

Modernes Schaltungsdesign

FPGA – Praktikum

am

II. Physikalischen Institut

Justus-Liebig-Universität Gießen

5 Einleitung

Das Praktikum beginnt mit einer Einführungsvorlesung. Die Themen dieser Vorlesung sind :

1. Programmierbare Logikbausteine / ASIC¹
FPGA², Aufbau, Funktionsweise
Vor- und Nachteile zu diskreter Logik, Mikroprozessoren und Mikrocontrollern
2. VHDL
Eine Sprache zur Beschreibung von Hardware
3. Designflow
 - (a) Definition : Design-Ziel
 - (b) Definition : Funktionsblöcke u. Verbindungen
 - (c) Blockdiagramm
 - (d) Übersetzen (coden) in VHDL
 - (e) Simulation (behavior)
 - (f) Synthese von Schaltungen
 - (g) Schaltplan, Schematic³ (View RTL⁴Schematic)
 - (h) user constraints (Verbinden der Schaltung mit den Pins)
 - (i) Fitten auf Architecture (FPGA)
 - (j) Laden der Schaltung in den Baustein und Starten
4. Zusätzliche nützliche Hilfsmittel
"Emacs", ein sprachsensitiver Editor. Der VHDL-Source-Code kann mit jedem beliebigen Editor erstellt und bearbeitet werden. Mit einem sprachsensitiven Editor lassen sich viele Syntax-Fehler vermeiden.

¹Application Specific Integrated Circuit oder anwendungsspezifischer integrierter Schaltkreis; individuell für eine spezifischen Anwendung entwickelter Schaltkreis. Zum Vergleich mit Standardbauelementen, die nicht kundenspezifisch konfiguriert sind wie z.B. Spannungsregler, Speicher, Prozessoren. Der Anwender legt die Funktion des Bausteins fest, der dann entsprechend seiner Vorgabe gefertigt wird.

²FPGA: Field Programmable Gate Array. Der integrierte Baustein enthält viele Zellen mit diskreten aber auch komplexeren Grundbausteinen. Diese Grundbauelemente können miteinander mehr oder weniger frei verdrahtet werden. Die Art der Verdrahtung werden durch bits in einem Memory festgelegt

³elektronischer Schaltplan

⁴RTL: Register Transfer Logic; Daten werden in einem Register zwischengespeichert

5. Logic-Analyzer

Ähnlich wie bei einem Oszilloskop werden Signale grafisch dargestellt. Beim Logic-Analyzer können jedoch wesentlich mehr Signale, die allerdings nur die Werte H oder L annehmen können, dargestellt werden.

Nach der Einführungsvorlesung beginnen die praktischen Übungen. Die Einzelnen Aufgaben sind durchnummeriert A01, A02,... Da wir davon ausgehen, dass die Vorkenntnisse und Erfahrungen der einzelnen Praktikanten sehr unterschiedlich sind, gibt es keine abgegrenzten Versuche oder Projekte, sondern jeder Praktikant oder Praktikanten-Gruppe geht je nach Fähigkeit voran. Sie sollten die einzelnen Aufgaben möglichst selbständig lösen. Falls Sie sich das nicht zutrauen können Sie die Assistenten fragen; diese haben zu jeder Aufgabe Musterlösungen, die jedoch nicht vollständig sind und/oder in die Fehler eingebaut sind.

5.1 Versuchsausarbeitung:

1. Die Versuchsausarbeitung soll die **ausführlich kommentierten** Quelltexte enthalten – Hardware-Beschreibung (*.vhd); sowie gegebenenfalls Testbench (*.tb.vhd) und user-constraints (*.ucf)
2. dokumentierte Schaltpläne
3. Es soll in kurzen Worten beschrieben werden wie man die Ergebnisse erzielt hat.
4. Grafiken, Diagramme und waves sollen als Screen-shot erstellt werden

Führen Sie dem Assistenten Ihre Lösung vor !

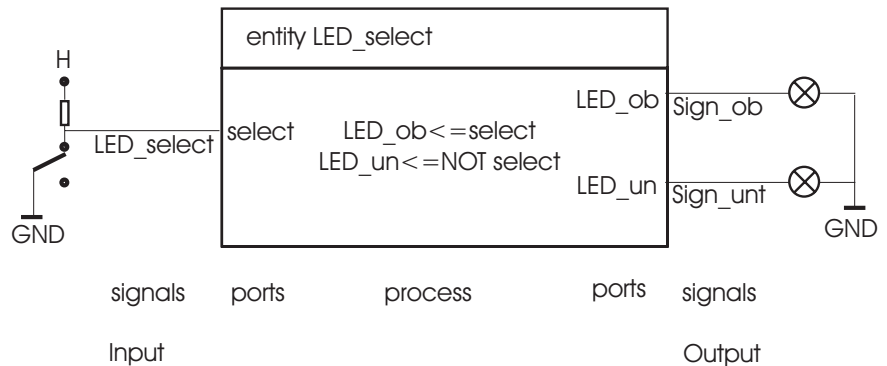
5.2 Einige nützliche Links

Informationen zu den Boards	http://www.digilentinc.com/assets/documents/dio2_rm.pdf
Fa. XILINX	http://www.xilinx.com/
VHDL-Tutorial	http://www.gmvhdl.com/VHDL.html

1 Aufgabe A01

Bedienen von I/O-Ports

Abhängig von der Position eines Schalters (oben oder unten) soll eine Lampe (LED "oben") und Lampe (LED "unten") eingeschaltet werden.



Lernziel:

- VHDL
 1. entity und deren ports
 2. Signale und elementare Typen (bit)
 3. instances und portmaps
 4. architecture
- design flow
 1. User Constraints (Pin-Zuordnung)
 2. Laden des FPGA

1.1 Aufgaben

A01.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A01.2 Verbinden Sie die Schaltung mit dem Schalter und den beiden LEDs

A01.3 Laden Sie den FPGA mit Ihrer Schaltung

A01.4 Testen Sie die Schaltung durch Betätigen des Schalters

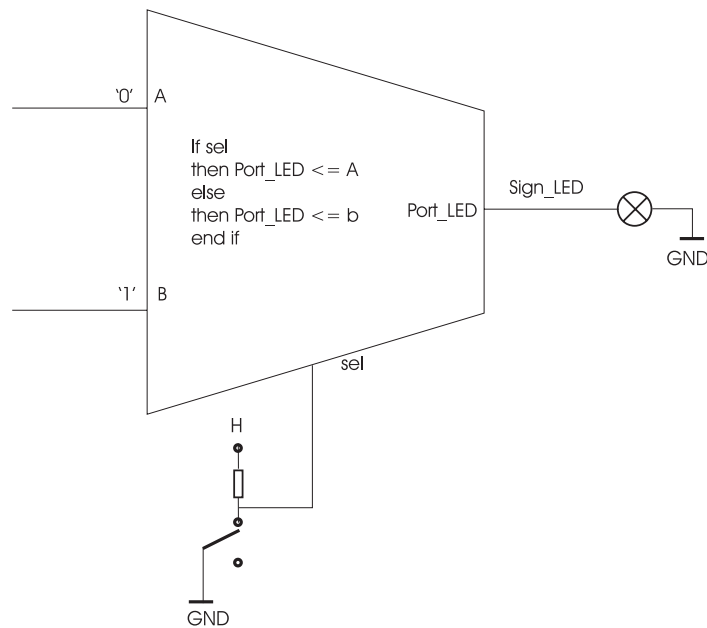
1.2 Pinbelegung

FPGA	I/O	Bauteilname
P180	Schalter	T46
P162	LED "oben"	T6
P149	LED "unten"	T26

2 Aufgabe A02

Multiplexer (MUX)

