PROJEKTOVANJE INTEGRISANIH KOLA SA MEŠOVITIM SIGNALIMA

4

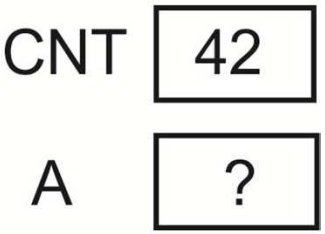
Non-Blocking assignment

Proces simulacije

- Pitanje: Koja će biti vrednost registra ‘A’ posle sledeće rastuće ivice CLK?

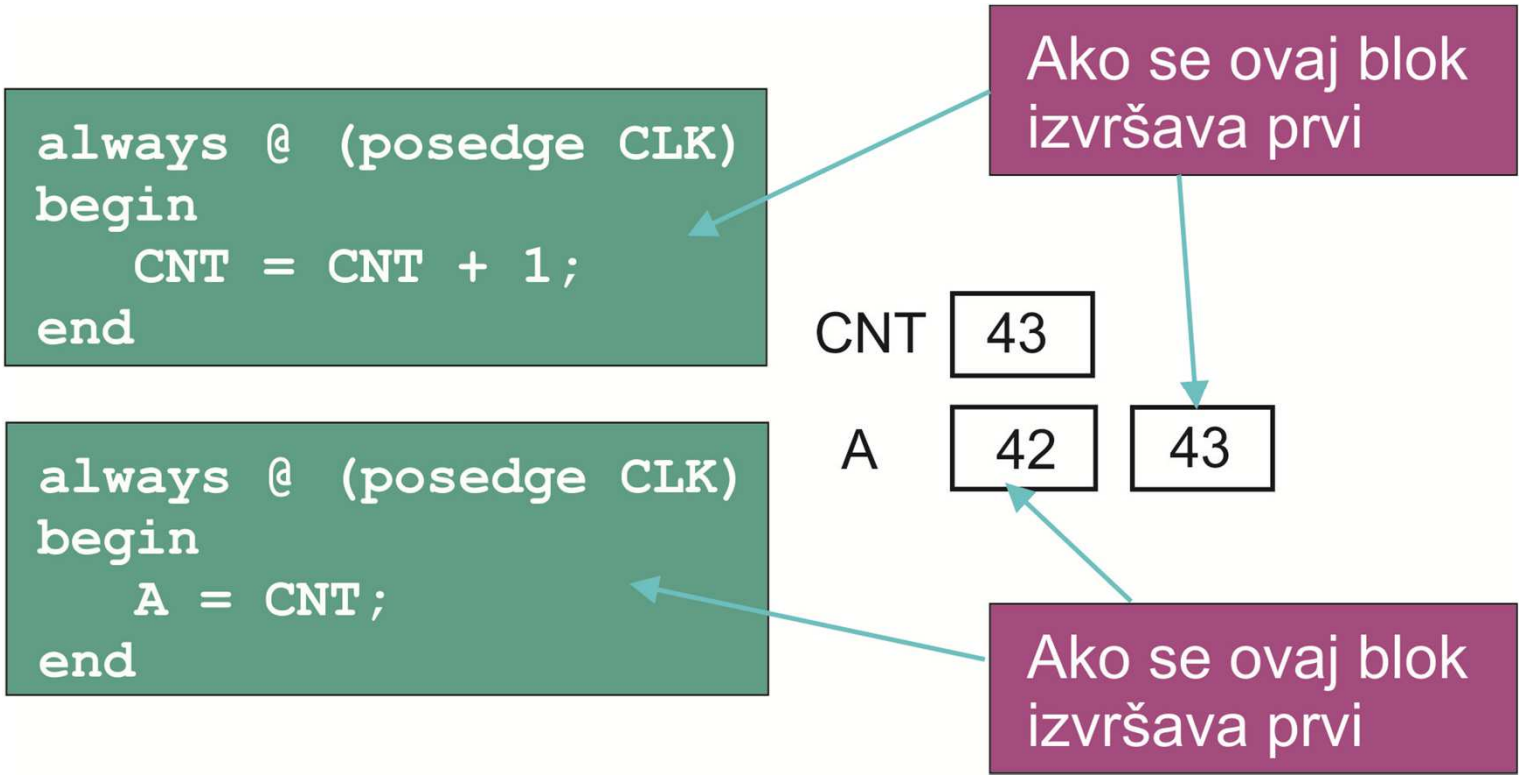
```
always @ (posedge CLK)
begin
    CNT = CNT + 1;
end
```

```
always @ (posedge CLK)
begin
    A = CNT;
end
```



Proces simulacije

- Rezultat zavisi od toga koji blok prvi stigne do ‘CNT’!!!

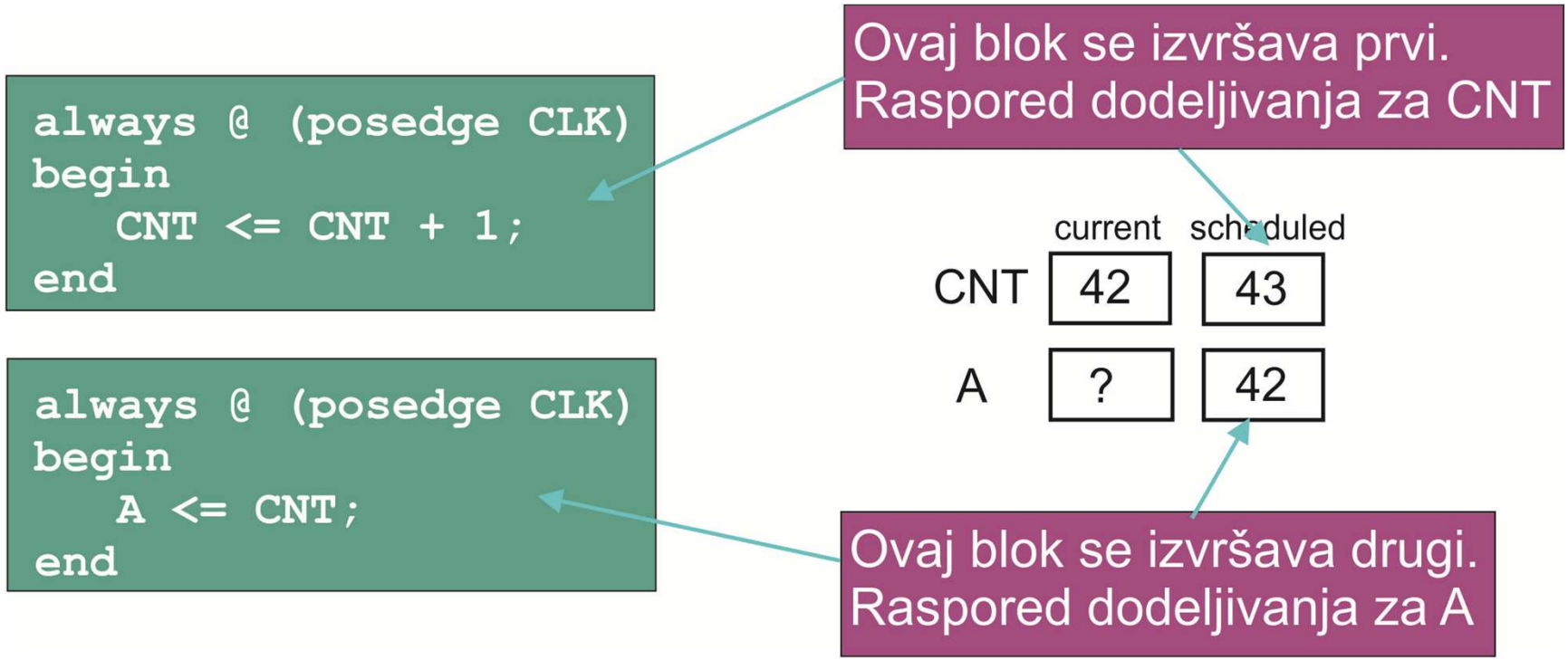


Non-Blocking assignment

- ne zavisimo više od redosleda
- Non-Blocking assignment raspoređuje kome se nova vrednost dodeljuje tek na kraju tekućeg vremenskog koraka
- Nova vrednost se stavlja u red i čeka pridruživanje
- Dodeljivanje se dešava tek kada su svi blokovi izvršeni u toku jednog vremenskog koraka
- Sintaksa:

```
<target> <= <expr>;
```

