Opdrachten practicum

DIGTEC

DE HAAGSE HOGESCHOOL

J.E.J op den Brouw
De Haagse Hogeschool
Opleiding Elektrotechniek
14 juni 2022
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 Jesse op den Brouw (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 zogenoemde Lite Edition-versie beschikbaar. Deze heeft geen licentie nodig. Er is een versies voor Windows (8, 10, 11). Er is geen versie voor macOS. **Aangeraden wordt om Windows Quartus versie 20.1.1 te installeren.**

De software is te downloaden via https://fpgasoftware.intel.com/20.1.1/?edition=lite&platform=windows#tabs-1. Je moet wel een account aanmaken; dat is gratis. Let erop dat de volledig geïnstalleerde versie bij elkaar zo'n 11 GB aan harddiskruimte in beslag nemen. Vergeet niet dat de download zelf zo'n 5,9 GB in beslag neemt. Dat geldt ook voor het uitpakken, dat is ook 5,9 GB groot.

Op het practicum wordt gebruik gemaakt van het DE0-CV-bordje[1] van Terasic. Hierop is een Cyclone V FPGA gemonteerd. Daarmee zullen we digitale schakelingen ontwikkelen.

Om de opdrachten goed uit te voeren, moet een zogenoemde *Desgin Flow* geïnstalleerd worden. Hoe je dat moet doen, staat beschreven in hoofdstuk 4 van de tutorial.

[1] Zie https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=921

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
Het practicum wordt uitgevoerd op individuele basis. Het is de bedoeling dat je op dit practicum leert om digitale schakelingen te maken. De practicumopdrachten worden afgesloten door een demonstratie van het werkende systeem aan de practicum-docent gevolgd door een evaluatiegesprek(je). De docent kan dan vragen naar de manier van aanpak, bepaalde details van jullie oplossing, achterliggende theorie enz.

Ieder practicum begint met een korte uitleg met voorbeeld/demo over het onderwerp waarover de opdracht gaat. Daarna krijgt je de tijd om de opdracht zelfstandig uit te voeren en te laten aftekenen. Indien je het niet lukt om een werkend programma af te hebben binnen het practicum, kan je dit in de tussenliggende tijd worden afmaken en in het volgende practicum laten aftekenen. Met dien verstande dat de docent gewoon iedere week verder gaat met de volgende opdracht.

Participatieplicht Bij het practicum van DIGTEC-pr1 en DIGTEC-pr2 geldt een participatieplicht. Participatieplicht is een inspanningsverplichting van jou die het volgende inhoudt:

1. Het practicum wordt voorbereid door de theorie te bestuderen die hoort bij de te maken practicumopgave.
2. Aanwezigheid bij practicumbijeenkomsten;
3. Een actieve, professionele en resultaatgerichte werkhouding van jou tijdens de ingeroosterde practicumbijeenkomsten. Dit houdt in dat:
   (a) Je verwacht wordt dat je een deel van het werk kan doen zonder aanwezigheid van begeleiding. Daartoe zijn de practicumruimten opengesteld als ze niet zijn ingeroosterd;
   (b) Je wordt verwacht, indien van toepassing, vragen te stellen en dat je daarbij je eigen hypothese of handelen goed kan beschrijven;
   (c) Het niet is toegestaan om te eten en te drinken in een practicumruimte;
   (d) Er bij ziekte en overmacht zo spoedig mogelijk voorafgaand aan de practicumbijeenkomst contact wordt gezocht met de docent. Je kunt met de docent een inhaalafspraak maken om bij een andere practicumgroep het practicum in te halen.
   (e) Verwacht wordt dat de practicumopdrachten zelfstandig worden uitgevoerd.

(f) Het laten beoordelen van een programma/ontwerp dat je niet zelf hebt bedacht en geprogrammeerd wordt beschouwd als mogelijke fraude. De mogelijke fraude wordt gemeld bij de examencommissie en deze neemt verdere vervolgstappen.

Beoordeling Het practicum wordt na de laatste ingeroosterde les beoordeeld met een O of een V. Voor het behalen van een "V" dient te worden voldaan aan de volgende criteria:

1. Participatieplicht
   Je hebt je gehouden aan de hierboven genoemde participatieplicht.

2. Aftekenen
   Op het practicum kan de docent beoordelen of de student de opdracht heeft voltooid. Als de opdracht is voltooid, dan wordt deze afgetekend op de practicumkaart/excel. Alle opdrachten die op Blackboard voor het practicum zijn gegeven dienen te worden voltooid.

   Als het programma nog niet in orde is bij het laten beoordelen door de practicumdocent, dan is dat niet direct een probleem. Je kunt het programma daarna weer aanpassen aan de hand van de aanwijzingen van de practicumdocent. Je kunt vervolgens het programma weer laten beoordelen mits je nog voldoet aan het tijdschema voor het aftekenen zoals is vermeld hieronder. Je dient het programma overigens wel goed te testen voordat je het laat beoordelen.

