Versuch 2: Kombinatorische Logik

Einführung

In diesem Versuch wird die Entwicklunsumgebung Altera MAX+plus II auf dem PC und das PLD-Board SB2 eingeführt. Beide werden auch in allen folgenden Versuchen verwendet. Der Versuch besteht deshalb aus zwei Teilen:

1 MAX+plusII und Simple-Logic. 2 Hex zu 7-Segment Decoder.

Im ersten Teil geht es darum, die PLD – **Entwicklungsumgebung** und das Board mit einem einfachen Beispiel **kennen zu lernen**. Im zweiten Teil wird eine sehr oft eingesetzte Schaltung zur Anzeige einer 4-bit Hexadezimalzahl (die Dezimalzahlen 0 – 9 und die Buchstaben A – F) auf einer 7-Segment Leuchtdioden-Anzeige entwickelt. Dieser **7-Segment Decoder** wird als Baublock in den folgenden Versuchen weiter verwendet. Über den eingesetzten PLD Altera EPM3064 müssen Sie nichts wissen, einige Informationen finden sich im Anhang.

Das Lösungsblatt zum Versuch erhalten Sie am Praktikumsnachmittag.

1 MAX+plusII und Simple-Logic

Die PLD-Entwicklungsumgebung MAX+plusII (im Folgenden mit MAX+ bezeichnet) setzt in mehreren Schritten die Beschreibung der gewünschten Schaltung in Programmierdaten für die PLDs um. Dabei benötigt der Ingenieur oder die Ingenieurin keine Kenntnisse über den inneren Aufbau der PLDs. Erst bei speziellen Problemen, wie grosse Komplexität oder hohe Geschwindigkeit kann mit gezielten Eingriffen ein optimales Resultat erhalten werden. Dazu sind dann allerdings sehr detaillierte Kenntnisse notwendig. Im ersten Semester können wir es uns noch problemlos leisten, nicht alles kennen zu müssen, sollen oder wollen.

Im Praktikum wird nur ein kleiner Teil der von MAX+ gebotenen Möglichkeiten benützt. Die Bedienung und Anwendung des Programms ist sehr einfach und intuitiv, es hat eine sehr lange Entwicklungszeit hinter sich und arbeitet ausserordentlich stabil.

Für die Beschreibung der Schaltungen benützen wir vor allem den komfortablen **Grafikeditor** über den direkt das **Schaltschema** eingegeben werden kann. Wo dies einfacher ist, werden wir auch Teile der Schaltung mit **AHDL**, der Altera Hardware Definition Language beschreiben und damit direkt **logische Gleichungen** oder **Wahrheitstabellen** vorgeben. Die Sprache selbst müssen wir dazu nicht kennen, die Anwendung lernen wir sofort aus Vorlagen.

> Das auf den PCs installierte Programm MAX+plusII – Baseline steht für nicht kommerzielle Anwendungen kostenlos zur Verfügung. Es kann von Altera.com herunter geladen werden (ca. 50Mbytes). Achtung: nicht die Student Edition, diese enthält keinen Grafik Editor!). Für den Betrieb ist eine an den PC gekoppelte Lizenz notwendig (Variante HD ID-Nummer benützen). Die Lizenz muss von Altera verlangt werden und wird umgehend per Email zugestellt. Die Anleitung für das Vorgehen findet man auf den Altera Seiten.

Simple-Logic Schaltung

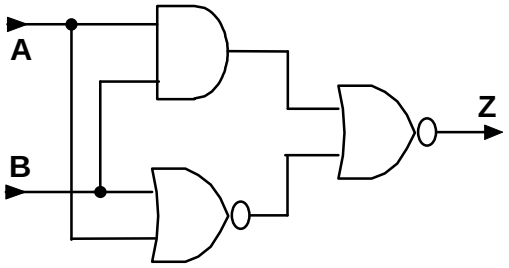
Wir wollen jetzt ein ganz einfaches Beispiel, die kombinatorische Schaltung "Simple-Logic" mit MAX+ definieren, compilieren, in den PLD programmieren und experimentell auf dem Board die Funktion testen.

Gegeben ist die logische Gleichung:

Z = ! ((A ? B) ? ! (A ? B))

(! ist das AHDL Inversionszeichen, ? das Mathematik Symbol für die UND, ? für die ODER Verknüpfung.)

