Opdrachten practicum

INLDIG

DE HAAGSE HOGESCHOOL

J.E.J op den Brouw
De Haagse Hogeschool
Opleiding Elektrotechniek
18 augustus 2020
J.E.J.opdenBrouw@hhs.nl

Inleiding

Het practicum is zodanig van opzet en moeilijkheidsgraad dat iedere student die voor aanvang van het practicum zijn/haar opdrachten goed voorbereidt en altijd op het practicum aanwezig is een voldoende kan halen.

Het is de bedoeling dat je op dit practicum eenvoudige digitale systemen met behulp van poorten leert ontwerpen, een competentie die onmisbaar is voor elke elektrotechnische ingenieur. Je kunt dit alleen leren door zelfstandig alle opdrachten uit te voeren. Het kopiëren van schema's of schemadelen van het internet of van collega-studenten is niet leerzaam! Het boek en de slides bieden voldoende voorbeelden die je ter inspiratie kunt gebruiken. Gebruik in je ontwerp nooit (deel-)schema's die je zelf niet begrijpt.

Als je niet weet hoe je moet beginnen of als je een bepaalde fout niet kunt vinden of als je niet weet hoe je een bepaalde actie met poorten beschrijft of ...vraag het dan aan de docent! Van vragen kun je veel leren. Je kunt je vraag ook altijd mailen naar J.E.J.opdenBrouw@hhs.nl.

Natuurlijk kun je ook dingen vragen aan medestudenten. Als jou iets wordt gevraagd geef je medestudent dan niet simpel een oplossing voor zijn/haar probleem maar help hem/haar om zelf het probleem op te lossen!

Thuis voorbereiden

De practicumopdrachten kunnen thuis uitgewerkt worden. Van Quartus is een zogenaamde Web Edition-versie[1] beschikbaar. Deze heeft geen licentie nodig. Er zijn versies voor Windows en Linux. Er is geen versie voor OS-X. **Voor alle versies van Windows en Linux wordt aangeraden om Quartus versie 13.0sp1 te installeren.** Lagere versies geven driver-problemen bij Windows 8. De projecten zijn uitwisselbaar met de op school geïnstalleerde versie 13.0sp1.

De software is te downloaden via https://dl.altera.com/13.0sp1/?edition=web#tabs-1. Je moet wel een account aanmaken; dat is gratis. Let erop dat de volledig geïnstalleerde versies bij elkaar zo'n 11 GB aan harddiskruimte in beslag nemen. Vergeet niet dat de download zelf zo'n 4,5 GB in beslag neemt. Dat geldt ook voor het uitpakken, dat is ook 4,5 GB groot.

Op het practicum wordt gebruik gemaakt van het DE0-bordje[2]. In een later project wordt ook gebruik gemaakt van het DE2-70-bordje. **De laatste versie die beide borden ondersteunt is 13.0sp1, hogere versies kunnen niet gebruikt worden.**

Om de opdrachten goed uit te voeren, moet een zogenaamd script geïnstalleerd worden. Hoe je dat moet doen, staat beschreven in bijlage C (voor Linux) en D (voor Windows) van de tutorial.

[1] Vanaf versie 13.0 is ModelSim geïntegreerd in het installiepakket van Quartus

[2] Zie https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=364

Huisregels

Tijdens het gebruik van de practicumruimte en -apparatuur gelden de volgende huisregels:

- Niet eten en/of drinken in het laboratorium.
- De apparatuur wordt alleen gebruikt voor practicum-doeleinden.
- Zet de apparatuur in de juiste staat terug.
- Praat rustig, niet schreeuwen.
- Telefoons graag op stil zetten.

Practicumregels

Voor het praticum geldt een aantal regels:

1. Aanwezigheid tijdens het practicum is verplicht.
2. Bij ziekte e.d. zo spoedig mogelijk (liefst voorafgaand aan het practicum) contact opnemen met de docent (liefst per mail) om inhaalmogelijkheden te bespreken.
3. Een student die de practicumlessen niet heeft bijgewoond (of ingehaald) kan derhalve geen voldoende halen.
4. Iedere practicumopdracht dient, op de daarvoor tijdens de les aangegeven datum, te worden afgerond.
5. Te laat ingeleverde opdrachten zijn automatisch onvoldoende en kunnen niet worden ingehaald!
6. Een ingeleverde opdracht die niet met een voldoende wordt beoordeeld kan (een week later) worden aangevuld. Als je schema niet helemaal perfect blijkt te zijn is dat dus geen probleem. Als je pas begint met schema's opstellen is het normaal dat alles niet meteen perfect is. Natuurlijk moet je wel proberen om je schema zo goed te maken als je zelf kunt.
7. Het laten nakijken van een schema die je niet zelf bedacht en beschreven hebt, wordt beschouwd als fraude (net zoals het spieken bij tentamens). Als je een schema die je niet zelf hebt gemaakt probeert in te leveren krijg je meteen een onvoldoende voor het gehele practicum. De fraude wordt ook gemeld bij de examencommissie. Fraude kan leiden tot een schorsing.
8. De deadline is week 7 van het kwartaal, de student krijgt dan zijn eindbeoordeling: voldoende of onvoldoende.
9. Een onvoldoende als eindresultaat kan (in overleg met de practicumdocent) worden herkanst in week 10. Zie hiervoor de modulewijzer.

