OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

PMOD MAXSONAR PMOD CLP

Fabian Becker, Jendrik Jürgens, Nicolas Koch, Franz Krempl, Daniel Sowada, Michael Specht

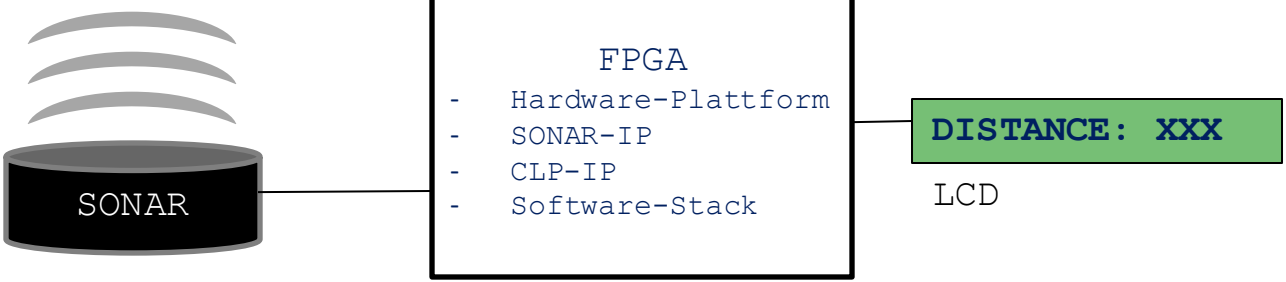
Einleitung/Motivation (Team)

- Vertiefung und Erweiterung des Wissens aus dem Modul "Digital Design" durch praktische Anwendung in einem Projekt
- Teambasierte Projektarbeit: von der Konzeptionierung über Implementation bis zur Integration und Dokumentation
- Enge Betreuung durch den Professor mit frühzeitiger Unterstützung bei Fragen und Problemen
- Seminaristischer Ansatz mit hohem Praxisanteil und Verwendung der Industrieapplikationen **AMD Vivado™** und **AMD Vitis™**
- Technische Kompetenzen in Timing-Diagrammen, Datenblättern und Entwicklung eigener IP's (Intellectual Property) vertiefen/erwerben
- AXI4-Lite als Schnittstellenprotokoll für On-Chip Kommunikation
- Programmiersprachen C, VHDL und SystemVerilog als zentrale Werkzeuge im Kurs

Aufgabenstellung/Zielsetzung (Team)

- Projekt: Pmod-MaxSonar/Pmod-CLP
- Ziel: Gemessene Distanz eines Sonar-Sensors auf einem LCD-Display anzeigen.
- Aufgaben:
  - IP für Sonar entwickeln, mit der man per Register interagieren kann
  - IP für Display entwickeln, mit der man per Register interagieren kann
  - Kombinieren der Projekte in einer Hardware
  - Treibersoftware zur einfachen Bedienung der IPs schreiben

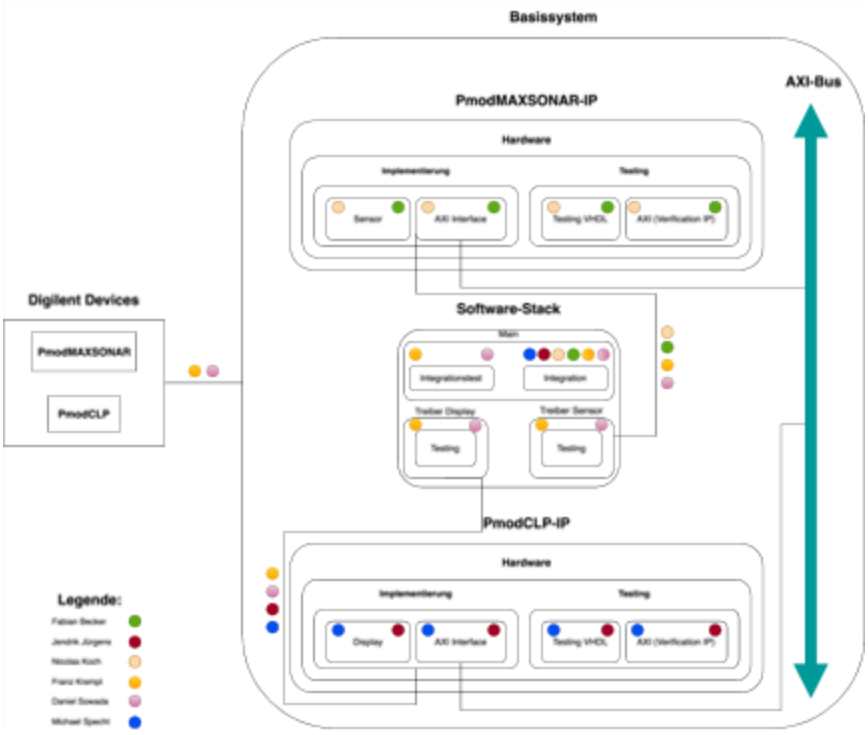
Aufbau



Lösungsansatz/Konzept (Team)

