Laboratorij za načrtovanje integriranih vezij

Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

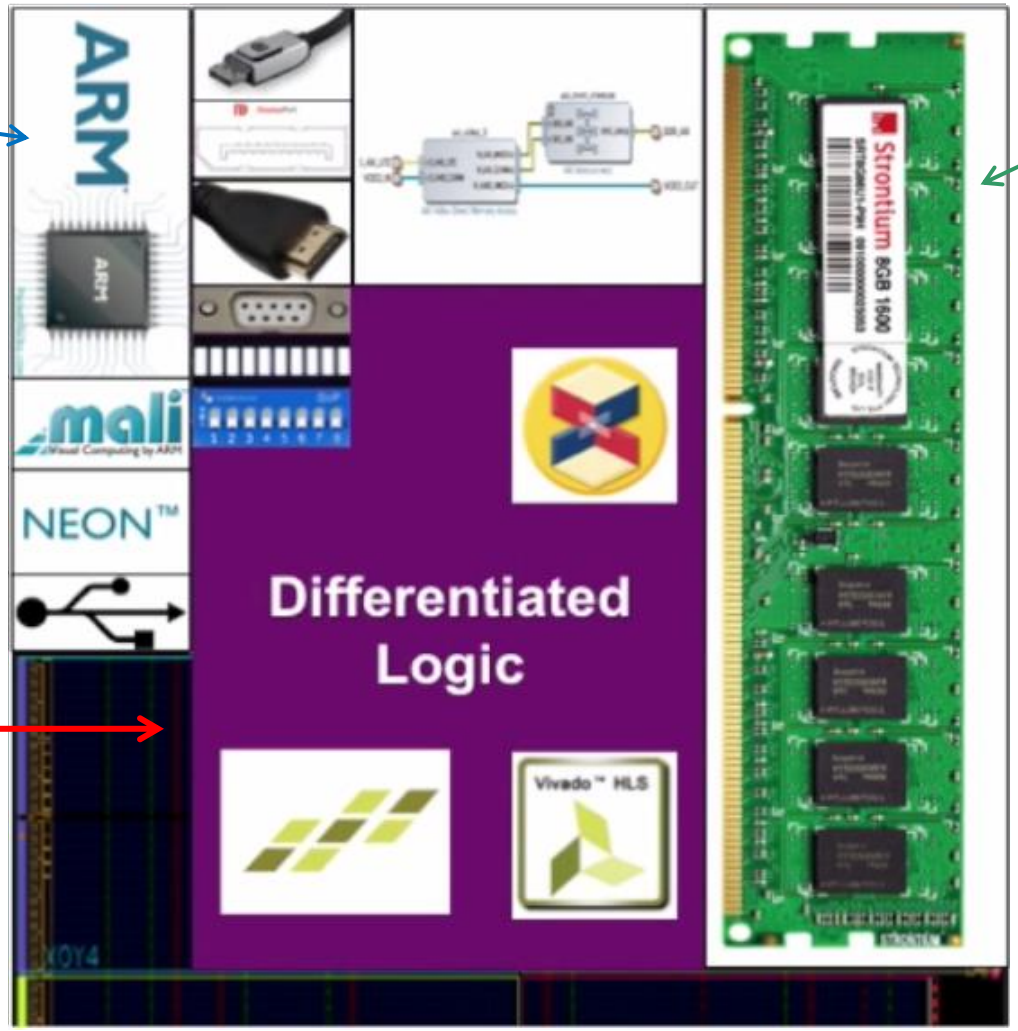
Načrtovanje digitalnih elektronskih sistemov

