Tijs Van Alphen

Opgave 5

Vermenigvuldiger

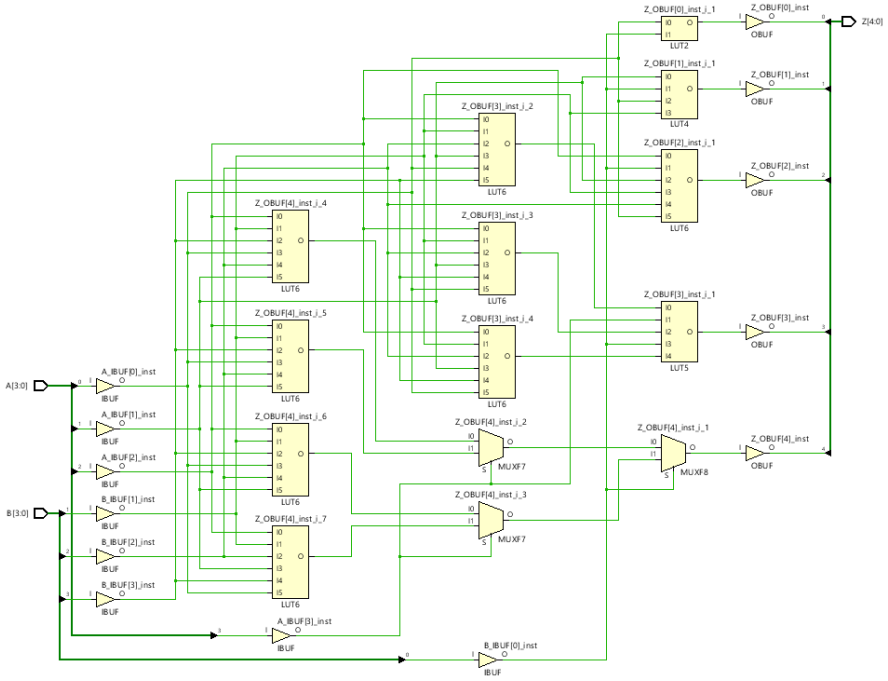
1) Het programma

(bij integer)

```
--Tijs Van Alphen

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;



entity Adder is
    Port( A : in integer range 0 to 10;
          B : in integer range 0 to 10;
          Z : out integer range 0 to 20);

end Adder;

architecture Behavioral of Adder is

attribute use_dsp : string;
attribute use_dsp of Z : signal is "NO" ;

begin

proc: process (A,B)
begin
Z <=0;

Z <= A*B;

end process proc;

end Behavioral;

```

2) Verschil tussen oppervlakte en snelheid

Bij het synthetiseren naar oppervlakte en naar snelheid zien we dat dit alle twee het zelfde schema geeft. Dit schema heeft dus zowel het kleinst mogelijk oppervlakte en de beste performance.



3) DSP blocks

Voor vermenigvuldigers gebruik Vivado net zoals bij de opteller niet standaard de DSP blocks . Het is echter op dezelfde manier aan te passen als bij de opteller d.m.v. onderstaand commando.

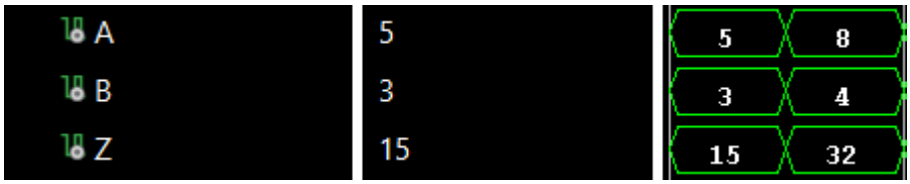
```
16  architecture Behavioral of Adder is
17
18  attribute use_dsp : string;
19  attribute use_dsp of Z : signal is "NO" ;
20
21  begin
```

