



# Projektovanje digitalnih sistema

Tipovi podataka



## ■ Verilog: primarni tipovi podataka

### ■ Verilog ima dva primarna tipa podataka:

- Nets – predstavlja strukturnu vezu između komponenata
- Registers – predstavlja promjenljive za smještanje/čuvanje podataka

## ■ Verilog: net (mreža)

- Na mrežama se nalaze vrijednosti postavljene od strane izlaza kola na koje su povezane
- Svaki od tipova mreže modeluje različitu tehnologiju izrade hardvera (PMOS, NMOS, CMOS, ...)

Net tip podataka	Funkcionalnost
wire, tri	Linija za povezivanje – nema posebnu funkciju
wor, trior	Ožičeno ILI – wired-OR (modeluje ECL kola)
wand, triand	Ožičeno I – wired-AND (modeluje <i>open-collector</i> kola)
tri0, tri1	pull-down ili pull-up kad nije pogonjen
supply0, supply1	Na liniji se nalazi konstantna logička nula ili jedinica ( <i>supply</i> jačina)
triereg	Zadržava prethodnu vrijednost kada je kolo u stanju visoke impedanse

- Od svih tipova najčešće se koristi **wire**

## ■ Verilog: net – primjer za *wired-OR*

```
module test_wor();  
  wor a;  
  reg b, c;  
  assign a = b;  
  assign a = c;  
  initial begin  
    $monitor("%g a = %b b = %b c = %b", $time, a, b, c);  
    #1 b = 0;  
    #1 c = 0;  
    #1 b = 1;  
    #1 b = 0;  
    #1 c = 1;  
    #1 b = 1;  
    #1 b = 0;  
    #1 $finish;  
  end  
endmodule
```

IZLAZ:

```
0 a = x b = x c = x  
1 a = x b = 0 c = x  
2 a = 0 b = 0 c = 0  
3 a = 1 b = 1 c = 0  
4 a = 0 b = 0 c = 0  
5 a = 1 b = 0 c = 1  
6 a = 1 b = 1 c = 1  
7 a = 1 b = 0 c = 1
```

## ■ Verilog: net – primjer za *wired-AND*

```
module test_wand();  
wand a;  
reg b, c;  
assign a = b;  
assign a = c;  
initial begin  
    $monitor("%g a = %b b = %b c = %b", $time, a, b, c);  
    #1 b = 0;  
    #1 c = 0;  
    #1 b = 1;  
    #1 b = 0;  
    #1 c = 1;  
    #1 b = 1;  
    #1 b = 0;  
    #1 $finish;  
end  
endmodule
```

IZLAZ:

```
0 a = x b = x c = x  
1 a = 0 b = 0 c = x  
2 a = 0 b = 0 c = 0  
3 a = 0 b = 1 c = 0  
4 a = 0 b = 0 c = 0  
5 a = 0 b = 0 c = 1  
6 a = 1 b = 1 c = 1  
7 a = 0 b = 0 c5 = 1
```

## ■ Verilog: net – primjer za *tri*

```
module test_tri();  
tri a;  
reg b, c;  
assign a = (b) ? c : 1'bz;  
initial begin  
    $monitor("%g a = %b b = %b c = %b", $time, a, b, c);  
    b = 0;  
    c = 0;  
    #1 b = 1;  
    #1 b = 0;  
    #1 c = 1;  
    #1 b = 1;  
    #1 b = 0;  
    #1 $finish;  
end  
endmodule
```

IZLAZ:

```
0 a = z b = 0 c = 0  
1 a = 0 b = 1 c = 0  
2 a = z b = 0 c = 0  
3 a = z b = 0 c = 1  
4 a = 1 b = 1 c = 1  
5 a = z b = 0 c = 1
```

## ■ Verilog: net – primjer za *tri*reg

```
module test_tri(reg);
    trireg a;
    reg b, c;
    assign a = (b) ? c : 1'bz;
    initial begin
        $monitor("%g a = %b b = %b c = %b", $time, a, b, c);
        b = 0;
        c = 0;
        #1 b = 1;
        #1 b = 0;
        #1 c = 1;
        #1 b = 1;
        #1 b = 0;
        #1 $finish;
    end
endmodule
```

