

**PROJEKTOVANJE  
INTEGRISANIH KOLA SA  
MEŠOVITIM SIGNALIMA**

**3**

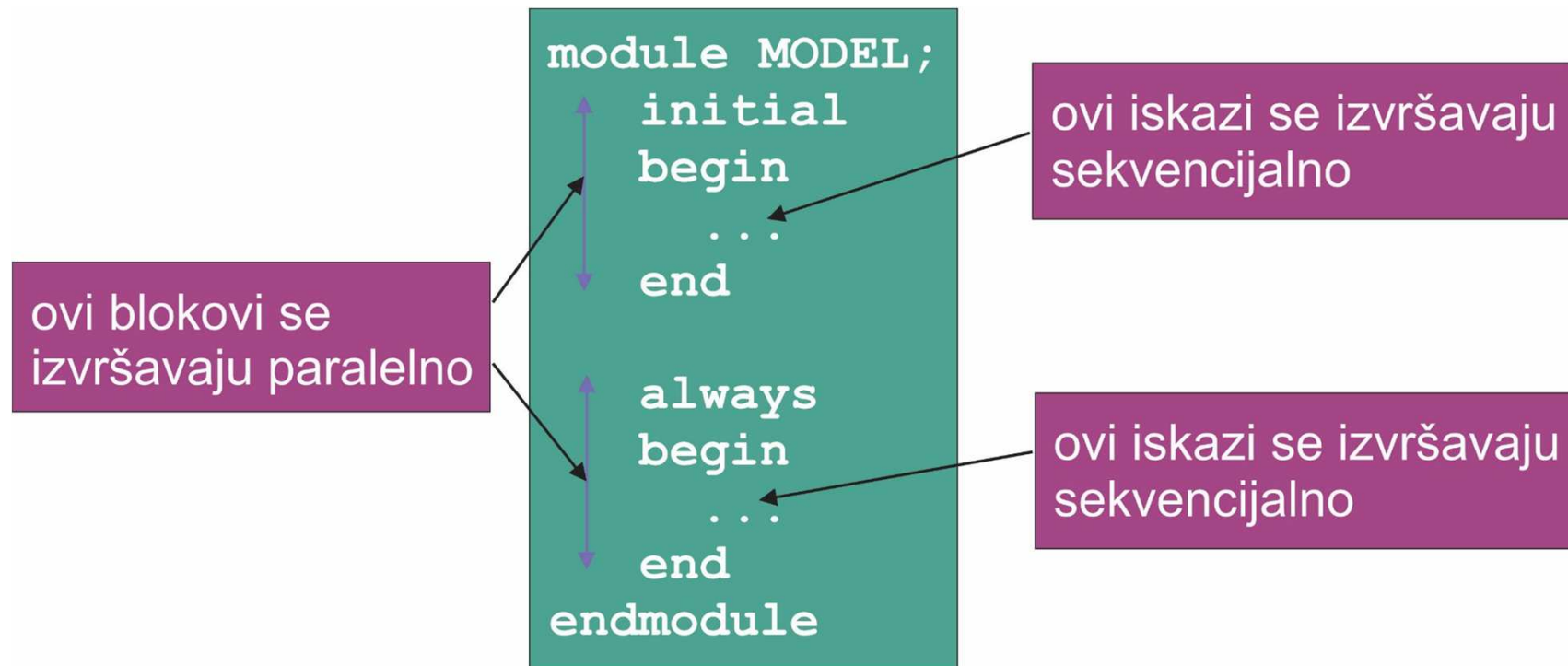
# Zašto je Verilog dobar?

- Sličan je programskim jezicima
  - koristi sekvencijalne iskaze
- Šta je potrebno još naučiti:
  - Konkurentni procesi
  - Pojam vremena

# Konkurentni procesi

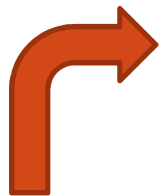
# Konkurentni procesi

- Proceduralni blok je osnovna jedinica konkurentnosti



# Konkurentni procesi

- Dilema: Računar izvršava sekvencijalne instrukcije
- Rešenje: Emulirati konkurentne procese
  - izvršiti jedan blok sekvencijalno
  - kada je blok završen, izvršiti drugi blok
- Želimo da blokovi izgledaju kao da se izvršavaju paralelno
- Kada je sa blokom završeno?
  
