OPDRACHTEN PRACTICUM

DIGSE1

DE HAAGSE HOGESCHOOL

J.E.J op den Brouw
De Haagse Hogeschool
Opleiding Elektrotechniek
19 februari 2019
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 digitale systemen leert beschrijven, een competentie die onmisbaar is voor elke elektrotechnische ingenieur. Je kunt dit alleen leren door zelfstandig alle opdrachten uit te voeren. Het kopiëren van code of codedelen van het internet of van collega-studenten is niet leerzaam! Het boek biedt voldoende voorbeelden die je ter inspiratie kunt gebruiken. Gebruik nooit code die je zelf niet begrijpt in je beschrijvingen.

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 in VHDL 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 zogenoemde 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 11.1sp1.

De software is te downloaden via http://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.

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

[2] Zie http://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.
- Altijd de aanwijzingen van het personeel opvolgen.

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 practicumopdracht moet uiterlijk één week later worden afgerond dan de week waarin de opdracht moet worden uitgevoerd.
4. Te laat ingeleverde opdrachten zijn automatisch onvoldoende en kunnen niet worden ingehaald!
5. De student moet tijdens de practicumuren aan de opgegeven practicumopdrachten werken.
6. Een ingeleverde opdracht die niet met een voldoende wordt beoordeeld kan (een week later) worden aangevuld. Als je code niet helemaal perfect blijkt te zijn is dat dus geen probleem. Als je pas begint met beschrijven is het normaal dat alles niet meteen perfect is. Natuurlijk moet je wel proberen om je code zo goed te maken als je zelf kunt.
7. Het laten nakijken van VHDL-code die je niet zelf bedacht en beschreven hebt, wordt beschouwd als fraude (net zoals het spieken bij tentamens). Als je code 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 blok, de student krijgt dan zijn eindbeoordeling: voldoende of onvoldoende.
9. Een onvoldoende als eindresultaat kan worden herkanst in week 10. De opdrachten die nog niet met een voldoende zijn beoordeeld, dienen binnen 15 minuten te worden gedemonstreerd en beoordeeld.

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 http://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. De uitvoerweek staat vermeld bij de opdracht.

Resultaat practicum
Het practicum wordt met een voldoende beoordeeld als:

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

[3] Zie http://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:

- Zie de algemene leerdoelen.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Log in op BlackBoard en meld je aan voor de Course DIGSE1.

b) In de course DIGSE1 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.

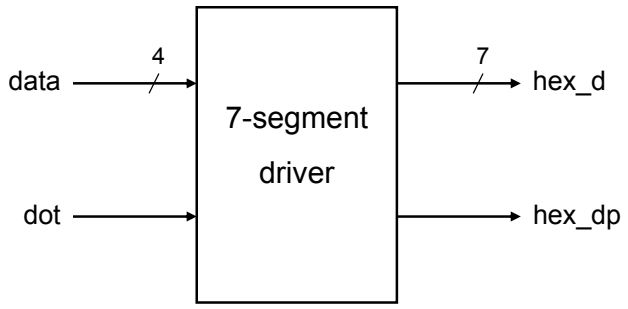
Opdracht week 2 – 7-segment display

Inleiding

De 7-segment displays spelen een belangrijke rol bij het afbeelden van decimale en hexadecimalen getallen. Het principe is eenvoudig. Zeven leds (segments) zijn zodanig gerangschikt dat de cijfers 0 t/m 9 zijn af te beelden. Daarnaast bevat elk display meestal een punt (dot) om een komma in een getal te introduceren (merk op dat angelsaskische landen een punt gebruiken tussen eenheden en decimalen, in Nederland gebruiken we een komma). Het is mogelijk om ook de letters A t/m F weer te geven, zodat hexadecimale getallen kunnen worden afgebeeld.

Op het ontwikkelbord zijn vier 7-segment displays naast elkaar geplaatst zodat de getallen 0 t/m 9999 (of FFFF) zijn weer te geven.

In deze opdracht ga je één 7-segment decoder in VHDL ontwikkelen voor één 7-segment display. Dat kan heel eenvoudig met behulp van Selected Signal Assignment. Je moet alleen goed in de gaten houden dat de 7-segment ledjes gaan branden als een logische 0 wordt aangeboden. Het ontwerp heeft twee ingangen en twee uitgangen, zie figuur 1.



