Tijs Van Alphen

Opgave 4: Opteller

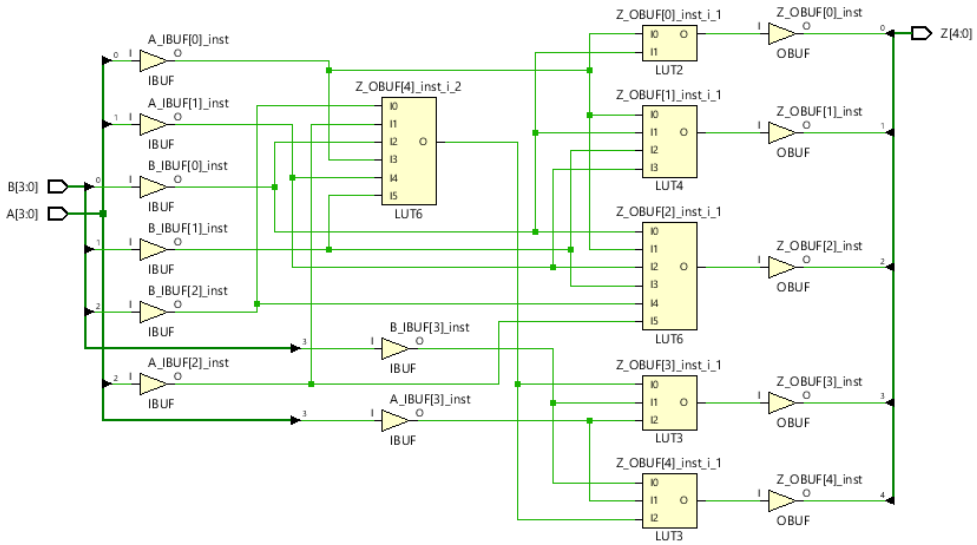
1) Het programma

(bij integer)

```
1  --Tijs Van Alphen
2
3  library IEEE;
4  use IEEE.STD_LOGIC_1164.ALL;
5  use IEEE.NUMERIC_STD.ALL;
6
7
8
9  entity Adder is
10     Port( A : in integer range 0 to 10;
11           B : in integer range 0 to 10;
12           Z : out integer range 0 to 10);
13
14 end Adder;
15
16 architecture Behavioral of Adder is
17
18 begin
19
20 proc: process (A,B)
21 begin
22 Z <=0;
23
24 Z <= A+B;
25
26 end process proc;
27
28 end Behavioral;
29
```

2) Verschil tussen oppervlakte en snelheid

Bij het synthetiseren naar oppervlakte en naar snelheid zien we dat dit alle twee hetzelfde schema geeft. Dit schema heeft dus zowel het kleinst mogelijk oppervlakte en werkt het snelst.



3) DSP blocks

Voor optellers gebruik Vivado DSP blocks niet als default. Het is echter mogelijk om dit aan de passen door het volgende commando binnen architecture te zetten (voor begin).

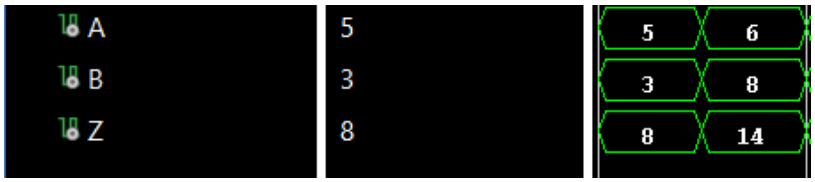
```
16  architecture Behavioral of Adder is
17
18  attribute use_dsp : string;
19  attribute use_dsp of Z : signal is "NO" ;
20
21  begin
```

Als het signaal “YES” is wordt er gebruik gemaakt van DSP blocks. Bij “NO” gebruik hij deze echter niet.

4) Overflow

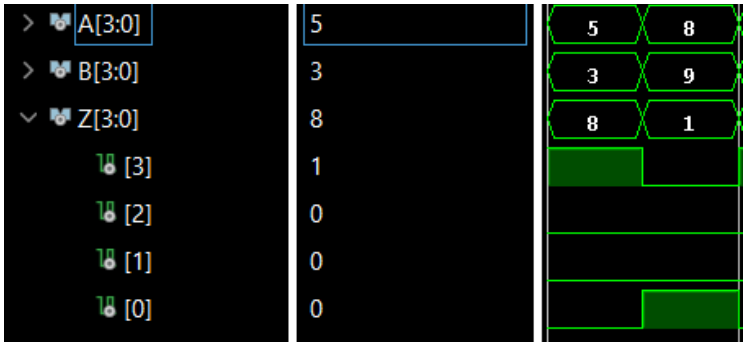
- Integer with range

  In het voorbeeld hebben alle integers een range 0 to 10. We zien als de som groter als 10 wordt deze nog steeds het correcte getal weergeeft. Enkel wanneer de ingelezen waarde groter dan hun range zijn krijg je een error.