Practicumkaart

Je ontvangt tijdens de eerste practicumbijeenkomst een practicumkaart. Op deze kaart worden aanwezigheid en opdrachten die goedgekeurd zijn afgetekend. Je moet je kaart voor elke bijeenkomst meenemen.

BlackBoard

Om de practicumopdrachten te kunnen maken, moet je aangemeld zijn bij BlackBoard[3] van De Haagse Hogeschool. Je moet inloggen met de gegevens die je van de IT-dienst hebt gehad.

Op BlackBoard worden ook mededelingen gepost. Hou BlackBoard dus altijd in de gaten.

Website

Alle practicumopdrachten en bestanden die je nodig hebt kan je ook vinden op de website https://ds.opdenbrouw.nl.

Algemene leerdoelen

De algemene leerdoelen zijn:

- Leren omgaan met de pakketten Quartus en ModelSim.
- Bediening DE0-experimenteerbord.

Deze leerdoelen gelden voor elke opdracht.

Afrondmoment opdrachten

Een practicumopdracht moet uiterlijk één week later worden afgerond dan de week waarin de opdracht moet worden uitgevoerd.

Resultaat practicum

Het practicum wordt met een voldoende beoordeeld als:

- de student alle practicumsessies aanwezig is geweest.
- alle opdracht correct zijn afgerond.

[3] Zie https://blackboard.hhs.nl/

Opdracht week 1 – tutorial

Inleiding
Tijdens het eerste practicum wordt een tutorial doorlopen om de software te leren kennen. Er zijn twee programma's:

> Quartus – het ontwikkelpakket voor digitale schakelingen met componenten van Altera.
>
> ModelSim – een veelgebruikt simulatiepakket voor digitale schakelingen.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het leren van alle 4-bits binaire codecombinaties.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Log in op BlackBoard en meld je aan voor de Course INLDIG.

b) In de course INLDIG vind je alle bestanden die nodig zijn om het practicum uit te voeren. Open de tutorial. Dit is een PDF-bestand dat je kan vinden onder Documents/Practicum.

c) Voer de tutorial uit vanaf hoofdstuk 4.

d) Laat de docent het geheel controleren.

Opmerkingen
Tijdens het doorlopen van de tutorial zul je af en toe foutmeldingen krijgen waarvan de beschrijving erg cryptisch is. Vraag de docent om hulp.

De hoofdstukken 1 t/m 3 kan je thuis rustig nalezen.

Opdracht week 2 – poorten

Inleiding
De opdracht voor deze week is het ontwerpen van een schakeling met vier ingangen en twee uitgangen. Het geheel kan gezien worden als twee schakelingen (dus totaal twee uitgangen) en beide schakelingen gebruiken dezelfde vier ingangen. De schakelingen moeten het volgende doen:

- De eerste schakeling moet een 1 afgeven als MEER dan twee ingangen 1 zijn, anders moet de schakeling een 0 afgeven.
- De tweede schakeling moet een 1 afgeven als er PRECIES twee ingangen 1 zijn, anders moet de schakeling een 0 afgeven.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Opstellen van een waarheidstabel voor de functies.
- Ontwerpen (synthese) van een schakeling met poorten vanuit een waarheidstabel.
- Invoeren van het ontworpen schema.
- Simuleren van het ingevoerde schema.
- Testen van het ingevoerde schema.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Stel één waarheidstabel op voor de bovenstaande schakelingen met daarin de ingangen en de twee uitgangen. Laat de tabel controleren door de docent.

b) Bepaal bij elk van de twee schakelingen hoeveel combinaties van de ingangssignalen de uitgangswaarden een 1 zijn. (Je kan dat uitrekenen!)

c) Haal van BlackBoard het zip-bestand `poorten.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `poorten`.

d) Voer het schema voor de schakelingen in. Hiervoor moet je een bestand aanmaken met de naam `poorten.bdf`. *De beide schakelingen moet je in één schemabestand onderbrengen* (zie ook Opmerkingen).

e) Simuleer het schema met ModelSim. Beide schakelingen moeten ingevoerd zijn voordat gesimuleerd kan worden.

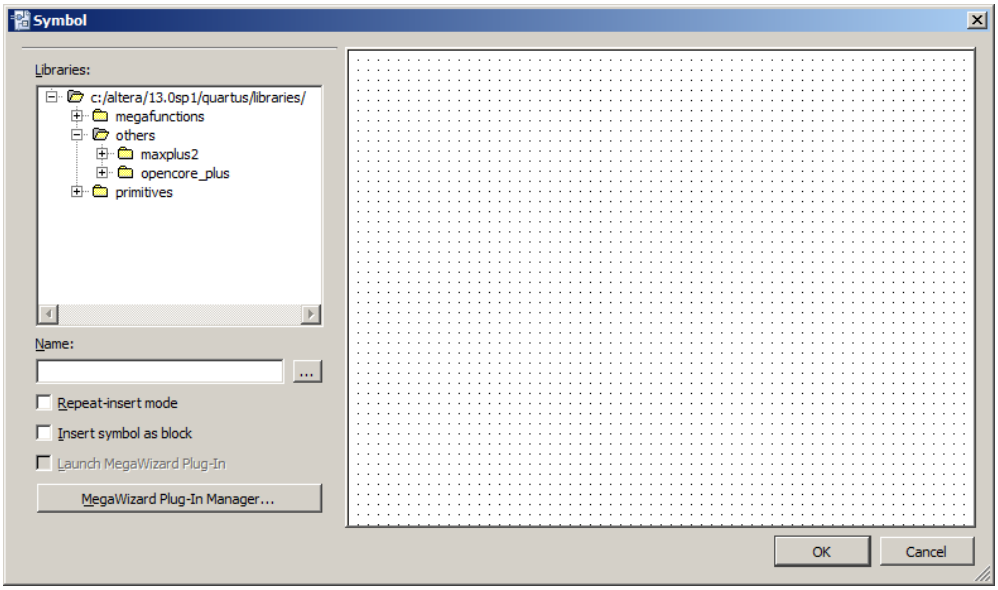
f) Compileer het schema en laadt het in het DE0-experimenteerbord.