Figuur 1: *Aansluiting 7-segment driver.*

Door de pijl bij `data` staat een streepje met daarbij het getal 4. Dit geeft aan dat data een vector is van vier bits breed. Vector `hex_d` is dus zeven bits breed.

In listing 1 op pagina 7 is al wat ingevuld, maar er zitten founten in en er ontbreekt nog wat.

Leerdoelen

De leerdoelen van deze opdracht zijn:

- Opzetten van een nieuw project in Quartus.
- Ontwerpen van een schakeling met behulp van VHDL's Selected Signal Assignment.
- Testen van de ingevoerde beschrijving.
- Diverse VHDL-taalconstructies leren kennen.

```
library ieee;
use ieee.std_logic_1164.all;
entity seven_segment is
    -- port definitie is niet correct
    port (data   : in std_logic_vector;
          dot    : in std_logic;
          hex_d  : out;
          hex_dp : out std_logic
    );
end seven_segment;

architecture behav of seven_segment is
begin
    with data select
        --       6543210
        hex_d <= "111111" when "0000",
                 "000000" when "0001",
                 -- nog aanvullen
                 "-------" when others;
    -- de punt
    hex_dp <= dot;
end behav;
```

Listing 1: *Beschrijving 7-segment decoder (foutief en incompleet)*

Opdrachten

De volgende opdrachten moeten gedaan worden:

a) Maak een nieuw project aan. Kies een geschikte map op de schijf en geef het project en de *top-level entity* de naam `seven_segment`.

b) Maak de beschrijving uit listing 1 compleet. De signalen zijn van het type `std_logic` of `std_logic_vector`, tenzij anders aangegeven. In bijlage B van de tutorial staat hoe de segmenten gepositioneerd zijn.

c) Voer de juiste pinkoppelingen in. Gebruik `HEX0_D` als 7-segment display. De pinaansluitingen staan vermeld in bijlage B van de tutorial.

d) Synthetiseer en implementeer de VHDL-code en laadt het in het DE0-bordje.

e) Test het ontwerp op een DE0-bord.

Opmerkingen

Bij deze opdracht wordt nog niet gesimuleerd.

De beschrijving uit de listing bevat tal van fouten en is incompleet. Je moet deze aanpassen en uitbreiden.

De leds van de 7-segment display werken actief laag. Een logische ‘0’ uit de schakeling zorgt ervoor dat de leds gaan branden.

Opdracht week 3 – schuifregister

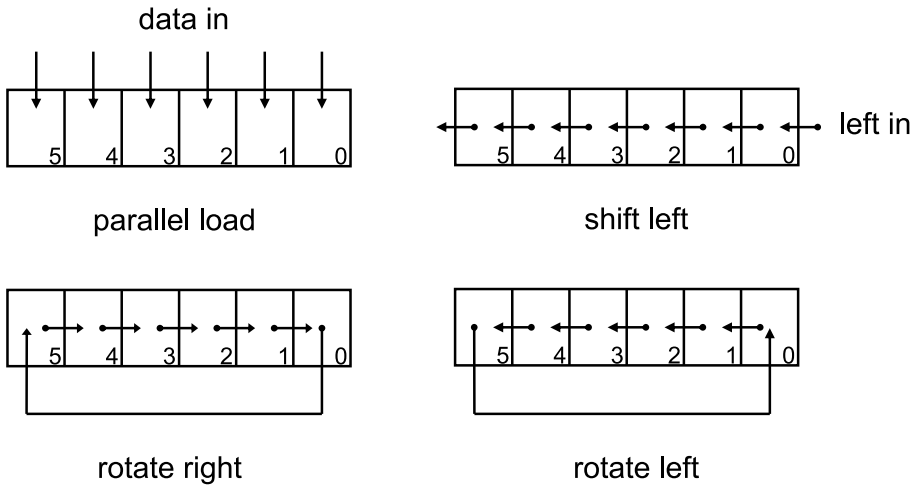
Inleiding

In de digitale techniek worden schuifregisters gebruikt voor tal van zaken. Zo kan je het gebruiken voor seriële communicatie (Ethernet, RS232, USB, SATA) of vermenigvuldigen met en delen door 2.