- Unterteilung in 3 Teams
  - Hardware – PMOD MAXSONAR (Becker, Koch); kurz **AS** (**A**XI-**S**onar)
    - IP-Core mit Submodulen: Startup & Calibration Timer, UART Receiver, ASCII Decoder
    - Anbindung an AXI-Bus
    - Testbenches für Submodule + AXI
  - Hardware – PMOD CLP (Jürgens, Specht); kurz **AD** (**A**XI-**D**isplay)
    - IP-Core mit Submodulen: Timing Controller, LCD Controller
    - Anbindung an AXI-Bus
    - Testbenches für Submodule + AXI
  - Software – Treiber-Stack (Krempl, Sowada); kurz **SW** (**S**oft**w**are)
    - Registertests
    - Treiber anhand Registermapping
- Codeverwaltung mittels Git

Lösungsansatz/Konzept - Block-Diagramm (Team)



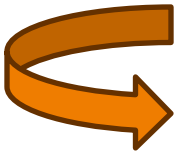
Änderungen:

- IP-Integration in Software-Stack

Lösung/Implementierung - AD - Timing Controller (Jürgens, Specht)

Entwurf/Abwägung:

- 3 verschiedene Ansätze diskutiert:
  - Option 1: Dedizierter Timer für jede Zeit
    - Vorteil: Einfach
    - Nachteil: Code-Verdopplung
  - Option 2: Ständig laufender Timer der mit Schranken (Timestamps) arbeitet
    - Vorteil: Einfach
    - Nachteil: evtl. schwierig saubere Waveform zu bilden, erschwertes Testing
  - Option 3: Variabler Counter, der bei Bedarf gestartet werden kann und sich nach Ablauf der Zeit zurückmeldet
    - Vorteil: variabler Ansatz (DRY – Dont Repeat Yourself, SoC – Separation of Concerns), strikte Trennung --> Übersichtlichkeit
    - Nachteil: Code ist komplexer



Option 3

Lösung/Implementierung - AD - Timing Controller (Jürgens, Specht)

Implementierungsdetails:

- Allgemein
  - Wenn i_delay_time und i_start von Top-Module gesetzt, muss sich Top-Module darum kümmern, i_start im nächsten Takt auf 0 zu ziehen

- States
  - IDLE: Default-State, Timer inaktiv
  - WORK: Zähler wird mit jedem Takt inkrementiert, bis counter = delay_time

- Delay Logik
  - i_delay_time (22-bit vector) wird beim Start gespeichert (Latch)
  - Zähler startet bei 1 wenn i_start = '1'

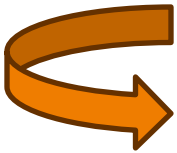
- Output Handling
  - o_done ist für einen Takt auf 1, wenn Timer fertig

Lösung/Implementierung - AD - LCD Controller (Jürgens, Specht)

Entwurf:

- Skeleton von ChatGPT benutzt, adaptiert und Timing Controller instanziert
- Negatives?
  - Es wurden grundlegende Charakteristiken des Timings, wie in den Datenblättern spezifiziert, vernachlässigt
  - Starrer Ansatz
  - Gleicher Code teilweise öfter vorhanden --> Duplizierung
  - Verständnisschwierigkeiten
  - Sehr viel Code: minimalistischer Ansatz bereits ~ 800 Zeilen Code

- Positives?
  - Erkenntnisgewinnung zur Lösung des Problems
  - Grundsätzliche Logiken zum Teil wiederverwendbar



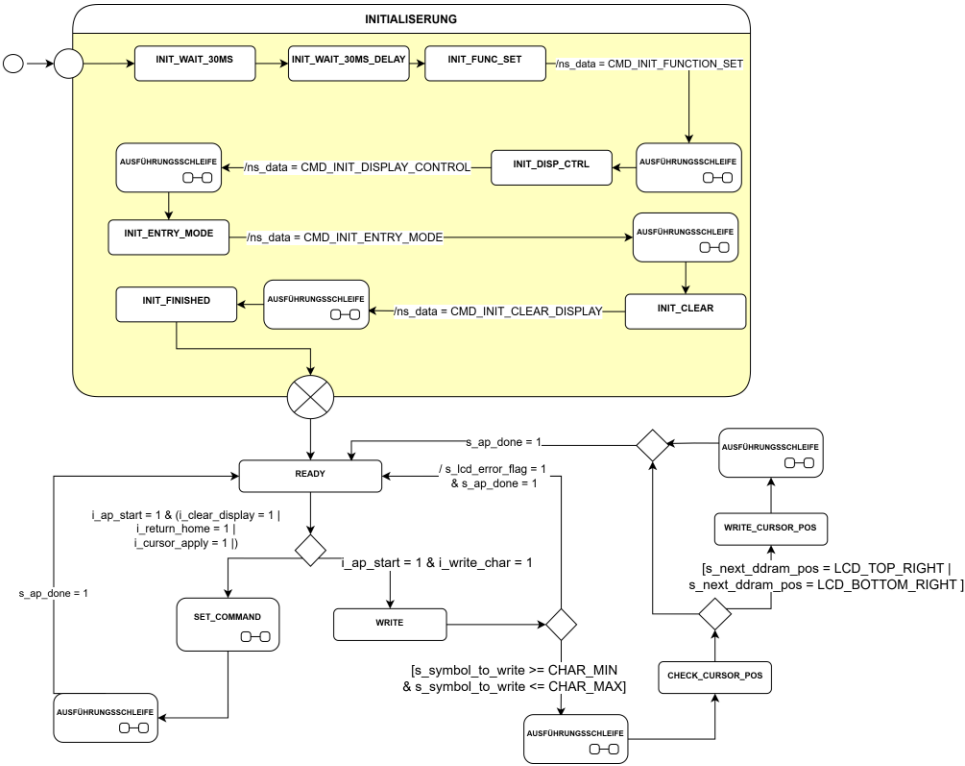
Bisherigen Stand verwerfen und neu beginnen!

