Opgave 4

Overzicht opdracht

Gebruik je kennis van voorgaande opgaven nu om een sterk vereenvoudigde *arithmetic logic unit* (ALU) te maken. Iedere processor in elke computer, laptop, smartphone en embedded system heeft minstens één ALU die verschillende aritmetische bewerkingen kan uitvoeren (zoals optellen, aftrekken, vermenigvuldigen, enz). Uiteraard wordt hier een veel eenvoudigere versie gevraagd. In latere vakken (zoals computerarchitectuur) worden processoren en ALU's in meer detail besproken.

Zoals in opgave 3 gebruik je de dip switches om beide positief binaire getallen in te lezen:

- SW15-11: getal A
- SW10-6: getal B

Maar ditmaal bepalen dip switches SW5 en SW4 de mode van de ALU:

- SW5-4 = 00 => A+B
- SW5-4 = 01 => A-B
- SW5-4 = 10 => A·B
- SW5-4 = 11 => $3 \cdot A^2 + 7 \cdot B^2$

De uitkomst van deze bewerkingen wordt opgeslagen in een getal van n bits[1], zodat voor de samengestelde bewerking van mode "11" genoeg bits voorzien zijn. Deze uitkomst wordt vervolgens aangeboden aan een "Scorebord" ontwerp zoals bij Mode "11" in opgave 2, waarbij het binair getal omgezet wordt in een BCD cijfer. Je zal voor deze opgave het n-bit getal moeten opsplitsen in vier BCD cijfers. **Bekijk zeker voorbeeldopgave 2 op de Mediasite, waarin een zeer gelijkaardig ontwerp wordt uitgewerkt**. De uitkomst moet je voorstellen met vier BCD cijfers: duizendtallen (DT), honderdtallen (HT), tientallen (TT) en eenheden (EH).

Wanneer het resultaat van de aftrekking (mode "01") negatief zou zijn, dan moeten de displays vier keer de letter E (van *error*) tonen, dus "EEEE", maar wel slechts 1 E tegelijk (zie volgende paragraaf).

Vermits de 7-segment displays op het bord gemeenschappelijke kathodes hebben, kan je echter slechts één 7-segment code tegelijk doorsturen naar één of meerdere displays. In latere opgaves zal je zien hoe je dit probleem moet oplossen (door zeer snel afwisselend naar de verschillende displays te sturen). Voor deze opgave moet je dit nog niet doen en gebruik je DIP switches SW3-2 om te bepalen welk display er aangestuurd wordt (inclusief voor de letter E: slechts één display tegelijk):

- SW3-2 = 00 => DT op meest linkse 7-segment display
- SW3-2 = 01 => HT op tweede 7-segment display van links
- SW3-2 = 10 => TT op derde 7-segment display van links
- SW3-2 = 11 => EH op vierde 7-segment display van links

De rechtse vier 7-segment displays moeten altijd uit staan. Vergeet dus niet de juiste anodes aan en uit te schakelen!

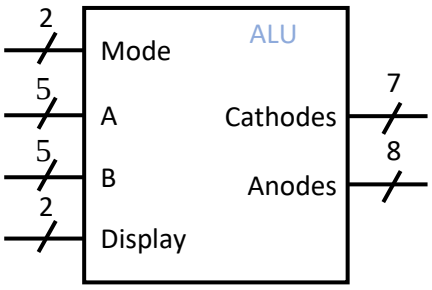
Voor deze opgave mag je kiezen of je hiërarchie gebruikt voor het Scorebord blok of niet. Vergeet echter niet om alle gebruikte VHDL bestanden in te dienen!

[1] Bereken op voorhand de waarde van n. Zorg ervoor dat je dit kan verklaren!

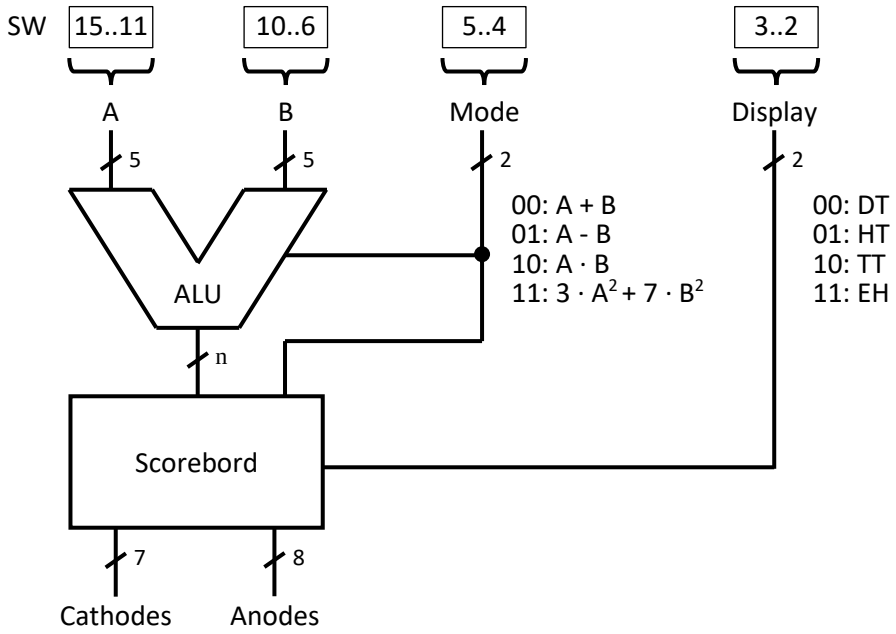
Merk op dat mode “11” perfect de kracht van hardwardesign illustreert: een gewone processor (software) zou deze bewerking stap voor stap moeten uitrekenen (eerst A^2, dan · 3, dan B^2, ·7 en dan pas de som van de twee) terwijl we dit in hardware in één keer kunnen doen. Hier gaan we het in het tweede jaar nog verder over hebben wanneer we meer focussen op hardware design en minder op de taal VHDL.

Beschrijving

De component ziet er dan als volgt uit, waarbij je uiteraard zelf de benamingen van de in- en uitgangen (en de component zelf) mag kiezen.



Het inwendige schema van het totaal zou er dus ongeveer als volgt moeten uitzien:



Om dezelfde reden als in voorbeeldopgave 2 en opgave 2 kunnen er geen verschillende cijfers tegelijkertijd op verschillende 7-segment displays van het Digilent Nexys A7 bord getoond worden. Daarom beslissen dip switches SW3 en SW2 over welk cijfer exact getoond moet worden. In de volgende opgave zal dan geleerd worden hoe dit probleem omzeild kan worden.

Gebruik je kennis uit opgave 2 en 3 om deze schakeling te ontwerpen. Bekijk zeker ook de voorgaande tutorials nog eens als je ergens vast moest zitten.

Als je meerdere VHDL bestanden gebruikt (met hiërarchie), vergeet dan niet dat je ze allemaal moet indienen!

Vergeet zeker het belangrijkste niet: debugging! Maak er een gewoonte van om je code ALTIJD te simuleren aan de hand van een testbench en gebruik deze simulatie om de functionele fouten uit je code te halen. De syntax fouten worden door Vivado ontdekt, maar de functionaliteit moet JIJ debuggen. Als je niet meer weet hoe dit moet: bekijk de videoles over debugging nog eens!

Als je daarna nog steeds vast zit, aarzel dan niet om vragen te stellen. Denk eraan: je mag ook altijd mailen als je hier thuis aan werkt.