Laborübung 1

Die Bearbeitung der Aufgaben erfolgt in Gruppen zu je max zwei Teilnehmern. Alle Aufgaben sind schriftlich zu lösen und in einem Protokoll festzuhalten. Alle praktische Realisierungen sind dem Dozenten im Rahmen der Gruppenrücksprache zu präsentieren. Erfolgreiche Abnahmen werden auf dem Aufgabenblatt schriftlich quittiert. Eine erfolgreiche Laborteilnahme ist Voraussetzung für die Teilnahme an der Klausur am Ende des Semesters. Labore werden benotet und gehen zu 30% in die Endnote ein. Bitte bedenken Sie, dass eine gute Vorbereitung zu den Laborterminen unabdingbar ist.

In dieser Laborübung lernen Sie mit zwei wichtigen Tools von Xilinx umzugehen: der Vivado IDE und dem Xilinx SDK. Zunächst werden Sie in der Vivado IDE die Verbindung des Prozessors zur Peripherie herstellen, um anschließend mit dem SDK ein Programm schreiben zu können, was es Ihnen erlaubt LEDs auf dem Zedboard blinken zu lassen.

Aufgabe 1

Machen Sie sich mit der Vivado Entwicklungsumgebung von Xilinx vertraut. Die Software ermöglicht das Konfigurieren von FPGAs in einer grafischen Nutzeroberfläche.

Diese Kurzanleitung fasst alle Schritte zusammen, um mit der Vivado IDE den FPGA zu konfigurieren, detaillierte Beschreibungen der Vivado IDE finden Sie in dem Vivado Design Suite User Guide (`https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug893-vivado-ide.pdf`)

Im ersten Schritt werden wir den FPGA des Zedboards programmieren. Dazu verwenden wir von Xilinx vorgegebene *IP Cores*. Der *AXI GPIO* Block ermöglicht hier die Kommunikation des Prozessors zur Peripherie.

Starten Sie die Vivado IDE (Vivado 2016.2 auf Ihrem Desktop) und legen Sie ein neues Projekt an. Speichern Sie das Projekt in ihrem User-Bereich unter H:\ (Netzwerklaufwerk). *Hinweis:* Es ist immer eine gute Idee, den Namen seines Projektes ohne Leerzeichen zu benennen, da sonst mitunter einige Programme nicht darauf zugreifen können. Wenn sie den FPGA angeben sollen, klicken Sie oben bei *Select* auf *Boards* und wählen Sie dann aus der Liste das Zedboard aus.

Nachdem Sie mit dem Projektwizard fertig sind, sehen sie das Hauptfenster wie im Nutzerhandbuch auf Seite 16. Damit alle aus dem Blockdesign erstellen Dateien auch in VHDL (und nicht in Verilog) sind, gehen sie bitte im *Flow Navigator* auf *Project Settings* und ändern sie bitte die *Target Language* auf VHDL und bestätigen Sie mit *OK*.

Als nächstes legen Sie ein neues Blockdesign an. Klicken Sie dazu im *Flow Navigator* auf *Create Block Design*. Im *Workspace* sehen sie darauf ein leeres Blockdesign. Mit einem Rechtsklick in dieses und dem Menüpunkt *Add IP* (Shortcut: Strg + I) können Sie nun IP Cores zu dem Design hinzufügen.

Um den auf dem Zynq enthaltenen ARM Prozessor einzubinden, fügen Sie den IP Core *ZYNQ7 Processing System* ein. Jeder IP Core hat viele Ein- und Ausgänge, welche verbunden werden müssen. Um das Einbinden von Xilinx-eigenen IP Cores zu vereinfachen, bietet Vivado in einem grünen Streifen oben am Workspace Automatisierungen an. Klicken Sie nun auf *Run Block Automation*.

Als nächstes fügen sie den IP Core *AXI GPIO* ein. Klicken Sie oben auf *Run Connection Automation* und wählen Sie die *S_AXI* Automatisierung. Diese fügt den AXI Interconnect sowie den Processor System Reset hinzu. Als nächstes klicken Sie wieder auf *Run Connection Automation* und wählen in den Optionen die *leds_8bits*. Damit ist das Design fertig. Um es ein wenig übersichtlicher zu gestalten, können Sie noch mit einem Rechtsklick und der Auswahl *Regenerate Layout* (alternativ auch durch Klick links in der Leiste mit kleinen Icons verfügbar) ihr Design automatisiert aufräumen lassen.

Validieren Sie das Design mit einem Klick auf das Icon *Validate Design* (gelber Würfel mit grünem Pfeil, Shortcut: F6), welches sich auf der linken Seite des Workspace befindet. Es sollten keine Fehler auftreten. Das fertige Block Design sehen Sie in der Abbildung 1.