g) Test het ontwerp op het DE0-experimenteerbord.

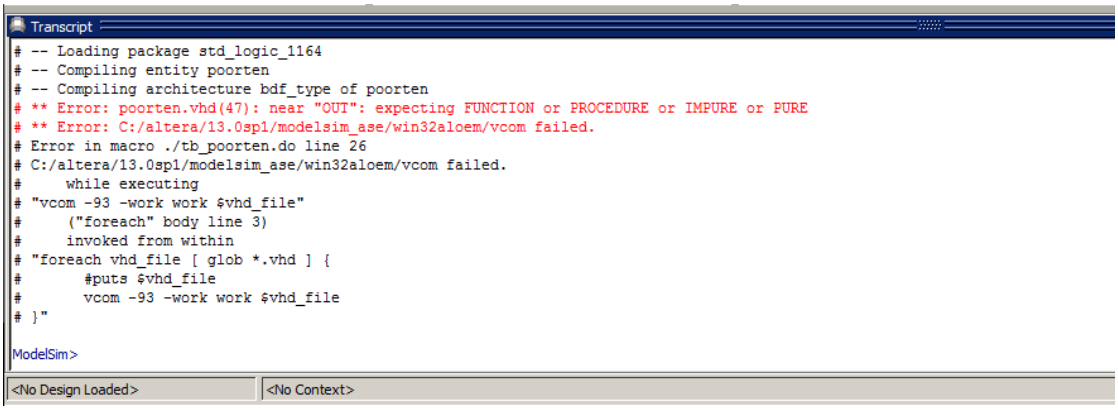
Opmerkingen
Voor beide schakelingen moet je voor de ingangen weer de namen `SW3`, `SW2`, `SW1` en `SW0` gebruiken, als uitgangen moet je weer de namen `LEDG0` en `LEDG1` gebruiken. Gebruik in geen geval de namen `SW7`, `SW6`, `SW5` en `SW4`.

Let op: elke schakelaar (inputs) kan je maar één keer in je schema invoeren.

Let op: je mag **geen** poorten gebruiken uit de `maxplus2`-bibliotheek! Die worden niet ondersteund op de gebruikte chip. Zie figuur 1. Je krijgt bij simulatie dan een foutmelding, zie figuur 2.



Figuur 1: *Overzicht beschikbare componentenbibliotheek.*

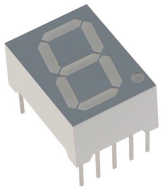


Figuur 2: *Foutmelding in ModelSim na gebruik van de `maxplus2`-bibliotheek.*

Opdracht week 3 – 7-segment display

Inleiding

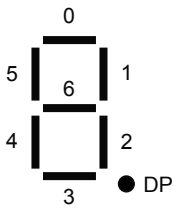
In de digitale techniek worden 7-segment displays gebruikt voor het afbeelden van decimale getallen. De display bestaat uit zeven segmenten (meestal uitgevoerd als leds) en een punt (ook uitgevoerd als led). De zeven segmenten zijn zo geconstrueerd dat ze samen de tien cijfers kunnen weergeven. Het is zelfs mogelijk een aantal letters weer te geven. Zie figuur 3 voor een praktische uitvoering.



Figuur 3: *Praktische uitvoering 7-segment display.*

Om tien verschillende cijfers weer te kunnen geven zijn vier bits nodig. De bitcombinaties 0000 t/m 1001 worden hiervoor gebruikt. Dit komt precies overeen met één BCD-cijfer. De bitcombinaties 1010 t/m 1111 worden niet gebruikt, maar zouden gebruikt kunnen worden om hexadecimale cijfers weer te geven.

Om cijfers weer te geven moeten de segmenten worden aangestuurd. Elk segment kan aan of uit staan, dus is een digitale schakeling te ontwerpen die dit doet. Dit is te realiseren met een schakeling die zeven (acht, als de punt wordt meegerekend) uitgangen heeft. Hiervoor wordt aan de segmenten een letter toegekend, zie figuur 4. Vervolgens kunnen de tien cijfers worden geconstrueerd, zie figuur 5.



Figuur 4: *De segmenten van de display.*

Figuur 5: *De tien cijfers op de display.*

De opdracht voor deze week is het invoeren van de schakeling en het beproeven van de 7-segment display.

