

VHDL-Workshop

Fotoapparat

Universität Erlangen-Nürnberg
Lehrstuhl für Rechnergestützten Schaltungsentwurf
Prof. Dr.-Ing. Wolfram H. Glauert
Cauerstraße 6, 91058 Erlangen
Tel.: 09131-8523100
Fax: 09131-8523111
Email: vhdl@lrs.e-technik.uni-erlangen.de
W3: <http://www.vhdl-online.de>

Inhalt

- 1. Einleitung**
- 2. IP-Adresse**
- 3. Gliederung**
- 4. Namensgebung**
- 5. Das Design**
- 6. Der Texteditor**
 - 6.1 VHDL-System-Befehle
- 7. Der VHDL Analyzer**
 - 7.1 VHDL-System-Befehle
- 8. Die VHDL-Simulation**
 - 8.1 VHDL-System-Befehle
 - 8.2 Bedienung des Simulators (1)
 - 8.3 Bedienung des Simulators (2)
- 9. Die VHDL-Synthese**
 - 9.1 VHDL-System-Befehle

1. Übung: Ein Multiplexer

- 1.1 ZIEL
- 1.2 BESCHREIBUNG
- 1.3 ERGÄNZUNGEN
- 1.4 ERLÄUTERUNGEN
- 1.5 AUSFÜHRUNGEN
- 1.6 VHDL-System-Befehle

2. Übung: Erweiterung des Multiplexers

- 2.1 ZIEL
- 2.2 BESCHREIBUNG
- 2.3 ERLÄUTERUNGEN
- 2.4 AUSFÜHRUNGEN:
- 2.5 VHDL-System-Befehle

3. Übung: Eine Siebensegment-Anzeige

- 3.1 ZIEL
- 3.2 BESCHREIBUNG
- 3.3 ERLÄUTERUNGEN
- 3.4 AUSFÜHRUNGEN:
- 3.5 VHDL-System-Befehle

4. Übung: Eine dreistellige Anzeige

- 4.1 ZIEL
- 4.2 ERGÄNZUNGEN
- 4.3 BESCHREIBUNG
- 4.4 ERLÄUTERUNGEN
- 4.5 AUSFÜHRUNGEN
- 4.6 VHDL-System-Befehle

5. Übung: Ein Decoder

- 5.1 ZIEL
- 5.2 BESCHREIBUNG
- 5.3 ERLÄUTERUNGEN
- 5.4 AUSFÜHRUNGEN:
- 5.5 VHDL-System-Befehle

6. Übung: Ein Register

- 6.1 ZIEL
- 6.2 BESCHREIBUNG
- 6.3 ERLÄUTERUNGEN
- 6.4 AUSFÜHRUNGEN:
- 6.5 VHDL-System-Befehle

7. Übung: Ein Zähler

- 7.1 ZIEL
- 7.2 BESCHREIBUNG
- 7.3 ERGÄNZUNGEN
- 7.4 ERLÄUTERUNGEN
- 7.5 AUSFÜHRUNGEN:
- 7.6 VHDL-System-Befehle

8. Übung: Ein Timer

- 8.1 ZIEL
- 8.2 BESCHREIBUNG
- 8.3 ERGÄNZUNGEN
- 8.4 ERLÄUTERUNGEN
- 8.5 AUSFÜHRUNGEN:
- 8.6 VHDL-System-Befehle

9. Übung: Zustandsautomat Fotosteuerung

- 9.1 ZIEL
- 9.2 BESCHREIBUNG
- 9.3 ERGÄNZUNGEN
- 9.4 ERLÄUTERUNGEN
- 9.5 AUSFÜHRUNGEN:
- 9.6 VHDL-System-Befehle

10. Übung: Zustandsautomat Display

- 10.1 ZIEL
- 10.2 BESCHREIBUNG
- 10.3 ERLÄUTERUNGEN
- 10.4 AUSFÜHRUNGEN:
- 10.5 VHDL-System-Befehle

11. Übung: Der Fotoapparat

- 11.1 ZIEL
- 11.2 BESCHREIBUNG
- 11.3 ERGÄNZUNGEN
- 11.4 ERLÄUTERUNGEN

- 11.5 AUSFÜHRUNGEN:
- 11.6 VHDL-System-Befehle

12. Übung: Simulation auf Gatterebene

- 12.1 ZIEL
- 12.2 BESCHREIBUNG
- 12.3 ERGÄNZUNGEN
- 12.4 ERLÄUTERUNGEN
- 12.5 AUSFÜHRUNGEN:
- 12.6 VHDL-System-Befehle

13. Übung: Synthese mit Optimierungs- und Arbeitskriterien

- 13.1 ZIEL
- 13.2 BESCHREIBUNG
- 13.3 ERGÄNZUNGEN
- 13.4 ERLÄUTERUNGEN
- 13.5 AUSFÜHRUNGEN:
- 13.6 VHDL-System-Befehle

14. Übung: Optimierung des VHDL Designs

- 14.1 ZIEL
- 14.2 BESCHREIBUNG
- 14.3 ERGÄNZUNGEN
- 14.4 ERLÄUTERUNGEN
- 14.5 AUSFÜHRUNGEN:
- 14.6 VHDL-System-Befehle

15. Übung: Realisierung in einem FPGA

- 15.1 ZIEL
- 15.2 BESCHREIBUNG
- 15.3 ERGÄNZUNGEN
- 15.4 AUSFÜHRUNGEN:
- 15.5 VHDL-System-Befehle

1. Einleitung

Dieses hypertextbasierte VHDL Lernprogramm teilt sich in einzelne Übungsaufgaben auf. Indem Sie typische VHDL Programme schreiben, werden Sie mit der Hardwarebeschreibungssprache vertraut.

Am Anfang werden Sie Schritt für Schritt an Ihre ersten Aufgaben hingeführt und später, wenn Sie mit der Sprache und den Tools vertrauter werden, sollen Sie die Übungen langsam selber lösen.

Wir beziehen uns auf den VHDL '87 Standard, da der VHDL '93 Standard noch nicht von allen Werkzeugherstellern unterstützt wird.

Am Anfang und Ende jeder Hypertextseite finden Sie folgende Icons:

IMGONhome.gifIMGOFF ...zu VHDL-Online

IMGONinhalt.gifIMGOFF ...zu VHDL-Workshop Inhaltsverzeichnis

IMGONzurueck.gifIMGOFF ...Zurück

IMGONvor.gifIMGOFF ...Vorwärts

IMGONhilfe.gifIMGOFF ...zu VHDL-Workshop Hilfe (enthält eine Übersicht der einzelnen Dateien + Lösungen und den Automatengraph für die Fotoapparatsteuerung).

IMGONreset.gifIMGOFF ...zu VHDL-Workshop File-Reset

IMGONmail.gifIMGOFF ...zu VHDL-Workshop E-Mail

2. IP-Adresse

Bevor Sie mit dem Workshop beginnen, müssen Sie noch die IP-Adresse Ihres Rechners und die Nummer des Bildschirms, vor dem Sie sitzen, angeben.

(Die Default Nummer des Bildschirms ist Null):

-- Display --

3. Gliederung

Die einzelnen Übungen gliedern sich in verschiedene Abschnitte:

- **ZIEL:** Hier wird ihnen das gewünschte Ergebnis erklärt und die Aufgabe des Modells innerhalb des Designs.
- **BESCHREIBUNG:** Hier wird Ihnen die Funktion des Designs erklärt.
- **ERGÄNZUNGEN:** In diesem Abschnitt werden Ihnen Probleme, die sich durch VHDL ergeben genauer beschrieben. Die Hilfestellung soll Ihnen zur Lösung des Problems dienen.
- **ERLÄUTERUNGEN:** Hier steht, wie Sie ihre Files, Entity, Architecture benennen sollen. Ebenfalls werden Ihnen die benötigten Datentypen vorgegeben. Gegebenenfalls bekommen Sie auch noch Hinweise bezüglich des Beschreibungsstils.
- **AUSFÜHRUNGEN:** Hier werden Ihre Aufgaben beschrieben.
- **VHDL System Befehle:** Hier stehen die benötigten Werkzeuge bzw. Ihre Files. Wie Sie diese Optionen bedienen wird Ihnen am Ende dieser Einleitung erklärt.

4. Namensgebung

In VHDL ist es außerordentlich wichtig, daß Sie eine einheitliche Namensgebung beibehalten. Beispielsweise arbeiten Sie an einem Design mit dem Namen: **MYDESIGN**

Folgende Namensgebung sollte beachtet werden:

Tabelle 1: Namensgebung

File name:	mydesign.vhdl	für die VHDL Datei
-------------------	----------------------	--------------------

Tabelle 1: Namensgebung

Entity name:	MYDESIGN TB_MYDESIGN	für das Modell Entity-Name der Testbench
Architecture name:	RTL BEH TEST	für RTL-code für Verhaltensbeschreibungen für die Architecture der Testbench
Configuration name:	CFG_TB_MYDESIGN	für die Configuration der Testbench
Typendefinition name:	T_<eigener_name>	vorangestelltes T_ ...

5. Das Design

Im Rahmen dieses Kurses sollen Sie eine Steuerung für einen Fotoapparat entwerfen. Bei diesem Kurs ist es nicht das Ziel ein optimales Ergebnis des Designs zu erhalten, sondern vielmehr die Erlernung der Hardwarebeschreibungssprache VHDL. In den ersten Übungen werden Sie langsam an Ihre Aufgabe hingeführt, später bekommen Sie immer weniger Hilfestellung, bis Sie zum Schluß die Aufgabe selber lösen.

Der Fotoapparat besteht aus sieben Modulen, die Sie entwerfen sollen:

1. **Der Decoder (decoder):** Hier werden die Signale die von der Eingabeeinheit kommen decodiert, um die Belichtungszeit geeignet darzustellen.
2. **Das Belichtungsregister (exposure_latch):** Hier wird die Belichtungszeit gespeichert.
3. **Der Timer für den Transport (timer_transport):** Falls beim Weitertransport des Films ein Fehler auftritt (Filmriß, Filmende), soll dieses Modul einen Fehler melden (maximale Transportzeit 2 Sekunden).
4. **Der Timer für die Belichtungszeit (timer_exposure):** Hier wird die eingegebene Belichtungszeit realisiert.
5. **Die Fotosteuerungen (photo_control):** Dieses Modul übernimmt die gesamte Steuerung des Fotoapparates.
6. **Der Displaycontroller (display_control):** Dies ist der Automat der die Displayanzeige steuert.
7. **Der Displaydriver (display_driver):** Dieses Modul bestimmt die Display-Anzeige.

Der Fotoapparat wird mit 8192 Hz getaktet und hat Belichtungszeiten von $2\text{exp}(-i)$ (für $i = 0..9$). Diese Angabe benötigen Sie später um die verlangten Zeiten zu realisieren. Das Design soll folgende Funktionen erfüllen:

- Belichtungszeit bestimmen
- Signal für einen Motoraufzug liefern
- Aufnehmen von Bildfolgen
- Verschußsignal liefern
- Filmende- und Fehlererkennung beim Filmtransport

6. Der Texteditor

Das Ziel dieser Übung ist es, mit den Simulations- und Synthesetools vertraut zu werden und die grundlegenden Befehle der Werkzeuge von Synopsys kennenzulernen. In dieser Übung werden keine VHDL-Konstrukte benötigt. Der Synopsys VHDL-Simulator unterstützt den IEEE 1076 VHDL Standard, so daß Sie sich keinen speziellen VHDL-Subset aneignen müssen. Das erste „Design“ ist ein UND-Gatter. Das File hat den Namen **und.vhd**. Es beschreibt ein

UND-Gatter mit zwei Eingängen.