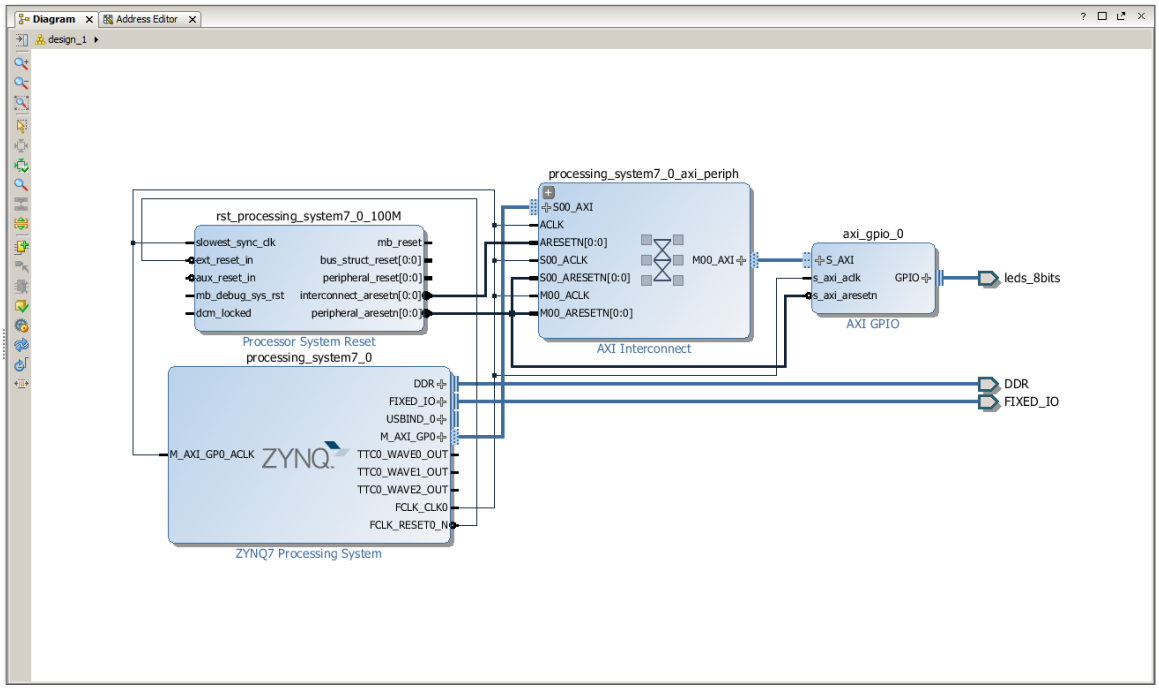

Abbildung 1: fertiges Block Design

Als nächstes klicken Sie im *Data Windows Area* unten den Reiter *Sources*. In dem Ordner *Design Sources* finden Sie ihr Blockdesign. Machen Sie darauf einen Rechtsklick und wählen Sie *Create HDL Wrapper...*, dadurch wird ihr eben erstelltes Design als VHDL-Code exportiert.

Im *Flow Navigator* wählen sie unter *Program and Debug* die Option *Generate Bitstream*. Dies fragt Sie außerdem, ob Sie ihr Design speichern wollen, sollten Sie das nicht bereits getan haben. Des weiteren startet dies die Synthese und Implementation Ihres Designs, wenn sie den Dialog *No Implementation Results Available* mit *Yes* bestätigen. Synthese, Implementation und die Erstellung des Bitstreams dauern ein wenig Zeit. Wundern Sie sich also nicht, wenn eine Weile nichts passiert. Sie können erkennen, dass das Programm noch am Arbeiten ist, wenn in der *Project Status Bar* oben rechts noch der blaue Balken hin- und herläuft. Sobald alles fertig ist, steht dort *Write_Bitstream Complete*. Wenn die Erstellung des Bitstreams fertig ist, öffnet sich ein Fenster mit Optionen. Klicken Sie dort einfach auf OK.

Verbringen Sie ein bisschen Zeit damit das Programm noch weiter zu erkunden. Nach erfolgreicher Synthese und Implementation stehen Ihnen diverse Informationen zur Verfügung. So gibt Ihnen das Programm nicht nur Auskunft darüber, wie viele Ressourcen ihr Design verbraucht, sondern auch wo auf dem FPGA ihr Design implementiert wird und wie das Timing dieses Designs ist.

Konkretere Angaben zur Synthese und der Implementation finden Sie in den *Reports*. Diese finden sie im *Results Windows Area* im Reiter *Reports*. Mit einem Doppelklick können Sie diese öffnen. Besonders interessant ist hier der *Utilization Report*, in welchem der konkrete Ressourcenverbrauch Ihres Designs aufgeführt ist.

