Praxissemesterbericht

"Inbetriebnahme eines Spartan 3E FPGA Evaluations Boards"

Torsten Krause

10. Oktober 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Die Max-Planck-Gesellschaft

Die Max-Planck-Gesellschaft zur Förderung der Wissenschaften e.V. (kurz MPG) wurde am 26. Februar 1948 gegründet. Sie ist eine unabhängige und gemeinnützige Forschungsorganisation.
Gefördert werden vorrangig die 81 eigenständigen Forschungsinstitute, so z.B. für Plasmaphysik, für experimentelle Medizien oder für terrestrische Mikrobiologie.
Diese forschen in „Natur-, Bio-, Geistes- und Solzialwissenschaften im Dienste der Allgemeinheit“.
Einige Max-Planck-Institute stellen ihre aufwendigen Einrichtungen und Geräte zur Mitnutzung den Forschern deutscher Universitäten zur Verfügung.
Insgesamt haben die 81 Institute rund 12.600 Mitarbeiter und 11.300 Doktoranten, Postdoktoranten, Gastwissenschaftler und studentische Hifskräfte.

1.2 Das Max-Planck-Institut für Radioastronomie

Die Arbeit des 1966 gegründeten Max-Planck-Instituts für Radioastronomie (kurz: MPIfR) beruht wesentlich auf Beobachtungen, die am Radio-Observatorium Effelsberg durchgeführt werden.
Da die vom Radioteleskop empfangenen Signale sehr schwach sind und das Umgebungsrauschen um ein vielfaches stärker ist, entwickelt die Elektronikabteilung extrem rauscharme, stark gekühlte Verstärker.
Regelmässig beteiligt sich das Institut an verschiedenen internationalen Netzwerken, bei denen mit Hilfe vieler Teleskope ein virtuelles Radioteleskop mit einem Durchmesser bis zu mehrerer Tausend Kilometern simuliert wird.
Diese Radiointerferometriebeobachtungen erreichen die höchsten räumlichen Auflösungen die zur Zeit möglich sind.

1.3 Das Radioteleskop Effelsberg

Das 100m-Radioteleskop Effelsberg (Abb.1) ist das zweitgrößte vollbewegliche Teleskop der Welt. Es ist seit 1972 in Betrieb und wird seitdem für radioastronomische Beobachtungen genutzt.
2352 Paneelen bilden die $7850m^2$ große Antennenoberfläche. Das Teleskop hat ein Gesamtgewicht von 3200t und kann mit einer Genauigkeit von 0,3 mm in 12 Minuten eine 360° Drehung vollziehen.
Radiowellen mit Wellenlängen zwischen 90 cm und 3,5 mm können empfangen und ausgewertet werden. Der Standort wurde in einem Tal mit geringer Bewohnung gewählt um den Einfluss von Fremdstrahlung zu minimieren.
Hauptbeobachtungsziele sind:

- Pulsare
- Schwarze Löcher
- Galaxiekerne
- kalte Gas- und Staubwolken

Abbildung 1: Radioteleskop Effelsberg

2 Grundlagen

2.1 Field Programmable Gate Array

Ein Field Programmable Gate Array (kurz: FPGA) ist ein Baustein, mit dessen Hilfe logische Schaltungen verwirklicht werden. Ein FPGA besteht hauptsächlich aus drei Komponenten:

- matrixförmig angeordnete Logigblöcke
- einem Verbindungsnetz (Routing-Kanäle)
- Ein- und Ausgangsblöcke

Die programmierte Funktion wird aus den Logigblöcken, meist Flip-Flops und Lookup-Tables, über die Routing-Kanäle verknüpft. Wobei die Flip-Flops als Speicher dienen, um den aktuellen Signalwert im nächsten Takt weiterverarbeiten zu können und die Lookup-Tabels die Funktion mit Hilfe der Grundlogigfunktionen (AND, OR, XOR, ...) „berechnet“. Abbildung 2 zeigt das FPGA auf dem Spartan 3E.

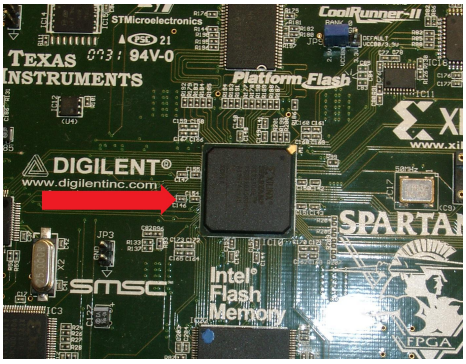


Abbildung 2: FPGA

2.2 Very High Speed Integrated Circuit Hardware Decription Language

Durch die Hardwarebeschreibungssprache „Very High Speed Integrated Circuit Hardware Decription Language“ (kurz: VHDL) ist es möglich, digitale Systeme zu beschreiben. Selbst komlizierte Schaltungen können übersichtlich dargestellt und simuliert werden.
Entwickelt wurde VHDL um eine standatisierte Dokumentation von Funktion und Struktur integrierter Schaltung zu erhalten. So ist es möglich bestehende Programme in verschiedene Bausteine zu laden oder mit verschiedenen Programmen zu simunlieren.

3 Aufgabenstellung

„ Inbetriebnahme eines Spartan 3E FPGA Evaluations Boards “

Aufgabe des Praxissemesters war die Datenerfassung und Verarbeitung mit einem FPGA. Dazu wurde mir das Starterkit Spartan 3E der Firma Xilinx (Abb.3) zur Verfügung gestellt.

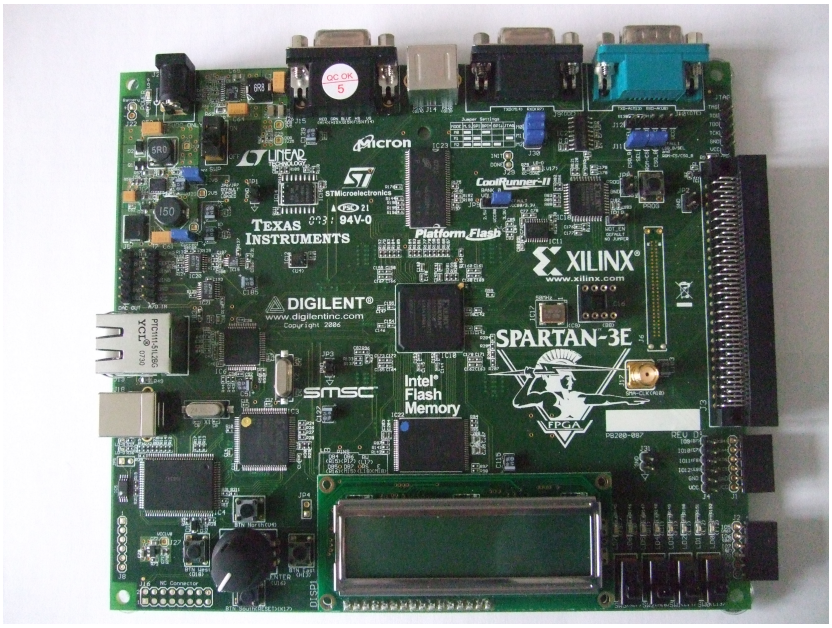


Abbildung 3: Xilinx Spartan 3E

Anmerkung:
Um die Grundlangen der FPGA-Programmierung in VHDL zu erarbeiten, wurde nicht ein großes Projekt, sondern viele Kleine verwirklicht. Am Ende jedes Projekts wurde ein Signalverlaufplan erstellt. Ein Siganlverlaufplan verdeutlicht den Aufbau des Programms und den Verlauf der einzelnen Signale. Hierbei steht das große grüne Rechteck für das Projekt. Jedes weiße Rechteck symbolisiert eine Instanz, welche durch Signalpfade verbunden sind. Die Signalnamen sind in Pink, die Ein- und Ausgänge der Instanzen in Grün und die Ein- und Ausgänge des Projekts in Blau beschrieben.

3.1 Takt teilen

Aufgabenstellung:
Runterteilen des vom FPGA-Board gelieferten Taktsignals um einen beliebigen Faktor

Beschreibung:
Das FPGA-Board ist mit 50 MHz getaktet. Dieser Takt wird um den Faktor 25.000.000 geteilt und auf ein LED gelegt, sodass diese im Takt blinkt.
Wie im Signalverlaufplan dargestellt, besteht das Projekt ledigtlich aus einer Instanz und je einem Ein- bzw. Ausgangssignal. Das Eingangssignal ist das ungeteilte und das Ausgangssignal das geteilte Clock-Signal.
In den Zeilen 5 bis 8 der Instanz „TOP“ wurden die Bibliotheken eingebunden. Die hier verwendeten sind Standard und wurden nicht verändert.
Es folgt (Zeilen 10 bis 13) die Entity, diese beschreibt die Schnittstellen nach Außen. In der Entity sind Signale deklariert, denen jeweils eine Richtung zugeordnet wird. Es gibt drei gebräuchliche Richtungen: in (Eingangssignal), out (Ausgangssignal) und inout (Bidirektionales Signal).
Das Signal „clock“ ist STD_ LOGIC, d.h. es nimmt nur die binären Werte 1 oder 0 an. In diesem Signaltyp sind auch noch andere Signalwerte definiert, wie z.B. H (High), L (Low)oder Z (Hochohmig). In allen folgenden Projekten werden nur die Signalwerte 1 und 0 genutzt.
Das Signal „leds_out“ ist vom Typ STD_LOGIC_VECTOR, also ein binärer Bus der Breite 8. Dieser Bus hätte anstatt wie hier abfallend (7 downto 0) auch aufsteigend (0 to 7) bezeichnet werden können. Hierbei wird nur die Durchnummerierung der einzelnen Bits des Vektors festgelegt.
Das restliche Programm (Zeilen 15 bis 29) ist die Architektur, welche das Innenleben, d.h. die Funktionalität des VHDL-Codes, beschreibt. Jeder Entity muss mindestens eine Architecture zugeordnet sein.
In dieser Architecture ist ein Prozess verwirklicht. In einem Prozess werden aus den Eingangssignalen die Ausgangssignale mit Hilfe der Signalzuweisungen errechnet. In Zeile 17 steht die „Sensitivity-Liste“, bei jeder Änderung eines der hier genannten Signale wird der Prozess einmal durchlaufen.
Im User-Constrant-File (kurz UCF) werden, die in der Entity beschriebenen Signale mit den physikalischen Ein- und Ausgangspins des FPGAs verbunden.

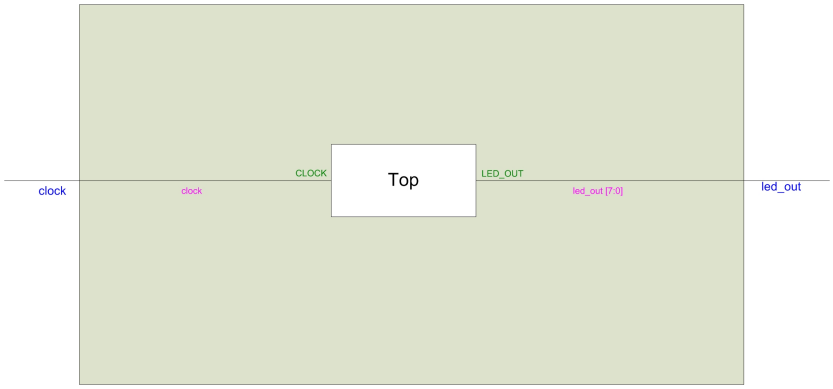


Abbildung 4: SignalverlaufIpan: Takt teilen

3.2 Ampelschaltung

Aufgabenstellung:
Realisierung einer Ampelschaltung, einbinden mehrerer Instanzen

Beschreibung:
Das Programm realisiert eine einfache Ampelschaltung mit einer Reset-Funktion. Das Projekt besteht aus der Top-Entity und drei Instanzen. In der Top-Entity werden die Signalkopplungen von mehreren Komponenten in einer Architektur erzeugt, also nur die Schnittstellen der einzelnen Instanzen untereinander und zu den Systemgrenzen. In diesem Projekt sind zwei Top- Entities vorhanden, „Schem“ und „Top“, die von der Funktionalität identisch sind. „Schem“ wurde mit Hilfe einer Funktion von ISE aus einer erstellten Grafik erzeugt. ISE ist die von Xilinx mitgelieferte Entwicklungsumgebung. In Zeile 23 der Instanz „Top“ ist ein Signal deklariert, welches nur innerhalb dieser Architektur existiert und nicht mit einem Ein- oder Ausgangspin belegt werden kann.
Die Instanz „Divider“ hat dieselbe Funktion wie die Instanz „Top“ des vorhergehenden Projekts „Takt teilen“.
Das Zählen der positiven Flanken des „runtergeteilten“ Taktes, sowie das zurücksetzen bei einer positiven Signalflanke des Reset-Signals wurde in der Instanz „Counter“ beschrieben.
Die Instanz „stat_mac“ wurde ebenfalls aus der oben genannten Grafik durch das ISE erzeugt. Diese Funktion ist aber umständlich und bei komplexeren Projekten sehr kompliziert, außerdem ist eine eventuelle Fehlersuche problematisch. Aus diesen Gründen wurde diese Funktion von ISE in den folgenden Projekten nicht mehr genutzt.

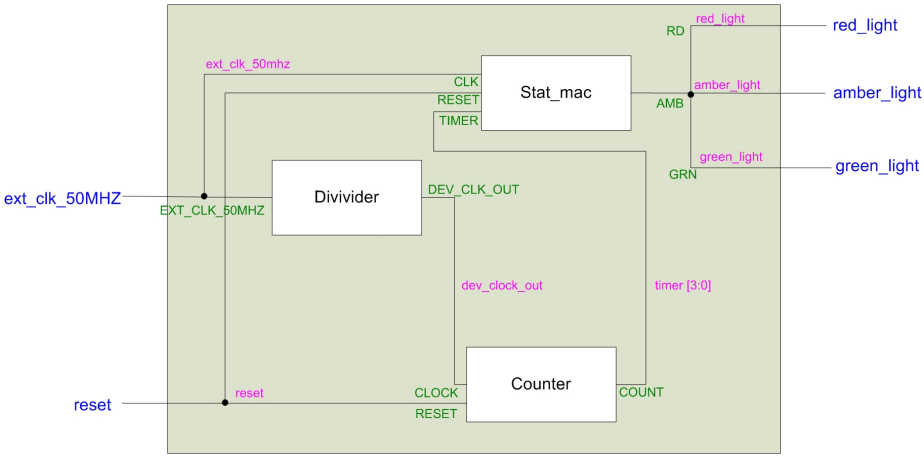


Abbildung 5: SignalverlaufIpan: Ampelschaltung

3.3 Binär zählen

Aufgabenstellung:
Erzeugung eines Ausgangsvektors, diesen verändern und an LEDs ausgeben

Beschreibung:
Dieses Projekt enthält wie die vorhergehenden, die Instanz „Divider“ und die Top-Entity „Top“. In der Instanz „Mac“ werden mehrere Funktionen verarbeitet. Es wird ein drei Bit großer Ausgangsvektor erzeugt.
Durch Betätigen des Button East wird dieser auf 0 gesetzt. Wird South gedrückt wird binär hochgezählt und auf West binär runtergezählt.
Außerdem wurden drei Schalter eingebunden die je einem Bit des Ausgangsvektors zugeordnet wurden. Durch Betätigen des Schalters wird das betreffende Bit negiert. Zu beachten ist, dass wenn ein Taster über mehrere positive Flanken gedrückt bleibt, zählt das Programm bei jeder positiven Flanke.

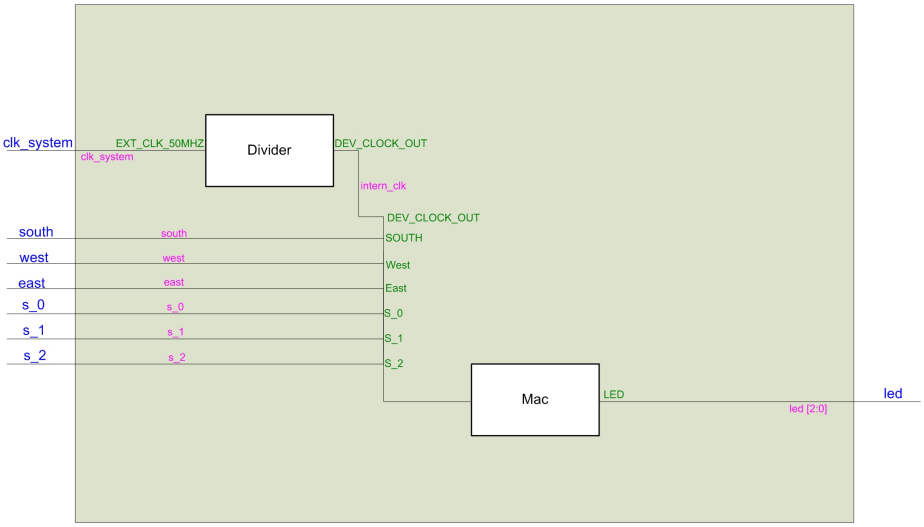


Abbildung 6: Signalverlaufplan: Binär zählen

3.4 Drehinkrementgeber

Aufgabenstellung:
Drehinkrementgeber implementieren und Prellen unterdrücken

Beschreibung:
Die Aufgabe war das Einbinden des Drehinkrementgebers. Das Programm hat, wie im Signalverlaufplan zu erkennen, drei Ausgänge. „Direction“ zeigt an in welche Richtung gedreht wird, „Position“ zeigt binär die Position an und „Test“ legt den geteilten Takt auf eine LED. Der rotary_contact_filter wurde eingefügt, da der Drehgeber prellt, d.h. dass die mechanischen Kontakte beim Drehen um eine Position durch Wippbewegungen mehrmals auslösen und so oft eine falsche Richtung anzeigen bzw. mehrere Positionen weiterzählen. Das Prinzip des Filters ist in Abb. 7 dargestellt. Durch geschickten Vergleich von „rotation_a_in“ und „rotation_b_in“ werden neue Signale erzeugt die zwar nicht den Eingängen gleich sind, aber ein ähnliches Verhalten haben und bei denen dieses Prellen nicht mehr auftritt.

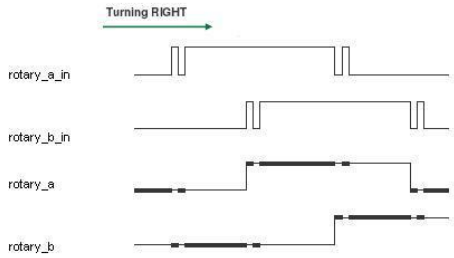


Abbildung 7: Funktionsdiagramm Rotary-Contact-Filter

In Zeile 27 der Instanz „Rotary Contact Filter“ ist eine &-Verknüpfung verwendet, dies ist aber nicht die logische AND-Verknüpfung, sondern bildet aus zwei binären Signalen einen Zwei-Bit-Vektor, wobei das erste Bit den Wert des ersten binären Signals hat und das zweite Bit des Vektors den Wert des zweiten Signals annimmt.
Außerdem mussten noch Pullup- und Pulldownwiderstände im User Constraint File (Zeilen 13 bis 15) aktiviert werden. Der Pullup-Wiederstand zieht den Pin auf Betriebsspannung und der Pulldownwiderstand zieht die Spannung auf ground.

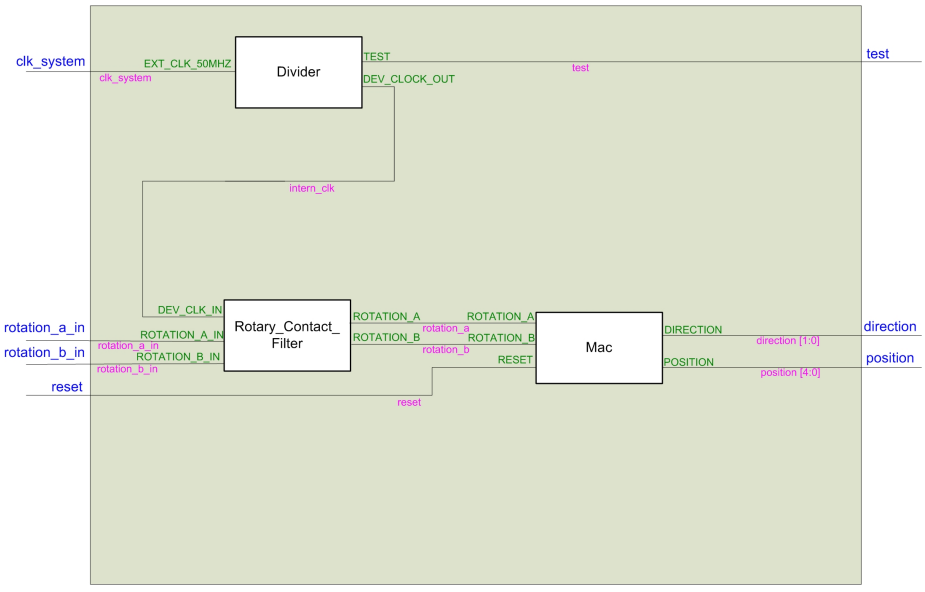


Abbildung 8: Signalverlaufplan: Drehinkrementgeber

3.5 Schnittstelle EIA 232

3.5.1 Konstante senden

Aufgabenstellung:
Senden einer Konstanten und Baudrate festlegen

Beschreibung:
Es sollte in diesem Projekt das Senden einer Konstanten über die serielle Schnittstelle EIA-232 programmiert werden. Die Schnittstelle EIA-232 (früher RS-232) ist eine spannungsgesteuerte Schnittstelle, die asynchron und bitseriell arbeitet, d.h. die einzelnen Bits werden nacheinander über eine Leitung gesendet und nach einem Wort kann eine beliebig lange Zeit vergehen, bis das Nächste gesendet wird. Hierbei war zu beachten, das jedes Datenwort mit einem Startbit (Logisch 0) „angekündigt“ und mit einem Stopbit (Logisch 1) zu versehen war(Abb.9).
Auf das Senden des Paritätsbits, das zur Vermeidung von Fehlern genutzt wird, wurde verzichtet.
Des Weiteren wurde eine negative Logik verwendet. Der Zustand logisch 1 wird in die negative Spannung -15 V und der Zustand logisch 0 in die positive Spannung +15 V übersetzt. Zur Erkennung des Startbits muss immer, wenn nicht gesendet wird, eine logische 1 am Ausgang liegen. Dies wurde im User Constraint File (kurz UCF) in der sechsten Zeile realisiert.
Es war noch die Baudrate zu wählen, diese legt die zeitliche Dauer eines Bits fest. Sender und Empfänger müssen eine gemeinsame Baudrate haben, da sonst keine Kommunikation möglich ist. Die Baudrate wurde auf 9600 Baud festgelegt, was einer Dauer von 104 μs entspricht. Im Divider (Zeile 29) musste die richtige Frequenz eingestellt werden.
Diese errechnet sich:

$$f = \frac{t_B \cdot f_{syst}}{2} = \frac{104\mu s \cdot 50MHz}{2} = 2600$$

In der Instanz „Mac“ wurde das Senden umgesetzt. Das Programm sendet nach und nach jedes Bit, der Baudrate entsprechend, an den Ausgang.
In den Zeilen 21 bis 23 der Instanz „Mac“ wurden lokale Signale deklariert. Diese sind nur intern, in dieser Architecture sichtbar und können nicht mit Ein- oder Ausgängen belegt werden. Die beiden Signale „counter_index“ und „help“ sind vom Typ integer. Integer-Signale nehmen keine binären Werte an, sondern ganzzahlige von -2.147.483.648 bis 2.147.483.647. Dem Signal „help“ wurde in der Instanz “Mac“ Zeile 23 der Anfangswert 0 zugewiese und sorgt dafür, dass die Bitfolge nur einmal gesendet wird.
Die „elsif“-Anweisung der Instanz „Mac“ (Zeilen 41 bis 52) realisiert zum Einen, dass die Bitfoge nur einmal gesendet wird und zum Anderen, dass die Bitfolge komplett gesendet wird, wenn zwischenzeitlich der Taster losgelassen wurde.