De opdracht voor deze week is het beschrijven en beproeven van een 6-bit schuifregister met de volgende werkmodi:

- één bit naar links schuiven en aanvullen met een extern databit.
- één bit linksom roteren
- één bit rechtsom roteren
- parallel laden van het register met externe data

De vier werkmodi zijn uitgebeeld in figuur 2. Merk op dat de parallelle uitgangen niet zijn getekend.



Figuur 2: *De vier werkmodi van het schuifregister*

Natuurlijk heeft het schuifregister een klokingang. De vier werkmodi worden aangeboden als een 2-bits waarde (vector). De externe data wordt aangeboden als een 6-bits binaire waarde (vector). De stand (inhoud) van het schuifregister wordt zichtbaar gemaakt via de uitgangen als een 6-bits vector. Het schuifregister moet een asynchrone reset hebben.

Een schuifregister bestaat uit een serieschakeling van D-flipflops. Dat houdt in dat alle acties zoals hierboven beschreven moeten worden uitgevoerd onder klokflanksturing.

De opdracht voor deze week is het ontwerpen en beproeven van een 6-bit schuifregister.

Leerdoelen

De leerdoelen van deze opdracht zijn:

- Opstellen van de juiste entity-beschrijving.
- Ontwerpen/beschrijven van kloksynchrone logica
  - Beschrijven van een asynchrone reset.
  - Beschrijven van de klokflank (opgaand of neergaand).
  - Beschrijven diverse acties die op de klokflank uitgevoerd moeten worden.

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Stel een lijst op met alle in- en uitgangen van het te ontwerpen schuifregister. Geef tevens aan uit hoeveel bits elk signaal bestaat.

b) Voer de beschrijving van het schuifregister in. De signalen zijn van het type `std_logic` of `std_logic_vector`, tenzij anders aangegeven.

c) Voer de juiste pinkoppelingen in. De pinaansluitingen staan vermeld in bijlage B van de tutorial.

d) Synthetiseer en implementeer de VHDL-code en laadt het in het experimenteerbord.

e) Test het ontwerp op een DE0-bord.

f) *Optioneel:* Breidt de mogelijkheden van het schuifregister uit met een *hold* en *shift right* mogelijkheid. Hiervoor is een extra stuursignaal nodig.

Noot: bij deze opdracht wordt nog niet gesimuleerd.

Opmerkingen
In VHDL mogen uitgangen (port-beschrijvingen) alleen maar links van de *signal assignment operator* staan, d.w.z. dat er alleen waarden aan toegekend kunnen worden. Ze kunnen niet aan de rechterkant van de signal assignment operator staan want dan worden ze “gelezen”, d.w.z. ze dienen als ingang voor een of andere digitale schakeling. Je zal hiervoor dus een intern signaal of variabele moeten aanmaken (de interne stand van het schuifregister) en die interne stand toekennen aan de uitgangen.

Schuiven in VHDL kan heel makkelijk door het gebruik van *vector slices* en de *concatenation operator*. Zo kan je naar links schuiven met

```
reg <= reg(6 downto 0) & '0';
```