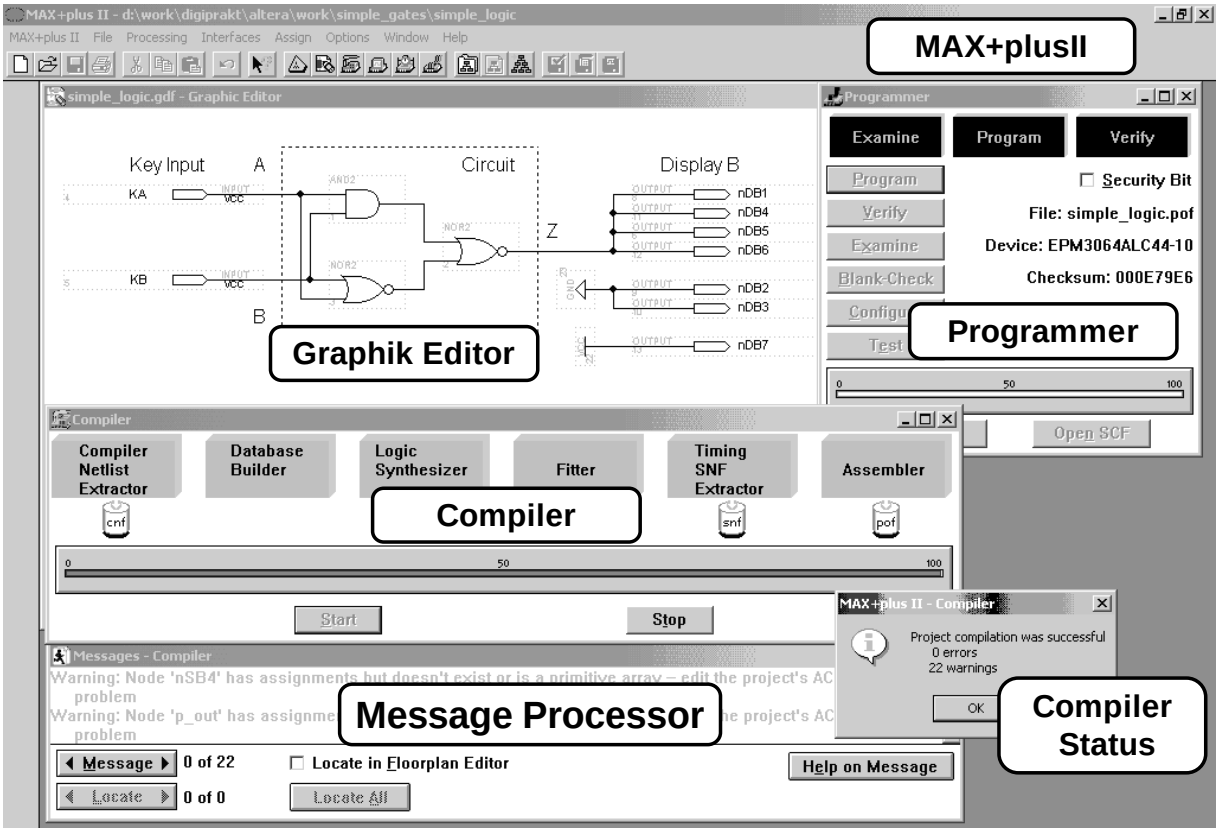
Daraus folgt das nebenstehende Schaltschema:



Schaltschema Simple-Logic

Lösung mit MAX+plusII:

Wir lösen die Aufgabe schrittweise. Am Schluss werden wir alle im folgenden Bild sichtbaren Fenster auf dem Bildschirm und die funktionierende "Simple-Logic" im PLD programmiert haben.



Bitte **jeden Aufgabenpunkt zuerst ganz durchlesen** bevor Sie mit der Bearbeitung beginnen. Damit verpassen Sie eventuell wichtige Hinweise die nicht gleich am Anfang stehen sicher nicht.

Achtung: Auf Designfiles nur über das **Pyramidenmenu** zugreifen, nicht mit Windows-Explorer, dann hat man sicher die gleichen Files wie der Compiler..

1. Auf dem PC einloggen mit:

   Username = **digiprakt** Passwort = **student** (case sensitiv!)

   Sofern nicht schon vorhanden, die folgende Verzeichnisstruktur einrichten:

   C:\DigiPrakt\V2\SimLog (Simple-Logic)

   \7SegDec (7-Segment Decoder)

2. Von der Praktikumsseite (www.ife.ee.ethz.ch/~zinniker/digiprakt) das Versuchsfile V2_MAX.exe ins Verzeichnis DigiPrakt laden und dort ausführen (self extracting zip File).

   Das Textfile SB2_Pin-Lock-List.txt nach SimLog und 7SegDec kopieren, Die restlichen files nach 7SegDec verschieben.

3. MAX+plusII starten (Icon auf dem Desktop).

4. Ein neues File für die Eingabe des Schaltschemas öffnen:

   File > New > GraphicEditorFile / .gdf

   Neben .gdf = Graphic Design Files, existieren auch .tdf = Text Design Files.

5. File ins vorbereitete Verzeichnis abspeichern:

   File > Save As > C:\Digiprakt\V2\SimLog\Simple_Logic.gdf

6. Den Projektnamen auf den Namen des Grafikfiles setzen:

   File >Project > Set Project to Current File

   Der Projektname erscheint im oberen Rahmen des Programmfensters.

7. Die benötigten Schaltsymbole in die Zeichenfläche des Grafikeditors holen:

   Rechtsclick ins Editor Fenster > Enter Symbol

   Die Symbol Librairy ....\prim öffnen (Doppelclick)

   Symbol im SymbolFiles Fenster auswählen (Doppelclick), in der Zeichnungsebene plazieren.

   Wir benötigen je ein AND und NOR Tor mit 2 Eingängen (and2, nor2) sowie je ein Eingangs- und Ausgangssymbol (input, output). Mehrfach gebrauchte Symbole werden wir einfach kopieren.

   Im Begrenzungsrahmen jedes Symbols erscheint eine Zahl. Dies ist eine fortlaufende Nummerierung, um die wir uns nicht kümmern müssen.

   Die Symbolbibliothek prim (primitives) enthält alle einfachen Symbole: Tore, Ein- und Ausgangsverbindungen zu PLD-Pins oder anderen Schaltungsteilen, GND (= 0V, Vss, 0, L), Vcc (= Vdd, 1, H).