Non-Blocking assignment



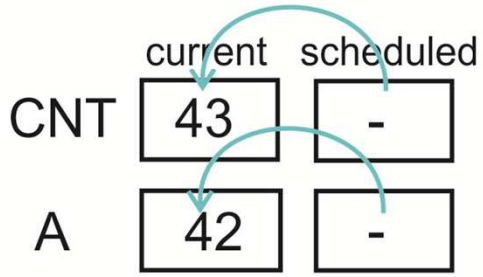
dodeljivanje je na kraju vremenskog koraka

Non-Blocking assignment

Svi blokovi su izvršeni.
Dodeljivanje je prema rasporedu.

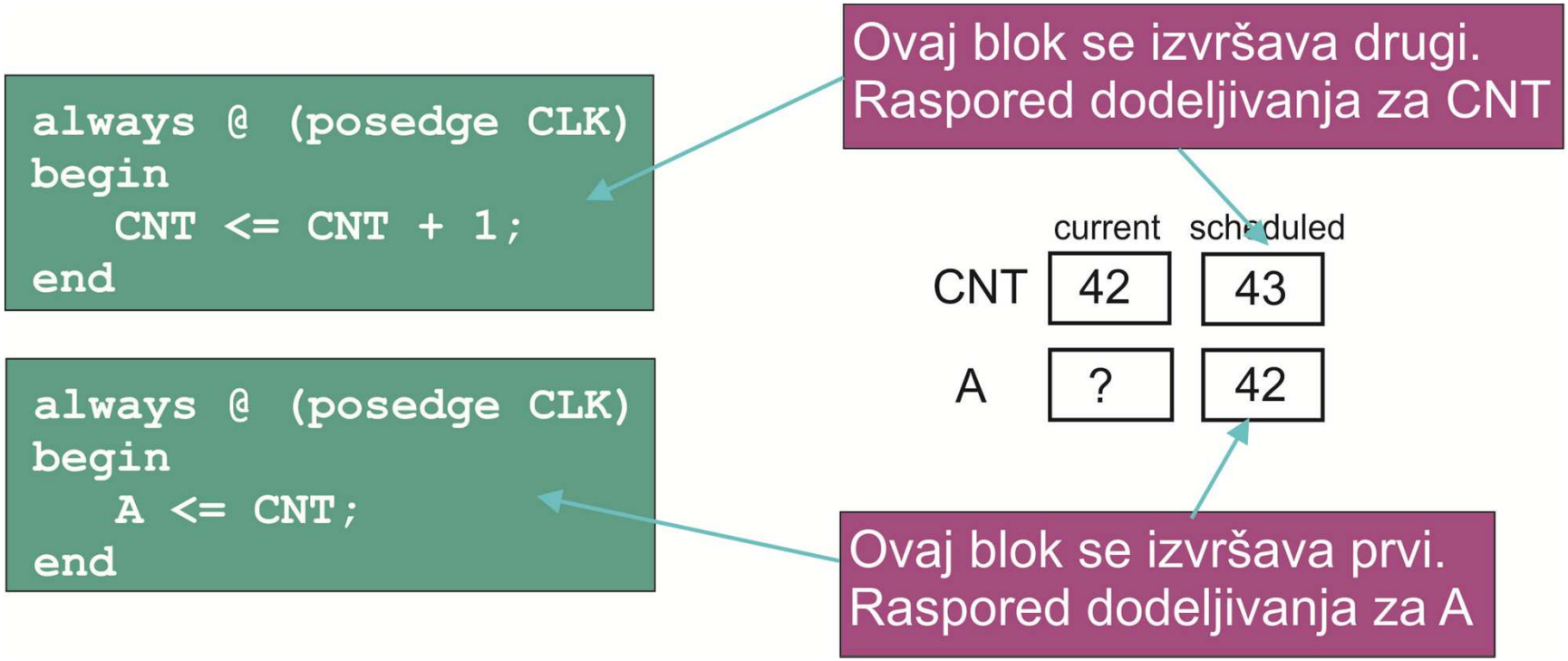
```
always @ (posedge CLK)
begin
    CNT <= CNT + 1;
end
```

```
always @ (posedge CLK)
begin
    A <= CNT;
end
```



vrednosti smo dodelili na kraju vremenskog koraka, pre nego što vreme napreduje

Non-Blocking assignment

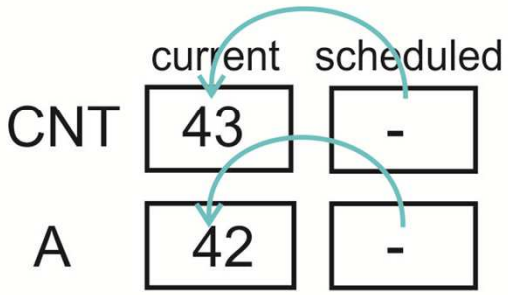


Non-Blocking assignment

Svi blokovi su izvršeni.
Dodeljivanje je prema rasporedu.

```
always @ (posedge CLK)
begin
    CNT <= CNT + 1;
end
```

```
always @ (posedge CLK)
begin
    A <= CNT;
end
```



Rezultat ne zavisi od redosleda izvršavanja.

Non-Blocking assignment

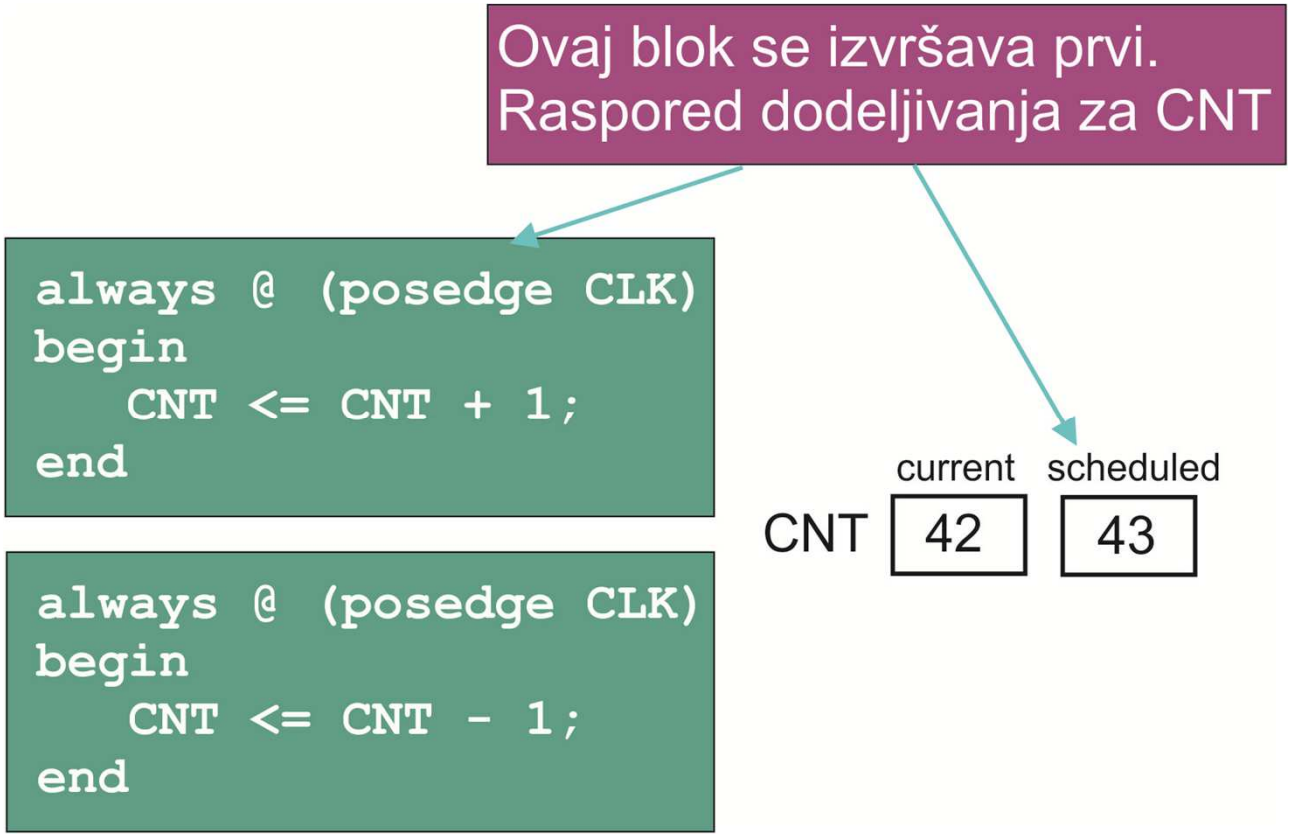
Pitanje:Koja će biti vrednost registra ‘CNT’ posle sledeće rastuće ivice CLK?

```
always @ (posedge CLK)
begin
    CNT <= CNT + 1;
end
```