3. Tijdschema aftekenen
   Iedere practicumopdracht dien je uiterlijk een week later dan de aangegeven week te laten beoordelen tijdens het practicum door de practicumdocent. Je kunt hier alleen van afwijken na tijdig overleg met de practicumdocent.

Herkansing
Als voldaan is aan de participatieplicht, zoals hierboven is vermeld, en het practicum desondanks niet met een voldoende is afgerond, dan is er een herkansingsmogelijkheid in week 10 voor DIGTEC-pr1 en week 19 voor DIGTEC-pr2. De herkansing bestaat uit het laten beoordelen van de opdrachten die niet zijn afgetekend. Je mag bij deze herkansing maximaal 3 opdrachten laten beoordelen/aftekenen.

BlackBoard
Om de practicumopdrachten te kunnen maken, moet je aangemeld zijn bij BlackBoard[2] 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/digtec.html.

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

Boek
Tijdens het practicum kan je het boek “Digitale techniek” gebruiken.

Algemene leerdoelen
De algemene leerdoelen zijn:

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

Deze leerdoelen gelden voor elke opdracht.

Verslag
Voor dit practicum is geen verslag noodzakelijk.

Opdracht lesweek 2 – tutorial

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

Quartus Prime Lite – het ontwikkelpakket voor digitale schakelingen met componenten van Altera.

Intel ModelSim Starter – een veelgebruikt simulatiepakket voor digitale schakelingen.

Deze programma's moeten geïnstalleerd zijn voordat de tutorial wordt doorlopen.

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 DIGTEC.

b) In de course DIGTEC 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 lesweek 3 – 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 de map `\QUARTUS\DIGTEC`. 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 schema-bestand 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-CV-experimenteerbord.

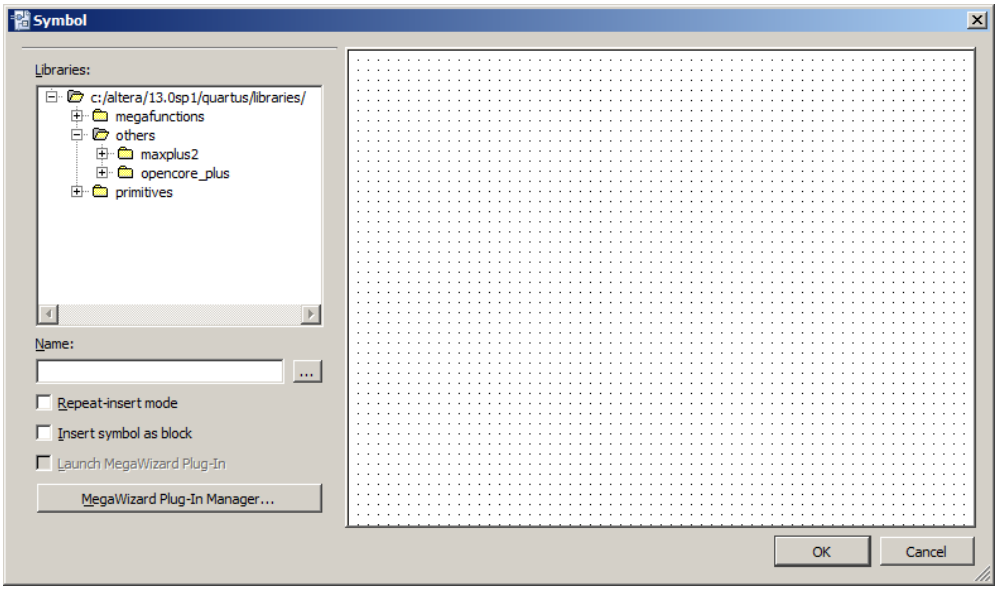
g) Test het ontwerp op het DE0-CV-experimenteerbord.

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 `LEDR0` en `LEDR1` gebruiken. Gebruik in

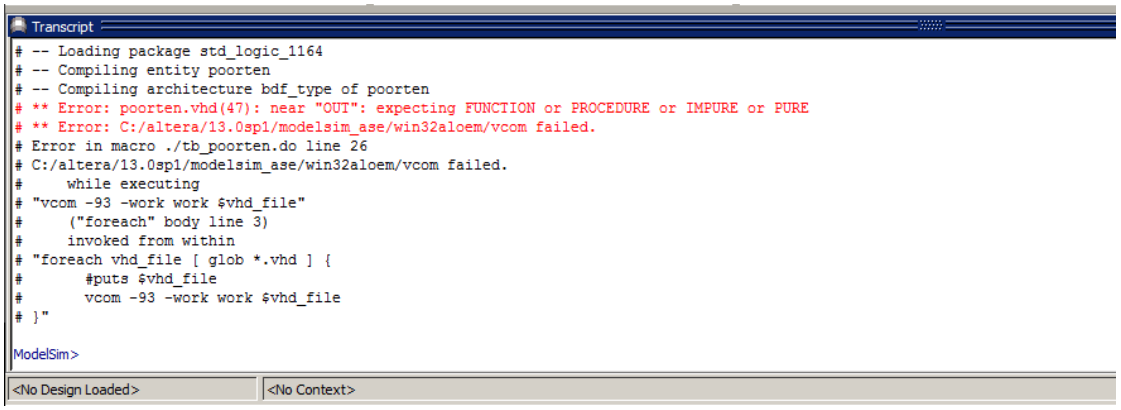
geen geval de namen SW7, SW6, SW5 en SW4.