Um mit den Werkzeugen vertraut zu werden, bearbeiten Sie die nächsten Punkte.

- Wählen Sie den Hyperlink **editieren** unten bei den VHDL-SYSTEM-Befehlen für das File **und.vhd** an. Es wird der Texteditor mit dem VHDL-Code des UND-Gatters erscheinen.
- **IMGONinfo.gifIMGOFF Wichtig:**
Betätigen Sie die Stop-Taste an Ihrem HTML-Browser, um den Dienst zu unterbrechen. (Der HTML-Daemon wartet immer auf eine Antwort des aufrufenden Prozesses. Die Antwort wird erst nach Beenden des Texteditors erzeugt. Also würde der HTML-Daemon ewig warten, oder in einen Timeout laufen. Sie könnten dann nicht mit dem HTML-Browser weiterarbeiten)
- Nun können Sie ganz normal mit dem Texteditor arbeiten. Zum iconifizieren betätigen Sie den Button oben links am Editorfenster. Wählen Sie den Hyperlink **editieren** für das File **tb_und.vhd** an. (Stop-Taste am HTML-Browser drücken)
- Einige Fragen zur Testbench:
Notieren Sie sich die Namen folgender VHDL-Objekte: Entity, Component
Welche andere Major Design Unit wird in diesem File verwendet (neben Entity und Architecture) ?
Notieren Sie den Typ der Design Unit und den logischen VHDL-Namen.

6.1 VHDL-System-Befehle

7. Der VHDL Analyzer

- Alle zu einem VHDL-Design gehörenden Files müssen in einer bestimmten Reihenfolge analysiert (=übersetzt) werden. Wie ist die Reihenfolge bei diesen beiden Files?
- Betätigen Sie die entsprechenden Hyperlinks in der richtigen Reihenfolge. Das Ergebnis der Analyse wird in einer extra Seite angezeigt. Von dort gelangen Sie mit dem BACK-Button wieder auf diese Seite.
- Korrigieren Sie die ermittelten Fehler mit dem Texteditor.

7.1 VHDL-System-Befehle

8. Die VHDL-Simulation

Nachdem Sie alle Files in der richtigen Reihenfolge analysiert haben, starten Sie den Simulator durch anwählen des Hyperlinks Simulation bei den System-Befehlen. Es erscheint das VHDL-Debugger-Select Fenster

IMGONsimulator.gifIMGOFF

Abbildung 1: VHDL-Debugger-Select-Window

- Der Simulator wird über dieses Fenster immer mit dem Namen der "top level configuration" aufgerufen
--> Die Configuration ist das simulierbare Objekt.
- Wie wird die "top level configuration" für diese "design hierarchy" genannt? Wählen Sie den entsprechenden Namen aus ("doppelklick"). Es erscheint der VHDL-Debugger-Simulator.

IMGONvhldsim.gifIMGOFF

Abbildung 2: VHDL-Debugger-Simulator-Interface

- **IMGONinfo.gifIMGOFF Wichtig:**
Wie beim Texteditor müssen Sie die **Stop-Taste** Ihres HTML-Browsers drücken, um den Dienst zu beenden.
- Sie müssen den Simulator nicht wieder verlassen, nur iconifizieren.
- Den Hyperlink für die Simulation kein zweites mal anwählen (auch bei den folgenden Übungen).

- Über die **RESTART** Option des Simulators (im Execute-Menue...) können Sie jederzeit ein anderes, übersetztes Design simulieren (Es erscheint wieder das Select-Window).
- **Das Simulator-Interface** ist eine Form von "Debugger", dessen "Top Window" den VHDL-Code enthält, und dessen "Bottom Window" das Befehls-Interface zum Simulator ist. Verschiedene Befehle können oben in den Menues aufgerufen werden, andere, die man häufiger braucht, können über die "Quick Buttons" in der Mitte des Displays angeklickt werden. Die restlichen Befehle können in die Befehlsleiste ganz unten eingegeben werden.
- Die **Daten** kann man mit verschiedenen Techniken verfolgen. Wir benutzen die nachfolgenden:
 - **Waveform Display:** das Synopsys "Waveform Display" kann "single bit" und "array type" Signale anzeigen, aber keine INTEGERS.
 - **Signal Evaluation:** ist die Möglichkeit irgendwelche Signal-Namen oder VHDL-Ausdrücke aus dem "Source-Code" auszuwählen, anzuzeigen und dessen Wert interaktiv während der Simulation zu ändern.

8.1 VHDL-System-Befehle

8.2. Bedienung des Simulators (1)

- Der Text im "Top Window" ist der Quell-Code für die "Testbench". Wählen Sie dieses "Code Window" aus und bewegen Sie den Cursor an verschiedene Stellen im Fenster.
- Die Hierarchie des Designs kann durch Befehle durchlaufen und betrachtet werden, die den Befehlen in UNIX ähneln. Sie müssen in die Kommando-Zeile eingegeben werden. "UNIX Type Wild Cards" können auch verwendet werden. Probieren Sie folgende Befehle aus:


```
cd /tb*
ls
ls *'sig    (um alle Signale in dieser Ebene zu sehen)
ls -t
ls -v
ls -t -v
pwd
cd unit_und (auch über die "Quick Buttons" erreichbar)
cd..

```
- um die Signale im "Waveform Display" zu sehen:


```
cd /TB*
trace *'sig

```

Abbildung 3: VHDL-Waveform-Viewer

8.3. Bedienung des Simulators (2)

Die Simulation kann wie folgt kontrolliert werden:

- Sie kann für eine bestimmte Zeitdauer gestartet werden,
- sie kann bis zu einem bestimmten Abbruchpunkt (**Breakpoint**) laufen,
- oder sie geht in kleinen Schritten durch den Code (Next-Button, Step-Button) !
- Wenn Sie einen **Breakpoint** verwenden (**Stop-at-Button**), beachten Sie, daß Sie die Simulation "mittendrin" unterbrechen. Das heißt nicht, daß alle anderen Signale zu dieser Simulationszeit auf den aktuellen Wert gesetzt wurden.
- Starten Sie die Simulation als Reihe von 10ns Schritten. Tippen Sie dazu "10" in das Fenster neben dem **run-Button**, und drücken Sie dann einfach "run", um einen 10ns Abschnitt zu starten:
 - Beobachten Sie das Verhalten des Designs in Ihrem "waveform display". Probieren Sie die verschiedenen Hilfsmittel des "waveform displays", die in den Menues erhältlich

sind. Sie werden zuerst einen Befehl aussuchen müssen und dann ein Objekt, an dem der Befehl ausgeführt wird. Drücken Sie den mittleren Maus-Button, um die Ausführung des bestimmten Befehls zu beenden. (Wenn Sie sich in einem geöffneten Fenster befinden, können Sie keines der "Selection Menues" schließen, welches Sie geöffnet haben. Bewegen Sie den Cursor einfach zur Seite!)

- Schrittweise Ausführung jeder Zeile des Codes unter Verwendung des "**step-Buttons**". Beobachten Sie wann die Werte des Signals aktualisiert werden. Versuchen Sie ein Signal oder einen ganzen VHDL-Ausdruck im Code auszuwählen und benutzen Sie den "**Eval Expr-Button**", um deren Wert zu sehen.
- Setzen Sie einen Breakpoint auf das Ausgangssignal **Value**, indem Sie das Signal im "Code Window" anwählen (markieren), und den "**Event Bkpt-Button**" betätigen. Löschen Sie danach die Zahl "10" in der "Run Box" (control-u löscht das ganze Feld) und drücken Sie den "**run-Button**".
- Breakpoints können auch in "Code-Lines" gesetzt werden: Setzen Sie den Cursor auf die gewünschte Zeile im "Code-Window", wählen Sie dann den "**Stop At-Button**".
- Benützen Sie die Menü-Option "**Breakpoints -> Delete Monitors**", um Breakpoints zu löschen. Löschen Sie aber keine "Items", die als M, PP1, PP2...etc. aufgelistet sind, weil diese mit dem Signal-Fenster verbunden sind.
- Vielleicht ist Ihnen ja aufgefallen, daß ein Fehler in der Ausführung des Modells ist. Wenn Sie die Simulation neu starten wollen (zur Zeit "0" zurückgehen), nehmen Sie die Option "**Execute -> Restart**" im Menü. Alles was Sie vor dem ersten **run**-Befehl festgelegt haben wird wieder ausgeführt.

9. Die VHDL-Synthese

Nachdem Sie die Funktion mit dem Simulator überprüft haben, starten Sie die Synthese durch anwählen des Hyperlinks Synthese bei den System-Befehlen. Es erscheint der Design-Analyzer der Firma Synopsys.

IMGONdesign_analyzer.gif
Abbildung 4: VHDL-Design-Analyzer

- **IMGONinfo.gif** **Wichtig:**
Wie bei den anderen Tools müssen Sie die **Stop-Taste** Ihres HTML-Browsers drücken, um den Dienst zu beenden.
- Sie müssen das Synthesewerkzeug nicht wieder verlassen, nur iconifizieren.
- Den Hyperlink für die Synthese kein zweites mal anwählen (auch bei den folgenden Übungen).
- Über die **Read...** Option des Synthesetools (im File-Menü...) können Sie jederzeit ein anderes Design einlesen.

Für den Design-Analyzer existiert ein sog. "Start-Up-File", der in Ihrer jeweiligen "Directory" enthalten ist. Über diesen File wird für Sie das **IEEE.Std_Logic_1164**-Package automatisch eingelesen. Der "Start-Up-File" legt auch die zu benutzende ASIC-Bibliothek fest (hier die "LSI Logic 10K Serie").

- Lesen Sie mit der Option im Menü File -> Read die Datei "und.vhd". Wenn sie einen "Doppel-Klick" auf den "File-"Namen machen, wird er eingelesen. Überprüfen Sie anhand der Tool-Mitteilungen die Richtigkeit Ihres Codes (Synthese Tools akzeptieren einen sehr geringen Subset des 1076-VHDL-Standards). Warnungen, wie "Design Library management not supported" und "Library clause not supported", können Sie ignorieren. Drücken Sie Die "Cancel"-Taste, um das Fenster zu schließen. (In "OpenWindows" entsteht hier manchmal ein Fehler. Manchmal wird der Text des Fensters nicht sofort vom "Display" gelöscht. Wählen Sie ein anderes Menü-Symbol aus und der Fehler sollte behoben sein.)

In zukünftigen Übungen werden wir eigene Packages definieren und verwenden. Das "Syn-

these Tool" erfordert, daß dieses Package zuerst gelesen wird (Vor den VHDL-Quelldateien, welche das Package referenzieren).

Die häufigste Fehlerquelle bei der Synthese ist das fehlende, nicht eingelesene Package!