Multiplexer sind gesteuerte Schalter zum Umschalten von Datenwegen. In unserem Praktikum werden digitale Daten elektronisch geschaltet. Ein Multiplexer besitzt viele Eingänge, von diesen kann einer ausgewählt werden und dessen Zustand erscheint am Ausgang.



Lernziel:

- VHDL
 1. entity und ports WDH.
 2. Signale und elementare Typen (bit) WDH.
 3. instances und portmaps WDH.
 4. bedingte Ausdrücke (if)
 5. architecture WDH.
- design flow
 1. User Constraints Pin-Zuordnung

2. Laden des FPGA

In dem Versuch werden nur zwei Eingänge verwendet, die zuerst – wie im Bild eingezeichnet - fest mit "0" bzw. "1" verdrahtet sind. Zusätzlich sollen Sie die Schaltung so ändern, dass die beiden Daten-Eingänge ihre Daten von zwei Schaltern (Schalter A) und (Schalter B) erhalten. Durch den MUX wird wahlweise der Zustand des Schalters A bzw. B an die LED am Ausgang gelegt.

Die Auswahl des Eingangs erfolgt durch einen weiteren Schalter (SEL).

2.1 Aufgaben

A02.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A02.2 Verbinden Sie die Schaltung mit dem Schalter und der LED

A02.3 Laden Sie den FPGA mit Ihrer Schaltung

A02.4 Testen Sie die Schaltung durch Betätigen des Schalters

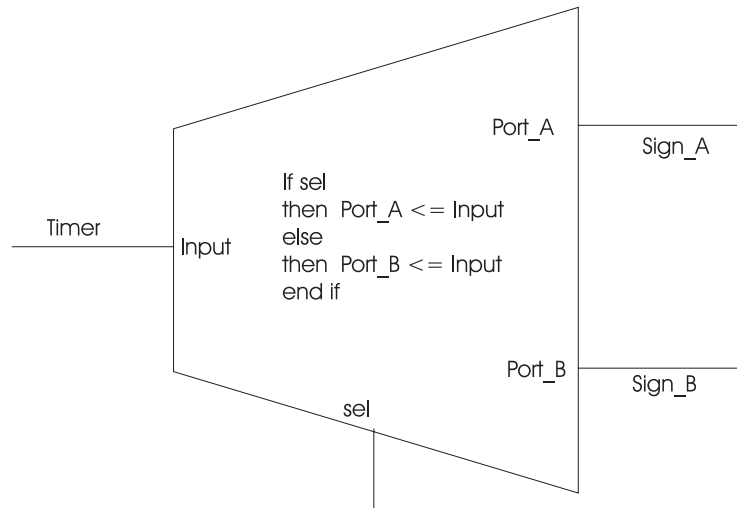
2.2 Pinbelegung

FPGA	I/O	Bauteilname
P178	Schalter SEL	T48
P180	Schalter A	T46
P87	Schalter B	T47
P162	Ausgang (LED)	T6

3 Aufgabe A03

Demultiplexer (DeMUX)

Ein Demultiplexer hat die zum Multiplexer inverse Funktion. Er erlaubt einen Ausgang anzuwählen auf den das Eingangssignal gelegt wird.



Lernziel:

- VHDL
 1. entity und ports WDH.
 2. Signale und elementare Typen (bit) WDH.
 3. instances und portmaps WDH.
 4. bedingte Ausdrücke (if) WDH.
 5. architecture WDH.
- design flow
 1. User Constraints (Pin-Zuordnung) WDH.
 2. Laden des FPGA WDH.
 3. Logic-Analyzer

3.1 Aufgaben

A03.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A03.2 Verbinden Sie die Schaltung mit den Schaltern, dem Signal des Timerbausteins sowie den 2 LEDs T6 und T26.

Achtung: Das Timersignal wird über den Schalter T49 herausgeführt !

A03.3 Laden Sie den FPGA mit Ihrer Schaltung

A03.4 Testen Sie die Schaltung durch Betätigen des Schalters und kontrollieren Sie die Funktion (Helligkeit der LEDs).

A03.5 Verbinden Sie die Signale mit dem Logic-Analyzers. Es sollen Eingang, Sel und die beiden Ausgangssignale Signal_A und Signal_B aufgezeichnet werden.

3.2 Pinbelegung

FPGA	I/O	Bauteilname	Pin Connector B
P178	Schalter SEL	T48	
P179	Signalquelle (Timer)	Schalter T49	
P162	Ausgang (Signal A)	LED T6	B4
P149	Ausgang (Signal B)	LED T26	B10

3.3 Pinbelegung und Anschluss Logicanalyzer (LA)

Schnelle Signale lassen sich nicht mit dem Auge verfolgen. Hier hilft ein Logik-Analysator (LA), der – ähnlich wie ein Oszilloskop - den zeitlichen Verlauf von Digitalsignalen aufzeichnet. Im Praktikum verwenden wir einen einfachen 18-Kanal-Logik-Analysator, der die erfassten Daten auf dem Monitor des PCs darstellt. Die Signale werden über 18 Kabel, die man auf die Pins der Praktikumsplatine stecken kann, erfasst. Auf jeden Fall muss immer Masse (GND) mit angeschlossen werden !!

4 Aufgabe A04

Flip-Flop (FF)

Toggle-FF, Register, Frequenzteiler

Falls der Enable-Eingang (clk_enable) gesetzt ist, übernimmt das FlipFlop nach einer fallenden Flanke am CLK-Eingang am Ausgang Q den Wert des D-Eingangs. Es soll ein Toggle-FlipFlops aufgebaut werden, das nach der fallenden Flanke am CLK-Eingang seinen Zustand negiert (der Ausgang toggled).

