TECHNISCHE UNIVERSITÄT ILMENAU

Fakultät für Elektrotechnik und Informationstechnik
Fachgebiet Elektronische Schaltungen und Systeme
Dipl.-Ing. Th. Rommel

Programmierbare Logikbausteine

Studiengang: Ingenieurinformatik
Studiengang: Elektrotechnik und Informationstechnik

Aufgabe 1
Aufgabe 2

- Praktikumsanleitung -

Für Altera's Design Software Quartus II(Version 13.1)
Baustein Cyclone III

Praktikumsort: H 1555

Grundlagen

Für die Praktikas werden eine Universalplatine DE0 der Firma terasIC und spezielle Zusatzplatinee verwendet. Die Hardware ist also vollständig vorgefertigt und ***Änderungen an dieser sind dem Praktikumsverantwortlichen vorbehalten***. Die Schaltung soll in einem Cyclone III Baustein (EP3C16F) realisiert werden.

Achtung! Bitte schalten Sie die Platine nach dem Testen immer wieder aus!

Die Cyclone III Serie

Altera's Cyclone III Serie ist eine Low Cost Serie in 65nm Low-Power Process Technology mit rekonfigurierbaren CMOS SRAM Elementen. Mit bis zu 120.000 Logik Elementen stellt die Cyclone III Familie die Geschwindigkeit, Dichte und Möglichkeiten zur Verfügung, ganze Systeme, einschließlich komplexer Controller in einem einzigen Baustein zu integrieren.
Die Cyclone Bausteine können für die spezifischen Funktionen, die benötigt werden, `on board´ konfiguriert werden.

Merkmale:

- Logikdichte bis 119.088 Logik Elemente
- Speicher bis zu 3.8 Mbits
- Extrem schnelle Multiplizierer für Digital Signal Processing
- Bis zu 4 PLL's
- Unterstützt High-Speed externen Speicher bis zu 400Mbps
- Bis zu 535 nutzbare Pins
- Unterstützt verschiedene I/O-Standards
- usw.

Die genauen Spezifikationen entnehmen Sie bitte der Tabelle 1.

Feature	EP3C5	EP3C10	EP3C16	EP3C25	EP3C40	EP3C55	EP3C80	EP3C120
Logic Elements	5,136	10,320	15,408	24,624	39,600	55,856	81,264	119,088
Memory (Kbits)	414	414	504	594	1,134	2,340	2,745	3,888
Multipliers	23	23	56	66	126	156	244	288
PLLs	2	2	4	4	4	4	4	4
Global ClockNetworks	10	10	20	20	20	20	20	20

Tabelle 1

Cyclone III Bausteinarchitektur Überblick

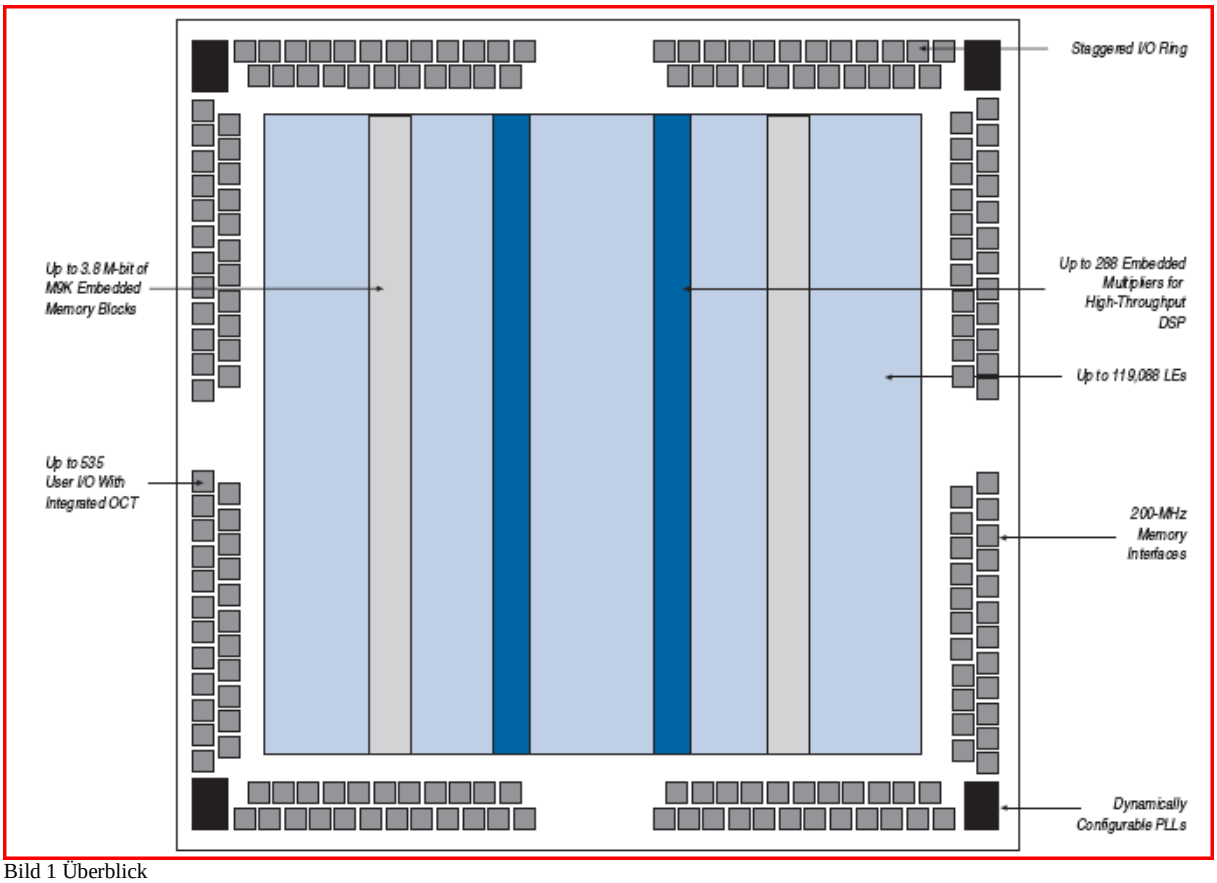


Bild 1 Überblick

Jeder Cyclone III Baustein enthält *LABs (logic array blocks)* für allgemeine Logik, *M9K (Memory Blocks)* um Speicher- und spezielle Logikfunktionen auszuführen und Multiplizierer. In jedem LAB befinden sich 16 Logic Elemente (LE).