Visokonivojska sinteza vezja

Načrtovanje programirljivih sistemov na čipu

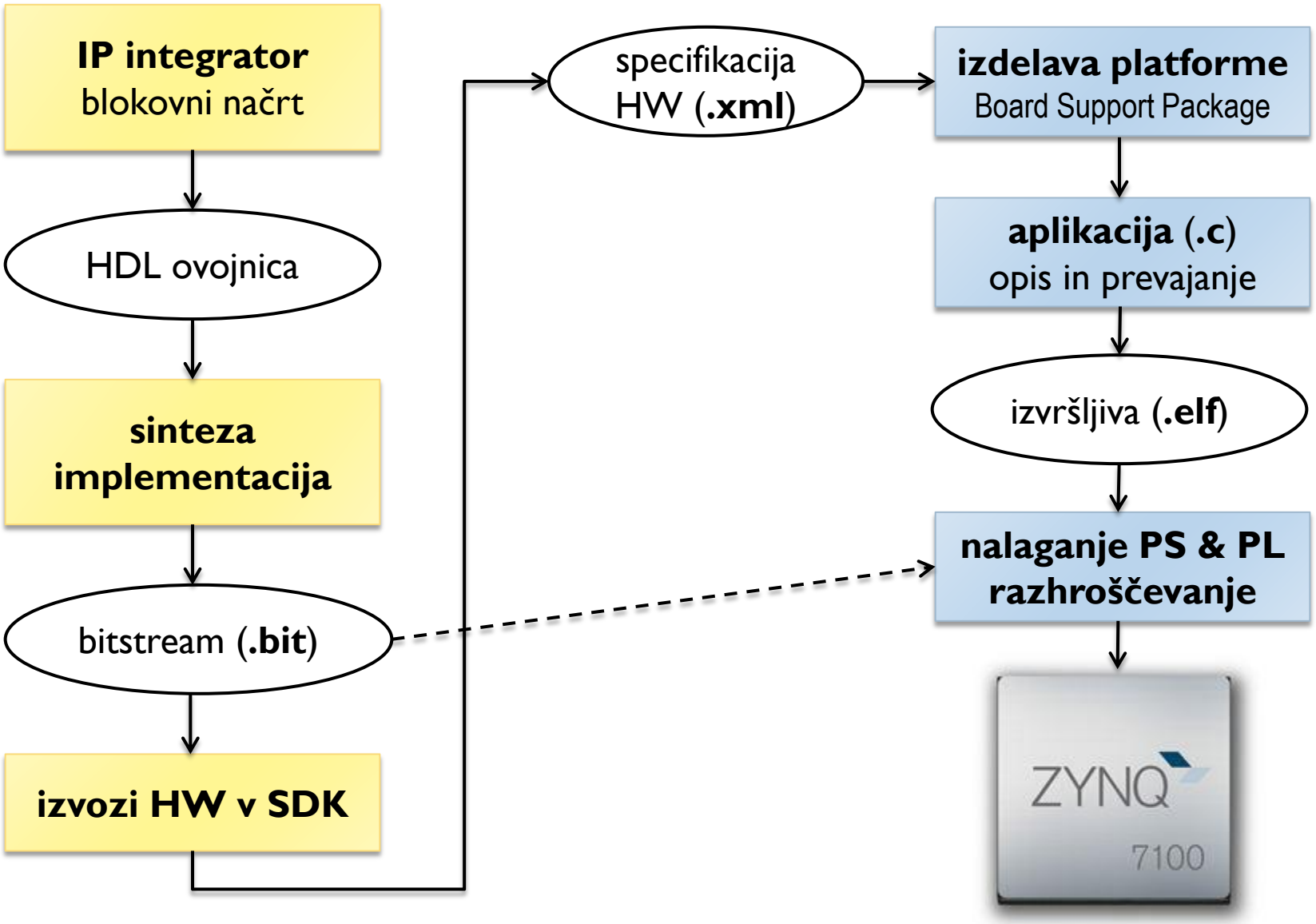
jedra IP
CPU, RAM…

spec. logika
HDL, HLS



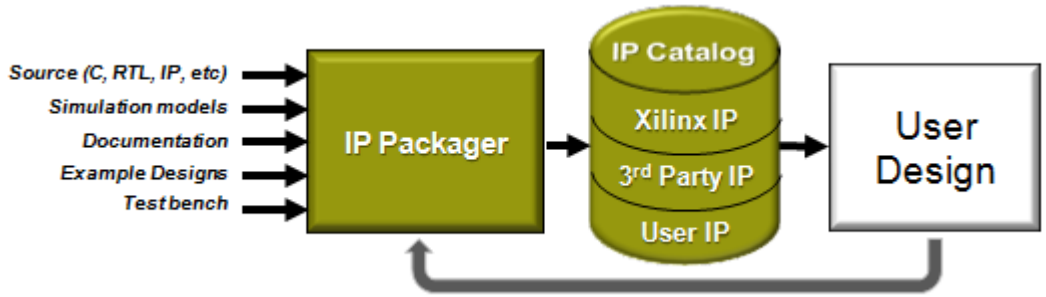
vmesniki
krmilnik SDRAM
HDL, natančni
časovni parametri

Programska orodja: Vivado in SDK



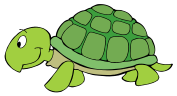
Komponente IP

1. izdelava IP iz HDL (opis na nivoju RTL v VHDL, Verilog)
   - opis logičnega vezja v HDL
   - IP Packager naredi definicijo IP (*.xml) – IP-XACT (IEEE-1685)