Während Reset (R) aktiv ist, wird das FF in einen definierten Anfangszustand gesetzt : $Q \leq '0'$

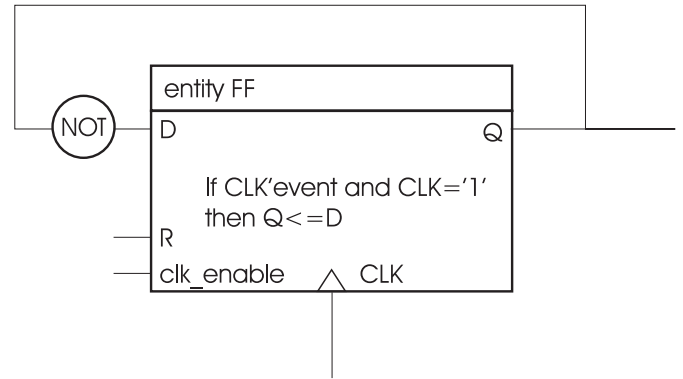
Achtung: Alle Taster sind **activ LOW**

Lernziel:

- VHDL
 1. flankengetriggerte Prozesse WDH.
 2. Attribut (events)
 3. bedingte Ausdrücke (if) WDH. Register und events
- Design Flow
 1. Testbench
 2. User Constraints Pin-Zuordnung WDH.
 3. Laden des FPGA WDH.
- Logic-Analyzer

4.1 Aufgaben

A04.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung. Enable liegt fest auf '1'.



A04.2 Mit einer Testbench (in VHDL formuliert) soll der CLK-Eingang des FFs mit Rechtecksignalen stimuliert werden.

Das Reset soll erst nach einigen CLK-Zyklen eingeschaltet werden. Das zeitliche Verhalten der Eingänge CLK, Reset und des Ausgangs Q soll mit Hilfe des Simulators aus ModelSim dargestellt werden.

A04.3 Verbinden Sie den Cl-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine und den Ausgang Q mit einer LED

A04.4 Laden Sie den FPGA mit Ihrer Schaltung

A04.5 Testen Sie die Schaltung, was sehen Sie an der LCD ? Warum erkennen Sie wenig ?

A04.6 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers

A04.7 Entwickeln Sie eine entity "Toggle-FlipFlop" (Schaltplan und VHDL-Code)

4.2 Pinbelegung

FPGA	I/O	Bauteilname
P182	50MHz-Oszillator	
P173	Taster (Reset) activ LOW	T52
P162	LED Q	T6

5 Aufgabe A05

Instantisieren von Baugruppen

Das FlipFlop aus der vorhergehenden Aufgabe soll in ein übergeordnetes Modul übernommen werden.

Lernziel:

- VHDL
 1. Instantisieren

5.1 Aufgaben

A05.1 Laden Sie den vhd-Code der vorherigen Aufgabe

A05.2 Laden Sie zusätzlich die übergeordnete Schaltung `Spartan2E_CHIP.vhd` als Kopie

A05.3 Laden Sie ebenso als Kopie zusätzlich das .ucf-File `Spartan2E_CHIP.ucf`

A05.4 Synthetisieren Sie das Flip-Flop. Damit erzeugen Sie gleichzeitig die "instantiation templates".

A05.5 Mit "View VHD instantiation template" erhalten Sie die Beschreibung der Baugruppe FlipFlops :

```
COMPONENT flipflop
PORT(
);
END COMPONENT;
```

Sowie ein Gerüst zum Verbinden der Baugruppe FlipFlop in einer übergeordneten Schaltung :

```
Inst_flipflop: flipflop PORT MAP(
);
```

A05.6 Diese beiden Blöcke setzen Sie in die markierten Bereiche der übergeordneten Schaltung `Spartan2E_CHIP.vhd` ein.

A05.7 Nun müssen Sie noch das FlipFlop mit der übergeordneten Schaltung verbinden. Dazu verbinden Sie die Anschlüsse des FlipFlops in der Port_map mit den Signalen aus `Spartan2E_CHIP.vhd`

A05.8 `Spartan2E_CHIP.vhd` enthält alle Signale, die vom Spartan2E-Chip angesteuert werden können, diese werden auch mit `Spartan2E_CHIP.ucf` mit den Pins des Chips verbunden. Jedoch werden bei der Synthese viele dieser Signale wegoptimiert, da sie von der instantiierten Baugruppe (in diesem Fall dem FlipFlop) nicht benutzt werden. Bei der Design Implementation führt das zu einer Fehlermeldung, da nicht existente Signale mit Pins verbunden werden sollen.

Um diese Fehlermeldungen zu unterdrücken, müssen Sie "unmatched LOC constraints" erlauben: Wählen Sie "Implement design", betätigen die rechte Maustaste, selektieren "Properties" und setzen den entsprechenden Haken unter "translate properties"

5.2 Pinbelegung

Signal vom FlipFlop	Signalname
Takt	50MHz-Oszillator
Reset	Button_T53
P	LED_T6_green
Q	LED_T26_green

In `Spartan2E_CHIP.vhd` sind bereits die meisten Ihnen zur Verfügung stehenden Signale definiert. Die folgende Tabelle gibt einen Überblick. Alle Namen einer Zeile der Tabelle steuern die gleichen Pins des FPGAs an, bzw. werden vom gleichen PIN angesteuert. Sie können daher einen beliebigen auswählen.

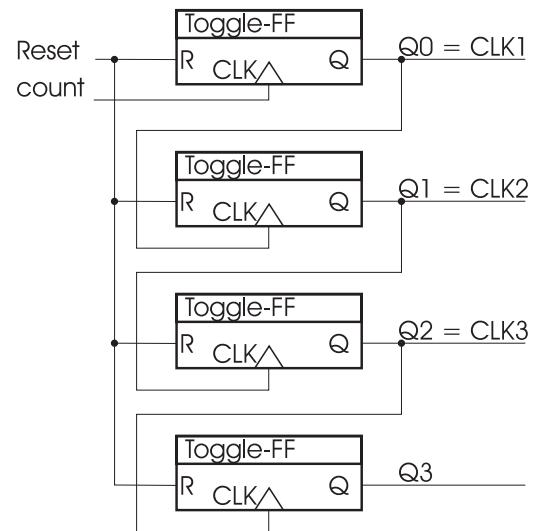
Die erste Spalte bezeichnet den PIN des FPGAs, die zweite den PIN der Steckerleisten der Zusatzplatine, und die folgenden einen aussagekräftigen Namen.