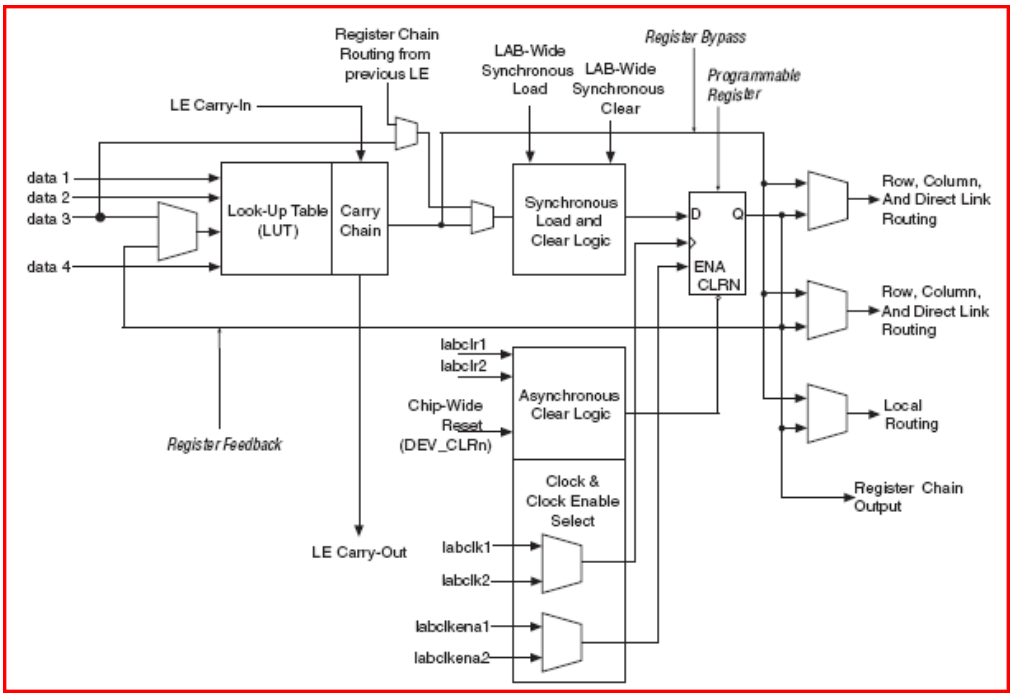


Bild 2: Logic Element

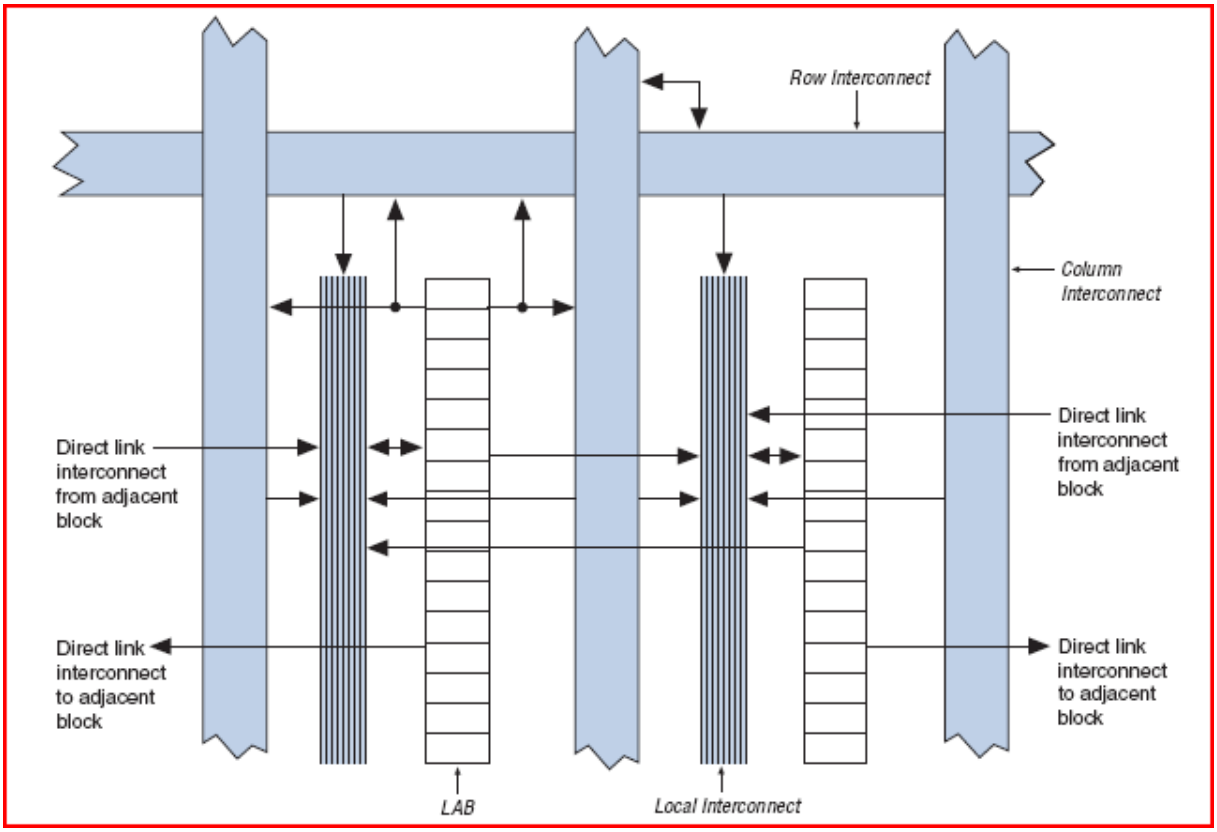


Bild 3: LAB Interconnects

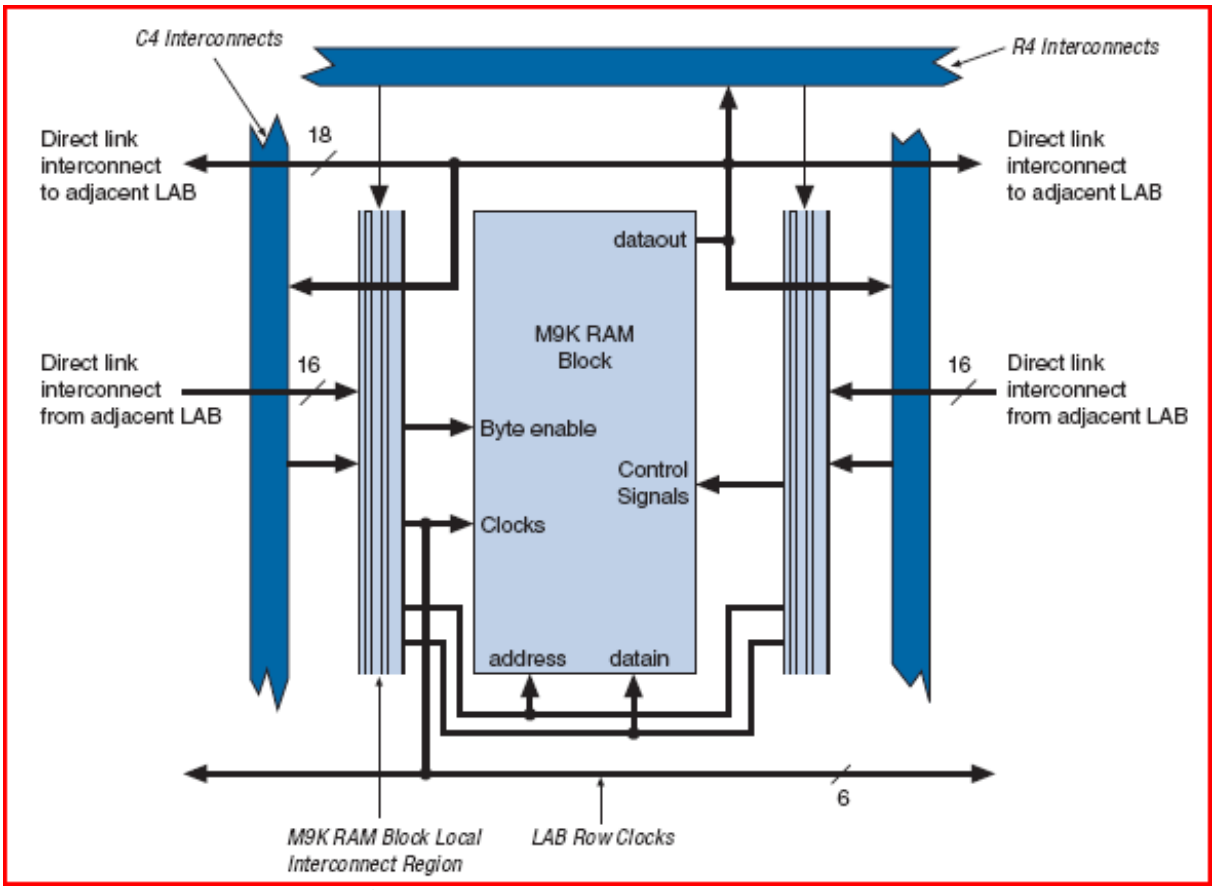


Bild 4: M9K RAM Block LAB Row Interface

Allgemeine Bedienungshinweise

Die Praktikumsrechner sind mit dem Betriebssystem Windows 7 ausgestattet.

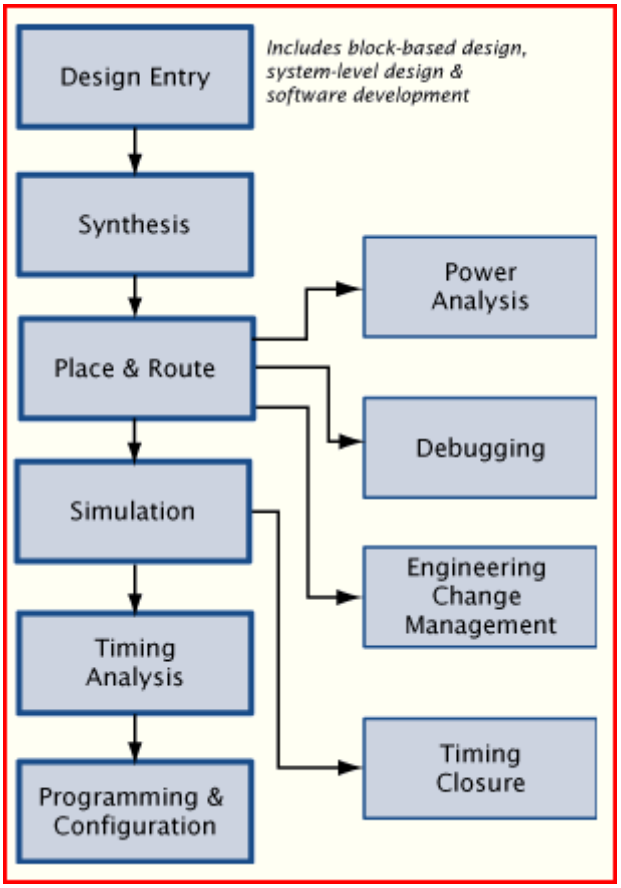
Login Benutzername: `prak_pld`
Kennwort: `prak_pld`

Arbeitsverzeichnis: `C:\PLD-Prak\GruppeXX`
Nur in diesem Verzeichnis werden Dateien gespeichert!

Quartus Einführung

Um die genannte Aufgabe zu lösen, soll hier die Design Software Quartus II (Web Edition) der Firma Altera verwendet werden. Dieses Programm stellt eine architekturunabhängige Entwicklungsumgebung zur Verfügung, die es erlaubt, jedes benötigte Design einzugeben, zu testen und zu programmieren. Ich mache darauf aufmerksam, dass die hier folgende Kurzanleitung nur einen Leitfaden für die Handhabung darstellt.

Design Flow:



Der grundlegende Ablauf zur Erstellung eines neuen Projektes vom Konzept bis zur Fertigstellung kann in einfacher Weise wie folgt dargestellt werden:

- Erzeugen von einem oder mehreren „Design Files“ mit Hilfe der Quartus II Editoren.
- Erstellen eines Top-Level Design Files mit dem Projektnamen
- Auswahl und Zuweisung der gewünschten Logikfamilie bzw. des gewählten Bausteins für das Projekt
- Compilieren des Projektes mit dem Quartus II Compiler
- Durchführung einer Simulation und Timing Analysis
- Programmieren der angeschlossenen Bausteine mit dem erstellten Projekt