Leerdoelen

De leerdoelen van deze opdracht zijn:

- Ontwerpen van een schakeling met poorten d.m.v. mintermen.
- Begrip binaire codering, binair tellen.
- Invoeren van het ontworpen schema.

- Simuleren van het ingevoerde schema.
- Testen van het ingevoerde schema.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Haal van BlackBoard het zip-bestand `seven_seg.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `seven_seg`.

b) Maak het schema voor het 7-segment display compleet. Merk op dat er al een bestand met de `seven_seg.bdf` aangemaakt is.

c) Simuleer het schema voor het 7-segment display met ModelSim. Gebruik het bestand `tb_seven_seg.do`, dit moet aangepast worden voor een goede werking.

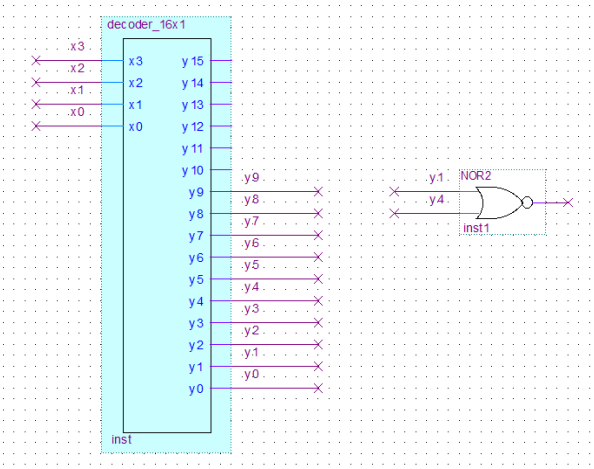
d) Synthetiseer en implementeer het schema en laadt het in het experimenteerbord.

e) Test het ontwerp op een DE0-bord.

Opmerkingen
Een deel van het schema is al ingevoerd, onder andere het deel voor het genereren van de functies van `y0` t/m `y9`. Dit wordt gedaan door de deelschakeling `decoder_16x1`. Deze heeft vier ingangen en zestien uitgangen.

De y-uitgangen vertegenwoordigen de bijbehorende *mintermen*. Zo is `y0` logisch 1 als de ingangen `x3`, `x2`, `x1` en `x0` allemaal logisch 0 zijn (minterm 0). Alle andere y-uitgangen zijn dan logisch 0. Er is dus op elk moment slechts één y-uitgang logisch 1.

Het invoeren van de complete schakeling zal nog wel problemen geven, er moeten veel poorten worden ingevoerd en nog veel meer verbindingen (wires). Dat levert een onwerkbaar schema op. Het is mogelijk om wires te koppelen zonder ze fysiek te verbinden door ze dezelfde naam te geven. In figuur 6 is te zien dat de signalen `y1` van de decoder verbonden is met signaal `y1` van de NOR-poort. Dit geldt ook voor `y4`.



Figuur 6: *De decoder met een poortje.*

De naamgeving gaat eenvoudig: selecteer een wire, klik rechtermuisknop en klik dan op `Properties`. Onder het tabblad `General` kan je een naam invullen.

Merk op dat één kant van de wires niet verbonden zijn, dat is te zien aan het kruisje.

Let op: de leds van de display zijn laag actief, een logisch 0 zorgt ervoor dat de led gaat branden.

Voor de ingangen moeten weer de schakelaars `SW3` t/m `SW0` gebruikt worden waarbij `SW3` het meest significante bit voorstelt. De uitgangen hebben de namen `HEX0_D0` t/m `HEX0_D6` waarbij `HEX0_D0` gelijk staat aan segment A en `HEX0_D6` gelijk staat aan segment G. Zie voor een volledig plaatje bijlage B van de tutorial.

Opdracht week 4 – Binair-naar-BCD omzetting

Inleiding

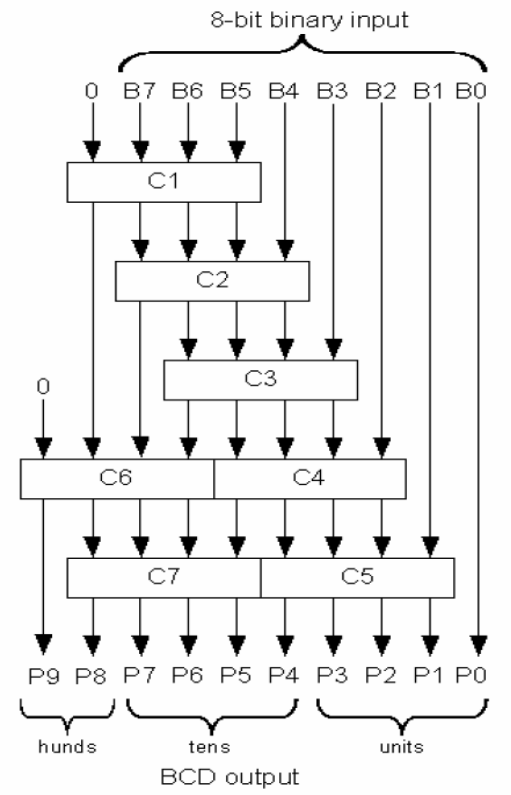
In de digitale techniek worden getallen opgeslagen in het binaire talstelsel. Rekenschakelingen zijn zo eenvoudig te ontwerpen. Helaas is het aflezen van binaire getallen voor de meeste mensen vervelend, zij hebben immers geleerd met decimale getallen te rekenen.