FPGA-Pin	Pin der Steckerleiste	Signalname			
auf dem Ampel-Board					
P166	A32	LED_T36_red		T36	
P167	A31	LED_T50_yellow		T50	
P101	A39	LED_T16_green		T16	
P145	B14	LED_T40_red		T40	
P165	A33	LED_T60_yellow		T60	
P163	A35	LED_T20_green		T20	
P174	A27	LED_T39_red		T39	
P100	A36	LED_T59_yellow	BEEPER	T59	
P99	A38	LED_T19_green		T19	
P168	A30	LED_T38_red		T38	
P164	A34	LED_T51_yellow		T51	
P98	A37	LED_T18_green		T18	
P97	A40	LED_T17_red	Pedestrian_red	T17	
P169	A29	LED_T37_green	Pedestrian_green	T37	
P162	B4	LED_T6_green		T6	
P160	B6	LED_T7_green		T7	
P161	B5	LED_T8_green		T8	
P151	B8	LED_T9_green		T9	
P152	B7	LED_T10_green		T10	
P149	B10	LED_T26_green		T26	
P150	B9	LED_T27_green		T27	
P147	B10	LED_T28_green		T28	
P148	B9	LED_T29_green		T29	
P176	A25	Button_T53		T53	activ LOW
P175	A28	Button_T54		T54	activ LOW
P173	A26	Button_T52		T52	activ LOW
P180	A22	Switch_T46		T46	
P87	A21	Switch_T47		T47	
P178	A24	Switch_T48		T48	
P179	A23	Switch_T49	Timer	T49	
P80	B39	Ampel_CLK			CLK_Var
auf dem Digilab 2SB Board					
P182		CLOCK	CLK	SYSCLK	CLK50MHZ
P154		DEV_Board_LED	LED1		
P187		DEV_Board_Button	Button		

6 Aufgabe A06

asynchroner Zähler

Vier FFs sollen zu einem 4-bit-Zähler verschaltet werden. Die Ausgänge Q0,Q1,Q2 und Q3 haben die Wertigkeiten 1, 2, 4 und 8. Die zu zählenden Pulse (count) gehen auf den CLK-Eingang des ersten FFs; Der Ausgang des ersten FFs Q0 steuert den CLK-Eingang des zweiten FFs CLK1, der Ausgang des zweiten FFs Q1 steuert den CLK-Eingang des dritten FFs CLK2 usw. Mit reset (R) wird der Zähler auf null gesetzt.



Lernziel:

- VHDL
 1. alias, ports, port-maps
 2. instances WDH.
- Testbench WDH.
- User Constraints Pin-Zuordnung WDH.
- Laden des FPGA WDH.
- Logic-Analyzer WDH.

6.1 Aufgaben

A06.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A06.2 Beschreiben Sie die Schaltung aus Sicht des Schematics

A06.3 Mit einer Testbench (in VHDL formuliert) soll der Eingang des Zählers mit Rechtecksignalen stimuliert werden. Das zeitliche Verhalten des Eingangs CLK und der Ausgänge Q0, Q1, Q2 und Q3 soll mit Hilfe des Simulators aus dargestellt werden.

A06.4 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instatisieren Sie den Zähler

A06.5 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine, den Reset-Eingang mit dem Button T53 und die Ausgänge mit vier LEDs

A06.6 Verbinden Sie noch die Clock mit dem Stecker-Pin B10, damit Sie alle Signale mit dem Logic-Analyzer betrachten können.

A06.7 Denken Sie daran, wie in dem vorhergehenden Versuch die "unmatched LOC constraints" zu erlauben: Wählen Sie "Implement design", betätigen die rechte Maustaste, selektieren "Properties" und setzen den entsprechenden Haken unter "translate properties"

A06.8 Laden Sie den FPGA mit Ihrer Schaltung

A06.9 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers

6.2 Pinbelegung

Signal	Bauteil	
Q0	LED_T6_green	Wertigkeit 1
Q1	LED_T7_green	Wertigkeit 2
Q2	LED_T8_green	Wertigkeit 4
Q3	LED_T9_green	Wertigkeit 8
CLK	clock	
reset	Button_T53	
clock	B10	

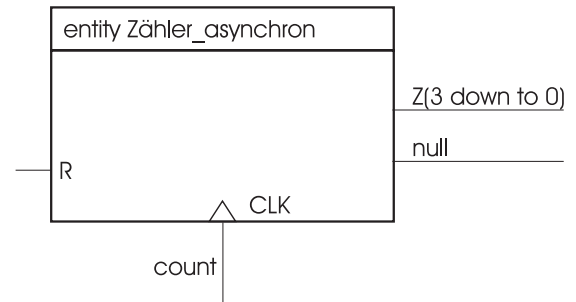
7 Aufgabe A07

Glitches

Mit den vier Ausgängen Q0 - Q3 des Zählers und seinen Wertigkeiten ergeben sich mit $Q0*1 + Q1*2 + Q2*4 + Q3*8$ je nach Zustand 0 oder 1 der Ausgänge Qi die 16 Werte 0, 1, 2,... 14, 15. Jeder dieser Werte erscheint innerhalb von 16 Takt-Zyklen genau einmal.

Mit einer logischen Verknüpfung soll dieser Übertrag (Cy) erkannt werden, wenn der Zähler auf Null steht (Ausgang null).

Die vier Zähler-Ausgänge Q0 - Q3 sollen zu einem 4 bit breiten Datenbus Z zusammengefasst werden.



Lernziel:

- VHDL
 1. komplexe Datentypen (arrays)
 2. instances und portmaps WDH.
- Testbench
- Timing-Probleme (asynchrony, Glitch)
- User Constraints Pin-Zuordnung WDH.
- Logic-Analyzer WDH.

7.1 Aufgaben

A07.1 Entwickeln Sie aus dem Zähler der vorhergehenden Aufgabe eine entity (siehe Bild) mit folgenden Eigenschaften. Z sei der Zählerstand des 4 bit-Counters. Der Ausgang "null" zeigt den Zählerstand 0 an.

A07.2 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A07.3 Beschreiben Sie die Schaltung aus Sicht des Schematics

A07.4 Mit einer Testbench (in VHDL formuliert) soll der Eingang des Zählers mit Rechtecksignalen stimuliert werden. Das zeitliche Verhalten des Clock-Eingangs, des Zähler-Ausgangs und der Null soll mit Hilfe des Simulators dargestellt werden.

Welches seltsame Verhalten zeigt der Ausgang "null"? Welche Ursache hat dieses Verhalten ?

A07.5 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Baugruppe

A07.6 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator, den Reset-Eingang mit dem Button T53 und die Ausgänge mit vier LEDs

A07.7 Verbinden Sie noch die Clock mit dem Stecker-Pin B10 und die Null mit dem Stecker-Pin B9.

A07.8 Denken Sie daran, wie in den vorhergehenden Versuchen die "unmatched LOC constraints" zu erlauben.

A07.9 Laden Sie den FPGA mit Ihrer Schaltung

A07.10 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers. Warum sehen Sie nicht die "seltsamen Signale" aus der Testbench ?