ERGO: ca. 20 – 30 h für Problemverständnis investiert ohne Resultat

Lösung/Implementierung - AD - LCD Controller (Jürgens, Specht)

Übersicht:

- Neuer Ansatz soll variabel sein, d. h. Wiederverwenden von Zuständen in der Befehlskette
- Vereinfachte State-Machine des LCD-Controllers
- Ausführungsschleife => zusammengesetzter Zustand (s. Backup)
- SET_COMMAND => zusammengesetzter Zustand (RETURN HOME, CLEAR DISPLAY, CURSOR (BLINK) ON/OFF)



Lösung/Implementierung - AD - LCD Controller (Jürgens, Specht)

Implementierungsdetails:

Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	I/D = '1' for right-moving cursor and address increment; SH = '1' for display shift (direction set by I/D bit).

- DDRAM Position
  - Wird mittels Entry Mode Bit (I/D) automatisch vom Display selbst verwaltet
  - Ausnahme: Line-Wrapping Zeile 1 -> 2 und Zeile 2 -> 1

- Timings
  - Generic Map für variable Timings: optimal für anschließenden AXI-Test

Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address counter AC5 - AC0

- Befehlszustände
  - READY: Wartet auf ap_start und prüft Eingangssignale
  - WRITE: Validiert Zeichen (0x20-0x7F) und schreibt zu LCD
  - CHECK_CURSOR_POS: Prüft Zeilenumbruch (Position 0x0F -> 0x40, 0x4F -> 0x00)
  - WRITE_CURSOR_POS: Setzt DDRAM Adresse (0x80 | position)
  - CLEAR_DISPLAY/RETURN_HOME/DISPLAY_CONTROL: Spezielle Befehle

- 4-Bit Übertragungssequenz
  - SETUP_CONTROL: Daten aufteilen in upper/lower nibble
  - WAIT_SETUP → WAIT_SETUP_DELAY: t_SU Setup Time (60ns)
  - PULSE_ENABLE → PULSE_ENABLE_DELAY: t_W Enable Pulse (450ns)
  - DISABLE_ENABLE → DISABLE_ENABLE_DELAY: t_H Hold Time (30ns)
  - EXEC_DELAY: Command-spezifische Wartezeit (40µs oder 1.64ms)

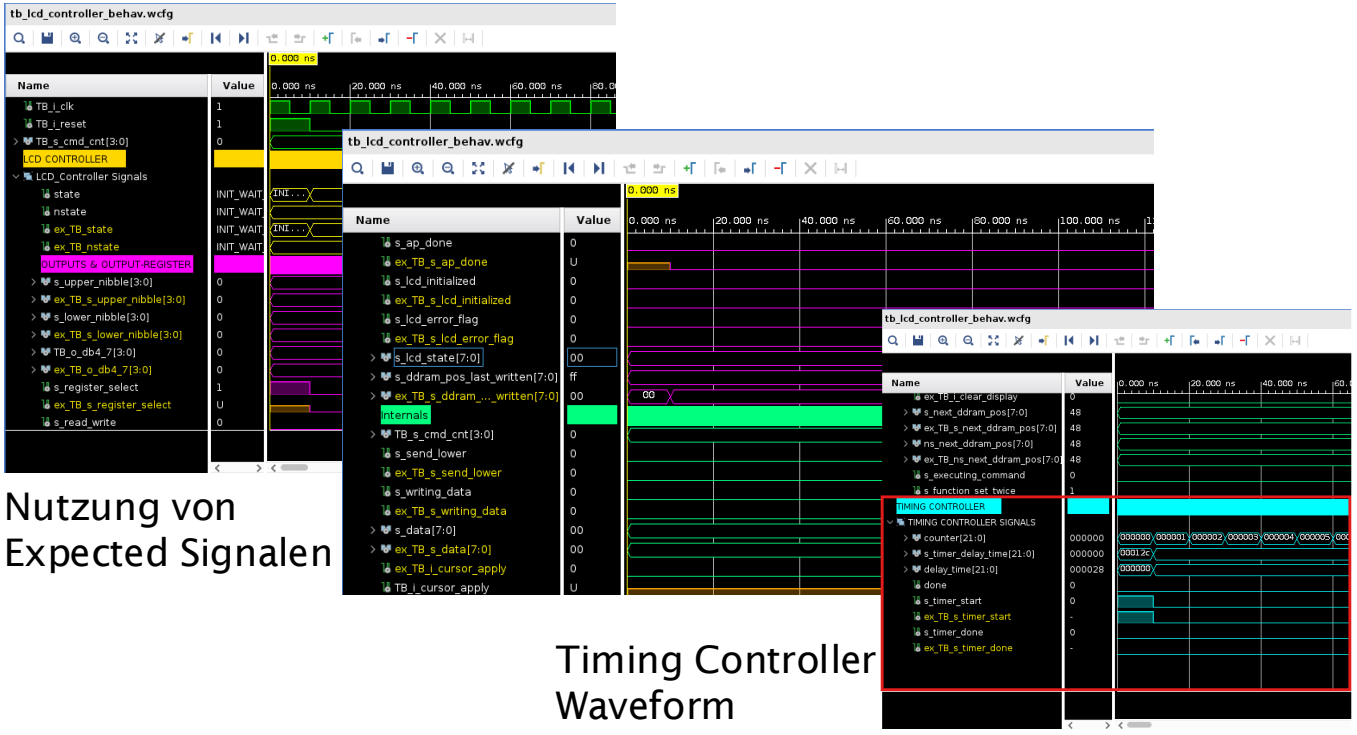
Lösung/Implementierung - AD - LCD Controller (Jürgens, Specht)