Het afbeelden van decimale cijfers kan eenvoudig met 7-segment displays zoals al in de opdracht van week 3 te zien is.

Als we een zuiver binair getal willen afbeelden op 7-segment displays, dan moet er eerst een omzetting plaatsvinden van het zuiver binaire getal naar het BCD-equivalent.

Het omzetten van een zuiver binair getal naar het BCD-equivalent is lastig, maar niet onmogelijk. In principe kan hiervoor een waarheidstabel worden opgezet. Dit levert echter hele grote, lastige functies op. Slimmer is om gebruik te maken van het zogenaamde *Double Dabble* algoritme[4]. Daarmee kan op een redelijk eenvoudige manier de omzetting uitgevoerd worden.

In figuur 7 is een schema te zien van een 8-bits binair getal naar een 10-bits BCD-getal.

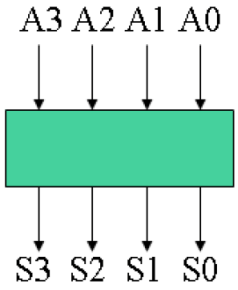


Figuur 7: *Implementie van het Double Dabble algoritme.*

Te zien is een zogenaamde structuur-schema (*structural*), er zijn geen echte digitale poorten te zien. Merk trouwens op dat de structuur erg elegant is.

[4] https://en.wikipedia.org/wiki/Double_dabble

De blokjes C1 t/m C7 in figuur 7 zijn identiek. Dit worden de zogenoemde *Add3*-modules genoemd. Het gaat nu te ver om uit te wijden over de werking van het systeem, dat is best complex. De werking van de schakeling van zo'n Add3-module is weergegeven in figuur 8.



A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Figuur 8: *De Add3-module.*

De Add3-functie laat zich ook wiskundig beschrijven als we de ingangen en uitgangen als decimaal getal zien. Het getal A bestaat uit de bits A_3, A_2, A_1 en A_0 en het getal S bestaat uit de bits S_3, S_2, S_1 en S_0. Dan is de functie van S als volgt:

$$S = \begin{cases} A & \text{voor } 0 \leq A \leq 4 \\ A+3 & \text{voor } 5 \leq A \leq 9 \end{cases}$$

Er is geen voorschrift voor de overige zes combinaties van A, zodat bij het vertalen van de getallen naar een waarheidstabel don't cares kunnen worden ingevuld.

Deze opdracht gaat over het ontwerpen en het gebruik van de Add3-module. Vanuit de waarheidstabel moeten de schakelfuncties worden gevonden zodat het schema kan worden ingevoerd.

Als de Add3-module correct werkt, kan het gebruikt worden als bouwsteen voor binair-naar-BCD-omzetter in figuur 7.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Opstellen van een waarheidstabel voor de functies.
- Gebruik van Karnaughdiagrammen voor het minimaliseren van de functies.
- Inzicht specificatie vs. realisatie.
- Ontwerpen van een schakeling met poorten.
- Invoeren van het ontworpen schema.
- Simuleren van het ingevoerde schema.
- Testen van het ingevoerde schema.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Leidt de functies af voor de variabelen S_3, S_2, S_1 en S_0 van Add3-module. Hiervoor moet voor elke uitgang een Karnaughdiagram worden opgesteld. Er moeten dus vier Karnaughdiagrammen worden getekend. In figuur 9 is te zien hoe een Karnaughdiagram voor vier variabelen wordt opgesteld. Laat de diagrammen en de gevonden functies controleren voor dat je aan opdracht b) begint. Noot: het is hier de bedoeling de Karnauhgdiagrammen uit te werken naar de standaard som-van-productenvorm.

b) Haal van BlackBoard het zip-bestand `add3.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `add3`.

c) Voer het schema voor de Add3-module in (alleen als de Karnaughdiagrammen in orde zijn bevonden). Zorg ervoor dat de namen van in- en uitgangen met hoofdletters wordt geschreven.

d) Simuleer het schema voor Add3-module met ModelSim.

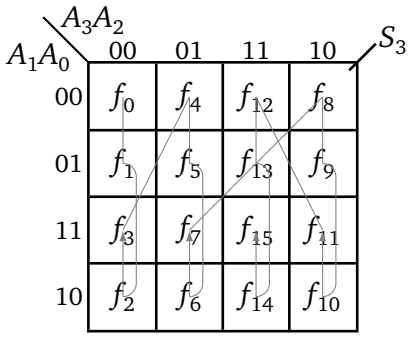
e) Compileer het schema en laadt het in het experimenteerbord.

f) Test het ontwerp op een DE0-bord.

Opmerkingen
De ingangen van de schakeling (figuur 8: A3 t/m A0) moeten gekoppeld worden aan de schakelaars SW3 t/m SW0, de uitgangen (S3 t/m S0) moeten gekoppeld worden aan de leds LEDG3 t/m LEDG0.

Op de volgende bladzijde is te zien hoe een Karnaughdiagram moet worden ingevuld.

Hieronder is het invullen van een Karnaughdigram voor vier variabelen weergegeven.



