Opgave 5

Overzicht opdracht

Maak een scorebord met alle acht 7-segment displays op het Nexys bord, net zoals in Mode 11 uit opgave 2, maar laat ditmaal alle acht displays "tegelijkertijd" verschillende cijfers afbeelden. Herinner je dat dit eigenlijk onmogelijk is, vermits de cathodes voor de acht displays gemeenschappelijk zijn, maar op p. 24 van de Nexys A7 handleiding kan je lezen hoe je de illusie kan creëren van "gelijktijdig" verschillende cijfers op de displays.

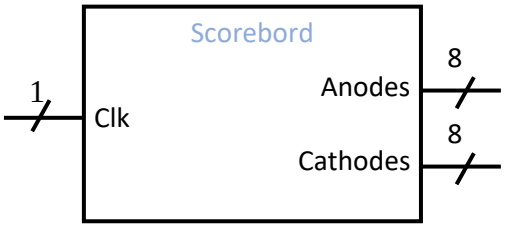
Voor deze opgave moet je werken met een *refresh* frequentie van 1kHz, dus iedere 1 ms moeten alle acht displays ververst zijn (125 µs per display).

Op de displays moeten van links naar rechts "gelijktijdig" de cijfers 1 tot en met 8 te zien zijn. Je moet dus beginnen met op het linkse display het cijfer 1 te tonen, 125 µs later het cijfer 2 op het display rechts ervan, enz. Na 1 ms ben je dan aan het meest rechts display gekomen (waar je cijfer 8 op toont) en vervolgens begin je opnieuw met een 1 op het linkse display.

Je gebruikt geen reset voor deze opgave, maar werkt met een initialisatie van je flip flops. Lees hiervoor zeker hoofdstuk 5.3 (Resets en Presets) in de cursus VHDL en/of bekijk de videoles over sequentiële logica.

Beschrijving

Het blokschema ziet er dus als volgt uit. Merk op dat de cathodes nu ook uit 8 bits bestaan, dus de puntjes van de displays (rechtsonder ieder cijfer) moeten ook aangestuurd en "uit" gezet worden.



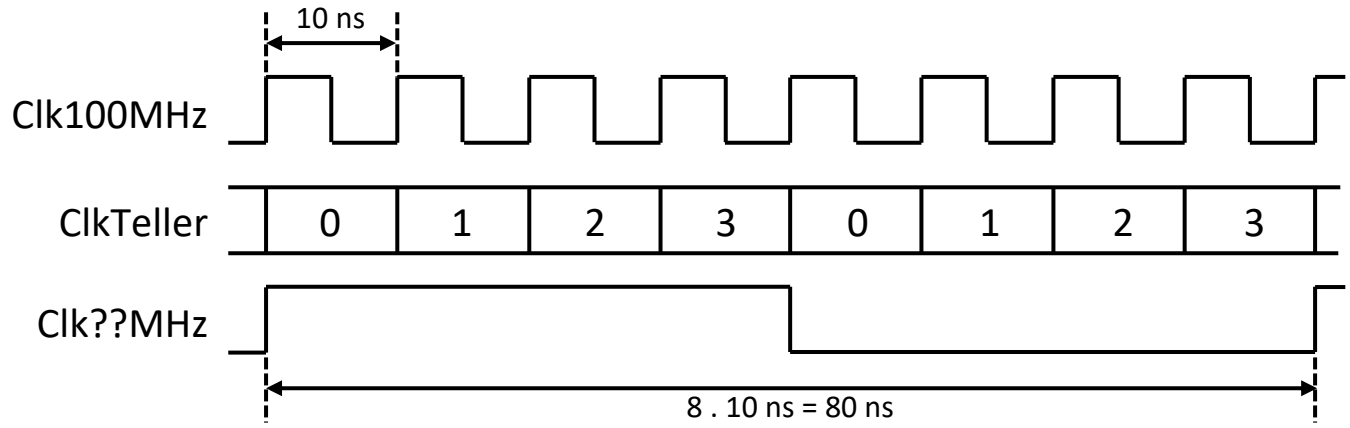
De opgave vraagt een klokfrequentie van 8 kHz, wat overeenkomt met een periode van $\frac{1}{8000} Hz = 125\ \mu s$. Zoals je in de handleiding van het Digilent bordje kan zien, is de FPGA met een 100 MHz kristal verbonden, waar je dus onvermijdelijk mee moet starten.

