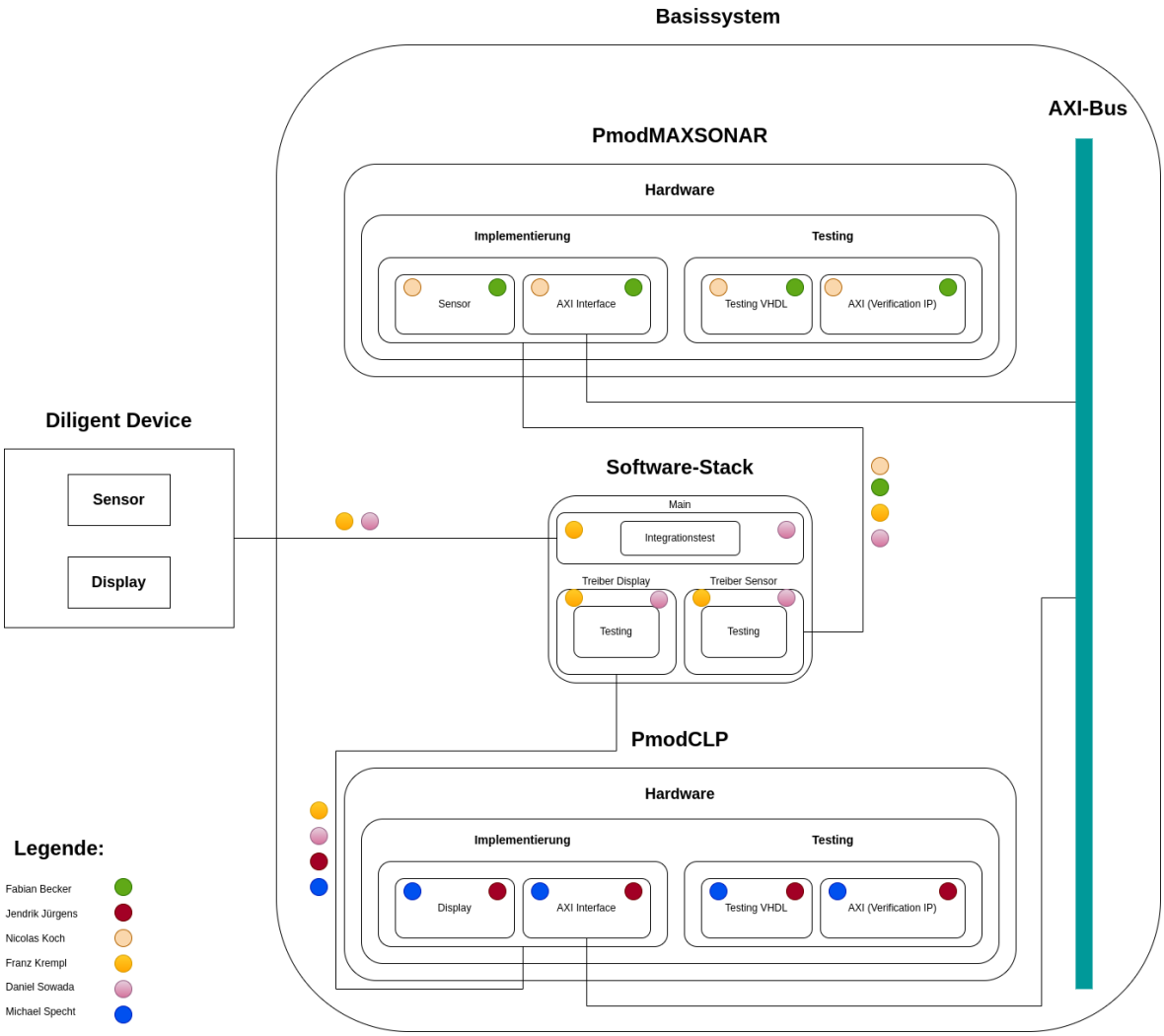

Abbildung 1.1: Systemblockbild

- **Exakte Timings für das Display (PmodCLP) in VHDL:**
  Die Ansteuerung im Nibble-Mode erfordert die präzise Umsetzung aller Timingbedingungen (Setup, Hold, Enable) in einer FSM, da keine automatische Verzögerung durch die CPU gegeben ist.

- **Entwicklung AXI4-Lite-kompatibler IP-Cores:**
  Für Sensor und Display werden eigenständige IPs mit klar strukturiertem Registermapping und AXI-Anbindung erstellt, basierend auf einer gemeinsamen IP-Vorlage.

- **Zuverlässiger Datenfluss zwischen Sensor, CPU und Display:**
  Die Software muss synchronisiert mit den IPs arbeiten, um Sensordaten korrekt auszulesen und anzuzeigen – inklusive Fehlerbehandlung und Statusabfrage.

- **Simulation und Verifikation:**
  Funktion und Schnittstellen werden über VHDL-Testbenches (Core + AXI) geprüft, um Designfehler frühzeitig zu erkennen.