- Erstellung Testbench mit Wave-Config => gleicher Ausgangspunkt für jeden

- Initial geplante Testfälle:
  - **Initialisierungsphase** mit hoher Detailtiefe
  - **Erfolgreiches Schreiben**
  - **Fehlerhaftes Schreiben** – "Schlechter Buchstabe"
  - **Cursor Einstellung ändern**
  - **Clear Display Befehl**
  - **String schreiben –** mit Zeilenumbruch (Edge-Case)
  - **Lesen eines Buchstaben** an Stelle x

- Tatsächlich implementierte Testfälle
  - Alle **außer Lesen eines Buchtaben (Keine Implementierung vorhanden)**

Lösung/Implementierung - AD - LCD Controller (Jürgens, Specht)



Nutzung von Expected Signalen

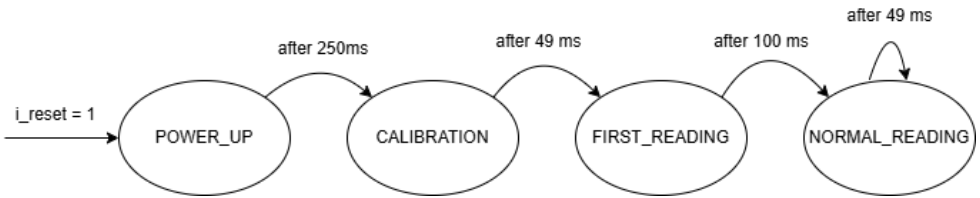
Timing Controller Waveform

Lösung/Implementierung - AS - Übersicht (Becker, Koch)

- Unterteilung der IP in drei Teilmodule
  - Control Timer (Koch)
    - sorgt für Einhaltung von Timing Constraints (Powerup & Calibration Delay)
    - fungiert als Watchdog Timer; meldet Sensor Timeouts
  - UART-Receiver mit Baudrate Generator (Becker & Koch)
    - Generierung von Ticks mit 16x Oversampling bei 9600 Baud
    - Empfang der UART-Daten vom Sensor
  - ASCII Decoder (Becker)
    - wandelt die per UART empfangenen Daten in eine 8-Bit Zahl
    - Fehlerbehandlung
  - Instanziierung der Teilemodule in gemeinsamen Top-Modul
- Verbinden von Top Modul mit AXI-Slave Interface (Becker & Koch)
  - anschließende Verifizierung mittels AXI Verification IP

Lösung/Implementierung - AS – Control Timer (Koch)

- Implementierung per State-Machine
  - nach Reset:
    - 250ms Powerup Delay
    - 49ms Kalibrierung
    - ~100ms Sensor Reading
  - darauffolgende Messungen:
    - 49ms Sensor Reading
- Zusätzliche Funktion als WatchDog Timer
  - Timing für Sensor Reading bekommt etwas Puffer
  - falls ASCII-Decoder nicht vor Timer fertig -> Timeout Fehler



Lösung/Implementierung - AS – UART Reciever (Becker, Koch)

- UART-Parameter (gemäß Datenblatt)
  - 9600 Baud
  - 8 Datenbits
  - 1 Stopbit
- Nutzung von 16-fachem Oversampling -> Bit wird in der Mitte gesamplet
- Implementierung per State-Machine basierend auf [1, S.164-168]
  - Startbit wird zur Synchronisation mit Systemtakt abgetastet
  - 8 Datenbits werden erfasst (Least-Significant-Bit-First)
  - bei fehlendem Stopbit wird ein Framing-Error angezeigt
- Oversampling-Ticks werden von Baudrate Generator erzeugt
  - simpler Clock-Divider
  - wird beim Empfang des Startbits gestartet

Lösung/Implementierung - AS – ASCII-Decoder (Becker)

- Sensor Reading wird in 5 Bytes per UART gesendet
  - Paketstruktur: *Rxxx\r; xxx* = 6 – 255
- Ziel: Ausgabe der Distanz in Zoll als 8-Bit Zahl in Register
- Implementierung mittels State-Machine
  - bei Verstoß gegen die Paketstruktur wird erster Fehler angezeigt
    - Position im Paket
    - Empfangenes Byte, welches den Fehler verursachte
  - wird im Fehlerzustand das Zeichen *R* empfangen
    - Rücksprung in validen Zustand
    - Resynchronisations-Punkt
  - Decoder kann durch Control Timer unterbrochen werden -> Timeout Fehler

Lösung/Implementierung - SW - Übersicht (Krempl, Sowada)

- Ziel:
  - Entwicklung von Treibern in C zur Realisierung einer registerbasierten Kommunikation
  - Mit Hilfe der Treiber die Funktionalitäten der IPs zu einem funktionierenden Projekt kombinieren

