Andrej Trost

Načrtovanje digitalnih el. sistemov

Načrtovanje vezij v jeziku VHDL

Podatkovni tipi in modeli na nivoju RTL

http://lniv.fe.uni-lj.si/ndes.html

Model obnašanja na nivoju RTL

- Algoritem določa obnašanje vezja
  - pri algoritmu ni določen časovni potek izvajanja
  - sinteza vezij iz algoritma je precej zahtevna
- Na nivoju RTL je določeno obnašanje vezja ob urinih ciklih
  - sinteza vezij iz RTL opisa je danes običajna
- Modeliramo prenos podatkov med pomnilnimi elementi in (kombinacijske) transformacije

Primer: šifriranje podatkov

- Blokovni algoritem s 4 iteracijami šifriranja

```
int main(void)
{
  char a, b, a_nov, b_nov;
  char k[] = {0xff, 0x0f, 0x03, 0x30};
  cin >> a;
  cin >> b;
  for (int i=0; i<=3; i++) {
    a_nov = b;
    b_nov = (b + k[i]) ^ a;
    a = a_nov;
    b = b_nov;
  }
 cout << a << b;
}
```

C++

Zanka za šifriranje v jeziku VHDL

- Jedro algoritma tvori zanka

```
for i in 0 to 3 loop
  a_nov := b;
  b_nov := (b + k(i)) xor a;
  a := a_nov;
  b := b_nov;
end loop;
```

VHDL
Behavioral

- Kako naredimo te operacije z vezjem?

1. vezje: razvijemo zanko

- rezultat operacij vsakokrat shranimo v nov signal

```
sifr1: process(vhod, a0, b0, a1, b1, a2, b2, a3, b3, a4, b4 )
 begin
    a0 <= vhod(15 downto 8);
    b0 <= vhod(7 downto 0);
    a1 <= b0;
    b1 <= (b0 + k(0)) xor a0;
    a2 <= b1;
    b2 <= (b1 + k(1)) xor a1;
    a3 <= b2;
    b3 <= (b2 + k(2)) xor a2;
    a4 <= b3;
    b4 <= (b3 + k(3)) xor a3;
end process;
```

VHDL

RTL

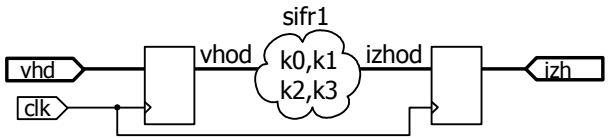
Signali v kombinacijskem vezju

- Rezultate zaporednih operacij nad istimi podatki vedno zapišemo v nov signal
  - v nasprotnem primeru lahko naredimo kombinacijsko zanko!

```
nasicenje: process(a,b,rez)
begin
  rez <= ('0' & a)+('0' & b);
  if rez(8)='1' then
     rez <= "011111111";
  end if;
end process;
```

```
if rez(8)='1' then
  rez2 <= "011111111";
else
  rez2 <= rez;
end if;
```

Rezultat 1. vezja



CPLD Xilinx XC95288XL-10

- Površina: 56 makrocelic, 32 flip-flopov
- Frekvenca ure: 26.8 MHz (53.6 MByte/s)

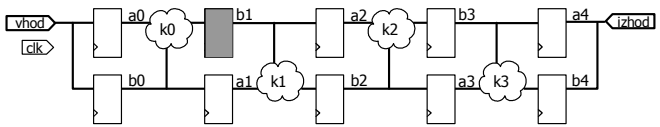
Kako povečati hitrost vezja?

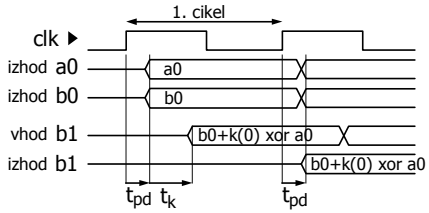
2. vezje: sinhrona izvedba (5 stopenj)

```
sifr2: process(clk)
 begin
  if rising_edge(clk) then
    a0 <= vhod(15 downto 8);
    b0 <= vhod(7 downto 0);
    a1 <= b0;
    b1 <= (b0 + k(0)) xor a0;
    a2 <= b1;
    b2 <= (b1 + k(1)) xor a1;
    a3 <= b2;
    b3 <= (b2 + k(2)) xor a2;
    a4 <= b3;
    b4 <= (b3 + k(3)) xor a3;
  end if;
 end process;
```

- operacije izvajamo ob fronti ure
- rezultat vsakokrat shranimo v nov signal
- 5-stopenjski cevovod: vsaka prireditev je register!

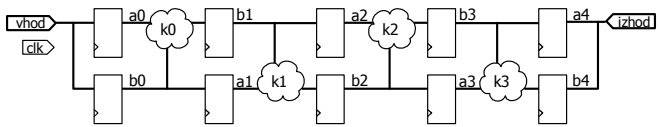
Delovanje sinhronega vezja





- opazujmo vhod in izhod registra b1
- frekvenca ure je omejena s $t_{pd}+t_k+t_s$
- vezje naj deluje le na eno fronto ure!