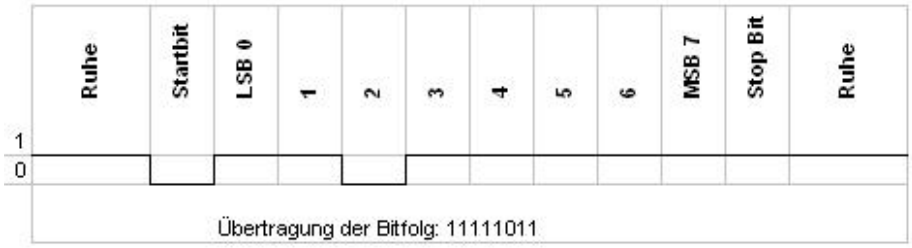


Abbildung 9: EIA-232 Übertragung

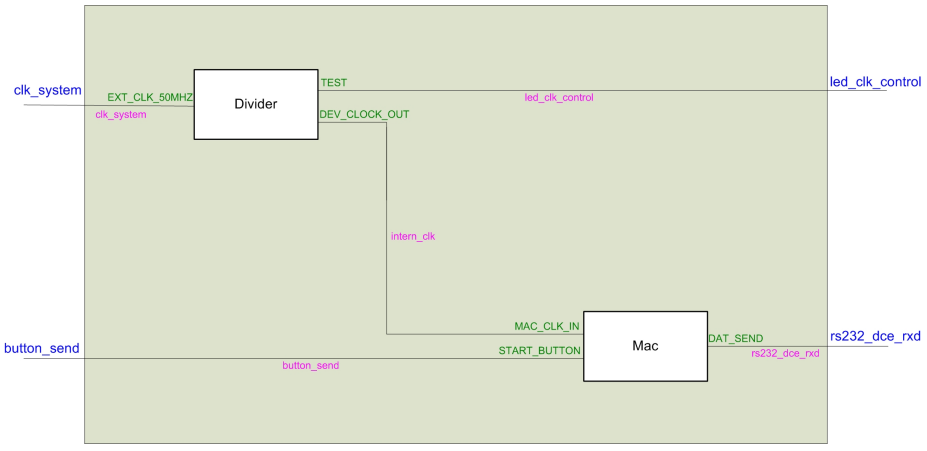


Abbildung 10: Signalverlaufplan: Konstante senden

3.5.2 Variable senden

Aufgabenstellung:
Senden einer Variablen und Drehinkrementgeber einbinden

Beschreibung:
Im nächsten Schritt wurde die Konstante aus dem vorherigen Projekt variabel gestaltet. Zum Einstellen des Signals „dat_send_binaer“ wurde der Drehinkrementgeber wieder eingebunden und zur optischen Kontrolle jedes Bit auf eine LED ausgegeben.
In der Instanz „Mac“ Zeile 41 wird das Startbit gesetzt und in den Zeile 32 und 33 das Stopbit.
Die Instanz „Set“ verwirklicht das Auswerten der Drehbewegung und die Verknüpfung des eingestellten Vektors mit den LEDs. Eine Reset-Funktion ist auch implementiert (Zeilen 33 und 34). Das FPGA ermöglicht eine parallele Abarbeitung der einzelnen Instanzen, daher gibt es keine Probleme beim Zusammenführen der Signale.

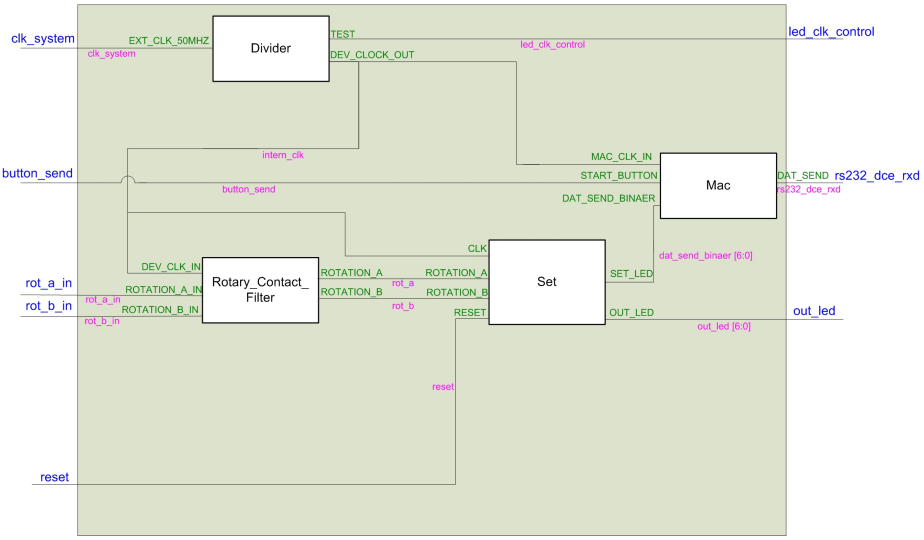


Abbildung 11: Signalverlaufplan: Variable senden

3.5.3 Senden mehrerer Zeichen

Aufgabenstellung:
Mehrere Zeichen senden

Beschreibung:
Bei dieser Aufgabe wurde das mehrfach hintereinander senden verschiedener Wörter programmiert.
Hierzu musste das Signal „help“ in der Instanz „Mac“ wieder auf 0 gesetzt werden, sobald der start-button-Knopf gedrückt wird. Dies war aber in der Architecture der Instanz „Mac“ nicht möglich, da es zu Multi-Source-Fehlern kommt, d.h. das ein Signal nicht mehrere Zuweisungen in einer Instanz haben kann.
Deshalb wurde eine weitere Instanz, „set_help“, implementiert. Diese führt nur diese einfache Zuweisung durch. Das Signal „help“ musste noch von einem lokalen zu einem globalenen Signal umdeklariert werden, da es in zwei Instanzen genutzt wurde und dort seinen Wert ändern kann.

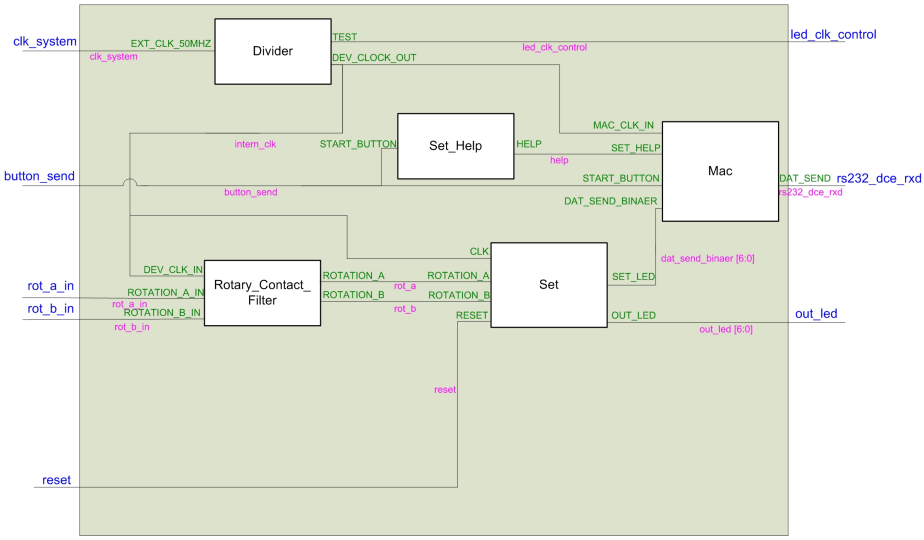


Abbildung 12: Signalverlaufplan: Senden mehrerer Zeichen

3.5.4 Konstante empfangen

Da die serielle Schnittstelle nicht nur senden sondern auch empfangen kann, sollte in diesem Projekt dieses verwirklicht werden. Dazu wurde das Eingangssignal 16-fach überabgetastet und eine Mehrheitsentscheidung programmiert. In jedem der 16 Werte pro empfangenem Bit wird der aktuelle Wert gespeichert. Besitzen mehr als 8 der 16 Werte „High“-Pegel, so wird dem Bit der Wert 1 zugeordnet, sonst 0.

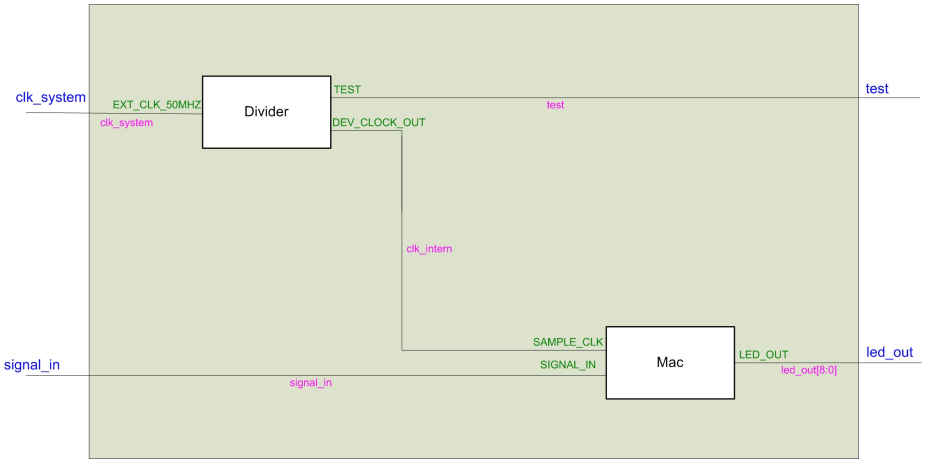


Abbildung 13: Signalverlaufplan: Konstante empfangen

3.5.5 Senden und Empfangen

Aufgabenstellung:
Verbinden der Sende- und Empfangsfunktion
Beschreibung:
In diesem Projekt wurde die Sende- und die Empfangsfunktion miteinander verknüpft. Es mussten zwei Instanzen von „Divider“ eingebunden werden, da das Abtasten des Eingangssignals eine 16-mal schnellen Takt benötigt als das Senden. Beim Testen des Programms fiel auf, dass in unregelmäßigen Abständen zu sendende Bits anstatt 1 den Wert 0 oder umgekehrt annahmen. Dieser Fehler konnte behoben werden, indem in der Instanz „Mac“ die internen Signale zu Variablen umdeklariert wurden (Zeilen 26 bis 31). Variablen nehmen immer sofort den neuen Wert an und Signale ändern ihren Zustand erst beim Verlassen des Prozesses, also zum nächsten Takt.

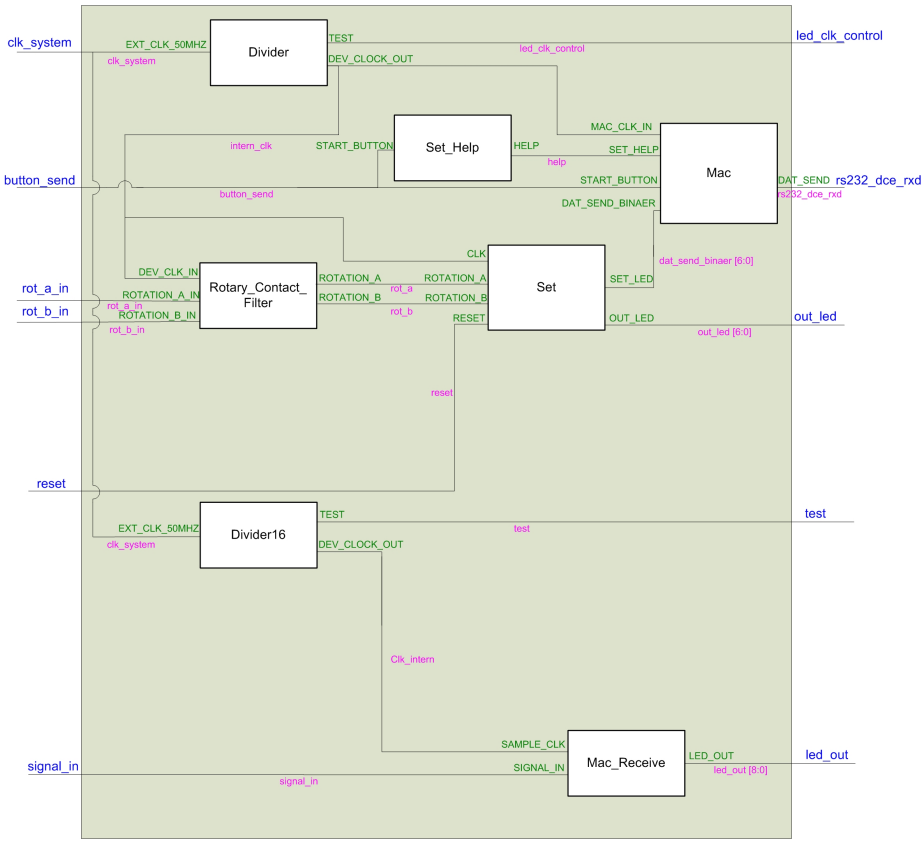


Abbildung 14: Signalverlaufplan: Senden und Empfangen

3.6 LC-Display

Aufgabenstellung:
LC-Display Initialisieren und Konstante ausgeben

Beschreibung:
Das LC-Display wird über die vier Signale „lcd_dat“ (4 Bit Bus), „lcd_e“, „lcd_rw“ und „lcd_rs“ gesteuert. Das Display reagiert auf steigende Flanken von Enable (lcd_e). Durch die „when others “ Anweisung in Zeile 117 wird es immer wieder auf 0 zurückgesetzt. Liegt „lcd_rw“ auf Low, handelt es sich um einen Schreibbefehl, sonst um einen Lesebefehl. “Lcd_rs“ entscheidet ob Daten aus dem Statusregister oder aus dem Textpuffer ausgegeben werden. Vor der Inbetriebnahme des Displays muss eine Initalisierung (Abb.15) durchgeführt werden. Diese ist in der Instanz „Mac“ in den Zeilen 31 bis einschließlich 59 geschehen. Es folgt (Zeilen 60 bis 114) das Senden der Bitfolgen, die die Ausgabe steuern, diese entsprechen den Ascii-Zeichen (Abb. 16). Der Zähler „count“ sorgt dafür, dass die notwendigen Zeitabstände zwischen den einzelnen Befehlen eingehalten werden.

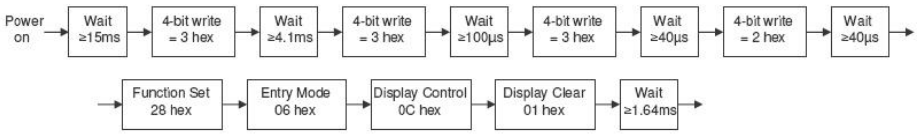


Abbildung 15: Initialisierung

Hohe 4 Bit / Tiefe 4 Bit	0010	0011	0100	0101	0110	0111
0000		0	@	P	`	p
0001	!	1	A	Q	a	q
0010	"	2	B	R	b	r
0011	#	3	C	S	c	s
0100	$	4	D	T	d	t
0101	%	5	E	U	e	u
0110	&	6	F	V	f	v
0111	'	7	G	W	g	w
1000	(	8	H	X	h	x
1001	)	9	I	Y	i	y
1010	*	:	J	Z	j	z
1011	+	;	K	[	k	{
1100	,	<	L	¥	l	|
1101	-	=	M	]	m	}
1110	.	>	N	^	n	→
1111	/	?	O	_	o	←

Abbildung 16: ASCII-Tabelle

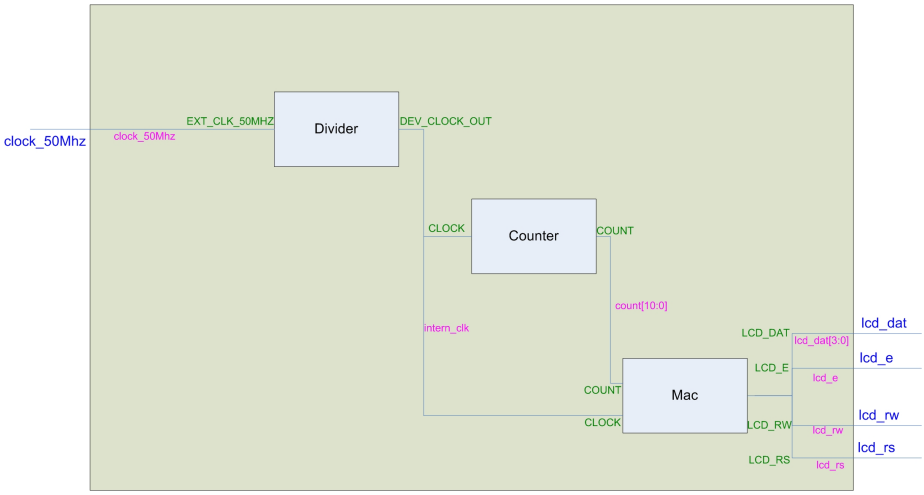


Abbildung 17: Signalverlaufplan: LC-Display

3.7 Vorverstärker

3.7.1 Verstärkung einstellen

Aufgabenstellung:
Vorverstärker ansteuern und Verstärkung einstellen

Beschreibung:
Der Vorverstärker regelt die Eingangsspannung des Analog-Digital-Wandlers auf Pegel, die den Bausteinparametern entsprechen.
Auf dem FPGA-Board sind zwei Ausgangspins mit Signalen belegt, die zum Analog-Digital-Wandler führen, die unterschiedlich verstärkt werden können. Angesteuert wird der Vorverstärker über den Serial-Peripheral-Interface-Bus (kurz: SPI-Bus).
Wie in Abbildung 18 dargestellt, mussten die drei Ausgangssignale „sck_out“, „amp_cs“ und „sdi“ geregelt werden. Das Signal „amp_do“ ist ein Kontrollsignal, an dem abzulesen ist, was der Vorverstärker bei vorherigem Einstellen erkannt hat.
„Amp_cs“ ist im „Ruhezustand“ immer auf high, sobald das Signal auf low wechselt ist der Baustein aktiv. Das Signal „sck_out“ generiert 8 Clock-Takte mit der Periodendauer 100ns und auf „sdi“ wird die Bitfolge übertragen, die die Verstärkungen einstellt. Der nächste Wert des Signals „sdi“ wird angelgt, wenn das Taktsignal 25ns den Wert 0 hat, damit er bei der nächsten steigenden Flanke von „sck_out“ anliegt.
Anschließend wird wieder der „Ruhezustand“ hergestellt, „amp_cs“ wird wieder high, „sck_out“ und „sdi“ wieder low.
Die Instanz „Set-Help“ stellt sicher, dass der Zähler „count“ der Instanz „Count_Set_Gain“ nur einmal durchlaufen wird.