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

4) Overflow

- Integer with range

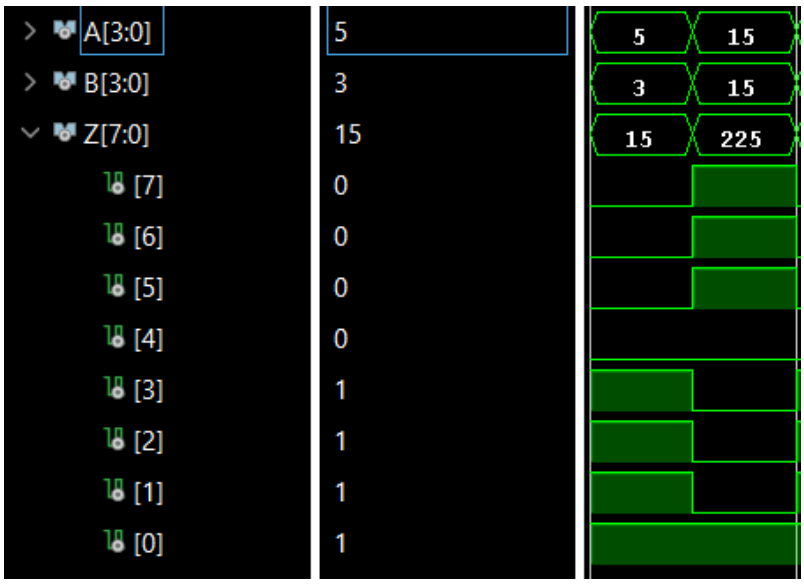
  In het voorbeeld hebben de ingelezen signalen range 0 to 10 en het uitgaande 0 to 20. Wanneer het product groter wordt dan 20 wordt deze nog steeds correct weergegeven. Enkel wanneer de ingelezen waarde groter dan hun range zijn krijg je een error.



We zien dat dit gelijkaardig is aan de opteller van opdracht 4.

- Unsigned

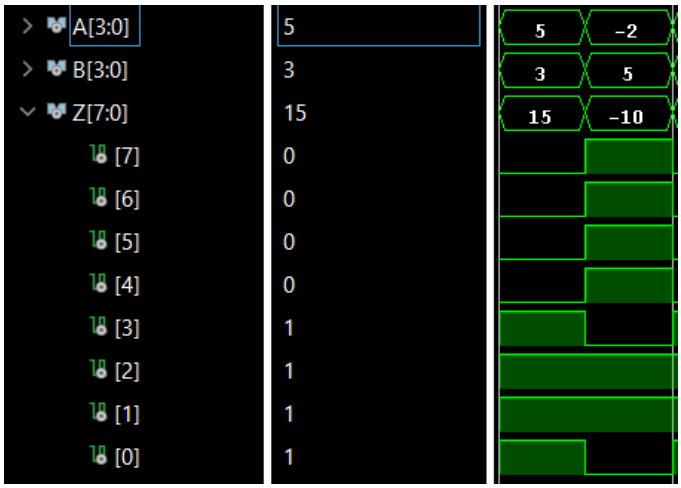
  Wanneer we vermenigvuldigen met unsigned is overflow niet mogelijk. Het is niet mogelijk om een simulatie te starten wanneer het aantal bits van het product niet gelijk is aan de som van de het aantal bits van de ingelezen signalen. In het voorbeeld zijn A en B van het type unsigned (3 downto 0) en Z moet zijn unsigned (7 downto 0).



  We zien dat er verschil is tussen de opteller en vermenigvuldiger. Bij de vermenigvuldiger moet het aantal bits van het product correct zijn voor je kan simuleren, maar bij de opteller maakte dat niet zoveel uit. Bij de opteller gaf deze gewoon het getal weer van dat gevormd werd door de bits die binnen het bereik vielen. Bijvoorbeeld als Z 3 downto 0 was werd getal 17 (10001) weergegeven als 1 (0001).

- Singed

  Net zoals bij unsigned is het niet mogelijk om te simuleren wanneer het aantal bits van het uitgaande signaal niet de som is van het aantal bits van de ingaande signalen.



We zien weer dat er verschil is tussen de opteller en de vermenigvuldiger. Bij de opteller kon je overflow creëren wanneer je 2 getallen optelde en de som kleiner was dan -8. Bij de vermenigvuldiger is overflow dus echter niet mogelijk.

5) 2^{de} macht

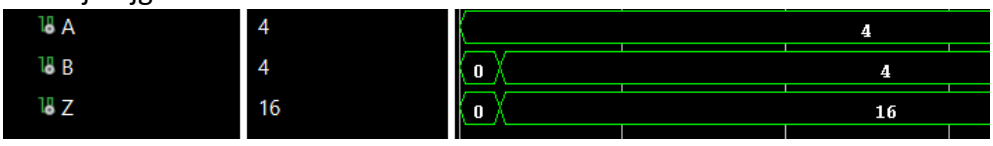
Je kan een exponent toevoegen door de vermenigvuldiging aan te passen naar Z <= A **2. Je kan ook A *A gebruiken. Er is ook de mogelijkheid om het programma te laten met Z <= A *B en in de testbench te schrijven

```
A <= 4;
B <= A;
```