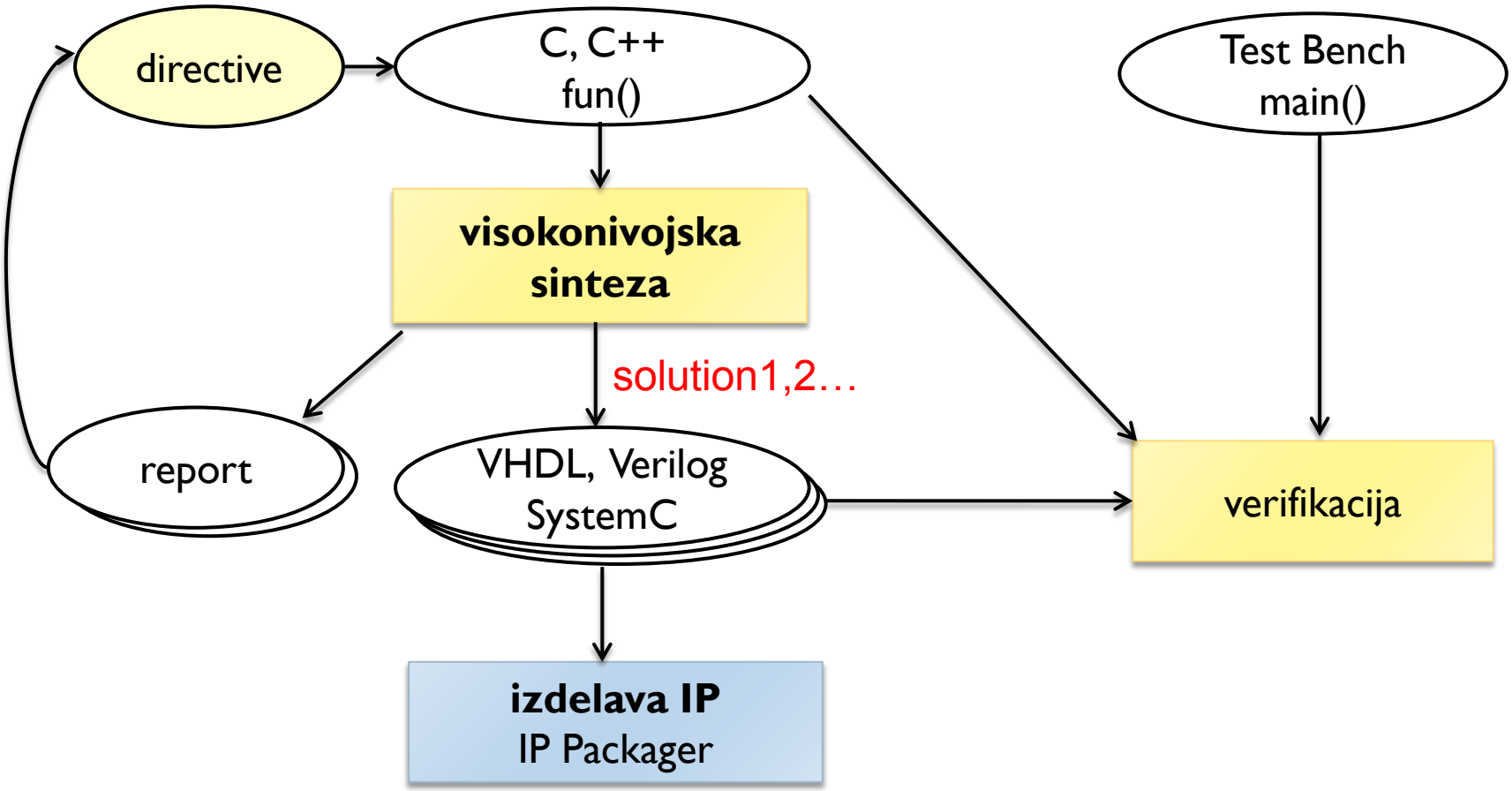


2. sinteza IP iz opisa na višjem nivoju abstrakcije
   - npr. program v jeziku C pretvorimo v RTL opis vezja

Načrtovanju namenske logike

- optimalna izvedba 
  - iskanje optimalne izvedbe pri natančno definiranem cilju
  - odlična izvedba zahteva veliko časa
    - zadnjih 10% zahteva 90% časa
- hitrost 
  - doseže le dovolj dobro izvedbo
  - 10x hitreje !
- cilj v realnih projektih ni vedno natančno definiran
  - razmislek ni dovolj za optimalno izvedbo, potrebni so poskusi
  - najpomembneje je pravočasno dobiti dovolj dober rezultat

Visokonivojska sinteza komponente (C/C++)

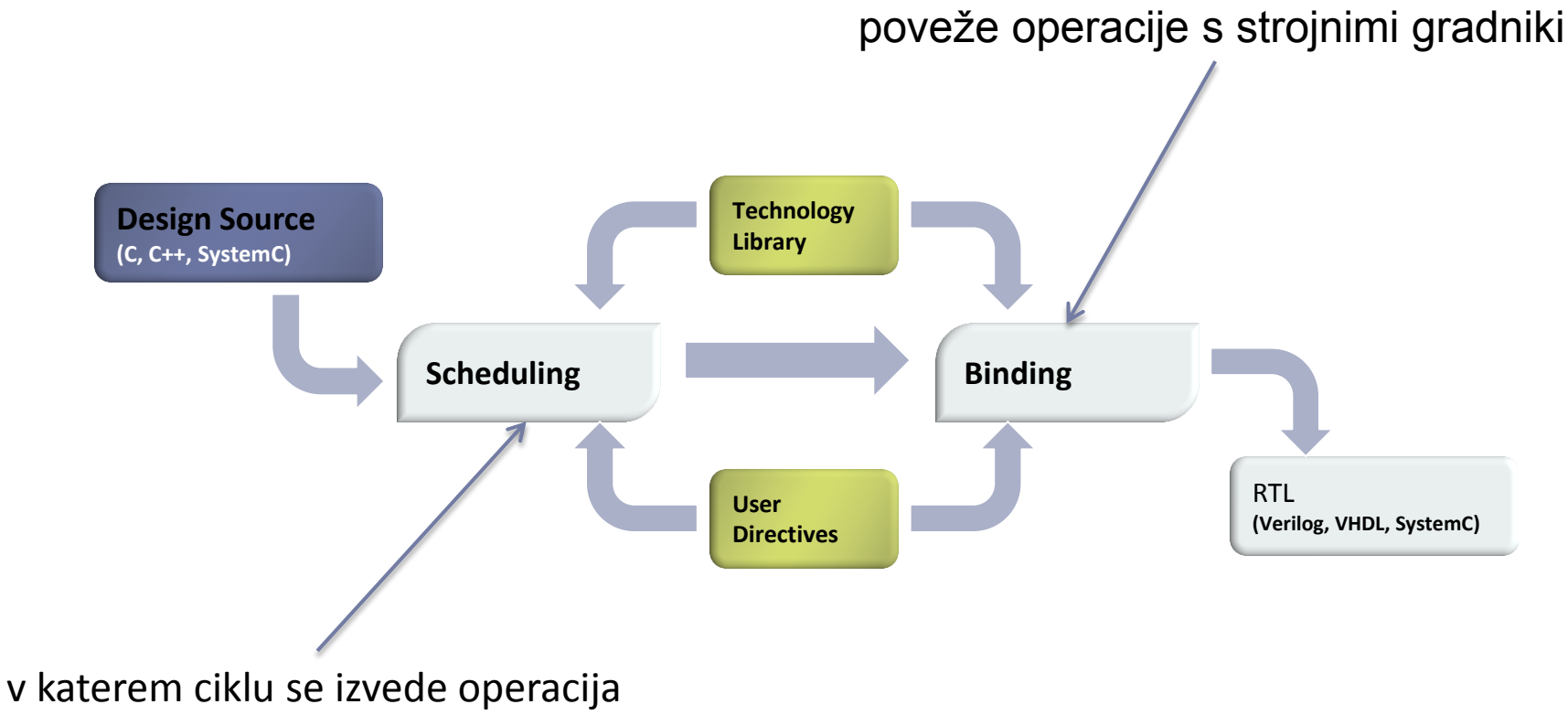


Prednosti visokonivojske sinteze

- opis algoritma v ANSI-C ali C++ z orodji GCC
- podpira realna števila (float IEEE-754) in operacije
- učinkovito dela s poljubno natančnimi celoštevilskimi vrednostmi (fixed point)
- avtomatizirana časovna delitev in delitev virov
- …