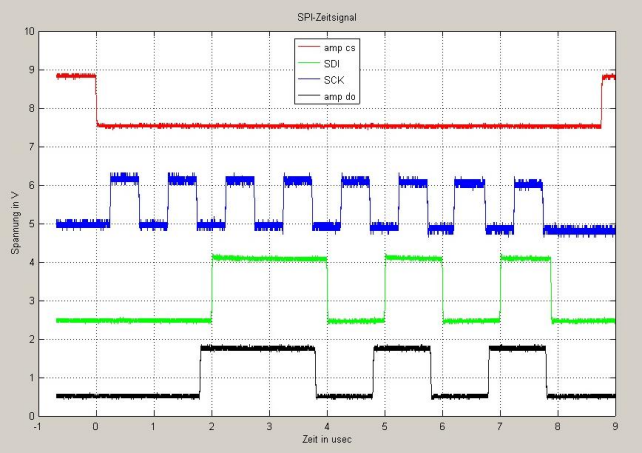


Abbildung 18: Timing-Diagramm SPI

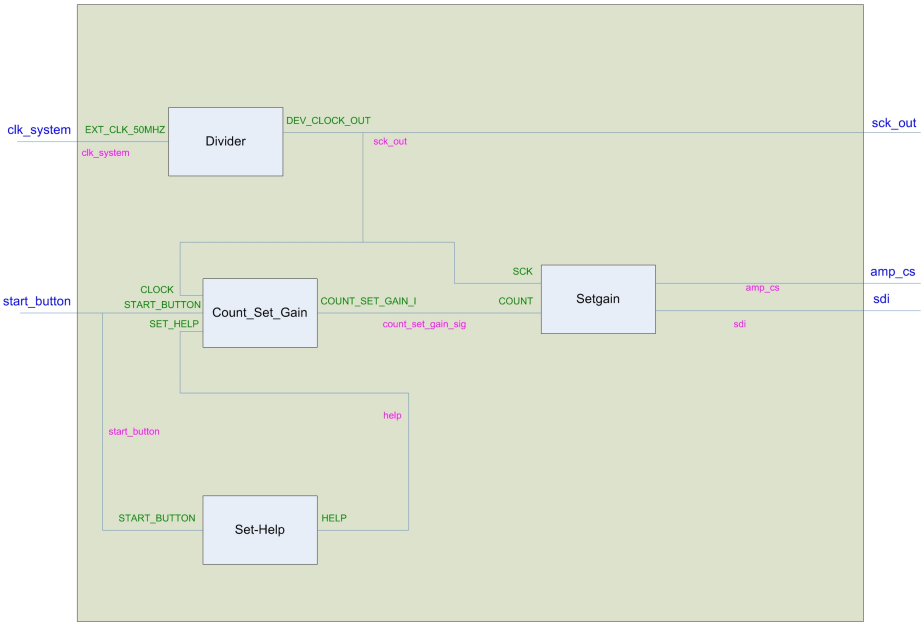


Abbildung 19: Signalverlaufplan: Verstärkung einstellen

3.7.2 Verstärkung anzeigen

Aufgabenstellung:
Verstärkungen einstellen und am LC-Display anzeigen

Beschreibung:
In diesem letzten Projekt wurden die beiden Projekte „LC-Display“ und „Verstärkung einstellen“ miteinander verbunden. Die eingestellte Verstärkung wird auf dem Dispaly angezeigt (Abb. 20). Dafür wird in der Instanz „Setgain“ das Siganl „amp_do“ gelesen und in eine Bitfolge umgewandelt. Diese wird dann in der Instanz „Set_gain_z_LCD“ umgeformt, sodass das LCD das Signal „verarbeiten“ kann und anschließend ausgeben kann. Die Initialisierung des LC-Displays und das Einstellen der Verstärkung sind gleich geblieben.

Abbildung 20: LC-Diplay

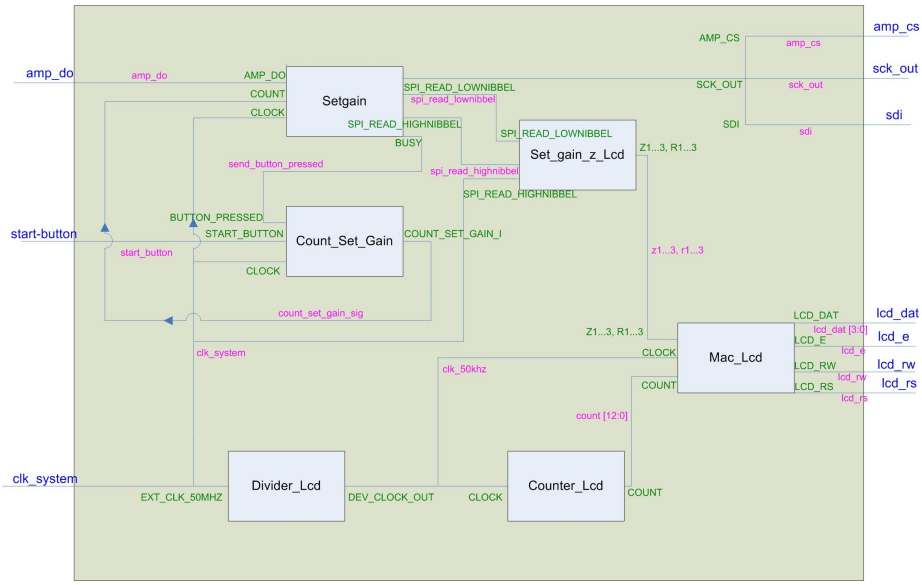


Abbildung 21: Signalverlaufplan: Verstärkung anzeigen

4 Zusammenfassung

Thema des Praxissemsters war die „ Inbetriebnahme eines Spartan 3E FPGA Evaluations Boards “. Zu dem FPGA Boards Spartan 3E der Firma Xilinx wurde die firmeneigene Entwicklungsumgebung ISE mitgeliefert, welches zur Programmentwicklung verwendet wurde .
Ziel des Praxissemsters war, sowohl für mich als auch für die betreuenden Ingenieure, einen Einblick in die Gebiete FPGA- und VHDL-Programmierung zu bekommen. Hierzu wurde nicht ein Großes, sondern viele kleine Projekte realisiert.
Nach einer kurzen Einarbeitung, in der geklärt wurde, was ein FPGA ist und wie man es programmiert, wurden die ersten Projekte verwirklicht.
Im Projekt „Takt teilen“ wurde der vom Evaluations Board gelieferte Takt von 50 MHz um einen beliebigen Faktor verringert. In „Ampelschaltung“ ging es primär um die Anstreuerung der LEDs und der Verknüpfung verschiedener Instanzen zu einem Projekt. Es folgte „Binär zählen“, hier wurde zum ersten Mal mit Vektoren gearbeitet und Taster verwendet. Als letztes Projekt das zum Block Einarbeitung gehörte, wurde „ Drehinkrementgeber“ programmiert. Dabei wurde der Drehinkrementgeber eingebunden und das Prellen unterdrückt.
Es folgten die drei Hauptaufgaben: das Programmieren der Schnittstelle EIA232, das Ansteuern des LC-Displays und das Einstellen des Vorverstärkers des Analog-Digital-Wandlers.
Über die Schnittstelle EIA232 (früher RS232) wurde zuerst schrittweise das Senden vom Board zum PC und anschließend das Empfangen von Daten vom PC umgesetzt.
Beim LC-Display war das Initialisieren des Displays die entscheidende Schwierigkeit, da es auf genaues Einhalten bestimmter Zeitabstände zwischen den einzelnen Initialisierungsbefehlen ankommt.
Als letzte Aufgabe wurde der Vorverstärker angesteuert. Hierbei wurde der Verstärkungsfaktor eingestellt, um den die Eingangsspannung verringert wurde. Dies wird zum Beispiel benötigt, damit am Analog-Digital-Wandler Signale ankommen, die seinen jeweiligen Bausteinparametern entspricht.

A Anhang

A.1 Takt teilen

led.vhd Wed Jun 18 10:51:59 2008

```
1   ----------------------------------------------------------------------------------
2   --Projekt: Takt teilen
3   --Instanz: LED
4   ----------------------------------------------------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity led is
11      Port ( clock : in  STD_LOGIC;
12             leds_out : inout  STD_LOGIC_VECTOR (7 downto 0));
13  end led;
14
15  architecture Behavioral of led is
16  begin
17     process (clock)
18     variable divide_count : integer := 0;
19     begin
20        if rising_edge(clock) then
21           DIVIDE_COUNT := DIVIDE_COUNT + 1;
22           if DIVIDE_COUNT = 25000000 then
23              LEDS_OUT(0) <= not LEDS_OUT(0);
24              DIVIDE_COUNT := 0;
25           end if;
26        end if;
27     end process;
28
29  end Behavioral;
30
31
```

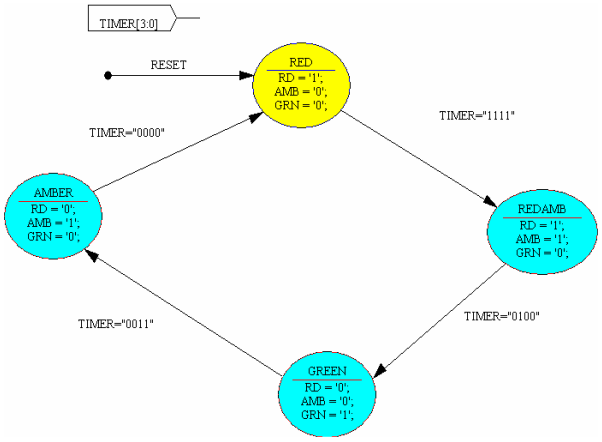
Page 1

led.ucf Wed Jun 18 10:52:16 2008

```
1   #----------------------------------------------------------------------
2   #--Projekt: Takt teilen
3   #--         User Constraint File
4   #----------------------------------------------------------------------
5
6
7   NET "clock"          LOC = "C9"  ;
8   NET "leds_out<0>"    LOC = "F12" ;
9   NET "leds_out<1>"    LOC = "E12" ;
10  NET "leds_out<2>"    LOC = "E11" ;
11  NET "leds_out<3>"    LOC = "F11" ;
12  NET "leds_out<4>"    LOC = "C11" ;
13  NET "leds_out<5>"    LOC = "D11" ;
14  NET "leds_out<6>"    LOC = "E9"  ;
15  NET "leds_out<7>"    LOC = "F9"  ;
16
17
```

Page 1

A.2 Ampelschaltung



schem.vhd **Wed Jun 18 13:08:57 2008**

```
1   ----------------------------------------------------------------------------
2   -- Copyright (c) 1995-2007 Xilinx, Inc.  All rights reserved.
3   ----------------------------------------------------------------------------
4   --   ____  ____
5   --  /   /\/   /
6   -- /___/  \  /    Vendor: Xilinx
7   -- \   \   \/     Version : 9.2i
8   --  \   \         Application : sch2vhdl
9   --  /   /         Filename : schem.vhf
10  -- /___/   /\     Timestamp : 03/27/2008 16:13:50
11  -- \   \  /  \
12  --  \___\/\___\
13  --
14  --Command: D:\Programme\Xilinx92i\bin\nt\sch2vhdl.exe -intstyle ise -family spartan3e -f
    -suppress -w D:/Programme/Xilinx92i/test2/schem.sch schem.vhf
15  --Design Name: schem
16  --Device: spartan3e
17  --Purpose:
18  --    This vhdl netlist is translated from an ECS schematic. It can be
19  --    synthesis and simulted, but it should not be modified.
20  --
21
22  library ieee;
23  use ieee.std_logic_1164.ALL;
24  use ieee.numeric_std.ALL;
25  library UNISIM;
26  use UNISIM.Vcomponents.ALL;
27
28  entity schem is
29     port ( clk_in   : in    std_logic;
30            reset_in : in    std_logic;
31            AMB      : inout std_logic;
32            GRN      : inout std_logic;
33            RD       : inout std_logic);
34  end schem;
35
36  architecture BEHAVIORAL of schem is
37     signal XLXN_61  : std_logic_vector (3 downto 0);
38     signal XLXN_674 : std_logic;
39     component Counter
40        port ( clock : in    std_logic;
41               reset : in    std_logic;
42               count : inout std_logic_vector (3 downto 0));
43     end component;
44
45     component divider
46        port ( ext_clk_50MHz : in    std_logic;
47               dev_clock_out : out   std_logic);
48     end component;
49
50     component STAT_MAC
51        port ( CLK   : in    std_logic;
52               RESET : in    std_logic;
53               TIMER : in    std_logic_vector (3 downto 0);
54               AMB   : out   std_logic;
55               GRN   : out   std_logic;
56               RD    : out   std_logic);
57     end component;
58
59  begin
60     XLXI_1 : Counter
```

schem.vhd **Wed Jun 18 13:08:57 2008**

```
61              port map (clock=>XLXN_674,
62                        reset=>reset_in,
63                        count(3 downto 0)=>XLXN_61(3 downto 0));
64
65          XLXI_2 : divider
66              port map (ext_clk_50MHz=>clk_in,
67                        dev_clock_out=>XLXN_674);
68
69          XLXI_3 : STAT_MAC
70              port map (CLK=>XLXN_674,
71                        RESET=>reset_in,
72                        TIMER(3 downto 0)=>XLXN_61(3 downto 0),
73                        AMB=>AMB,
74                        GRN=>GRN,
75                        RD=>RD);
76
77      end BEHAVIORAL;
78
79
80
```

Page 2

top.vhd Wed Jun 18 13:09:16 2008

```
1   ----------------------------------------------------------------------------
2   -- Projekt: Ampelschaltung
3   -- Instanz: TOP
4   ----------------------------------------------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity top is
12      Port ( ext_clk_50MHz : in  STD_LOGIC;
13             reset         : in  STD_LOGIC;
14             red_light     : out  STD_LOGIC;
15             amber_light   : out  STD_LOGIC;
16             green_light   : out STD_LOGIC;
17             dev_clock_out : inout STD_LOGIC
18             );
19  END entity top;
20
21  architecture Behavioral of top is
22
23  signal timer : std_logic_vector(3 downto 0);
24
25
26  COMPONENT counter
27     PORT(
28        CLOCK : IN std_logic;
29        RESET : IN std_logic;
30        COUNT : INOUT std_logic_vector(3 downto 0)
31         );
32     END COMPONENT;
33
34  COMPONENT stat_mac
35     PORT (
36           TIMER : IN std_logic_vector(3 downto 0);
37           CLK   : IN std_logic;
38           RESET : IN std_logic;
39           AMB   : OUT std_logic;
40           GRN   : OUT std_logic;
41           RD    : OUT std_logic
42          );
43     END COMPONENT;
44
45  COMPONENT divider
46      PORT (
47            ext_clk_50MHz : in  STD_LOGIC;
48            dev_clock_out : out  STD_LOGIC
49           );
50  end COMPONENT;
51
52  begin
53     Inst_counter: counter PORT MAP
54     (
55        CLOCK => dev_clock_out,
56        RESET => reset,
57        COUNT => timer
58     );
59
60     Inst_stat_mac: stat_mac PORT MAP
61     (
```

Page 1

top.vhd **Wed Jun 18 13:09:16 2008**

```
62              TIMER => timer,
63              CLK   => dev_clock_out,
64              RESET => reset,
65              AMB   => amber_light,
66              GRN   => green_light,
67              RD    => red_light
68          );
69
70          Inst_divider: divider PORT MAP(
71                  ext_clk_50MHz => ext_clk_50MHz,
72                  dev_clock_out => dev_clock_out
73          );
74
75      end Behavioral;
```

divider.vhd **Wed Jun 18 13:09:28 2008**

```
1   ----------------------------------------------------
2   --Projekt: Ampelschaltung
3   --Instanz: Divider
4   ----------------------------------------------------
5
6
7   library IEEE;
8   use IEEE.STD_LOGIC_1164.ALL;
9   use IEEE.STD_LOGIC_ARITH.ALL;
10  use IEEE.STD_LOGIC_UNSIGNED.ALL;
11
12
13
14  entity divider is
15      Port (
16              ext_clk_50MHz : in  STD_LOGIC;
17              dev_clock_out : inout  STD_LOGIC   -- 2Hz
18            );
19  end divider;
20
21
22  architecture Behavioral of divider is
23  signal divide_counter : integer := 0;
24
25  begin
26
27  Process(ext_clk_50MHz)
28        begin
29        if (rising_edge (ext_clk_50MHz)) then
30           if (divide_counter /= 12500000) then
31               divide_counter <= divide_counter + 1;
32           else
33              divide_counter <= 0;
34              dev_clock_out <= not (dev_clock_out);
35           end if;
36        end if;
37   end process;
38  end Behavioral;
```

Counter.vhd **Wed Jun 18 13:09:34 2008**

```
1   ----------------------------------------------------
2   --Projekt: Ampelschaltung
3   --Instanz: Counter
4   ----------------------------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10  ----Berechnet den zustand der Ampel
11
12  entity Counter is
13      Port ( clock : in  STD_LOGIC;
14             reset : in  STD_LOGIC;
15             count : inout  STD_LOGIC_VECTOR (3 downto 0));
16  end Counter;
17
18  architecture Behavioral of Counter is
19
20  begin
21  process (clock, reset)
22     begin
23        if reset='1'  then
24           count <= "0000";
25        elsif clock='1' and clock'event then
26           count <= count + 1;
27        end if;
28  end process;
29  end Behavioral;
```

Page 1

schem.ucf **Wed Jun 18 13:09:41 2008**

```
1   #----------------------------------
2   #Projekt: Ampelschaltung
3   #   User Constraint File
4   #----------------------------------
5   NET "amber_light"    LOC = "E12"  ;
6   NET "ext_clk_50MHz" LOC = "C9"  ;
7   NET "green_light"    LOC = "F12"  ;
8   NET "red_light"      LOC = "E11"  ;
9   NET "reset"          LOC = "K17"  ;
10  NET "dev_clock_out" LOC = "F9";
11
12
```

A.3 Binär zählen

top.vhd Wed Jun 18 13:20:05 2008

```
  1  ----------------------------------
  2  --Projekt: Binär Zählen
  3  --Instanz: Top
  4  ----------------------------------
  5  library IEEE;
  6  use IEEE.STD_LOGIC_1164.ALL;
  7  use IEEE.STD_LOGIC_ARITH.ALL;
  8  use IEEE.STD_LOGIC_UNSIGNED.ALL;
  9
 10
 11
 12  entity top is
 13      Port ( SOUTH        : in  STD_LOGIC;
 14             WEST         : in  STD_LOGIC;
 15             EAST         : in  STD_LOGIC;
 16             S_0          : in  STD_LOGIC;      -- Schalter
 17             S_1          : in  STD_LOGIC;      -- Schalter
 18             S_2          : in  STD_LOGIC;      -- Schalter
 19             LED          : inout STD_LOGIC_VECTOR (2 downto 0);
 20             clk_system   : in  STD_LOGIC;
 21             intern_clk   : inout STD_logic
 22             );
 23  end top;
 24
 25  architecture Behavioral of top is
 26
 27  COMPONENT divider
 28      PORT (
 29             ext_clk_50MHz : in  STD_LOGIC;
 30             dev_clock_out : inout  STD_LOGIC
 31           );
 32  end COMPONENT;
 33
 34  COMPONENT mac
 35      PORT (
 36             SOUTH         : in  STD_LOGIC;
 37             WEST          : in  STD_LOGIC;
 38             EAST          : in  STD_LOGIC;
 39             S_0           : in  STD_LOGIC;      -- Schalter
 40             S_1           : in  STD_LOGIC;      -- Schalter
 41             S_2           : in  STD_LOGIC;      -- Schalter
 42             LED           : inout STD_LOGIC_VECTOR (2 downto 0);
 43             dev_clock_out : in STD_LOGIC
 44           );
 45  end COMPONENT;
 46
 47  begin
 48
 49  Inst_divider: divider PORT MAP
 50     (
 51             ext_clk_50MHz => clk_system,
 52             dev_clock_out => intern_clk
 53     );
 54
 55  Inst_mac: mac PORT MAP
 56     (
 57           dev_clock_out  => intern_clk,
 58           SOUTH          => SOUTH,
 59           WEST           => West,
 60           EAST           => EAST,
 61           S_0            => S_0,
```

Page 1

top.vhd Wed Jun 18 13:20:05 2008

```
62                  S_1             => S_1,
63                  S_2             => S_2,
64                  LED             => LED
65          );
66      end Behavioral;
67
68
```

Page 2

MAC.vhd **Wed Jun 18 13:20:15 2008**

```
1   ----------------------------------
2   --Projekt: Binär Zählen
3   --Instanz: Mac
4   ----------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11  entity MAC is
12      Port ( SOUTH         : in  STD_LOGIC;
13             EAST          : in  STD_LOGIC;
14             WEST          : in  STD_LOGIC;
15             S_0           : in STD_LOGIC;
16             S_1           : in STD_LOGIC;
17             S_2           : in STD_LOGIC;
18             LED           : inout  STD_LOGIC_VECTOR (2 downto 0);
19             dev_clock_out : in STD_LOGIC
20           );
21  end MAC;
22
23  architecture Behavioral of MAC is
24
25  begin
26
27
28  process (WEST,SOUTH,EAST,S_0,S_1,S_2,dev_clock_out)
29  begin
30     if East='1'   then             -- reset
31           LED <= "000";
32     elsif rising_edge(dev_clock_out) then
33        if SOUTH='1' then
34           LED <= LED + 1;
35        elsif WEST='1' then
36           LED <= LED - 1;
37        elsif S_0='1'
38           then LED(0) <= not LED (0);
39        elsif S_1='1'
40           then LED(1) <= not LED (1);
41        elsif S_2='1'
42           then LED(2) <= not LED (2);
43        end if;
44     end if;
45  end process;
46
47  end Behavioral;
```