Rezultat 2. vezja



CPLD Xilinx XC95288XL-10

- Površina: 80 makrocelic, 80 flip-flopov
- Frekvenca ure: 90.9 MHz
- Zmogljivost: 181.8 MByte/s

Signali in spremenljivke

- Signali predstavljajo povezave v vezju
  - v sinhronih procesih (I. ali II. oblika) vsak signal, ki mu prirejamo vrednost pomeni register ali flip-flop
  - signali niso primerni za izračun vmesnih rezultatov s kombinacijsko logiko
- Spremenljivke uporabljamo v procesnem okolju za izračun vmesnih rezultatov
  - ob izračunu spremenljivka takoj dobi novo vrednost

Spremenljivke (variable)

- Deklaracija spremenljivke:

```
p: process (clk)
   variable a1,b1: std_logic_vector(7  downto 0);
begin
```

- Prireditveni operator (:=)

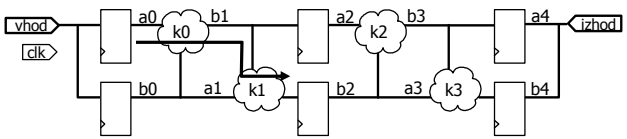
```
a1 := b0;
b1 := (b0 + k(0)) xor a0;
```

3. vezje: sinhrono s 3 stopnjami

```
if rising_edge(clk) then
  a0 <= vhod(15 downto 8);
  b0 <= vhod(7 downto 0);
  a1 := b0;
  b1 := (b0 + k(0)) xor a0;
  a2 <= b1;
  b2 <= (b1 + k(1)) xor a1;
  a3 := b2;
  b3 := (b2 + k(2)) xor a2;
  a4 <= b3;
  b4 <= (b3 + k(3)) xor a3;
end if;
```

- operacije izvajamo ob fronti ure
- prireditev spremenljivki ne predstavlja registra
- 3-stopenjski cevovod

Rezultat 3. vezja

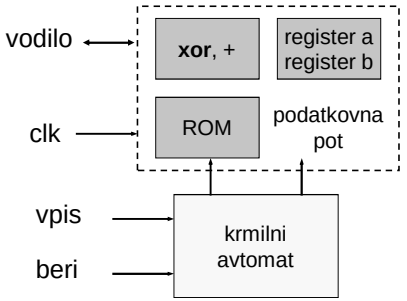


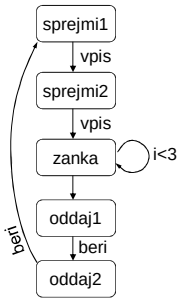
CPLD Xilinx XC95288XL-10

- Površina: 54 makrocelic, 48 flip-flopov
- Frekvenca ure: 46.1 MHz
- Zmogljivost: 92.2 MByte/s

4. vezje: sekvenčni procesor

- Vezje razdelimo na podatkovni in kontrolni del





Opis krmilnega avtomata

- Definiramo naštevni podatkovni tip in signal za stanje avtomata

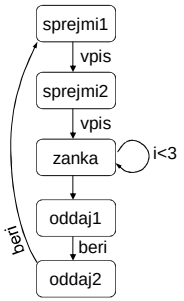
```
type tip_stanja is (sprejmi1, sprejmi2, zanka, oddaj1, oddaj2);
signal stanje: tip_stanja;
```

- Definiramo polje za ključe (ROM pomnilnik) in števec (i)

```
type keys is array(0 to 3) of std_logic_vector(7 downto 0);
constant k: keys := (x"ff", x"0f", x"03", x"30");
signal i: integer range 0 to 4;
```

Opis prehajanja stanj avtomata

```
avt: process(clk)
begin
  if rising_edge(clk) then
    case stanje is
      when sprejmi1 =>        -- čakaj na prvi vpis
        if vpis='1' then
          stanje <= sprejmi2;
        end if;
      when sprejmi2 =>        -- čakaj na drugi vpis
        ...
      when zanka =>           -- naredi 4 iteracije
        i <= i + 1;
        if i=3 then
          stanje <= oddaj1;
        end if;
        ...
```



Podatkovni del vezja

```
sifr4: process(clk)
begin
  if rising_edge(clk) then
    if vpis='1' and stanje=sprejmi1 then
      a <= vodilo;
    end if;
    if vpis='1' and stanje=sprejmi2 then
      b <= vodilo;
    end if;
    if stanje=zanka then
      a <= b;
      b <= (b + k(i)) xor a;
    end if;
  end if;
end process;
```

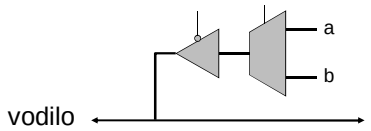
register a
register b

xor, +

Opis vmesnika

- Vmesnik za dvosmerno vodilo

```
vodilo <= a when br='1' and stanje=oddaj1 else
          b when br='1' and stanje=oddaj2 else
          "ZZZZZZZZ";
```



Rezultat in primerjava

CPLD Xilinx XC95288XL-10

- Površina: 31 makrocelic, 22 flip-flopov
- Frekvenca ure: 84.7 MHz (10,6 Mbyte/s)

Vezje	Površina	f [MHz]	Mbyte/s
1. vezje	56	26.8	53,6
2. vezje	80	90.9	181,8
3. vezje	54	46.1	92,2
4. vezje	31	84.7	10,6