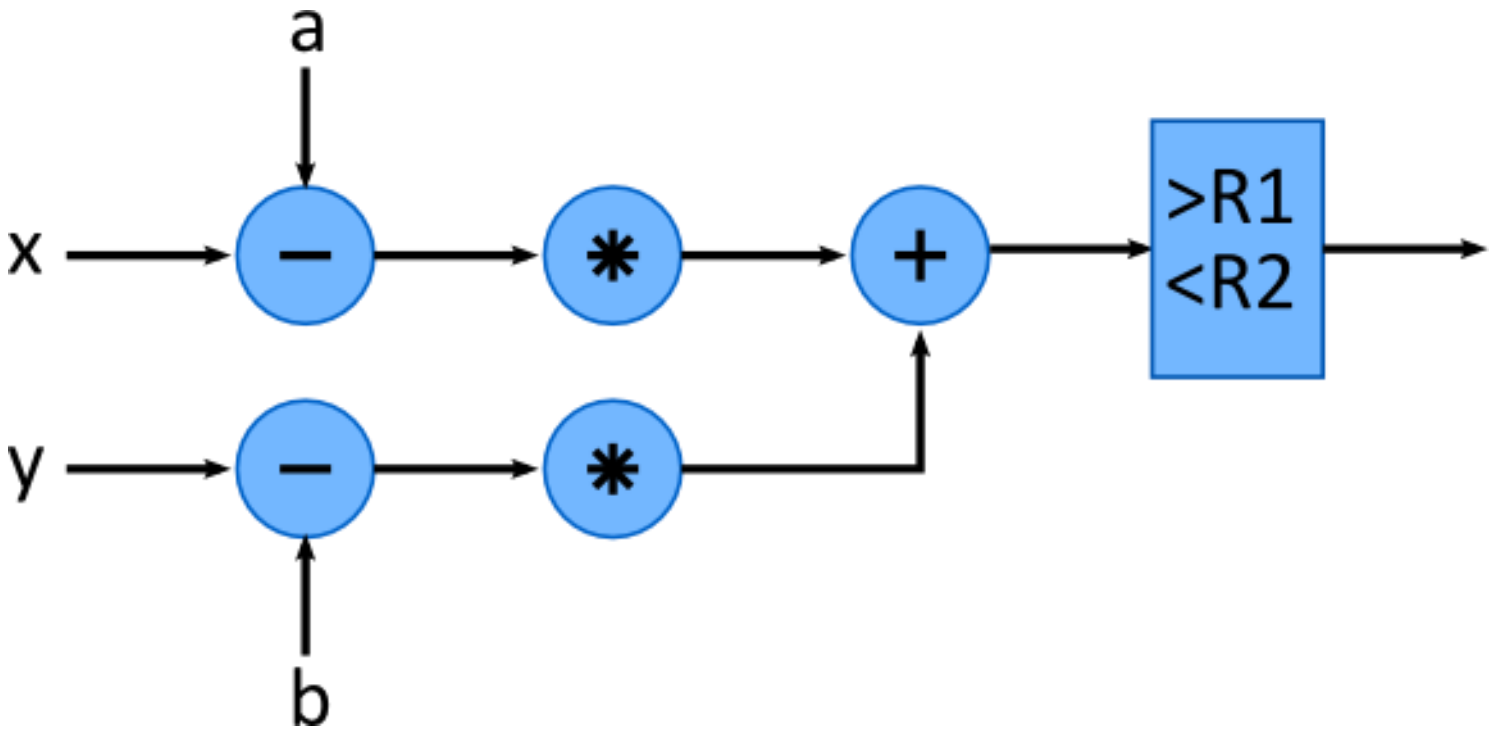
Kvaliteta rezultatov primerljiva s kodo na nivoju RTL !

Postopek visokonivojske sinteze



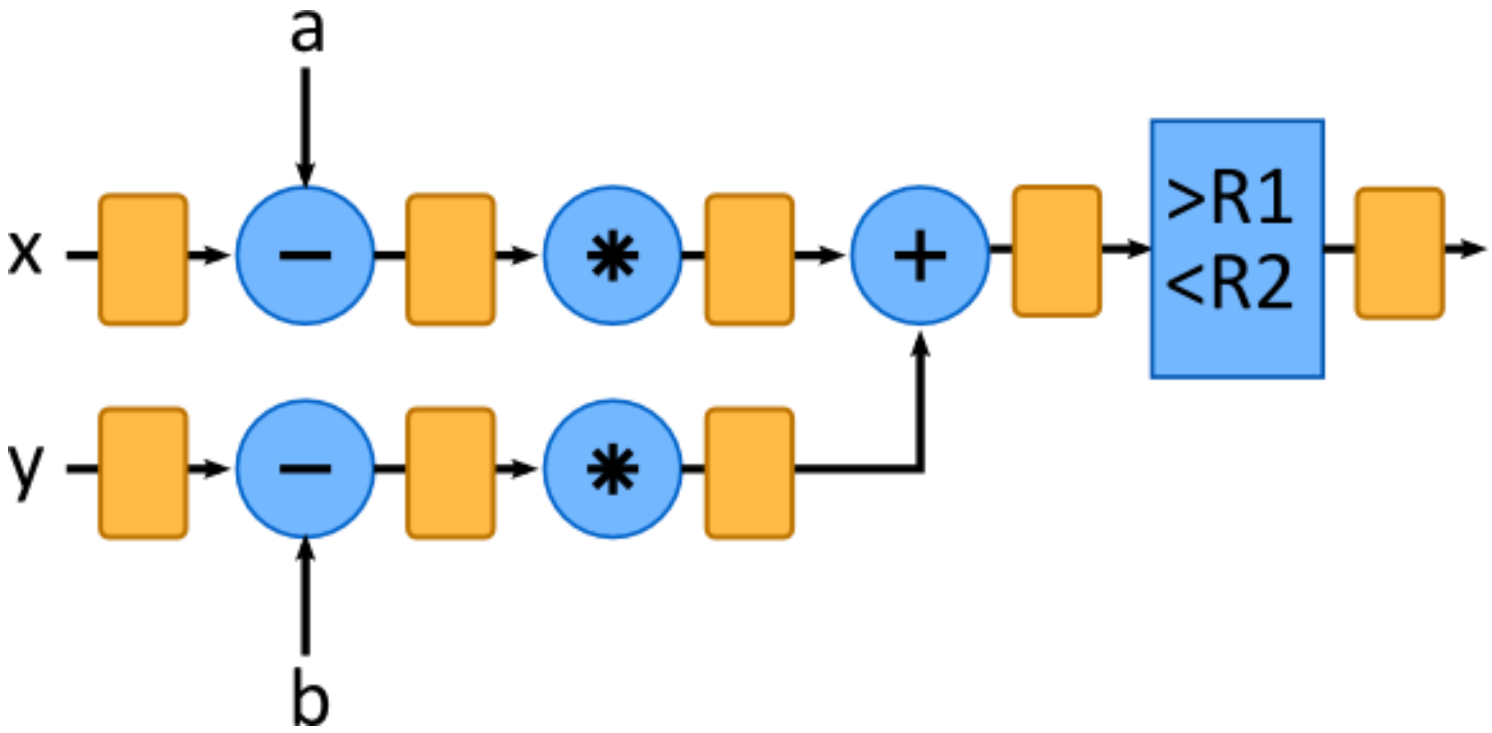
Primer: sinteza operatorjev iz opisa v jeziku C

```
x2 = (x-a)*(x-a);
y2 = (y-b)*(y-b);
if ((x2 + y2 > R1) && (x2 + y2 < R2)) p = 1;
else p = 0;
```