Page 1

DIVIDER.vhd **Wed Jun 18 13:20:24 2008**

```
1   ----------------------------------
2   --Projekt: Binär Zählen
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14              ext_clk_50MHz : in      STD_LOGIC;
15              dev_clock_out : inout   STD_LOGIC
16              );
17  end divider;
18
19
20  architecture Behavioral of divider is
21  signal divide_counter : integer := 0;
22
23
24  begin
25
26  Process(ext_clk_50MHz)
27          begin
28            if (rising_edge (ext_clk_50MHz)) then
29              if (divide_counter /= 2600) then
30                   divide_counter <= divide_counter + 1;
31              else
32                  divide_counter <= 0;
33                  dev_clock_out <= not ( dev_clock_out);
34              end if;
35          end if;
36  end process;
37  end Behavioral;
```

top.ucf **Wed Jun 18 13:20:32 2008**

```
1   #---------------------------------
2   #Projekt: Binär Zählen
3   #  User Constraint File
4   # ---------------------------------
5   NET "clk_system"  LOC = "C9"  ;
6   NET "EAST"        LOC = "H13"  ;
7   NET "LED<0>"      LOC = "F12"  ;
8   NET "LED<1>"      LOC = "E12"  ;
9   NET "LED<2>"      LOC = "E11"  ;
10  NET "S_0"         LOC = "L13"  ;
11  NET "S_1"         LOC = "L14"  ;
12  NET "S_2"         LOC = "H18"  ;
13  NET "SOUTH"       LOC = "K17"  ;
14  NET "WEST"        LOC = "D18"  ;
15
16
```

Page 1

A.4 Drehinkrementgeber

top.vhd Wed Jun 18 13:32:04 2008

```
1   ----------------------------------
2   --Projekt: Drehinkrementgeber
3   --Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity top is
11      Port ( rotation_b_in :    in  STD_LOGIC;
12             rotation_a_in :    in  STD_LOGIC;
13             reset         :    in  STD_LOGIC;
14             clk_system    :    in  STD_LOGIC;
15             test          :    out STD_LOGIC;
16             direction     :    inout STD_LOGIC_VECTOR (1 downto 0);
17             position      :    inout STD_LOGIC_VECTOR (4 downto 0)
18            );
19  end top;
20
21  architecture Behavioral of top is
22
23  signal rotation_a : STD_LOGIC;
24  signal rotation_b : STD_LOGIC;
25  signal intern_clk : STD_LOGIC;
26
27  COMPONENT mac PORT (    rotation_b :   in  STD_LOGIC;
28                          rotation_a :   in  STD_LOGIC;
29                          reset      :   in  STD_LOGIC;
30                          test       :   out STD_LOGIC;
31                          direction  :   out STD_LOGIC_VECTOR (1 downto 0);
32                          position   :   inout STD_LOGIC_VECTOR (4 downto 0)
33
34                     );
35  end COMPONENT;
36
37  COMPONENT rotary_contact_filter PORT
38                         (
39                          rotation_a_in  :  in    STD_LOGIC;
40                          rotation_b_in  :  in    STD_LOGIC;
41                          dev_clk_in     :  in    STD_LOGIC;
42                          rotation_a     :  inout STD_LOGIC;
43                          rotation_b     :  inout STD_LOGIC
44                         );
45  end COMPONENT;
46
47  COMPONENT DIVIDER PORT
48                         (
49                          ext_clk_50MHz : in      STD_LOGIC;
50                          test          : OUT     STD_LOGIC;
51                          dev_clock_out : inout   STD_LOGIC
52                         );
53  end COMPONENT;
54
55  begin
56  Inst_DIVIDER: DIVIDER PORT MAP
57     (
58        ext_clk_50MHz  => clk_system,
59        dev_clock_out  => intern_clk,
60        test           => test
61      );
```

Page 1

top.vhd Wed Jun 18 13:32:04 2008

```
62
63          Inst_rotary_contact_filter: rotary_contact_filter PORT MAP
64              (
65                 rotation_a_in  => rotation_a_in,
66                 rotation_b_in  => rotation_b_in,
67                 dev_clk_in     => intern_clk,
68                 rotation_a     => rotation_a,
69                 rotation_b     => rotation_b
70              );
71
72
73
74          Inst_mac: mac PORT MAP
75              (
76                 position    => position,
77                 rotation_a  => rotation_a,
78                 rotation_b  => rotation_b,
79                 direction   => direction,
80                 reset       => reset
81              );
82          end Behavioral;
83
84
```

mac.vhd **Wed Jun 18 13:32:11 2008**

```
1   ----------------------------------
2   --Projekt: Drehinkrementgeber
3   --Instanz: Mac
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity mac is
11      Port ( rotation_b      :  in  STD_LOGIC;
12             rotation_a      :  in  STD_LOGIC;
13             reset           :  in  STD_LOGIC;
14             test            :  out STD_LOGIC;
15             direction       :  out STD_LOGIC_VECTOR (1 downto 0);
16             position        :  inout STD_LOGIC_VECTOR (4 downto 0)
17            );
18  end mac;
19
20  architecture Behavioral of mac is
21  begin
22  process (reset,rotation_a,rotation_b)
23  begin
24  if reset='1' then
25     position <= "00000";
26  elsif rising_edge(rotation_a) then
27        if rotation_b = '1' then
28           direction <= "10";
29           position <= position -1;
30        else direction <= "01";
31           position <= position +1;
32        end if;
33
34  end if;
35  end process;
36  end Behavioral;
```

Rotary_contact_filter.vhd **Wed Jun 18 13:32:19 2008**

```
1   ----------------------------------
2   --Projekt: Drehinkrementgeber
3   --Instanz: rotary_contact_filter
4   ----------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11
12  entity rotary_contact_filter is
13      Port (  rotation_a_in  :  in    STD_LOGIC;
14              rotation_b_in  :  in    STD_LOGIC;
15              dev_clk_in     :  in    STD_LOGIC;
16              rotation_a     :  inout STD_LOGIC;
17              rotation_b     :  inout STD_LOGIC
18            );
19  end rotary_contact_filter;
20
21  architecture Behavioral of rotary_contact_filter is
22  signal rotation_in : STD_LOGIC_VECTOR (1 downto 0);
23  begin
24  process(dev_clk_in)
25  begin
26     if (rising_edge (dev_clk_in)) then
27        rotation_in <= rotation_b_in & rotation_a_in;
28        case rotation_in is
29              when "00" => rotation_a <= '0';
30                    rotation_b <= rotation_b;
31              when "01" => rotation_a <= rotation_a;
32                    rotation_b <= '0';
33              when "10" => rotation_a <= rotation_a;
34                    rotation_b <= '1';
35              when "11" => rotation_a <= '1';
36                    rotation_b <= rotation_b;
37              when others => rotation_a <= rotation_a;
38                    rotation_b <= rotation_b;
39        end case;
40     end if;
41  end process;
42
43  end Behavioral;
```

DIVIDER.vhd **Wed Jun 18 13:32:25 2008**

```
1   ----------------------------------
2   --Projekt: Drehinkrementgeber
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14              ext_clk_50MHz : in      STD_LOGIC;
15              dev_clock_out : inout   STD_LOGIC
16              );
17  end divider;
18
19
20  architecture Behavioral of divider is
21  signal divide_counter : integer := 0;
22
23
24  begin
25
26  Process(ext_clk_50MHz)
27          begin
28            if (rising_edge (ext_clk_50MHz)) then
29              if (divide_counter /= 2500) then
30                   divide_counter <= divide_counter + 1;
31              else
32                  divide_counter <= 0;
33                  dev_clock_out <= not ( dev_clock_out);
34              end if;
35          end if;
36  end process;
37  end Behavioral;
```

top.ucf 1 / 1

```
#-------------------------------------
# Projekt: Drehinkrementgeber
#          User Constraint File
#-------------------------------------------
NET "clk_system"    LOC = "C9"  ;
NET "direction<0>"  LOC = "F12" ;
NET "direction<1>"  LOC = "E12"  ;
NET "position<0>"   LOC = "F11"  ;
NET "position<1>"   LOC = "C11"  ;
NET "position<2>"   LOC = "D11"  ;
NET "position<3>"   LOC = "E9"  ;
NET "position<4>"   LOC = "F9"  ;
NET "reset"         LOC = "V16"  |PULLDOWN ;
NET "rotation_a_in" LOC = "K18"  |PULLUP ;
NET "rotation_b_in" LOC = "G18"  |PULLUP;
NET "test"          LOC = "E11"  ;
```

June 18, 2008 Crimson Editor

A.5 Konstante senden

TOP.vhd Wed Jun 25 14:12:03 2008

```
1   ----------------------------------
2   -- Projekt: Konstante senden
3   -- Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity TOP is
13        PORT    (
14                  clk_system     : in STD_LOGIC;
15                  button_send    : in STD_LOGIC;
16                  intern_clk     : inout STD_LOGIC;
17                  RS232_DCE_RXD  : out STD_LOGIC
18                  );
19  end TOP;
20
21
22
23  architecture Behavioral of TOP is
24  COMPONENT divider
25     PORT  (
26               ext_clk_50MHz     : in     STD_LOGIC;
27               dev_clock_out     : inout  STD_LOGIC
28             );
29  end COMPONENT;
30
31  COMPONENT MAC
32     PORT (
33               dat_send       : out STD_LOGIC;
34               mac_clk_in     : in  STD_LOGIC;
35               start_button   : in  STD_LOGIC
36               );
37  end COMPONENT;
38  begin
39
40  Inst_divider: divider PORT MAP
41            (
42               ext_clk_50MHz  => clk_system,
43               dev_clock_out  => intern_clk
44            );
45
46  Inst_mac: mac PORT MAP
47            (
48               dat_send       => RS232_DCE_RXD,
49               mac_clk_in     => intern_clk,
50               start_button   => button_send
51            );
52  end Behavioral;
53
54
```

Page 1

MAC.vhd **Wed Jun 25 14:12:18 2008**

```
1   ----------------------------------
2   -- Projekt: Konstante senden
3   -- Instanz: Mac
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity MAC is
12        PORT(
13              dat_send       : out STD_LOGIC;
14              start_button   : in  STD_LOGIC;
15              mac_clk_in     : in  STD_LOGIC
16             );
17  end MAC;
18
19  architecture Behavioral of MAC is
20
21  signal dat_send_complete   : STD_LOGIC_VECTOR (9 downto 0) := "1111110110";
22  signal counter_index       : integer  ;
23  signal help: integer := 0;
24
25  begin
26  process (mac_clk_in, start_button)
27
28  begin
29
30  if rising_edge (mac_clk_in) THEN
31     if start_button = '1' THEN
32        if help = 0 then
33           if counter_index < 10 THEN
34              counter_index <= counter_index + 1;
35              dat_send <= dat_send_complete(counter_index);
36              help <= 0;
37           else counter_index <= 0;
38                help <= 1;
39           end if;
40        end if;
41     elsif start_button = '0' THEN
42      if help = 0 Then
43        if counter_index > 0 THEN
44              if Counter_index < 10 THEN
45                 counter_index <= counter_index + 1;
46                 dat_send <= dat_send_complete(counter_index);
47              else Counter_index <= 0;
48                   help <= 1;
49              end if;
50           end if;
51        end if;
52     end if;
53  end if;
54  end process;
55  end Behavioral;
```

Page 1

DIVIDER.vhd **Wed Jun 25 14:12:54 2008**

```
1   ----------------------------------
2   --Projekt: Konstante Senden
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14              ext_clk_50MHz : in      STD_LOGIC;
15              dev_clock_out : inout   STD_LOGIC
16              );
17  end divider;
18
19
20  architecture Behavioral of divider is
21  signal divide_counter : integer := 0;
22
23
24  begin
25
26  Process(ext_clk_50MHz)
27          begin
28            if (rising_edge (ext_clk_50MHz)) then
29              if (divide_counter /= 2500) then
30                   divide_counter <= divide_counter + 1;
31              else
32                  divide_counter <= 0;
33                  dev_clock_out <= not ( dev_clock_out);
34              end if;
35          end if;
36  end process;
37  end Behavioral;
```

TOP.ucf **Wed Jun 25 14:13:06 2008**

```
1   #----------------------------------
2   # Projekt: Konstante senden
3   #    User Constraint File
4   #----------------------------------
5   NET "button_send"       LOC = "K17";
6   NET "clk_system"        LOC = "C9";
7   NET "intern_clk"        LOC = "E12";
8   NET "RS232_DCE_RXD"     LOC = "M14";
9   INST "Inst_mac/dat_send"              INIT = 1;
```

A.6 Variable senden

TOP.vhd Wed Jun 18 13:55:31 2008

```
1   ----------------------------------------
2   --Projekt: Variable senden
3   --Instanz: Top
4   ----------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity TOP is
12           PORT (
13                    clk_system     :  in    STD_LOGIC;
14                    rot_a_in       :  in    STD_LOGIC;
15                    rot_b_in       :  in    STD_LOGIC;
16                    reset          :  in    STD_LOGIC;
17                    button_send    :  in    STD_LOGIC;
18                    led_clk_control:  out   STD_LOGIC;
19                    set_led        :  inout STD_LOGIC_VECTOR (6 downto 0);
20                    out_led        :  out   STD_LOGIC_VECTOR (6 downto 0);
21                    RS232_DCE_RXD  :  out   STD_LOGIC
22                  );
23  end TOP;
24
25  architecture Behavioral of TOP is
26
27
28  signal rot_a            : STD_LOGIC;
29  signal rot_b            : STD_LOGIC;
30  signal dat_send_binaer  : STD_LOGIC_VECTOR (6 downto 0);
31  signal intern_clk       : STD_LOGIC;
32
33  COMPONENT divider
34      PORT    (
35                ext_clk_50MHz           : in    STD_LOGIC;
36                test                    : out   STD_LOGIC;
37                dev_clock_out           : inout STD_LOGIC
38              );
39  end COMPONENT;
40
41  COMPONENT MAC
42      PORT(
43              dat_send          : out STD_LOGIC;
44              start_button      : in  STD_LOGIC;
45              dat_send_binaer   : in STD_LOGIC_VECTOR (6 downto 0);
46              mac_clk_in        : in  STD_LOGIC
47             );
48  end COMPONENT;
49
50  COMPONENT rotary_contact_filter
51   Port       (  rotation_a_in   :   in      STD_LOGIC;
52                 rotation_b_in   :   in      STD_LOGIC;
53                 dev_clk_in      :   in      STD_LOGIC;
54                 rotation_a      :   inout   STD_LOGIC;
55                 rotation_b      :   inout   STD_LOGIC
56              );
57  end COMPONENT;
58
59  COMPONENT set
60        PORT (   rotation_a  :  in STD_LOGIC;
61                 rotation_b  :  in STD_LOGIC;
```

Page 1

TOP.vhd **Wed Jun 18 13:55:31 2008**

```
 62                      clk          :  in STD_LOGIC;
 63                      reset        :  in STD_LOGIC;
 64                      out_led      :  out STD_LOGIC_VECTOR (6 downto 0);
 65                      set_led      :  inout STD_LOGIC_VECTOR (6 downto 0)
 66                  );
 67      end COMPONENT;
 68      begin
 69
 70      Inst_divider: divider PORT MAP
 71                  (
 72                      ext_clk_50MHz       => clk_system,
 73                      test                => led_clk_control,
 74                      dev_clock_out       => intern_clk
 75                  );
 76
 77      Inst_mac: mac PORT MAP
 78                  (
 79                      dat_send        => RS232_DCE_RXD,
 80                      mac_clk_in      => intern_clk,
 81                      start_button    => button_send,
 82                      dat_send_binaer => dat_send_binaer
 83                  );
 84
 85      Inst_rotary_contact_filter: rotary_contact_filter PORT MAP
 86                  (
 87                   rotation_a_in   => rot_a_in,
 88                   rotation_b_in   => rot_b_in,
 89                   dev_clk_in      => intern_clk,
 90                   rotation_a      => rot_a,
 91                   rotation_b      => rot_b
 92                  );
 93
 94      Inst_set: set PORT MAP
 95                  (
 96                     rotation_a      => rot_a,
 97                     rotation_b      => rot_b,
 98                     clk             => intern_clk,
 99                     out_led         => out_led,
100                     reset           => reset,
101                     set_led         => dat_send_binaer
102                  );
103
104     end Behavioral;
```

Page 2

MAC.vhd **Wed Jun 18 13:55:40 2008**

```
1   ----------------------------------
2   --Projekt: Variable senden
3   --Instanz: Mac
4   ----------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11
12  entity MAC is
13        PORT(
14              dat_send          : out STD_LOGIC;
15              start_button      : in  STD_LOGIC;
16              dat_send_binaer   : in STD_LOGIC_VECTOR (6 downto 0);
17              mac_clk_in        : in  STD_LOGIC
18             );
19  end MAC;
20
21  architecture Behavioral of MAC is
22
23  signal counter_index       : integer  ;
24  signal help                : integer := 0;
25  signal dat_send_complete   : STD_LOGIC_VECTOR (9 downto 0);
26
27  begin
28  process (mac_clk_in)
29  begin
30  if rising_edge (mac_clk_in) then
31
32     dat_send_complete(9) <= '1';
33     dat_send_complete(8) <= '1';
34     dat_send_complete(7) <= dat_send_binaer(6);
35     dat_send_complete(6) <= dat_send_binaer(5);
36     dat_send_complete(5) <= dat_send_binaer(4);
37     dat_send_complete(4) <= dat_send_binaer(3);
38     dat_send_complete(3) <= dat_send_binaer(2);
39     dat_send_complete(2) <= dat_send_binaer(1);
40     dat_send_complete(1) <= dat_send_binaer(0);
41     dat_send_complete(0) <= '0';
42  end if;
43  end process;
44
45
46  process (mac_clk_in)
47
48  begin
49
50  if rising_edge (mac_clk_in) THEN
51     if start_button = '1' THEN
52        if help = 0 then
53           if counter_index < 10 THEN
54              counter_index <= counter_index + 1;
55              dat_send <= dat_send_complete(counter_index);
56              help <= 0;
57           else counter_index <= 0;
58                help <= 1;
59           end if;
60        end if;
61     elsif start_button = '0' THEN
```

MAC.vhd Wed Jun 18 13:55:40 2008

```
62            if help = 0 Then
63              if counter_index > 0 THEN
64                    if Counter_index < 10 THEN
65                       counter_index <= counter_index + 1;
66                       dat_send <= dat_send_complete(counter_index);
67                    else Counter_index <= 0;
68                       help <= 1;
69                    end if;
70                end if;
71              end if;
72           end if;
73        end if;
74        end process;
75        end Behavioral;
```

FILTER.vhd **Wed Jun 18 13:55:48 2008**

```
1   ----------------------------------
2   --Projekt: Variable senden
3   --Instanz: Rotary_Contact_Filter
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity rotary_contact_filter is
12      Port (  rotation_a_in    :   in      STD_LOGIC;
13              rotation_b_in    :   in  STD_LOGIC;
14              dev_clk_in       :   in      STD_LOGIC;
15              rotation_a       :   inout   STD_LOGIC;
16              rotation_b       :   inout STD_LOGIC
17            );
18  end rotary_contact_filter;
19
20  architecture Behavioral of rotary_contact_filter is
21  signal rotation_in : STD_LOGIC_VECTOR (1 downto 0);
22  begin
23  process(dev_clk_in)
24  begin
25      if (rising_edge (dev_clk_in)) then
26          rotation_in <= rotation_b_in & rotation_a_in;
27          case rotation_in is
28                  when "00" => rotation_a <= '0';
29                               rotation_b <= rotation_b;
30                  when "01" => rotation_a <= rotation_a;
31                               rotation_b <= '0';
32                  when "10" => rotation_a <= rotation_a;
33                               rotation_b <= '1';
34                  when "11" => rotation_a <= '1';
35                               rotation_b <= rotation_b;
36                  when others => rotation_a <= rotation_a;
37                               rotation_b <= rotation_b;
38          end case;
39      end if;
40  end process;
41
42  end Behavioral;
43
```

SET.vhd **Wed Jun 18 13:55:57 2008**

```
1   ----------------------------------
2   --Projekt: Variable senden
3   --Instanz: SET
4   ----------------------------------
5
6
7   library IEEE;
8   use IEEE.STD_LOGIC_1164.ALL;
9   use IEEE.STD_LOGIC_ARITH.ALL;
10  use IEEE.STD_LOGIC_UNSIGNED.ALL;
11
12  ----------------------------
13  ---DREHGEBER-->LEDS_SETZEN---
14  ----------------------------
15
16
17  entity SET is
18        PORT (   rotation_a  :  in STD_LOGIC;
19                 rotation_b  :  in STD_LOGIC;
20                 reset       :  in STD_LOGIC;
21                 out_led     :  out STD_LOGIC_VECTOR (6 downto 0);
22                 clk         :  in STD_LOGIC;
23                 set_led     :  inout STD_LOGIC_VECTOR (6 downto 0)
24               );
25
26  end SET;
27
28  architecture Behavioral of SET is
29  begin
30
31  process (reset, rotation_a, rotation_b)
32  begin
33  if reset='1' then
34     set_led <= "0000000";
35  elsif rising_edge (rotation_a) then
36     if rotation_b ='1' then
37        set_led <= set_led -1;
38        else set_led <= set_led +1;
39     end if;
40  end if;
41  end process;
42
43  process (clk)
44  begin
45  if rising_edge (clk) then
46     out_led(6) <= set_led(6);
47     out_led(5) <= set_led(5);
48     out_led(4) <= set_led(4);
49     out_led(3) <= set_led(3);
50     out_led(2) <= set_led(2);
51     out_led(1) <= set_led(1);
52     out_led(0) <= set_led(0);
53  end if;
54
55  end process;
56  end Behavioral;
57
58
```

Page 1

divider.vhd **Wed Jun 18 13:56:05 2008**

```
1   ----------------------------------
2   --Projekt: Variable senden
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14                  ext_clk_50MHz       : in          STD_LOGIC;
15                  test                : out STD_LOGIC;
16                  dev_clock_out       : inout   STD_LOGIC
17                );
18  end divider;
19
20
21  architecture Behavioral of divider is
22  signal divide_counter : integer := 0;
23
24
25  begin
26
27  Process(ext_clk_50MHz)
28          begin
29            if (rising_edge (ext_clk_50MHz)) then
30              if (divide_counter /= 2600) then
31                   divide_counter <= divide_counter + 1;
32              else
33                  divide_counter <= 0;
34                  dev_clock_out <= not (dev_clock_out);
35                  test <= dev_clock_out ;
36              end if;
37          end if;
38  end process;
39  end Behavioral;
40
41
```