- Registertests in Software:
  - Jedes Register mit Einsen beschrieben, gelesen und mit erwarteten Werten verglichen.
    Anschließend wird dasselbe mit Nullen durchgeführt
  - Keine Abweichungen außer bei GCSR-Registern.
    Dieses Verhalten war zu erwarten und ist kein Fehler

Lösung CLP IP - SW - Übersicht (Krempl, Sowada)

- Ziele:
  - Strings auf Display anzeigen
  - Display leeren
  - Cursor aktivieren/deaktivieren

- Umsetzung:
  - Anzeige
    - Write-Char aktivieren
    - Jeden Char wird nacheinander in ein Register geschrieben und IP gestartet
    - Am Ende des Strings wird Write-Char deaktiviert
  - Leeren
    - Clear-Bit aktivieren -> Display wird geleert -> Clear-Bit wieder deaktiviert
  - Cursor
    - Cursor-Bits wie gewünscht setzen und Write-Cursor-Bit setzen
    - IP starten

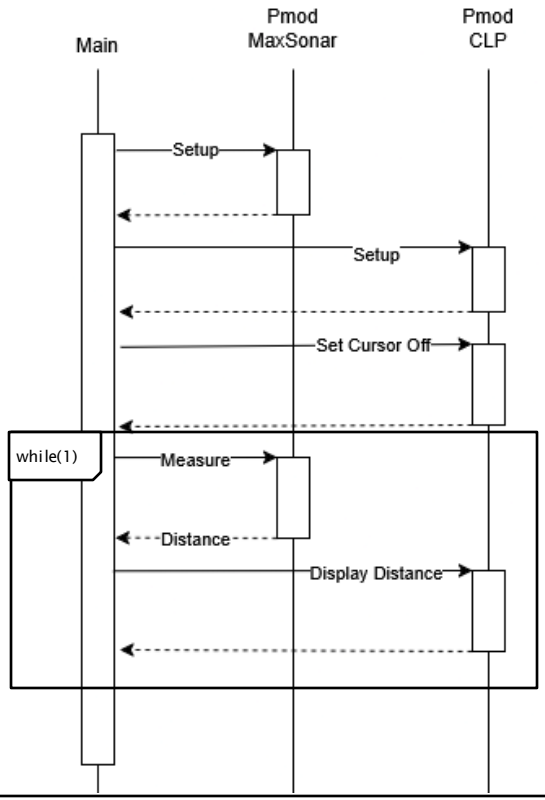
Lösung SONAR IP- SW - Übersicht (Krempl, Sowada)

- Ziele:
  - Auslesen der Distanz
  - Rücksetzen des Sensors

- Umsetzung:
  - Auslesen
    - Bei Auto-Restart auf AP-Done warten
    - Prüfung auf mögliche Messfehler
    - Auslesen der Distanz aus dem Distanz Register
  - Zurücksetzen
    - Setzen der Reset Bits
    - Warten auf Bestätigung eines erfolgreichen Neustarts

Beispielablauf - SW - Übersicht (Krempl, Sowada)

- IPs initialisiert
  - CLP: Warten auf Initialisierung
  - Sonar: Reset, warten bis Startvorgang abgeschlossen und Auto-Restart aktiviert wird
- Cursor abschalten
- Schleife:
  - Distanz messen
    - Sonar: Wenn bereit Distanz in Zoll und Cm aus Register lesen
  - Distanz anzeigen
    - Distanzwert in String umwandeln
    - CLP: String ausgeben

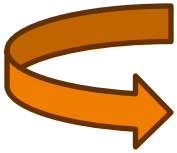


Übersicht der verwendeten Register(AS & AD) - SW - Übersicht (Krempl, Sowada)

- AS:
  - GCSR (General/Global Control and Status Register)
    - Starten der IP
    - Aktivieren Auto-Restart Modus
    - Überprüfen, ob die IP die Berechnung abgeschlossen hat
  - SCSR0 (Special Control and Status Register)
    - Erkennen ob IP Hochgefahren ist
    - Neustarten der IP
  - ADSR (ASCII Decoder Status Register)
    - Erkennen von Fehlern beim Messen
  - DIST0 (Distance Value Register)
    - Auslesen der gemessenen Distanz in Zoll
- AD:
  - GCSR (General/Global Control and Status Register)
    - Um Befehle wie Write, Clear oder setCursor auszuführen
  - SCSR0 (Special Control and Status Register)
    - Erkennen ob IP Hochgefahren ist
    - Neustarten der IP
  - DCR(Display Control Register)
    - Um das Display zu leeren und den Cursor einzustellen
  - CDR(Charakter Data Register)
    - Ermöglicht die Ausgabe am Display
  - CCR(Charakter ControlRegister)
    - Aktiviert den Schreibvorgang

Evaluierung/Ergebnisse (Team)

- Interpretation der UART-Daten in Hardware (ASCII-Decoder)
- Testbenches beider IP's valide und konform
- Ausführliche und erfolgreiche AXI-Verifikation beider IP's
- Funktionsfähige Integration beider IP's in die HW-Plattform
- Erfolgreiches Ansprechen der IP's via SW
- Daten von Sonar mittels SW-Polling auslesbar
- Daten an CLP mittels SW übertragbar und darstellbar



Anforderungen komplett erfüllt

Diskussion/Schlusszusammenfassung (Team)