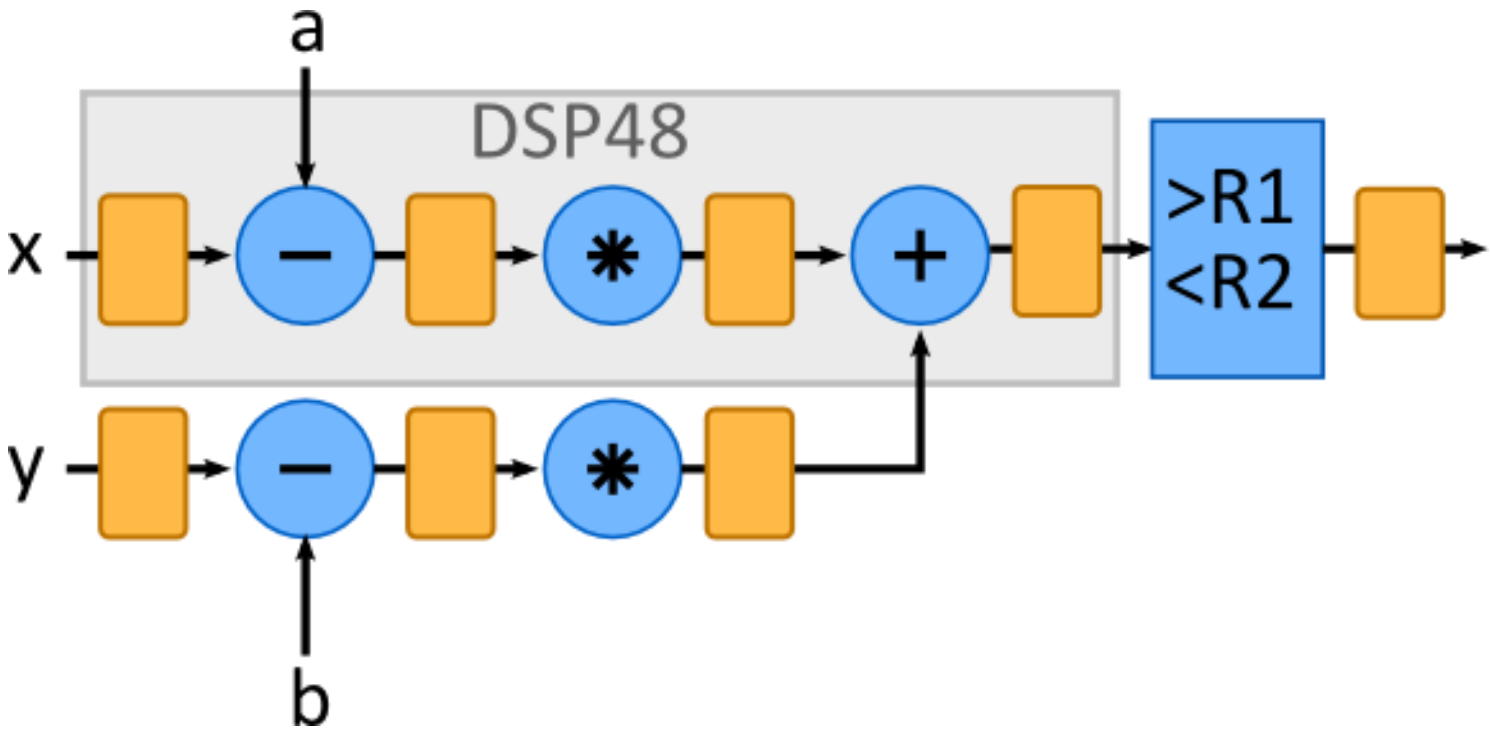
Primer: časovni potek izvajanja

```
x2 = (x-a)*(x-a);
y2 = (y-b)*(y-b);
if ((x2 + y2 > R1) && (x2 + y2 < R2)) p = 1;
else p = 0;
```