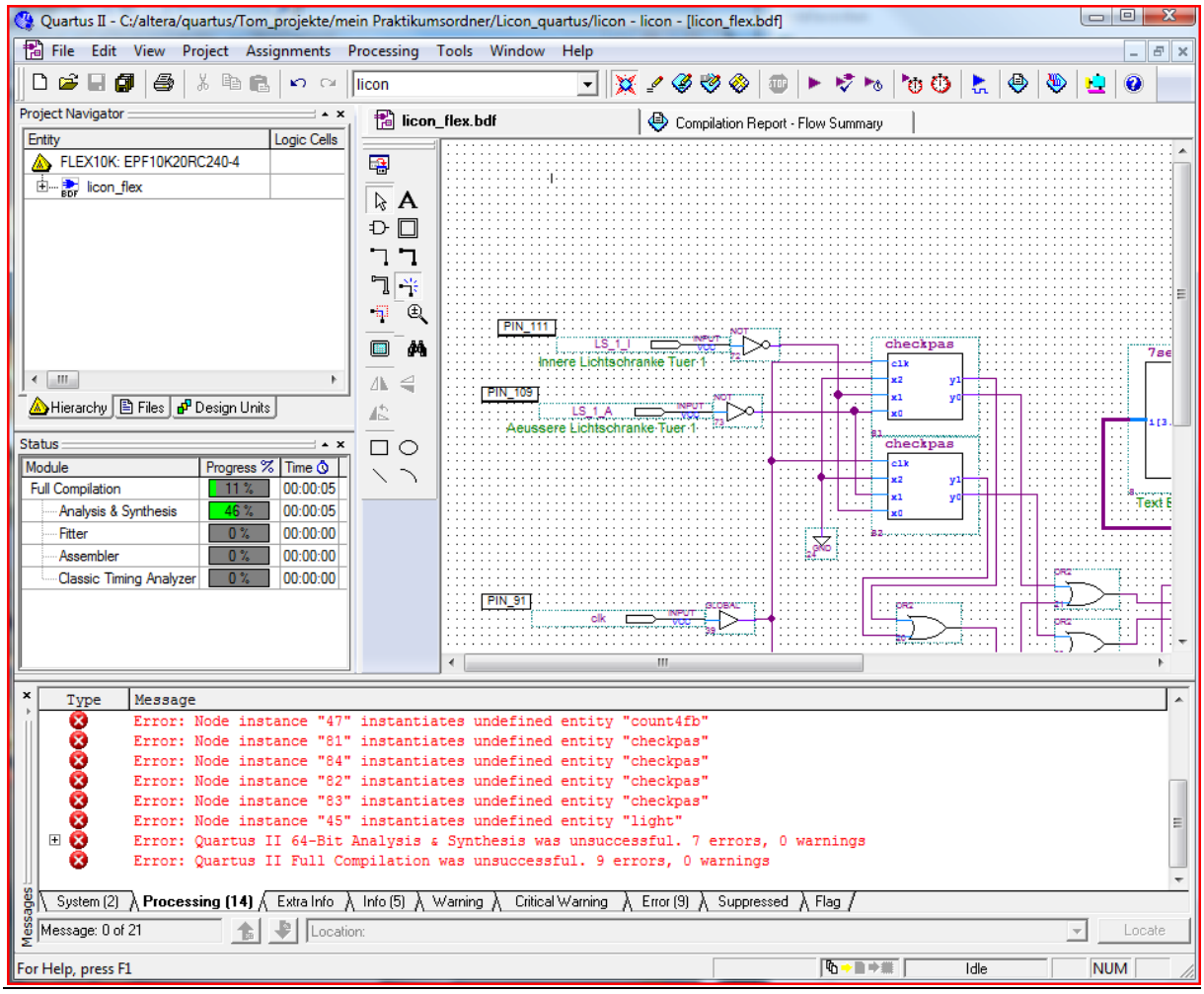
Starten des Programms

Starten Sie ***Quartus II (Web Edition)*** aus dem Menü Programme oder vom Desktop.

Grundlagen

Alle Tools zum Erstellen eines Logikdesigns sind im Programm enthalten. Quartus II enthält vielfältige Möglichkeiten um ein neues Design File einzugeben, z.B. als Block Diagram/Schematic File (Graphische Eingabe), als AHDL-, VHDL,- oder Verilog File (Text Editor), als Tcl Script File oder auch als State Machine File (Graphische Eingabe von Zustandsautomaten). Nach der Eingabe gibt es alle notwendigen und hilfreichen Tools um das Design zu Compilieren, zu Simulieren, zu Analysieren und auf die entsprechende Hardware zu programmieren.

Screenshot Quartus II:



Hier ist eine fürs Praktikum sinnvolle Fenstereinteilung angezeigt:
Arbeitsfläche: Eingabe und Ausgabe von Schematics, State Machines, Simulation usw.
Project Navigator: erlaubt eine schnelle Navigation durch das hierarchische Design
Status: zeigt Fortschritt des Designs
Message: Alle notwendigen Informationen, Fehler, Warnungen und Hinweise für das Design

Project erstellen bzw. auswählen

Um ein neues Project zu erstellen nutzt man für gewöhnlich den „`New Project Wizard`“, in dem alle notwendigen Einstellungen (Bibliotheken, zusätzliche Softwaretools, Devices usw.) vorgenommen werden. Für Praktikumsaufgabe 2 ist vom Praktikumsverantwortlichen dieses Project schon angelegt worden, das sie nur noch mit „`Open Project`“ öffnen müssen. Da sie bei Aufgabe 1 einen kompletten Entwurf durchziehen sollen, müssen Sie hier alle notwendigen Projekteinstellungen vornehmen.

Design Entry

Erstellen eines Schematic Files

1. Wählen Sie „`New`“ (File menu)
2. Markieren Sie „Block Diagram/Schematic File“
3. Wählen Sie „`Save As`“ (File menu)
4. Geben Sie den Pfad und den Namen für ihr File ein (Wenn der Schematic Filename mit dem Blockname im übergeordneten Level übereinstimmt, geschieht eine automatische Zuordnung)

Jetzt können Sie beginnen, ihre Schaltung einzugeben. In Quartus II haben Sie eine Vielzahl von Symbolen für verschiedene Logikfunktionen zur Verfügung, die Sie im Graphic Editor verwenden können. Diese beinhalten sowohl sogenannte „*Primitives*“ das sind Basisfunktionen wie Buffer, Flipflops, Latches, I/O, usw., wie auch „*Megafunctions*“, welche komplexe Funktionen darstellen und zusammen mit *Primitives* oder anderen *Megafunctions* benutzt werden können um Logic Designs zu erstellen.

Eingabe eines „Logic Functions Symbol“ (Logikgatter):

1. Doppelklicken Sie auf eine freie Stelle im Fenster, was gleichzeitig den Einfügepunkt festlegt und das Fenster zur Auswahl des Symbols öffnet.
2. Wählen Sie das benötigte Symbol aus der entsprechenden Bibliothek aus (für die Aufgabe 1 benötigen Sie nur die Bibliothek „primitives“). Hier können Sie alle benötigen Flip-Flops, Gatter und I/O-Pins (Jedes Subdesign benötigt Input und Output Pins, damit eine Anbindung in einer höheren Hierarchieebene erfolgen kann.)
3. Drücken Sie „`OK`“.
4. Wiederholen Sie die Schritte 1 bis 3, um alle benötigten Elemente einzufügen.

In der linken unteren Ecke jedes Symbols befindet sich eine Identifikationsnummer. Diese wird automatisch in der Reihenfolge der Symbolaufrufe vergeben. Diese Nummer hat nichts mit der Funktion der Elemente zu tun, sondern dient lediglich der Identifikation.