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.*

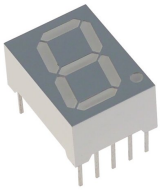


Figuur 2: *Foutmelding in ModelSim na gebruik van de `maxplus2`-bibliotheek.*

Opdracht lesweek 4 – 7-segment display

Inleiding

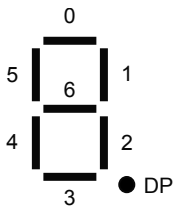
In de digitale techniek worden 7-segment displays gebruikt voor het afbeelden van decimale getallen. De display bestaat uit zeven segmenten (meestal uitgevoerd als leds) en een punt (ook uitgevoerd als led). De zeven segmenten zijn zo geconstrueerd dat ze samen de tien cijfers kunnen weergeven. Het is 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 `\QUARTUS\DIGTEC`. 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.

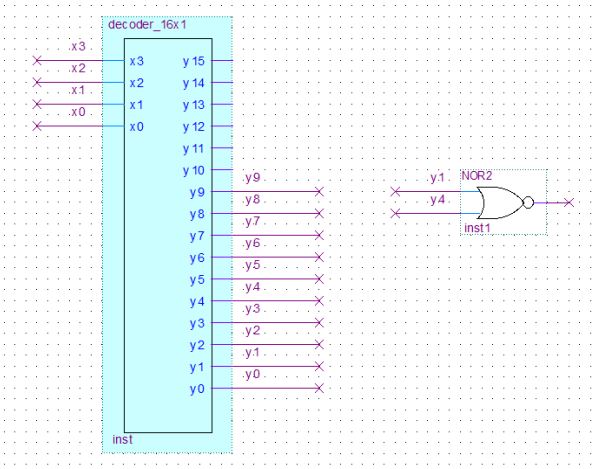
d) Synthetiseer en implementeer het schema en laadt het in het experimenteerbord.

e) Test het ontwerp op een DE0-CV-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 HEX00 t/m HEX06 waarbij HEX00 gelijk staat aan segment A en HEX06 gelijk staat aan segment G. Zie voor een volledig plaatje bijlage B van de tutorial.

Opdracht lesweek 5 – Binair-naar-BCD omzetting

Inleiding

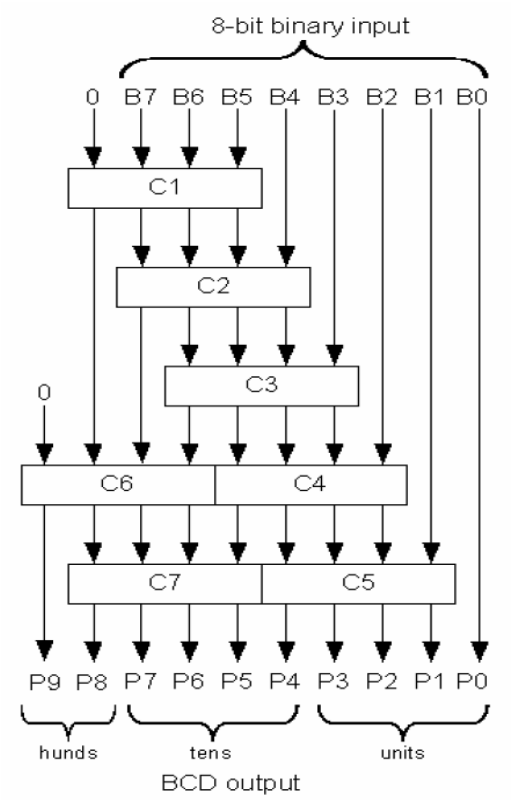
In de digitale techniek worden getallen opgeslagen in het binaire talstelsel. Rekenschakelingen zijn zo eenvoudig te ontwerpen. Helaas is het aflezen van binaire getallen voor de meeste mensen vervelend, zij hebben immers geleerd met decimale getallen te rekenen.

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[3]. 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.

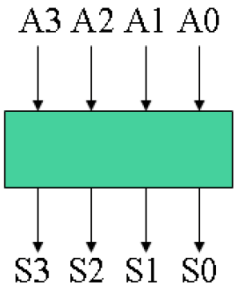


Figuur 7: *Implementie van het Double Dabble algoritme.*

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

[3] 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 `\QUARTUS\DIGTEC`. 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.