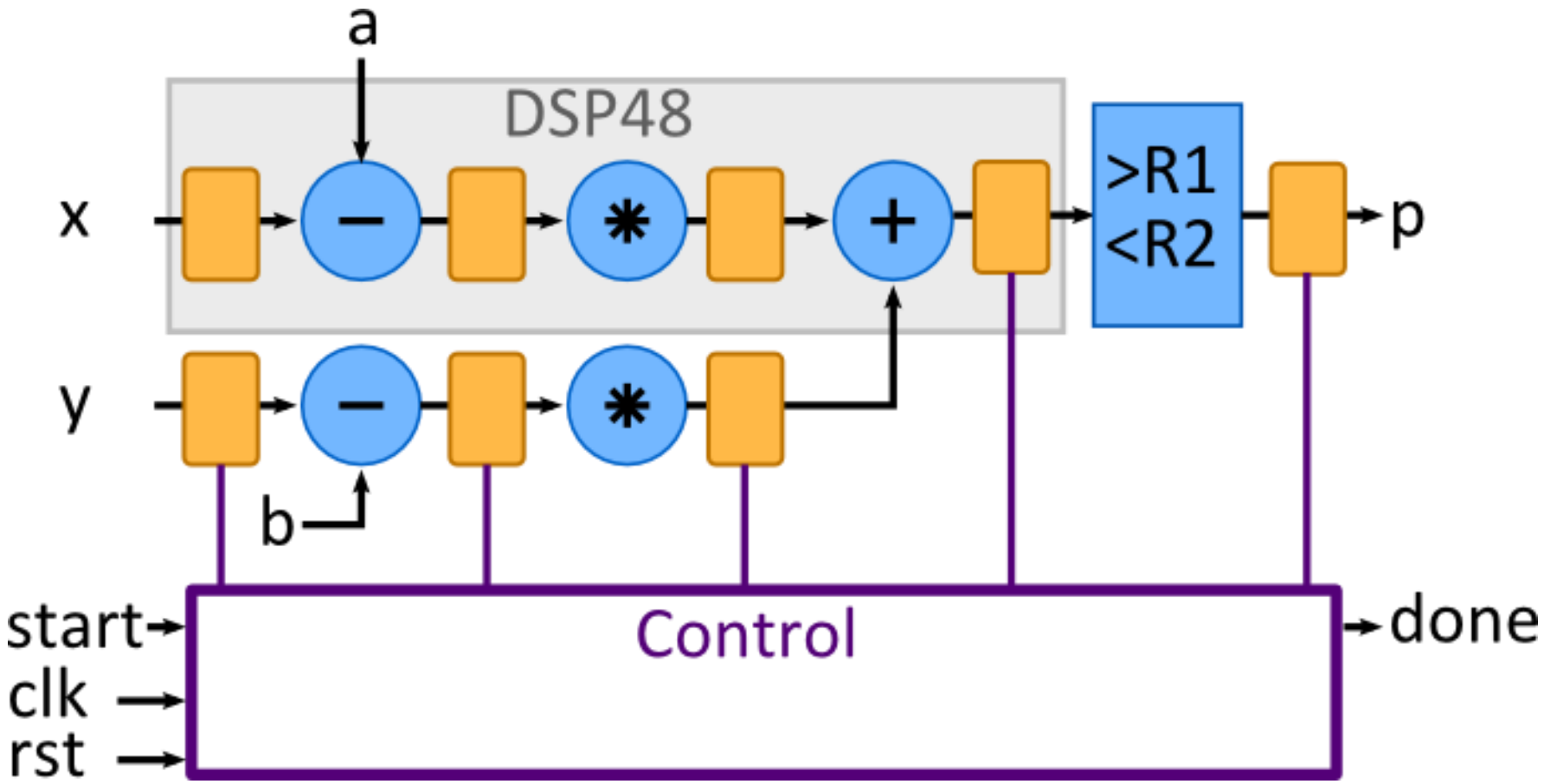
Primer: povezava s strojnimi gradniki

```
x2 = (x-a)*(x-a);
y2 = (y-b)*(y-b);
if ((x2 + y2 > R1) && (x2 + y2 < R2)) p = 1;
else p = 0;
```



Primer: sinteza krmilne logike

```
x2 = (x-a)*(x-a);
y2 = (y-b)*(y-b);
if ((x2 + y2 > R1) && (x2 + y2 < R2)) p = 1;
else p = 0;
```



Optimizacija

- postopek optimizacije določajo direktive
  - razvijanje zank (Loop unrolling, hitrejša izvedba / večje vezje)
  - cevljenje (Loop pipelining, učinkovita obdelava toka podatkov)
  - deljenje pomnilnikov (Memory partitioning, uporabi več blokov RAM)
  - delitev virov (Resource allocation, npr. omejimo št. seštevalnikov)

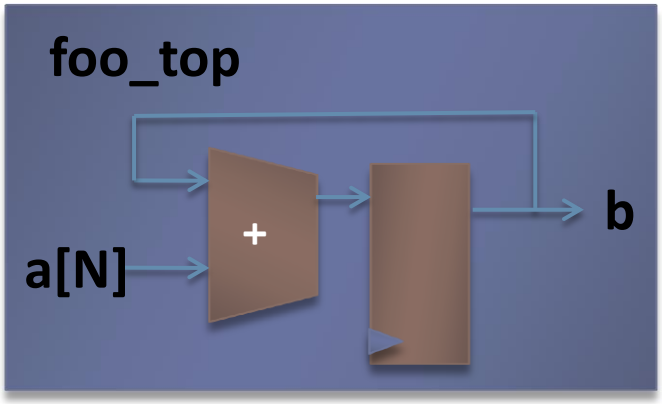
- orodje Vivado pozna ~30 direktiv
  - direktive sinteze so ločene od algoritma
  - lahko dosežemo zmogljivost, ki je primerljiva z ročno izdelanim opisom na nivoju RTL