IZLAZ:

```
0 a = z b = 0 c = 0
1 a = 0 b = 1 c = 0
2 a = 0 b = 0 c = 0
3 a = 0 b = 0 c = 1
4 a = 1 b = 1 c = 1
5 a = 1 b = 0 c = 1
```



## ■ Verilog: Registers

- Registarske promjenljive predstavljaju elemente za čuvanje podataka
- Zadržavaju posljednju vrijednost koja im je dodijeljena, dok im se ne dodijeli nova vrijednost
- Praviti razliku između registarskog tipa promjenljive i hardverskog registra u realnim kolima (napravljenog pomoću flip flopa)
- U Verilogu *registar* označava promjenljivu koja može da čuva vrijednost
- Za razliku od *net*-a kod registarske promjenljive nije potreban sklop koji će da “drži” vrijednost promjenljive
- Verilog registarskim promjenljivima ne treba taktni signal kao hardverskim registrima
- Vrijednost registarske promjenljive se može promijeniti u bilo kojem trenutku tokom simulacije, dodjeljivanjem nove vrijednosti

## ■ Verilog: Registers – nastavak

- Registarski tipovi promjenljivih:

Data Types	Functionality
reg	Promjenljiva bez znaka
integer	Promjenljiva sa znakom - 32 bita
time	Promjenljiva bez znaka - 64 bita
real	Promjenljiva sa pokretnim zarezom dvostruke preciznosti

- Od svih tipova najčešće se koristi **reg**
- Podrazumijevana vrijednost za *reg* tip je **x**
- Primjer:  

```
reg reset; // deklarisanje promjenljive koja može čuvati svoju vrijednost  
initial  
begin  
    reset = 1'b1; // inicijalizuje promjenljivu na jedinicu  
    #100 reset = 1'b0; // poslije 100 vremenskih jedinica postavlja na nulu  
end
```



## ■ Verilog: integer

- Registarski tip promjenljive koji se koristi za rad sa brojnim vrijednostima
- Iako je moguće da se koristi i **reg** kao promjenljiva opšte namjene, pogodnije je deklarirati cjelobrojnu (integer) promjenljivu za neke namjene (npr. brojanje)
- Promjenljiva zauzima najmanje 32 bita, a tačna širina zavisi od računara na kome se simulacija izvršava
- Vrijednosti se čuvaju sa znakom (za razliku od *reg*)
- Primjer:

`integer brojac; // promjenljiva opšte namjene uzeta kao brojač`

`initial`

`brojac = -1; // negativna vrijednost je smještena u brojač`



## ■ Verilog: real

- Registarski tip promjenljive koji se koristi za rad sa decimalnim brojnim vrijednostima
- Mogu se specificirati u *decimalnoj* (npr. 1.25) ili *naučnoj* notaciji (npr. 1.2e4, što iznosi  $1.2 \times 10^4$ )
- Podrazumijevana vrijednost je 0

- Primjer:

```
real decimalni;  
initial  
begin  
    decimalni = 3e10; // dodijeljena vrijednost u naučnoj notaciji  
    decimalni = 2.13; // dodijeljena vrijednost u decimalnoj notaciji  
end  
integer i;  
initial  
    i = decimalni; // promjenljiva i ima vrijednost 2 (zaokruženo 2.13)
```



## ■ Verilog: time

- Simulacije u Verilogu se obavljaju saglasno *vremenu simulacije*
- Da bi se ovo vrijeme čuvalo, koristi se poseban tip promjenljive
- Promjenljiva zauzima najmanje 64 bita, a tačna širina zavisi od računara na kome se simulacija izvršava
- Sistemska funkcija **\$time** se koristi da se dobije trenutno vrijeme
- Primjer:

```
time save_sim_time; // definiše se vremenska promj. save_sim_time  
initial
```

```
    save_sim_time = $time; // smješta se trenutno vrijeme simulacije
```



## ■ Verilog: vektori

- I **net** (wire) i **register** (reg) tip promjenljive se može deklarirati kao niz od više bitova (vektor određene dužine)
- Ako se dužina ne specificira podrazumijeva se da je 1 (skalar)
- Primjer:

```
wire a; // skalarna net promjenljiva
wire [7:0] bus; // 8-bitna magistrala (promjenljiva po imenu bus)
wire [31:0] busA,busB,busC; // 3 magistrale od po 32 bita
reg clock; // skalarna registarska promjenljiva
reg [0:40] virtual_addr; // register vektor, dužine 41 bit
```
- Vektori se mogu deklarirati kao [high# : low#] ili [low# : high#], ali je uvijek lijevi broj u uglastim zagradama najznačajniji bit u vektoru (MSB)
- U gornjem primjeru bit 0 je MSB vektora *virtual\_addr*



