Projektthemen – 02aufgabe_pmod_hygro_pmod_clp

- Integrieren Sie die gegebenen getrennten Demosysteme zu einem gemeinsamen Demosystem (gemeinsame HW und gemeinsame SW) (so dass die Sensorwerte auf dem Display ausgegeben werden)
- Erstellen Sie eine eigene IP für den Sensor (Konzept, Code, Testbench core und Testbench AXI, Treibersoftware)
- Erstellen Sie eine eigene IP für das Display (Konzept, Code, Testbench core und Testbench AXI, Treibersoftware)
- Integrieren Sie alles auf einer gemeinsamen HW-Platform zu einem laufenden Demosystem

1. Bringen Sie die SW-basierten und auf AMD/Xilinx-IP basierten Demo-Referenzsysteme wieder zum Laufen in ihrer VM (siehe 00readme.txt)
2. Lesen Sie sich in den Code (SW und HW) und die Dokumentation/Datenblätter ein um die Funktion zu verstehen
3. Integrieren Sie diese getrennten Demosysteme zu einem gemeinsamen Demosystem (gemeinsame HW und gemeinsame SW) (so dass die Sensorwerte auf dem Display ausgegeben werden)

4a. Erstellen Sie ein Konzept (Blockblid und Registermapping mit Bitbeschreibung) für eine eigene UART-IP für pmod_maxsonar
Nehmen Sie dabei die IP von tut_int (ip_at_int_sl (Musterlösung)) als template (d.h. behalten Sie die ganzen allgemeinen Register bei (GCSR, interrupt register, IDR, VERR)
Ab incl. dem SCSR bringen Sie dann Ihre eigenen benötigten Register ein.
Orientieren Sie sich dabei an der AMD/Xilinx UART Lite IP aber reduzieren Sie die Funktionalität/Register auf das wesentliche (was hier in dieser Anwendung benötigt wird)
Tipp: Verändern Sie möglichst wenig, wenn dann entferenen Sie -> erlaubt Ihnen möglicht viel des AMD/Xilinx SW Codes zu übernehmen
4b Erstellen Sie ein Konzept (Blockbild und Registermapping mit Bitbeschreibung) für eine eigene IP zur Ansteuerung des pmod_clp in HW das den Nibble mode (also 4 Datenbits benutzt)
Nehmen Sie dabei die IP von tut_int (ip_at_int_sl (Musterlösung)) als template (d.h. behalten Sie die ganzen allgemeinen Register bei (GCSR, interrupt register, IDR, VERR)
Ab incl. dem SCSR bringen Sie dann Ihre eigenen benötigten Register ein.
Soweit es geht, soll dabei die Ansteuerung in HW stattfinden und nur das nötigste in SW (z.B. Mode-Konfiguration sowie Schreiben der anzuzeigenden Daten).
!!!Achtung achten Sie bei pmod_clp genau darauf die Timingdiagramme mit den Wartezeiten genau umzusetzen. Die Softwareimplementierung handhabt das recht lax (es funktioniert zwar in SW (aufgrund das die Execution ja auch Zeit braucht und somit Wartezeit darstellt) aber in HW müssen Sie genauer arbeiten (siehe auch Kommentare im SW Code) um das Timingdiagram der Ansteuerung exact nachzubilden - sonst wird es nicht funktionieren

5. Bereiten Sie auf Basis des gemeinsamen Demosystem eine Hardwareplatform so vor, dass sie als HW-Basisplattform die spätere Integration dienen kann (die AMD/Xilix entfernen, Interrupt controller, Hauptspeichergröße in Memorymap konfigurieren (bei sehr großen Bedarf (>128kB/256kB) durch externen DDR Speicher ersetzen (sprechen Sie mich an - dann kann ich Ihnen ein entsprechendes HW-Platform-System zur Verfügung stellen)

6. Codieren Sie die den Kern der IPs (ohne AXI) und erstellen Sie dafür geeignete Testbenches (erstmal alles nur mit Polling, aber benutzen Sie unbedingt das IP-project von tut_int (ip_at_int_sl (Musterlösung)) als template - vergleiche dazu letze Folien ganz hinten in Vorlesung interrupt, damit das später erweiterbar ist)
7. Codieren Sie die IPs (incl AXI) und erstellen Sie dafür geeignete AXI Testbenches (erstmal alles nur mit Polling, aber benutzen Sie unbedingt das IP-project von tut_int (ip_at_int_sl (Musterlösung)) als template - vergleiche dazu letze Folien ganz hinten in Vorlesung interrupt, damit das später erweiterbar ist)
8. Schreiben Sie die Treiber für die IPs (erstmal alles nur mit Polling; die Funktion der AXI Testbenches wird dabei hilfreich sein)
Verwenden Sie auch hier die SW von tut_int (sw_at_int_sl (Musterlösung)) als template - vergleiche dazu letze Folien ganz hinten in Vorlesung interrupt, damit das später erweiterbar ist)
Orientieren Sie sich dabei an der Struktur wie Xilinx/AMD seine Treiber strukituriert und halten Sie sich daran.

Aber reduzieren Sie die Funktionalität/Register auf das Wesentliche (was hier in dieser Anwendung benötigt wird)

Tipp: Verändern Sie möglichst wenig, wenn dann entferenen Sie -> erlaubt Ihnen möglicht viel des AMD/Xilinx SW Codes zu übernehmen

9. Integrieren Sie alles zu einem laufenden Demosystem

10. Bei Zeitreserve erstellen sie eine KOPIE der Projekte der HW platform, der HW IPs sowie der SW (so dass SIe auf jeden Fall eine lauffähige Demo mit Polling behalten) und erweitern Sie diese gemäß der Punkte 6-9 die Projekte um Interrupt Funktionalität