A_3	A_2	A_1	A_0	S_3
0	0	0	0	f_0
0	0	0	1	f_1
0	0	1	0	f_2
0	0	1	1	f_3
0	1	0	0	f_4
0	1	0	1	f_5
0	1	1	0	f_6
0	1	1	1	f_7
1	0	0	0	f_8
1	0	0	1	f_9
1	0	1	0	f_{10}
1	0	1	1	f_{11}
1	1	0	0	f_{12}
1	1	0	1	f_{13}
1	1	1	0	f_{14}
1	1	1	1	f_{15}

Figuur 9: *Karnaughdiagram voor vier variabelen.*

Implementatie Bin-to-BCD-omzetter
De Add3-module is nu klaar en kan gebruikt worden in het schema in figuur 7. Hiervoor is een project beschikbaar.

De volgende opdrachten moeten gedaan worden:

g) Haal van BlackBoard het zip-bestand `bin8_to_bcd10.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `bin8_to_bcd10`. Het bestand `bin8_to_bcd10.bdf` bevat het schema van de Bin-to-BCD-omzetter. Zie figuur 10.

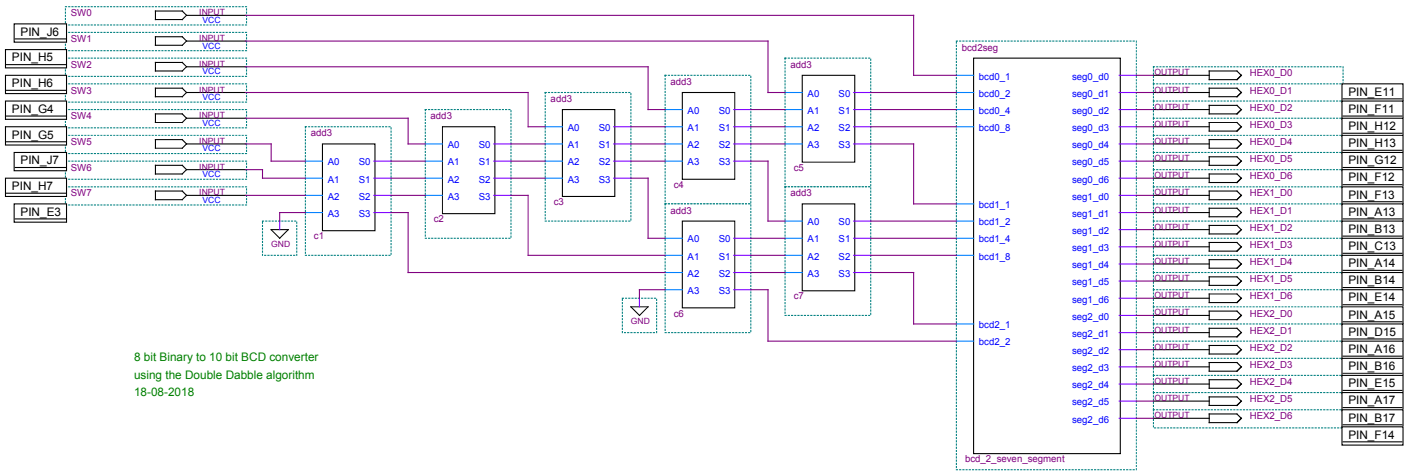
h) Kopieer het bestand `add3.bdf` uit het eerder gemaakte Add3-project naar het project `bin8_to_bcd10`. De naam van het bestand is al toegevoegd aan de lijst van projectbestanden.

i) **Verander de pinnamen in `add3.bdf`.** De ingangen SW3 t/m SW0 moeten vervangen worden door A3 t/m A0 en de uitgangen LEDG3 t/m LEDG0 moeten vervangen worden door S3 t/m S0. Zie ook figuur 8.

j) Simuleer het complete schema met ModelSim.

k) Compileer het schema en laadt het in het DE0-bord.

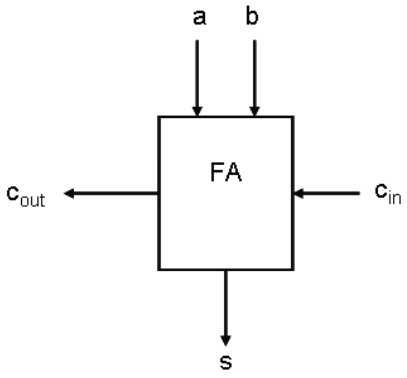
l) Test het ontwerp op een DE0-bord.



Figuur 10: *Een 8-bits binair naar 7-segmenten omzetter.*

Opdracht week 5 – 4-bits Full Adder met overflow-detectie

Inleiding
In de digitale techniek worden getallen opgeslagen in het binaire talstelsel. Rekenschakelingen zijn eenvoudig te ontwerpen. Zo kan een ontwerp worden gemaakt voor een full adder voor één bit. Zie figuur 11.



Figuur 11: *1-bit Full Adder.*

Meerdere van deze 1-bit Full Adders zijn te cascaderen tot grotere optellers, bijvoorbeeld 4-bits. Het ontwerp komt overeen met de manier waarop wij optellen, namelijk kolomsgewijs.

Deze opdracht bevat het ontwerpen en testen van zowel een 1-bit Full Adder als een 4-bits Full Adder met overflow-detectie. Eerst wordt de 1-bit Full Adder ontworpen en getest, daarna de 4-bits Full Adder (en de overflow-detectie).