- Bedeutung der Ergebnisse
  - Verständnis über Aufbau eines Prozessorsystems erlangt
  - Prinzip der Umsetzung von Timing-Constraints in Hardware verstanden
  - Projekterfahrung bzw. erste Schritte im Bereich FPGA-Programmierung
- Offene Punkte
  - KEINE
- Ausblick
  - Polling durch Interrupts ersetzen
  - Einheitenwechsel in der Software (inch, cm) durch Knopfdruck
  - Gegenlesen von geschriebenen Zeichen des Displays
  - Scrolling bei Display
  - Zeilenwechsel bei Display
  - Positionsspezifisches Schreiben eines Zeichens auf das Display

Quellcodeübersicht (Team)

- Quellcode/src/vhdl-clp-base --> Jürgens, Specht
  - Source Code für CLP IP (Vivado Projekt)
- Quellcode/src/vhdl-maxsonar-base --> Becker, Koch
  - Source Code für MaxSonar IP (Vivado Projekt)
- Quellcode/src/sw --> Krempl, Sowada
- Quellcode/src/hw-platform --> Krempl, Sowada

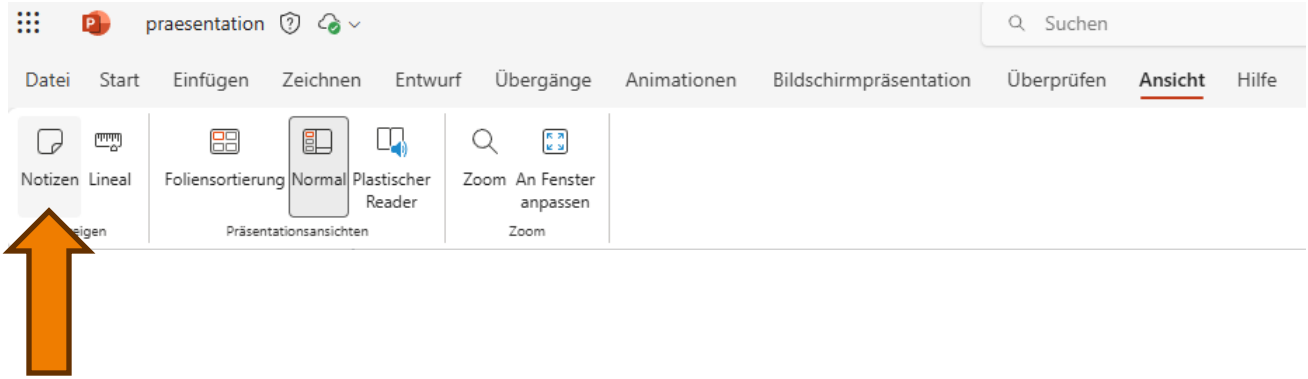
Literaturverzeichnis (Team)

- [1] Pong P. Chu, FPGA prototyping by VHDL examples, Wiley, New Jersey, 2008
- vom Dozenten in der Projektaufgabe bereitgestellte Datenblätter

Backup

Allgemein (Team)

- Bitte Notizen zu den Folien beachten

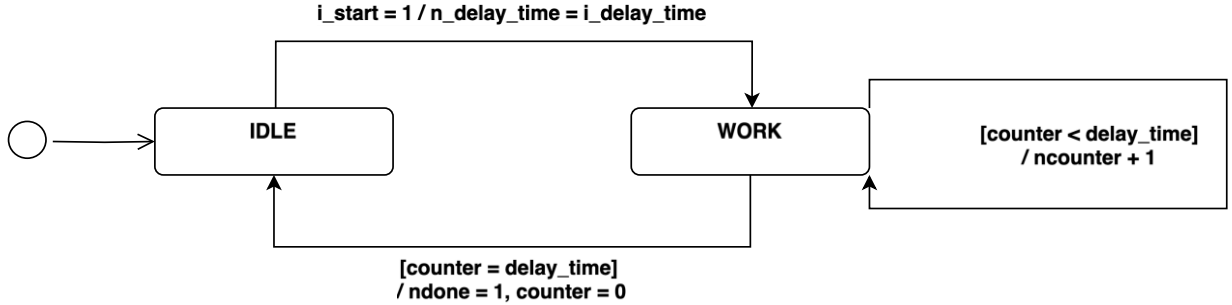


Backup – FSM Timing Controller (Jürgens, Specht)

Übersicht:

- Moore FSM
- Testbench mit Waveform
- Taktfrequenz 100MHz - Auflösung 1 Takt = 10ns
- Relevant für korrekte Timings

- Kombinatorischer Prozess (FSM):