- Sa blokom je završeno kada je tako rečeno
- Sa blokom je završeno kada treba da čeka
- Blok može da čeka:
  - promenu vrednosti registra (događaj)
  - neko vreme (kašnjenje)
  - uslov koji treba da se ispuni



**Timing kontrole**

# Timing kontrole

- Zasnovane na kašnjenju
  - regularno kašnjenje `#10;`
  - kašnjenje u okviru zadatka `Y = #5 X + Z;`
  - kašnjenje nula `#0;`
- Zasnovane na događaju
  - regularni događaj `@(posedge CLK);`
  - imenovani događaj `event rx_data;`
  - događaj sa OR uslovom `@(posedge CLK or negedge RSTn);`
- Zavisi od nivoa `wait (cnt == 2);`

# @ iskaz

- @ iskaz se koristi da bi se čekalo na jedan ili više događaja
- Sintaksa:

```
@ ( [posedge | negedge] <name>  
    [{or [posedge | negedge] <name>}] );
```

- Edges:

Svaka promena vrednosti  
u ovom smeru je 'negedge'



Svaka promena vrednosti  
u ovom smeru je 'posedge'

# @ iskaz

- Primeri:
- @ (posedge CLK) ; čekanje na rastuću ivicu CLK
- @ (negedge RSTn) ; čekanje na opadajuću ivicu RSTn
- @ (STR) ; čekanje na bilo koju promenu STR
- @ (posedge CLK or  
negedge RSTn) ; čekanje na rastuću ivicu CLK ili  
opadajuću ivicu RSTn



# @ iskaz

- @ iskaz može da sadrži sekvencijalni iskaz
- Sintaksa `@ (<edge-control>) [<statement>]`
- Primer: `@ (posedge CLK) CNT = CNT + 1;`
- Ekvivalentno je sa:

```
@ (<edge-control> );  
  [<statement>]
```

```
@ (posedge CLK) ;  
CNT = CNT + 1;
```

# @ iskaz

- Popularan stil za sinhrono opise:

```
always @ (<event-control>)  
begin  
    {<statement>}  
end
```

- Primer:

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

Koristiti ovako ako se ne koristi nijedan drugi @ iskaz

# # iskaz

- # iskaz se koristi da bi se čekalo neki vremenski period
- Sintaksa: #<constant>;  
# (<expr>);
- Čeka se specificirani broj vremenskih jedinica

# # iskaz

- Primeri:

```
#10;  
#0;  
#(Tsetup);  
#(cycle/2);
```

Čeka se 10 v.j.  
Čeka se 0 v.j.  
Čeka se <izraz> v.j.  
Čeka se <izraz> v.j.

```
initial  
begin  
    CLK = 1'b0;  
    forever begin  
        #20;  
        CLK = ~CLK;  
    end  
end
```

generator takta sa  
faktorom ispune 50%


# # iskaz

- # iskaz može da uključi sekvencijalni iskaz
- Sintaksa: # (<expr>) [<statement>]
- Primer: #33 CLK = ~CLK;
- Ekvivalentno sa:

```
# (<expr> );  
[<statement>]
```

**BOLJE**

```
#33;  
CLK = ~CLK;
```



# Blocking assignment

- Blocking assignment se može koristiti za čekanje određeni vremenski period pre nego se izvrši dodeljivanje registru
- Sintaksa: `<target> = #(<time-expr>) <expr>;`
- Primer: `CLK = #50 ~CLK;`

# Blocking assignment

- Eliminirane potrebu za privremenim smeštanjem između semplovanja i dodeljivanja

- Ekvivalentno sa:

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

- Primer:

```
NEXT_CLK = ~CLK;  
#50;  
CLK = NEXT_CLK;
```

Složenije  
da se razume

# *wait* iskaz

- *Wait* iskaz se koristi za čekanje dok uslov ne postane TRUE
- Ne čeka se ako je uslov već TRUE
- Sintaksa: `wait (<expr>);`