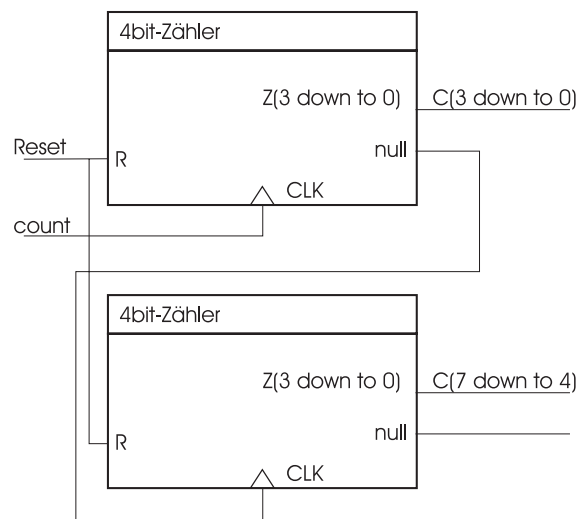
7.2 Pinbelegung

Signal	Bauteil	Pin Steckerleiste	
Q0	LED_T6_green	B4	Wertigkeit 1
Q1	LED_T7_green	B6	Wertigkeit 2
Q2	LED_T8_green	B5	Wertigkeit 4
Q3	LED_T9_green	B8	Wertigkeit 8
CLK	clock		
reset	Button_T53		
clock		B10	
Null		B9	

8 Aufgabe A08

Auswirkungen der Glitches

Zwei 4-bit-Zähler sollen kaskadiert werden. Dabei soll der zweite Zähler um eins erhöht (inkrementiert) werden, wenn der erste Zähler auf Null (Weiterzählen mit ausdekodierter Null) springt.



Lernziel:

- VHDL
 1. array, vector WDH.
 2. instances, portmaps WDH.
- Testbench WDH.
- Timing-Probleme (glitches)
- User Constraints Pin-Zuordnung WDH.
- Logic-Analyzer WDH.

8.1 Aufgaben

A08.1 Beschreiben Sie das Blockdiagramm in VHDL

A08.2 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung

A08.3 Beschreiben Sie das schematic, das Ihnen das Synthese-tool liefert.

A08.4 Mit einer Testbench (in VHDL formuliert) soll der Eingang des Zählers mit Rechtecksignalen stimuliert werden. Das zeitliche Verhalten der Eingänge CLK und R, der 8 Ausgänge C(7 downto 0) sowie der Null sollen mit Hilfe des Simulators dargestellt werden.

Wann sollte der zweite Zähler schalten und wann schaltet er nach der Simulation?

A08.5 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Baugruppe

A08.6 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator, den Reset-Eingang mit dem Button T53 und die Ausgänge mit 8 LEDs bzw. den Pins der Steckerleisten

A08.7 Verbinden Sie noch die Clock mit dem Stecker-Pin A32 und die Null mit dem Stecker-Pin B9.

A08.8 Denken Sie an die "unmatched LOC constraints".

A08.9 Laden Sie den FPGA mit Ihrer Schaltung

A08.10 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers. Wie verhalten sich die Signale, die die Wertigkeiten 16, 32, 64 und 128 haben sollten ?

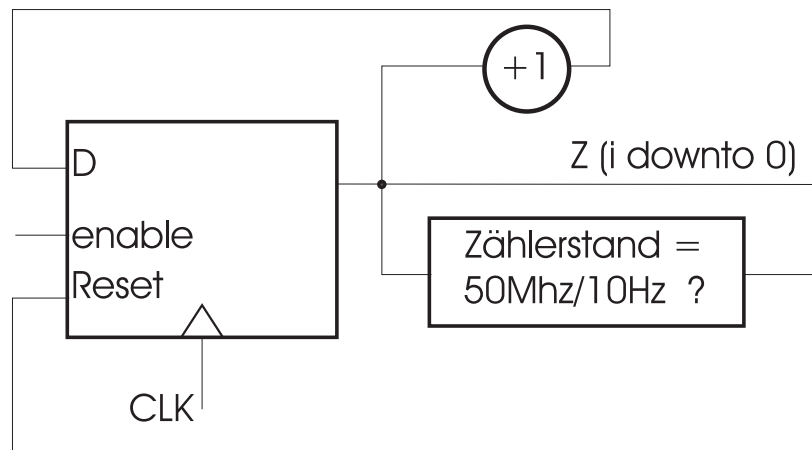
8.2 Pinbelegung

Signal	Bauteil	Pin Steckerleiste	
Q0	LED_T6_green	B4	Wertigkeit 1
Q1	LED_T7_green	B6	Wertigkeit 2
Q2	LED_T8_green	B5	Wertigkeit 4
Q3	LED_T9_green	B8	Wertigkeit 8
Q4	LED_T10_green	B7	Wertigkeit 16 ?
Q5	LED_T26_green	B10	Wertigkeit 32 ?
Q6	LED_T50_yellow	A31	Wertigkeit 64 ?
Q7	LED_T16_red	A39	Wertigkeit 128 ?
CLK	clock		
reset	Button_T53		
clock		A32	
Null		B9	

9 Aufgabe A09

Synchroner Zähler

Entwickeln Sie einen Taktgeber für 0.1s. Gezählt werden darf nur bei enable aktiv (auf Schalter legen). Unabhängig von enable wird mit Reset (auf Taster legen) der Zählerstand auf 0 gesetzt.



Lernziel:

- VHDL
 1. verschiedene Interpretationen von Bitmustern
 2. Datentypen WDH.
 3. arithmetische Operationen
 4. Defaultwerte
- Design Flow
 1. Schematic WDH.
 2. Testbench WDH.
 3. Timing-Probleme WDH.
 4. User Constraints Pin-Zuordnung WDH.
- Logic-Analyzer WDH.

9.1 Aufgaben

A09.1 Wieviele bits benötigen Sie ?

A09.2 Entwickeln Sie den VHDL-Code der synchronen (flankengesteuerten Schaltung) und synthetisieren Sie die Schaltung

A09.3 Welche Schaltung erzeugt das Synthese-tool ?

A09.4 Simulieren Sie die Schaltung

A09.5 Warum ist dieser Zähler synchron und warum treten hier die Timingprobleme aus den vorhergehenden Aufgaben nicht auf ?

A09.6 Welches Tastverhältnis (1- zu 0-Zeit) entsteht am Ausgang des MSB ?

A09.7 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Baugruppe

A09.8 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator, den Reset-Eingang mit dem Button T53, enable mit einem Schalter T46 und die Ausgänge des Zählers mit Pins auf der Steckerleiste, um sie mit dem Logic-Analyzer anzusehen

A09.9 Wählen Sie sowohl niederwertige Ausgänge (Q0,Q1,Q2,...) als auch die höchstwertigen (Qx...)

A09.10 Verbinden Sie noch die Clock mit dem Stecker-Pin A32.

A09.11 Denken Sie an die "unmatched LOC constraints".

A09.12 Laden Sie den FPGA mit Ihrer Schaltung

A09.13 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers.

A09.14 Ist der Ausgang des 10Hz-Oszillators symmetrisch (Tastverhältnis 1:1)?