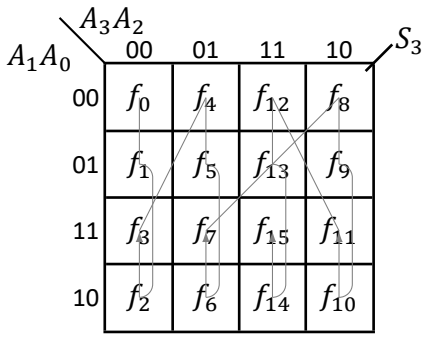
e) Compileer het schema en laadt het in het experimenteerbord.

f) Test het ontwerp op een DE0-CV-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 LEDR3 t/m LEDR0.

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 `\QUARTUS\DIGTEC`. 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.

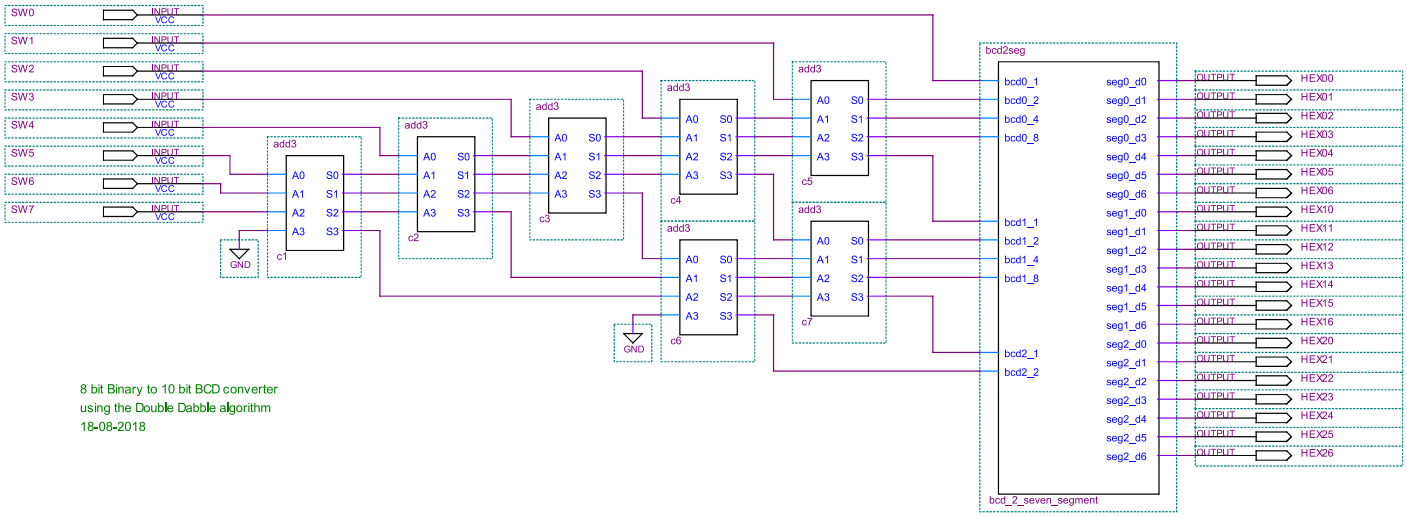
h) Kopieer het bestand `add3.bdf` uit het eerder gemaakte Add3-project naar het project `bin8_to_bcd10`. De naam van het bestand is al toegevoegd aan de lijst van projectbestanden.

i) **Verander de pinnamen in `add3.bdf`.** De ingangen SW3 t/m SW0 moeten vervangen worden door A3 t/m A0 en de uitgangen LEDR3 t/m LEDR0 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-CV-bord.

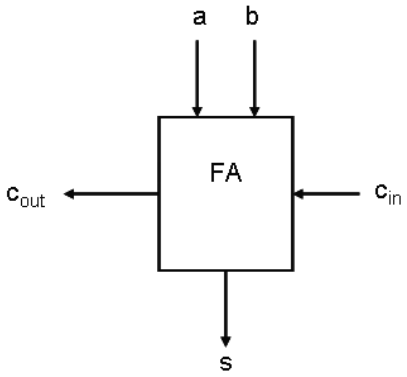
l) Test het ontwerp op een DE0-CV-bord.



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

Opdracht lesweek 6 – 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.
- Leren optellen met unsigned en two's complement getallen.