- Doppelklicken sie auf das Symbol das Ihr Design repräsentiert, und Sie sehen die erste, generische Netzliste (Umsetzung des VHDL-Codes in toolspezifische Grundgatter).
- Schalten Sie "mapping to gates" im Menue Tools -> Design Optimisation aus. Wenn keine speziellen Synthese-"Constraints" gesetzt sind arbeiten die "Tools" platzoptimiert. Benutzen Sie die Voreinstellungen (defaults) im nachfolgenden "Design-Optimisation-"Menue: Drücken Sie einfach "OK"!
- Nun erhalten Sie die auf die Zielbibliothek hin (LSI10k) optimierte Netzliste (Umsetzung generische Netzliste auf ASIC-Bibliothek).
- Wenn die Synthese fertig ist, machen Sie einen Doppelklick auf das Symbol, um die "gate-level-implementation" zu sehen. "Zoom-"Befehle sind über den rechten "Mouse-Button" zu erreichen: Wählen Sie zuerst den Zoom-Befehl und danach den Ort aus.
- Führen sie die Menue-Option Analysis -> Report aus, um verschiedene Meldungen zu erzeugen.

Wunderbar, Sie haben die erste Runde heil überstanden!

| [9.1 VHDL-System-Befehle](#)

1. Übung: Ein Multiplexer

1.1 ZIEL

- In Ihrer ersten Übung sollen Sie einen Multiplexer beschreiben. Die Testbench ist Ihnen vorgegeben. Sie müssen eventuell noch weitere Stimulivektoren hinzufügen, um die Funktion des Multiplexers zu überprüfen. Anschließend sollen Sie das Modell simulieren und synthetisieren.

1.2 BESCHREIBUNG

- Für den Fotoapparat benötigen Sie ein Display. Dort soll entweder die Bilderanzahl oder die Belichtungszeit angezeigt werden. Der Multiplexer hat folgendes Schaltbild:

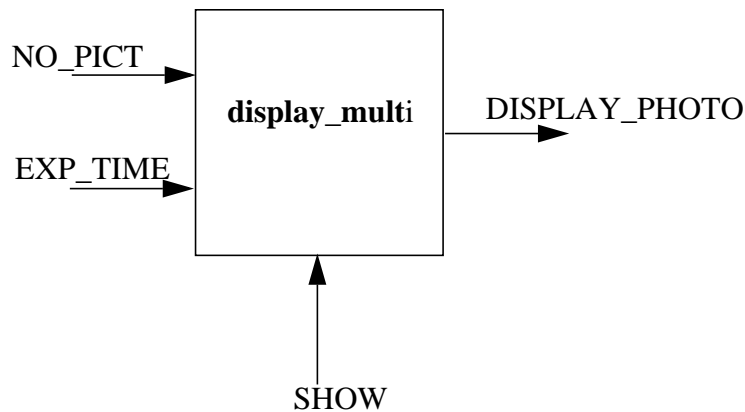


Abbildung 5: Der Multiplexer

- Der Algorithmus für den Multiplexer soll folgendermaßen aussehen:
Wenn $SHOW = 0$ ist, dann soll die Bilderanzahl angezeigt werden, andernfalls die Belichtungszeit.

1.3 ERGÄNZUNGEN

- Die Configuration erzeugt das simulierbare Objekt. Sie wird der Einfachheit wegen, in der Testbench nach der Architecture ausgeführt, könnte aber auch in einem extra File stehen. Wir verwenden die sogenannte Default-Configuration. Hier werden automatisch, für die Simulation, gleichnamige Component und Entity Objekte über die Position (Nicht den Namen) ihrer Port-Signale verdrahtet.

1.4 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie für die Belichtungszeit, Bilderanzahl und Displayanzeige Datentypen Integer. Für das Signal SHOW den Datentyp Std_ulogic.
- **Namen der Designeinheiten:** Nennen Sie die Entity DISPLAY_MULTI, und die Architecture BEHAVE.
- **Filenamen:** Das File heißt **display_multi.vhd** und dementsprechend die Testbench **tb_display_multi.vhd**.

1.5 AUSFÜHRUNGEN

- Beschreiben Sie den Multiplexer. Rufen Sie den Editor auf und ergänzen Sie den VHDL Code. (Hyperlink editieren)
- Analysieren Sie Ihr Programm, und beheben Sie alle Fehler die beim Analysieren auftreten. (Hyperlink analysieren)
- Bearbeiten Sie die Testbench, und fügen Sie die nötigen Stimulivektoren hinzu.

- Analysieren Sie die Testbench. Simulieren Sie Ihr Modell und überprüfen Sie die Funktion. Treten Fehler auf, dann versuchen Sie sie zu beheben.
- Synthetisieren Sie das Modell. Die erzeugten Signale sind in 32 bit breite Vektoren umgewandelt worden. Dies kann durch eine integer RANGE Deklaration vermieden werden. In den nachfolgenden Übungen werden wir darauf zurückkommen.

1.6 VHDL-System-Befehle

2. Übung: Erweiterung des Multiplexers

2.1 ZIEL

- In dieser Übung sollen Sie den Multiplexer um eine Funktion erweitern, die Testbench dementsprechend ergänzen und das neue Modell simulieren.

2.2 BESCHREIBUNG

- Falls während des Betriebs des Fotoapparates ein Fehler auftritt (d.h. Motoraufzug defekt, Film gerissen, etc...), soll eine Fehlermeldung auf dem Display erscheinen. Dazu benötigen wir ein neues Eingangssignal ERR, das '1' ist, falls ein Fehler aufgetreten ist. Der Multiplexer ändert sich folgendermaßen:

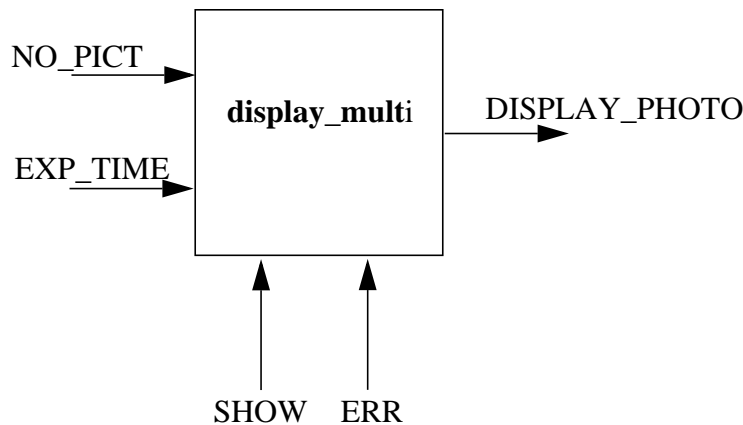


Abbildung 6: Der Multiplexer

- Ist ein Fehler aufgetreten, dann soll DISPLAY_PHOTO eine '10' zugewiesen werden, andernfalls soll die Fallunterscheidung bezüglich SHOW gemacht werden.

2.3 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie für das Signal ERR den Datentyp Std_ulogic.
- **Namen:** Die Namen bleiben gleich. Also DISPLAY_MULTI für die Entity und BEHAVE für die Architecture.

2.4 AUSFÜHRUNGEN:

- Führen Sie die benötigte Modifikation durch.
- Beschreiben Sie in Ihrer Testbench neue Stimulivektoren, mit denen Sie die Funktion Ihres neuen Multiplexer überprüfen können.
- Analysieren Sie die Testbench, und simulieren Sie Ihr Modell anschließend.
- Synthetisieren Sie Ihr Modell.

2.5 VHDL-System-Befehle

3. Übung: Eine Siebensegment-Anzeige

3.1 ZIEL

- Es soll ein VHDL-Modell entwickelt werden, das eine Zahl über eine Siebensegment-Anzeige ausgibt. Später soll diese Anzeige erweitert werden, um letztendlich eine dreistellige Siebensegment-Anzeige zu betreiben. Hierzu steht uns ein bereits geschriebenes Package mit dem Namen **p_digit** zur Verfügung.

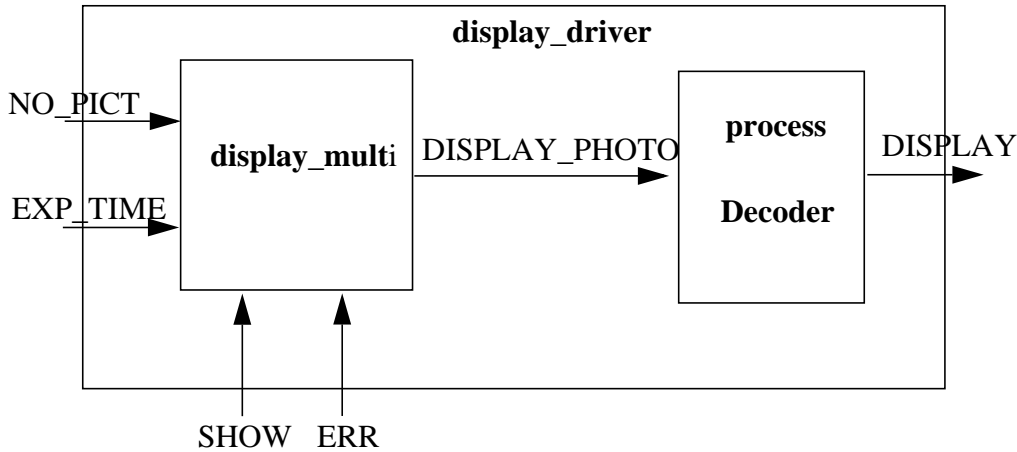


Abbildung 7: Der Display-Driver

3.2 BESCHREIBUNG

- Das Design heißt jetzt DISPLAY_DRIVER. Es hat im Grunde die gleichen Funktionen wie das DISPLAY_MULT, mit der Ausnahme, daß jetzt das neue Ausgangssignal jetzt DISPLAY heißen soll und sieben Bit breit ist, um die Siebensegment-Anzeige zu betreiben. Das alte Signal DISPLAY_PHOTO soll als internes Signal weiterbestehen bleiben, d.h. es ist nach außen nicht sichtbar. Der Grund dafür liegt darin, daß die Zuweisungen auf das neue Ausgangssignal DISPLAY nicht mehr zulässig sind, da sie unterschiedliche Datentypen haben. Unser internes Signal DISPLAY_PHOTO hingegen behält seinen alten Datentyp weiter.
- Wir benötigen jetzt einen Decoder, um die Siebensegment-Anzeige anzusteuern, da wir für die Anzeige ein sieben-Bit breites Signal (DISPLAY) fordern. Es muß also unser internes Signal DISPLAY_PHOTO auf der Siebensegment-Anzeige dargestellt werden. Um den Decoder-Process zu beschreiben, verwenden Sie die Zahlen von 0 bis 9 für die jeweiligen Anzeigenstellen ZERO_SEG bis NINE_SEG. Für die Anzeige eines Fehlers steht Ihnen eine Fehlermeldung E_SEG zur Verfügung, die mit 10 codiert werden soll. Für die Definition der Anzeigenstellen, betrachten Sie das Package **p_digit** mit dem Texteditor.
- Da Sie bei dieser Übung nur eine einstellige Anzeige zur Verfügung haben, muß der Wertebereich der Integerzahlen auf 10 eingeschränkt werden, d.h. es können die Ziffern 0 bis 9 und eine Ziffer E (für die Fehleranzeige) ausgegeben werden.