Ziel:
Ein lauffähiges Demosystem mit eigenentwickelten, erweiterbaren IP-Cores, das Messwerte des Sonar Sensors auf einem LCD-Display ausgibt – mit strukturierter Dokumentation, Tests und einer funktionalen Ergebnisvorführung.

2. PmodCLP

Der PmodCLP besteht aus einem Samsung KS0066 LCD Controller und einem Sunlike LCD Panel, worüber Informationen dargestellt werden können lcd-desc. Es ist möglich 32 Positionen auf dem 16x2 LCD Panel zu nutzen. Pro Position werden die Zeichen dabei mit einer Auflösung von 5x8 angezeigt.

Das System besteht im Wesentlichen aus drei Komponenten. Der character-generator ROM (CGROM) hält 192 vordefinierte Zeichen, darunter 93 ASCII Charaktere. Anschaulich gesehen können die Zeichen über eine matrixartige Struktur indiziert werden, welche im Datenblatt festgelegt ist. Neben den nicht-volatilen Daten im CGROM ist es möglich bis zu 8 eigene Zeichen volatil im character-generator RAM (CGRAM) zu halten. Um nun Zeichen aus diesen beiden Repositories auf dem Panel anzeigen zu können, gibt es den data RAM (DDRAM). Hier können bis zu 80 Zeichencodes gespeichert werden. Er fungiert als Indexspeicher für Daten innerhalb des CGROM oder CGRAM. Wird ein Index aus der matrixartigen Struktur in den DDRAM geladen, erscheint das entsprechende Zeichen auf dem Display.

Das Display selbst verfügt über 2 Zeilen á 16 Positionen. Insgesamt stehen jedoch nicht 32 Speicherplätze zur Verfügung, sondern 39, um beispielsweise Scrolling zu verwenden.

Wichtige Schnittstellen des Samsung KS0066 LCD Controller sind

- *DB4-DB7*: Datenbits im Nibble-Mode zur Codierung von Befehlen/Zeichen
- *RS (Register Select)*: High für Daten, Low für Instruktionen
- *RW (Read/Write)*: High = Read, Low = Write
- *E (Enable)*: High für Read, Falling Edge für Write

Um diese nutzen zu können wird folgendes Mapping auf dem FPGA hinterlegt:

```
set_property -dict {PACKAGE_PIN D13 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[4]}];
    #db04
set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[5]}];
    #db05
set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[6]}];
    #db06
set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[7]}];
    #db07
set_property -dict {PACKAGE_PIN E2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[0]}];
    #lcd_rs
set_property -dict {PACKAGE_PIN D2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[1]}];
    #lcd_rw
set_property -dict {PACKAGE_PIN H2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[2]}];
    #lcd_e
```

Listing 2.1: Pin-Zuordnung im Constraints-File

Die Zuordung in Aufzählung 2.1 greift auf zwei Header des PmodCLP-Boards zurück. Die Daten-Bits db04-db07 sind an die untere Hälfte des Headers J1 gebunden. Die Steuersignale Register Select, Read/Write und Enable hingegen an Header J2.

2.1. Ansatz Custom IP

Im ersten Schritt soll die IP funktional fertiggestellt werden. Sie soll dabei mittels Polling eingesetzt werden. Sobald die Funktionalität des Gesamtsystems vollumfänglich gegeben ist, wird anstatt Polling über eine AXI Schnittstelle kommuniziert.

Das Projektteam hat sich auf folgenden Entwurf geeinigt:

- *Submodul 1*: Finite-State-Machine (FSM)
  Mittels einer FSM wird aus den Registern, welche von Microprozessor gesetzt werden können, der Befehl interpretiert.
- *Submodul 2*: Init/Reset
  Für die Initialisierung bzw. Reset müssen diverse Zeitbedinungen eingehalten werden.
  Die Graphik 2.1 stellt die verschiedenen Schritte während des Starts dar. Erst nach 21.594ms

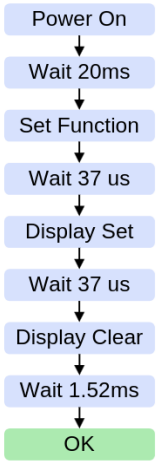

Abbildung 2.1: Startup Sequence

  ist es möglich, Daten in den DDRAM zu schreiben.
- *Submodul 3*: ASCII zu Zeichencode CGROM/CGRAM
- *Submodul 4*: DDRAM Adressierung
- *Submodul 4*: PmodCLP Kommunikation

TODO

- Set Function: Busbreite, Zeilenanzahl, Zeichenmuster konfigurieren
- Display Set: Display anschalten, Cursor an-/ausschalten, Cursor-Blinken konfigurieren
- Entry Mode Set: Adress-Inkrement/ -Dekrement, Display-Shift aktivieren/deaktivieren

Danach können Daten ins DDRAM geschrieben werden, um Informationen anzuzeigen.