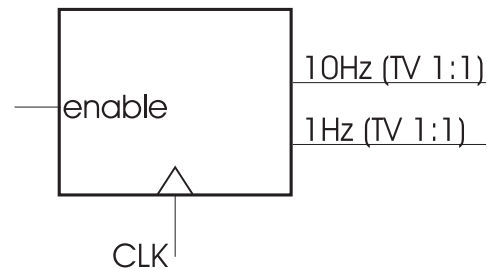
9.2 Pinbelegung

Signal	Bauteil	Pin Steckerleiste	
Q0	LED_T6_green	B4	Wertigkeit 1
Q1	LED_T7_green	B6	Wertigkeit 2
Q2	LED_T8_green	B5	Wertigkeit 4
Q3	LED_T9_green	B8	Wertigkeit 8
Q4	LED_T10_green	B7	Wertigkeit 16
Qx	LED_T26_green	B10	
Qx	LED_T50_yellow	A31	
Qx	LED_T16_red	A39	
CLK	clock		
reset	Button_T53		
enable	Switch_T46		
clock		A32	

10 Aufgabe A10

Synchroner Zähler II

Ändern Sie den 0,1s-Taktgeber der vorhergehenden Aufgabe, so dass das Ausgangstastverhältnis 1:1 ist. Stellen Sie außerdem ein zweites Signal mit 1Hz zur Verfügung. Auch das 1Hz-Signal soll ein Tastverhältnis 1:1 haben.



Lernziel:

- VHDL
 1. verschiedene Interpretationen von Bitmustern WDH.
- Design Flow
 1. Schematic WDH.
 2. Timing-Probleme WDH.
 3. User Constraints Pin-Zuordnung WDH.
- Logic-Analyzer WDH.

10.1 Aufgaben

A10.1 Welches Tastverhältnis (1- zu 0-Zeit) hat der 0,1s-Taktgeber von der vorhergehenden Aufgabe ?

A10.2 Entwickeln Sie den Schaltplan aus der Aufgabenstellung

A10.3 Entwickeln Sie den VHDL-Code der synchronen (flankengesteuerten Schaltung)

A10.4 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Baugruppe

A10.5 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine und den symmetrischen 0.1s- und 1s-Ausgang mit je einer LED sowie den Reset mit Taster T_53 und enable mit Schalter T_46.

A10.6 wählen Sie sowohl niederwertige Ausgänge (Q0,Q1,Q2,...) als auch die höchstwertigen (Qx...)

A10.7 Verbinden Sie noch die Clock mit dem Stecker-Pin A32.

A10.8 Denken Sie an die "unmatched LOC constraints".

A10.9 Laden Sie den FPGA mit Ihrer Schaltung

A10.10 Testen Sie die Schaltung mit Hilfe des Logic-Analyzers.

10.2 Pinbelegung

Signal	Bauteil	Pin Steckerleiste	
Q0	LED_T6_green	B4	Wertigkeit 1
Q1	LED_T7_green	B6	Wertigkeit 2
Q2	LED_T8_green	B5	Wertigkeit 4
Q3	LED_T9_green	B8	Wertigkeit 8
Q4	LED_T10_green	B7	Wertigkeit 16
Qx	LED_T26_green	B10	
Qx	LED_T50_yellow	A31	
Qx	LED_T16_red	A39	
10Hz	LED_T27_green		
1Hz	LED1		
CLK	clock		
reset	Button_T53		
enable	Switch_T46		
clock		A32	

11 Aufgabe A11

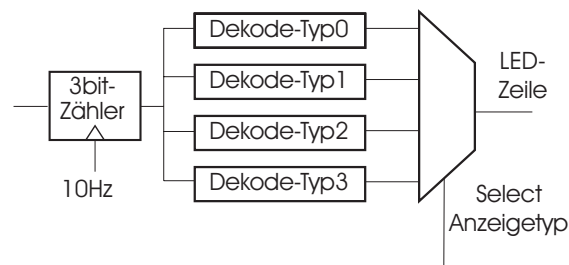
Dekoder

Der aktuelle Wert (Binärcode) eines Zählers kann auf unterschiedliche Weise anschaulich dargestellt werden. Entwickeln Sie einen Encoder, der den Zählerstand eines 3 bit Zählers verschiedenartig anzeigen kann. Mit einem 2-bit Schalter sollen vier unterschiedliche Anzeigen ausgewählt werden.

Als Anzeige sollen realisiert werden :

- 0 eine LED in einer Zeile soll leuchten, die Position der leuchtenden LED gibt den Zählerstand an
- 1 Es sollen so viele LEDs leuchten, wie der Zählerstand vorgibt, sodass ein leuchtender Balken mit der Länge des Zählerstandes entsteht
- 2 Zwei leuchtende Balken laufen gegeneinander (magisches Auge)
- 3 Der Binärcode kann angezeigt werden
- 4 Sie können einen eigenen Anzeigetyp entwerfen

Als darzustellende Daten kann der Zählerstand des langsamen Zählers der vorhergehenden Aufgabe verwendet werden.



Lernziel:

- VHDL
 1. verschiedene Interpretationen von Bitmustern WDH.
- Design Flow
 1. Schematic WDH.
 2. Testbench WDH.
 3. User Constraints Pin-Zuordnung WDH.

11.1 Aufgaben

A11.1 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung.

A11.2 verwenden Sie eine Testbench, um die Funktion der Anzeige zu überprüfen. Lassen Sie von einem Counter alle Zustände durchlaufen. Die unteren 3 bit gehen zur Anzeige, die oberen beiden bits selektieren den Anzeigetyp

A11.3 Instantisieren Sie Dekoder und Counter

A11.4 Verbinden Sie Anzeige und Counter zu einem Zähler mit wählbarem Anzeigetyp

A11.5 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre den Zähler

A11.6 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine und den Reset mit Taster T_54 und legen Sie die Anzeige auf 8 LEDs. Wählen Sie dazu am besten LEDs von Ampeln, so dass möglichst ein zusammenhängendes 8-Feld entsteht.

A11.7 Denken Sie an die "unmatched LOC constraints".

A11.8 Laden Sie den FPGA mit Ihrer Schaltung

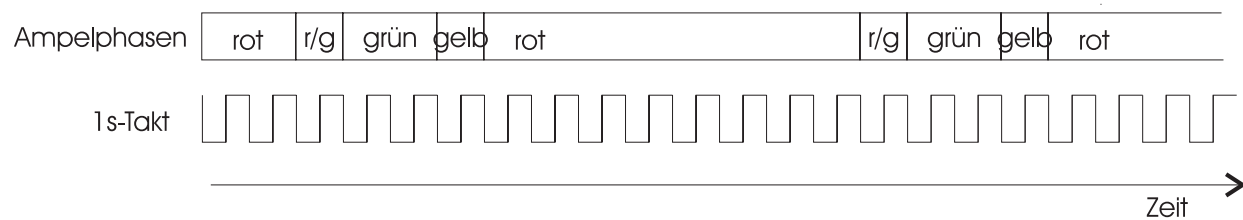
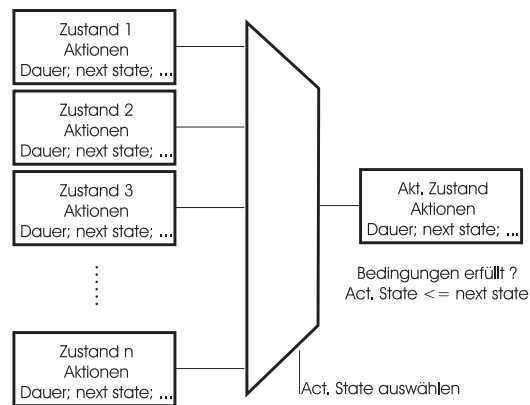
11.2 Pinbelegung