   Für Masse und Speisung sind verschiedene synonyme Bezeichnungen üblich die sich aus der historischen Entwicklung der Logik-ICs ergaben:

   Vss (für GND) bedeutet, dass mit diesem Anschluss Source-Elektroden der n-Kanal FETs verbunden sind, nicht nur eine sondern viele, - deshalb die 2 s.

   Vdd (für + Speisespannung) bedeutet, dass die Drain-Elektroden der n-Kanal FETs von diesem Anschluss gespeist werden (in der Frühzeit baute man Logik nur mit n-Kanal FETs).

   Vcc (für + Speisespannung) stammt aus der Zeit der Bipolartransistoren und bedeutet, dass deren Collectoren (npn Transistor) von diesem Anschluss gespeist werden.

Je nach Vorliebe wird diese oder jene Bezeichnung verwendet. In MAX+ gibt es nur GND und Vcc, in den Versuchsanleitungen finden sich wohl alle gemischt.

8. Mehrfach benötigte Symbole in der Zeichnungsebene kopieren:

   Symbol selektieren (Linksclick darauf, es wird rot eingerahmt)

   Mit gedrückter Ctrl – Taste drag and drop (im Symbolrahmen erscheint ein + Zeichen zur Anzeige des Kopiermodus).

   Mehrere Ein- und Ausgangssymbole je mit Vorteil untereinander anordnen (die folgende Beschriftung wird dadurch einfacher).

9. **Ein- Ausgänge beschriften**. Die Eingangssignale A und B werden mit den Drucktasten KA und KB auf dem Board erzeugt. Beide Signale sind nicht invertiert (positiv true), das heisst 1 bei gedrückter Taste), Ausgang mit Z bezeichnen:

   DoppelClick auf den Pinnamen im obersten Eingangssymbol. Den bisherigen selektierten Namen mit der neuen Bezeichnung (KA) überschreiben, abschliessen mit Return. Es wird automatisch der Name des darunter liegenden Eingangssymbols selektiert, diesen auf KB ändern. Ausgangssymbol in gleicher Art mit Z beschriften.

10. **Schaltung verdrahten**:

    Mit dem Cursor auf den zu verbindenden Symbolanschluss fahren, in der Nähe des Anschlusses verwandelt sich der Pfeilcursor in ein Kreuz, jetzt mit gedrückter linker Taste die Leitung ziehen.

11. Das soeben erstellte **Schema auf Fehler prüfen**:

    File > Project > Save & Check

    Der Compiler startet und der Netlist Extractor findet jeden strukturellen Fehler. Ist alles korrekt, wird im Statusfenster *0 errors, 0 warnings* angezeigt.

    Status- und Compilerfenster schliessen.

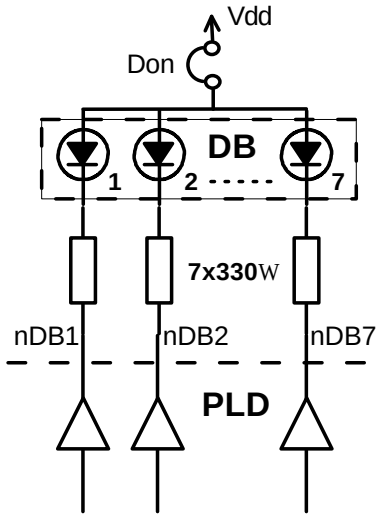
    Bestehen Fehler oder Warnungen, so zeigt der Message Processor Fehlermeldungen (rot) und Warnungen (blau) an. Die Meldungen können mit (< Message >) durchgegangen werden und ihr Ursprung im Designfile mit (Locate) angezeigt werden. Fehler müssen unbedingt behoben werden, Warnungen sind Hinweise auf mögliche logische Fehler und können bestehen bleiben. Sie können sowohl lebensrettend wie auch nervtötend sein.

    Achtung: Vor Save&Check immer kontrollieren ob das Projekt auf das zu prüfende Designfile gesetzt ist. (Files > Project > Set Project to Current File). Bei der Arbeit mit mehreren Design-Files vergisst man dies nur zu oft und wundert sich dann über unerwartete Ergebnisse.

Anzeige des Simple-Logic Ausgangssignals

Der Wert des Signales Z soll auf einer der beiden 7-Segment-Leuchtdioden-Anzeigen angezeigt werden. Alle Segmente beider Anzeigen sind auf dem Print je an einem Pin des PLD angeschlossen; (fast) jeder PLD Pin kann als Ausgang oder Eingang verwendet werden, - es sind I/O Pins. Die Schaltung (siehe Bild) ist so ausgelegt, dass ein Segment leuchtet, wenn das Ausgangssignal 0 (L) ist (negativ true). Die Widerstände bestimmen den Strom der durch die Leuchtdioden fliesst. Dies entspricht der üblichen Anordnung, da ein CMOS-Ausgang leichter einen Strom im 0-Zustand nach GND ableiten als im 1 Zustand von Vdd liefern kann.

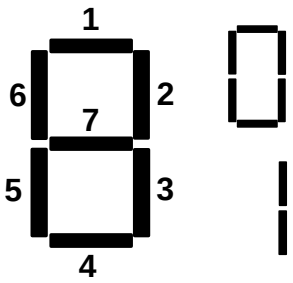
Der nach GND schaltende n-Kanal FET benötigt für den gleichen Strom nur halb soviel Fläche wie der nach Vdd schaltende p-Kanal FET (die Elektronen im n-Kanal haben eine doppelt so grosse Beweglichkeit wie die Löcher im p-Kanal).