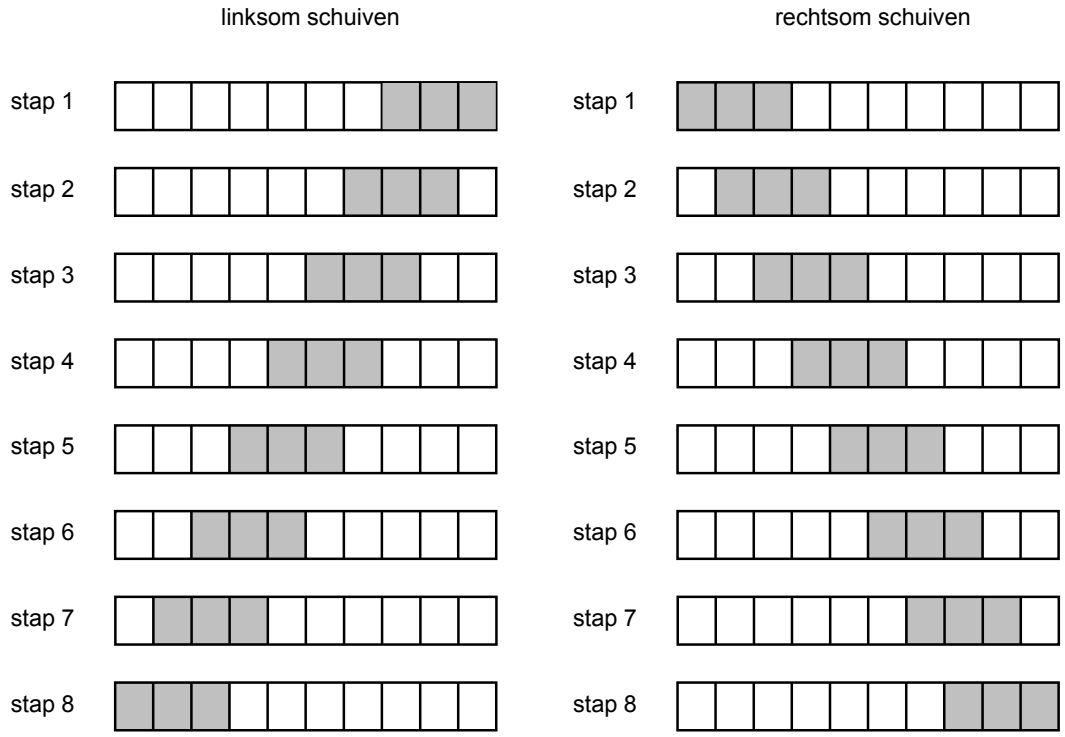
als signaal `reg` een 8-bits schuifregister is. Er wordt hier een 0 ingeschoven. Zie hoofdstuk 8 van het dictaat voor meer informatie over schuifregisters.

De klokgenerator op het DE0-board loopt op 50 MHz. Dat is veel te snel voor mensen om goed te kunnen testen. Verbind daarom (via de Pin Planner) de klokingang van de VHDL-beschrijving met BUTTON2. Nu kan je een kloksignaal aanbieden door op het knopje te drukken.

Opdracht week 4 – Knight Rider

Inleiding

Knight Rider is een televisieserie uit de jaren '80[4]. Daarin speelde een man samen met een high-tech auto als strijder tegen het onrecht. Op de auto was een *lamp bar* geplaatst met een tiental lampen. De lampen gaan één voor één aan en uit zodanig dat het lijkt alsof een drietal lampjes heen-en-weer schuift; ze stuiteren tegen de randen. Zie figuur 3.



Figuur 3: *aansturing lampjes Knight Rider*

De opdracht voor deze week is het beschrijven en beproeven van een Knight Rider-systeem. Het systeem als geheel werkt als volgt:

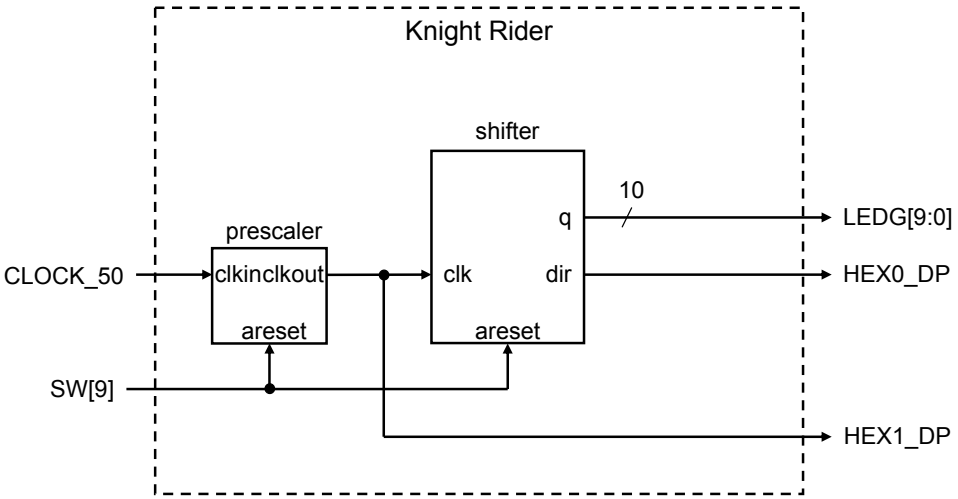
- Het systeem moet drie bits heen-en-weer schuiven; dit komt overeen met drie ledjes op het DE0-bord.
- Het totaal aantal ledjes waarover het drie-bits patroon heen en weer schuift is tien (alle groene ledjes).
- De schuifsnelheid bedraagt 20 Hz.
- Het systeem moet herstart kunnen worden door middel van een asynchrone reset.
- Bij het schuiven mag elk patroon slechts één stand actief zijn, dus na stap 8 (linksom) volgt stap 2 (rechtsom).