## ■ Verilog: vektori – nastavak

```
wire [7:0] bus; // 8-bitna magistrala (promjenljiva po imenu bus)
wire [31:0] busA,busB,busC; // 3 magistrale od po 32 bita
reg [0:40] virtual_addr; // register vektor, dužine 41 bit
```

- Za gore navedene deklaracije moguće je adresirati pojedine bitove, ili djelove vektora:

```
busA[7] // bit broj 7 vektora busA
bus[2:0] // tri najmanje značajna bita vektora bus
// korišćenje bus[0:2] nije dozvoljeno jer značajniji bit uvijek treba
// da bude sa lijeve strane u specifikaciji opsega
virtual_addr[0:1] // dva najznačajnija bita vektora virtual_addr
```



## ■ Verilog: nizovi

- U Verilogu su dozvoljeni nizovi za **reg**, **integer**, **time** i **vektore** registarskih tipova podataka
- Nisu dozvoljeni za promjenljive tipa **real**
- Višedimenzionalni nizovi nisu dozvoljeni
- Elementima niza se pristupa sa: `<ime_niza> [<indeks>]`
- Ne miješati nizove sa vektorima:
  - Vektor je jedan element dužine n bita
  - Niz čini više elemenata dužine 1 ili više bitova



## ■ Verilog: nizovi

```
integer count[0:7] ; // niz od 8 promjenljivih count  
reg bool[31:0]; // niz od 32 jednobitne prom. bool registarskog tipa  
time chk-point[1:100]; // niz od 100 promjenljivih tipa time  
reg [4:0] port_id[0:7]; // niz od 8 prom. port_id; svaka je dužine 5 bita  
integer matrix[4:0][4:0]; // nelegalna deklaracija – višedimenzionalni niz
```

```
count[5] // 5-ti element niza promjenljivih count  
chk_point[100] // 100-ti element niza chk_point  
port_id[3] // treći element niza port_id. To je 5-tobitna vrijednost
```



## ■ Verilog: memorije

- Često je potrebno modelovati registarski fajl, RAM ili ROM memoriju
- Memorije se u Verilogu modeluju kao niz registara
- Svaki element niza predstavlja jednu memorijsku riječ
- Svaka riječ može biti dužine jednog ili više bitova
- Važno je razlikovati **n** jednobitnih registara i **1** n-bitni registar
- Pojedinoj memorijskoj riječi se pristupa koristeći adresu kao indeks niza

`reg mem1bit[0:1023] ; // memorija mem1bit sa 1K 1-bitnih riječi`

`reg [7:0] membyte[0:1023]; // memorija membyte sa 1K 8-bitnih riječi`

`membyte [511]; // pristup riječi (bajtu) čija je adresa 511`



## ■ Verilog: parametri

- Verilog omogućava definisanje **konstanti** koje se koriste u modulima
- Koristi se ključna riječ **parameter**:

```
parameter port_id = 5; // definiše konstantu port_id
```

- Parametri se ne mogu koristiti kao promjenljive
- Prilikom pisanja modula treba izbjegavati korišćenje brojnih vrijednosti – umjesto njih treba koristiti predefinisane konstante
- Promjenom vrijednosti konstante (parametra) mijenja se i ponašanje modula



## ■ Verilog: sistemske funkcije

- Sve sistemske funkcije počinju sa znakom \$
- Prikazivanje informacija:
  - `$display(p1, p2, p3, ..... , pn);`
  - *p1, p2, p3, ..... , pn* mogu biti stringovi (pod navodnicima), promjenljive ili izrazi
  - format je vrlo sličan funkciji *printf* u C-u
  - Prelazi u novi red na kraju ispisa
- Primjer:

*// prikazuje string pod navodnicima*

`$display("Zdravo Verilog svijete") ;`

`>Zdravo Verilog svijete`



## ■ Verilog: sistemske funkcije – nastavak

### ■ Specificiranje formata

Format	Prikaz
%d ili %D	Decimalni oblik
%b ili %B	Binarni oblik
%s ili %S	String
%h ili %H	Heksadekadni oblik
%c ili %C	ASCII karakter
%m ili %M	Hijerarhijsko ime (nema argumenta)
%v ili %V	Snaga
%o ili %O	Oktalni oblik
%t ili %T	Vremenski zapis
%e ili %E	Realni broj u naučnoj notaciji
%f ili %F	Realni broj u decimalnoj notaciji
%g ili %G	Realni broj u notaciji koja daje kraći zapis



## ■ Verilog: sistemske funkcije – nastavak

### ■ Primjer:

```
// prikazuje vrijednost trenutnog vremena u simulaciji  
$display($time);  
> 230
```

```
// prikazuje vrijednost 41-bitne virtuelne adrese 1fe0000001c i  
// sistemskog vremena  
reg [0:40] virtual_addr;  
$display("U trenutku %d virtuelna adresa je %h", $time, virtual_addr);  
> U trenutku 200 virtuelna adresa je 1fe0000001c
```

```
// prikazuje vrijednost promjenljive port_id (iznosi 5) u binarnom obliku  
reg [4:0] port_id;  
$display("ID porta je %b", port_id);  
> ID porta je 00101
```



## ■ Verilog: sistemske funkcije – nastavak

### ■ Monitoring signala

- Prikazuje signal kad mu se vrijednost promijeni
- `$monitor(p1,p2,p3,.... ,pn);`
- *p1,p2,p3,.... ,pn* mogu biti promjenljive, imena signala ili string pod navodnicima
- Koristi se specifikacija formata kao kod **\$display** funkcije
- Za razliku od \$display, \$monitor se uvodi samo jednom
- Permanentno nadgleda vrijednosti promjenljivih ili signala navedenih u listi parametara i prikazuje sve parametre iz liste kada se vrijednost makar jednog promijeni
- U jednom trenutku može biti aktivna samo jedna lista parametara za monitoring



## ■ Verilog: sistemske funkcije – nastavak

### ■ Monitoring signala – nastavak

- Ako ima više od jednog izraza \$monitor u simulaciji, posljednji je aktivan
- Za uključivanje odnosno isključivanje monitoringa koriste se \$monitoron i \$monitoroff, respektivno

### ■ Primjer:

```
// nadgleda vrijeme i vrijednost signala clock i reset  
// clock se mijenja svakih 5 vremenskih jedinica, a reset se postavlja na  
// nulu na 10-tu vremensku jedinicu
```

```
initial
```

```
begin
```

```
    $monitor($time, " Vrijednost clock=%b reset=%b", clock,reset);
```

```
end
```

Djelimični izlaz funkcije monitoringa:

0 Vrijednost clock = 0 reset = 1

5 Vrijednost clock = 1 reset = 1

10 Vrijednost clock = 0 reset = 0



## ■ Verilog: sistemske funkcije – nastavak

### ■ Zaustavljanje simulacije

- `$stop;`

- Simulacija se prebacuje u interaktivni mod

### ■ Završavanje simulacije

- `$finish;`

### ■ Primjer:

- `// zaustavlja simulaciju u trenutku 100 i provjerava rezultate`

- `// završava simulaciju u trenutku 1000`

- `initial // početni trenutak (time = 0)`

- `begin`

- `clock = 0;`

- `reset = 1;`

- `#100 $stop; // ovo će zaustaviti simulaciju u trenutku time=100`

- `#900 $finish; // ovo će završiti simulaciju u trenutku time=1000`

- `end`



## ■ Verilog: direktive kompajleru

### ■ 'define

- Definiše tekstualne makroe ili konstante
- Slično direktivi #define u C-u
- Definisane konstante/makroi se u Verilog kodu koriste sa znakom ' (apostrof) ispred naziva
- Primjer:

```
'define WORD_SIZE 32 // koristi se kao 'WORD_SIZE u kodu
```

```
// Definicija alias-a. Svako pojavljivanje 'S će biti zamijenjeno sa $stop  
'define S $stop;
```

```
// definicija često korišćenog tekstualnog stringa
```

```
'define WORD_REG reg [31:0]
```

```
// 32-bitni registar se sada može definisati sa 'WORD_REG reg32;
```