TOP.ucf **Wed Jun 18 13:56:15 2008**

```
1   #-------------------------------------
2   #Projekt: Variable senden
3   #  User Constraint File
4   #-------------------------------------
5   NET "button_send"       LOC = "K17" ;
6   NET "clk_system"        LOC = "C9"  ;
7   NET "led_clk_control"   LOC = "F12" ;
8   NET "out_led<0>"        LOC = "E12" ;
9   NET "out_led<1>"        LOC = "E11" ;
10  NET "out_led<2>"        LOC = "F11" ;
11  NET "out_led<3>"        LOC = "C11" ;
12  NET "out_led<4>"        LOC = "D11" ;
13  NET "out_led<5>"        LOC = "E9"  ;
14  NET "out_led<6>"        LOC = "F9"  ;
15  NET "rot_a_in"          LOC = "K18" | PULLUP ;
16  NET "rot_b_in"          LOC = "G18" | PULLUP ;
17  NET "RS232_DCE_RXD"     LOC = "M14"  ;
18  NET "reset"             LOC = "V16" | PULLDOWN ;
19  INST "Inst_mac/dat_send"                      INIT = 1;
20
21
```

Page 1

A.7 Senden mehrerer Zeichen

TOP.vhd Wed Jun 18 14:33:08 2008

```
1   ----------------------------------
2   --Projekt: Senden mehrerer Zeichen
3   --Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity TOP is
12          PORT (
13                  clk_system     :  in    STD_LOGIC;
14                  rot_a_in       :  in    STD_LOGIC;
15                  rot_b_in       :  in    STD_LOGIC;
16                  reset          :  in    STD_LOGIC;
17                  button_send    :  in    STD_LOGIC;
18                  led_clk_control:  out   STD_LOGIC;
19                  set_led        :  inout STD_LOGIC_VECTOR (6 downto 0);
20                  out_led        :  out   STD_LOGIC_VECTOR (6 downto 0);
21                  RS232_DCE_RXD  :  out   STD_LOGIC
22                );
23  end TOP;
24
25  architecture Behavioral of TOP is
26
27
28  signal rot_a            : STD_LOGIC;
29  signal rot_b            : STD_LOGIC;
30  signal dat_send_binaer  : STD_LOGIC_VECTOR (6 downto 0);
31  signal intern_clk       : STD_LOGIC;
32  signal help             : STD_LOGIC;
33
34  COMPONENT divider
35      PORT    (
36              ext_clk_50MHz           : in    STD_LOGIC;
37              test                    : out   STD_LOGIC;
38              dev_clock_out           : inout STD_LOGIC
39            );
40  end COMPONENT;
41
42  COMPONENT MAC
43      PORT(
44            dat_send          : out STD_LOGIC;
45            start_button      : in STD_LOGIC;
46            dat_send_binaer   : in STD_LOGIC_VECTOR (6 downto 0);
47            set_help          : inout STD_LOGIC;
48            mac_clk_in        : in  STD_LOGIC
49           );
50  end COMPONENT;
51
52  COMPONENT rotary_contact_filter
53   Port       (  rotation_a_in   :   in      STD_LOGIC;
54              rotation_b_in   :   in      STD_LOGIC;
55              dev_clk_in      :   in      STD_LOGIC;
56              rotation_a      :   inout   STD_LOGIC;
57              rotation_b      :   inout   STD_LOGIC
58            );
59  end COMPONENT;
60
61  COMPONENT set
```

Page 1

TOP.vhd **Wed Jun 18 14:33:08 2008**

```
62              PORT (   rotation_a  :  in STD_LOGIC;
63                       rotation_b  :  in STD_LOGIC;
64                       clk         :  in STD_LOGIC;
65                       reset       :  in STD_LOGIC;
66                       out_led     :  out STD_LOGIC_VECTOR (6 downto 0);
67                       set_led     :  inout STD_LOGIC_VECTOR (6 downto 0)
68                     );
69        end COMPONENT;
70
71        COMPONENT set_help
72            PORT   ( help         : inout STD_LOGIC;
73                     start_button : in STD_LOGIC
74                   );
75        end COMPONENT;
76        begin
77        Inst_set_help: set_help PORT MAP
78                     ( start_button    => button_send,
79                       help            => help
80
81                     );
82
83        Inst_divider: divider PORT MAP
84                     (
85                        ext_clk_50MHz       => clk_system,
86                        test                => led_clk_control,
87                        dev_clock_out       => intern_clk
88                     );
89
90        Inst_mac: mac PORT MAP
91                     (
92                        dat_send        => RS232_DCE_RXD,
93                        mac_clk_in      => intern_clk,
94                        start_button    => button_send,
95                        set_help        => help,
96                        dat_send_binaer => dat_send_binaer
97                     );
98
99        Inst_rotary_contact_filter: rotary_contact_filter PORT MAP
100                    (
101                     rotation_a_in   => rot_a_in,
102                     rotation_b_in   => rot_b_in,
103                     dev_clk_in      => intern_clk,
104                     rotation_a      => rot_a,
105                     rotation_b      => rot_b
106                    );
107
108       Inst_set: set PORT MAP
109                    (
110                       rotation_a      => rot_a,
111                       rotation_b      => rot_b,
112                       clk             => intern_clk,
113                       out_led         => out_led,
114                       reset           => reset,
115                       set_led         => dat_send_binaer
116                    );
117
118       end Behavioral;
```

Page 2

MAC.vhd **Wed Jun 18 14:33:17 2008**

```
1   ----------------------------------
2   --Projekt: Senden mehrerer Zeichen
3   --Instanz: Mac
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity MAC is
12          PORT(
13                  dat_send            : out STD_LOGIC;
14                  start_button        : in  STD_LOGIC;
15                  dat_send_binaer     : in STD_LOGIC_VECTOR (6 downto 0);
16                  mac_clk_in          : in  STD_LOGIC;
17                  set_help            : inout STD_LOGIC
18                );
19  end MAC;
20
21  architecture Behavioral of MAC is
22
23  signal counter_index        : integer  ;
24  signal help                 : STD_LOGIC;
25  signal dat_send_complete    : STD_LOGIC_VECTOR (9 downto 0);
26
27  begin
28  process (mac_clk_in)
29  begin
30  if rising_edge (mac_clk_in) then
31
32      dat_send_complete(9) <= '1';
33      dat_send_complete(8) <= '1';
34      dat_send_complete(7) <= dat_send_binaer(6);
35      dat_send_complete(6) <= dat_send_binaer(5);
36      dat_send_complete(5) <= dat_send_binaer(4);
37      dat_send_complete(4) <= dat_send_binaer(3);
38      dat_send_complete(3) <= dat_send_binaer(2);
39      dat_send_complete(2) <= dat_send_binaer(1);
40      dat_send_complete(1) <= dat_send_binaer(0);
41      dat_send_complete(0) <= '0';
42  end if;
43  end process;
44
45
46  process (mac_clk_in, start_button)
47  begin
48
49  if rising_edge (mac_clk_in) THEN
50     if (start_button = '1') AND (counter_index < 10) THEN
51        help <= set_help;
52        if help = '0' then
53            if counter_index < 10 THEN
54                 counter_index <= counter_index + 1;
55                 dat_send <= dat_send_complete(counter_index);
56            else counter_index <= 0;
57                help <= '1';
58             end if;
59          end if;
60     elsif (start_button = '0') AND (counter_index = 10) THEN
61        counter_index <= 0;
```

MAC.vhd **Wed Jun 18 14:33:17 2008**

```
62              help <= '1';
63            elsif start_button = '0' THEN
64                 if help = '0' Then
65                     if counter_index > 0 THEN
66                         if Counter_index < 10 THEN
67                             counter_index <= counter_index + 1;
68                             dat_send <= dat_send_complete(counter_index);
69                         else Counter_index <= 0;
70                              help <= '1';
71                         end if;
72                     end if;
73                 end if;
74             end if;
75         end if;
76         end process;
77
78
79
80         end Behavioral;
81
```

Page 2

rotary_contact_filter.vhd **Wed Jun 18 14:33:25 2008**

```
1   ----------------------------------
2   --Projekt: Senden mehrerer Zeichen
3   --Instanz: Rotary-Contact-Filter
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity rotary_contact_filter is
12      Port (      rotation_a_in   :   in      STD_LOGIC;
13                  rotation_b_in   :   in      STD_LOGIC;
14                  dev_clk_in      :   in      STD_LOGIC;
15                  rotation_a      :   inout   STD_LOGIC;
16                  rotation_b      :   inout   STD_LOGIC
17              );
18  end rotary_contact_filter;
19
20  architecture Behavioral of rotary_contact_filter is
21  signal rotation_in : STD_LOGIC_VECTOR (1 downto 0);
22  begin
23  process(dev_clk_in)
24  begin
25      if (rising_edge (dev_clk_in)) then
26          rotation_in <= rotation_b_in & rotation_a_in;
27          case rotation_in is
28                  when "00" => rotation_a <= '0';
29                               rotation_b <= rotation_b;
30                  when "01" => rotation_a <= rotation_a;
31                               rotation_b <= '0';
32                  when "10" => rotation_a <= rotation_a;
33                               rotation_b <= '1';
34                  when "11" => rotation_a <= '1';
35                               rotation_b <= rotation_b;
36                  when others => rotation_a <= rotation_a;
37                               rotation_b <= rotation_b;
38          end case;
39      end if;
40  end process;
41
42  end Behavioral;
```

SET.vhd **Wed Jun 18 14:33:33 2008**

```
 1
 2        ----------------------------------
 3        --Projekt: Senden mehrerer Zeichen
 4        --Instanz: Set
 5        ----------------------------------
 6
 7        library IEEE;
 8        use IEEE.STD_LOGIC_1164.ALL;
 9        use IEEE.STD_LOGIC_ARITH.ALL;
10        use IEEE.STD_LOGIC_UNSIGNED.ALL;
11
12        ------------------------------
13        ---DREHGEBER-->LEDS_SETZEN---
14        ------------------------------
15
16        entity SET is
17              PORT (   rotation_a  :  in STD_LOGIC;
18                       rotation_b  :  in STD_LOGIC;
19                       reset       :  in STD_LOGIC;
20                       out_led     :  out STD_LOGIC_VECTOR (6 downto 0);
21                       clk         :  in STD_LOGIC;
22                       set_led     :  inout STD_LOGIC_VECTOR (6 downto 0)
23                    );
24        end SET;
25
26        architecture Behavioral of SET is
27        begin
28
29        process (reset, rotation_a, rotation_b)
30        begin
31        if reset='1' then
32           set_led <= "0000000";
33        elsif rising_edge (rotation_a) then
34           if rotation_b ='1' then
35              set_led <= set_led -1;
36              else set_led <= set_led +1;
37           end if;
38        end if;
39        end process;
40
41        process (clk)
42        begin
43        if rising_edge (clk) then
44           out_led(6) <= set_led(6);
45           out_led(5) <= set_led(5);
46           out_led(4) <= set_led(4);
47           out_led(3) <= set_led(3);
48           out_led(2) <= set_led(2);
49           out_led(1) <= set_led(1);
50           out_led(0) <= set_led(0);
51        end if;
52
53        end process;
54        end Behavioral;
55
56
```

set_help.vhd **Wed Jun 18 14:33:40 2008**

```
1   ----------------------------------
2   --Projekt: Senden mehrerer Zeichen
3   --Instanz: Set_Help
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity set_help is
13   PORT       ( help         : inout STD_LOGIC;
14                start_button : in STD_LOGIC
15              );
16  end set_help;
17
18  architecture Behavioral of set_help is
19
20  signal count : integer;
21  begin
22
23  process ( start_button)
24  begin
25     if rising_edge (start_Button) THEN
26        help <= '0';
27     end if;
28  end process;
29
30
31  end Behavioral;
32
33
```

Page 1

divider.vhd **Wed Jun 18 14:33:48 2008**

```
1   ----------------------------------
2   --Projekt: Senden mehrerer Zeichen
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14                  ext_clk_50MHz       : in          STD_LOGIC;
15                  test                : out STD_LOGIC;
16                  dev_clock_out       : inout   STD_LOGIC
17                );
18  end divider;
19
20
21  architecture Behavioral of divider is
22  signal divide_counter : integer := 0;
23
24  begin
25
26  Process(ext_clk_50MHz)
27          begin
28            if (rising_edge (ext_clk_50MHz)) then
29              if (divide_counter /= 2600) then
30                   divide_counter <= divide_counter + 1;
31              else
32                  divide_counter <= 0;
33                  dev_clock_out <= not (dev_clock_out);
34                       test <= dev_clock_out ;
35              end if;
36          end if;
37  end process;
38  end Behavioral;
39
40
```

TOP.ucf **Wed Jun 18 14:33:56 2008**

```
1   #------------------------------------------
2   #Projekt: Senden mehrerer Zeichen
3   #  User Constraint File
4   #------------------------------------------
5   INST "Inst_mac/dat_send"                          INIT = 1;
6   NET "button_send"          LOC = "K17" | PULLDOWN;
7   NET "clk_system"           LOC = "C9"  ;
8   NET "led_clk_control"      LOC = "F12" ;
9   NET "out_led<0>"           LOC = "E12" ;
10  NET "out_led<1>"           LOC = "E11" ;
11  NET "out_led<2>"           LOC = "F11" ;
12  NET "out_led<3>"           LOC = "C11" ;
13  NET "out_led<4>"           LOC = "D11" ;
14  NET "out_led<5>"           LOC = "E9"  ;
15  NET "out_led<6>"           LOC = "F9"  ;
16  NET "reset"                LOC = "V16" | PULLDOWN ;
17  NET "rot_a_in"             LOC = "K18" | PULLUP ;
18  NET "rot_b_in"             LOC = "G18" | PULLUP ;
19  NET "RS232_DCE_RXD"        LOC = "M14" ;
20
21
```

Page 1

A.8 Konstante empfangen

TOP.vhd Wed Jun 18 14:12:06 2008

```
1   ----------------------------------
2   -- Projekt: Konstante empfangen
3   -- Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity TOP is
11     PORT  (
12              signal_in      :  in STD_LOGIC;
13              led_out        :  out STD_LOGIC_VECTOR (8 downto 0);
14              test           :  out STD_LOGIC;
15              clk_system     :  in STD_LOGIC
16              );
17  end TOP;
18
19  architecture Behavioral of TOP is
20
21  signal sample_clk : STD_LOGIC;
22  signal clk_intern : STD_LOGIC;
23
24  COMPONENT MAC
25     PORT     (
26               signal_in           :  in STD_LOGIC;
27               led_out             : out STD_LOGIC_VECTOR (8 downto 0);
28               sample_clk          :  in STD_LOGIC
29
30              );
31  end COMPONENT;
32
33  COMPONENT divider
34      Port (
35                  ext_clk_50MHz : in  STD_LOGIC;
36                  test          : out STD_LOGIC;
37                  dev_clock_out : inout   STD_LOGIC
38             );
39  end COMPONENT;
40
41  begin
42
43  Inst_divider: divider PORT MAP
44     (            ext_clk_50MHz    => clk_system,
45                  test             => test,
46                  dev_clock_out    => clk_intern
47
48     );
49
50  Inst_mac: mac PORT MAP
51     (
52             signal_in         => signal_in,
53              led_out          => led_out,
54              sample_clk       => clk_intern
55     );
56
57
58
59  end Behavioral;
60
61
```

Page 1

MAC.vhd **Wed Jun 18 14:12:20 2008**

```
1   ----------------------------------
2   -- Projekt: Konstante empfangen
3   -- Instanz: Top
4   ----------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11
12  entity MAC is
13     PORT     (
14                signal_in           :  in STD_LOGIC;
15                led_out             :  out STD_LOGIC_VECTOR (8 downto 0);
16                sample_clk          :  in STD_LOGIC
17
18               );
19  end MAC;
20
21  architecture Behavioral of MAC is
22  signal wert      : integer ;
23  signal count     : integer;
24  signal count_led : integer;
25  signal help      : integer ;
26  signal ausgabe   : STD_LOGIC ;
27
28  begin
29  process (sample_clk)
30  begin
31  if rising_edge (sample_clk) then
32     if signal_in = '0' then
33        help <= 1;
34        wert <= 0;
35     end if;
36     if help = 1 then
37        count <= count + 1;
38        if signal_in = '1' then
39           wert <=  wert + 1;
40        end if;
41        if count = 15 then               -- 1 Bit empfangen
42           count_led <=  count_led + 1;
43           if wert > 8 then              -- Empf. Bit =1
44              ausgabe <= '1';
45              wert <= 0;
46           else                          -- Empf. Bit =0
47              ausgabe <= '0';
48              wert <= 0;
49           end if;
50           led_out(count_led) <= ausgabe;
51           count <= 0 ;
52           if count_led = 9 then
53              count_led <= 0;
54              help <= 0;
55              wert <= 0;
56           end if;
57        end if;
58     end if;
59  end if;
60  end process;
61  end Behavioral;
```

Page 1

DIVIDER.vhd **Wed Jun 18 14:12:27 2008**

```
1   ----------------------------------
2   -- Projekt: Konstante empfangen
3   -- Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity divider is
11      Port (
12                  ext_clk_50MHz : in          STD_LOGIC;
13                  test              : out STD_LOGIC;
14                  dev_clock_out : inout   STD_LOGIC
15              );
16  end divider;
17
18
19  architecture Behavioral of divider is
20  signal divide_counter : integer := 0;
21
22
23  begin
24
25  Process(ext_clk_50MHz)
26          begin
27            if (rising_edge (ext_clk_50MHz)) then
28              if (divide_counter /= 2600) then
29                  divide_counter <= divide_counter + 1;
30              else
31                  divide_counter <= 0;
32                  dev_clock_out <= not ( dev_clock_out);
33                  test <= dev_clock_out ;
34              end if;
35          end if;
36  end process;
37  end Behavioral;
38
39
40
```

TOP.ucf **Wed Jun 18 14:12:34 2008**

```
1   #----------------------------------
2   #Projekt: Konstante empfangen
3   #   User Constraint File
4   #----------------------------------
5   NET "clk_system"  LOC = "C9"   ;
6   NET "led_out<2>"  LOC = "F12"  ;
7   NET "led_out<3>"  LOC = "E12"  ;
8   NET "led_out<4>"  LOC = "E11"  ;
9   NET "led_out<5>"  LOC = "F11"  ;
10  NET "led_out<6>"  LOC = "C11"  ;
11  NET "led_out<7>"  LOC = "D11"  ;
12  NET "led_out<8>"  LOC = "E9"   ;
13  NET "signal_in"   LOC = "R7"   ;
```

Page 1

A.9 Senden und Empfangen

top.vhd Wed Jun 18 14:57:43 2008

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity top is
12                  PORT (
13                          clk_system      :   in     STD_LOGIC;
14                          rot_a_in        :   in     STD_LOGIC;
15                          signal_in       :   in     STD_LOGIC;
16                          rot_b_in        :   in     STD_LOGIC;
17                          reset           :   in     STD_LOGIC;
18                          button_send     :   in     STD_LOGIC;
19                          set_led         :   inout  STD_LOGIC_VECTOR (7 downto 0);
20                          led_out         :   out    STD_LOGIC_VECTOR (9 downto 0);
21                          RS232_DCE_RXD   :   out    STD_LOGIC;
22                          out_led_sig     :   out    STD_LOGIC_VECTOR (7 downto 0)
23                        );
24  end top;
25
26  architecture Behavioral of top is
27
28  signal rot_a               : STD_LOGIC;
29  signal rot_b               : STD_LOGIC;
30  signal dat_send_binaer     : STD_LOGIC_VECTOR (7 downto 0);
31  signal intern_clk          : STD_LOGIC;
32  signal test                : STD_LOGIC;
33  signal led_clk_control     : STD_LOGIC;
34  signal help                : STD_LOGIC;
35  signal edge_detect_sig     : STD_LOGIC;
36  signal clk16               : STD_LOGIC;
37
38
39  COMPONENT MAC_receive is
40      PORT        (
41                   signal_in              :   in STD_LOGIC;
42                   led_out                :   out STD_LOGIC_VECTOR (9 downto 0);

43                   sample_clk             :   in STD_LOGIC
44                   );
45  end COMPONENT;
46
47  COMPONENT set_help is
48   PORT       (
49                help         : inout STD_LOGIC;
50                start_button : in    STD_LOGIC
51               );
52  end COMPONENT;
53
54  COMPONENT MAC is
55          PORT(
56                   dat_send            : out   STD_LOGIC;
57                   start_button        : in    STD_LOGIC;
58                   set_help            : inout STD_LOGIC;
59                   dat_send_binaer     : in    STD_LOGIC_VECTOR (7 downto 0);
60                   mac_clk_in          : in    STD_LOGIC
```

Page 1

top.vhd Wed Jun 18 14:57:44 2008