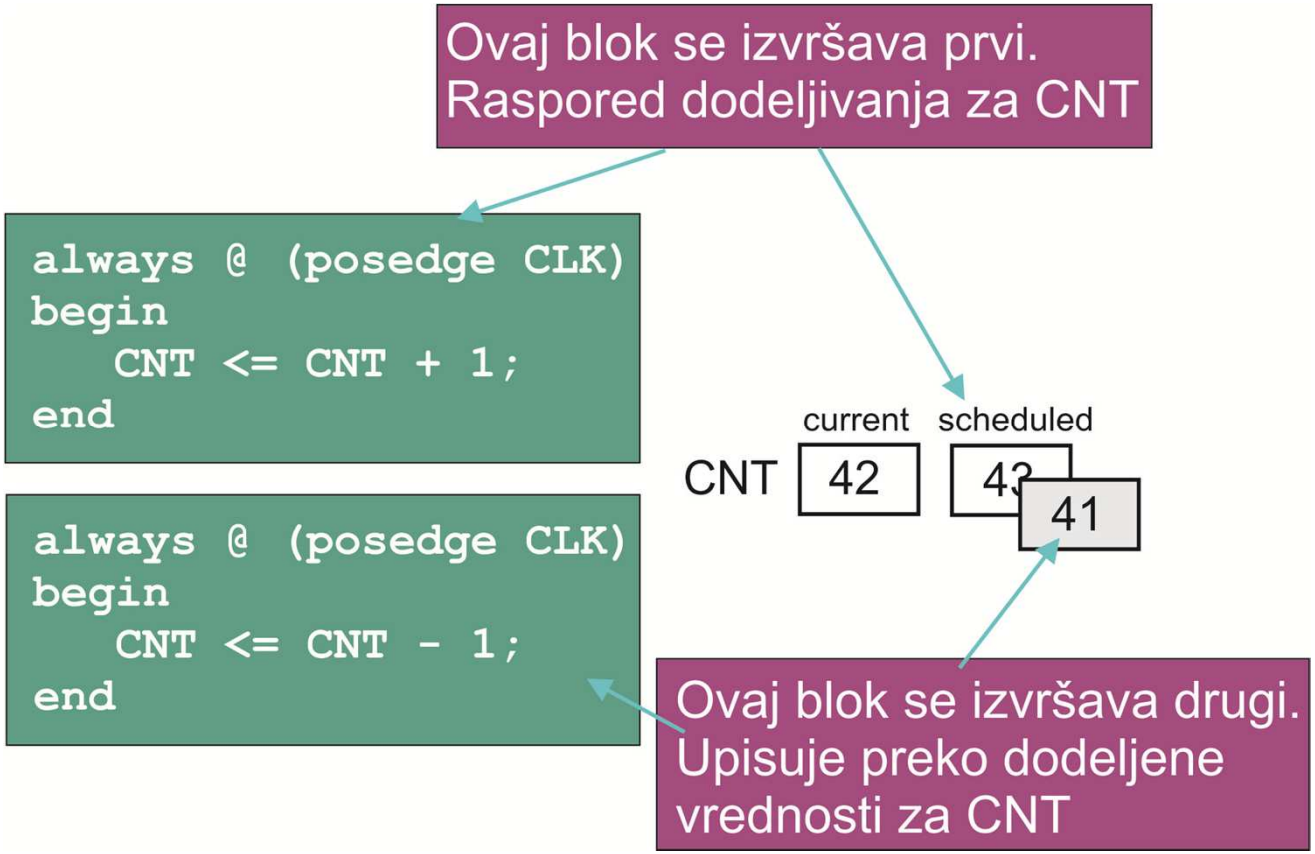
```
always @ (posedge CLK)
begin
    CNT <= CNT - 1;
end
```