Optimizacija zank

- privzeto delovanje sinteze je, da naredi zanke v obliki zaporedja operacij
  - vsaka iteracija zanke se izvede z istimi strojnimi komponentami

```
void foo_top (…)
   for(i=3;i>=0;i--)
        b = a[i] + b;
```

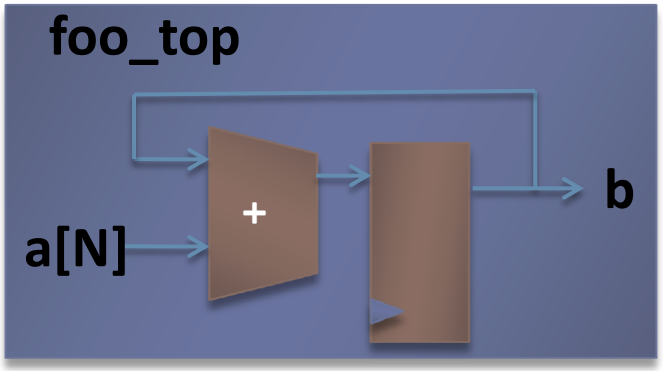
Standard HLS



Optimizacija zank: razvijanje

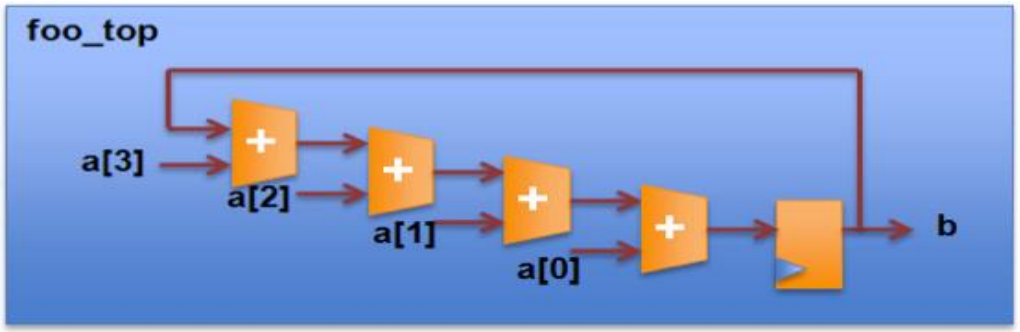
- izkoristi paralelizem v iteracijah zank
  - naredi strojne komponente za vsako iteracijo
  - upošteva podatkovne odvisnosti