```
61                      );
62          end COMPONENT;
63
64          COMPONENT SET is
65                  PORT (      rotation_a  :   in STD_LOGIC;
66                              rotation_b  :   in STD_LOGIC;
67                              reset       :   in STD_LOGIC;
68                              out_led     :   out STD_LOGIC_VECTOR (7 downto 0);
69                              clk         :   in STD_LOGIC;
70                              set_led     :   inout   STD_LOGIC_VECTOR (7 downto 0)
71                          );
72
73          end COMPONENT;
74
75
76
77          COMPONENT rotary_contact_filter is
78              Port (  rotation_a_in       :   in      STD_LOGIC;
79                      rotation_b_in       :   in      STD_LOGIC;
80                      dev_clk_in          :   in      STD_LOGIC;
81                      rotation_a          :   inout   STD_LOGIC;
82                      rotation_b          :   inout   STD_LOGIC
83                  );
84          end COMPONENT;
85
86          COMPONENT divider is
87              Port (
88                      ext_clk_50MHz : in     STD_LOGIC;
89                      test          : out    STD_LOGIC;
90                      dev_clock_out : inout  STD_LOGIC
91                  );
92          end COMPONENT;
93
94
95          COMPONENT divider16 is
96              Port (
97                      ext_clk_50MHz : in      STD_LOGIC;
98                      test          : out     STD_LOGIC;
99                      dev_clock_out : inout   STD_LOGIC
100                 );
101         end COMPONENT;
102
103
104         begin
105
106
107
108
109         Inst_set_help: set_help PORT MAP
110                 ( help          => help,
111                   start_button  => button_send
112                 );
113
114
115         Inst_divider: divider PORT MAP
116                 (
117                     ext_clk_50MHz       => clk_system,
118                     test                => led_clk_control,
119                     dev_clock_out       => intern_clk
120                 );
121
```

Page 2

top.vhd Wed Jun 18 14:57:44 2008

```
122         Inst_mac: mac PORT MAP
123                   (
124                       dat_send        => RS232_DCE_RXD,
125                       mac_clk_in      => intern_clk,
126                       start_button    => button_send,
127                       set_help        => help,
128                       dat_send_binaer => dat_send_binaer
129                   );
130
131         Inst_rotary_contact_filter: rotary_contact_filter PORT MAP
132                       (
133                       rotation_a_in   => rot_a_in,
134                       rotation_b_in   => rot_b_in,
135                       dev_clk_in      => intern_clk,
136                       rotation_a      => rot_a,
137                       rotation_b      => rot_b
138                       );
139
140         Inst_set: set PORT MAP
141                       (
142                           rotation_a      => rot_a,
143                           rotation_b      => rot_b,
144                           clk             => intern_clk,
145                           out_led         => out_led_sig,
146                           reset           => reset,
147                           set_led         => dat_send_binaer
148                       );
149
150         Inst_mac_receive: mac_receive PORT MAP
151                        (
152                        signal_in           => signal_in,
153                        led_out             => led_out,
154                        sample_clk          => clk16
155                        );
156
157
158
159         INST_divider16: divider16 PORT MAP
160              (
161                       ext_clk_50MHz => clk_system,
162                       test          => test,
163                       dev_clock_out => clk16
164                   );
165
166
167
168         end Behavioral;
169
170
```

MAC.vhd **Wed Jun 18 14:57:50 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Mac-Receive
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity MAC_receive is
12      PORT        (
13                   signal_in              :   in STD_LOGIC;
14                   led_out                :   out STD_LOGIC_VECTOR (9 downto 0);
15                   sample_clk             :   in STD_LOGIC
16                   );
17  end MAC_receive;
18
19  architecture Behavioral of MAC_receive is
20
21  signal help : integer ;
22
23  begin
24  process (sample_clk)
25
26  variable count_led   : integer;
27  variable wert        : integer ;
28  variable count       : integer;
29  variable vergleich   : STD_LOGIC_VECTOR (1 downto 0);
30  variable edge        : Std_LOGIC;
31  variable ausgabe     : STD_LOGIC ;
32   begin
33
34  if rising_edge (sample_clk) then
35   vergleich := vergleich(0)& signal_in;
36     case vergleich is
37        when "00" => edge := '0';
38        when "01" => edge := '1';
39        when "10" => edge := '1';
40        when "11" => edge := '0';
41        when others => vergleich := vergleich;
42     end case;
43
44     if signal_in = '0' then
45        help <= 1;
46        wert := 0;
47     end if;
48   if help = 1 then
49          count := count + 1;
50          if signal_in = '1' then
51              wert :=  wert + 1;
52          end if;
53          if count = 15 then
54              count_led :=  count_led + 1;
55              if wert > 8 then
56                 ausgabe := '1';
57                 wert := 0;
58              else
59                 ausgabe := '0';
60                 wert := 0;
61              end if;
```

MAC.vhd **Wed Jun 18 14:57:50 2008**

```
62                    if edge = '1' then
63                       count := 0;
64                    end if;
65                    led_out(count_led) <= ausgabe;
66                    count := 0 ;
67                    if count_led = 10 then
68                       count_led := 0;
69                       help <= 0;
70                       wert := 0;
71                    end if;
72                end if;
73             end if;
74          end if;
75          end process;
76          end Behavioral;
```

Page 2

MAC_send.vhd **Wed Jun 18 14:57:59 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Mac [send]
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity MAC is
12          PORT(
13                  dat_send            : out   STD_LOGIC;
14                  start_button        : in    STD_LOGIC;
15                  dat_send_binaer     : in    STD_LOGIC_VECTOR (7 downto 0);
16                  mac_clk_in          : in    STD_LOGIC;
17                  set_help            : inout STD_LOGIC
18                );
19  end MAC;
20
21  architecture Behavioral of MAC is
22
23  signal counter_index        : integer  ;
24  signal help                 :  STD_LOGIC;
25  signal dat_send_complete    : STD_LOGIC_VECTOR (9 downto 0);
26
27  begin
28  process (mac_clk_in)
29  begin
30  if rising_edge (mac_clk_in) then
31
32      dat_send_complete(9) <= '1';
33      dat_send_complete(8) <= dat_send_binaer(7);
34      dat_send_complete(7) <= dat_send_binaer(6);
35      dat_send_complete(6) <= dat_send_binaer(5);
36      dat_send_complete(5) <= dat_send_binaer(4);
37      dat_send_complete(4) <= dat_send_binaer(3);
38      dat_send_complete(3) <= dat_send_binaer(2);
39      dat_send_complete(2) <= dat_send_binaer(1);
40      dat_send_complete(1) <= dat_send_binaer(0);
41      dat_send_complete(0) <= '0';
42  end if;
43  end process;
44
45
46  process (mac_clk_in, start_button)
47  begin
48
49  if rising_edge (mac_clk_in) THEN
50     if (start_button = '1') AND (counter_index < 10) THEN
51        help <= set_help;
52        if help = '0' then
53              if counter_index < 10 THEN
54                  counter_index <= counter_index + 1;
55                  dat_send <= dat_send_complete(counter_index);
56              else counter_index <= 0;
57                help <= '1';
58              end if;
59          end if;
60     elsif (start_button = '0') AND (counter_index = 10) THEN
61        counter_index <= 0;
```

MAC_send.vhd **Wed Jun 18 14:57:59 2008**

```
62              help <= '1';
63            elsif start_button = '0' THEN
64              if help = '0' Then
65                 if counter_index > 0 THEN
66                       if Counter_index < 10 THEN
67                           counter_index <= counter_index + 1;
68                           dat_send <= dat_send_complete(counter_index);
69                       else Counter_index <= 0;
70                           help <= '1';
71                       end if;
72                   end if;
73                 end if;
74             end if;
75        end if;
76        end process;
77
78
79
80        end Behavioral;
81
```

Filter.vhd **Wed Jun 18 14:58:06 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Rotary_Contact_Filter
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity rotary_contact_filter is
12      Port (  rotation_a_in       :   in      STD_LOGIC;
13              rotation_b_in       :   in      STD_LOGIC;
14              dev_clk_in          :   in      STD_LOGIC;
15              rotation_a          :   inout   STD_LOGIC;
16              rotation_b          :   inout   STD_LOGIC
17            );
18  end rotary_contact_filter;
19
20  architecture Behavioral of rotary_contact_filter is
21  signal rotation_in : STD_LOGIC_VECTOR (1 downto 0);
22  begin
23  process(dev_clk_in)
24  begin
25      if (rising_edge (dev_clk_in)) then
26          rotation_in <= rotation_b_in & rotation_a_in;
27          case rotation_in is
28                  when "00" => rotation_a <= '0';
29                               rotation_b <= rotation_b;
30                  when "01" => rotation_a <= rotation_a;
31                               rotation_b <= '0';
32                  when "10" => rotation_a <= rotation_a;
33                               rotation_b <= '1';
34                  when "11" => rotation_a <= '1';
35                               rotation_b <= rotation_b;
36                  when others => rotation_a <= rotation_a;
37                               rotation_b <= rotation_b;
38          end case;
39      end if;
40  end process;
41
42  end Behavioral;
43
44
45
```

set.vhd **Wed Jun 18 14:58:12 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Set
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity SET is
13          PORT (  rotation_a      :   in    STD_LOGIC;
14                  rotation_b      :   in    STD_LOGIC;
15                  reset           :   in    STD_LOGIC;
16                  out_led         :   out   STD_LOGIC_VECTOR (7 downto 0);
17                  clk             :   in    STD_LOGIC;
18                  set_led         :   inout STD_LOGIC_VECTOR (7 downto 0)
19                );
20
21  end SET;
22
23  architecture Behavioral of SET is
24  begin
25
26  process (reset, rotation_a, rotation_b)
27  begin
28  if reset='1' then
29      set_led <= "00000000";
30  elsif rising_edge (rotation_a) then
31      if rotation_b ='1' then
32          set_led <= set_led -1;
33          else set_led <= set_led +1;
34      end if;
35  end if;
36  end process;
37
38  process (clk)
39  begin
40  if rising_edge (clk) then
41      out_led(7) <= set_led(7);
42      out_led(6) <= set_led(6);
43      out_led(5) <= set_led(5);
44      out_led(4) <= set_led(4);
45      out_led(3) <= set_led(3);
46      out_led(2) <= set_led(2);
47      out_led(1) <= set_led(1);
48      out_led(0) <= set_led(0);
49  end if;
50
51  end process;
52  end Behavioral;
53
54
55
56
57
58
```

Page 1

set_help.vhd **Wed Jun 18 14:58:20 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Set_Help
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity set_help is
13   PORT   (     help         : inout STD_LOGIC;
14                start_button : in STD_LOGIC
15           );
16  end set_help;
17
18  architecture Behavioral of set_help is
19
20  begin
21
22  process ( start_button)
23  begin
24    if rising_edge (start_Button) THEN
25       help <= '0';
26    end if;
27  end process;
28
29
30  end Behavioral;
31
32
```

DIVIDER16.vhd **Wed Jun 18 14:58:27 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Divider16
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity divider16 is
11      Port (
12                  ext_clk_50MHz     : in    STD_LOGIC;
13                  test              : out   STD_LOGIC;
14                  dev_clock_out     : inout STD_LOGIC
15              );
16  end divider16;
17
18
19  architecture Behavioral of divider16 is
20  signal divide_counter : integer := 0;
21
22
23  begin
24
25  Process(ext_clk_50MHz)
26          begin
27            if (rising_edge (ext_clk_50MHz)) then
28              if (divide_counter /= 162) then
29                  divide_counter <= divide_counter + 1;
30              else
31                  divide_counter <= 0;
32                  dev_clock_out <= not ( dev_clock_out);
33                  test <= dev_clock_out ;
34              end if;
35          end if;
36  end process;
37  end Behavioral;
38
39
```

Page 1

DIVIDER.vhd **Wed Jun 18 14:58:34 2008**

```
1   ----------------------------------
2   --Projekt: Senden und Empfangen
3   --Instanz: Divider
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity divider is
11      Port (
12                  ext_clk_50MHz : in     STD_LOGIC;
13                  test          : out    STD_LOGIC;
14                  dev_clock_out : inout  STD_LOGIC
15              );
16  end divider;
17
18
19  architecture Behavioral of divider is
20  signal divide_counter : integer := 0;
21
22
23  begin
24
25  Process(ext_clk_50MHz)
26          begin
27            if (rising_edge (ext_clk_50MHz)) then
28              if (divide_counter /= 2600) then
29                   divide_counter <= divide_counter + 1;
30              else
31                  divide_counter <= 0;
32                  dev_clock_out <= not ( dev_clock_out);
33                  test <= dev_clock_out ;
34              end if;
35          end if;
36  end process;
37  end Behavioral;
38
39
40
```

Page 1

top.ucf Wed Jun 18 14:58:40 2008

```
1   #----------------------------------------
2   #Projekt: Senden und Empfangen
3   #   User Constraint File
4   #----------------------------------------
5   NET "led_out<9>"  LOC = "F9"   ;
6   NET "led_out<8>"  LOC = "E9"   ;
7   NET "led_out<7>"  LOC = "D11"  ;
8   NET "led_out<6>"  LOC = "C11"  ;        # LED - PC -> Board
9   NET "led_out<5>"  LOC = "F11"  ;
10  NET "led_out<4>"  LOC = "E11"  ;
11  NET "led_out<3>"  LOC = "E12"  ;
12  NET "led_out<2>"  LOC = "F12"  ;
13  #---------------------------------------
14  #NET "out_led_sig<0>"  LOC = "F9"   ;
15  #NET "out_led_sig<1>"  LOC = "E9"   ;
16  #NET "out_led_sig<2>"  LOC = "D11"  ;
17  #NET "out_led_sig<3>"  LOC = "C11"  ;    #LED  Board -> PC
18  #NET "out_led_sig<4>"  LOC = "F11"  ;
19  #NET "out_led_sig<5>"  LOC = "E11"  ;
20  #NET "out_led_sig<6>"  LOC = "E12"  ;
21
22
23  INST "Inst_mac/dat_send"                          INIT = 1;
24  NET "button_send"           LOC = "K17" | PULLDOWN ;
25  NET "clk_system"            LOC = "C9"  ;
26  NET "reset"                 LOC = "V16" | PULLDOWN;
27  NET "rot_a_in"              LOC = "K18" | PULLUP ;
28  NET "rot_b_in"              LOC = "G18" | PULLUP ;
29  NET "RS232_DCE_RXD"         LOC = "M14" ;
30  NET "signal_in"             LOC = "R7"  ;
31
32
33
```

Page 1

A.10 LC - Display

top.vhd Wed Jun 25 11:58:31 2008

```
1   ----------------------------------------
2   --Projekt: LC Display
3   --Instanz: Top
4   ----------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity top is
12  PORT ( clock_50MHz  : in    STD_LOGIC;
13         lcd_dat      : inout STD_LOGIC_VECTOR (3 downto 0);
14         lcd_e        : out   STD_LOGIC;
15         lcd_rw       : out   STD_LOGIC;
16         lcd_rs       : out   STD_LOGIC
17         );
18  end top;
19
20  architecture Behavioral of top is
21
22  signal intern_clk   : STD_LOGIC;
23  signal count        : STD_LOGIC_VECTOR (10 downto 0);
24
25  Component divider is
26      Port (
27                  ext_clk_50MHz       : in       STD_LOGIC;
28                  dev_clock_out       : inout    STD_LOGIC
29                );
30  end Component;
31
32  Component mac is
33  PORT (   clock        : in    STD_LOGIC;
34           lcd_dat      : inout STD_LOGIC_VECTOR (3 downto 0);
35           lcd_e        : out   STD_LOGIC;
36           lcd_rw       : out   STD_LOGIC;
37           count        : in    STD_LOGIC_VECTOR (10 downto 0);
38           lcd_rs       : out   STD_LOGIC
39         );
40  end Component;
41
42  COMPONENT counter is
43   PORT (
44            clock  : in    STD_LOGIC;
45            count  : inout STD_LOGIC_VECTOR (10 downto 0)
46          );
47  end COMPONENT;
48  begin
49
50  Inst_counter: counter PORT MAP
51        (
52          clock  => intern_clk,
53          count  => count
54          );
55
56  Inst_mac: mac PORT MAP
57          (
58          clock           => intern_clk,
59          lcd_dat         => lcd_dat,
60          lcd_e           => lcd_e,
61          lcd_rw          => lcd_rw,
```

Page 1

top.vhd Wed Jun 25 11:58:31 2008

```
62                lcd_rs          => lcd_rs,
63                count           => count
64                );
65
66        Inst_divider: divider PORT MAP
67                (
68                ext_clk_50MHz   => Clock_50MHZ,
69                dev_clock_out   => intern_clk
70                );
71
72        end Behavioral;
73
74
75
76
77
78
79
80
81
82
83
84
85
86
```

mac.vhd **Wed Jun 25 11:58:43 2008**

```
1   ----------------------------------------
2   --Projekt: LC Display
3   --Instanz: Mac
4   ----------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity mac is
12  PORT ( clock        : in    STD_LOGIC;
13         lcd_dat      : out   STD_LOGIC_VECTOR (3 downto 0);
14         lcd_e        : out   STD_LOGIC;
15         lcd_rw       : out   STD_LOGIC;
16         count        : in    STD_LOGIC_VECTOR (10 downto 0);
17         lcd_rs       : out   STD_LOGIC
18         );
19  end mac;
20
21  architecture Behavioral of mac is
22  begin
23
24
25  process (clock)
26  variable onecycle: STD_LOGIC := '0';
27  begin
28  if rising_edge (clock) then
29  if onecycle = '0' Then
30  case count is
31      when "01011101110" => lcd_dat <= "0011";   ----750
32                            lcd_e   <= '0';
33                            lcd_rs  <= '0';
34                            lcd_rw  <= '0';
35      when "01011101111" => lcd_e   <= '1';      -----751
36      when "01110111101" => lcd_e   <= '1';      -----957
37      when "01111000011" => lcd_e   <= '1';      -----963
38      when "01111000110" => lcd_dat <= "0010";   -----966
39      when "01111000111" => lcd_e   <= '1';      -----967
40      when "01111001010" => lcd_dat <= "0001";   ----------------------
41      when "01111001011" => lcd_e   <= '1';
42      when "01111001110" => lcd_dat <= "1100";   --- function set
43      when "01111001111" => lcd_e   <= '1';      ----------------------
44      when "01111010001" => lcd_dat <= "0000";
45      when "01111010010" => lcd_e   <= '1';      ------- entry mode
46      when "01111010100" => lcd_dat <= "0110";
47      when "01111010101" => lcd_e   <= '1';      ----------------------
48      when "01111010110" => lcd_dat <= "0000";
49      when "01111010111" => lcd_e   <= '1';      -------- display control
50      when "01111011001" => lcd_dat <= "1100";
51      when "01111011010" => lcd_e   <= '1';      --------------------------
52      when "01111011100" => lcd_dat <= "0000";
53      when "01111011101" => lcd_e   <= '1';      --- display clear
54      when "01111011111" => lcd_dat <= "0001";
55      when "01111100000" => lcd_e   <= '1';       ------------------
56      when "11000000000" => lcd_dat <= "0000";
57      when "11000000010" => lcd_e   <= '1';
58      when "11000000100" => lcd_dat <= "1101";
59      when "11000000101" => lcd_e   <= '1';     ---------------
60      when "11000001000" => lcd_dat <= "0100";
61                            lcd_rs  <= '1';
```

Page 1

mac.vhd Wed Jun 25 11:58:43 2008

```
   62            when "11000001010" => lcd_e   <= '1';
   63            when "11000001100" => lcd_dat <= "1000";
   64            when "11000010000" => lcd_e   <= '1';        ----- H
   65            ------------------------------------------------
   66            when "11000011000" => lcd_dat <= "0100";
   67            when "11000011011" => lcd_e   <= '1';
   68            when "11000011110" => lcd_dat <= "0101";
   69            when "11000100010" => lcd_e   <= '1';     --------- E
   70            ------------------------------------------------
   71            when "11000101000" => lcd_dat <= "0100";
   72            when "11000101011" => lcd_e   <= '1';
   73            when "11000101111" => lcd_dat <= "1100";
   74            when "11000110010" => lcd_e   <= '1';     --------- L
   75            ------------------------------------------------
   76            when "11000110100" => lcd_dat <= "0100";
   77            when "11000111000" => lcd_e   <= '1';
   78            when "11000111010" => lcd_dat <= "1100";
   79            when "11000111100" => lcd_e   <= '1';     --------- L
   80            ------------------------------------------------
   81            when "11000111110" => lcd_dat <= "0100";
   82            when "11001000001" => lcd_e   <= '1';
   83            when "11001000100" => lcd_dat <= "1111";
   84            when "11001000111" => lcd_e   <= '1';     --------- O
   85            ------------------------------------------------------
   86            when "11001001001" => lcd_dat <= "0010";
   87            when "11001001011" => lcd_e   <= '1';
   88            when "11001001101" => lcd_dat <= "0000";
   89            when "11001001111" => lcd_e   <= '1';     ---------
   90            ------------------------------------------------------
   91            when "11001010001" => lcd_dat <= "0101";
   92            when "11001010011" => lcd_e   <= '1';
   93            when "11001010101" => lcd_dat <= "0111";
   94            when "11001010111" => lcd_e   <= '1';     --------- W
   95            ------------------------------------------------------
   96            when "11001011001" => lcd_dat <= "0100";
   97            when "11001011011" => lcd_e   <= '1';
   98            when "11001011101" => lcd_dat <= "1111";
   99            when "11001011111" => lcd_e   <= '1';     --------- O
  100            ------------------------------------------------------
  101            when "11001100011" => lcd_dat <= "0101";
  102            when "11001100110" => lcd_e   <= '1';
  103            when "11001101001" => lcd_dat <= "0010";
  104            when "11001101011" => lcd_e   <= '1';     --------- R
  105            ------------------------------------------------------
  106            when "11001110000" => lcd_dat <= "0100";
  107            when "11001110010" => lcd_e   <= '1';
  108            when "11001110100" => lcd_dat <= "1100";
  109            when "11001110111" => lcd_e   <= '1';     --------- L
  110            ------------------------------------------------------
  111            when "11010000000" => lcd_dat <= "0100";
  112            when "11010000011" => lcd_e   <= '1';
  113            when "11010000101" => lcd_dat <= "0100";
  114            when "11010001000" => lcd_e   <= '1';     --------- D
  115            ------------------------------------------------------
  116            when "11111111111" => onecycle := '1';
  117            when others        => lcd_e   <= '0';
  118        end case;
  119        end if;
  120        end if;
  121        end process;
  122        end Behavioral;
```

Page 2

counter.vhd **Wed Jun 25 11:58:53 2008**