CNT 42

Non-Blocking assignment



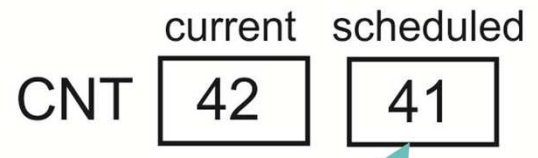
Non-Blocking assignment



Non-Blocking assignment

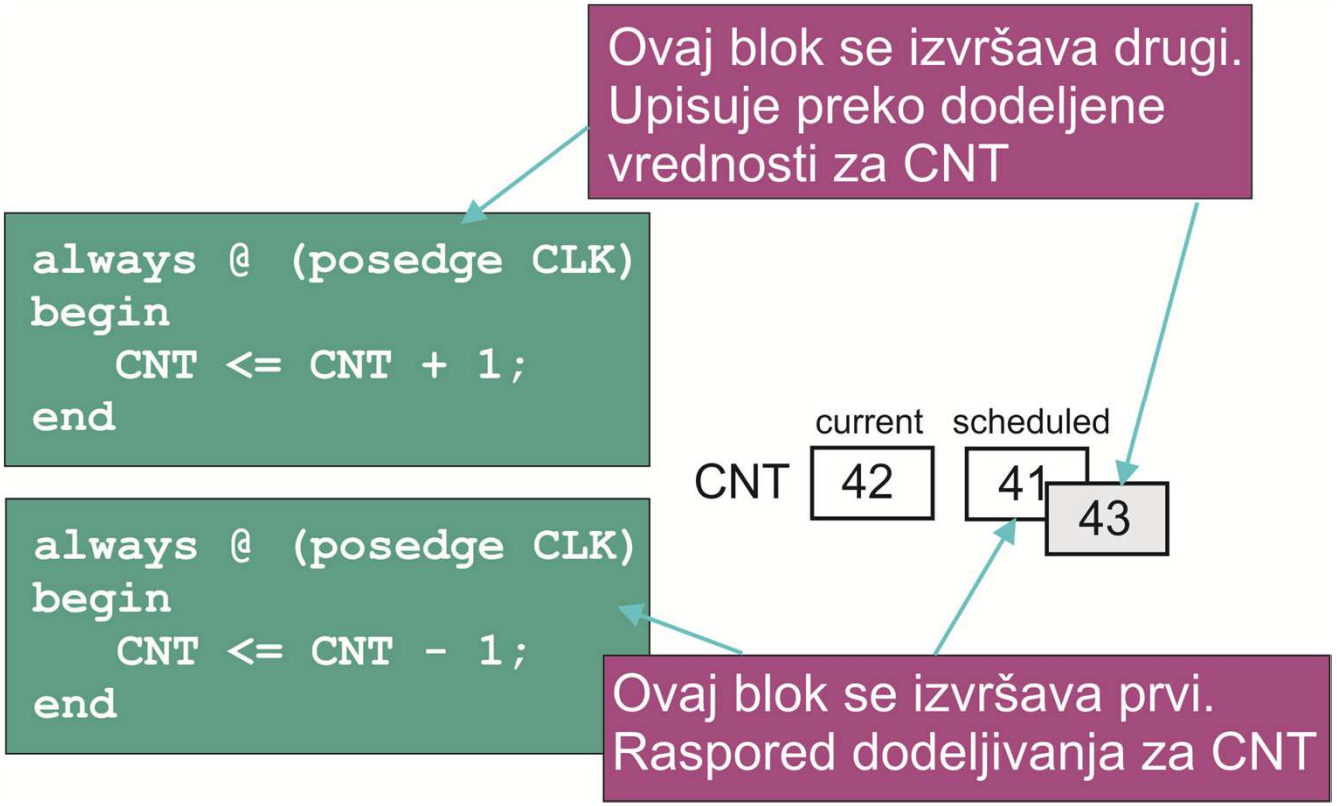
```
always @ (posedge CLK)
begin
    CNT <= CNT + 1;
end
```

```
always @ (posedge CLK)
begin
    CNT <= CNT - 1;
end
```



Ovaj blok se izvršava prvi.
Raspored dodeljivanja za CNT

Non-Blocking assignment



rezultat je neodređen, pa ne treba koristiti non-blocking za ovakve primere

Non-Blocking assignment
napomene

- Koristiti non-blocking assignment za dodeljivanje bilo kom registru koji koriste drugi proceduralni blokovi
- Dodeljivati registrima iz jednog proceduralnog bloka, ne u 2 bloka kao u prethodnom primeru
- Non-blocking assignment garantuje simulaciju bez utrkivanja- ali se moraju poštovati prethodne napomene

Non-Blocking assignment

- Non-blocking assignments mogu biti raspoređeni u proizvoljnim trenucima u budućnosti
- Nova vrednost se dodaje u red čekanja tako da ne utiče na dodeljivanja koja su prethodno obavljena
- Oni zamenjuju prethodno raspoređene zadatke sa istim vremenskim ofsetom

- Sintaksa:

```
<target> <= #(<time-expr>)<expr>;
```

Non-Blocking assignment

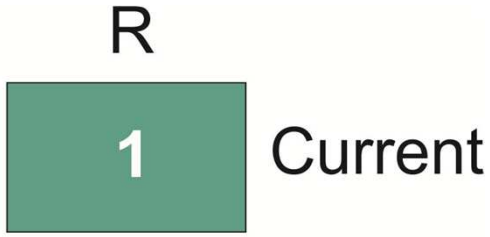
- Primer:

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

Non-Blocking assignment

- Primer:

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

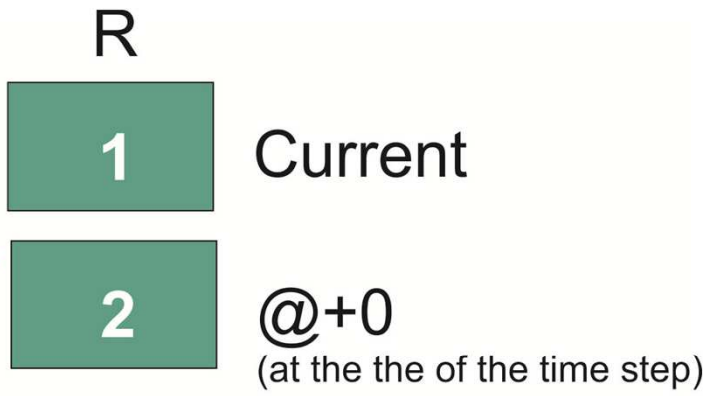


prva dodela je blocking assignment, i dešava se trenutno

Non-Blocking assignment

- Primer:

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

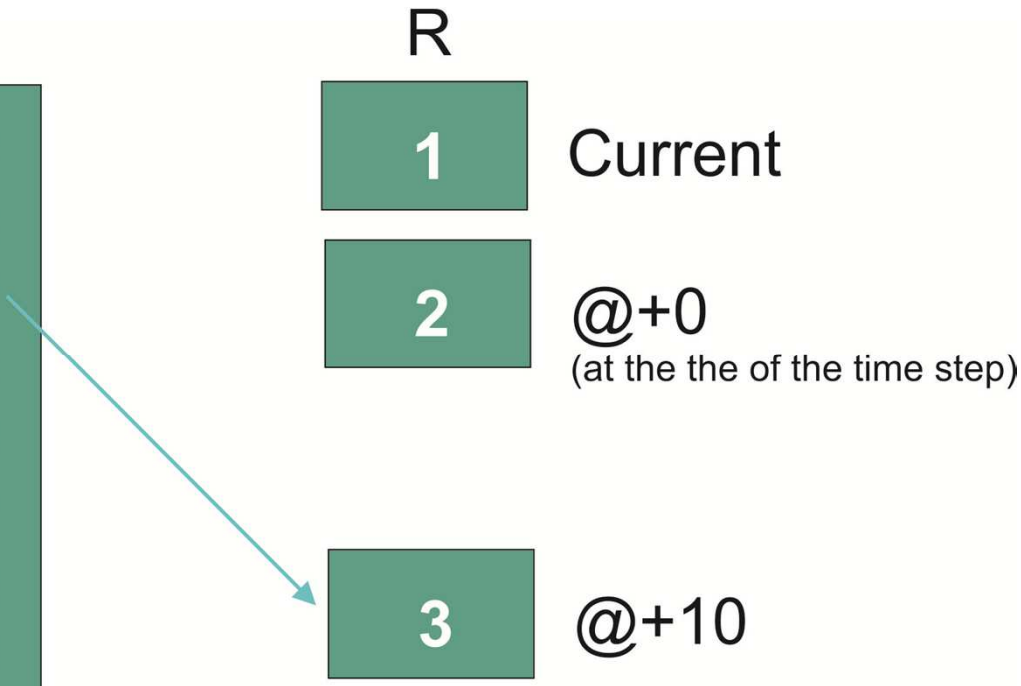


dodeljivanje na kraju prvog vremenskog koraka

Non-Blocking assignment

- Primer:

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

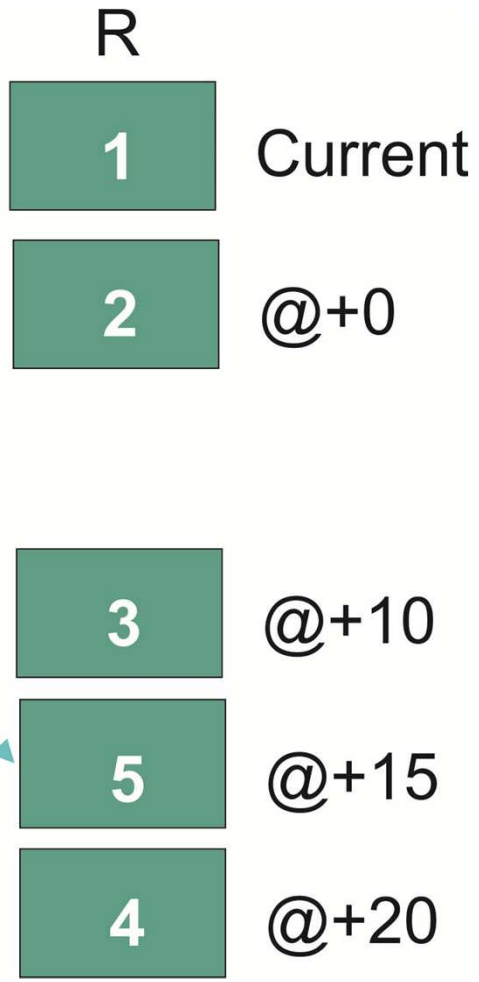


Non-Blocking assignment

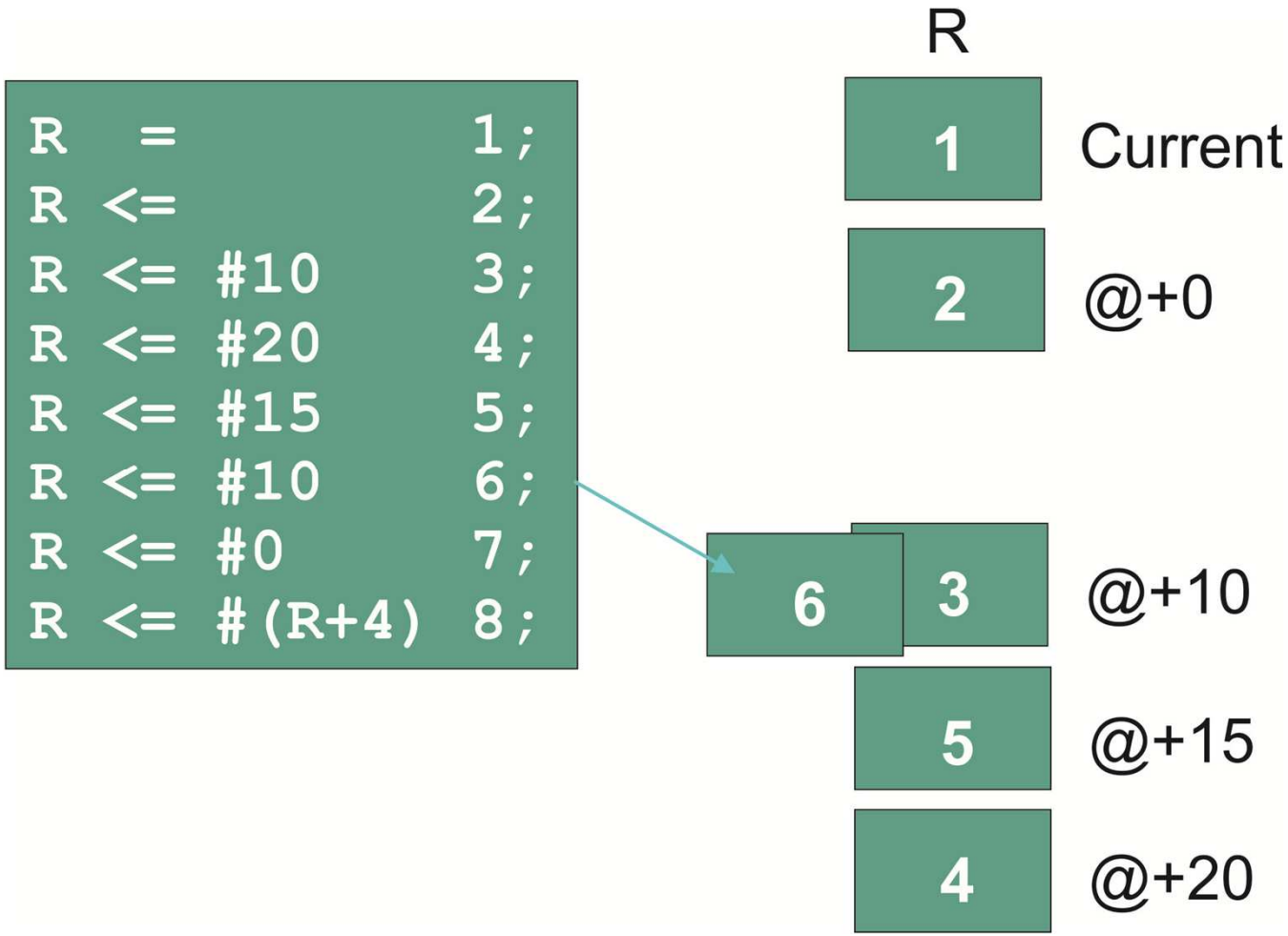
```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

Non-Blocking assignment

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```

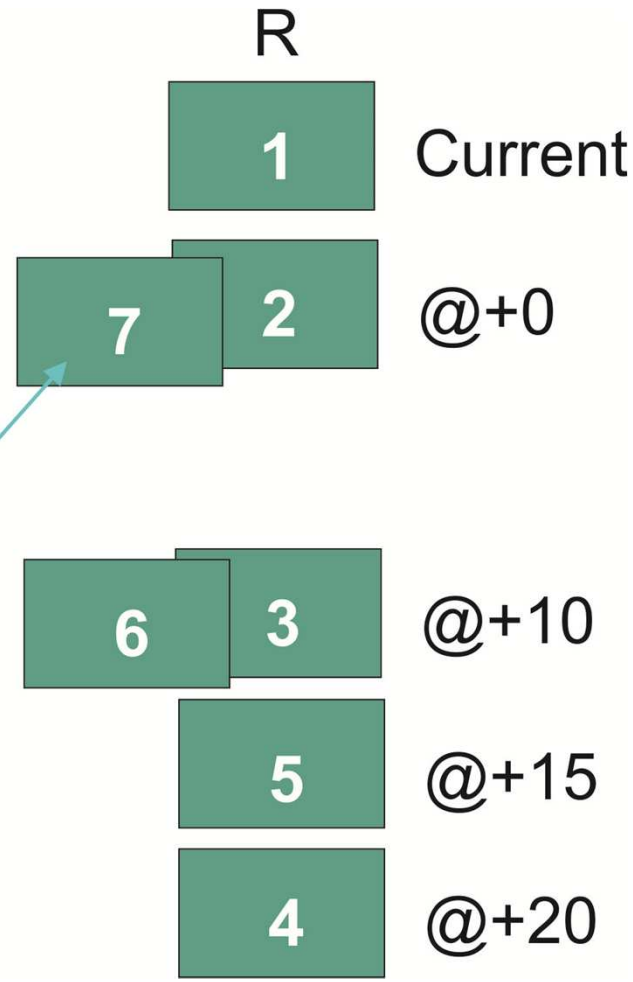


Non-Blocking assignment

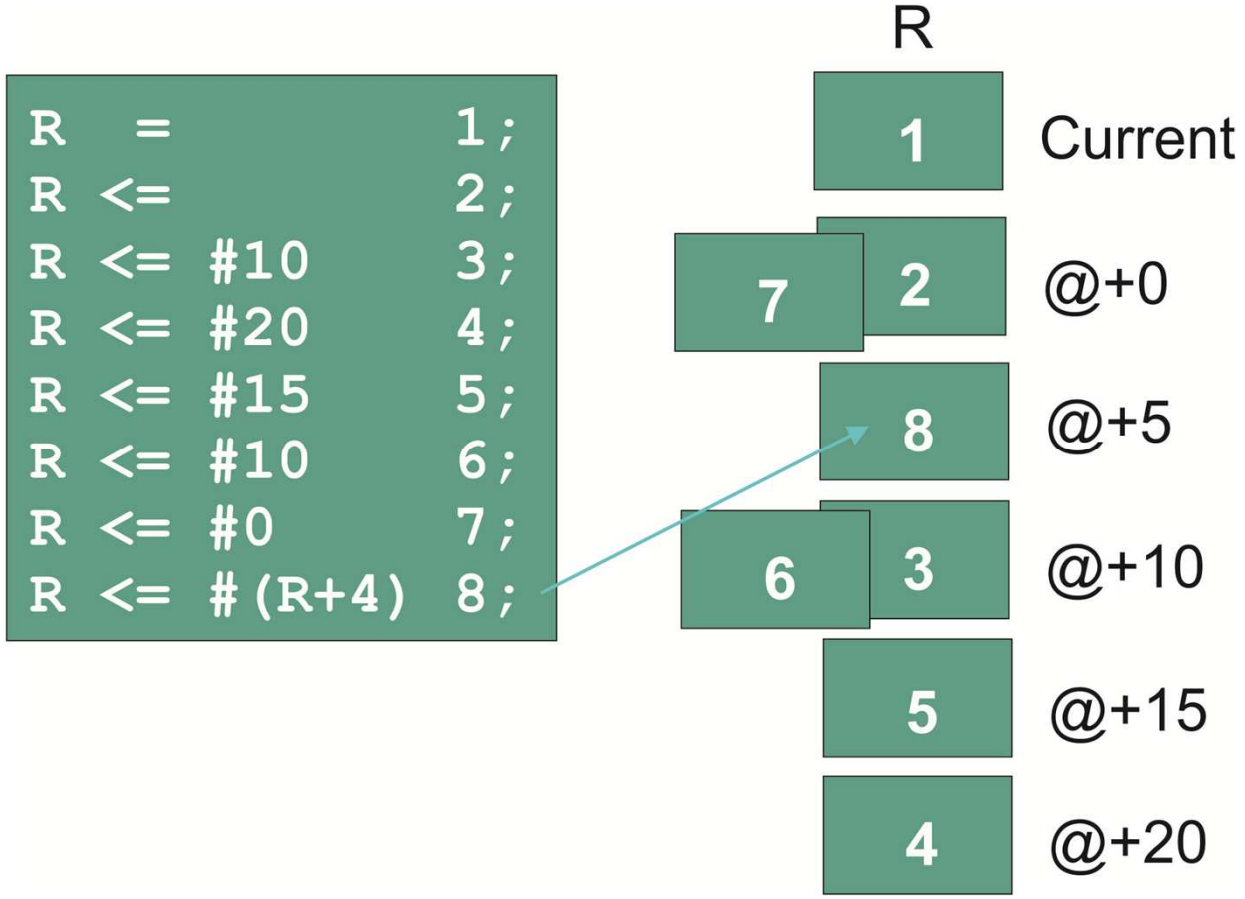


Non-Blocking assignment

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```



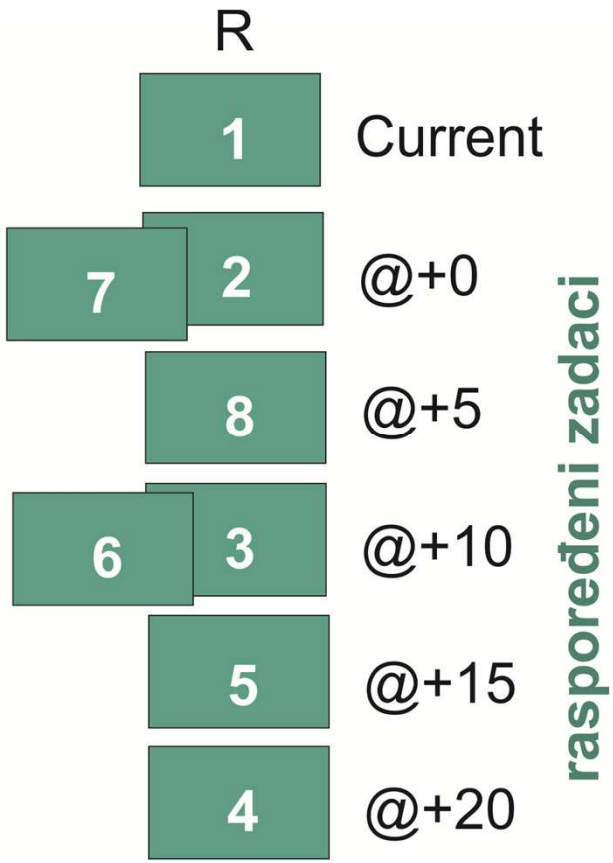
Non-Blocking assignment



Non-Blocking assignment

Vreme nije napredovalo između ove dve tačke