## ■ Verilog: direktive kompajleru

### ■ 'include

- Omogućava uključivanje sadržaja jednog Verilog fajla u drugi, prilikom kompajliranja
- Slično direktivi #include u C-u
- Tipično se koristi za uključivanje tzv. zaglavlja (*header* fajl) u kome se nalaze globalne ili često korišćene definicije
- Primjer:

```
// Uključuje fajl header.v, koji sadrži deklaracije korišćene u glavnom  
// verilog fajlu dizajn.v
```

```
'include header.v
```

```
...
```

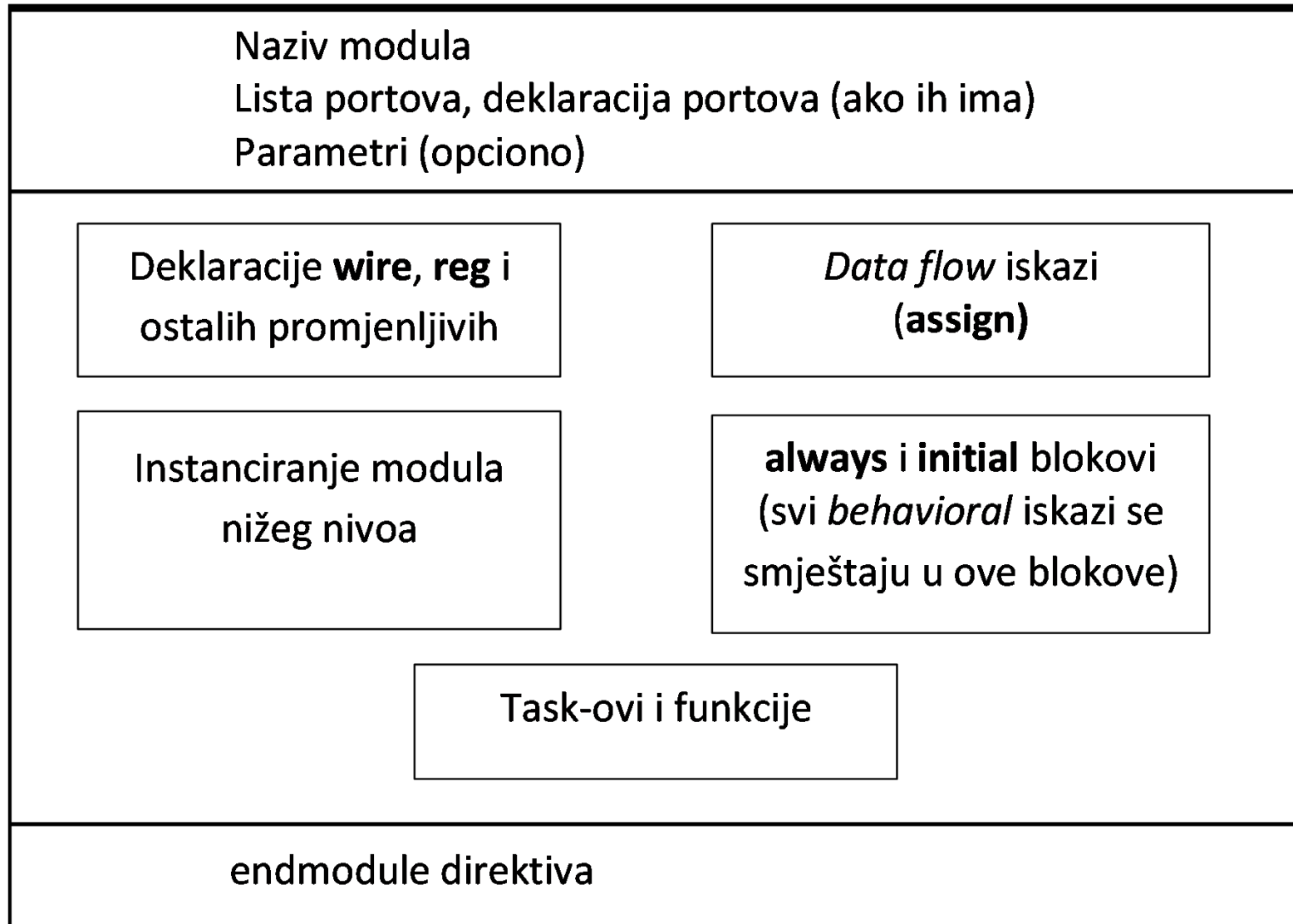
```
<Verilog kod u fajlu dizajn.v>
```

```
...
```