Standard HLS



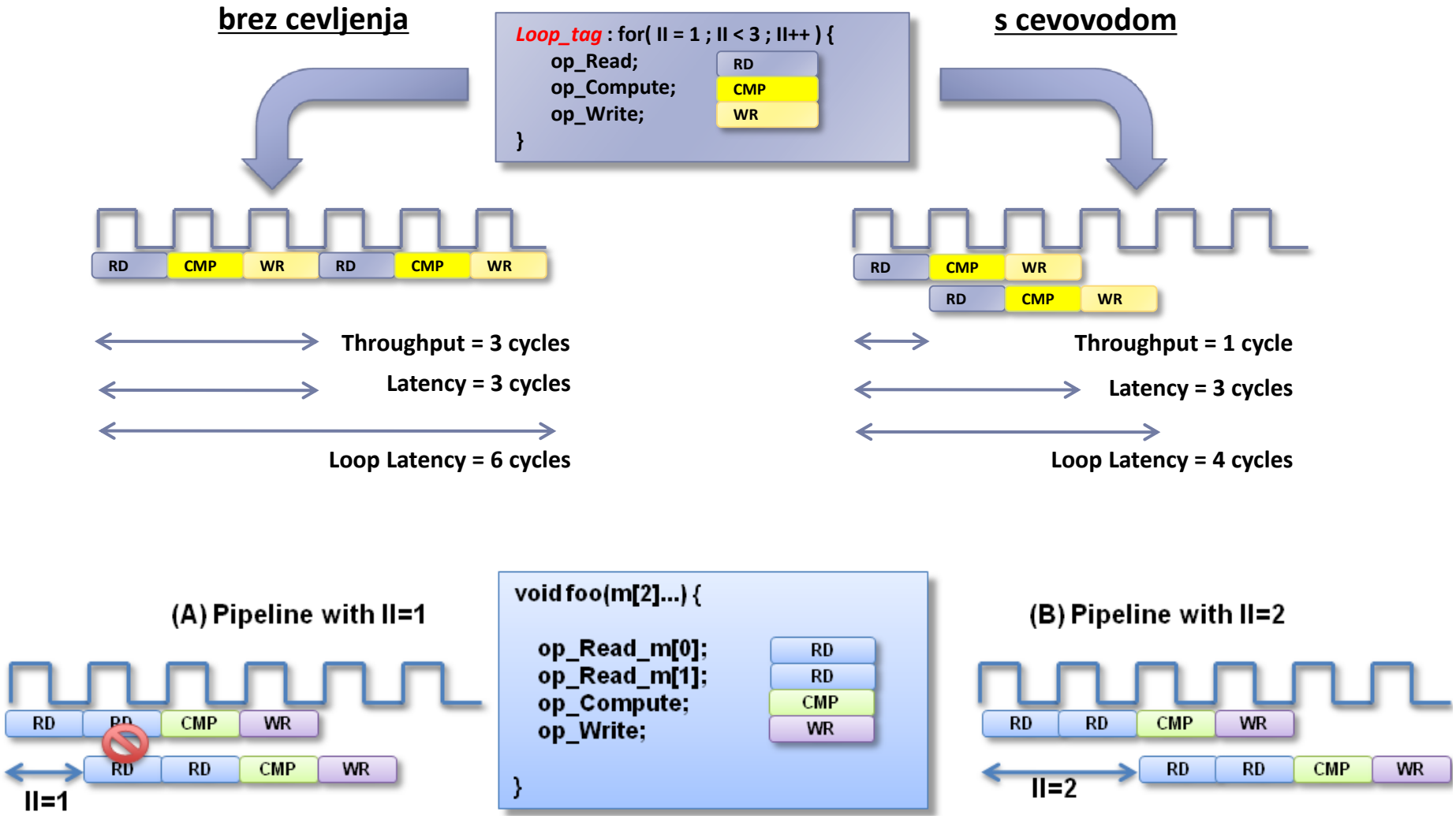
izvaja se 4 cikle

HLS & Unrolling



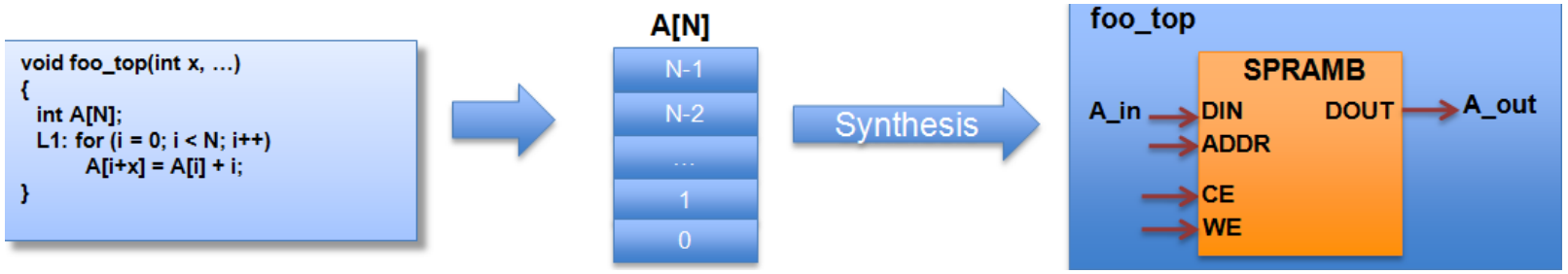
izvaja se le 1 cikel

Optimizacija zank: cevovod

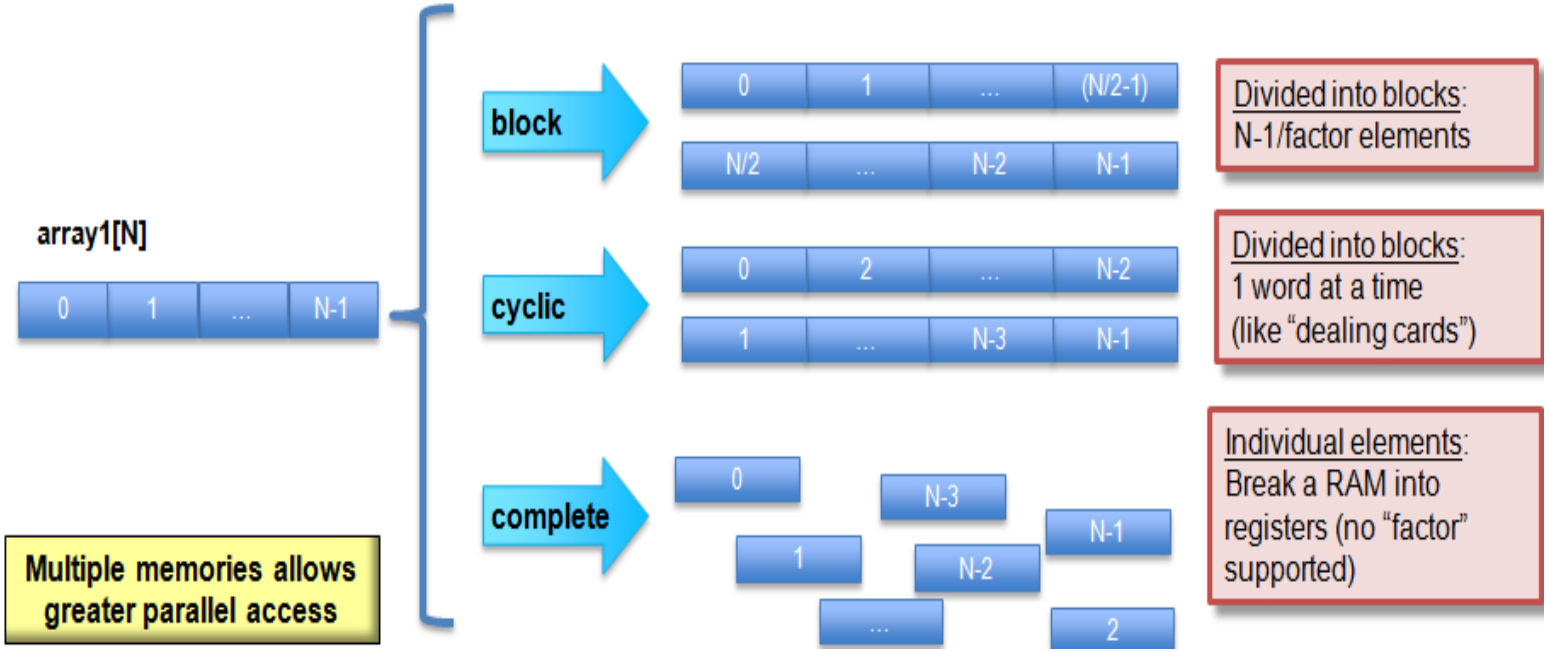


Optimizacija podatkovnih zbirk (pomnilnikov)

- podatkovna zbirka (tabela) je v pomnilniku



- sinteza razdeli pomnilnik na več komponent s hkratnim dostopom



Optimizacija strojnih virov

- uporabnik določi ***implementacijo***
  - označi operator
  - izbere izvedbo (npr. IP jedro) za določen operator (npr. *thisMult)*

```
thisMult = b[i] * c[i];
a[i] = thisMul;
```

jedro	opis
Mul	kombinacijski množ.
Mul3s	3-stanjski cevovodni
MulnS	HLS določi št. stanj