Danach verdrahten sie die Symbole und Pins.

Speichern des Files im Filemenu unter „Save as“.

Wenn alles fehlerfrei funktioniert, müssen Sie nun noch ein „Default Symbol“ anlegen, welches Ihr aktuelles File repräsentiert. Dieses Symbol kann dann in allen anderen Graphic Design Files

verwendet und kombiniert werden. Wählen Sie dazu „`Create/Update – Create Symbol Files for current File`“ (File menu). Jetzt haben Sie ein Symbol File mit dem Namen <filename>.bsf angelegt. Dieses kann dann in einer anderen Hierarchistufe (z.B. im Top-Level File) wiederverwendet werden. Achten Sie darauf, dass die entworfenen Subdesigns, den gleichen Namen wie im Top-Level Design haben.

Erstellen eines Text Design Files

1. Wählen Sie „`New`“ (File menu). Markieren Sie „`AHDL File`“ (`VHDL File`)und drücken Sie dann „`OK`“. Jetzt wird automatisch ein File angelegt, dass <Filename>.tdf (vhd) heißt und der Compiler weiß jetzt, dass es sich um ein AHDL-File (VHDL-File) handelt.
2. Um ein AHDL(VHDL)-Textfile zu schreiben, können Sie Templates nutzen, die Ihnen unter dem Symbol „  Insert Templates“ neben vielen anderen nützlichen Hilfen für die Hardwarebeschreibungssprache zu Verfügung stehen. Dort wird Ihnen der genaue Syntax vorgegeben und somit die Fehlerquote drastisch gesenkt.

Wenn Sie alle benötigten Symbole erstellt haben, müssen diese noch zusammengebracht und verbunden werden. Nun müssen Sie noch entsprechende Inputs und Outputs anschließen. Dies geschieht im Top-Level Graphic Design File. Danach wird die Schaltung compiliert und dabei auf Fehler getestet und anschließend simuliert. Sie können auch jedes Symbol einzeln compilieren und simulieren. Das vereinfacht eine mögliche Fehlersuche und erhöht die Wahrscheinlichkeit, dass die gesamte Schaltung auf Anhieb funktioniert, erfordert allerdings einen erhöhten Aufwand.

Compilieren und Fehlersuche

Jeder einzelne Block, den Sie eingegeben haben, sollte separat compiliert und simuliert werden. Dazu müssen Sie dem Compiler mitteilen, welcher Block compiliert werden soll. Das geschieht, in dem Sie im *„Project Navigator“* Fenster den entsprechenden Block mit der rechten Maus anklicken und den Punkt *„Set as Top-Level Entity“* anwählen. Jetzt beziehen sich alle Tools (z.B. auch Simulation, die sie anwählen nur auf diesen Block!!!!

1. Unter „Settings“ (Assingment Menue) können alle Einstellungen, wie Devicename, Pinassingments, usw. eingestellt werden.
2. Wählen Sie „`Start Compilation`“ (Processing menu).

Wenn der Compiling Vorgang erfolgreich beendet ist, kommt ein entsprechender Hinweis. Ansonsten erscheinen alle Fehler und Warnungen im Messagefenster. Vorhandene Fehler finden Sie, indem Sie einen Doppelklick auf dem Fehler machen.

Im Compilation Report werden alle Informationen aufbereitet.

Nachdem ihre Schaltung fehlerfrei compiliert wurde, ist es ratsam, eine Simulation durchzuführen, um zu gewährleisten, dass die Schaltung die entsprechenden Timing- und Funktionbedingungen erfüllt.

Simulation (schnelle einfache graphische Simulation)

1. Erstellen Sie über „`New`“ (File menu) ein „`University Program VWF`“!
2. Über rechte Mausklick im Feld „`Name`“ des Waveform1.vwf können Sie „`Insert Node or Bus`“ auswählen und danach über „Node Finder..“ alle notwendigen Pins, Signalnamen oder Nodes aufLISTen lassen. Übernehmen Sie die für Sie interessanten gefunden Nodes in die „`Selected Nodes`“-Liste!
3. Zum Bearbeiten der Signalverläufe, markieren Sie das Signal oder auch nur bestimmte Intervalle, die sie editieren wollen. Mit Hilfe der in der linken Leiste des Waveformfensters befindlichen Buttons können Sie den markierten Intervallen des Signals den entsprechenden Wert zuweisen.
4. Speichern Sie das VWF-File mit „`Save`“ (File menu).
5. „`Run Functional Simulation`“ oder „`Run Timing Simulation`“ (Simulation Menu).

Wenn Fehler auftreten bzw. Signale sich nicht wie erwartet verhalten, müssen Sie den Fehler suchen, das notwendige File ändern, compilieren und wieder alle Schritte bis hierher durchgehen.

Simulation (komplexe textbasierende Simulation)

Wollen sie jedoch eine umfangreiche Simulation (z.B. auch mit Testbench) durchführen, nutzen Sie dazu ModelSim.

Programmierung eines Bausteins

Die Programmierung wird vom Betreuer durchgeführt!

Aufgabe 1a

Problemstellung

Es existiert ein Raum mit zwei Türen in das kein Licht von außen einfallen kann. Die Innenraumbeleuchtung soll automatisch eingeschaltet werden, wenn sich jemand im Raum aufhält. Da in dem Raum oft bewegungsarme Arbeiten ausgeführt werden, fällt eine Lösung mit Infrarotsensor aus. Ein 7-Segment Display soll außerdem stets anzeigen, wie viele Personen sich im Raum befinden. Zusätzlich dazu soll ein Warnsignal ertönen, wenn sich mehr als 4 Leute im Raum aufhalten.

Lösung

Um das obige Problem zu lösen, sind an beiden Türen jeweils zwei Lichtschranken installiert (Bild 1). Diese sollen alle Personen, die den Raum betreten oder verlassen, registrieren. Eine Lichtschranke befindet sich an der Innenseite der Tür und eine zweite an der Außenseite. Somit kann genau festgestellt werden, ob eine Person den Raum betritt oder verlässt, bzw. ob sie eventuell zwischendurch die Richtung wechselt. Damit eine Person als ein- bzw. austretend erkannt wird, muss sie beide Lichtschranken durchlaufen, so dass ein Signalverlauf, wie in Bild 2 gezeigt, erfolgt.