[4] Zie http://knightrideronline.com/

In deze opdracht is het gebruik van structural VHDL een noodzaak. Er zijn drie VHDL-componenten nodig:

- Een schuifeenheid, voor het heen en weer schuiven van het patroon. Deze eenheid geeft naast de stand (`q`) ook de schuifrichting door (`dir`).
- Een *prescaler*. Het schuiven gaat met een klokfrequentie van 50 MHz razendsnel. Om het één en ander zichtbaar te maken is een noodzakelijk de frequentie te verlagen. De verlaagde fequentie wordt zichtbaar gemaakt op de punt van een 7-segment display.
- Een top-level component die de structuur beschrijft van de Knight Rider. Deze component doet niets anders dan het instantiëren van de twee andere componenten.

In figuur 4 is het blokschema getekend.



Figuur 4: *Blokschema van de Knight Rider*

Voor het afbeelden van het signaal `clkout` van de prescaler en de schuifrichting van de shifter worden twee decimale punten van de zeven segmenten displays gebruikt.

Natuurlijk moet de werking van de shifter worden getest. Dat kan prima met de simulator. De prescaler werkt al; die hoeft niet te worden gesimuleerd.

Omdat het opzetten van een project met daarin structural VHDL best wat typewerk vergt, is er al een project aangemaakt. Het project is uiteraard niet volledig, er moet nog het nodige aan code geschreven worden.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Gebruik van structural VHDL voor het beschrijven van hiërarchieën.
- Scheiding structural VHDL en hardware-genererende VHDL.
- Beschrijven van het gedrag van een schuifregister met specifieke eigenschappen.

- Gebruik van generieke parameters tijdens instantiëring van componenten.
- Opzetten en uitvoeren van een simulatie van VHDL-code.
- Invoeren en testen van de gegenereerde hardware.

Opdrachten
De volgende opdrachten moeten gedaan worden. Er is al een opzet van het project beschikbaar, dus hoeft er geen nieuw project te worden aangemaakt.

a) Haal van BlackBoard het bestand `digse1_knight_rider.zip` binnen en pak het uit in `H:\QUARTUS\DIGSE1` of een andere geschikte map op de disk. Het zip-bestand bevat het project `knight_rider`.

b) Maak de VHDL-beschrijving voor de `shifter`.

c) Maak de structural VHDL-beschrijvingen voor `knight_rider` compleet.

d) Schrijf een testbench en een ModelSim command file voor de **shifter** (dus niet voor het hele ontwerp).

e) Simuleer de shifter.

f) Synthetiseer en implementeer het hele ontwerp en laadt het in het DE0-bordje. De pinaansluitingen kan je als bestand terugvinden op BlackBoard.

g) Test het ontwerp op een DE0-bord.

Opmerkingen
In VHDL mogen uitgangen (port-beschrijvingen) alleen maar links van de *signal assignment operator* staan, d.w.z. dat er alleen waarden aan toegekend kunnen worden. Ze kunnen niet aan de rechterkant van de signal assignment operator staan want dan worden ze “gelezen”, d.w.z. ze dienen als ingang voor een of andere digitale schakeling. Je zal hiervoor dus een intern signaal of variabele moeten aanmaken (de interne stand van het schuifregister) en die interne stand toekennen aan de uitgangen.

Let erop dat tijdens de asynchrone reset het schuifregister geladen moet worden met een patroon met drie aaneensluitende enen. De overige bits moeten nullen zijn.

