Opdracht week 4 – Binair-naar-BCD omzetting

Inleiding

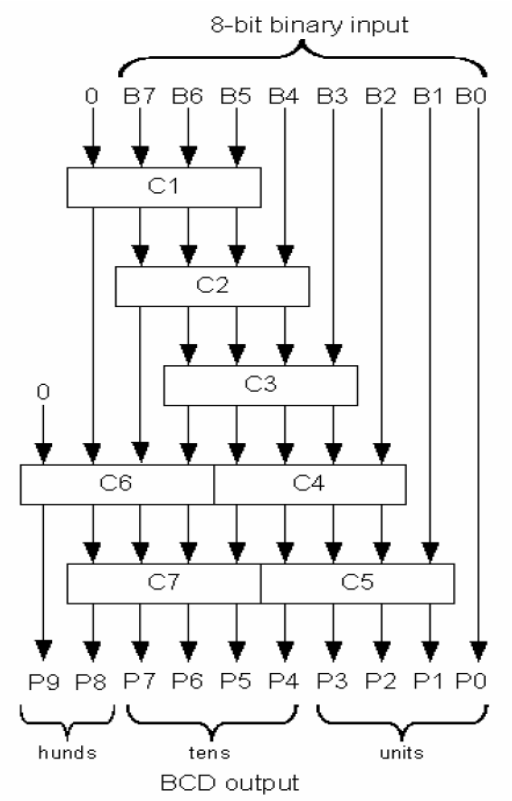
In de digitale techniek worden getallen opgeslagen in het binaire talstelsel. Rekenschakelingen zijn zo eenvoudig te ontwerpen. Helaas is het aflezen van binaire getallen voor de meeste mensen vervelend, zij hebben immers geleerd met decimale getallen te rekenen.

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[1]. Daarmee kan op een redelijk eenvoudige manier de omzetting uitgevoerd worden.

In figuur 1 is een schema te zien van een 8-bit binair getal naar een 10-bit BCD-getal.

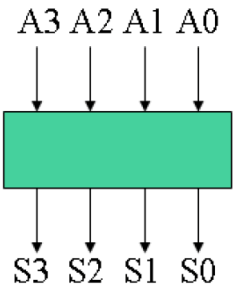


Figuur 1: *Implementie van het Double Dabble algoritme*

[1] http://en.wikipedia.org/wiki/Double_dabble

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.

De blokjes C1 t/m C7 in figuur 1 zijn identiek. Dit worden de zogenaamde *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 onderstaande figuur.



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 2: *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 moeten 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 1.

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 3 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.

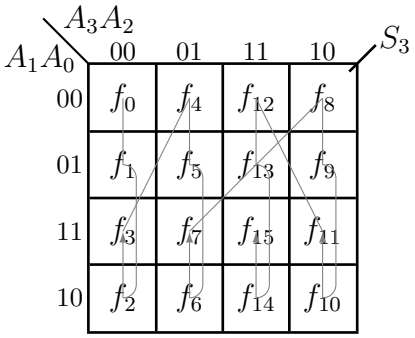
f) Test het ontwerp op een DE0-bord.

Opmerkingen

De ingangen van de schakeling (figuur 2: 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_1	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 3: *Karnaughdiagram voor vier variabelen.*

Implementatie Bin-to-BCD-omzetter

De Add3-module is nu klaar en kan gebruikt worden in het schema in figuur 1. 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 4.

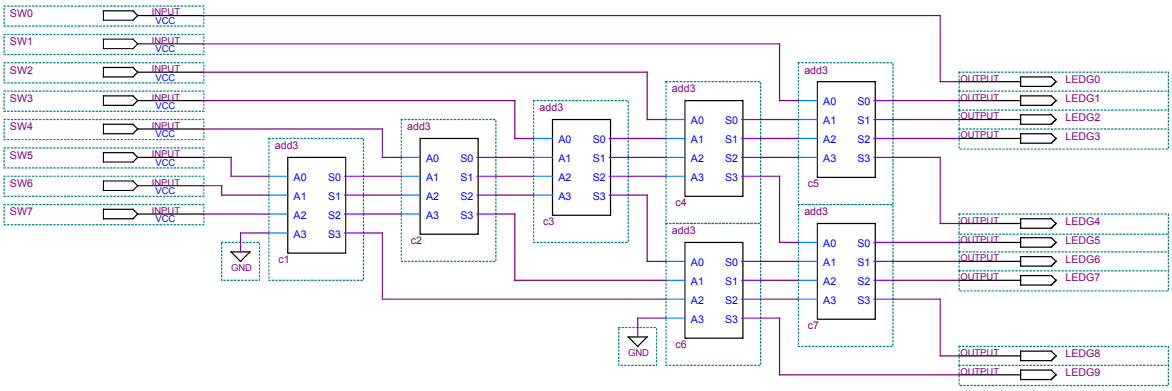
h) Kopieer het bestand `add3.bdf` uit het eerder gemaakte Add3-project naar het project `bin8_to_bcd10`. De naam van het bestand is al toegevoegd aan de lijst van projectbestanden.

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 2.

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 4: *Een 8-bit binair naar 10-bit BCD-omzetter*