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Ontwerpen van een schakeling met poorten met hiërarchieën.
- Instellen van de juiste Top Level Entity.
- Invoeren, simuleren en testen van de ontworpen schakeling.

Opdrachten
De volgende opdrachten moeten gedaan worden. Eerst moet het project ingericht worden.

a) Haal van BlackBoard het zip-bestand `full_adder.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `full_adder`.

Vervolgens moet een 1-bit Full Adder ontworpen en getest worden.

b) Maak een nieuw Block Design File aan. Voer hier het schema van een 1-bit Full Adder in. Gebruik als ingangen `a`, `b` en `cin` en als uitgangen `cout` en `s` (zie figuur 11). *Sla dit bestand op als `fa_onebit.bdf`.*

c) Simuleer het schema van de 1-bit Full Adder met ModelSim en controleer de goede werking. De bijbehorende simulatie-script heet `tb_fa_onebit.do`.

d) Genereer een symbool (Block Symbol File) van de 1-bit Full Adder. Zie tutorial.

Nu moet een 4-bits Full Adder geconstrueerd worden uit losse 1-bit Full Adders.

e) Maak een nieuw Block Design File aan. Voer hier het schema van de 4-bits Full Adder. *Sla dit bestand op als* `full_adder.bdf`. De *mappings* van de ingangen en uitgangen staan in onderstaande tabel:

```
B3  ->  SW8          A3  ->  SW4          C0  ->  SW0
B2  ->  SW7          A2  ->  SW3
B1  ->  SW6          A1  ->  SW2
B0  ->  SW5          A0  ->  SW1

S3  ->  LEDG3        C4  ->  LEDG4
S2  ->  LEDG2        V   ->  LEDG9
S1  ->  LEDG1
S0  ->  LEDG0
```

(B3 t/m B0 zijn de bits van getal B, A3 t/m A0 zijn de bits van getal A, C0 in de carry-in. S3 t/m S0 zijn de som-bits van uitkomst S, C4 is de uitgaande carry van de 4-bits Full Adder en V is de overflow-detectie).

f) Synthetiseer de 4-bits Full Adder middels `Start Analysis & Synthesis` (Ctrl-K). Hierdoor wordt een *entity* `full_adder` aangemaakt.

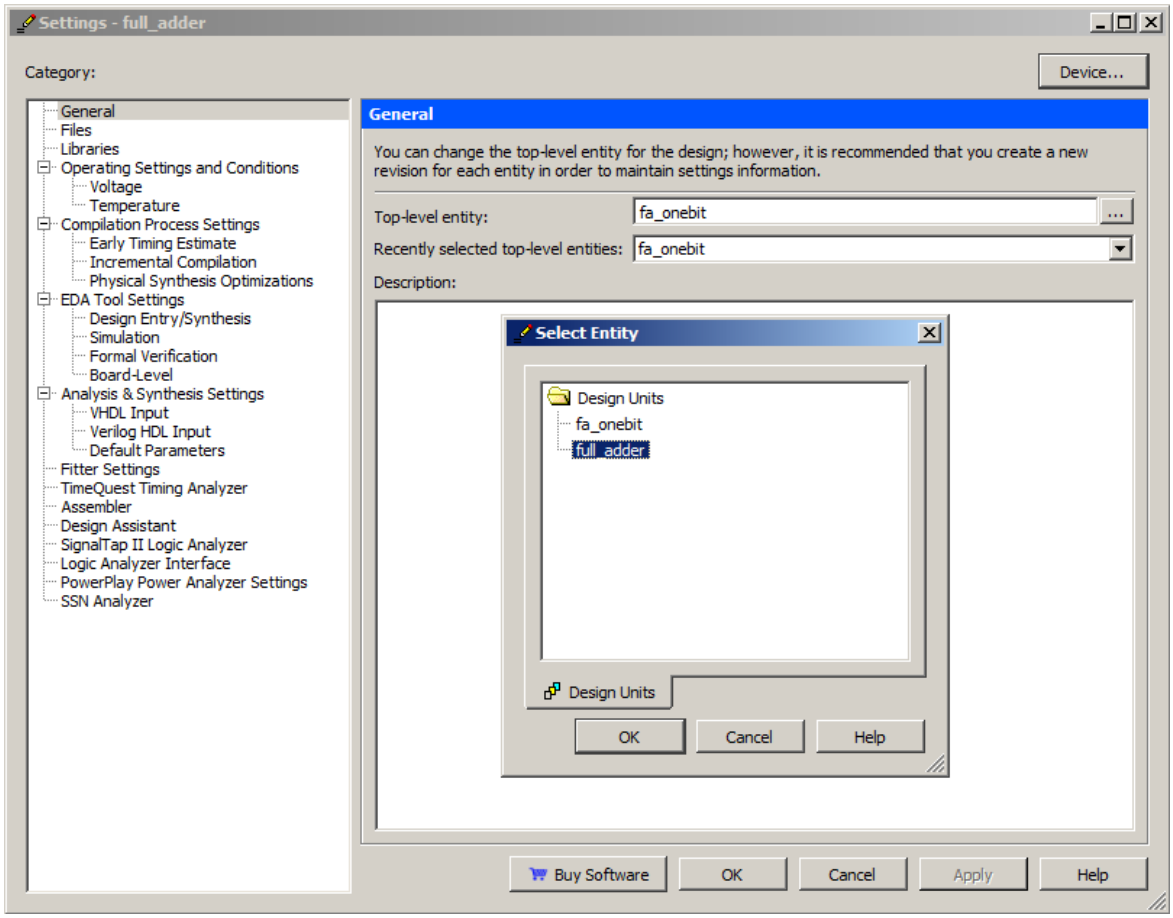
Nu moet de 4-bits Full Adder gesimuleerd worden (en later ook geïmplementeerd). Hiervoor moet eerst de juiste *top level entity* geselecteerd worden (dat is bij opdracht f) namelijk nog `fa_onebit`).