Opdracht week 5 en 6 – vier-cijfer-teller

Inleiding

In de digitale techniek worden tellers gebruikt voor tal van zaken. Zo kan je het gebruiken om een aantal gebeurtenissen te tellen, of om tijd te meten.

De opdracht voor deze week is het ontwerpen (en beschrijven) van een vier-cijferige teller, waarbij hiërarchie (structural VHDL) een belangrijke rol speelt. Bij deze opdracht wordt gebruik gemaakt van een prescaler (zie project), een omschakelbare BCD/HEX tellers (vier stuks) en de eerder ontworpen 7-segment decoder (ook vier stuks).

Ter voorbereiding op de opdracht is een Quartus II project beschikbaar dat de prescaler beschrijft. Dit project kan gecompileerd worden en in een DE0-ontwikkeldbord geplaatst worden. Bestudeer de beschrijving van de prescaler en zorg dat je snapt hoe de werking ervan is. Tevens is een project geplaatst waarin al een begin is gemaakt met het beschrijven van de vier-cijfer-teller. Het geheel werkt nog niet. Lees de bijbehorende opdracht voor meer informatie.

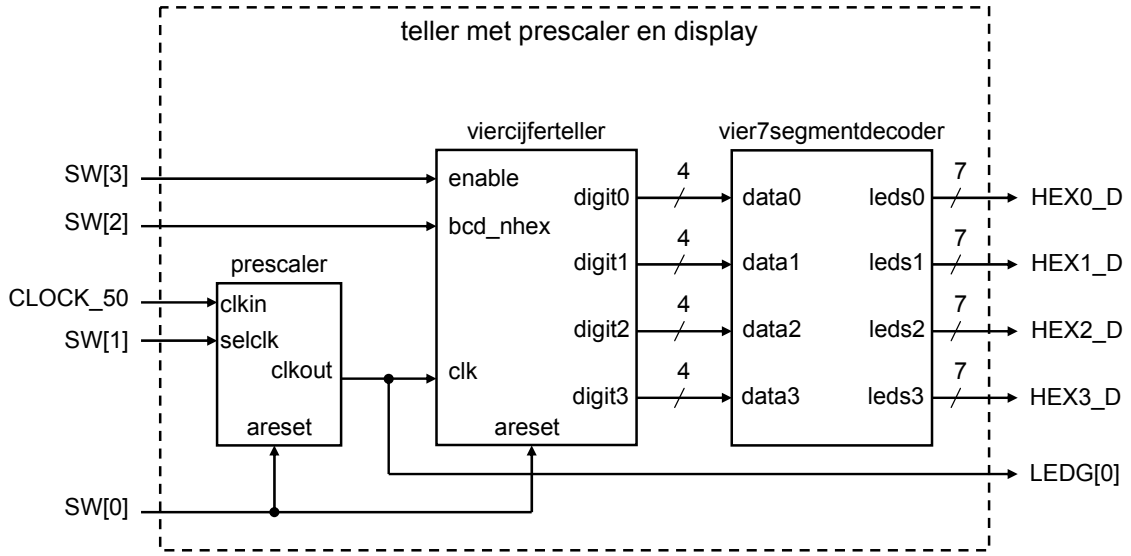
De vier-cijfer-teller als geheel werkt als volgt:

- De vier-cijfer-teller moet kunnen tellen in het BCD- en hexadecimale talstelsel.
- De vier-cijfer-teller bestaat uit vier identieke, gecascadeerde tellers. De tellers moeten dus als BCD- of hex-tellers ingesteld kunnen worden.
- De stand van de vier-cijfer-teller (BCD of hexadecimaal) moet op de vier 7-segment displays worden afgebeeld.
- De telfrequentie kan ingesteld worden op 1 Hz of 10 Hz. Aangezien de klokfrequentie van het DE0-board 50 MHz bedraagt, moet een prescaler gebruikt worden om de klokfrequentie te verlagen. Deze prescaler is gegeven.
- De vier-cijfer-teller moet herstart kunnen worden door middel van een asynchrone reset.
- De vier-cijfer-teller moet kunnen tellen of gestopt worden door middel van een enable-signaal.