2.2. Registermapping

2.2.1. I/Os

Signal Name	I/O	Initial State	Description
ap_clk(s00_axi_aclk)	I	NA	AXI Clock
ap_rst_n (s00_axi_aresetn)	I	NA	AXI Reset, active-Low
s_axi_control* (s00_axi*)	NA	NA	AXI4-Lite Slave Interface signals
interrupt	O	0x0	Indicates that the condition for an interrupt on this timer has occurred. (timer rolled over) 0 = No interrupt has occurred 1 = Interrupt has occurred
o_t0_out	O	0x0	Timer generated blinking for led (timer rolled over condition toggles signal)

Tabelle 2.1: PmodCLP Überblick I/O

2.2.2. Registerbereich

Address Offset	Register Name	Description
0x00	GCSR	General/Global Control and Status Register
0x04	GIER	Global Interrupt Enable Register
0x08	IPIER	IP Interrupt Enable Register
0x0C	IPISR	IP Interrupt Status Register (Interrupt Pending)
0x10	IDR	ID Register
0x14	VERR	Version Register
0x18	SCSR0	Special Control and Status Register
0x1C	CR0	Counter Register
0x20	LR0	Load Register
0x24	reserved	reserved

Tabelle 2.2: PmodCLP Registerbereich

2.2.3. Registerbeschreibung (MSB bit31 LSB bit0)

0x00 GCSR

Bit	Name	Access Type	Reset Value	Description

0	ap_start	R/W	0	Asserted when the kernel can start processing data. Cleared on handshake with ap_done being asserted.
1	ap_done	R	0	Asserted when the kernel has completed operation. Cleared on read.
2	ap_idle	R	0	Asserted when the kernel is idle.
3	reserved (ap_ready)	R	0	Asserted by the kernel when it is ready to accept the new data (used only by AP_CTRL_CHAIN))
4	reserved (ap_continue)	R/W	0	Asserted by the XRT to allow kernel keep running (used only by AP_CTRL_CHAIN)
5:6	reserved			
7	auto_restart	R/W	0	Used to enable automatic kernel restart. This bit determines whether the counter reloads the load register value and continues running or holds at the termination value. 0 = Hold counter 1 = Reload load register value
31:8	reserved			

Tabelle 2.3: PmodCLP Registerbeschreibung

0x04 GIER

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
0	gie	R/W	0	When asserted, along with the IP Interrupt Enable bit, the interrupt is enabled.
31:1	reserved			

Tabelle 2.4: PmodCLP Register GIER Details

0x08 IPIER

Bit	**Name**	**Access Type**	**Reset Value**	**Description**

0	ipie	R/W	0	When asserted, along with the Global Interrupt Enable bit, the interrupt is enabled. (default: uses the internal ap_done signal to trigger an interrupt)
31:1	reserved			

Tabelle 2.5: PmodCLP Register IPIER Details

0x0C IPISR

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
0	ipis	R/W	0	Toggle on write. (write 1 to clear(W1C))
31:1	reserved			

Tabelle 2.6: PmodCLP Register IPISR Details

0x10 IDR

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
31:0	ID	R	0x8001DEEF	Distinct ID

Tabelle 2.7: PmodCLP Register IDR Details

0x14 VERR

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
31:0	VER	R	0x80001000	Version

Tabelle 2.8: PmodCLP Register VERR Details

0x18 SCSR0

Bit	**Name**	**Access Type**	**Reset Value**	**Description**

0	reserved (ENALL)	R/W	0	ap_start is used Enable All Timers 0 = No effect on timers 1 = Enable all timers (counters run) This bit is mirrored in all control/status registers and is used to enable all counters simultaneously. Writing a 1 to this bit sets ENALL, EN0, and EN1. Writing a 0 to this register clears ENALL but has no effect on EN0 and EN1.
1	reserved (en0)	R/W	0	ap_start is used Enable Timer 0 0 = Disable timer (counter halts) 1 = Enable timer (counter runs)
2	ent0_out	R/W	0	Enable external pin t0_out 0 = Disable 1 = Enable
3	reserved			
4	load0	R/W	0	Load Timer 0 0 = No load 1 = Loads timer with value in LR0 Setting this bit loads counter register (CR0) with a specified value in the load register (LR0). This bit prevents the running of the timer/counter; hence, this should be cleared alongside setting Enable Timer/Counter (EN0) bit.
5	ud0	R/W	0	Up/Down Count Timer 0 0 = Timer functions as up counter 1 = Timer functions as down counter
6	reserved			
7	reserved			
8	reset_ip	R/W	0	Stops the whole IP and resets all values
9	freeze_ip	R/W	0	Stops the whole IP but does not reset the values (useful for debugging)

10	reserved			
11	reserved			
31:12	reserved			

Tabelle 2.9: PmodCLP Register SCSR0 Details

0x1C CR0

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
31:0	CR0	R	0x0	Counter Register 0

Tabelle 2.10: PmodCLP Register CR0 Details

0x20 LR0

Bit	**Name**	**Access Type**	**Reset Value**	**Description**
31:0	LR0	R/W	0x0	Load Register 0

Tabelle 2.11: PmodCLP Register LR0 Details

3. Sensor