3.3 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie Std_ulogic, Std_ulogic_vector(6 DOWNTO 0) für das neue Ausgangssignal DISPLAY und Integer RANGE 0 TO 10 für die Bilderanzahl, Belichtungszeit und DISPLAY_PHOTO.
- **Files:** Kopieren Sie den Inhalt der Files aus Übung 1 in die unten anzuwählenden.
- **Namen der Designeinheiten:** Ändern Sie die Namen der Entities.
Modell : Entity DISPLAY_DRIVER, Architecture BEHAVE

Testbench: Entity TB_DISPLAY_DRIVER, Architecture TEST, Configuration CFG_DISPLAY_DRIVER

- **Package:** Verwenden Sie das Package p_digit. Vor der Entity-Beschreibung benötigen Sie zuerst die Anweisung

```
USE work.p_digit.all
```

um das Package in Ihr Design einzubinden. Es ist wichtig, das Package vor dem Design zu analysieren.

- **Beschreibungsstil:** Verwenden Sie einen separaten Process (sensitiv auf DISPLAY_PHOTO), um die Codierung für Ihre Siebensegment-Anzeige zu beschreiben. Verwenden Sie die Case-Anweisung. Es sollen die Zahlen 0 bis 9 angezeigt werden können sowie die Fehleranzeige (E_SEG). Die CASE-Auswahlen erfolgen mit dem Signal DISPLAY_PHOTO und die Zuordnungen des jeweiligen Anzeigenelements erfolgen auf das Ausgangssignal DISPLAY.
- **Siebensegment-Anzeige:** Wie schon erwähnt, steht uns ein Package mit dem Namen **p_digit** zur Verfügung. Mit Hilfe dieses Packages ist es möglich, die Siebensegment-Anzeigestelle graphisch in einem Fenster darzustellen. Verwenden Sie eine nebenläufige Anweisung

```
DISPLAY_DIGIT(DISPLAY);
```

in Ihrer Testbench, die dafür sorgt, daß das Ausgangssignal DISPLAY an das "Fenster" übergeben wird. (Diese Procedure ist im Package gegeben)

Öffnen Sie das Fenster, indem Sie **display_digit** anklicken.

3.4 AUSFÜHRUNGEN:

- Ändern Sie die Namen, wie oben beschrieben.
- Beschreiben Sie den Process, der die Siebensegment-Anzeige ansteuert. Anschließend analysieren Sie Ihr Design, und beheben Sie eventuelle Fehler. Beachten Sie jedoch, Ihr Package vorher zu analysieren.
- Führen Sie die nötigen Änderungen in Ihrer Testbench durch, und vergessen Sie nicht, die Anweisung für Ihre Siebensegment-Anzeige zu ergänzen.
- Öffnen Sie das Fenster für die Siebensegment-Anzeige, und simulieren Sie Ihr Modell. Beheben Sie auftretende Funktionsfehler.
- Synthetisieren Sie Ihr Modell, und optimieren Sie es auf ein Minimum der Fläche. Vergessen Sie nicht vorher, das Package zu synthetisieren. Notieren Sie sich die benötigte Fläche.

3.5 VHDL-System-Befehle

4. Übung: Eine dreistellige Anzeige

4.1 ZIEL

- Ihr DISPLAY_DRIVER soll so modifiziert werden, daß eine dreistellige Anzeige angesteuert werden kann. Sie erlernen die Verwendung eines gemeinsamen Packages, in dem die von mehreren Anwendern benützte Datentypen, definiert sind.

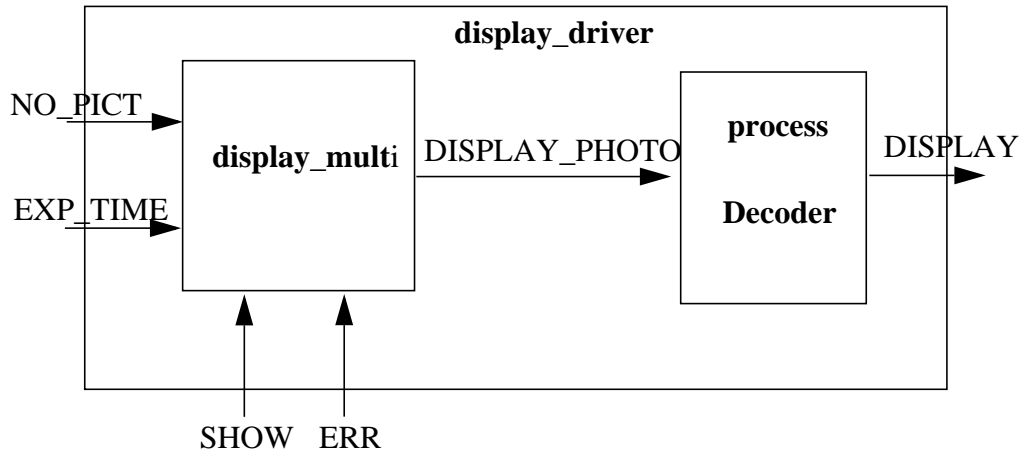


Abbildung 8: Der Display-Driver

4.2 ERGÄNZUNGEN

- Vorteile eines Packages:** An einem Entwurf einer Schaltung mit VHDL arbeiten in der Regel mehrere Entwickler. Deshalb ist es sinnvoll, bei der Verwendung gemeinsamer Signale den gleichen Typ zu definieren. Dies geschieht in einem Package, auf das alle Entwickler zugreifen können. In diesem Fall sollen nun die Datentypen NO_PICT und EXP_TIME vom Typ T_ANZEIGE sein. Dieser Datentyp ist im Package **showdisplay** (Dateiname ist p_showdisplay.vhd) definiert. Ebenfalls ändert sich das Ausgangssignal DISPLAY zu dem Datentyp T_DISPLAY.
- Arrays:** Der Datentyp T_ANZEIGE ist ein dreistelliges Array mit Integerwerten von 0 bis 10. Der Typ T_DISPLAY ist ein dreistelliges Array mit 7 Bit Signalen für die Siebensegment-Anzeige. Damit wird ein dreistelliges Anzeigenfeld angesteuert.
- Integer mit Range:** In VHDL kann man den Bereich einer Integerzahl mit Hilfe der RANGE-Anweisung einschränken. Wird dies nicht gemacht, so wird das jeweilige Signal bei der Synthese als 32 Bit Signal dargestellt, das dem Zahlenbereich einer Integerzahl dual entspricht. Bei der Angabe eines Bereiches spart man Fläche ein, da nur der angegebene Bereich dementsprechend synthetisiert wird (bei RANGE 0 TO 64 wird also ein 7 Bit breites Signal synthetisiert).
- Schleife:** Um den Process für die Ansteuerung des Displays zu beschreiben, können Sie die FOR-LOOP-Anweisung verwenden. Die Schleife muß so oft durchlaufen werden, wie Anzeigestellen betrieben werden sollen. Pro Schleifendurchgang soll dem jeweiligen Integerwert die entsprechende Anzeige zugeordnet werden.

4.3 BESCHREIBUNG

- Die Funktion des DISPLAY_DRIVER bleibt identisch, jedoch verändern sich die Datentypen. Verwenden Sie jetzt für die Eingangssignale NO_PICT und EXP_TIME den Datentyp T_ANZEIGE und für das Ausgangssignal DISPLAY den Typ T_DISPLAY, die beide in einem Package showdisplay definiert sind.

```
TYPE T_ANZEIGE IS ARRAY (2 DOWNTO 0) OF integer RANGE 0 TO 10;TYPE
```

```
T_DISPLAY IS ARRAY (2 DOWNTO 0) OF std_ulogic_vector(6 DOWNTO 0);
```

4.4 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie die im Package vordefinierten Datentypen T_ANZEIGE und T_DISPLAY.
- **Filenamen:** Die Namen bleiben gleich, also display_driver.vhd und tb_display_driver.vhd.
- **Namen der Designeinheiten:** Die Namen bleiben ebenfalls gleich.
- **Testbench:** Die Stimuli Ihrer Testbench müssen entsprechend den neuen Datentypen verändert werden. Sie können die FOR-LOOP-Anweisung verwenden, um Stimuli zu beschreiben.
- **Package:** Beachten Sie, daß Sie jetzt ein anderes Package **showdisplay** zum Betrieb der dreistelligen Anzeige verwenden.
Öffnen Sie das neue Anzeigenfenster, indem Sie
display_photo
anklicken und in der Testbench die Anweisung
display_photo(DISPLAY);
verwenden, um das Signal an das Fenster zu übergeben.

4.5 AUSFÜHRUNGEN

- Führen Sie die notwendigen Änderungen in Ihrem Design und der Testbench durch. Analysieren Sie beide, und beheben Sie alle auftretenden Fehler.
- Simulieren Sie Ihr Design, und überprüfen Sie die Funktion.
- Synthetisieren Sie Ihr Design, und optimieren Sie es auf ein Minimum der Fläche.

4.6 VHDL-System-Befehle

5. Übung: Ein Decoder

5.1 ZIEL

- Es soll ein Decoder in einer eigenen Entity/Architecture für die Tasteneingabe entworfen werden.

5.2 BESCHREIBUNG

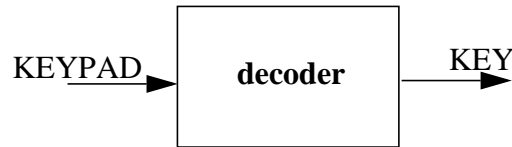


Abbildung 9: Der Decoder

- Es sind zehn verschiedene Knöpfe für die Belichtungszeiten vorgegeben. Der Decoder soll die Eingangssignale so umwandeln, daß das Ausgangssignal KEY vom Typ T_ANZEIGE dargestellt werden kann. Das Eingangssignal ist vom Typ Std_ulogic_vector (1 TO 10) und kommt von einer festen Keypad-Einheit. Es besteht aus neun Nullen und einer Eins, die jeweils nach rechts durchgeschoben wird. Das Signal KEY soll den Zahlenwert der Belichtungszeit bekommen, d.h. wird der Keypad "1000000000" gedrückt, so entspricht das einer Belichtungszeit von einer Sekunde und KEY wird (0,0,1) zugewiesen, für "0100000000" soll KEY den Wert (0,0,2) bekommen, usw (Belichtungszeiten im Raster von $2 \exp(-i)$ für $i = 0 \dots 9$). Alle übrigen Werte (others) des Keypadsignals soll (0,0,0) zugewiesen werden. Verwenden Sie eine CASE-Anweisung.

5.3 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie Std_ulogic_vector, und den selbstdefinierten Typ T_ANZEIGE (siehe Package showdisplay).
- **Filenamen:** Das Design heißt **decoder.vhd**, die Testbench **tb_decoder.vhd**.
- **Namen der Designeinheiten:** Die Entity heißt DECODER, die Architecture RTL und die Configuration CFG_DECODER.

5.4 AUSFÜHRUNGEN:

- Beschreiben Sie den Decoder. Analysieren Sie ihn, und beheben Sie auftretende Fehler.
- Schreiben Sie eine Testbench dazu, mit der die Funktion Ihres Designs überprüft werden kann, und simulieren Sie den Decoder.
- Synthetisieren Sie Ihr Design, und optimieren Sie es auf ein Minimum der Fläche.

5.5 VHDL-System-Befehle

6. Übung: Ein Register

6.1 ZIEL

- Es soll ein Register beschrieben werden, das die Belichtungszeit speichert.