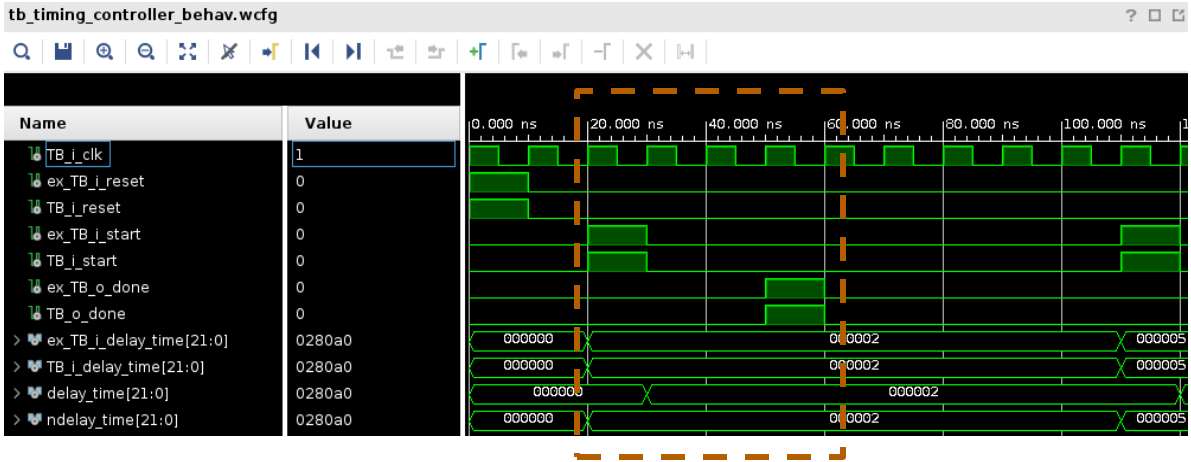
Backup – Probleme Timing Controller (Jürgens, Specht)

Aufgetretene Probleme:

- Timer lief nur für 2 Takte
  - Ausgangssituation: i_delay_time wurde für den Vergleich in WORK verwendet
  - Identifizierung: Waveform analysiert
  - Auffälligkeit: i_delay_time nach einem Takt 0
  - Grund: i_delay_time wurde nach einem Takt im Top-Module auf 0 gezogen (default value)
  - Lösung: delay time muss gelatcht werden

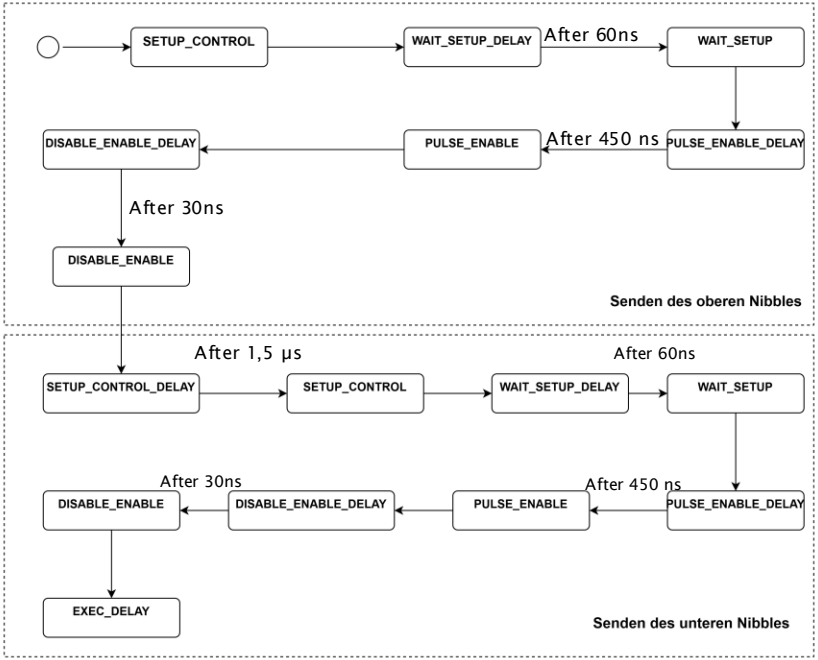
Backup - Waveform Timing Controller (Jürgens, Specht)

Ausschnitt Waveform Testbench:



Backup – FSM Ausführungsschleife (Jürgens, Specht)

- Ausführungsschleife die Timings durchsetzt
- Erst wird das obere Nibble gesendet, dann das untere
- Referenziert in: LCD_Controller FSM



Backup - Entwurf LCD Controller (Jürgens, Specht)

Ausschnitt ChatGPT:

- Verwendete für jeden Command eigene States, bspw. für 'Function Set':

```
232 +        ------------------------------------------------------------------------
233 +        ---------------------         Function Set          --------------------
234 +        ------------------------------------------------------------------------
235 +        when SEND_FUNCTION_SET_HI =>
236 +            read_write_enable <= '1';
237 +            register_select <= '0';
238 +            read_write <= '0';
239 +            db4_7 <= CMD_INIT_FUNCTION_SET(7 downto 4);
240 +            timer_delay_time <= std_logic_vector(C_EN_PULSE);
241 +            timer_read_write <= '1';
242 +            timer_start <= '1';
243 +            nstate <= PULSE_FUNCTION_SET_HI;
244 +
245 +        when PULSE_FUNCTION_SET_HI =>
246 +            if timer_done = '1' then
247 +                timer_start <= '0';
248 +                timer_delay_time <= TIME_FUNCTION_SET;
249 +                timer_read_write <= '0';
250 +                timer_start <= '1';
251 +                nstate <= WAIT_FUNCTION_SET_HI;
252 +            end if;
253 +
254 +        when WAIT_FUNCTION_SET_HI =>
255 +            if timer_done = '1' then
256 +                timer_start <= '0';
257 +                nstate <= SEND_FUNCTION_SET_LO;
258 +            end if;
```

```
260 +        when SEND_FUNCTION_SET_LO =>
261 +            db4_7 <= CMD_INIT_FUNCTION_SET(3 downto 0);
262 +            timer_delay_time <= std_logic_vector(C_EN_PULSE);
263 +            timer_read_write <= '1';
264 +            timer_start <= '1';
265 +            nstate <= PULSE_FUNCTION_SET_LO;
266 +
267 +        when PULSE_FUNCTION_SET_LO =>
268 +            if timer_done = '1' then
269 +                timer_start <= '0';
270 +                timer_delay_time <= TIME_FUNCTION_SET;
271 +                timer_read_write <= '0';
272 +                timer_start <= '1';
273 +                nstate <= WAIT_FUNCTION_SET_LO;
274 +            end if;
275 +
276 +        when WAIT_FUNCTION_SET_LO =>
277 +            if timer_done = '1' then
278 +                timer_start <= '0';
279 +                nstate <= SEND_DISPLAY_ON_OFF_HI;
280 +            end if;
```