Das Ausgangssignal Z kann nur die beiden Werte 0 oder 1 annehmen, dadurch kann die Anzeige ohne Decoder (siehe 2.Teil der Aufgabe) direkt angesteuert werden (die Bezeichnung der Segmente und die Darstellung von 0 und 1 zeigt das nebensetehende Bild): die Segmente 2 und 3 leuchten immer, 1, 4, 5, 6 nur bei Signal 0.

12. Ausgangs Symbol 6 mal untereinander kopieren (Ctrl Drag&Drop), es ergeben sich total 7 Ausgänge für je ein Segment der Anzeige.

13. GND und Vcc Symbole auf die Zeichenfläche holen.



14. Ausgänge verdrahten:

    Oberste 4 mit dem Simple-Logic Ausgang verbinden, sie werden anschliessend den Segmenten 1, 4, 5 und 6 zugeordnet).

    Nächste 2 mit GND verbinden (immer leuchtende Segmente 2 und 3)

    Das unterste mit Vcc verbinden (Segment 7, immer dunkel).

15. Ausgänge (7-Segment Pins) beschriften:

    Im obersten Ausgangssymbol Doppelclick auf den Pinnamen. Auf nDB1 (Display B, Segment 1, negativ true) ändern, mit Return beenden.

    Die folgenden 3 Ausgänge mit nDB4, nDB5, nDB6 (alle mit Logikausgang verbunden) bezeichnen.

    Die nächsten zwei mit nDB2 und nDB3 (beide sind auf GND verdrahtet).

    Den untersten mit nDB7 (ist auf Vcc verdrahtet).

    Es wird dringend empfohlen, für aktiv L (invertierte) Signale immer den Präfix n (negativ true) zu verwenden.

16. Jetzt können noch zur Information die Ein-und Ausgänge der Simple-Logic entsprechend dem Schema am Anfang der Aufgabe mit A, B und Z beschriftet werden:

    Linksklick an einem freien Platz in der Zeichnungsebene und Text eingeben, mit Escape abschliessen selektiert den Text, er kann jetzt einfach an die gewünschte Position verschoben werden.

    Achtung: Informative Beschriftungen (Kommentare) dürfen keine Leitung berühren, sonst werden sie als Signalnamen für diese interpretiert!

17. Das fertige Schaltschema speichern und auf Fehler überprüfen (falls nötig Fehler korrigieren):

    File > Project > Save&Check

Pin Locking

Alle Signale sind auf dem Board fest auf IC-Pins verdrahtet. Die entsprechende Information, das Pin-Locking, muss jetzt noch dem Compiler vorgegeben werden.

Jeder PLD-Compiler ist darauf aus, die Zuordnung der logisch definierten Ein-und Ausgangssignale zu den physikalischen Pins des Bausteins selbst so zuzuordnen wie es ihm am besten passt, in der Meinung, dass der Anwender dann den Print danach auslegt. Dies ist natürlich völlig unbrauchbar, wenn der Print, wie in unserem Fall, bereits besteht. Deshalb müssen zu allen benützten Pins Signale fest zugeordnet werden. Dies bezeichnet man als Pin-Locking. (Komplexe Schaltungen, die einen PLD maximal ausnützen, können Compiler oft erst im IC unterbringen, wenn kein Pin-Locking vorgegeben wird.)

Das acf File:

Zu jedem Projekt erzeugt MAX+ ein .acf, Assignment & Configuration File. In diesem Textfile wird das Pin-Locking festgelegt. Die physikalische Zuordnung aller Signale auf dem Board ist im Textfile SB2_Pin-Lock_List.txt vorgegeben. Es ist im Dokument SB2 Board ausgedruckt (liegt am Arbeitsplatz auf).

Im acf File wird auch ein Chipname (Name des .pof Programmierfiles für den Programmer) und der PLD Baustein vorgegeben. Beide sind in SB2_Pin-Lock_List.txt enthalten (Als Argumente der Schlüsselwörter CHIP und DEVICE).

18. Pin-Locking durchführen:

    Die Pin-Lock-Liste aus SB2_Pin-Lock_List.txt (alles von und inklusive CHIP bis END;) in Simple_Logic.acf direkt am Anfang nach dem Kommentarblock (mit -- beginnende Zeilen) kopieren (Notepad oder Wordpad Texteditor verwenden).

19. Chip Namen definieren:

    Als Chipnamen (Argument des Schlüsselwortes CHIP) im acf File den Projektnamen einsetzen (diesen wird das Programmierfile erhalten). Das bearbeitete acf File speichern und Editor unbedingt schliessen.

20. Schaltung compilieren:

    MAX+plusII (ganz links in der Menüleiste) > Compiler

    Jetzt wird hoffentlich keine Fehlermeldung erscheinen, sicher jedoch eine ganz Menge Warnungen:

    Jeder gelockte aber in der Schaltung nicht verwendete Pin ergibt eine Warnung. Um dies zu vermeiden, könnten im acf File alle nicht verwendeten Pins aus der Liste entfernt werden, tun wir aber nicht. (Nur auskommentierte Pins werden beim nächsten Compilerrun gelöscht).