```
1   ----------------------------------------
2   --Projekt: LC Display
3   --Instanz: Counter
4   ----------------------------------------
5   use IEEE.STD_LOGIC_1164.ALL;
6   use IEEE.STD_LOGIC_ARITH.ALL;
7   use IEEE.STD_LOGIC_UNSIGNED.ALL;
8
9
10  entity counter is
11   PORT (
12            clock  : in    STD_LOGIC;
13            count  : inout STD_LOGIC_VECTOR (10 downto 0)
14         );
15  end counter;
16
17  architecture Behavioral of counter is
18
19  begin
20  process (clock)
21  begin
22    if rising_edge (clock) THEN
23        count <= count + '1';
24    end if;
25  end process;
26  end Behavioral;
27
28
29
30
31
32
33
34
35
36
37
38
```

Page 1

divider.vhd **Wed Jun 25 11:59:00 2008**

```
1   ----------------------------------------
2   --Projekt: LC Display
3   --Instanz: Divider
4   ----------------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity divider is
13      Port (
14                  ext_clk_50MHz       : in      STD_LOGIC;
15                  dev_clock_out       : inout   STD_LOGIC --25 kHz
16                );
17  end divider;
18
19
20  architecture Behavioral of divider is
21  signal divide_counter : integer := 0;
22
23
24  begin
25
26  Process(ext_clk_50MHz)
27          begin
28            if (rising_edge (ext_clk_50MHz)) then
29              if (divide_counter /= 1000) then
30                   divide_counter <= divide_counter + 1;
31              else
32                  divide_counter <= 0;
33                  dev_clock_out <= not (dev_clock_out);
34              end if;
35          end if;
36  end process;
37  end Behavioral;
38
```

top.ucf Wed Jun 25 11:59:08 2008

```
1   #----------------------------------
2   #Projekt: LC Display
3   #      User Constraint File
4   #----------------------------------
5   NET "clock_50MHz" LOC = "C9"   ;
6   NET "lcd_dat<0>"  LOC = "R15"  ;
7   NET "lcd_dat<1>"  LOC = "R16"  ;
8   NET "lcd_dat<2>"  LOC = "P17"  ;
9   NET "lcd_dat<3>"  LOC = "M15"  ;
10  NET "lcd_e"       LOC = "M18"  ;
11  NET "lcd_rs"      LOC = "L18"  ;
12  NET "lcd_rw"      LOC = "L17"  ;
13
14
```

Page 1

A.11 Verstärkung einstellen

top.vhd Wed Jun 18 16:10:52 2008

```
1   ----------------------------------
2   --Projekt: Verstärkung einstellen
3   -- Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity top is
12   PORT (
13         clk_system   : in    STD_LOGIC ;
14         start_button : in    STD_LOGIC;
15         sdi          : out   STD_LOGIC;
16         sck_out      : inout STD_LOGIC;
17         amp_cs       : out   STD_LOGIC
18         );
19  end top;
20
21  architecture Behavioral of top is
22
23  signal help               : STD_LOGIC;
24  signal count_set_gain_sig  : STD_LOGIC_VECTOR (3 downto 0);
25
26
27  COMPONENT setgain is
28    PORT (
29          amp_cs  : out     STD_LOGIC;
30          sck     : in      STD_LOGIC;   -- 1 MHz Takt
31          sdi     : out     STD_LOGIC;
32          count   : in      STD_LOGIC_VECTOR (3 downto 0)
33         );
34  end COMPONENT;
35
36  COMPONENT divider is
37      Port (
38                  ext_clk_50MHz     : in     STD_LOGIC;
39                  dev_clock_out     : inout  STD_LOGIC
40           );
41  end COMPONENT;
42
43  COMPONENT count_set_gain is
44     PORT (
45           clock            : in    STD_LOGIC;
46           start_button     : in    STD_LOGIC;
47           set_help         : in    STD_LOGIC;
48           count_set_gain_i : inout STD_LOGIC_VECTOR( 3 downto 0)
49          );
50
51  end COMPONENT;
52
53  COMPONENT set_help is
54   PORT       ( help         : inout STD_LOGIC;
55                start_button : in    STD_LOGIC
56               );
57  end COMPONENT;
58
59
60  begin
61
```

Page 1

top.vhd Wed Jun 18 16:10:52 2008

```
62          Inst_setgain: setgain PORT MAP
63           (
64                  amp_cs  => amp_cs,
65                  sck     => sck_out,
66                  sdi     => sdi,
67                  count   => count_set_gain_sig
68            );
69
70
71          Inst_divider: divider PORT MAP
72           ( ext_clk_50MHz => clk_system,
73             dev_clock_out => sck_out
74           );
75
76          Inst_set_help: set_help PORT MAP
77           (
78            help         => help,
79            start_button => start_button
80           );
81
82          Inst_count_set_gain: count_set_gain PORT MAP
83           (
84           clock             => sck_out,
85           start_button      => start_button,
86           set_help          => help,
87           count_set_gain_i  => count_set_gain_sig
88           );
89
90          end Behavioral;
91
92
```

setgain.vhd **Wed Jun 18 16:10:58 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung einstellen
3   -- Instanz: Setgain
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity setgain is
11    PORT (
12          amp_cs  : out STD_LOGIC;
13          sck     : in  STD_LOGIC;   -- 1 MHz Takt
14          sdi     : out STD_LOGIC;
15          count   : in  STD_LOGIC_VECTOR (3 downto 0)
16         );
17  end setgain;
18
19  architecture Behavioral of setgain is
20
21  signal gain : STD_LOGIC_VECTOR (7 downto 0) := "00010001"  ; -- gain = -1 für Kanal A un
22  -------------------
23  --gain -1  : 0001
24  --     -2  : 0010
25  --     -5  : 0011
26  --     -10 : 0100
27  --     -20 : 0101
28  --     -50 : 0110
29  --     -100: 0111
30  --------------------
31  begin
32
33  process (sck)
34  begin
35   if rising_edge (sck) THEN
36     case count is
37        when "0001" => amp_cs <= '0';
38        when "0010" => sdi    <= gain(7);
39        when "0011" => sdi    <= gain(6);
40        when "0100" => sdi    <= gain(5);
41        when "0101" => sdi    <= gain(4);
42        when "0110" => sdi    <= gain(3);
43        when "0111" => sdi    <= gain(2);
44        when "1000" => sdi    <= gain(1);
45        when "1001" => sdi    <= gain(0);
46        when "1010" => amp_cs <= '1';
47        when others => amp_cs <= '1';
48     end case;
49   end if;
50  end process;
51  end Behavioral;
52
53
```

Page 1

count_set_gain.vhd **Wed Jun 18 16:11:08 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung einstellen
3   -- Instanz: Count_set_Gain
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity count_set_gain is
12
13     PORT (
14           clock          : in  STD_LOGIC;
15           start_button   : in  STD_LOGIC;
16           set_help       : in STD_LOGIC;
17           count_set_gain_i : inout STD_LOGIC_VECTOR ( 3 downto 0)
18          );
19
20  end count_set_gain;
21
22  architecture Behavioral of count_set_gain is
23
24  signal Counter_index  : integer;
25  signal help           : STD_LOGIC;
26
27  begin
28  process (clock, start_button)
29  begin
30  if rising_edge (clock) THEN
31      if (start_button = '1') AND (counter_index < 11) THEN
32        help <= set_help;
33          if help = '0' then
34              if counter_index < 12 THEN
35                  counter_index <= counter_index + 1;
36                  count_set_gain_i <= count_set_gain_i + '1';
37              else counter_index <= 0;
38                   help <= '1';
39              end if;
40          end if;
41      elsif (start_button = '0') AND (counter_index = 11) THEN
42          counter_index <= 0;
43          help <= '1';
44      elsif start_button = '0' THEN
45       if help = '0' Then
46           if counter_index > 0 THEN
47                if Counter_index < 11 THEN
48                    counter_index <= counter_index + 1;
49                    count_set_gain_i <= count_set_gain_i + 1;
50                else Counter_index <= 0;
51                     help <= '1';
52                end if;
53           end if;
54         end if;
55      end if;
56  end if;
57  end process;
58  end Behavioral;
59
60
```

set_help.vhd **Wed Jun 18 16:11:16 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung einstellen
3   -- Instanz: Set_Help
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity set_help is
13   PORT       ( help         : inout STD_LOGIC;
14                start_button : in    STD_LOGIC
15      );
16  end set_help;
17
18  architecture Behavioral of set_help is
19
20  begin
21
22  process ( start_button)
23  begin
24    if rising_edge (start_Button) THEN
25      help <= '0';
26    end if;
27  end process;
28
29
30  end Behavioral;
31
```

divider.vhd **Wed Jun 18 16:11:25 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung einstellen
3   -- Instanz: Divider
4   ----------------------------------
5
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use IEEE.STD_LOGIC_ARITH.ALL;
9   use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11  entity divider is
12      Port (
13                  ext_clk_50MHz     : in     STD_LOGIC;
14                  dev_clock_out     : inout  STD_LOGIC   -- 1MhZ
15              );
16  end divider;
17
18
19  architecture Behavioral of divider is
20  signal divide_counter : integer := 0;
21
22
23  begin
24
25  Process(ext_clk_50MHz)
26          begin
27            if (rising_edge (ext_clk_50MHz)) then
28              if (divide_counter /= 25) then
29                  divide_counter <= divide_counter + 1;
30              else
31                  divide_counter <= 0;
32                  dev_clock_out <= not ( dev_clock_out);
33              end if;
34          end if;
35  end process;
36  end Behavioral;
37
```

top.ucf **Wed Jun 18 16:11:32 2008**

```
1   #--------------------------------
2   #Projekt: Verstärkung einstellen
3   #   User Constraint File
4   #--------------------------------
5   NET "amp_cs"         LOC = "N7"  ;
6   NET "clk_system"     LOC = "C9"  ;
7   NET "sck_out"        LOC = "U16" ;
8   NET "sdi"            LOC = "T4"  ;
9   NET "start_button"   LOC = "K17" ;
10
11
```

Page 1

A.12 Verstärkung anzeigen

top.vhd Wed Jun 18 16:23:08 2008

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Top
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity top is
12   PORT (
13         clk_system            : in      STD_LOGIC ;
14         start_button          : in      STD_LOGIC;
15         sdi                   : out     STD_LOGIC;
16         sck_out               : inout   STD_LOGIC;
17         amp_do                : in      STD_LOGIC;
18         amp_cs                : inout   STD_LOGIC;
19         lcd_dat               : inout   STD_LOGIC_VECTOR (3 downto 0);
20         lcd_rw                : out     STD_LOGIC;
21         lcd_rs                : out     STD_LOGIC;
22         lcd_e                 : out     STD_LOGIC
23        );
24  end top;
25
26  architecture Behavioral of top is
27
28
29  signal intern_clk          : STD_LOGIC;
30  signal count_set_gain_sig  : STD_LOGIC_VECTOR (8 downto 0);
31  signal Send_Button_Pressed : STD_LOGIC;
32  signal clk_50kHZ           : STD_LOGIC;
33  signal count               : STD_LOGIC_VECTOR (12 downto 0);
34  signal z1                  : STD_LOGIC_VECTOR (3 downto 0);
35  signal z2                  : STD_LOGIC_VECTOR (3 downto 0);
36  signal z3                  : STD_LOGIC_VECTOR (3 downto 0);
37  signal r1                  : STD_LOGIC_VECTOR (3 downto 0);
38  signal r2                  : STD_LOGIC_VECTOR (3 downto 0);
39  signal r3                  : STD_LOGIC_VECTOR (3 downto 0);
40  signal SPI_read_HighNibble : STD_LOGIC_VECTOR (3 downto 0);
41  signal SPI_read_LowNibble  : STD_LOGIC_VECTOR (3 downto 0);
42
43
44
45
46
47  COMPONENT setgain is
48    PORT (
49          amp_cs              : out     STD_LOGIC;
50          clock               : in      STD_LOGIC;
51          sdi                 : out     STD_LOGIC;
52          sck_out             : out     STD_LOGIC;
53          busy                : out     STD_LOGIC;
54          SPI_read_HighNibble : out     STD_LOGIC_VECTOR (3 downto 0);
55          SPI_read_LowNibble  : out     STD_LOGIC_VECTOR (3 downto 0);
56          amp_do              : in      STD_LOGIC;
57          count               : in      STD_LOGIC_VECTOR (8 downto 0)
58         );
59  end COMPONENT;
60
61
```

Page 1

top.vhd Wed Jun 18 16:23:08 2008

```
 62
 63        COMPONENT count_set_gain is
 64           PORT (
 65                 clock            : in  STD_LOGIC;
 66                 start_button     : in  STD_LOGIC;
 67                 button_pressed   : in STD_LOGIC;
 68                 count_set_gain_i : inout STD_LOGIC_VECTOR( 8 downto 0)
 69                );
 70
 71        end COMPONENT;
 72
 73
 74        COMPONENT counter_lcd is
 75         PORT (
 76                clock  : in STD_LOGIC;
 77                count  : inout STD_LOGIC_VECTOR (12 downto 0)
 78               );
 79        end COMPONENT;
 80
 81
 82
 83        COMPONENT divider_lcd is
 84            Port (
 85                       ext_clk_50MHz       : in      STD_LOGIC;
 86                       dev_clock_out       : inout   STD_LOGIC
 87                  );
 88        end COMPONENT;
 89
 90        COMPONENT mac_lcd is
 91        PORT ( clock        : in    STD_LOGIC;
 92               lcd_dat      : out   STD_LOGIC_VECTOR (3 downto 0);
 93               lcd_e        : out   STD_LOGIC;
 94               lcd_rw       : out   STD_LOGIC;
 95               count        : in    STD_LOGIC_VECTOR (12 downto 0);
 96               lcd_rs       : out   STD_LOGIC;
 97               z1           : in    STD_LOGIC_VECTOR (3 downto 0);
 98               z2           : in    STD_LOGIC_VECTOR (3 downto 0);
 99               z3           : in    STD_LOGIC_VECTOR (3 downto 0);
100               r1           : in    STD_LOGIC_VECTOR (3 downto 0);
101               r2           : in    STD_LOGIC_VECTOR (3 downto 0);
102               r3           : in    STD_LOGIC_VECTOR (3 downto 0)
103               );
104        end COMPONENT;
105
106
107
108        COMPONENT set_gain_z_lcd is
109          PORT (
110                 z1                   : out STD_LOGIC_VECTOR (3 downto 0);
111                 z2                   : out STD_LOGIC_VECTOR (3 downto 0);
112                 z3                   : out STD_LOGIC_VECTOR (3 downto 0);
113                 r1                   : out STD_LOGIC_VECTOR (3 downto 0);
114                 r2                   : out STD_LOGIC_VECTOR (3 downto 0);
115                 r3                   : out STD_LOGIC_VECTOR (3 downto 0);
116                 SPI_read_LowNibble   : in  STD_LOGIC_VECTOR (3 downto 0);
117                 SPI_read_HighNibble  : in  STD_LOGIC_VECTOR (3 downto 0);
118                 clk                  : in  STD_LOGIC
119                 );
120        end COMPONENT;
121
122
```

Page 2

top.vhd Wed Jun 18 16:23:08 2008

```
123         begin
124
125
126
127
128         Inst_set_gain_z_lcd: set_gain_z_lcd PORT MAP
129             (
130                 z1                      => z1,
131                 z2                      => z2,
132                 z3                      => z3,
133                 r1                      => r1,
134                 r2                      => r2,
135                 r3                      => r3,
136                 SPI_read_HighNibble     => SPI_read_HighNibble,
137                 SPI_read_LowNibble      => SPI_read_LowNibble,
138                 clk                     => clk_system
139             );
140
141         Inst_counter_lcd: counter_lcd PORT MAP
142                 (
143                clock    => clk_50kHz,
144                count    => count
145                );
146
147         Inst_mac_lcd: mac_lcd PORT MAP
148                 (
149                 clock           => clk_50kHZ,
150                 lcd_dat         => lcd_dat,
151                 lcd_e           => lcd_e,
152                 lcd_rw          => lcd_rw,
153                 lcd_rs          => lcd_rs,
154                 z1              => z1,
155                 z2              => z2,
156                 z3              => z3,
157                 r1              => r1,
158                 r2              => r2,
159                 r3              => r3,
160                 count           => count
161                 );
162
163         Inst_divider_lcd: divider_lcd PORT MAP
164                  (
165                    ext_clk_50MHz => clk_system,
166                    dev_clock_out => clk_50kHz
167                  );
168
169
170
171         Inst_setgain: setgain PORT MAP
172          (
173                 amp_cs                  => amp_cs,
174                 clock                   => clk_system,
175                 sdi                     => sdi,
176                 amp_do                  => amp_do,
177                 sck_out                 => sck_out,
178                 busy                    => Send_Button_Pressed,
179                 SPI_read_HighNibble     => SPI_read_HighNibble,
180                 SPI_read_LowNibble      => SPI_read_LowNibble,
181                 count                   => count_set_gain_sig
182           );
183
```

Page 3

top.vhd Wed Jun 18 16:23:08 2008

```
184
185
186
187
188         Inst_count_set_gain: count_set_gain PORT MAP
189          (
190            clock            => clk_system,
191            start_button     => start_button,
192            count_set_gain_i => count_set_gain_sig,
193            button_pressed   => Send_Button_Pressed
194          );
195
196
197         end Behavioral;
```

mac_lcd.vhd **Wed Jun 18 16:23:20 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Mac_LCD
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity mac_lcd is
12  PORT ( clock        : in    STD_LOGIC;
13         lcd_dat      : out   STD_LOGIC_VECTOR (3 downto 0);
14         lcd_e        : out   STD_LOGIC;
15         lcd_rw       : out   STD_LOGIC;
16         count        : in    STD_LOGIC_VECTOR (12 downto 0);
17         lcd_rs       : out   STD_LOGIC;
18         z1           : in    STD_LOGIC_VECTOR (3 downto 0);
19         z2           : in    STD_LOGIC_VECTOR (3 downto 0);
20         z3           : in    STD_LOGIC_VECTOR (3 downto 0);
21         r1           : in    STD_LOGIC_VECTOR (3 downto 0);
22         r2           : in    STD_LOGIC_VECTOR (3 downto 0);
23         r3           : in    STD_LOGIC_VECTOR (3 downto 0)
24         );
25  end mac_lcd;
26
27  architecture Behavioral of mac_lcd is
28  begin
29
30
31  process (clock)
32  variable Init_LCD_finished: STD_LOGIC := '0';
33  begin
34  if rising_edge (clock) then
35  if Init_LCD_finished = '0' Then
36    case count is
37      when "0001011101110" => lcd_dat <= "0011"; ----750
38                              lcd_e   <= '0';
39                              lcd_rs  <= '0';
40                              lcd_rw  <= '0';
41      when "0001011101111" => lcd_e   <= '1';      -----751
42      when "0001110111101" => lcd_e   <= '1';      -----957
43      when "0001111000011" => lcd_e   <= '1';      -----963
44      when "0001111000110" => lcd_dat <= "0010";   -----966
45      when "0001111000111" => lcd_e   <= '1';      -----967
46      when "0001111001010" => lcd_dat <= "0001";   ----------------------
47      when "0001111001011" => lcd_e   <= '1';
48      when "0001111001110" => lcd_dat <= "1100";   --- function set
49      when "0001111001111" => lcd_e   <= '1';      ----------------------
50      when "0001111010001" => lcd_dat <= "0000";
51      when "0001111010010" => lcd_e   <= '1';      ------- entry mode
52      when "0001111010100" => lcd_dat <= "0110";
53      when "0001111010101" => lcd_e   <= '1';      ----------------------
54      when "0001111010110" => lcd_dat <= "0000";
55      when "0001111010111" => lcd_e   <= '1';      -------- display control
56      when "0001111011001" => lcd_dat <= "1100";
57      when "0001111011010" => lcd_e   <= '1';      --------------------------
58      when "0001111011100" => lcd_dat <= "0000";
59      when "0001111011101" => lcd_e   <= '1';      --- display clear
60      when "0001111011111" => lcd_dat <= "0001";
61      when "0001111100000" => lcd_e   <= '1';    -----------------
```

Page 1

mac_lcd.vhd Wed Jun 18 16:23:20 2008