In deze opdracht is het gebruik van structural VHDL een noodzaak. In onderstaande figuren wordt één en ander duidelijk.

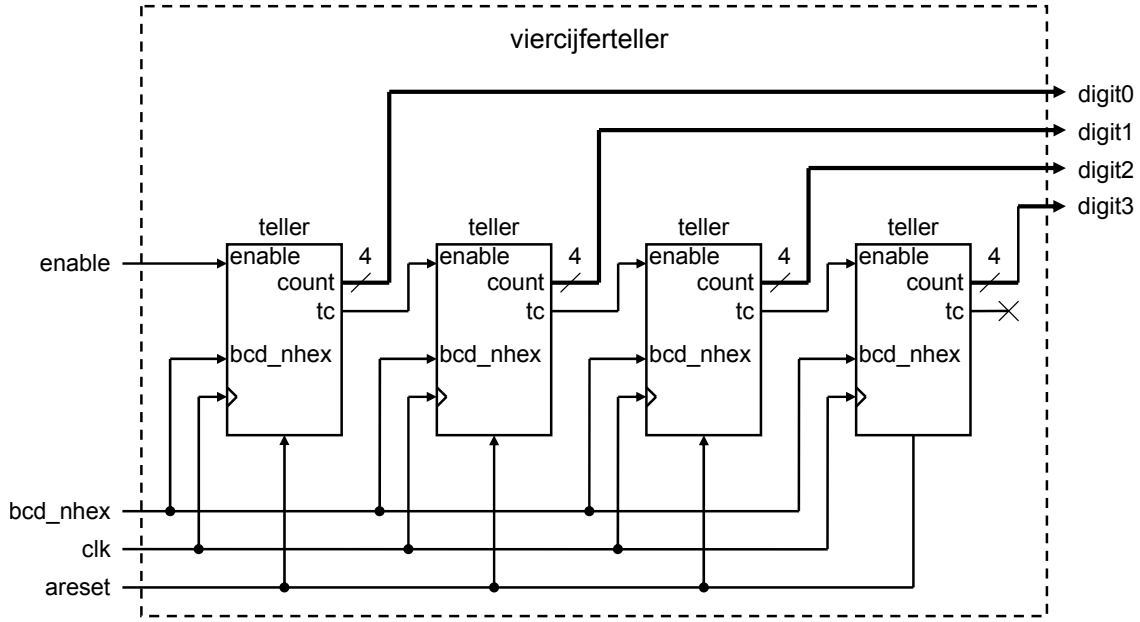
Op het hoogste niveau, de zogenoemde *top level*, zijn drie componenten te onderscheiden: een prescaler, een vier-cijfer-teller en een 7-segment display, zie figuur 5. De signaalnamen aan de rand komen overeen met de pinbenamingen van het DE0-bordje. Bij een aantal signalen is het aantal bits opgegeven, deze signalen moeten als vector worden gebruikt.

De vier-cijfer-teller en 7-segment display bestaan zelf weer uit componenten.



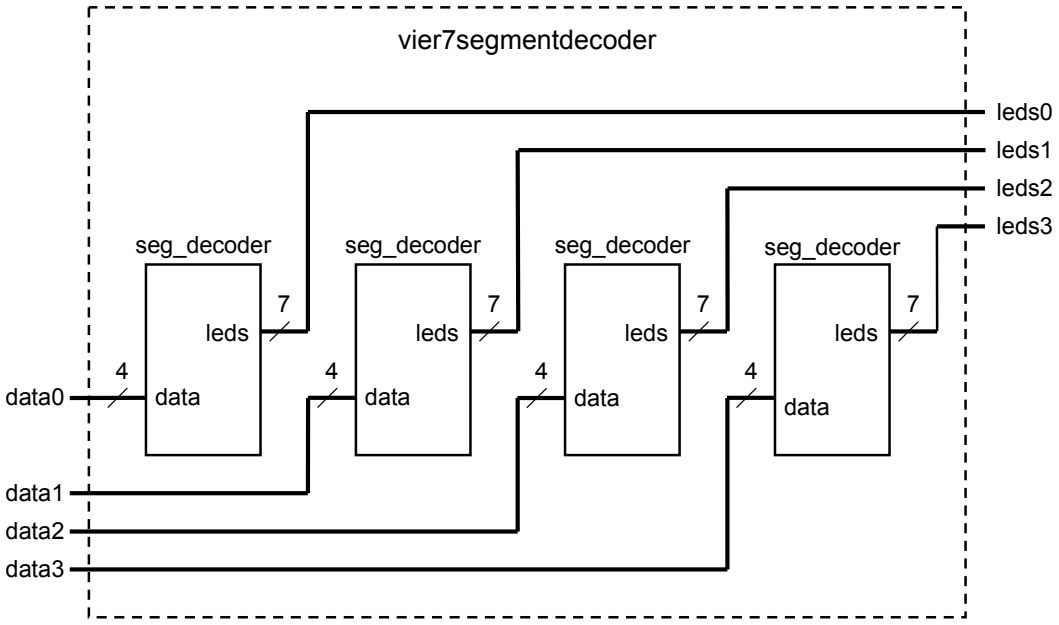
Figuur 5: *Top-level blokdiagram met prescaler, teller en display*