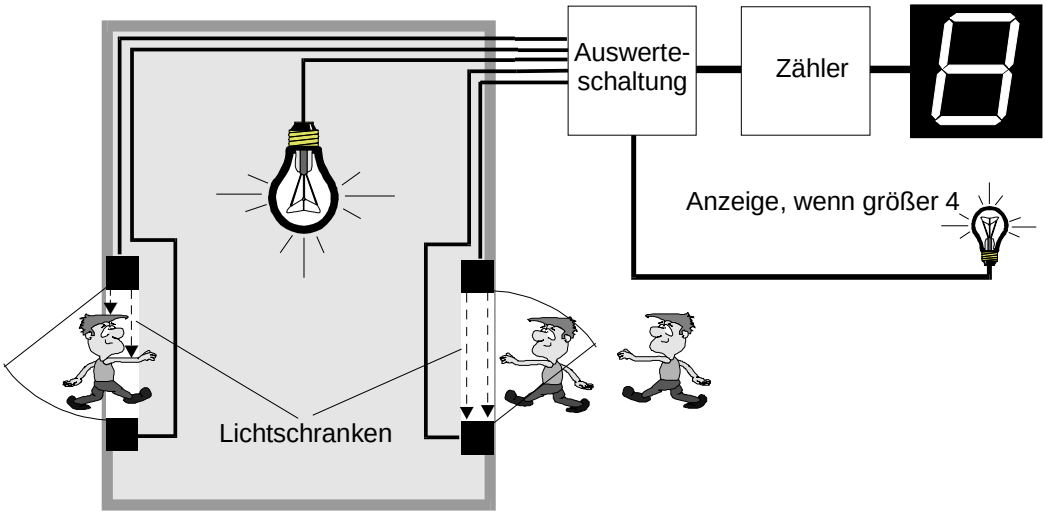


Bild 1

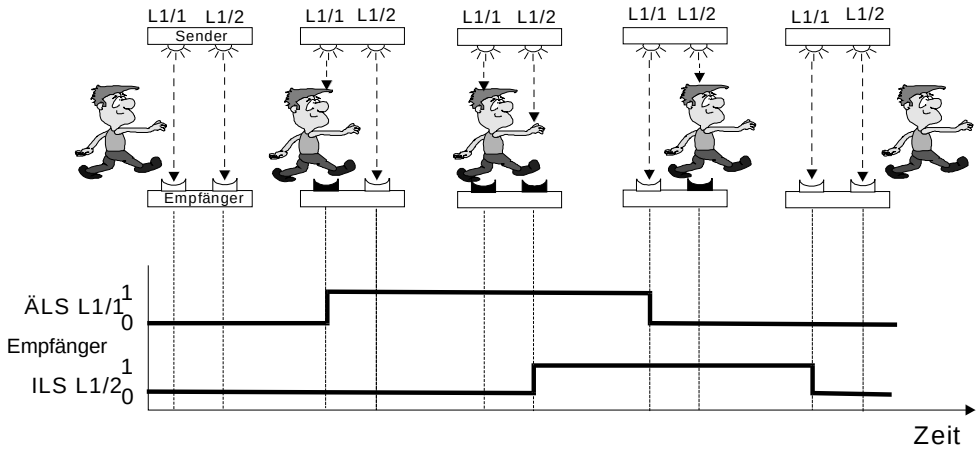


Bild 2

Projekteinstellungen:

Wenn Sie über den Menüpunkt: `File/New Project Wizard` ihr neues Projekt erstellen benutzen Sie bitte folgende Einstellungen:

working directory: `C:\PLD-Prak\GruppeXX\licon`
name of project: `licon`
Device family: `Cyclone III`
Package: `FBGA`
Pin Count: `484`
Available devices: `EP3C16F484C6`

Allen anderen Einstellungen bleiben bitte auf `default`.

Pinbelegung:

Pinbezeichnung		***Pinbelegung***
clk	(Systemtakt, 50 MHz)	PIN_G21
LS_1_A	(Tür 1, äußere Lichtschranke)	PIN_AB18
LS_1_I	(Tür 1, innere Lichtschranke)	PIN_AA18
LS_2_A	(Tür 2, äußere Lichtschranke)	PIN_AA19
LS_2_I	(Tür 2, innere Lichtschranke)	PIN_AB19
Full	(Mehr als 4 Leute im Raum)	PIN_AB20
Light	(Lichtansteuerung)	PIN_AA20
Q[3..0]	(Ausgänge vom Zähler)	PIN_H1 PIN_J3 PIN_J2 PIN_J1

Alle anderen Pinbelegungen, die Sie für das Praktikum noch benötigen (z.B. für die 7 Segment-Anzeige) entnehmen Sie bitte der Boardbeschreibung DE0_User_manual.pdf, die sich auf dem Desktop ihres Praktikumsrechner befindet.

Hinweise für den Entwurf:
Teilen Sie den Entwurf in sinnvolle Blöcke auf und nutzen Sie den hierarchischen Entwurf.
Die Schaltung wird in jedem Fall folgende Komponenten enthalten müssen:

- Zähler (graphischer Entwurf zwingend erforderlich),
- Komparator,
- Zustandsautomaten,
- 7-Segment Decoder.

Meine Empfehlung für den Toplevel wäre eine graphische Lösung. Sie dient der besseren Übersichtlichkeit. Wenn Sie aber des hierachischen Entwurfs mit VHDL mächtig sind, geht selbstverständlich auch das. Die Pinzuweisung ist erst erforderlich, wenn Sie die Gesamtschaltung entworfen haben. Nutzen Sie dazu den `Pin Planner` unter dem Menüpunkt `Assignments`.

Aufgabe 1b

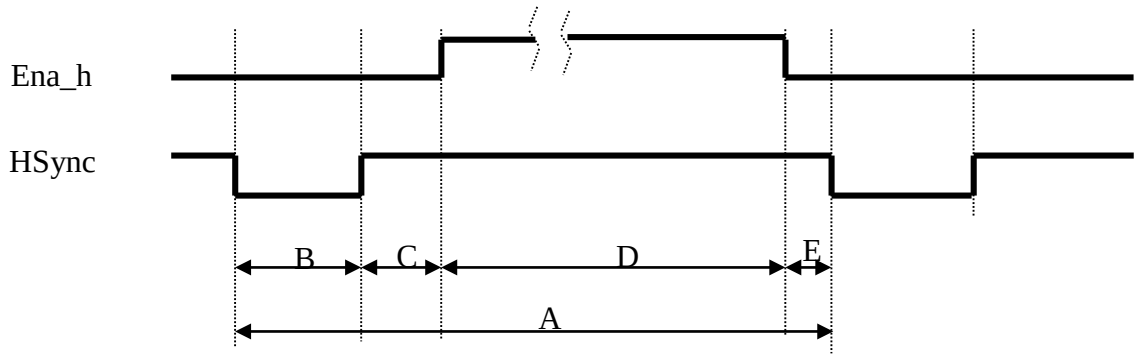
Erweitern Sie die Schaltung so, dass der Zähler nicht nur bis 9 sondern bis 99 (dezimal) zählt.