6.2 BESCHREIBUNG

- Wird eine neue Belichtungszeit ausgewählt, so soll dieser Wert gespeichert werden. Dieses Register arbeitet synchron und behält oder erneuert seinen gespeicherten Wert mit jeder ansteigenden Taktflanke. Das Register soll rücksetzbar auf die Belichtungszeit von 1 sec. sein. Das Modell sieht folgendermaßen aus:

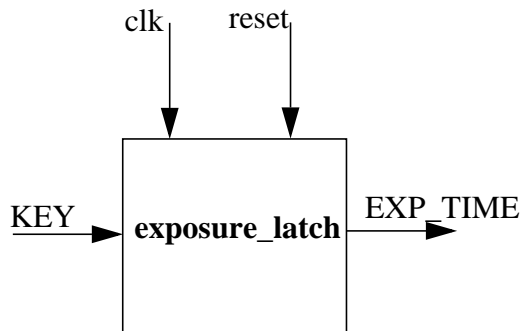


Abbildung 10: Das Belichtungsregister

- Der Algorithmus lautet:

```

IF reset = '1' THEN
  reset design
ELSIF rising clock
  IF KEY pressed
    load it
  END IF
END IF

```

6.3 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie Std_ulogic und den selbstdefinierten Typ T_ANZEIGE.
- **Filenamen:** Das Design heißt exposure_latch.vhd und die Testbench tb_exposure_latch.vhd.
- **Namen der Designeinheiten:** Die Entity heißt EXPOSURE_LATCH, die Architecture RTL und die Configuration CFG_EXPOSURE_LATCH .
- **Package:** Verwenden Sie das Package showdisplay .
- **Beschreibungsstil:** Verwenden Sie einen Process, der auf die Signale clk und reset sensitiv ist.

6.4 AUSFÜHRUNGEN:

- Beschreiben Sie das Belichtungsregister, analysieren Sie es, und beheben sie auftretende Fehler.
- Schreiben Sie eine Testbench, mit der die Funktion Ihres Designs überprüft werden kann, und simulieren sie das Belichtungsregister.
- Synthetisieren Sie Ihr Design, und optimieren Sie es auf ein Minimum der Fläche.

6.5 VHDL-System-Befehle

7. Übung: Ein Zähler

7.1 ZIEL

- Realisieren Sie einen Zähler zur Überwachung der maximalen Transportzeit.

7.2 BESCHREIBUNG

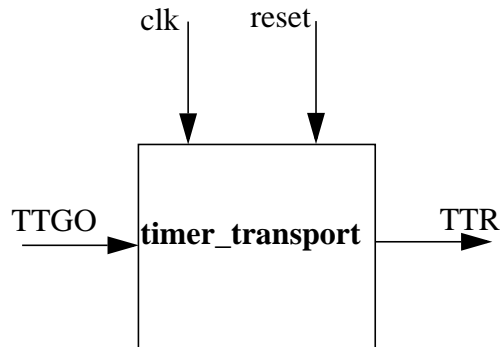


Abbildung 11: Der Zähler für die Transportzeit

- Sobald das Signal TTGO (Timer Transport Go) auf '0' geht, soll ein Signal TTR um zwei Sekunden verzögert für einen Takt auf '1' gehen. TTGO bleibt einen Takt auf dem Wert 0 (wird durch das Modul photo_control gewährleistet). Die Verzögerung um zwei Sekunden entspricht bei einem Takt von 8192 Hz einer Zähldauer bis 16384. Der Zähler soll bei einem Startsignal von 0 bis 16384 zählen und dann das erforderliche Ausgangssignal liefern.

7.3 ERGÄNZUNGEN

- Vorteile von Variablen: Signale stehen global zur Verfügung und werden immer erst am Ende eines Prozesses erneuert, d.h. wird innerhalb eines Process dem Signal zuerst eine '2' und anschließend der Wert '4' zugewiesen, so wird am Ende des Process das Signal auf '4' gesetzt, und die '2' wird überhaupt nicht beachtet. Variablen hingegen sind lokal nur innerhalb eines Process gültig, und ihr Wert wird sofort zugewiesen (in der überarbeiteten Norm VHDL 93 werden auch globale Variable zugelassen).
- Bei dem Design soll das Ausgangssignal TTR für nur einen Takt auf '1' gehen, wenn die Zeit abgelaufen ist. Dieses Problem kann man mit Variablen lösen. Durch Verwendung von Variablen beginnt der Zähler bei dem Startsignal sofort loszulaufen und nicht erst einen Takt später. Die Zählvariable soll counter heißen. Unser Zählalgorithmus sieht folgendermaßen aus:

```

IF reset ...
ELSIF clk....
  TTR is assigned '0'
  IF TTGO.....
    var is set to '1'
    counter is set to '0'
  END..
  IF var = '1' ...
    IF counter /= ...
      ...
    ELSE
      TTR ...
      var is set to '0'
    END..
  END..
END...
END...

```

7.4 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie Integer mit Range-Angabe für das Zählsignal und Std_ulogic für die anderen Signale.
- **Filenamen:** Das Design heißt **timer_transport.vhd** und die Testbench **tb_timer_transport.vhd**.
- **Namen der Designeinheiten:** Die Entity heißt TIMER_TRANSPORT, die Architecture BEHAVE, die Testbench TB_TIMER_TRANSPORT und die Configuration CFG_TIMER_TARNSPORT.
- **Package:** Bei dieser Beschreibung benötigen Sie kein Package.

7.5 AUSFÜHRUNGEN:

- Beschreiben Sie den Zähler, und schreiben Sie eine Testbench dafür.
- Simulieren Sie Ihr Design, und überprüfen Sie die Funktion.
- Wieviele FlipFlops muß der Zähler haben ?
- Synthetisieren Sie Ihren Zähler, optimieren Sie Ihn auf ein Minimum der Fläche, und überprüfen Sie Ihre Antwort.

7.6 VHDL-System-Befehle

8. Übung: Ein Timer

8.1 ZIEL

- Es soll der Timer für die Belichtungszeit entworfen werden. Er wird durch einen Zähler realisiert, der bis zu einer bestimmten Zahl hochzählt. Sensitiv auf das Eingangssignal TEGO soll der Bilderzähler realisiert werden.

8.2 BESCHREIBUNG

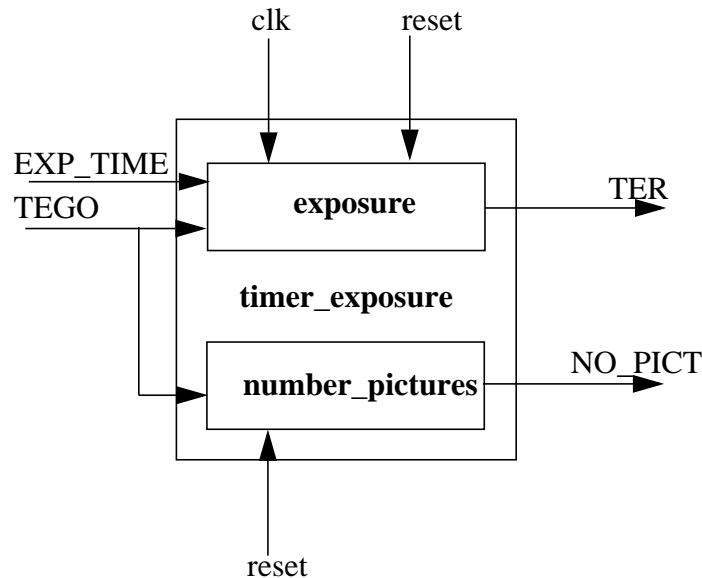


Abbildung 12: Der Zähler für die Belichtungszeit

- Das Design besteht aus zwei Modulen. Das erste ist der Timer für die Belichtungszeit, das zweite der Bilderzähler. Sobald TEGO für einen Takt auf '0' springt, soll der Timer für die Belichtungszeit loslaufen und nach der gewünschten Zeit das Ausgangssignal TVR liefern. Auch hier tritt wieder das Problem auf, daß das Ausgangssignal TVR für einen Takt auf '1' gehen soll und sonst auf '0' bleibt. Über den Reset-Eingang, wird der Zähler auf '0' zurückgesetzt. Sobald der Zähler seinen Endwert erreicht hat, soll er ebenfalls wieder auf '0' gesetzt werden.
- Der Bilderzähler soll sich bei der abfallenden Flanke von TVGO um '1' erhöhen und über den Reset-Eingang rücksetzbar auf '0' sein.

8.3 ERGÄNZUNGEN

- Lesen von Ausgangssignalen: In VHDL ist es nicht möglich, Ausgangssignale vom Modus OUT zu lesen. In Ihrem Fall jedoch müssten Sie dem Ausgangssignal NO_PICT, pro definiertem Event einen Wert um '1' erhöht zuweisen. (Dies führt beim Analysieren zur Fehlermeldung: ports of mode out cannot be read.) Man kann dieses Problem lösen, indem man diesen Port als INOUT definiert, doch würde sich das durch die ganze Hierarchie ziehen. Ein besserer Weg, dieses Problem zu lösen, ist es, sich ein Zwischensignal zu definieren, mit dem der Zählalgorithmus ausführt und später mit einer nebenläufigen Anweisung dem Ausgangssignal zugewiesen wird.
- Das Ausgangssignal vom Bilderzähler soll von Typ T_ANZEIGE sein. Dieser Zähler wird mit einem anderen Algorithmus beschrieben. Das Zählsignal soll ebenfalls vom Typ T_ANZEIGE sein. Man definiert sich Variablen für die jeweiligen Stellen von T_ANZEIGE. Die Erste zählt nun bis 9 hoch, danach wird der nächsten Stelle 1 zugewiesen und der Ersten 0 usw... . Am Schluß erfolgen die entsprechenden Zuweisungen an die

Ausgangssignale.

8.4 ERLÄUTERUNGEN

- **Datentypen:** Verwenden Sie Integer mit Range-Angabe für das Zählsignal, Std_ulogic und T_ANZEIGE für die anderen Signale.
- **Filenamen:** Das Design heißt **timer_exposure.vdh** und die Testbench **tb_timer_exposure.vhd**.
- **Namen der Designeinheiten:** Die Entity heißt TIMER_EXPOSURE, die Architecture BEHAVE, die Testbench TB_TIMER_EXPOSURE und die Configuration CFG_TIMER_EXPOSURE.
- **Package:** Sie benötigen wieder das Package **p_showdisplay.vhd**.
- **Beschreibungsstil:** Sie benötigen zwei Prozesse. Einen für die Belichtung und einen für die Bilderanzahl. Beachten Sie, daß das Eingangssignal vom Typ T_ANZEIGE ist. Sie müssen das Signal in eine Integerzahl umwandeln. Tun Sie dies indem Sie sich eine Variable definieren. Überlegen Sie sich einen passenden Algorithmus um die Belichtungszeit zu realisieren.

8.5 AUSFÜHRUNGEN:

- Beschreiben Sie beide Module, und schreiben Sie eine Testbench dafür.
- Simulieren Sie Ihr Design, und überprüfen Sie die Funktion.
- Wieviele FlipFlops müssen diesmal die Zähler haben ?
- Synthetisieren Sie Ihr Design, optimieren Sie es auf ein Minimum der Fläche, und überprüfen Sie Ihre Antwort.

8.6 VHDL-System-Befehle

9. Übung: Zustandsautomat Fotosteuerung

9.1 ZIEL

- Sie sollen einen Zustandsautomaten für die Fotosteuerung beschreiben.

9.2 BESCHREIBUNG

- Die Fotosteuerung übernimmt die Steuerung aller internen Signale für den Fotoapparat.