Backup – Specs LCD Controller (Jürgens, Specht)

Sehr relevante Teile der Datenblätter:

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode	E Cycle Time	tc	1000	-	-	ns
	E Rise / Fall Time	$t_R t_F$	-	-	25	
	E Pulse Width (High, Low)	tw	450	-	-	
	R/W and RS Setup Time	tsu1	60	-	-	
	R/W and RS Hold Time	t_{H1}	20	-	-	
	Data Setup Time	tsu2	195	-	-	
	Data Hold Time	t_{H2}	10	-	-	

iii. Execution time of commands

The Execution times (Texec) of the commands are as follows:

Instruction	Execution time (fosc=270kHz)
Clear Display	1.53 ms (1.64ms recommended for compatibility)
Return Home	1.53 ms (1.64ms recommended for compatibility)
Entry Mode Set	39 us (40us recommended for compatibility)
Display ON/ OFF Control	39 us (40us recommended for compatibility)
Cursor or Display Shift	39 us (40us recommended for compatibility)
Function Set	39 us (40us recommended for compatibility)
Set CGRAM Address	39 us (40us recommended for compatibility)
Set DDRAM Address	39 us (40us recommended for compatibility)
Read Busy Flag and Address	0 us
Write Data to RAM	39 us (40us recommended for compatibility)
Read Data from RAM	39 us (40us recommended for compatibility)

2) Special considerations for 4-bit mode:
The upper nibble (4-bit) of function set commands has to be written two times (0x2) then the lower nibble of function set is written.

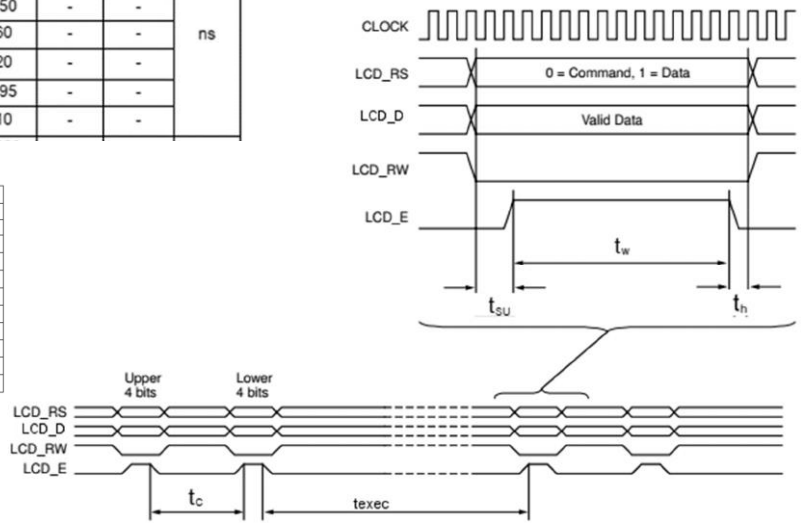
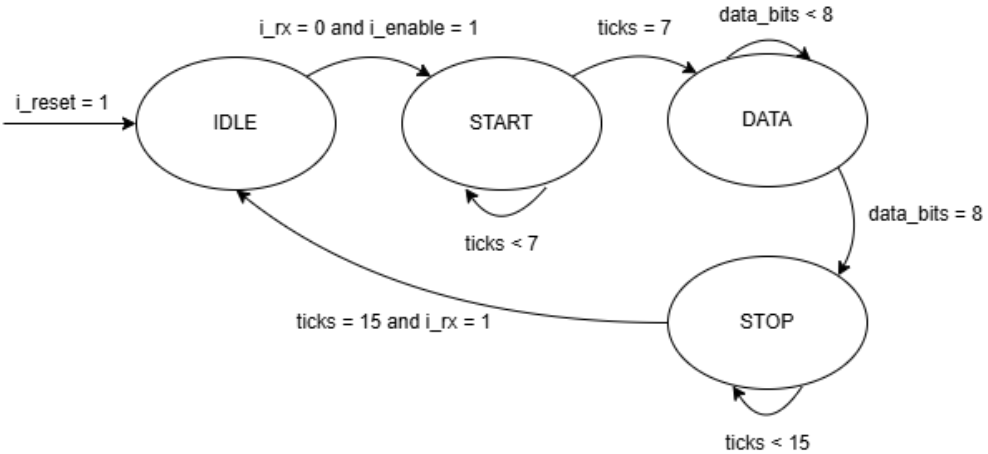


Figure 6: Bus timing Diagram (4-bit mode) for a whole 8-bit command consisting of two nibbles

Backup - UART Reciever State-Machine (Becker, Koch)



Backup - ASCII-Decoder State-Machine (Becker)