Aufgabe 2

Einleitung:

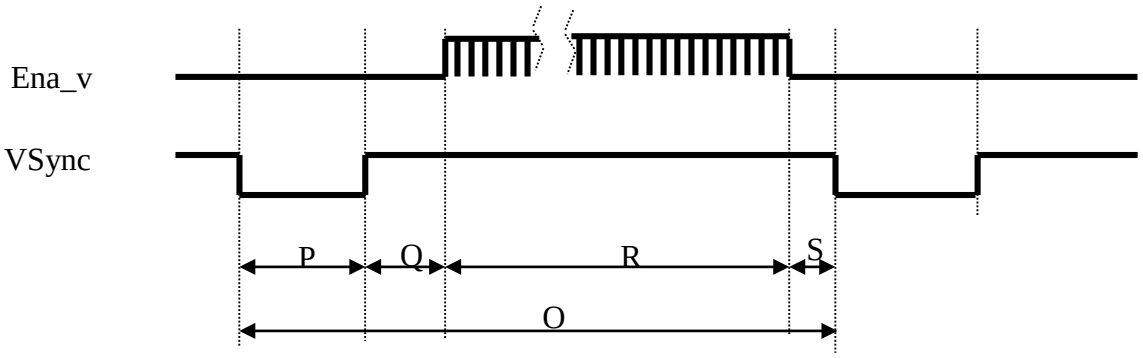
In diesem Praktikum soll eine bestehende VGA-Ansteuerung (640 x 480) zu einem Art Videospiel erweitert werden. Auf dem vorhandenen DEO-Board ist ein Cyclone III FPGA der Firma Altera und die nötigen Tasten, sowie die notwendige Hardwareverdrahtung, um einen VGA-Bildschirm anzusteuern, implementiert.
Die notwendige VGA-Schaltung ist in der Schaltung VGA.bdf (siehe Anhang) bereits entworfen und soll dementsprechend modifiziert und weiterentwickelt werden.
Jeder VGA Monitor hat einen internen Takt, der bestimmt, wann jedes Pixel neu geschrieben wird. Dieser Takt arbeitet bei der VGA-spezifischen Frequenz von 25,0 MHz. Der Monitor erneuert den Bildschirm (Refresh) in einer bestimmten Art und Weise, die gesteuert wird durch ein horizontales und vertikales Synchronisationssignal. Der Monitor beginnt jeden Refreshzyklus mit einem Update der Pixel in der linken oberen Ecke (0,0) des Bildschirmes. Nachdem das erste Pixel einen Refresh erhält, erzeugt der Monitor für die verbleibenden Pixel in der Reihe einen Refresh. Wenn der Monitor einen Impuls für die horizontale Synchronisation erhält, erzeugt er die nötigen Refreshsignale für die nächste Reihe. Dieser Prozess wird wiederholt bis der Monitor die unterste Zeile des Schirmes erreicht. Mit dem enstprechenden verticalen Synchronisationssignal springt er wieder auf die erste Zeile (0,0) des Schirmes.
Damit der Monitor richtig arbeitet, muss er seine Impulse zu bestimmten Zeiten empfangen. Horizontale und vertikale Synchronisations-Impulse müssen zu einer bestimmten Zeit anliegen, damit die Farbsignale (Red, Green, Blue) entsprechend über die Enable-Signale freigegeben werden können.
Bild 1-3 zeigen die Waveform der entsprechenden Farbinformation in Abhängigkeit von den horizontalen und vertikalen Synchronisationssignalen.

Diese Signale stellt der Block counter_h_v zur Verfügung.



Parameter	***A=H_TOTAL***	***B=H_SYNC***	***C=H_BACK***	***D=H_ACT***	***E=H_FRONT***
Time	***32,0 µs***	***3,84 µs***	***1,92 µs***	***25,6 µs***	***0,64 µs***
Anzahl der Takte(T_{pixel})	***800***	***96***	***48***	***640***	***16***
	Line				

Bild 1



Parameter	*O=V_TO-TAL*	*P=V_SYNC*	*Q=V_BACK*	*R=V_ACT*	*S=V_FRONT*
Time	***16,8 ms***	***64 µs***	***1,056ms***	***15,36ms***	***0,32 ms***
Anzahl der Takte (Horizontal Refresh Cycles)	***525***	***2***	***33***	***480***	***10***

Bild 2

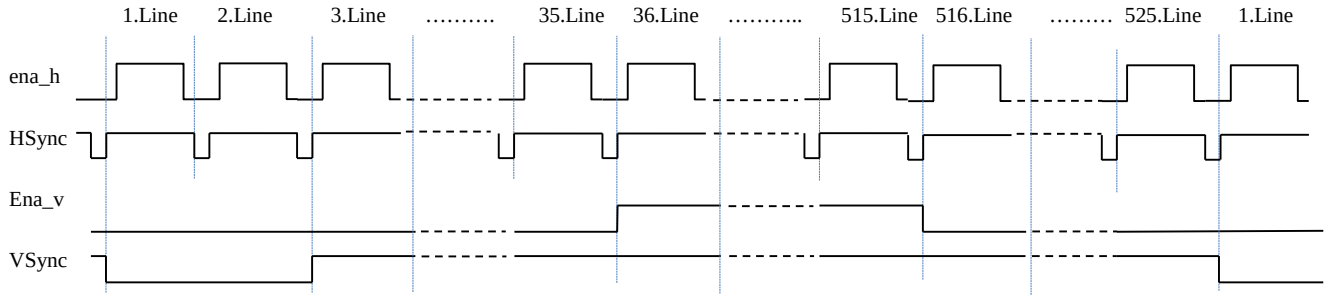


Bild 3

Bild 4 zeigt den Toplevel der VGA-Ansteuerung in etwas komprimierter Form. Die Schaltung enthält folgende Blöcke:

- *VGA-Control*
  Dieser Block stellt alle notwendigen Synchronisations- und Enable-Signale für die VGA-Ansteuerung zu Verfügung. (darf nicht verändert werden!)
- *Graphik-Huhn*
  beinhaltet 2 ROMs + die notwendige Ansteuerung der ROMs, die auf dem Bildschirm ein flatterndes Huhn darstellt.
- *Hintergrundbild*
  ROM + notwendige Ansteuerung für Hintergrundbild in der Größe 640x128
- *Position von Backround*
  sorgt dafür das Hintergrundbild den unteren Bereich des Bildschirmes belegt
- *Clk-Divider*
  stellt verschiedene geteilte Frequenzen zu Verfügung(z.B. für die Flatterfrequenz des Huhnes)

- *Output-Multiplexer*
  dort lassen sich verschiedene Graphiken, jeweils nach Farbe getrennt, zusammen schalten
- *Power-on-Reset*
  sorgt für definierte Signale nach dem Einschalten