Wenn Sie damit fertig sind, exportieren Sie ihr Projekt, indem Sie im Menü auf *File / Export / Export Hardware* klicken. Stellen Sie dabei sicher, dass der Haken bei *Include Bitstream* gesetzt ist.

Als Letztes können Sie nun das SDK im Menü unter *File / Launch SDK* öffnen.

Vorbereitung für Aufgabe 2

Hat der ARM Prozessor Memory Mapped oder Isolated IO?

Aufgabe 2

Öffnen sie im Menü unter *File / New / Application Project* ein neues Projekt. Geben Sie diesem einen Namen und wählen Sie bei den Templates *Hello World* aus.

Auf der linken Seite finden Sie im *Project Explorer* in Ihrem Projektordner im Ordner *src* eine Quelldatei *helloworld.c*. Öffnen Sie diese mit einem Doppelklick und modifizieren sie diese dann so, dass sie wie folgt aussieht:

```
#include <stdio.h>
#include "platform.h"
// Include Files
#include "xparameters.h"
#include "xgpio.h"
#include "xstatus.h"
#include "xil_printf.h"

// Definitions
#define GPIO_DEVICE_ID XPAR_AXI_GPIO_0_DEVICE_ID
#define LED 0x55//0xC3
#define LED_DELAY 10000000
#define LED_CHANNEL 1
#define printf xil_printf

XGpio Gpio;

// void print( char *str );

int LEDOutputExample( void ) {
   volatile int Delay;
   int Status;
   int led = LED;

   // GPIO driver initialization
   Status = XGpio_Initialize( &Gpio, GPIO_DEVICE_ID );
   if ( Status != XST_SUCCESS ) {
      return XST_FAILURE;
   }

   // Set the direction for the LEDs to output
   XGpio_SetDataDirection( &Gpio, LED_CHANNEL, 0x00 );

   // loop forever blinking the LED.
   while( 42 ) {
      // write output to LEDs
      XGpio_DiscreteWrite( &Gpio, LED_CHANNEL, led );

      // flip LEDs
      led = ~led;

      for ( Delay = 0; Delay < LED_DELAY; Delay++ );
   }

   return XST_SUCCESS;
}
```

```
int main()
{
    init_platform();

    print("Hello meine LED\n\r");

    LEDOutputExample();

    cleanup_platform();
    return 0;
}
```

Als nächstes programmieren Sie den FPGA mit dem Block Design, welches Sie in der Vivado IDE erstellt haben. Wählen Sie dazu im Menü unter *Xilinx Tools / Program FPGA*.

Um das erstellte C-Programm auf dem Zedboard zum Laufen zu bringen, klicken Sie nun noch auf *Menü / Run / Run As / 1 Launch on Hardware (System Debugger)*. Nachdem das Programm auf das Zedboard geladen wurde, sollten Sie die LEDs blinken sehen.

Das SDK von Xilinx hat ein praktisches Feature, um Funktionsaufrufe nachvollziehen zu können: Hält man die Strg-Taste gedrückt und klickt auf den Funktionsnamen im Code, öffnet es die Datei mit der dazugehörigen Funktionsdefinition.

Nutzen Sie dieses Feature, um den Code zu analysieren und beschreiben Sie kurz, wie die Funktion *XGpio_DiscreteWrite* funktioniert.

Woran / Wo sieht man die Antwort von der Vorbereitung im Code?

Aufgabe 3 (Zusatz)

In dieser Aufgabe sollen die Schalter ausgelesen und anschließend auf den LEDs ausgegeben werden. Legen Sie für diese Aufgabe ein neues Projekt an.

Um diese Aufgabe zu realisieren, müssen folgende zwei Änderungen vorgenommen werden:

1. In dem Block Design wird ein zweiter GPIO Block benötigt, um die Schalter auszulesen. In der *Connection Automation* steht dazu die Option *switches* zur Verfügung.
2. Der Quellcode ist anzupassen.

Nur vom Dozenten nach erfolgreicher Abnahme auszufüllen!

Gruppenname

Abnahme

Aufgabe 1	Aufgabe 2	Aufgabe 3

Beurteilung

Kriterium	Max.	Punkte
Themenbearbeitung entsprechend Zielstellung	15	
Fundierte Erarbeitung des Lösungsansatzes	20	
Qualität der technischen Umsetzung	15	
Funktionsfähigkeit der technischen Umsetzung	20	
Darstellung von Ergebnissen und Lösungsvorschlägen	15	
Bearbeitungszeit	15	
Gesamtpunkte für Beurteilungskriterium	100	
Note		

Index	95	90	85	80	75	70	65	60	55	50
Note	1,0	1,3	1,7	2,0	2,3	2,7	3,0	3,3	3,7	4,0