```
R  =          1;
R <=          2;
R <= #10      3;
R <= #20      4;
R <= #15      5;
R <= #10      6;
R <= #0       7;
R <= #(R+4)   8;
```



Timing kontrole -> Događaji

- Zasnovane na vremenu
  - regularno kašnjenje
  - kašnjenje u okviru zadatka
  - kašnjenje nula

```
#10;
Y = #5 X + Z;
#0;
```

- Zasnovane na događaju
  - regularni događaj
  - imenovani događaj
  - događaj sa OR uslovom
  - promena vrednosti reg ili wire

```
@(posedge CLK);
event rx_data;
@(posedge CLK or negedge RSTn);
```

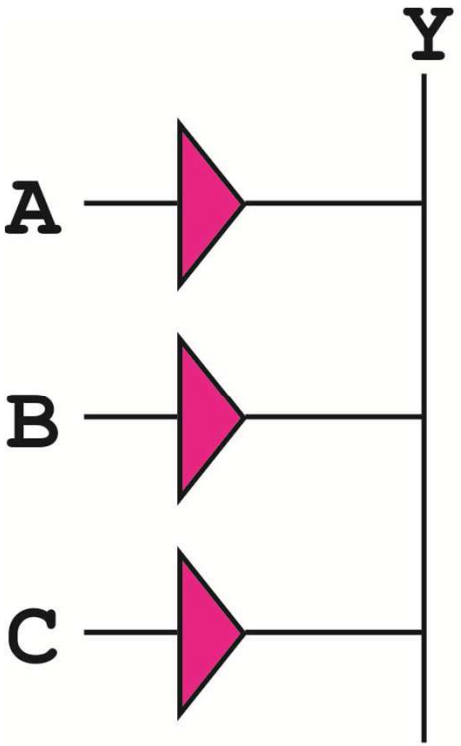
- Zavisi od nivoa

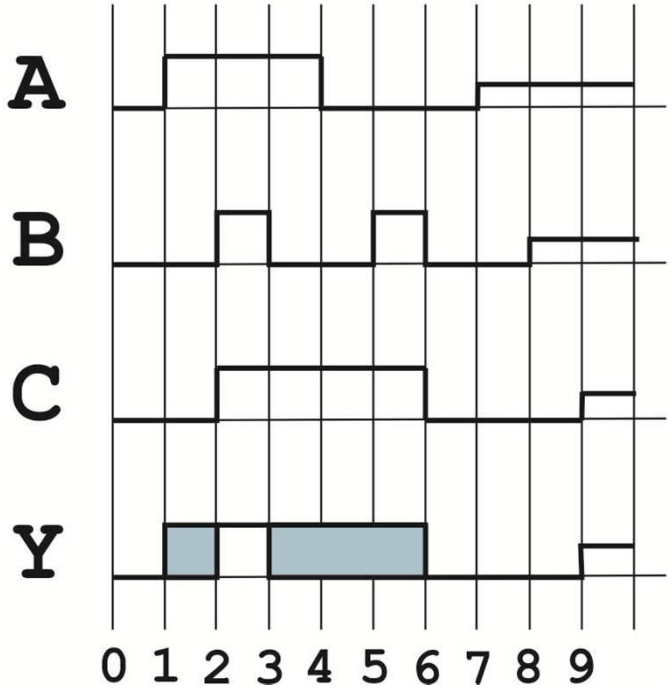
```
wait (cnt == 2);
```

Wires

Višestruki drajveri

- Zadatak: modelovati bus i drajvere u Verilogu





Višestruki drajveri

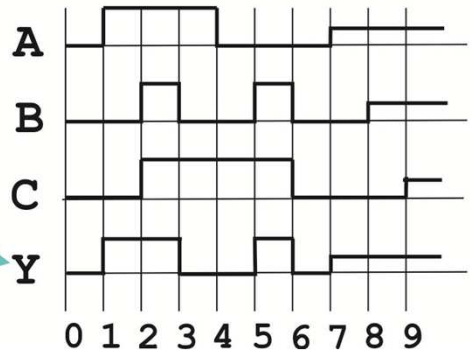
- Korišćenje jednog proceduralnog bloka po drajveru