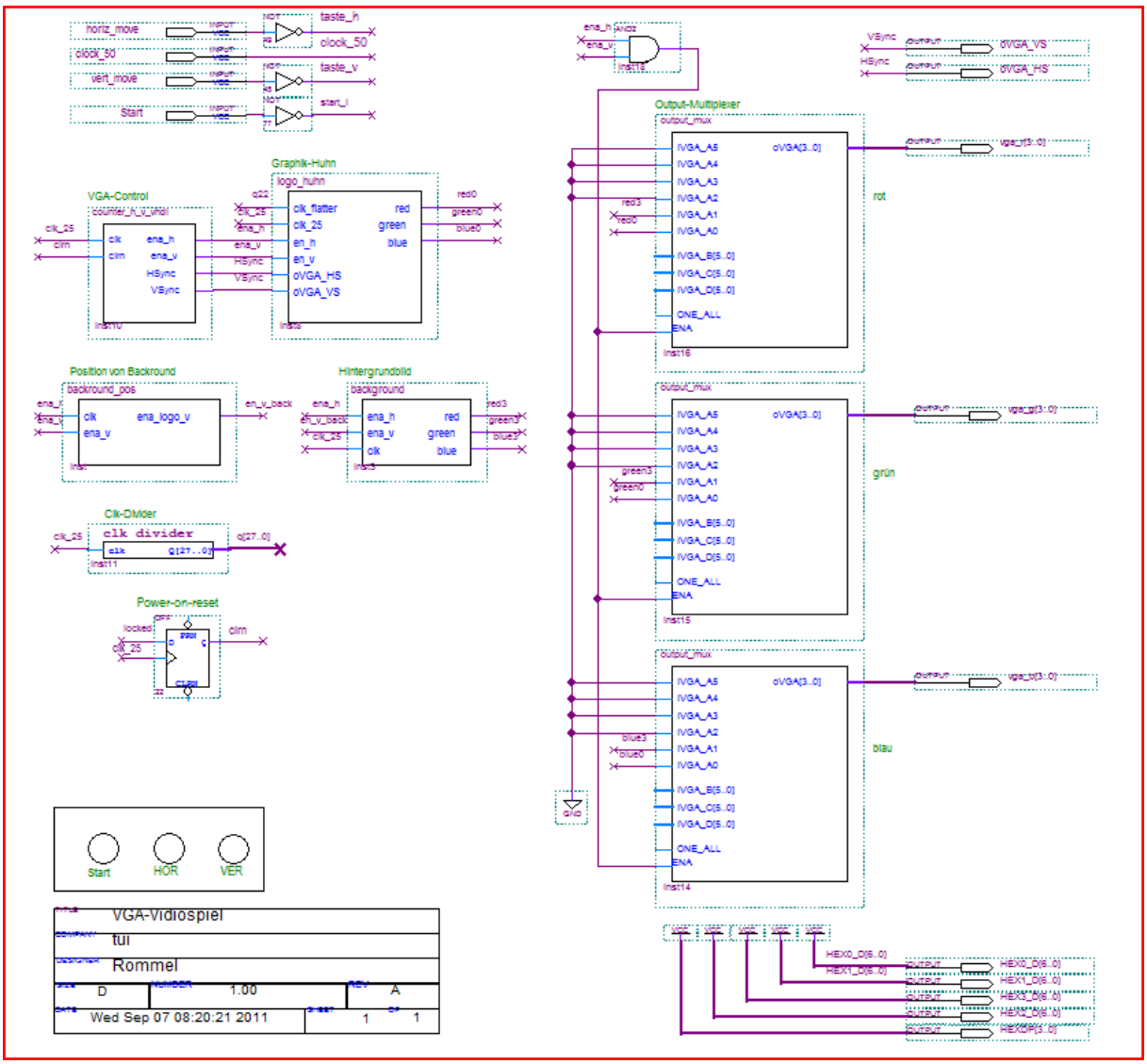


Bild 4

Teilaufgabe I:

Machen sie sich mit der bestehenden Schaltung soweit vertraut, dass sie den schaltungstechnischen Hintergrund verstehen. Ein Verständnis der Schaltung ist notwendig, da sonst keine Modifizierung erfolgen kann.

Teilaufgabe II:

Die Pixel-Frequenz für die VGA-Ansteuerung beträgt 25 MHz. Auf dem Board ist aber ein Taktgenerator, der eine 50 MHz-Frequenz bereitstellt, die am Pin clock_50 angeschlossen ist. Bevor Sie die Schaltung in Betrieb nehmen müssen Sie einen 25 MHz erzeugen. Benutzen Sie

dazu eine der vier PLL-Schaltungen (Modul altpll), die in dem Baustein bereits integriert sind. Nutzen Sie auch den Output „Locked“, da dieser die Power-On-Reset Schaltung bedient.

Teilaufgabe III:

Tauschen sie das Modul *logo_huhn* gegen das Modul *logo_visier_64x32* aus.

Entwickeln Sie die Schaltung VGA.bdf so weiter, das das Visier (auf dem Bildschirm oben links) mit Hilfe der Tasten right/left und up/down auf dem Bildschirm in X- und Y-Richtung positioniert werden kann.

Die Schaltungsblöcke Counter_h_v und Logo_Huhn, dürfen nicht verändert werden, genauso wenig, wie die direkten Ausgangs_Multiplexer, da sonst eine Zerstörung des Bildschirmes möglich ist.

Auf welche Art und Weise (Graphisch, VHDL, AHDL) Sie die Schaltung entwerfen ist Ihnen überlassen!

Im Schematic haben sie die Möglichkeit, sogenannte LPM (library of parameterized modules) zu verwenden. LPM sind vorgefertigte Module (in AHDL geschrieben), die sie über Dialog-fenster modifizieren können und somit ihren Erfordernissen anpassen können. Solche Makros sind für fast alle digitalen Funktionen, wie ADDER, COUNTER, SHIFTREGISTER, RAM, ROM usw. verfügbar

Teilaufgabe IV:

Entwickeln sie die nun vorhandene Schaltung so weiter, dass sie das flatternde Huhn wieder auf dem Bildschirm sehen können.

Teilaufgabe V:

Verändern sie die in Teilaufgabe IV entworfene Schaltung so, dass das flatternde Huhn auf Druck auf Button2 an einer zufälligen Position des Bildschirmes erscheint.

Teilaufgabe VI (optional):

Lassen Sie ihrer Phantasie freien Lauf und entwickeln Sie weitere Module, die ein Spiel spielenswert machen, z.B. Huhn abschießen, Trefferzahlanzeige, unterschiedliche Level und und und.