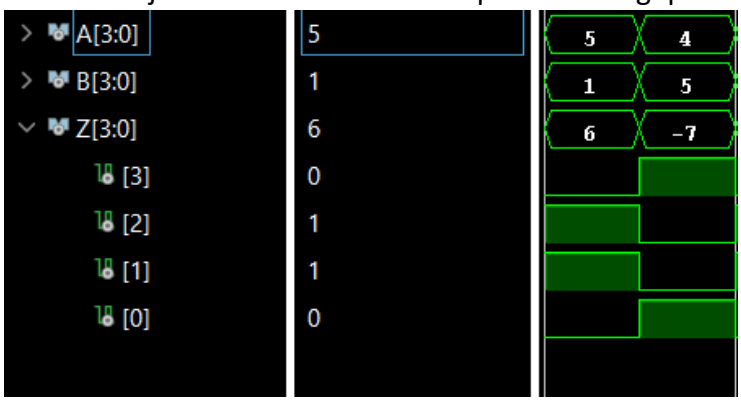
- Unsigned

  In dit voorbeeld zijn alle variabelen van de vorm unsigned (3 downto 0). Zolang we geen getal groter als 15 maken (15 grootste getal met 4 bits) krijgen we voor Z het juiste getal, zowel decimaal als binair. Wanneer het getal wel groter is, wordt er voor Z enkel het getal gegeven dat overeenkomt met de laatste 4 bits. In dit voorbeeld is 9+8=17 in binair 10001, Z wordt dus 0001 ofwel decimaal voor 1.

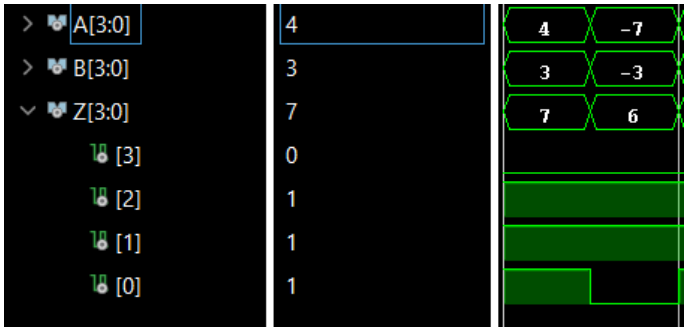


- Singed

  We gebruiken voor de variabele het type signed (3 downto 0). Uit het voorbeeld kunnen we opmerken dat zolang de som van de 2 getallen kleiner is als 8 (binair 1000) dat dit dezelfde uitkomst geeft als unsigned. Wanneer de som groter is hebben we een 1 vooraan staan en wordt er bij de laatste 3 bits 2's complement toegepast.



  We kunnen geen overflow creëren bij signed als we enkel positieve getallen optellen, het grootste dat we kunnen vormen is 2 keer 7 (binair 0111), dit geeft als resultaat -2. Het resultaat is echter niet juist omdat we om 14 te bekomen de eerste bit op 1 staat, maar er is ook niet echt sprake van overflow omdat we nog steeds genoeg bits hebben om het resultaat weer te geven. We hebben echter wel overflow bij signed enkel als de som kleiner is dan -8 (signed 1000). Als we bijvoorbeeld -7 + (-3) doen verwachten we -10, dit kan enkel met vijf bits worden weergegeven (10110). Omwille van dat er enkel 4 bits zijn, krijgt hij enkel de laatste 4 bits (0110). Het getal dat we krijgen voor Z wordt dus 6.



- Ik heb deze opstelling niet meer in de les kunnen testen. Mijn vermoedens zijn dat Z bij integer de waarde nul krijgt (die ik in het begin heb gegeven) als het getal te groot wordt. Bij de andere types vermoed ik dat deze hetzelfde gaat doen als in de simulatie.