Signal	Bauteil
CLK	clock
reset	Button_T53
sel(0)	switch_T46
sel(1)	switch_T47
Anzeige(0)	LED_T19_green
⋮	⋮
Anzeige(7)	LED_T20_green

12 Aufgabe A12

Ampelsteuerung I

Es soll eine Ampelsteuerung, die ihren Zustand im Sekundentakt ändert, entwickelt werden. Verwenden Sie dazu einen Zustandsautomaten (finite state machine; FSM), um die drei Farben einer Ampel zu steuern.



Lernziel:

- VHDL
 1. Finite-State-Machine
 2. Zustandsvariable
- Design Flow
 1. Schematic WDH.
 2. Testbench WDH.

3. User Constraints Pin-Zuordnung WDH.

12.1 Aufgaben

A12.1 Entwickeln Sie den VHDL-Code für die Ampelphasen und synthetisieren Sie die Schaltung. Benutzen Sie einen Untersetzer, um die Dauer der einzelnen Phasen der Ampeln aus dem 10Hz-Takt festzulegen.

A12.2 verwenden Sie eine Testbench, um die Funktion der Ampel zu überprüfen.

A12.3 Verwenden Sie den 0,1s-Taktgeber, um die Zustände weiterzuschalten.

A12.4 Instantisieren Sie Ampel und Taktgeber

A12.5 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Ampel

A12.6 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine und den Reset mit Taster T_54.

A12.7 Denken Sie an die "unmatched LOC constraints".

A12.8 Laden Sie den FPGA mit Ihrer Schaltung

12.2 Pinbelegung

Signal	Bauteil
rot	LED_T40_red
gelb	LED_T60_yellow
gruen	LED_T20_green
10Hz	LED1
CLK	clock
reset	Button_T53

13 Aufgabe A13

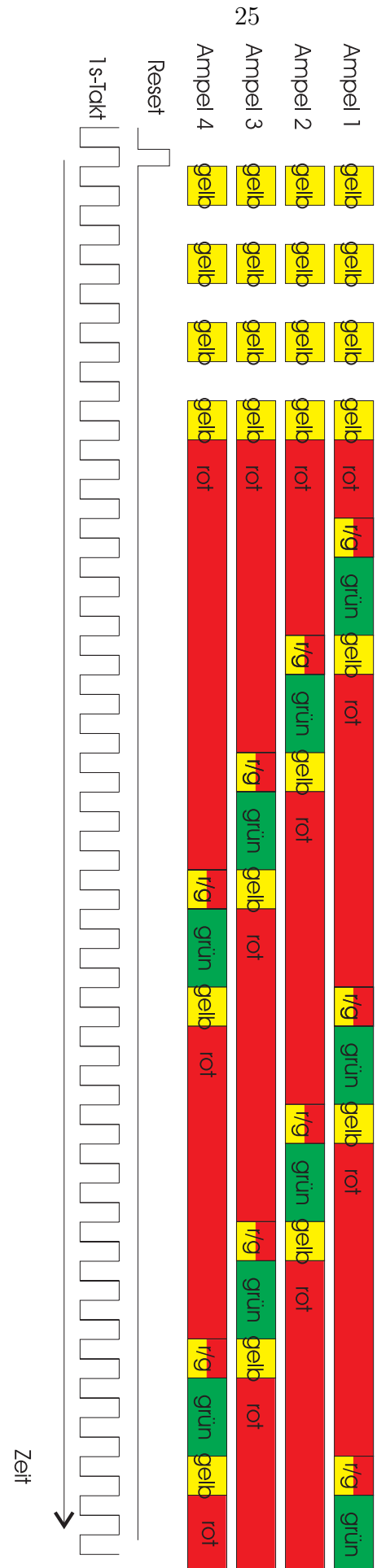
Ampelsteuerung II

Die Ampelsteuerung der vorhergehenden Aufgabe soll auf vier Ampeln erweitert werden.
 Die Phasen aller Ampeln laufen identisch ab, jedoch müssen die Phasen versetzt und miteinander synchronisiert werden.

Den Ablauf zeigt das nebenstehende Diagramm
 Nach Reset starten die Phasen von Neuem.

Lernziel:

- VHDL
 1. Finite-State-Machine WDH.
 2. Zustandsvariable WDH.
- Design Flow
 1. Schematic WDH.
 2. Testbench WDH.
 3. User Constraints Pin-Zuordnung WDH.



13.1 Aufgaben

- A13.1 Ergänzen Sie die Ampel der vorhergehenden Aufgabe durch ein zusätzliches Startsignal sowie einen Idle-Ausgang.
- A13.2 Nach Reset soll die Ampel in den Zustand "gelb-blinkend" versetzt werden. Nach 20 bis 30-mal Blinken geht die Ampel auf rot und in den Idle-Zustand. Aus Idle aufgeweckt wird die Ampel mit dem Startsignal.
- A13.3 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung.
- A13.4 Entwickeln Sie eine Schaltung, die 4 einzelne Ampeln enthält. Mit einer Finite-State-Machine sollen die vier Ampeln angesteuert werden. Die Synchronisierung der Ampeln erfolgt über die Signale IDLE und START.
- A13.5 verwenden Sie eine Testbench, um die Funktion der Ampel zu überprüfen.
- A13.6 Verwenden Sie den 0,1s-Taktgeber, um die Zustände weiterzuschalten. Dazu instantisieren Sie die 4-fach-Ampel und den Taktgeber und verbinden den 10Hz-Ausgang mit dem CLK-Eingang der Ampelsteuerung
- A13.7 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Ampelanlage
- A13.8 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine, den Reset mit Taster T_54 sowie den 10Hz-Takt mit der LED1. Die Anschlüsse für die Ampellampen der 4 Ampeln entnehmen Sie aus den Definitionen in `Spartan2E_CHIP.vhd` oder der Tabelle in der Anleitung zu Versuch 5.
- A13.9 Denken Sie an die "unmatched LOC constraints".
- A13.10 Laden Sie den FPGA mit Ihrer Schaltung und testen Sie das Verhalten

14 Aufgabe A14

Ampelsteuerung III

Erweitern Sie die Ampelsteuerung um eine Fußgängerampel :

Nach Betätigung einer Fußgängeranforderungstaste soll - bevor die nächste gelbgrün-Phase beginnt, eine Fußgängerphase eingebaut werden.