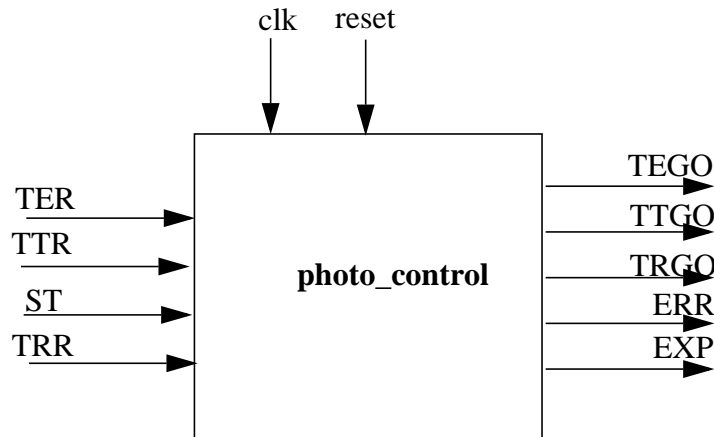


Abbildung 13: Die Fotosteuerung

- **Die Eingänge (Entscheidungssignale) sind:**
 - TER: Timer für Belichtungszeit ist abgelaufen (Signal für eine Taktperiode high)
 - TTR: Timer für die maximale Transportzeit ist abgelaufen (Signal für eine Taktperiode high)
 - ST: Auslösetaste des Fotoapparates (aktiv = high)
 - TRR: Filmtransport erfolgreich beendet (Signal für eine Taktperiode high)
- **Die Ausgänge (Steuersignale) sind:**
 - TEGO: Start des Timers für Belichtungszeit (Signal für eine Taktperiode low)
 - TTGO: Start des Timers für maximale Transportzeit (Signal für eine Taktperiode low)
 - TRGO: Motoraufzug (an = high, aus = low)
 - ERR: Fehleranzeige (an = high, aus = low)
 - EXP: Verschluss (auf = high, zu = low)
- Das Schaltwerk Fotosteuerung soll nach Niederdrücken der Auslösetaste ST den Verschluss öffnen, die eingestellte Belichtungszeit abwarten und danach gleichzeitig den Verschluss schließen und das Startsignal für den Motoraufzug liefern.
- Ein dauerhaftes Niederdrücken der Auslösetaste ST soll das Aufnehmen von Bildfolgen ermöglichen.
- Ein erfolgreicher Filmtransport wird durch das synchrone Signal TRR (Transport Ready) gemeldet (für eine Taktperiode high).
- Bei Filmende oder Verklemmen des Films soll nach 2 sec. der Motoraufzug gestoppt werden und eine Fehleranzeige ERR ausgegeben werden. Die erneute Betätigung der Auslösetaste (z.B. nach einem Filmwechsel) in diesem Zustand führt zu einer erneuten Betätigung des Motoraufzugs und zum Rücksetzen der Fehleranzeige ERR.
- Die maximale Transportzeit wird von dem Modell **timer_transport** realisiert.
- Die Belichtungszeit wird von dem Modell **timer_exposure** realisiert.
- Die beiden Modelle **timer_transport** und **timer_exposure** werden durch die beiden Signale TTGO bzw. TEGO gestartet, und setzen nach Ablauf der maximalen Filmtransportzeit bzw. der eingestellten Belichtungszeit die Signale TTR und TER für eine Taktperiode

auf high.

- Das Schaltwerk Fotosteuerung soll als ein Moore-Schaltwerk ausgeführt werden.

9.3 ERGÄNZUNGEN

- **Zustandsautomat** : Es gibt in VHDL mehrere Möglichkeiten einen Zustandsautomaten zu beschreiben (vgl. Tutorial, Moore/Mealy Realisierungen). Die saubere Realisierung besteht aus einem Prozeß für die Zustandsweitschaltung mit der Berechnung des Folgezustandes und einem Prozeß zur Berechnung der Ausgangsgrößen. Wenn man versucht den Automaten in nur einem Process zu beschreiben, der lediglich vom Takt und dem Rücksetzsignal getriggert wird, werden Flip-Flops nicht nur für das Zustandssignal, sondern auch für die Ausgänge erzeugt und die Ausgangssignale werden einen Takt später aktualisiert. Dieses Verhalten würde der Spezifikation widersprechen.

Zuerst definiert man sich einen Typ `zustand_typ`, in dem alle Zustände definiert sind. Dann definiert man sich ein Signal dieses Typs. Mit einer CASE-Anweisung können die entsprechenden Aktionen durchgeführt werden, die in dem jeweiligen Zustand ausgeführt werden sollen. Bei dieser Übung setzt sich die Beschreibung aus zwei Prozessen zusammen. Im ersten wird die Übergangslogik beschrieben.

Bei der Fotosteuerung gibt es Übergänge, in denen Entscheidungsvariablen mit don't-care-Bedingungen auftreten. Für die Synthese müssen '-' (don't cares) explizit beschrieben werden. Man bedient sich einer UND-Verknüpfung mit (1, 0, 0, 0) und vergleicht mit (0, 0, 0, 0) um einen Übergang (1, -, -, -) zu beschreiben. Also:

```
IF (input AND "1000") = "0000" THEN ...
```

Im zweiten Process wird die Ausgabelogik mit Hilfe der SELECT-Anweisung beschrieben ("|" ist eine ODER Verknüpfung). Zum Beispiel:

```
WITH state SELECT
    TVGO <=
        '0' WHEN ...
        '1' WHEN state2 | state3 | ....
```

9.4 ERLÄUTERUNGEN

- **Datentypen** : Verwenden Sie `Std_ulogic` und `Std_ulogic_vector` für Ihre Signale. Definieren Sie sich einen eigenen Typ mit Namen z.B `zustand_typ` Dieser Typ enthält die möglichen Zustände des Automaten. Definieren Sie ein Signal `zustand` dieses Typs.
- **Namen**: Wählen Sie sich Ihre Namen selber, achten Sie auf sinnvolle Bezeichnungen.

9.5 AUSFÜHRUNGEN:

- Beschreiben Sie den Automaten, und schreiben Sie eine Testbench dafür.
- Simulieren Sie Ihr Design, und überprüfen Sie die Funktion.
- Wieviele FlipFlops muß Ihr Design haben ?
- Synthetisieren Sie Ihr Design, optimieren Sie es auf ein Minimum der Fläche, und überprüfen Sie Ihre Antwort.

9.6 VHDL-System-Befehle

10. Übung: Zustandsautomat Display

10.1 ZIEL

- Es soll ein Zustandsautomaten für die Anzeige entworfen werden.

10.2 BESCHREIBUNG

- Der Displaycontroller übernimmt die Steuerung der Ausgabe auf das Display.

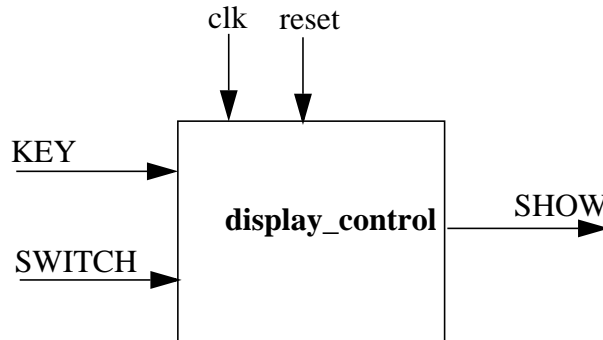


Abbildung 14: Der Displaycontroller

- **Die Eingänge (Entscheidungssignale) sind:**

KEY: Ein Knopf für die Belichtungszeit wurde gedrückt

SWITCH: Der Schalter zum Umschalten zwischen der Bilderanzahl und der Belichtungszeit

Der Ausgang (Steuersignal) ist:

SHOW: Wenn SHOW = '0' ist dann wird die Bilderanzahl angezeigt, andernfalls, wenn SHOW = '1' ist, dann die Belichtungszeit.

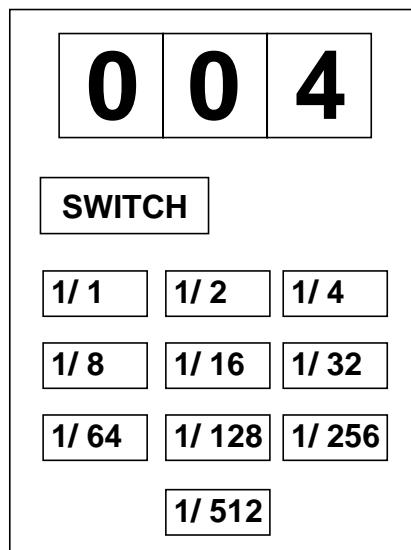


Abbildung 15: Die Eingabeeinheit

- Wenn ein Belichtungs Knopf gedrückt wird, soll auf dem Display sofort die Belichtungszeit angezeigt werden.
- Wird SWITCH gedrückt, so kann man zwischen den beiden Anzeigemöglichkeiten wechseln.
- Das Schaltwerk soll als ein Moore-Schaltwerk ausgeführt werden.

10.3 ERLÄUTERUNGEN

- Nachdem Sie in der vorherigen Übung schon einen Zustandsautomaten beschrieben haben,

sollen Sie dies nun allein tun.

10.4 AUSFÜHRUNGEN:

- Beschreiben Sie den Automaten, und schreiben Sie eine Testbench dafür.
- Simulieren Sie Ihr Design, und überprüfen Sie die Funktion.
- Wieviele FlipFlops muß Ihr Design enthalten ?
- Synthetisieren Sie Ihr Design, optimieren Sie es auf ein Minimum der Fläche, und überprüfen Sie Ihre Antwort.

10.5 VHDL-System-Befehle

11. Übung: Der Fotoapparat

11.1 ZIEL

- Zusammenfassung aller Einzelmodule zur kompletten Fotosteuerung

11.2 BESCHREIBUNG

- Nachdem Sie nun alle Designs des Fotoapparates fertig beschrieben haben, müssen Sie nur noch alles zusammenschließen und eine Gesamtsimulation durchführen.

11.3 ERGÄNZUNGEN

- Strukturelle Modellierung: Strukturelle Modellierung bedeutet im allgemeinen die Verwendung (= Instantiierung) und das Verdrahten von Komponenten in Form einer Netzliste. VHDL bedient sich einer dreistufigen Vorgehensweise:
 - Komponentendeklaration
 - Komponenteninstantiierung
 - Komponentenkonfiguration
- Beschreiben Sie dementsprechend Ihren Fotoapparat.

11.4 ERLÄUTERUNGEN

- Sie sollen diese Aufgabe selber lösen. Im Bild ist die Fotosteuerung skizziert, allerdings fehlen die meisten Signalnamen. Fertigen Sie eine eigene Zeichnung Ihres Designs an und benennen Sie alle Signale.