```
 62            when "0011000000000" => lcd_dat <= "0000";
 63            when "0011000000010" => lcd_e   <= '1';
 64            when "0011000000100" => lcd_dat <= "1101";
 65            when "0011000000101" => lcd_e   <= '1';     ----------------
 66            when "0011000001000" => lcd_dat <= "0100";
 67                                    lcd_rs  <= '1';
 68            when "0011000001010" => lcd_e   <= '1';
 69            when "0011000001100" => lcd_dat <= "0111";
 70            when "0011000010000" => lcd_e   <= '1';         ----- G
 71             ------------------------------------------------
 72            when "0011000011000" => lcd_dat <= "0110";
 73            when "0011000011011" => lcd_e   <= '1';
 74            when "0011000011110" => lcd_dat <= "0001";
 75            when "0011000100010" => lcd_e   <= '1';      --------- a
 76             -----------------------------------------------
 77            when "0011000101000" => lcd_dat <= "0110";
 78            when "0011000101011" => lcd_e   <= '1';
 79            when "0011000101111" => lcd_dat <= "1001";
 80            when "0011000110010" => lcd_e   <= '1';      --------- i
 81             ----------------------------------------------
 82            when "0011000110100" => lcd_dat <= "0110";
 83            when "0011000111000" => lcd_e   <= '1';
 84            when "0011000111010" => lcd_dat <= "1110";
 85            when "0011000111100" => lcd_e   <= '1';      --------- n
 86             ---------------------------------------------
 87            when "0011000111110" => lcd_dat <= "0100";
 88            when "0011001000001" => lcd_e   <= '1';
 89            when "0011001000100" => lcd_dat <= "0001";
 90            when "0011001000111" => lcd_e   <= '1';      --------- A
 91             ------------------------------------------------------
 92            when "0011001001001" => lcd_dat <= "0011";
 93            when "0011001001011" => lcd_e   <= '1';
 94            when "0011001001101" => lcd_dat <= "1010";
 95            when "0011001001111" => lcd_e   <= '1';      --------- :
 96             ------------------------------------------------------
 97            when "0011001010001" => lcd_dat <= "0010";
 98            when "0011001010011" => lcd_e   <= '1';
 99            when "0011001010101" => lcd_dat <= "0000";
100            when "0011001010111" => lcd_e   <= '1';      ---------
101             ------------------------------------------------------
102            when "0011001011001" => lcd_dat <= "0011";
103            when "0011001011011" => lcd_e   <= '1';
104            when "0011001011101" => lcd_dat <= z1;
105            when "0011001011111" => lcd_e   <= '1';      --------- Z1
106             ------------------------------------------------------
107            when "0011001100011" => lcd_dat <= "0011";
108            when "0011001100110" => lcd_e   <= '1';
109            when "0011001101001" => lcd_dat <= z2;
110            when "0011001101011" => lcd_e   <= '1';      --------- z2
111             ------------------------------------------------------
112            when "0011001110000" => lcd_dat <= "0011";
113            when "0011001110010" => lcd_e   <= '1';
114            when "0011001110100" => lcd_dat <= z3;
115            when "0011001110111" => lcd_e   <= '1';      --------- z3
116             ------------------------------------------------------
117            when "0011010000000" => lcd_dat <= "0010";
118            when "0011010000011" => lcd_e   <= '1';
119            when "0011010000101" => lcd_dat <= "0000";
120            when "0011010001000" => lcd_e   <= '1';      ---------
121             ------------------------------------------------------
122            when "0011010001100" => lcd_dat <= "0100";
```

mac_lcd.vhd Wed Jun 18 16:23:20 2008

```
123             when "0011010001110" => lcd_e   <= '1';
124             when "0011010010000" => lcd_dat <= "0010";
125             when "0011010010100" => lcd_e   <= '1';     --------- B
126             ------------------------------------------------------------
127             when "0011010011100" => lcd_dat <= "0011";
128             when "0011010011110" => lcd_e   <= '1';
129             when "0011010100000" => lcd_dat <= "1010";
130             when "0011010100100" => lcd_e   <= '1';     --------- :
131             ------------------------------------------------------------
132             when "0011010110000" => lcd_dat <= "0011";
133             when "0011010110010" => lcd_e   <= '1';
134             when "0011010110100" => lcd_dat <= r1;
135             when "0011010110111" => lcd_e   <= '1';     --------- r1
136              ----------------------------------------------------------
137             when "0011100000000" => lcd_dat <= "0011";
138             when "0011100000010" => lcd_e   <= '1';
139             when "0011100000100" => lcd_dat <= r2;
140             when "0011100000111" => lcd_e   <= '1';     --------- r2
141              ----------------------------------------------------------
142             when "0011100100000" => lcd_dat <= "0011";
143             when "0011100100010" => lcd_e   <= '1';
144             when "0011100100100" => lcd_dat <= r3;
145             when "0011100100111" => lcd_e   <= '1';     --------- r3
146              ----------------------------------------------------------
147
148             when "1111111111111" => Init_LCD_finished := '1';
149             when others          => lcd_e   <= '0';
150         end case;
151
152
153
154
155         elsif Init_LCD_finished = '1' then
156         case count is
157
158             when "0000000000000" => lcd_dat <= "0000";
159                                     lcd_rs  <= '0';
160             when "0000000000011" => lcd_e   <= '1';
161             when "0000000001110" => lcd_dat <= "0001";
162             when "0000000010010" => lcd_e   <= '1';     --------- CURSOR HOME
163              ----------------------------------------------------------
164             when "0011000000101" => lcd_rs  <= '1';
165             when "0011000001000" => lcd_dat <= "0100";
166             when "0011000001010" => lcd_e   <= '1';
167             when "0011000001100" => lcd_dat <= "0111";
168             when "0011000010000" => lcd_e   <= '1';       ----- G
169              ----------------------------------------------------------
170             when "0011000011000" => lcd_dat <= "0110";
171             when "0011000011011" => lcd_e   <= '1';
172             when "0011000011110" => lcd_dat <= "0001";
173             when "0011000100010" => lcd_e   <= '1';     --------- a
174              ----------------------------------------------------------
175             when "0011000101000" => lcd_dat <= "0110";
176             when "0011000101011" => lcd_e   <= '1';
177             when "0011000101111" => lcd_dat <= "1001";
178             when "0011000110010" => lcd_e   <= '1';     --------- i
179              ----------------------------------------------------------
180             when "0011000110100" => lcd_dat <= "0110";
181             when "0011000111000" => lcd_e   <= '1';
182             when "0011000111010" => lcd_dat <= "1110";
183             when "0011000111100" => lcd_e   <= '1';     --------- n
```

Page 3

mac_lcd.vhd Wed Jun 18 16:23:20 2008

```
184             ---------------------------------------------------
185           when "0011000111110" => lcd_dat <= "0100";
186           when "0011001000001" => lcd_e   <= '1';
187           when "0011001000100" => lcd_dat <= "0001";
188           when "0011001000111" => lcd_e   <= '1';     --------- A
189             ---------------------------------------------------
190           when "0011001001001" => lcd_dat <= "0011";
191           when "0011001001011" => lcd_e   <= '1';
192           when "0011001001101" => lcd_dat <= "1010";
193           when "0011001001111" => lcd_e   <= '1';     --------- :
194             ---------------------------------------------------
195           when "0011001010001" => lcd_dat <= "0010";
196           when "0011001010011" => lcd_e   <= '1';
197           when "0011001010101" => lcd_dat <= "0000";
198           when "0011001010111" => lcd_e   <= '1';     ---------
199             ---------------------------------------------------
200           when "0011001011001" => lcd_dat <= "0011";
201           when "0011001011011" => lcd_e   <= '1';
202           when "0011001011101" => lcd_dat <= z1;
203           when "0011001011111" => lcd_e   <= '1';     --------- Z1
204             ---------------------------------------------------
205           when "0011001100011" => lcd_dat <= "0011";
206           when "0011001100110" => lcd_e   <= '1';
207           when "0011001101001" => lcd_dat <= z2;
208           when "0011001101011" => lcd_e   <= '1';     --------- z2
209             ---------------------------------------------------
210           when "0011001110000" => lcd_dat <= "0011";
211           when "0011001110010" => lcd_e   <= '1';
212           when "0011001110100" => lcd_dat <= z3;
213           when "0011001110111" => lcd_e   <= '1';     --------- z3
214             ---------------------------------------------------
215           when "0011010000000" => lcd_dat <= "0010";
216           when "0011010000011" => lcd_e   <= '1';
217           when "0011010000101" => lcd_dat <= "0000";
218           when "0011010001000" => lcd_e   <= '1';     ---------
219             ---------------------------------------------------
220           when "0011010001100" => lcd_dat <= "0100";
221           when "0011010001110" => lcd_e   <= '1';
222           when "0011010010000" => lcd_dat <= "0010";
223           when "0011010010100" => lcd_e   <= '1';     --------- B
224           -----------------------------------------------------------
225           when "0011010011100" => lcd_dat <= "0011";
226           when "0011010011110" => lcd_e   <= '1';
227           when "0011010100000" => lcd_dat <= "1010";
228           when "0011010100100" => lcd_e   <= '1';     --------- :
229           -----------------------------------------------------------
230           when "0011010110000" => lcd_dat <= "0011";
231           when "0011010110010" => lcd_e   <= '1';
232           when "0011010110100" => lcd_dat <= r1;
233           when "0011010110111" => lcd_e   <= '1';     --------- r1
234             ---------------------------------------------------
235           when "0011100000000" => lcd_dat <= "0011";
236           when "0011100000010" => lcd_e   <= '1';
237           when "0011100000100" => lcd_dat <= r2;
238           when "0011100000111" => lcd_e   <= '1';     --------- r2
239             ---------------------------------------------------
240           when "0011100100000" => lcd_dat <= "0011";
241           when "0011100100010" => lcd_e   <= '1';
242           when "0011100100100" => lcd_dat <= r3;
243           when "0011100100111" => lcd_e   <= '1';     --------- r3
244             ---------------------------------------------------
```

Page 4

mac_lcd.vhd Wed Jun 18 16:23:20 2008

```
245             when "1111111111111" => Init_LCD_finished := '1';
246             when others          => lcd_e   <= '0';
247
248         end case;
249         end if;
250         end if;
251
252         end process;
253         end Behavioral;
```

setgain.vhd **Wed Jun 18 16:23:30 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Setgain
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10  entity setgain is
11    PORT (
12            amp_cs                : out STD_LOGIC;
13            clock                 : in  STD_LOGIC;
14            sdi                   : out STD_LOGIC;
15            sck_out               : out STD_LOGIC;
16            amp_do                : in  STD_LOGIC;
17            busy                  : out STD_LOGIC;
18            SPI_read_LowNibble    : out STD_LOGIC_VECTOR (3 downto 0);
19            SPI_read_HighNibble   : out STD_LOGIC_VECTOR (3 downto 0);
20            count                 : in  STD_LOGIC_VECTOR (8 downto 0)
21          );
22  end setgain;
23
24  architecture Behavioral of setgain is
25
26  signal gain : STD_LOGIC_VECTOR (7 downto 0) := "00010101" ;
27
28  ------------------
29  --gain -1  : 0001
30  --     -2  : 0010
31  --     -5  : 0011
32  --     -10 : 0100
33  --     -20 : 0101
34  --     -50 : 0110
35  --     -100: 0111
36  ------------------
37  begin
38
39
40
41  process (clock)
42  begin
43   if rising_edge (clock) THEN
44
45      case count is
46          when "000000000"   => amp_cs  <= '1';
47                                sdi     <= '0';
48                                busy    <= '0';
49                                sck_out <= '0';
50          when "000000001"   => busy <= '1';
51          when "000001100"   => sdi     <= '0';    --10
52                                amp_cs  <= '0';
53          when "000010100"   => sdi     <= gain(7);--20
54          when "000011001"   => sck_out <= '1';    --25
55                                SPI_read_HighNibble(3)<= amp_do;
56          when "000110010"   => sck_out <= '0';    --50
57          when "000111111"   => sdi     <= gain(6);--63
58          when "001001011"   => sck_out <= '1';    --75
59                                SPI_read_HighNibble(2)<= amp_do;
60          when "001100100"   => sck_out <= '0';    --100
61          when "001110001"   => sdi     <= gain(5);--113
```

Page 1

setgain.vhd Wed Jun 18 16:23:30 2008

```
62                when "001111101"   => sck_out <= '1';    --125
63                                      SPI_read_HighNibble(1)<= amp_do;
64                when "010010110"   => sck_out <= '0';    --150
65                when "010101101"   => sdi     <= gain(4);--163
66                when "010101111"   => sck_out <= '1';    --175
67                                      SPI_read_HighNibble(0)<= amp_do;
68                when "011001000"   => sck_out <= '0';    --200
69                when "011010101"   => sdi     <= gain(3);--213
70                when "011100001"   => sck_out <= '1';    --225
71                                      SPI_read_LowNibble(3)<= amp_do;
72                when "011111010"   => sck_out <= '0';    --250
73                when "100000111"   => sdi     <= gain(2);--263
74                when "100010011"   => sck_out <= '1';    --275
75                                      SPI_read_LowNibble(2)<= amp_do;
76                when "100101100"   => sck_out <= '0';    --300
77                when "100111001"   => sdi     <= gain(1);--313
78                when "101000011"   => sck_out <= '1';    --325
79                                      SPI_read_LowNibble(1)<= amp_do;
80                when "101011110"   => sck_out <= '0';    --350
81                when "101101011"   => sdi     <= gain(0);--363
82                when "101110111"   => sck_out <= '1';    --375
83                                      SPI_read_LowNibble(0)<= amp_do;
84                when "110010000"   => sck_out <= '0';    --400
85                when "110010111"   => sdi     <= '0';    --413
86                when "111000010"   => amp_cs  <= '1';    --450
87                when "111000011"   => busy    <= '0';    --451
88                when others        =>
89             end case;
90          end if;
91       end process;
92       end Behavioral;
```

count_set_gain.vhd Wed Jun 18 16:23:37 2008

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Count_Set_Gain
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity count_set_gain is
12
13     PORT (
14           clock          : in  STD_LOGIC;
15           start_button   : in  STD_LOGIC;
16           button_pressed : in STD_LOGIC;
17           count_set_gain_i : inout STD_LOGIC_VECTOR ( 8 downto 0) := "000000000"
18          );
19
20  end count_set_gain;
21
22  architecture Behavioral of count_set_gain is
23
24
25  begin
26
27  process (clock)
28  begin
29
30
31  if rising_edge (clock)THEN
32
33   if start_button = '0' AND count_set_gain_i = "000000000" AND button_pressed = '0' THEN
34     count_set_gain_i <= "000000000" ;
35   end if;
36   if start_button = '1' AND count_set_gain_i = "000000000" THEN
37     count_set_gain_i <= "000000001" ;
38   end if;
39   if count_set_gain_i < "111000010" AND button_pressed = '1' THEN
40     count_set_gain_i <= count_set_gain_i + 1;
41   end if;
42   if count_set_gain_i = "111000010"  AND start_button ='0' then
43     count_set_gain_i <= "111000011";
44   end if;
45   if count_set_gain_i = "111000011"  AND button_pressed = '0' AND start_button ='0' THEN
46     count_set_gain_i <= count_set_gain_i + 1;
47   end if;
48   if count_set_gain_i > "111000011"  THEN
49     count_set_gain_i <= "000000000";
50   end if;
51  end if;
52
53
54  end process;
55
56  end Behavioral;
57
```

Page 1

set_gain_z_lcd.vhd **Wed Jun 18 16:23:45 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Set_Gain_Z_LCD
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11
12  entity set_gain_z_lcd is
13    PORT (
14          z1                  : out STD_LOGIC_VECTOR (3 downto 0);
15          z2                  : out STD_LOGIC_VECTOR (3 downto 0);
16          z3                  : out STD_LOGIC_VECTOR (3 downto 0);
17          r1                  : out STD_LOGIC_VECTOR (3 downto 0);
18          r2                  : out STD_LOGIC_VECTOR (3 downto 0);
19          r3                  : out STD_LOGIC_VECTOR (3 downto 0);
20          SPI_read_LowNibble  : in  STD_LOGIC_VECTOR (3 downto 0);
21          SPI_read_HighNibble : in  STD_LOGIC_VECTOR (3 downto 0);
22          clk                 : in  STD_LOGIC
23          );
24  end set_gain_z_lcd;
25
26  architecture Behavioral of set_gain_z_lcd is
27
28  begin
29  Process (clk)
30  begin
31   if rising_edge (clk) THEN
32        case SPI_read_LowNibble is
33           WHEN "0001" => z1 <= "0000";
34                          z2 <= "0000";      -- GainA: 001
35                          z3 <= "0001";
36           WHEN "0010" => z1 <= "0000";
37                          z2 <= "0000";      -- GainA: 002
38                          z3 <= "0010";
39           WHEN "0011" => z1 <= "0000";
40                          z2 <= "0000";      -- GainA: 005
41                          z3 <= "0101";
42           WHEN "0100" => z1 <= "0000";
43                          z2 <= "0001";      -- GainA: 010
44                          z3 <= "0000";
45           WHEN "0101" => z1 <= "0000";
46                          z2 <= "0010";      -- GainA: 020
47                          z3 <= "0000";
48           WHEN "0110" => z1 <= "0000";
49                          z2 <= "0101";      -- GainA: 050
50                          z3 <= "0000";
51           WHEN "0111" => z1 <= "0001";
52                          z2 <= "0000";      -- GainA: 100
53                          z3 <= "0000";
54           WHEN OTHERS => z1 <= "1111";
55                          z2 <= "1111";      -- Gain: ???
56                          z3 <= "1111";
57        end case;
58
59        case SPI_read_HighNibble is
60           WHEN "0001" => r1 <= "0000";
61                          r2 <= "0000";      -- GainB: 001
```

Page 1

set_gain_z_lcd.vhd Wed Jun 18 16:23:45 2008

```
62                                  r3 <= "0001";
63                  WHEN "0010" => r1 <= "0000";
64                                  r2 <= "0000";       -- GainB: 002
65                                  r3 <= "0010";
66                  WHEN "0011" => r1 <= "0000";
67                                  r2 <= "0000";       -- GainB: 005
68                                  r3 <= "0101";
69                  WHEN "0100" => r1 <= "0000";
70                                  r2 <= "0001";       -- GainB: 010
71                                  r3 <= "0000";
72                  WHEN "0101" => r1 <= "0000";
73                                  r2 <= "0010";       -- GainB: 020
74                                  r3 <= "0000";
75                  WHEN "0110" => r1 <= "0000";
76                                  r2 <= "0101";       -- GainB: 050
77                                  r3 <= "0000";
78                  WHEN "0111" => r1 <= "0001";
79                                  r2 <= "0000";       -- GainB: 100
80                                  r3 <= "0000";
81                  WHEN OTHERS => r1 <= "1111";
82                                  r2 <= "1111";       -- GainB: ???
83                                  r3 <= "1111";
84              end case;
85          end if ;
86        end process;
87        end Behavioral;
88
89
```

Page 2

counter_lcd.vhd **Wed Jun 18 16:23:52 2008**

```
1   ----------------------------------
2   --Projekt: Verstärkung anzeigen
3   --Instanz: Counter_LCD
4   ----------------------------------
5   library IEEE;
6   use IEEE.STD_LOGIC_1164.ALL;
7   use IEEE.STD_LOGIC_ARITH.ALL;
8   use IEEE.STD_LOGIC_UNSIGNED.ALL;
9
10
11  entity counter_lcd is
12   PORT (
13          clock  : in STD_LOGIC;
14          count  : inout STD_LOGIC_VECTOR (12 downto 0)
15        );
16  end counter_lcd;
17
18  architecture Behavioral of counter_lcd is
19
20  begin
21  process (clock)
22  begin
23    if rising_edge (clock) THEN
24        count <= count + '1';
25    end if;
26  end process;
27  end Behavioral;
28
29
```

divider_lcd.vhd **Wed Jun 18 16:23:59 2008**

```
 1
 2        ----------------------------------
 3        --Projekt: Verstärkung anzeigen
 4        --Instanz: Divider_LCD
 5        ----------------------------------
 6        library IEEE;
 7        use IEEE.STD_LOGIC_1164.ALL;
 8        use IEEE.STD_LOGIC_ARITH.ALL;
 9        use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11
12
13        entity divider_lcd is
14            Port (
15                      ext_clk_50MHz       : in      STD_LOGIC;
16                      dev_clock_out       : inout   STD_LOGIC
17                  );
18        end divider_lcd;
19
20
21        architecture Behavioral of divider_lcd is
22        signal divide_counter : integer := 0;
23
24
25        begin
26
27        Process(ext_clk_50MHz)
28                begin
29                  if (rising_edge (ext_clk_50MHz)) then
30                    if (divide_counter /= 1000) then
31                         divide_counter <= divide_counter + 1;
32                    else
33                        divide_counter <= 0;
34                        dev_clock_out <= not (dev_clock_out);
35                    end if;
36                end if;
37        end process;
38        end Behavioral;
39
40
41
```

top.ucf Wed Jun 18 16:24:07 2008

```
1   #-------------------------------------
2   #Projekt: Verstärkung anzeigen
3   #    User Constraint File
4   #-------------------------------------
5   INST "Inst_setgain/amp_cs" INIT = 1;
6   INST "Inst_setgain/sck_out" INIT = 0;
7   NET "amp_cs"         LOC = "n7"   ;
8   NET "amp_do"         LOC = "e18"  ;
9   NET "clk_system"     LOC = "c9"   ;
10  NET "lcd_dat<0>"     LOC = "r15"  ;
11  NET "lcd_dat<1>"     LOC = "r16"  ;
12  NET "lcd_dat<2>"     LOC = "p17"  ;
13  NET "lcd_dat<3>"     LOC = "m15"  ;
14  NET "lcd_e"          LOC = "m18"  ;
15  NET "lcd_rs"         LOC = "l18"  ;
16  NET "lcd_rw"         LOC = "l17"  ;
17  NET "sck_out"        LOC = "u16"  ;
18  NET "sdi"            LOC = "t4"   ;
19  NET "start_button"   LOC = "K17" | PULLDOWN ;
20
```

Page 1

Literatur

[Rei01] Jürgen Reichard, Bernd Schwarz . *VHDL-Synthese: Entwurf digitaler Schaltungen; 2Aufl.* Oldenbourg Wissenschaftsverlag GmbH, München, 2001.

[Tiet03] Ulrich Tietze, Christoph Schenk. *Halbleiter-Schaltungstechnik; 12Aufl.* Springer-Verlag, Berlin, 2003.

[Urb03] Klaus Urbanski, Roland Woitowitz. *Digitaltechnik; 4Aufl.* Springer-Verlag, Berlin, 2003.

[Wann98] Markus Wannemacher. *Das FPGA-Kochbuch; 1Aufl.* International Thomson Publishing GmbH, Bonn, 1998.