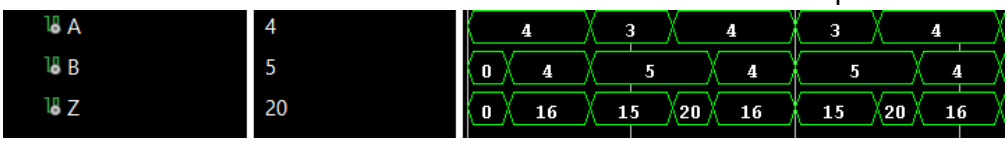
Maar wanneer je dit doet wordt B pas gelijkgesteld aan A de tweede keer als het proces wordt overlopen. Dit is makkelijk op te lossen op de volgende manier:

```
A <= 4;
wait for 50ns;
B <= A;
wait for 100ns;
```

Hierbij krijgen we :



Dit is echter een minder goede manier omdat we hier met een vertraging te maken hebben, als we daarna bijvoorbeeld met andere getallen werken loopt dat allemaal in elkaar over omdat B 50 nanoseconden achterloopt.



Tijs Van Alphen

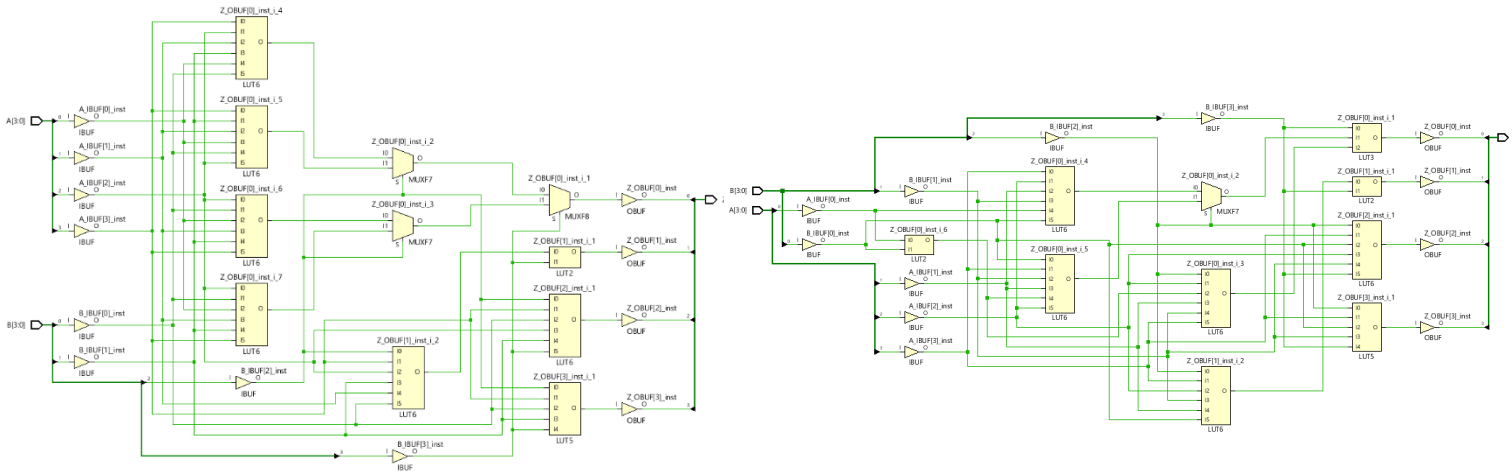
Deler

1) Het programma

```
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 Divisor is
10     Port( A : in unsigned (3 downto 0);
11           B : in unsigned (3 downto 0);
12           Z : out unsigned (3 downto 0));
13
14 end Divisor;
15
16 architecture Behavioral of Divisor is
17
18 attribute use_dsp : string;
19 attribute use_dsp of Z : signal is "NO" ;
20
21 begin
22
23 proc: process (A,B)
24 begin
25
26 Z <="0000";
27 Z <= A / B;
28
29 end process proc;
30
31 end Behavioral;
32
```

2) Verschillen tussen oppervlakte en snelheid

We zien bij de deler dat er wel een verschil is tussen synthetiseren naar snelheid en naar oppervlakte. Links naar snelheid en rechts naar oppervlakte.



3) DSP-blocks

Bij een deler is het niet mogelijk om gebruik te maken van DSP blocks. Wanneer we in het commando “YES” schrijven, zien we in het schema dat deze niet wordt aangepast.

4) Overflow

Er is geen overflow mogelijk bij de deler, het uitgaand signaal moet evenveel bits hebben als de het grootst aantal bits van de ingaande signalen. Enkel wanneer je deelt door nul komt er geen output bij de simulatie. Wanneer je een kommagetal bekomt wordt de floor van dit getal genomen (voorbeeld 2,8 wordt 2).