    Jeder Ausgangspin der Schaltung der seinen Zustand nicht ändern kann führt zu einer Warnung. In unserem Fall sind dies die fest auf 0 oder 1 verdrahteten Segmente 2, 3 und 7. Diese Warnungen sind nicht zu vermeiden.

Programmierung des PLDs

Schlussendlich bleibt nur noch das PLD zu programmieren. Dazu muss zuerst das Board angeschlossen werden.

21. Am Netzgerät eine Ausgangsspannung von ca 6-7V einstellen, Strombegrenzung in Mittelstellung (das Board benötigt maximal, alle Segmente leuchten, bis zu 150mA). Board mit dem Speisekabel am Netzgerät anschliessen, Spannung ein.

22. Board über die Nabelschnur (25pol Kabel) am parallelen Druckerport (LPT1) des PC anschliessen.

23. PLD programmieren:

    MAX+plusII > Programmer > Program

    Im Programmer Fenster muss das richtige Programmierfile (.pof) angezeigt sein.

    Nach erfolgreicher Programmierung muss auf dem Board in der Anzegie B 0 oder 1 stehen, die Anzeige A ist nicht definiert, es werden alle Segmente leuchten.

    Bleiben die Anzeigen dunkel, kontrollieren ob der Kurzschlusstecker (Jumper) *DA DB on* links oberhalb der Taste KA gesteckt ist.

Funktion der Schaltung auf dem Board austesten

24. Wahrheitstabelle aufstellen:

    Die beiden Tasten KA und KB in allen 4 möglichen Kombinationen drücken und jeweils das Ausgangssignal notieren. ➔ **B**

25. Was für eine Funktion implementiert die Schaltung Simple-Logic? ➔ **B**

Diejenigen die alles ganz genau wissen wollen, möchten jetzt wohl noch die Verbindung zwischen Board und PC auftrennen um sicher zu sein, dass der PLD tatsächlich die Funktion der Schaltung selbständig ausführt. Um die Steckverbindung zu schonen, sollten Sie jedoch darauf verzichten und mir glauben, es ist tatsächlich so.

2 7-Segment Decoder

Die 7-Segment Anzeige ist das Standardbauteil zur Anzeige numerischer Werte (Digitalvoltmeter, Radiowecker, Eieruhr, Waschmaschine, Hometrainer...). In der Digitaltechnik werden neben den Ziffern 0-9 noch die Buchstaben A-F verwendet um eine Hexadezimalzahl (4 bits = 16 Werte) anzuzeigen, auch das ist mit der 7-Segment Anzeige möglich. Das folgende Bild zeigt die Bezeichnung der 7 Segmente, wie üblich mit den Buchstabenf a-g, und MAX+ gerecht mit den Zahlen 1-7 (siehe Abschnitt *Bezeichnung der Signale* im Anhang). Die übliche Darstellung der 16 Zeichen ist angefügt.

Vorbereitung auf den Praktikumsnachmittag

Die folgenden Aufgaben sind vor dem Praktikumsnachmittag zu lösen.

26. Ergänzen Sie die Wahrheitstabelle des 7-Segment Decoders auf dem Arbeitsblatt im Anhang mit den noch fehlenden Segmenten D2, D3 und D5.

27. Bestimmen Sie mit Hilfe der Karnaughtafeln minimisierte disjunktive logischen Gleichungen für die Segmente D3 und D5 invertiert (also für nD3 und nD5). Als Beispiel ist die KT für das Segment 2 angefügt. ➔ **B**

28. Fakultativ: Warum wird in der vorausgehenden Aufgabe mit den invertierten (aktiv L oder negativ true) Signalen gearbeitet? (Es gibt zwei Gründe). ➔ **B**

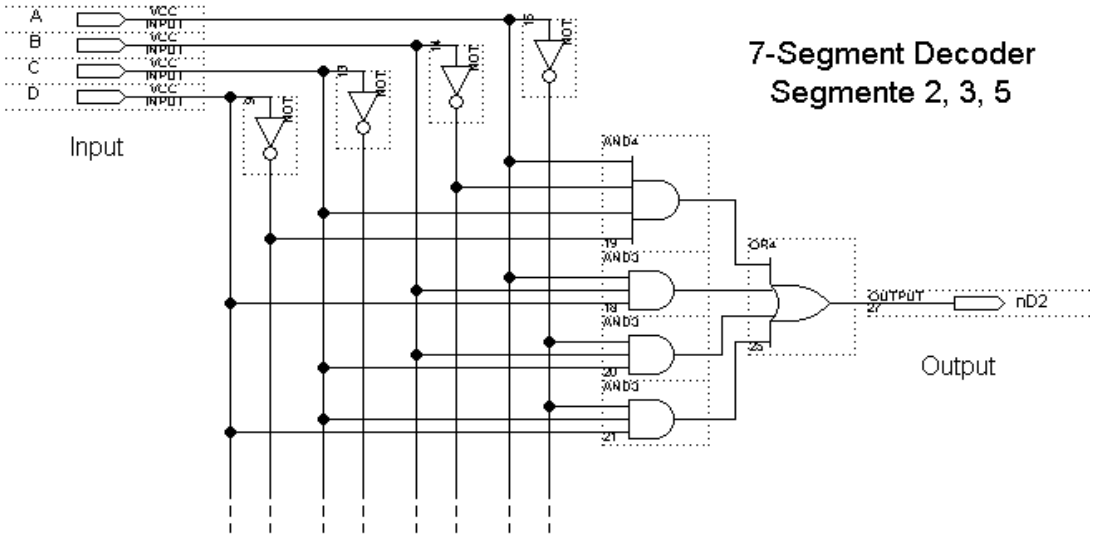
29. Fakultativ: Wie kann die Schaltung noch ein wenig vereinfacht werden? (Minimisierte Gleichungen für die Segmente nD2 und nD3 miteinander vergleichen). ➔ **B**

Am Praktikumsnachmittag zu lösende Aufgaben

Die folgenden Aufgaben sind am Praktikumsnachmittag mit Hilfe der MAX+plusII auf dem PC und dem Experimentierboard SB2 zu lösen.