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 `\QUARTUS\DIGTEC`. 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  ->  LEDR3      C4  ->  LEDR4
S2  ->  LEDR2      V   ->  LEDR9
S1  ->  LEDR1
S0  ->  LEDR0
```

(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.

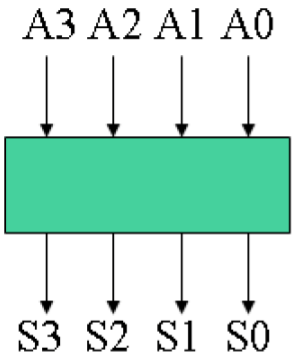
Nu moet de 4-bits Full Adder gesimuleerd worden (en later ook geïmplementeerd). Hiervoor moet eerst de juiste *top level entity* geselecteerd worden (dat is bij opdracht f) namelijk nog `fa_onebit`).

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 simulatie-script heet `tb_full_adder.do`.

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

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



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 12: *Veranderen van Top Level Entity Name.*

Opdracht lesweek 7 – 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 `\QUARTUS\DIGTEC`. 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  ->  LEDR0      (A less than B)
AeqB  ->  LEDR1      (A equals B)
AgtB  ->  LEDR2      (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-CV-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	

Opdracht lesweek 11 – tutorial

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

Quartus Prime Lite – het ontwikkelpakket voor digitale schakelingen met componenten van Altera.

Intel ModelSim Starter – een veelgebruikt simulatiepakket voor digitale schakelingen.

Deze programma's moeten geïnstalleerd zijn voordat de tutorial wordt doorlopen.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het opzetten van een Quartus-project.
- Het opzetten en uitvoeren van een simulatie.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Voer de tutorial hofdstuk uit vanaf hoofdstuk 5.

b) 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 lesweek 12 – Ontwerp van een 6-teller

Inleiding
Tijdens het dit practicum wordt een 6-teller ontwikkeld. Een 6-teller telt de telstanden 101 — $\underbrace{000 - 001 - 010 - 011 - 100 - 101}_{\text{De zes telstanden}}$ — 000. We ontwikkelen de zesteller met T-flipflops en poorten.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het ontwikkelen van een binaire teller met ingekorte telcyclus
- Het opzetten en uitvoeren van een simulatie.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Stel een tabel op met de telstanden van de zesteller en geef aan of een flipflop toggelt of niet.

b) Bepaal de functies van de T-ingangen van de flipflops.

c) Maak een nieuw Quartus-project aan. Noem het project `zesteller`.

d) Voer het schema van de zesteller in en sla het bestand op onder de naam `zesteller.bdf`. Zorg voor een asynchrone, actief lage, reset.

e) Synthetiseer de schakeling.

f) Koppel de ingangen en uitgangen van de zesteller aan pinnen van de FPGA. Sluit het kloksignaal aan op een drukknop.

g) Maak een ModelSim commando-script aan en sla dit bestand op onder de naam `tb_zesteller.do`. Zorg ook voor stimuli van de klok en de reset.

h) Simuleer de teller met ModelSim.

i) Test de teller op het bordje.