De `viercijferteller` (zie figuur 6) bestaat uit vier identieke, gecascadeerde, omschakelbare BCD/hex-tellers, met een *enable* en een *terminal count*. Merk op dat de terminal count van de meest significantie sectie niet naar buiten wordt gevoerd.



Figuur 6: *Blokdiagram van de vier-cijfer-teller.*

De `vier7segmentdecoder` (zie figuur 7) bestaat uit vier identieke 7-segment decoders. Deze zijn al tijdens een eerder practicum beschreven en getest.



Figuur 7: *Blokdiagram van de vier 7-segment decoder.*

Het is duidelijk dat dit een wat groter project is. Het kost aanzienlijk veel tijd om alles in VHDL te coderen en natuurlijk het project helemaal in te richten.

Op blackboard vind je een Quartus-project waarin het één en ander is ingericht. Je kan je nu concentreren op de VHDL-beschrijvingen van de tellers, van de `viercijferteller` en `vier7segmentdecoder`.

Opmerking: voor de tellers moeten unsigned signals/variables gebruikt worden.

Leerdoelen
De leerdoelen van deze opdracht zijn:

- Gebruik van structural VHDL voor het beschrijven van hiërarchieën.
- Scheiding structural VHDL en hardware-genererende VHDL.
- Gebruik van datatype `unsigned` voor rekenkundige bewerkingen.
- Opzetten en uitvoeren van een simulatie van VHDL-code.
- Beschrijven van het gedrag van een teller (BCD/hex, enable).

Opdrachten
De volgende opdrachten moeten gedaan worden:

a) Haal het bestand `digse1_viercijfertellermetprescalerendisplay.zip` van BlackBoard en pak het uit in `H:\QUARTUS\DIGSE1`. Het zip-bestand bevat het Quartus-project `viercijfertellermetprescalerendisplay`. Pak het zip-bestand uit en plaats het in een map op de disk.

b) Maak de structural VHDL-beschrijvingen voor `viercijferteller`, `vier7segmentdecoder` en `viercijfertellermetprescalerendisplay` compleet.

c) Schrijf een testbench en een ModelSim command file voor de **teller** (dus niet voor het hele ontwerp).

d) Simuleer de teller.

e) Compileer de VHDL-code en laadt het in het DE0-bordje. De pinaansluitingen zijn al ingevuld.

f) Test het ontwerp op een DE0-bord.

Opmerkingen
In VHDL mogen uitgangen (port-beschrijvingen) alleen maar links van de *signal assignment operator* staan, d.w.z. dat er alleen waarden aan toegekend kunnen worden. Ze kunnen niet aan de rechterkant van de signal assignment operator staan want dan worden ze "gelezen", d.w.z. ze dienen als ingang voor een of andere digitale schakeling. Je zal hiervoor dus een intern signaal of variabele moeten aanmaken die interne stand toekennen aan de uitgangen (noot: in VHDL-2008 mag dit weer wel).

De *terminal count* is een combinatorische uitgang en moet dus buiten de klokflank beschreven worden.

Zie hoofdstuk 8 van het dictaat voor meer informatie over tellers.