30. Erstellen Sie mit dem Grafikeditor das Schaltschema für die Segmente 3 und 5 nach den minimisierten Gleichungen Ihrer Vorbereitung und erzeugen Sie für die weitere Verwendung dafür ein Makrosymbol:

    MAX+plusII starten (falls nicht mehr geöffnet). Das vorbereitete File 7seg235.gdf öffnen. Darin ist bereits eine sinnvolle Struktur vorbereitet und die Logik für das Segment 2 eingezeichnet. Sie brauchen nur noch die Schaltung für die Segmente 3 und 5 einzutragen:



Vorbereitetes Aufgabenfile 7seg235.gdf

Wenn man die Struktur der Schaltung nicht geschickt wählt, so wird das Schaltschema rasch völlig chaotisch. Der Versuch soll schliesslich keine Übung im Schemazeichen werden.

Das File zum aktuellen Projekt machen:
File > Project > SetProjectToCurrentFile.

Das Schema speichern und auf Fehler prüfen evt. korrigieren:
File > Project > Save&Check (oder einfacher Ctrl K).

Für die weitere Verwendung davon ein Makro Symbol generieren
File > CreateDefaultSymbol.

31. Erstellen Sie mit dem Grafikeditor das Schaltschema für den vollständigen Decoder und erzeugen Sie auch davon wieder für die weitere Verwendung dafür ein Makrosymbol: (Saubere Darstellung gibt Bonuspunkte!) ➔ **B**

Ein neues Grafik Design File anlegen und zum aktuellen Projekt machen:
File > New > GraphicEditorFile
File > SaveAs > / 7seg.gdf / (zum Benennen sofort speichern)
File > Project > SetProjectToCurrentFile.

Die Makrosymbole der Teildecoder für die Segmente 2, 3, 5 sowie 1, 4, 6, 7 in das Editorfenster holen:
Rechtsklick ins Fenster > EnterSymbol > / 7seg235 /
Analog Symbol 7seg1467 holen
Beide Symbole etwa in der Mitte untereinander positinieren.

Input und Output Symbole ins Fenster holen (input links, output rechts):

Input Symbol 3 mal darunter kopieren (Ctrl drag&drop), Output 6 mal

Schaltung verdrahten:
Eingänge parallelschalten (Verbindungsleitungen ziehen).
Die 4 Input Symbole mit den 4 parallelgeschalteten Eingängen der Teildecoder verdrahten.
Die 7 Output Symbole mit den 7 Ausgängen der Teildecoder verbinden.

Input und Output Symbole mit den Bezeichnungen der Ein- und Ausgänge der Teildecoder Makrosymbole beschriften.(jeweils mit dem obersten beginnen, das Weiterschalten auf die unteren erfolgt automatisch):
Inputs A, B, C, D Outputs nD1 – nD7.

Das Schema speichern und auf Fehler prüfen evt. korrigieren:

Für die weitere Verwendung davon ein Makro Symbol generieren

32. Jetzt wollen wir endlich den Decoder in den den PLD programmieren und auf dem Board ausprobieren! Die 4 Eingangssignale sollen mit dem dem 4-Bit Schalter SB erzeugt werden und die 7 Ausgangssignalen sollen die Anzeige DB angesteuern.

Ein neues Grafik Design File 7seg_sb-db.gdf anlegen und zum aktuellen Projekt machen.:

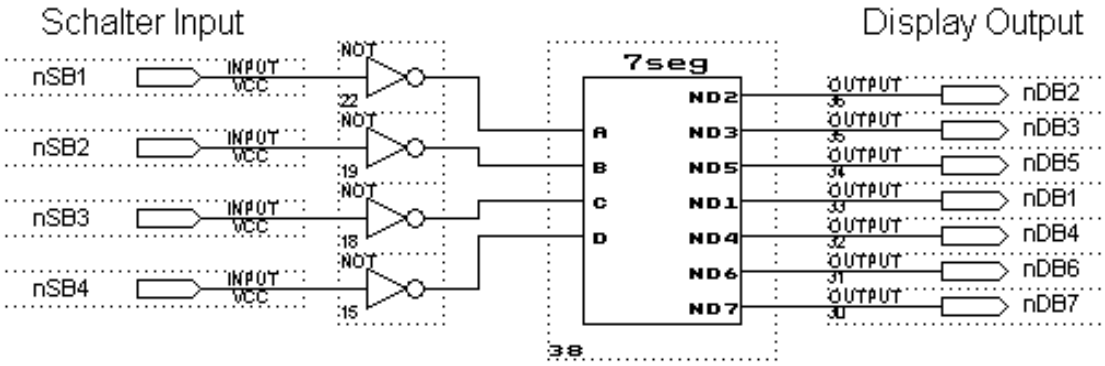
Makrosymbol des vollständigen Decoders, 4 Input und 7 Output Symbole in das Editorfenster holen.