Jedoch müssen alle Ampeln mindestens einmal eine Grünphase gehabt haben !

Lernziel:

- VHDL
 1. Finite-State-Machine WDH.
 2. Zustandsvariable WDH.
 3. generic
- Design Flow
 1. Schematic WDH.
 2. Testbench WDH.
 3. User Constraints Pin-Zuordnung WDH.

14.1 Aufgaben

A14.1 Ergänzen Sie die Ampel der vorhergehenden Aufgabe durch ein ”**generic**”.

Sie können eine Baugruppe in verschiedenen Ausführungsformen programmieren. Mit dem generic können Sie festlegen, welche reale Baugruppe letztlich erzeugt wird.

In unserem Fall soll die Ampel entweder eine normale Ampel oder eine Fußgängerampel entstehen.

A14.2 Nach Reset soll die normale Ampel wieder in den Zustand ”gelb-blinkend” versetzt werden, die Fußgängerampel soll rot anzeigen.

Auch im Ablauf weicht die Fußgängerampel ab. Sie startet sofort mit grün und springt danach wieder nach rot zurück.

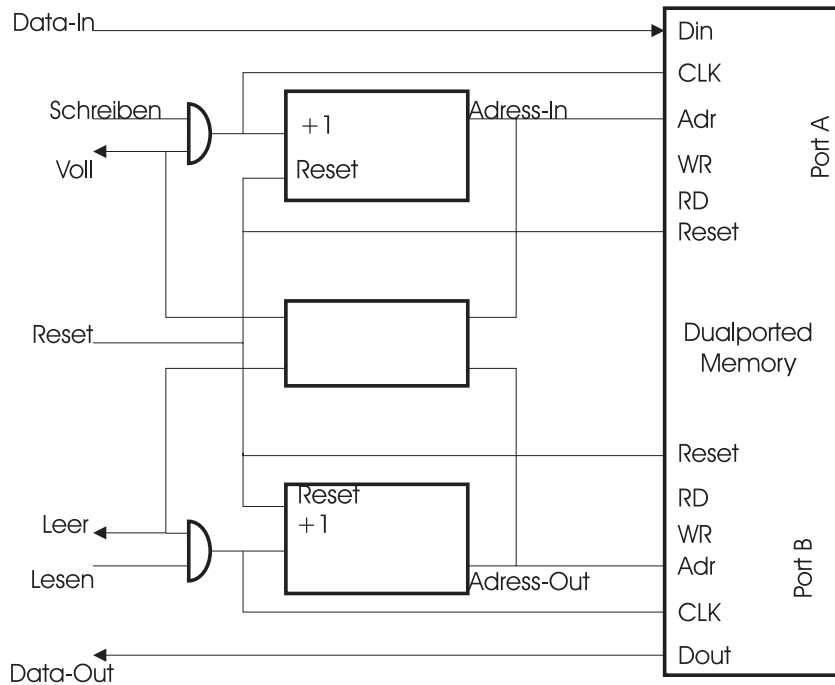
A14.3 Entwickeln Sie den VHDL-Code und synthetisieren Sie die Schaltung.

- A14.4 Entwickeln Sie eine Schaltung, die 5 Ampeln (4 normale sowie eine Fußgängerampel) enthält. Wieder werden mit einer Finite-State-Machine die fünf Ampeln angesteuert.
- A14.5 Ein Signal "Anforderung erkannt" soll anzeigen, dass eine Fußgängeranforderung erkannt aber noch nicht ausgeführt wurde
- A14.6 verwenden Sie eine Testbench, um die Funktion der Ampel zu überprüfen. Die Testbench soll die Signale "Reset", "CLK" sowie zeitverzögert "Fußgängeranforderung" simulieren
- A14.7 Verwenden Sie den 0,1s-Taktgeber, um die Zustände weiterzuschalten. Dazu instantisieren Sie die 5-fach-Ampel und den Taktgeber und verbinden den 10Hz-Ausgang mit dem CLK-Eingang der Ampelsteuerung
- A14.8 Laden Sie eine Kopie von `Spartan2E_CHIP.vhd` und `Spartan2E_CHIP.ucf` und instantisieren Sie Ihre Ampelanlage
- A14.9 Verbinden Sie den CLK-Eingang der Schaltung mit dem 50MHz-Oszillator der Testplatine, den Reset mit Taster T_54, Fußgängeranforderungstaste mit T_53 sowie den 10Hz-Takt mit der LED1. "Anforderung erkannt" zeigen Sie mit der LED T6 an. Die Anschlüsse für die Ampellampen der 5 Ampeln entnehmen Sie aus den Definitionen in `Spartan2E_CHIP.vhd` oder der Tabelle in der Anleitung zu Versuch 5.
- A14.10 Denken Sie an die "unmatched LOC constraints".
- A14.11 Laden Sie den FPGA mit Ihrer Schaltung und testen Sie das Verhalten

15 Aufgabe A15

FIFO

Ein FIFO (First In First Out) ist ein Speicher, in dem Daten nacheinander eingespei-



chert werden und in der gleichen Reihenfolge wieder entnommen werden. Es dient zur Synchronisation von Systemen, die unabhängig voneinander Daten liefern bzw. abholen.

Dieses FIFO enthält ein dual ported Memory (über zwei unabhängige Ports können Speicherzelle angesprochen werden). Ein Port dient zum Lesen, das andere zum Schreiben von Daten. Zwei Adresszähler verwalten die beiden Adressen unabhängig voneinander. Nur ein Vergleichler überwacht den Zustand des FIFOs (voll oder leer). Ist das FIFO nicht voll, darf geschrieben und ist das FIFO nicht leer kann gelesen werden.

Lernziel:

- VHDL
 1. Finite-State-Machine
 2. Zustandsvariable

- Design Flow
 1. Schematic
 2. Testbench
 3. User Constraints Pin-Zuordnung

15.1 Aufgaben

A15.1 Entwickeln Sie den VHDL-Code für ein FIFO zuerst ohne das dual ported RAM mit einem Adressraum von nur 4bit.

A15.2 Testen Sie Ihren Code, in dem Sie Ihr FIFO in den beigefügten Testcode `FIFOtst.vhd` einbinden. Dieser Testcode erzeugt (quasi unabhängig) Lese- und Schreibbefehle. Mit den Testsschaltern lassen sich die Geschwindigkeiten dieser Lese- und Schreibbefehle variieren.

A15.3 Verwenden Sie den Logic-Analyzer, um das Verhalten bei vollem und leerem FIFO zu untersuchen. Über das Port `DATAout` können Sie Parameter an die testende Instanz übergeben und mit dem Logic-Analyzer sichtbar machen.

A15.4 Instantisieren Sie das Dual-ported-RAM "RAMB4_S8_S8"

15.2 Pinbelegung

FPGA	I/O	Bauteilname
P80	50MHz-Oszillator	
P33	Taster (Reset)	T42