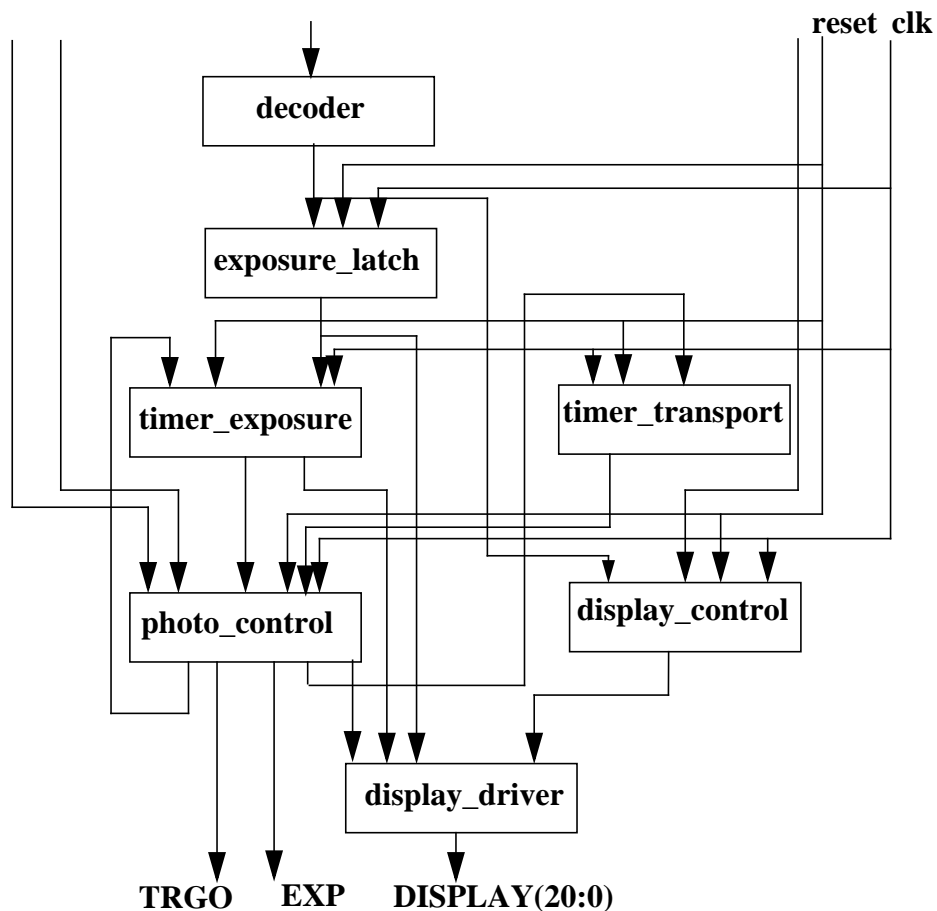


Abbildung 16: Die gesamte Fotosteuerung

11.5 AUSFÜHRUNGEN:

- Fassen Sie Ihre Module zusammen. Simulieren und synthetisieren Sie den Fotoapparat.
- Notieren Sie sich die Fläche und Gatteranzahl des gesamten Designs.
- Ist die Summe der einzelnen Ergebnisse gleich der des Gesamtergebnisses?
(Die Synthesetools können bei der Optimierung des Gesamtdesigns zu besseren, sprich kleineren Ergebnissen gelangen).
- Notieren Sie sich Fläche und Laufzeiten.
- Markieren Sie alle Module in der oberen Ebene des Fotoapparat und lösen sie die Hierarchie des Designs auf (Edit -> Ungroup (all Levels)).
- Speichern Sie das Synthesergebnis als VHDL-File (photo_gates.vhd) für die Nachfolgende Simulation auf Gatterebene ab.

11.6 VHDL-System-Befehle

12. Übung: Simulation auf Gatterebene

12.1 ZIEL

- Der Fotoapparat soll auf Gatterebene verifiziert werden.

12.2 BESCHREIBUNG

- Nach der Synthese muß überprüft werden, ob das Ergebnis immer noch das gleiche Verhalten wie vor der Synthese besitzt. Hierzu muß die Gatterebenenbeschreibung übersetzt und mit der alten Testbench verifiziert werden.

12.3 ERGÄNZUNGEN

- Für eine Simulation auf Gatterebene benötigt man eine passende Bibliothek. In dieser Übung wird die schon vorhandene FullTimingGateSimulation (FTGS) Bibliothek der lsi_10k Bibliothek verwendet. Ansonsten gibt es zwei Möglichkeiten, an eine solche Bibliothek heranzukommen:
 - Der Hersteller liefert die Simulationsbibliotheken
 - Man erzeugt aus den Synthesebibliotheken die passenden Simulationsbibliotheken (z.B. unter Synopsys mit dem Library Analyzer; *liban -arch FTGS -hazard inertial lsi_10k.db*)
- Eine Gatterebenenimulation nach der RTL-Synthese wird auch als pre-layout Simulation bezeichnet. Vergleichen Sie hierzu Abbildung 17:

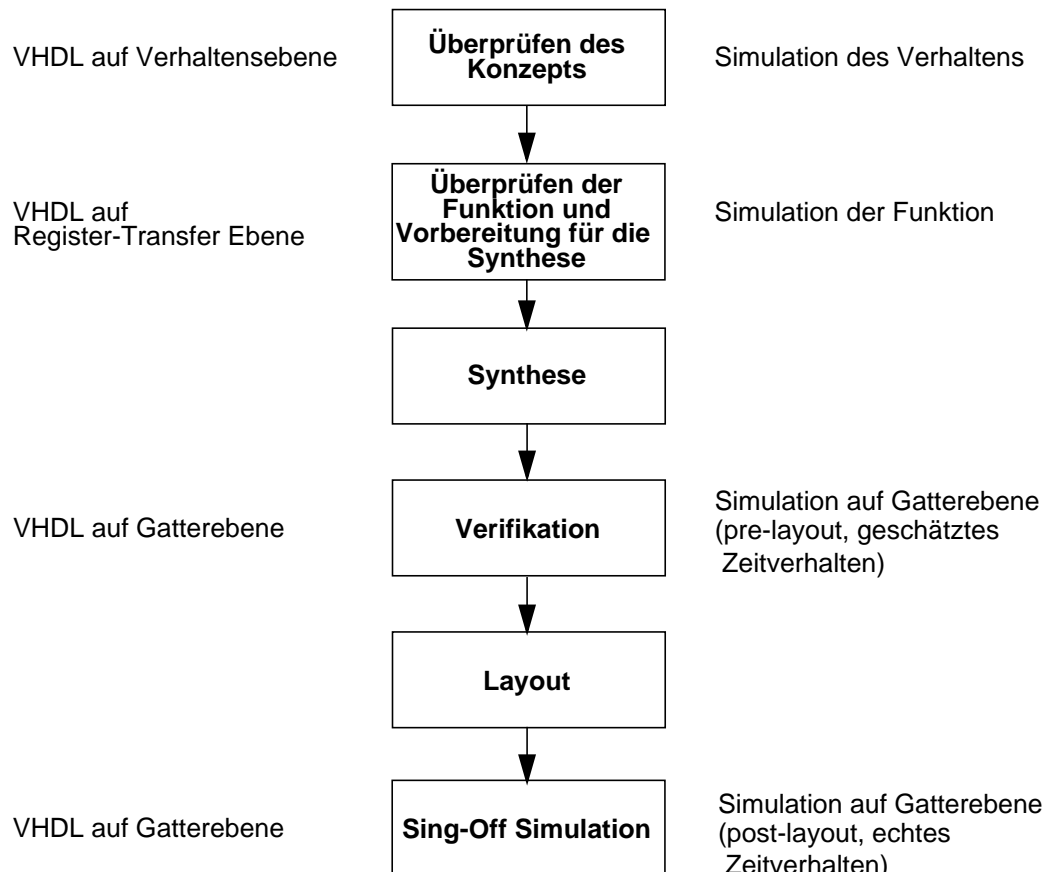


Abbildung 17: Einordnung der Gatterebenenimulation im Design Ablauf.

12.4 ERLÄUTERUNGEN

- Sie sollen diese Aufgabe selber lösen. Nehmen Sie die nötigen Änderungen im erzeugten VHDL-File vor und überprüfen Sie die Funktionalität.

12.5 AUSFÜHRUNGEN:

- Löschen Sie die Attribut- und Typendeklarationen aus dem Package Header CONV_PACK_photo in der Gatternetzliste (photo_gates.vhd).
- Übersetzen Sie die nötigen VHDL-Files und simulieren Sie den Fotoapparat.

12.6 VHDL-System-Befehle

13. Übung: Synthese mit Optimierungs- und Arbeitskriterien

13.1 ZIEL

- Der Fotoapparat soll unter Berücksichtigung mehrerer Optimierungskriterien synthetisiert und anschließend auf Gatterebene verifiziert werden.

13.2 BESCHREIBUNG

- In dieser Übung setzen Sie vor der Synthese willkürliche Werte für verschiedene Kriterien fest und synthetisieren den Fotoapparat erneut. Anschließend wird die synthetisierte Schaltung mit dem neuen Zeitverhalten auf Gatterebene simuliert.

13.3 ERGÄNZUNGEN

- Um ein möglichst gutes Syntheseresultat zu bekommen, muß man die spätere Arbeitsumgebung der Fotosteuerung berücksichtigen. Daß heißt, vor der Synthese müssen nicht nur Optimierungskriterien (Flächen-, Zeit-, Verlustleistungsoptimierung) sondern auch Arbeitskriterien angegeben werden.
- Generell können in Synopsys folgende Attributklassen verwendet werden:
 - *Clocks* -> Zur Kennzeichnung der Taktleitung.
 - *Operating Environment* -> Charakterisierung der Umgebung.
 - *Optimization Constraints* -> Definition von Optimierungszielen (z.B. bestimmte Fläche)
 - *Optimization Directives* -> Weitere Attribute (zumeist physikalische), die die Optimierung beeinflussen.
- Eine FullTimingGateSimulation (FTGS) Bibliothek besitzt Defaultwerte für die verschiedenen Zeitkonstanten. Um diese mit den aktuellen, sich **nach** der Synthese ergebenden Werten überschreiben zu können, muß man diese Werte in einem File abspeichern. Dies erfolgt in einem Standard Delay Format, dem SDF. Das abgespeicherte File kann dann vor der Simulation eingelesen werden.

13.4 ERLÄUTERUNGEN

- Lesen Sie das gesamte Design ein, stellen Sie die entsprechenden Attribute ein und synthetisieren Sie es. Führen Sie danach eine Simulation auf Gatterebene mit den neuen Zeitkonstanten durch.

13.5 AUSFÜHRUNGEN:

- Lesen Sie das gesamte Design in den Design-Analyzer ein.
- Markieren Sie alle Module in der oberen Ebene des Fotoapparats und lösen Sie die Hierarchie des Designs auf (Edit -> Ungroup (all Levels)).
- Setzen Sie (willkürliche, aber „vernünftige“) Werte für folgende Attribute: (Menu *Attributes* ->)
 - Clock -> Period, Dont Touch Network, Skew und Uncertainty (Clock Leitung vorher selektieren!)
 - Operating Environment -> Operating Conditions (Top Entity vorher selektieren!)
 - Operating Environment -> Wire Load (Mit *report_lib lsi_10k* im Setup -> Commanda-Window kann man sich u.a. über die Wire Loads informieren)
 - Operating Environment -> Timing Ranges sind für die lsi_10k Bibliothek **nicht** definiert.
 - Optimization Constraints -> Design Constraints -> Max Area (An vorherigem Syntheseresultat orientieren, 0 käme einer absoluten Flächenoptimierung gleich!), Max Fanout (für alle Ausgänge).