g) Selecteer in het menu `Assignments->Settings` het tabblad `General`. Selecteer vervolgens (rechts) bij de regel `Top Level Entity` de module `full_adder`. Zie figuur 12.

h) Simuleer de 4-bits Full Adder met behulp van Modelsim. De bijbehorende simulatiescript heet `tb_full_adder.do`.

i) Test de schakeling met behulp van het DE0-ontwikkelbord. Voer de volgende berekeningen uit:

- $+15 + +15$
- $-1 + -1$
- $-6 + +6$
- $+10 + -6$



Figuur 12: *Veranderen van Top Level Entity Name.*

Opdracht week 6 – 4-bits two's complement comparator

Inleiding
In de digitale techniek worden getallen opgeslagen in het binaire talstelsel. Rekenschakelingen zijn eenvoudig te ontwerpen.

Een *comparator* is een schakeling die twee binaire getallen vergelijkt en aangeeft hoe de verhoudingen liggen. Zo geeft de schakeling aan of de twee getallen gelijk zijn, of dat het ene getal groter is dan het andere getal. Natuurlijk zijn meer vergelijkingen mogelijk zoals ongelijk, groter of gelijk en kleiner of gelijk.

Het hart van de comparator is een aftrekschakeling. Met deze schakeling is mogelijk om uitspraken te doen over twee getallen:

$A-B=0$ getallen zijn gelijk

$A-B>0$ A is groter dan B

$A-B<0$ A is kleiner dan B

Een aftrekschakeling is te bouwen uit een optelschakeling, immers $A-B=A+(-B)$.

De opdracht is om een 4-bits two's complement comparator te ontwerpen op basis van de eerder ontworpen 4-bits full adder.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Ontwerpen van een rekenschakeling voor two's complement getallen.
- Ontwerpen van een schakeling voor testen of een getal groter is dan een ander getal.
- Ontwerpen van een schakeling voor testen of een getal kleiner is dan een ander getal.
- Ontwerpen van een schakeling voor testen of een getal gelijk is aan een ander getal.
- Invoeren, simuleren en testen van de ontworpen schakeling.

Opdrachten
De volgende opdrachten moeten gedaan worden. Eerst moet het project ingericht worden.

a) Haal van BlackBoard het zip-bestand `comparator.zip` binnen en pak het uit in `H:\QUARTUS\INLDIG`. Het zip-bestand bevat het Quartus-project `comparator`.

b) Ontwerp de comparator-schakeling. Het ontwerpen van een is-gelijk-schakeling is niet zo lastig, het ontwerpen van groter-dan en kleiner-dan wel. Ga met meerdere mensen aan de slag, overleg over hoe de schakeling moet worden opgebouwd. De *mappings* van de ingangen en uitgangen staan in onderstaande tabel:

```
A3  ->  SW7        B3  ->  SW3
A2  ->  SW6        B2  ->  SW2
A1  ->  SW5        B1  ->  SW1
A0  ->  SW4        B0  ->  SW0

AltB  ->  LEDG0      (A less than B)
AeqB  ->  LEDG1      (A equals B)
AgtB  ->  LEDG2      (A greater than B)
```

In het schema zijn de signalen `S3`, `S2`, `S1`, `S0` en `C` geplaatst die respectievelijk de vier sombits en de carry-out representeren. Plaats in het schema zelf de signalen `N` en `V`. De simulator toont deze in de waarheidstabel.

c) Simuleer de 4-bits two's complement comparator met behulp van ModelSim. De bijbehorende simulatiescript heet `tb_comparator.do`.

d) Test de schakeling met behulp van het DE0-ontwikkelbord.

Opmerkingen

Merk op dat er zes *relationele operatoren* zijn: $=$, $\neq$, $<$, $>$, $\leq$ en $\geq$. Je hoeft er maar twee te ontwerpen, de overige zijn af te leiden uit de eerste twee.

Bij het vergelijken van two's complement getallen komen de overflow (V) en de negative (N) flag in beeld. Stel een tabel op met alle combinaties van N en V en doe een uitspraak over wat de combinatie voorstelt. Bijvoorbeeld het feit dat V logisch 1 is wil zeggen dat het *resultaat* niet geldig is maar het zegt *wel* wat over de verhoudingen van de twee getallen.

Reken als voorbeeld maar $+7-(-7)$ en $-7-(+7)$ en $+3-(-2)$ uit.

N	V	relationele operatie
0	0	
0	1	
1	0	
1	1	