## ■ Verilog: moduli i portovi

### Komponente Verilog modula





## ■ Verilog: moduli i portovi – nastavak

- Naziv modula, lista portova, njihove deklaracije i parametri se moraju specificirati **na početku**
- Lista portova i njihove deklaracije se navode ako modul ima portove preko kojih vrši interakciju sa okruženjem
- Parametri su opcioni
- Ostale komponente se mogu navoditi u proizvoljnom redosljedu unutar definicije modula
- Definicija modula sa uvijek završava sa **endmodule**

## ■ Verilog: moduli i portovi – nastavak

### ■ Primjer – SR latch

// naziv modula i lista portova

```
module SR_latch(Q, Qbar, Sbar, Rbar);
```

// deklaracija portova

```
output Q, Qbar;
```

```
input Sbar, Rbar;
```

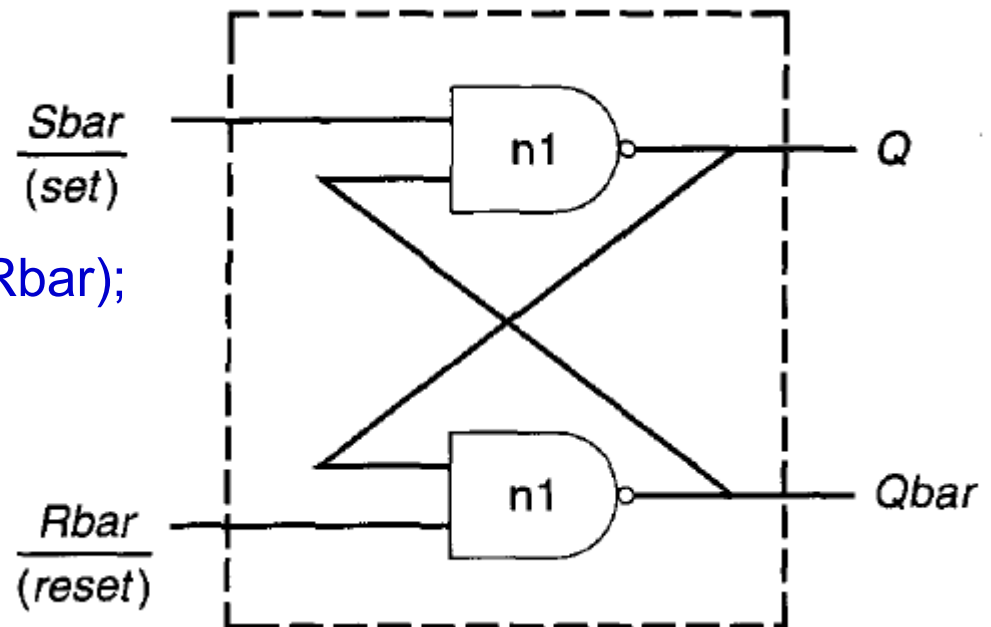
// instanciranje modula nižeg nivoa

// ovdje su to Verilog elementi – nand kola

```
nand n1(Q, Sbar, Qbar);
```

```
nand n2(Qbar, Rbar, Q);
```

```
endmodule
```



## ■ Verilog: moduli i portovi – nastavak

### ■ Primjer – SR latch (stimulus modul)

// Stimulus modul – on će biti modul najvećeg nivoa

```
module Top;
```

// deklaracija promjenljivih

```
wire q, qbar;
```

```
reg set, reset;
```

// instanciranje modula nižeg nivoa – u našem slučaju SR latch

// na SR latch dovodimo invertovane signale *set* i *reset*

```
SR_latch m1(q, qbar, ~set, ~reset);
```

```
initial
```

```
begin
```

```
    $monitor($time, " set = %b, reset= %b, q= %b\n",set,reset,q);
```

```
    set = 0; reset = 0;
```

```
    #5 reset = 1;
```

```
    #5 reset = 0;
```

```
    #5 set = 1;
```

```
end
```

```
endmodule
```