Der Schalter SB erzeugt 4 invertierte (aktiv L) Signale. Zur Ansteuerung des Decoders mit normalen (aktiv H) Eingängen, müssen diese invertiert werden:
Invertersymbole (not) ins Editorfenster holen und 3 mal kopieren.

Die 4 Eingänge über die Inverter, die 7 Ausgänge direkt mit dem Makrosymbol verbinden.

Ein- und Ausgänge mit den Signalnamen des Experimentierboards SB2 beschriften (jeweils mit dem obersten beginnen, das Weiterschalten auf die unteren erfolgt automatisch):
Inputs nSB1 – nSB4, Outputs nDB1 – nDB7.

Ihr Schema sollte jetzt etwa so aussehen:



Das Schema speichern, auf Fehler prüfen und diese evt. korrigieren.

Pin Locking durchführen

Schaltung compilieren, Fehler korrigieren, ins PLD programmieren

33. Decoder auf dem SB2-Board austesten: ➔ **B**

    Auf der rechten Anzeige (DB) soll der Wert der am 4-Bit Hex-Codeschalter SB eingestellten Hexadezimalzahl angezeigt werden.
    Alle 16 Werte durchgehen und Anzeige kontrollieren. Zum einstellen des Schalters den Spannungsprüfer-Schraubenzieher verwenden!

34. Wenn der Decoder Fehler aufweist, so müssen diese lokalisiert und korrigirt werden. Dazu wird durch Doppelclick ins Makrosymbol dieses geöffnet und die Korrektur kann direkt im entsprechenden Design File vorgenommen werden..

Der Compiler erzeugt immer ein Report-Text-File (.rpt). Dieses gibt Auskunft über die durchgeführte Bearbeitung des Projektes und die Ausnützung der Resourcen im PLD. Wer mehr wissen will schaue sich dieses an, wir können im Moment nicht weiter darauf eingehen.

Wir haben einen hierarchisch gegliederten Design durchgeführt. Dies hat den grossen Vorteil, von unten nach oben unnötige Details hinter Makrosymbolen zu verbergen. Durch Doppelklick in ein Makrosymbol kann jederzeit in der Hierarchie um eine Stufe zurück gegangen werden.

Im Compiler wird zuerst die ganze Hierarchie wieder in eine einstufige (flache) Beschreibung aufgelöst. Damit ist in jedem Fall eine maximale Optimierung und beste Anpassung auf den PLD möglich. Bei jeder Compilation wird alles neu compiliert. Dadurch können keine Fehler durch nicht aktuelle precompilierte Module entstehen. Als Nachteil dauert die Compilation länger. Die gesamte Hierarchie eines Projektes kann mit MAX+plusII > HierarchyDisplay angezeigt werden.

Der Compiler bildet nicht unsere Bschreibung sondern die beschriebene Funktion in den PLD ab. Es ist deshalb nicht nötig, die Beschreibung optimieren zu wollen, sie soll vielmehr möglichst verständlich (auch noch nach Jahren!) und einfach zu erstellen sein. Wer macht sich schon gerne mehr Arbeit als nötig, - ausser unfreiwillig die Studenten im Digitalpraktikum (sorry auch die Studentinnen). Wenn er dies könnte, so würde der Compiler herzhaft über unsere Bemühungen lachen, denn bei unseren einfachen Problemen kann er es allemal mindestens genau so gut. Erst später, bei komplexen Aufgaben, etwa in einer Diplomarbeit, müsste er anerkennen, dass wir es doch besser wissen. Da Sie jetzt bis hierher gelesen haben, will ich Sie nicht mehr weiter beschwatzen, fahren wir lieber wieder mit ernsthafteren Dingen fort.

Zum Schluss

Gratulation, Sie haben den ersten Versuch mit MAX+ und dem PLD Board erfolgreich abgeschlossen! Ich hoffe, es hat Ihnen gefallen und Sie haben dabei viel gelernt.

Den 7-Segment Decoder werden wir in den folgenden Versuchen weiter verwenden. Deshalb sollten Sie Ihre heutige Arbeit sicherstellen. Dazu gehen Sie wir folgt vor:

File 7seg.gdf öffnen (z.B. durch Doppelklick in Symbol 7seg), nicht 7seg_SB-DB!.

7seg als aktuelles Projekt definieren: File > Project > SetProjectToCurrentFile.

Projekt archivieren: File > Project > Archive

Jetzt sind alle für die weitere Verwendung wichtigen Files ins Archivverzeichnis kopiert. Dieses können Sie jetzt als Ganzes einfach kopieren:

- Auf Ihren persönlichen eigenen studentischen Speicherplatz.
- Auf einen persönlichen USB-Memorystick.
- Auf dem PC in ein persönliches Verzeichnis (dabei müssen Sie allerdings weiterhin am gleichen Arbeitsplatz arbeiten).
- Auf eine von den Betreuern erhältliche Floppydisc. (Sie dürfen ruhig lachen, aber diese Urmethode ist oft immer noch die sicherste und einfachste!)

Wenn Sie nichts von alledem machen, so können Sie für die weiteren Versuche dann immer noch die auf der Praktikumsseite bereitgestellten Lösungsfiles herunterladen. Darin ist der 7-Segment Decoder vollständig in AHDL direkt durch die Wahrheitstabelle definiert (ja, so einfach geht es auch).