Om tot een klokfrequentie van 8 kHz te komen, zal je dus die 100 MHz klok moeten "delen", wat je kan doen aan de hand van een teller. Om een teller te beschrijven in VHDL moet je werken met een synchroon (of "sequentieel") proces. Let erop dat je voor het synchroon/sequentieel proces je VHDL in de juiste vorm schrijft! Hou je hierbij aan de regels die Xilinx in zijn handleiding aangeeft.

Op een correcte manier VHDL schrijven voor een synchroon/sequentieel proces (al of niet met (a)synchrone reset) en voor een combinatorisch proces is een belangrijke competentie voor dit vak. Zorg ervoor dat je dit onder de knie hebt! Indien dit niet het geval is, dan vraag je het aan mij.

Je zal hierover ondervraagd worden op een test en/of je verdediging van deze opgave.

Stel dat je de 100 MHz klok deelt aan de hand van een teller die van 0 tot en met 3 telt, waarbij je de "nieuwe" klok telkens omklapt bij waarde 3, dan krijg je dit tijdsdiagram:



Denk even na over de klokfrequentie die je hiermee zal bekomen en pas je teller vervolgens aan zodat je als resultaat een 8 kHz klok krijgt. Er staat ook een hoofdstuk over klokdeling in de cursus VHDL én er is ook een videoles over.

Beschrijving: klok in XDC bestand

In het .xdc bestand waar de verbindingen gelegd worden tussen je VHDL en de pinnen van de FPGA, zal je de volgende lijnen in commentaar vinden:

```
#set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports {
Clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports {Clk}];
```

Uiteraard moet je hier nu de bovenste lijn uit commentaar halen, want je wil de verbinding leggen tussen je klokingang *Clk* en pin E3 van de FPGA. Dit is hetzelfde als voor alle andere in- en uitgangen die je tot hiertoe gebruikt hebt. Een klok is echter iets speciaals in VHDL en dus ook in een FPGA, want er worden flip flops mee aangestuurd en zoals je weet uit de theorie, kunnen hier problemen ontstaan met *clock skew* (en nog andere waar we het niet over zullen hebben). Daarom is het belangrijk dat je ook de tweede regel uit commentaar haalt, zodat je Vivado duidelijk maakt dat dit geen gewone ingang is, maar een klokingang. Vivado zal dit signaal dan overal in de FPGA speciale aandacht geven omdat het een klok is en strengere eisen heeft dan een gewone ingang.

Beschrijving: klok in een testbench

Een klok in een testbench genereren kan op een zeer bondige manier:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity TB is
end TB;

architecture Behavioral of TB is
  signal Clk100MHz: std_logic := '0'; -- initialisatie op '0'
begin
  Clk100MHz <= not Clk100MHz after 5 ns; -- 5 ns is dus de halve periode
end Behavioral;
```

Merk op dat zo'n statement met "after x ns" enkel toegestaan is in VHDL voor simulatie. Dit is nooit synthetiseerbaar!

Beschrijving: simuleren met een sterk gedeelde klok

Als je even snel rekent, dan zou je tot de conclusie moeten komen dat je voor deze opgave minstens 100 000 klokflanken moet simuleren vooraleer je de volledige cyclus van je scorebord doorlopen hebt. Dat begint al aardig op te lopen en dat kan je simulatietijd ook stilaan wel beginnen verhogen. Je kan dit oplossen door **-enkel tijdens simulatie-** je klokdeling even te verminderen tot bijvoorbeeld “delen door 4”. Op die manier zal je simulatie veel sneller “klaar” zijn en kan je de correcte werking van je ontwerp sneller verifiëren. **Vergeet dan echter zeker niet om je klokdeling achteraf terug correct in te stellen, vooraleer je je opgave indient!**

Onthoud deze manier van simuleren ook voor je laatste opgave.