```
initial
begin // A
   y <=    1'b0;
   y <= #1 1'b1;
   y <= #4 1'b0;
   y <= #7 1'bz;
end
```

```
reg Y;
initial
begin // C
   y <=    1'b0;
   y <= #2 1'b1;
   y <= #6 1'b0;
   y <= #9 1'bz;
end
```

```
initial
begin // B
   y <=    1'b0;
   y <= #2 1'b1;
   y <= #3 1'b0;
   y <= #5 1'b1;
   y <= #6 1'b0;
   y <= #8 1'bz;
end
```

'Y' prati poslednje dodeljivanje



'Y' nikada ne ide u 1'bx

Višestruki drajveri

- Registri se ne mogu koristiti za modelovanje bus-ova
  - uzimaju vrednost zadnjeg dodeljivanja
  - poslednji drajver odlučuje konačnu vrednost
  - ne postoje ‘x’ vrednosti, uzima se ili ‘0’ ili ‘1’, u zavisnosti od poslednjeg dodeljivanja

- Potreban je novi objekat da bi se modelovali višestruki drajveri: **WIRE**
  - konačna vrednost je kontinualna funkcija svih drajvera

- Sintaksa:

```
wire <list-of-names>;
wire [msb:lsb]<list-of-names>;
```

Wires

Primeri:

`wire W;`	1-bitna wire 'W'
`wire A,B;`	Dve 1-bitne wires, 'A' i 'B'
`wire [7:0] BYTE;`	8-bitna wire 'BYTE'
`wire [1:16] WORD;`	16-bitna wire 'WORD'
`wire [3:0] A;` `[8:1] B;`	Sintaksna greška
`wire [3:0] A;` `wire [8:1] B;`	dve wire- 4-bitna 'A' i 8-bitna 'B'

Wires

- Samo jedan ili više bitova
  - neoznačene integer vrednosti
  - MSB sa leve strane
  - LSB sa desne strane

- Nema integer
- Nema time
- Nema real
- Nema memorija

Continuous assignment

- Wires se pobuđuju korišćenjem *continuous assignment*
- *Continuous assignment* kontinualno pobuđuje wire vrednošću izraza
- Konačna vrednost na wire je funkcija svih continuous assignment-a koji su na njoj
- Sintaksa:

```
assign <wire-target> = <expr>;
```

Continuous assignment

- Continuous assignment se izvršava paralelno sa proceduralnim blokovima
- Nalazi se u modulu

```
module MODEL;
   [{<procedural-block>}]

   assign <wire-target> = <expr>;

   [{<procedural-block>}]
endmodule
```

Continuous assignment

- Primeri:

```
assign W = 1'b1;
```

```
assign W = (CS == 1'b1) ? D : 1'bz;
```

```
assign DATA = {MSW, LSW};
```

```
assign {DB1, DB0} = WORD;
```

Continuous assignment

Vraćamo se na naš model bus-a:

drajveri

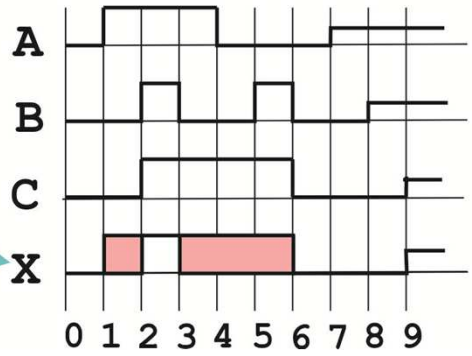
```
wire X;
assign X = A;
assign X = B;
assign X = C;
```

```
initial
begin
   A <=    1'b0;
   A <= #1 1'b1;
   A <= #4 1'b0;
   A <= #7 1'bz;
end
```

```
initial
begin
   C <=    1'b0;
   C <= #2 1'b1;
   C <= #6 1'b0;
   C <= #9 1'bz;
end
```