Bevor Sie den PC herunterfahren und den Arbeitsplatz spannunslos schalten bitte noch den Inhalt des heutigen Versuchsverzeichnisses löschen (Ihre Mitstudierenden sollen schliesslich am nächsten Praktikumsnachmittag auch noch etwas tun).

Weitere Informationen

Weitere Informationen, auch zu den Themen dieses Versuches, finden Sie im Dokument SB2_Info das am Arbeitsplatz aufliegt und von der Praktikumsseite herunter geladen werden kann:

Bedienungselemente und Bauteile

Schaltschema

SB2_Pin-Lock-List.txt:

PLD Signale und Anschlüsse

PLD Signal und Pin Tabellen

PLD Altera EPM3064

Dieser Versuch ist obligatorisch und gibt maximal 34 Punkte.

Wichtige Punkte der MAX+plusII Anwendung

Signalnamen:

Maximal **32 Alphnumerische Zeichen und Underline** zulässig.

Signalnamen mit Grossbuchstaben beginnen.

Für **negativ true** (aktiv L, invertiert) Signale **Präfix n** verwenden (z.B nSB1).

Namen von einzelnen Signalen nicht mit einer Zahl abschliessen.

Signale in einer Gruppe mit Zahl am Schluss kennzeichnen, z.B. nDB1, nDB2 .... nDB7 und nicht nDBa, nDBb, nDBc.... Dadurch wird es möglich, diese gesamthaft als Bus einzusetzen, z.B. nDB[1..7] (brauchen wir später).

Graphik Editor, *.gdf Files

Objekte Selektieren (Zum Bearbeiten, Kopieren, Löschen, Verschieben):

Einzelnes Objekt: Linksclick ins Objekt, roter Begrenzungsrahmen erscheint

Mehrere Objekte: Rahmen um die Objekte aufziehen (mit gedrückter Linkstaste ab freiem Punkt in der Zeichnungsebene einziehen, nur vollständig innerhalb des Rahmens liegende Objekte werden selektiert, Leitungen nur bis zum Rahmen.

Kopieren: Ctrl Drag&Drop, natürlich auch Ctrl C, Ctrl V oder Edit Menü.

Verschieben: Verbindungen nicht trennen, Options > Rubberbanding = on
Verbindungen trennen: Options > Rubberbanding = off

Altera Hardward Description Language, AHDL *.tdf Files

Logische Operatoren:
NOT (!), AND (&), NAND (!&), OR (#), NOR (!#), XOR ($), and XNOR (!$).

Subdesign Name zwingend gleich Filenamen, genau 1 Subdesign pro File.

Kommentare in % eingeschlossen, innerhalb einer Zeile und über mehrere Zeilen möglich. Beispiel % Kommentar %. Der MAX+ Text-Editor hebt Kommentare farblich hervor. Praktisch: Er lässt sich mit der Insert-Taste von Insert auf Overwrite Modus umschalten (Anzeige in unterem Fensterrand). Kolonnen selektieren mit Ctrl – drag (nicht Alt – drag wie in Word!)

Vorlagen für die Beschreibung mit logischen Gleichungen und Wahrheitstabellen verwenden, wir beschränken uns darauf.

Graphik und AHDL Design Files

Speichern und Fehlercheck:
File zum aktuellen Projekt machen, File > Project > SetProjectToCurrentFile,
File > Project > Save&Check
Fehler müssen behoben werden, Warnungen nicht.

Compilieren: MAX+plusII > Compiler

Programmieren: MAX+plusII > Programmer (pof File des aktuellen Projektes)

Hierarchie anzeigen:MAX+plusII > HierarchyDisplay

Archivieren: File > Project > Archive
Archiviert alles in der Hierarchie nach unten ab dem aktuellen Projekt File in einem Archivverzeichnis.

Arbeitsblatt Vorbereitung 7-Segment Decoder

Wahrheitstabelle und Karanughtafeln

D	C	B	A	D1	D2	D3	D4	D5	D6	D7	
0	0	0	0	1	**1**		1		1	0	**0**
0	0	0	1	0	**1**		0		0	0	**1**
0	0	1	0	1	**1**		1		0	1	**2**
0	0	1	1	1	**1**		1		0	1	**3**
0	1	0	0	0	**1**		0		1	1	**4**
0	1	0	1	1	**0**		1		1	1	**5**
0	1	1	0	1	**0**		1		1	1	**6**
0	1	1	1	1	**1**		0		0	0	**7**
1	0	0	0	1	**1**		1		1	1	**8**
1	0	0	1	1	**1**		1		1	1	**9**
1	0	1	0	1	**1**		0		1	1	**A**
1	0	1	1	0	**0**		1		1	1	**B**
1	1	0	0	1	**0**		1		1	0	**C**
1	1	0	1	0	**1**		1		0	1	**D**
1	1	1	0	1	**0**		1		1	1	**E**
1	1	1	1	1	**0**		0		1	1	**F**

nD2

C \ B A	00	01	11	10
00				
01			1	
11	1		1	1
10		1		1

nD2 = !D C !B A + D B A + C B !A + D C !A

nD3

C \ B A	00	01	11	10
00				
01				
11				
10				

C \ B A	00	01	11	10
00				
01				
11				
10				

C \ B A	00	01	11	10
00				
01				
11				
10				

nD5

C \ B A	00	01	11	10
00				
01				
11				
10				