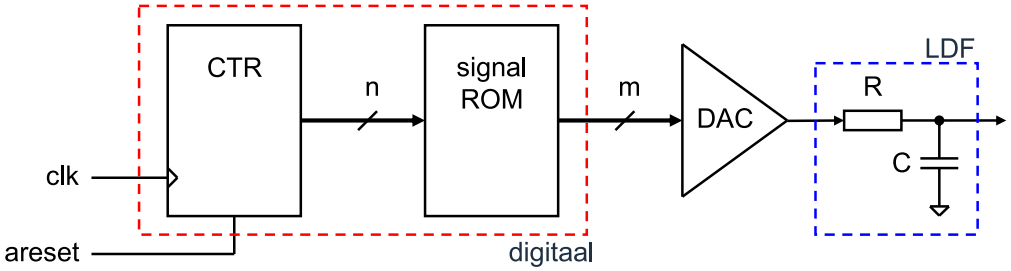
j) Laat de docent het geheel controleren.

Opmerkingen
In het ontwerp wordt een kloksignaal gebruikt. Sluit dit signaal aan op een drukknop.

Opdracht lesweek 13 – Ontwerp van een digitale sinusgenerator

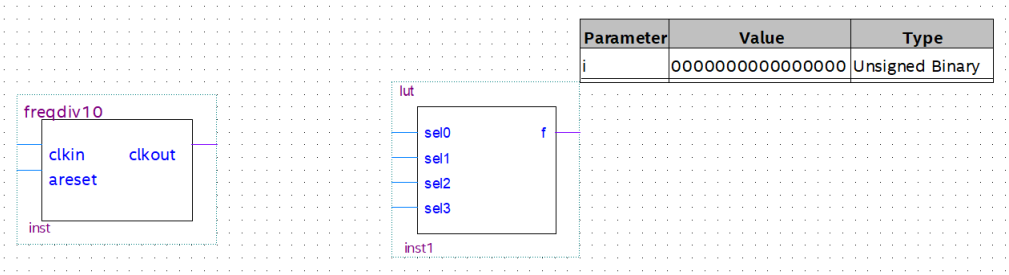
Inleiding

Tijdens dit practicum wordt een digitale sinusgenerator ontwikkeld. De sinusgenerator bestaat uit een 4-bits teller, een 16x4-bits ROM en een 4-bits Digital Analog Converter (DAC). Een principeschema is te vinden in figuur 13 met $n = m = 4$. Het laagdoorlaatfilter LDF kan in eerste instantie achterwege worden gelaten.



Figuur 13: *Principeschema van de sinusgenerator.*

Als teller gebruiken we een standaard 4-bits teller gemaakt met T-flipflops en poorten. De uitgangen van de teller sturen een signal ROM aan. De signal ROM is in feite een combinatorische schakeling die met bekende methoden ontwikkeld kan worden. De DAC zit niet in de FPGA maar is een externe schakeling. Als kloksignaal gebruiken we de onboard klokgenerator. Aangezien de frequentie vrij hoog is (50 MHz) gebruiken we een *frequency divider* om de frequentie omlaag te brengen naar 5 MHz. Het ontwikkelen van de combinatorische logica van de ROM laten we door Quartus uitvoeren. Daarvoor gebruiken we een Lookup Table (LUT). De symbolen van beide schakelingen is te zien in figuur 14. De LUT kan elke functie van 4 variabelen maken door de bits onder **Value** aan te passen. Links is het minst significante bit (functiewaarde f_0), rechts het meest significante bit (functiewaarde f_{15}).



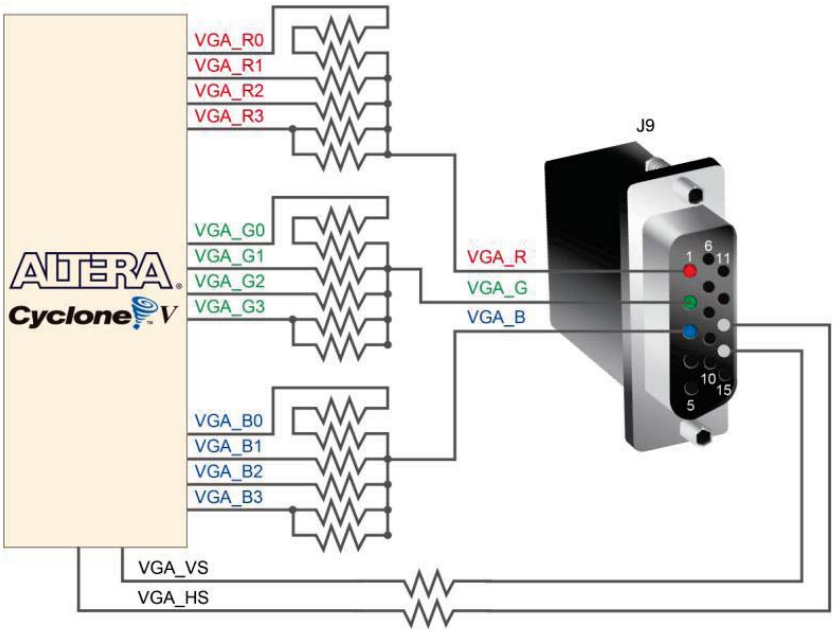
Figuur 14: *De symbolen van de frequency divider en de LUT.*

Als DAC gebruiken we één van de uitgangen van de VGA-connector. In dit geval gebruiken we het rood-signaal. Zie figuur 15. Om het analoge signaal te zien gebruiken we een oscilloscoop en een probe. Gebruik pin 1 voor het rood-signaal en gebruik pin 6 als referentie.

Leerdoelen

De leerdoelen van deze opdracht zijn:

- Het ontwikkelen van een digitale sinusgenerator.
- Het opzetten en uitvoeren van een simulatie.



Figuur 15: *De aansluitingen van de VGA-connector.*

Opdrachten

De volgende opdrachten moeten gedaan worden:

a) Maak een nieuw Quartus-project aan. Noem het project `singen`.

b) Kopieer van BlackBoard de bestanden `lut.vhd`, `lut.bsf`, `freqdiv10.bdf` en `freqdiv10.bsf` en plaats deze bestanden in de map van het project.Zorg ervoor dat `freqdiv10.bdf` geplaatst is in de Project Navigator.

c) Voer het schema van de 4-bits teller in en sla het bestand op onder de naam `teller.bdf`. Zorg voor een asynchrone, actief lage, reset.

d) Maak een nieuw schemabestand aan en plaats daarin de frequency divider, de teller en 4 LUTs in. Verbind de componenten met elkaar en plaats de ingangen en de uitgangen. Noem de vier uitgangen van de LUTs `s0`, `s1`, `s2` en `s3`.

e) Synthetiseer de schakeling om Quartus de ingangen en uitgangen te laten ontdekken.

f) Koppel de ingangen en uitgangen aan pinnen van de FPGA. De pin mappings staan hieronder:

```
clk    -> PIN_M9          s0 -> PIN_A9
areset -> PIN_P22         s1 -> PIN_B10
                          s2 -> PIN_C9
                          s3 -> PIN_A5
```

g) Compileer de gehele schakeling.

h) Test de schakeling op het bordje. Maak gebruik van een oscilloscoop om het signaal te zien.

i) Laat de docent het geheel controleren.

j) Maak een ModelSim commando-script aan en sla dit bestand op onder de naam `tb_singen.do`. Zorg ook voor stimuli van de klok en de reset.

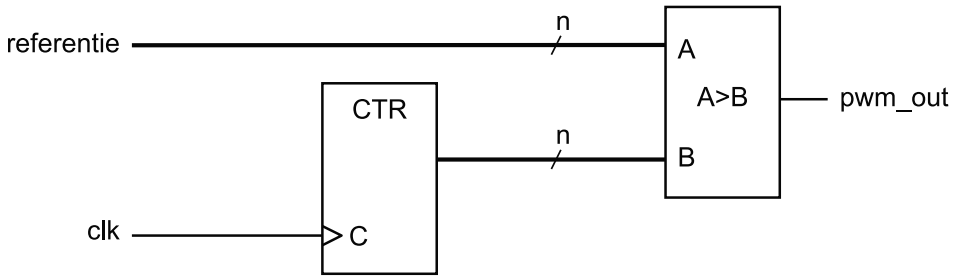
k) Simuleer de schakeling.

l) Laat de docent het geheel controleren.

Opmerkingen
In het ontwerp wordt een kloksignaal gebruikt. Sluit dit signaal aan op onboard klokgenerator.

Opdracht lesweek 14 – Ontwerp van een PWM-generator

Inleiding
Tijdens dit practicum wordt een digitale PWM-generator ontwikkeld. De PWM-generator bestaat uit een 4-bits teller en een 4-bits unsigned vergelijkschakeling. Een principeschema is te vinden in figuur 16 met $n = 4$.



Figuur 16: *Principeschema van de PWM-generator.*

Als teller gebruiken we een standaard 4-bits teller gemaakt met T-flipflops en poorten. De uitgangen van de teller zijn gekoppeld aan de B-ingang van de vergelijkschakeling. Aan de A-ingang van de vergelijkschakeling worden vier schakelaars gekoppeld. Deze schakelaars vormen een 4-bits unsigned getal. De uitgang van de vergelijkschakeling is 1 als het (unsigned) getal op de A-ingang groter is dan het (unsigned) getal op de B-ingang, anders is de uitgang 0.

De vergelijkschakeling is op twee manieren te maken: direct uit de bits van de twee getallen, zie het boek paragraaf 5.19.3, of met behulp van een aftrekschakeling, zie boek paragraaf 5.20 (tabel 5.12). Deze aftrekschakeling is al gemaakt in de opdracht van lesweek 7.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het ontwikkelen van een PWM-sinusgenerator.
- Het opzetten en uitvoeren van een simulatie.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Maak een nieuw Quartus-project aan. Noem het project `pwmgen`.

b) Kopieer de teller en de frequency divider van de sinusgenerator en plaats deze in de map van het project. Voeg deze componenten toe in de project navigator.

c) Maak een nieuw schemabestand aan en ontwerp daarin de vergelijkschakeling. Sla de schakeling op onder de naam `agtb.bdf`. Als gebruik gemaakt wordt van de aftrekschakeling, zorg dat alle BDF-bestanden in de project navigator geplaatst zijn.

d) Maak een nieuw schemabestand aan en plaats daarin de teller, de frequency divider en de vergelijkschakeling. Verbind de componenten met elkaar en plaats de ingangen en de uitgangen. Noem de vier uitgangen van de teller `q0`, `q1`, `q2` en `q3`.

e) Synthetiseer de schakeling om Quartus de ingangen en uitgangen te laten ontdekken.

f) Koppel de ingangen en uitgangen aan pinnen van de FPGA. De pin mappings staan hieronder:

```
clk      -> PIN_M9
areset   -> PIN_P22
pwm_out  -> PIN_AA2
q0       -> PIN_U2
q1       -> PIN_U1
q2       -> PIN_L2
q3       -> PIN_L1
ref0     -> PIN_U13
ref1     -> PIN_V13
ref2     -> PIN_T13
ref3     -> PIN_T12
```

(De pinnen voor de telleruitgangen zijn nodig voor de simulatie. De `ref`-signalen zijn verbonden met schakelaars voor het aanbieden van de referentie.)

g) Compileer de gehele schakeling.

h) Test de schakeling op het bordje. Test de lichtsterkte van de led voor alle 16 mogelijke instellingen van de schakelaars.

i) Laat de docent het geheel controleren.

j) Maak een ModelSim commando-script aan en sla dit bestand op onder de naam `tb_pwmgen.do`. Zorg ook voor stimuli van de klok en de reset.

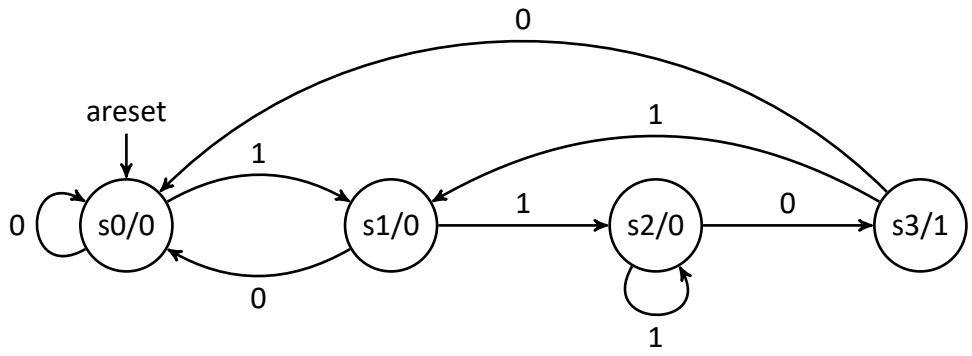
k) Simuleer de schakeling.

l) Laat de docent het geheel controleren.

Opmerkingen
In het ontwerp wordt een kloksignaal gebruikt. Sluit dit signaal aan op onboard klokgenerator.

Opdracht lesweek 15 – Ontwerp van een toestandsmachine

Inleiding
Tijdens dit practicum wordt een toestandsmachine voor het herkennen van een bitpatroon ontwikkeld. Het te herkennen patroon is 110. De machine wordt als Moore-machine ontwikkeld. Het toestandsdiagram van de machine is te zien in figuur 17



Figuur 17: *Toestandsdiagram voor machine.*

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het ontwikkelen van een toestandsmachine.
- Het opzetten en uitvoeren van een simulatie.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Stel op papier de waarheidstabel op voor de next-state functies en de uitgang gebruikmakend van D-flipflops en bepaal de functies. Maak gebruik van Karnaughdiagrammen. Laat de functies controleren door de docent.

b) Maak een nieuw Quartus-project aan. Noem het project `herkenner`.

c) Maak een nieuw schemabestand aan en ontwerp daarin de toestandsmachine. Gebruik de binaire telcode als toestandscodering. Sla de schakeling op onder de naam `herkenner.bdf`.

d) Synthetiseer de schakeling om Quartus de ingangen en uitgangen te laten ontdekken.

e) Koppel de ingangen en uitgangen aan pinnen van de FPGA. Gebruik een drukknop als kloksignaal.

f) Compileer de gehele schakeling.

g) Test de schakeling op het bordje.

h) Laat de docent het geheel controleren.

i) Maak een ModelSim commando-script aan en sla dit bestand op onder de naam `tb_herkenner.do`. Zorg ook voor stimuli van de klok en de reset.

j) Simuleer de schakeling.

k) Laat de docent het geheel controleren.

Opmerkingen

In het ontwerp wordt een kloksignaal gebruikt. Sluit dit signaal aan op drukknop.

Opdracht lesweek 16 – Ontwerp van een toestandsmachine

Inleiding
Tijdens dit practicum wordt een toestandsmachine voor het herkennen van een bitpatroon ontwikkeld. Het te herkennen patroon is 1001, doorlopend en overlappend. De machine wordt als Mealy-machine ontwikkeld. Voor de toestandscodering wordt de one-hot-code gebruikt.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Het ontwikkelen van een toestandsmachine.
- Het opzetten en uitvoeren van een simulatie.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Stel op papier het toestandsdiagram op voor deze machine. Laat het toestandsdiagram controleren door de docent.

b) Stel op papier de waarheidstabel op voor de next-state functies en de uitgang gebruikmakend van D-flipflops en bepaal de functies. Gebruik de one-hot-codering. Laat de functies controleren door de docent.

c) Maak een nieuw Quartus-project aan. Noem het project `onehot`.

d) Maak een nieuw schemabestand aan en ontwerp daarin de toestandsmachine. Sla de schakeling op onder de naam `onehot.bdf`.

e) Synthetiseer de schakeling om Quartus de ingangen en uitgangen te laten ontdekken.

f) Koppel de ingangen en uitgangen aan pinnen van de FPGA. Gebruik een drukknop als kloksignaal.

g) Compileer de gehele schakeling.

h) Test de schakeling op het bordje.

i) Laat de docent het geheel controleren.

j) Maak een ModelSim commando-script aan en sla dit bestand op onder de naam `tb_onehot.do`. Zorg ook voor stimuli van de klok en de reset.

k) Simuleer de schakeling.

l) Laat de docent het geheel controleren.

Opmerkingen
In het ontwerp wordt een kloksignaal gebruikt. Sluit dit signaal aan op drukknop.