- Optimization Constraints -> Timing Constraints (Hier nicht unbedingt erforderlich)
- Optimization Directives -> Design (Top Entity!) -> Ungroup, Boundary Optimization, Flip Flop, Port is Pad, Design Pad Attributes (z.B. 2,3,0,5), Flatten Logic (Effort, Flatten Minimize, Flatten Phase), Structure Logic (Beide).
- Zusätzlich für die Eingangs-Ports (selektieren!) der obersten Entity:
 - Operating Environment -> Input Delay (z.B. je 2 ns zum Taktsignal)
 - Operating Environment -> Drive Strength (z.B. 0.02; je größer der Wert, desto stärker der externe Treiber; 0 ist der Defaultwert).
- Zusätzlich für die Ausgangs-Ports (selektieren!) der obersten Entity:
 - Operating Environment -> Output Delay (z.B. je 2 ns zum Taktsignal)
- Markieren Sie alle Module in der oberen Ebene des Fotoapparats und lösen Sie die Hierarchie des Designs auf (Edit -> Ungroup (all Levels)).
- Synthetisieren Sie das Design.
- Vergleichen Sie die Fläche und Laufzeiten mit dem vorherigen Syntheseergebnis.
- Speichern Sie das Syntheseergebnis als VHDL-File (photo_gates.vhd) für die Nachfolgende Simulation auf Gatterebene ab.
- Speichern Sie die Zeitinformationen in einem SDF-File (File -> Save Info -> Design Timing -> photo_gates.sdf) für die Nachfolgende Simulation auf Gatterebene ab.
- Löschen Sie die Attribut- und Typendeklarationen aus dem Package Header CONV_PACK_photo in der Gatternetzliste (photo_gates.vhd).
- Führen Sie eine Simulation durch.

13.6 VHDL-System-Befehle

14. Übung: Optimierung des VHDL Designs

14.1 ZIEL

- Die Fotosteuerung soll funktional verbessert werden.

14.2 BESCHREIBUNG

- Die Funktion der gesamten Fotosteuerung ist zu betrachten und auf Einsparungsmöglichkeiten untersucht werden. Hier zeigt sich, wie modular und damit leicht veränderbar man einen Entwurf gestaltet hat. Desweiteren wird der Umgang mit Packages und die Verwendung von eigenen Untertypen vertieft. Auch hier wird der Vorteil sichtbar, alle verwendeten Typen und Konstanten in einem Package zu verwalten.

14.3 ERGÄNZUNGEN

- Während eines Entwurfes kommt es oft vor, daß sich die ursprüngliche Spezifikation ändert, daß konzeptionelle Fehler in der Simulation sichtbar werden, daß sich das Modellierungsziel (Geschwindigkeits-, flächen- oder leistungsoptimiert) ändert oder daß man schlichtweg feststellt, redundante Funktionen eingebaut zu haben. Es kommt also seltenst vor, daß ein Entwurf sich nach dem erstmaligen Kodieren nicht mehr ändert. Als Konsequenz daraus ergibt sich, daß man sein Design modular gestalten sollte und alle verwendeten Typen, Konstanten und Funktionen in Packages verwalten sollte.
- In dieser Übung werden Sie nun selber feststellen, wie einfach oder schwer sich das Design modifizieren läßt. Da bei den ersten Übungen der Schwerpunkt nicht auf der Verwendung von Packages liegt, wurden diese bisher immer bereitgestellt und enthielten nur die notwendigen Deklarationen. Um aber auch seine eigenen Typen im gesamten Design leicht ändern zu können, ist es notwendig, diese und die abgeleiteten Untertypen in einem Package zu verwalten. Das heißt, daß Sie nun selbst eigene Untertypen in das entsprechende Package aufnehmen sollen und diese zumindest bei den neu geschriebenen Modulen verwenden.

14.4 ERLÄUTERUNGEN

- Sie sollen diese Aufgabe selber lösen. Sehen Sie sich den gesamten Entwurf an und überlegen Sie sich, ob Sie einzelne Funktionen einsparen können (z.B. doppelt vorhandene Funktionen).

14.5 AUSFÜHRUNGEN:

- Aktualisieren Sie das Package showdisplay. Führen Sie eigene „Grundtypen“ (wahlweise subtypes oder types) ein, von denen Sie alle anderen notwendigen (Unter-) Typen ableiten.
- Tragen Sie später alle benötigten Konstanten in das Package ein (mit dem entsprechenden neuen Typen!). Z.B. können die Belichtungszeitwerte als Konstanten im Package abgelegt werden, oder auch die festen Werte bei Vergleichen (Signal /= Wert)
- Optimieren Sie den Entwurf. Hierzu ein paar Anregungen:
 - Schauen Sie sich die zwei Zähler in timer_exposure und timer_transport an. Kann daraus eventuell ein Zähler gemacht werden?
 - Wie ist Ihr Zähler realisiert? Versuchen Sie eine Variante zu finden, bei der ein Incrementer (stetiges aufaddieren um den Wert 1) statt einem Addierer verwendet wird.
 - Haben Sie die passende Zählervariante gefunden, versuchen Sie ebenfalls die Zählergrenze ohne Verwendung von Addierern festzulegen. (Da unter Synopsys in einer Case-Anweisung kein array of integers abgeprüft werden kann, ist es notwendig hier ein ande-

res Format für die Exposure Time zu verwenden.) Ist es möglich, die Kodierung des Keypads nach „hinten“ zu verschieben?

- Um Ihre Verbesserungen durchzuführen ist es notwendig, daß Sie Ihr Design neu strukturieren. Mach Sie dies und kodieren Sie die neuen Module, die die alten ersetzen.
- Simulieren und synthetisieren Sie Ihren neuen Entwurf. Um wieviel Fläche oder Laufzeit hat sich Ihr Entwurf verbessert?

14.6 VHDL-System-Befehle

15. Übung: Realisierung in einem FPGA

15.1 ZIEL

- Der Fotoapparat soll in einem programmierbaren Baustein (Field Programmable Gate Array) realisiert werden. Verwendet wird hier ein Complex Programmable Logic Device von Altera verwendet.

15.2 BESCHREIBUNG

- Der Fotoapparat muß in eine übergeordnete Architecture eingebunden werden, welche die Auswertung der Eingabemöglichkeiten (Schalter, Dips und Tasten) und Ansteuerung der Ausgabemöglichkeiten (LEDs und Siebensegmentanzeigen) bewerkstelligt. Das komplette Design muß mit den passenden FPGA-Bibliotheken neu synthetisiert werden. Danach kann es in Form einer EDIF-Netzliste in das FPGA-Tool maxplus2 portiert werden. Hier wird es für den entsprechenden Baustein partitioniert und „gefittet“. Anschließend wird der Baustein auf dem Evaluationboard konfiguriert und kann dann getestet werden.
- Da hier kein FPGA Kurs stattfinden soll wird nur das Notwendigste erklärt. Sollte die Schaltung trotz erfolgreicher Simulation unter Synopsys nicht funktionieren, ist das Pech. Zur Behebung des Fehlers müsste man zu genau auf die FPGA Architektur und das verwendete Werkzeug eingehen. Ansonsten steht eine Online Help zu maxplus2 zur Verfügung.

15.3 ERGÄNZUNGEN

- Eine Möglichkeit, eine Schaltung unter realistischen Bedingungen zu testen, ist es einen Prototypen zu erzeugen. Beim ASIC-Design kann dies mit Hilfe eines FPGAs erfolgen. Hierzu müssen die VHDL Sourcen mit den passenden FPGA-Bibliotheken neu synthetisiert werden.
- Man hat prinzipiell die Möglichkeit, mit der VHDL Beschreibung direkt in das FPGA Tool hineinzugehen, sofern dieses einen VHDL Reader besitzt.
- Besitzt das FPGA Tool keinen VHDL Reader oder sind die Optimierungsmöglichkeiten dieses Tools nicht ausreichend, geht man oft den Umweg über ein spezielles Synthesetool wie z.B. Synopsys. Daß heißt aber, man muß die entsprechenden Eintragungen für die Synthesebibliotheken in dem Initialisierungsfile der entsprechenden Synthesesoftware ändern, so daß dem Synthesetool nun der Aufbau des FPGAs bekannt ist.
- Hat man das Design erfolgreich synthetisiert Anschließend muß das Syntheseergebnis in das FPGA-Tool portiert werden, wo es dann partitioniert und „gefittet“ werden kann. Dies erfolgt sehr oft mit dem EDIF Format.
- Nach dem Partitionieren und Fitten steht dem Designer ein sogenanntes Programmierfile zur Verfügung, mit dessen Hilfe er den Baustein (FPGA) konfigurieren kann. Zur Prototypenentwicklung verwendet man meistens SRAM basierte Bausteine, da diese öfters beschrieben werden können und nicht nur einmal, wie z.B. Antifuse basierte Bausteine. Mit einem Programmierfile können also die entsprechenden SRAM-Blöcke programmiert werden.

15.4 AUSFÜHRUNGEN:

- Binden Sie Ihre Top-Entity in die Architecture struct in der Datei photo_fpga.vhd ein.
- Deklarieren Sie die noch fehlenden Objekte.
- Wenn Sie nochmals eine Gesamtsimulation durchführen wollen, müssen Sie zuerst alle VHDL-Files neu analysieren, da ab in dieser Übung neue Bibliotheken verwendet werden.
- Führen Sie gegebenenfalls eine Simulation durch.

- Führen Sie die Synthese durch:
 - Clockattribute vergeben.
 - Eventuell andere (Optimierungs-) Attribute vergeben.
 - Den FPGA Compiler oeffnen (Menu Tools -> FPGA Compiler.
 - Design Optimization starten.
 - Hierarchieebenen aufoesen (ungroup -all -flatten im Command Window)
 - FPGA Zellen aufoesen (replace_fpga im Command Window)
 - Als EDIF Netzliste abspeichern (photo_fpga.edf)
- Starten Sie die FPGA Software MaxPlus2
 - Wählen Sie unter *File->Project->Name* die Top-Entity aus (photo_fpga.edf)
 - Wählen Sie unter *Assign->Device* den passenden Baustein aus (Siehe FPGA auf dem Evaluation Board).
 - Starten Sie den Compiler (*MAX+plus II->Compiler*)
 - Stellen Sie den EDIF-Netzlisten Reader auf Synopsys um (*Interfaces->EDIF Netlist Reader Settings*).
 - Starten Sie die Compilierung.
- Starten Sie den Floorplan Editor (*MAX+plus II*) und wechseln Sie die Ansicht auf *Layout->Current Assignments Floorplan*. Nehmen Sie folgende Zuweisungen vor (Drag&Drop): (Siehe auch Abbildung 18: Pinbelegung)

clk	73
reset	18
UNUSED_INPUTS(0-18)	(14,15, 19-25, 27-30, 48, 3, 58, 63, 67, 72)
SWITCH	16
ENABLES(0-1)	(34, 36)
SELECTORS(2-0)	(35, 37, 39)
DISPLAY(0-7)	(41-45, 40, 46, 51)
TRR	55
KEYPAD(1-10)	(69-71, 64-66, 60-62, 56)
ST	57
CNT	76
TRGO	79
EXP	82
UNUSED_SET	83

- Deaktivieren Sie die Reservierung der Pins Data0 und nWS, nRS, ... (Assign->Device->Device Options).

- Starten Sie erneut eine Compilierung, damit die neuen Pinbelegungen wirksam werden.
- Transferieren Sie das Programmierfile mit der Endung *.tff* auf den PC.
- Stellen Sie sicher, daß das Evaluationboard angeschlossen ist und starten Sie das Programm /DOS/FELXCONF.EXE.

IMGONFPGA.gifIMGOFF

Abbildung 18: Pinbelegung für den EPF8452ALC84-5

- Testen Sie das Design.

15.5 VHDL-System-Befehle