```
initial
begin
   B <=    1'b0;
   B <= #2 1'b1;
   B <= #3 1'b0;
   B <= #5 1'b1;
   B <= #6 1'b0;
   B <= #8 1'bz;
end
```

ispravno ponašanje na 'X'



Continuous assignment

- Continuous assignment može da sadrži i kašnjenje
- Sintaksa:

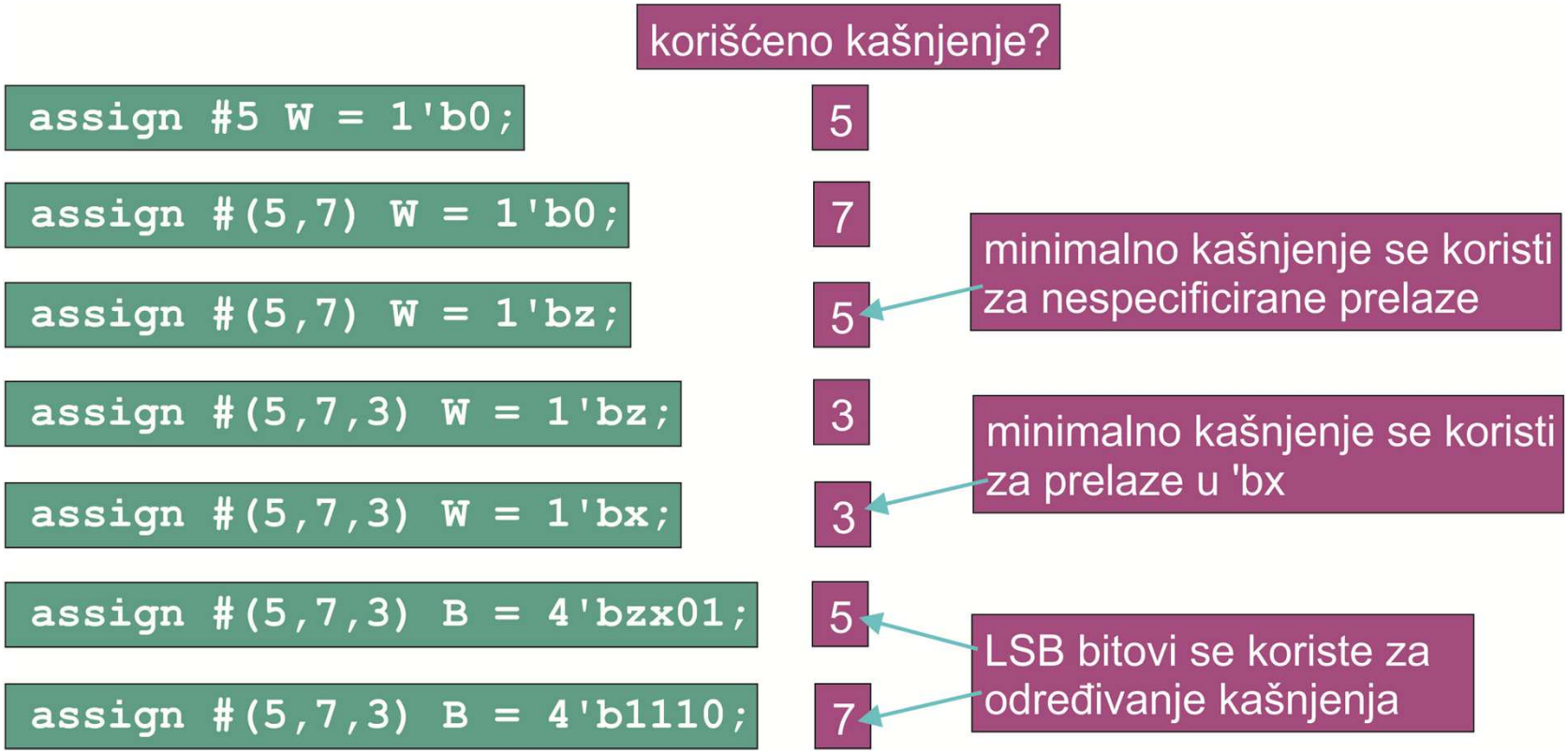
```
assign [#(<expr>[, <expr>[, <expr>]])]
       <wire-target> = <expr>;
```

- Moguće je specificirati različita kašnjenja za rastući, opadajući i turn-off prelaz (u ‘bz)

`#(<delay>)`	Svi prelazi
`#(<rise>, <fall>)`	Rise/fall kašnjenja
`#(<rise>, <fall>, <off>)`	Rise/fall/turn-off kašnjenja

Continuous assignment

- Primeri:



Continuous assignment

- Koristi *inercioni* model kašnjenja
  - ako se ulaz promeni pre nego što je prethodna ulazna vrednost dodeljena wire-u, onda će zadatak koji čeka biti otkazan
  - odbacuje uske impulse

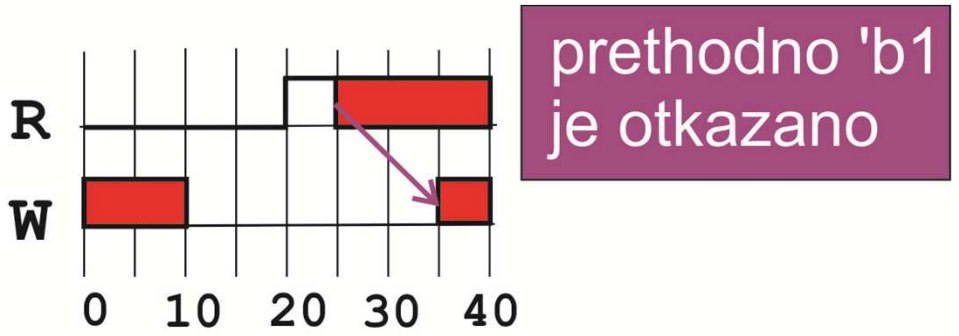
- Slično baferu sa ograničenom širinom impulsa

Inerciono kašnjenje

- Ako se ulaz promeni, zadatak koji čeka biće otkazan

```
wire W;
reg R;
assign #10 W = R;

initial
begin
       R = 1'b0;
  #20  R = 1'b1;
   #5  R = 1'bx;
end
```

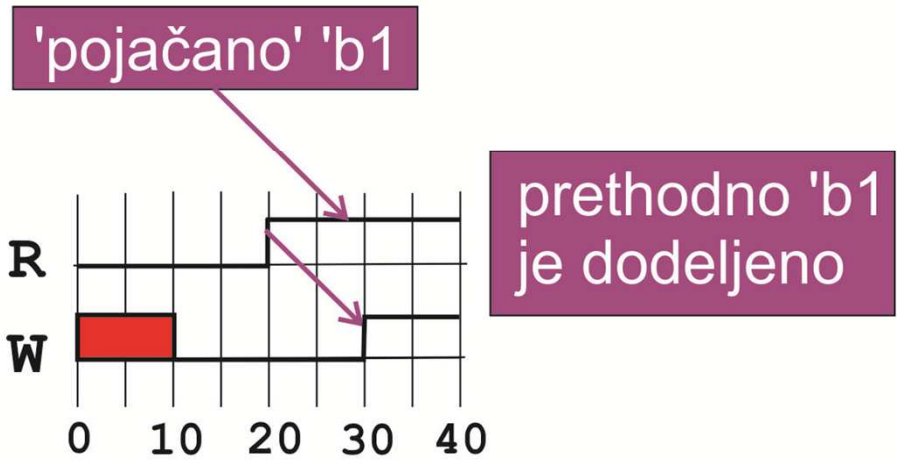


Inerciono kašnjenje

- Ako je ulaz ‘pojačan’, tj. ako se dodeljuje ista vrednost, prethodni zadatak prolazi

```
wire W;
reg R;
assign #10 W = R;

initial
begin
       R = 1'b0;
  #20  R = 1'b1;
   #5  R = 1'b1;
end
```



Jačine

- Continuous assignment može da pobuđuje sa različitim jačinama
  - može da ima različite jačine za ‘b1 i ‘b0
- Jači drajver ‘pobeđuje’ slabiji drajver
- Konflikt između ‘b1 i ‘b0 sa istim jačinama proizvodi ‘bx
- ‘bz ima najmanju jačinu, bez obzira na jačinu continuous assignment-a

Jačine

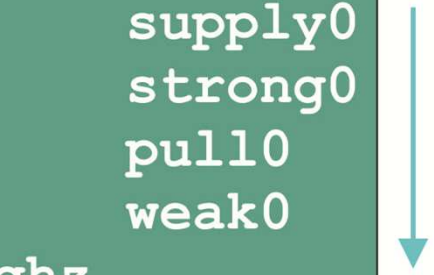
- Sintaksa:

```
assign [(<strengh>, <strengh>)]
       [#(<expr>[, <expr>[, <expr>]])]
       <wire-target> = <expr>;
```

- Nivoi jačine:

default jačina je "strong"

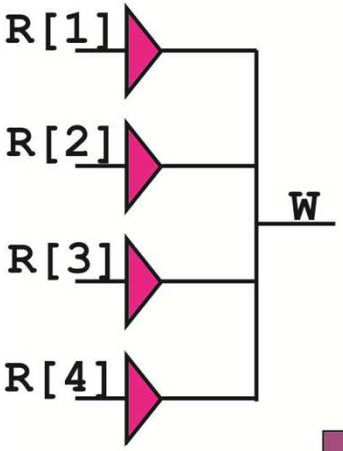
supply1	supply0
strong1	strong0
pull1	pull0
weak1	weak0
highz	



jačina opada

Jačine

- Primeri:




```
wire W;
reg [1:4] R;

assign (supply1, strong0) W = R[1];
assign (strong1, pull0)   W = R[2];
assign (pull1, weak0)     W = R[3];
assign (weak1, supply0)   W = R[4];
```

vrednost na 'W'

`R = 4'b1000;`	'bx
`R = 4'b100z;`	'b1
`R = 4'bzzzz;`	'bz
`R = 4'b0011;`	'b0

default jačina je "strong"

supply1	supply0
strong1	strong0
pull1	pull0
weak1	weak0
highz	

jačina opada

Tipovi wire

- Postoji mnogo tipova wire
- Tip utiče na rezultat kod višestrukih drajvera

Tip	Ponašanje
wire, tri	3-state net
tri0	3-state net with pulldown
tri1	3-state net with pullup
wand, triand	wired-and net
wor, trior	wired-or net
trireg	capacitive net
supply0	net with a supply0 driver
supply1	net with a supply1 driver