Univerza v Ljubljani
Fakulteta za elektrotehniko

Osnove kombinacijskih vezij.

1 Seštevalnik in primerjalnik

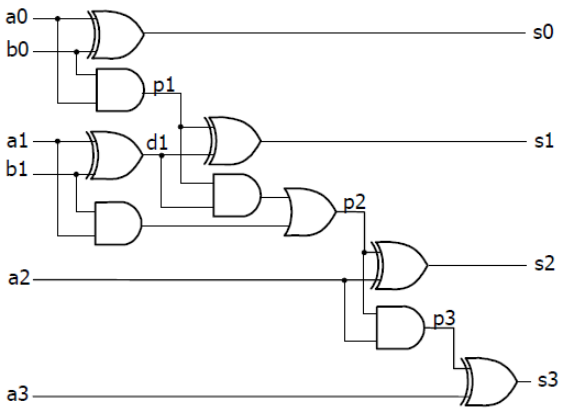
V jeziku VHDL naredi opis seštevalnika, ki sešteje 4-bitno (a) in 2-bitno vrednost (b).

a. Dopolni stavek `port` in deklaracije notranjih signalov ter dodaj stavke, ki opisujejo celotno vezje. Štiribitni signal deklariraj kot `std_logic_vector(3 downto 0)`, posamezni bit dobiš z indeksom v oklepaju, npr. a(0)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity add is
 port (
   a0 : in std_logic;
   b0 : in std_logic;
   s0 : out std_logic
 );
end add;

architecture logic of add is
 signal p1: std_logic;
begin
 s0 <= a0 xor b0;
 p1 <= a0 and b0;
end logic;
```



b. Preizkusi delovanje seštevalnika s simulacijo v programu Modelsim, kjer nastavi nekaj kombinacij na vhode in ugotovi ali je rezultat seštevanja pravilen. Uporabi skripto:

```
set DefaultRadix unsigned
force a 10#0, 10#1 @200 ps, 10#5 @300 ps, 10#10 @400 ps, 10#15 @500 ps
force b 10#0, 10#3 @100 ps
run 600 ps
```

Ugotovi zakaj je rezultat zadnjega seštevanja napačen!

c. Deklariraj knjižnico `IEEE.numeric.std.all;` in spremeni podatkovni tip vektorjev v `unsigned`, namesto prireditvenih stavkov z logičnimi izrazi pa uporabi izraz z operatorjem +. Preveri delovanje s simulacijo.

d. Popravi opis seštevalnika, tako da bo izračunal 5-bitno vsoto, pri kateri ne bo prišlo do preliva. V računskem izrazu uporabi funkcijo za razširitev signala na 5 bitov: `resize(a, 5)`
in preveri delovanje s simulacijo. Nato spremeni še vhod b, da bo 4-biten in podatkovni tip signalov na signed. Preizkusi simulacijo z negativnimi vrednostmi:

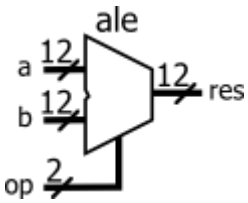
```
set DefaultRadix signed
force a 10#0, 10#1 @200 ps, 10#5 @300 ps, -10#7 @400 ps, -10#1 @500 ps
force b 10#0, -10#1 @100 ps
run 600 ps
```

e. Dodaj med priključke vezja enobitne izhode Z,N in L in naredi opis primerjav s pogojnim prireditvenim stavkom **when ... else**.

- Z, ki se postavi na '1', kadar je vsota enaka 0
- N, ki se postavi na '1', kadar je vsota manjša od 0
- L, ki se postavi na '1', kadar je a < b

f. Razmisli, ali za izhod nega potrebuješ stavek **when...else** s primerjalnim operatorjem ali bi ga lahko opisal enostavneje.
Razmisli, kako bi z operacijo odštevanja naredil primerjavo a < b !

2 Aritmetično logična enota




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity ale is
   port ( a, b : in signed(11 downto 0);
          op : in unsigned(1 downto 0);
          res: out signed(11 downto 0));
end ale;

architecture opis of ale is
 constant add : unsigned(1 downto 0):= "00";
 constant sub : unsigned(1 downto 0):= "01";
 constant anda: unsigned(1 downto 0):= "10";
 constant ora : unsigned(1 downto 0):= "11";
begin

end opis;
```

a. Opiši vezje 12-bitne aritmetično logične enote, ki izvaja štiri operacije nad predznačenima vhodnima signaloma a in b. Operacije določa 2-bitni signal op: "00" določa seštevanje, "01" odštevanje, "10" logično in operacijo (**and)** in "11" logično ali operacijo (**or)**. Delovanje ALE opiši s stavkom **when...else**. Za bolj pregleden opis vezja definiraj konstante, ki določajo operacijo.

Določi primerno simulacijsko skripto in preizkusi delovanje ALE na simulatorju.

b. Spremeni opis ALE, tako da boš uporabil procesni stavek in stavek **case**. Katere signale je potrebno navesti v oklepaju pri procesnem stavku ?

```
case op is
  when add =>
  ...
  when sub =>
  ...
end case;
```

c. Dodaj v opis ALE enobitni izhodni signal C, ki predstavlja izhodni prenos pri seštevanju in odštevanju.
Dodaj še izhod Z, ki predstavlja ničelno zastavico. Signal naj dobi vrednot '1', kadar je rezultat operacije enak 0.