- Primeri:

```
wait (RST != 1'b0);
```

```
wait (ACK == 1'b1 && ERR == 1'b0);
```

```
wait (1'b0);
```

Čeka zauvek






# *wait* iskaz

- Wait iskaz može da sadrži sekvencijalni iskaz
- Sintaksa: `wait (<expr>) [<statement>];`
- Primer: `wait (RST == 1'b1) CNT = 0;`
- Ekvivalentno sa:

```
wait (<expr>);  
[<statement>]
```

```
wait (RST == 1'b1);  
CNT = 0;
```

  
**BOLJE**

# ***Event driven simulacija***

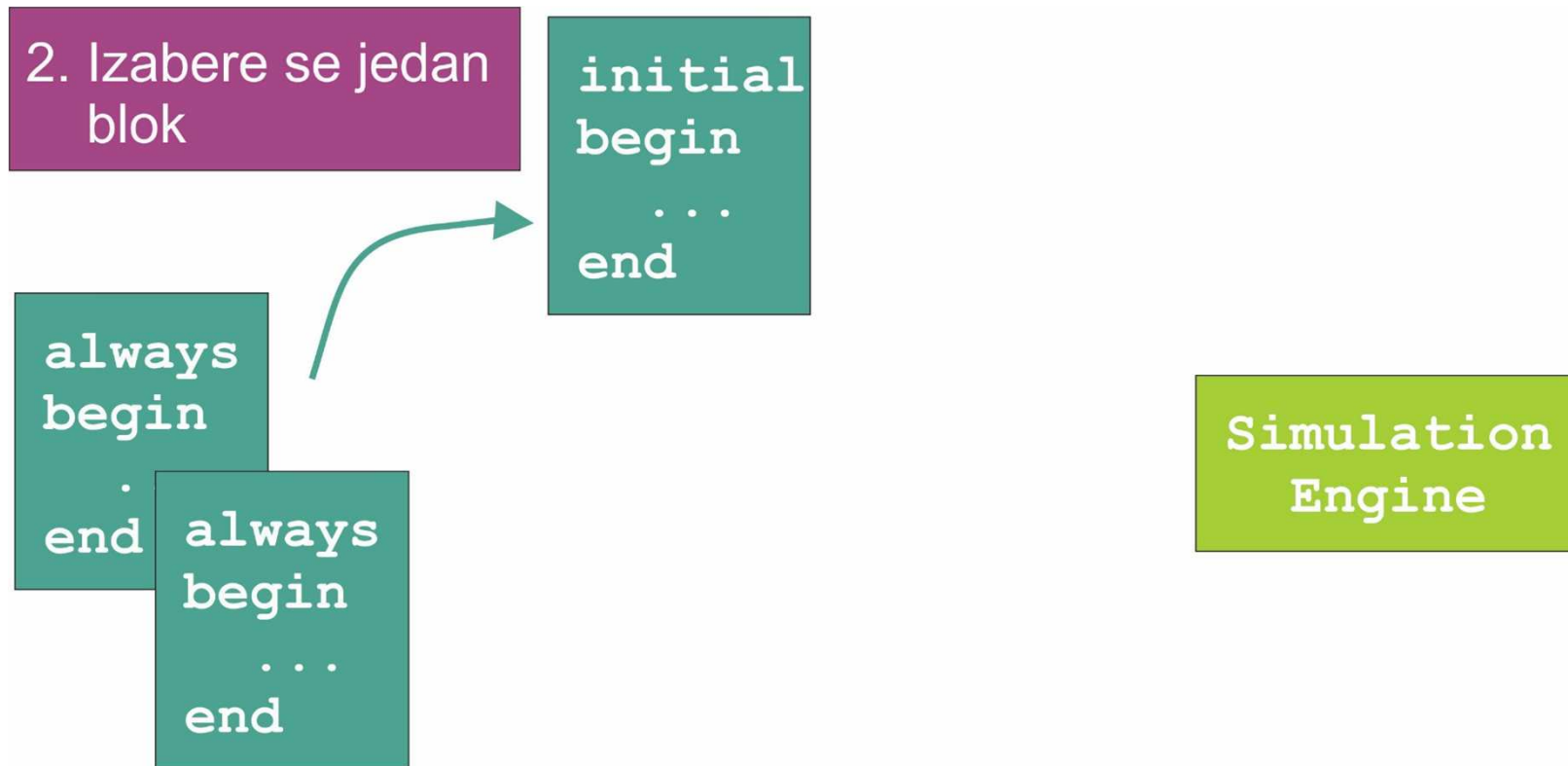
# Proces simulacije

1. Blokovi u «ready to execute pool-u»

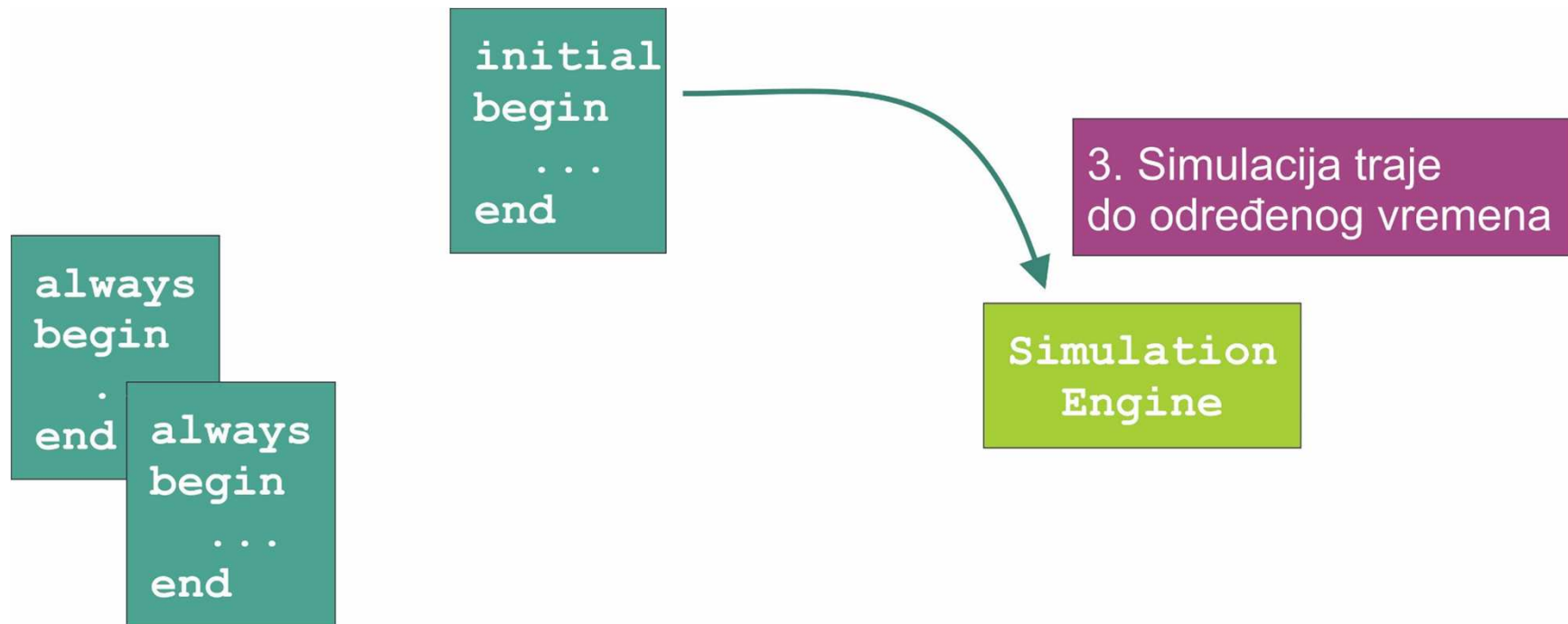
```
always  
b initial  
e be  
er always  
begin  
...  
end
```

Simulation  
Engine

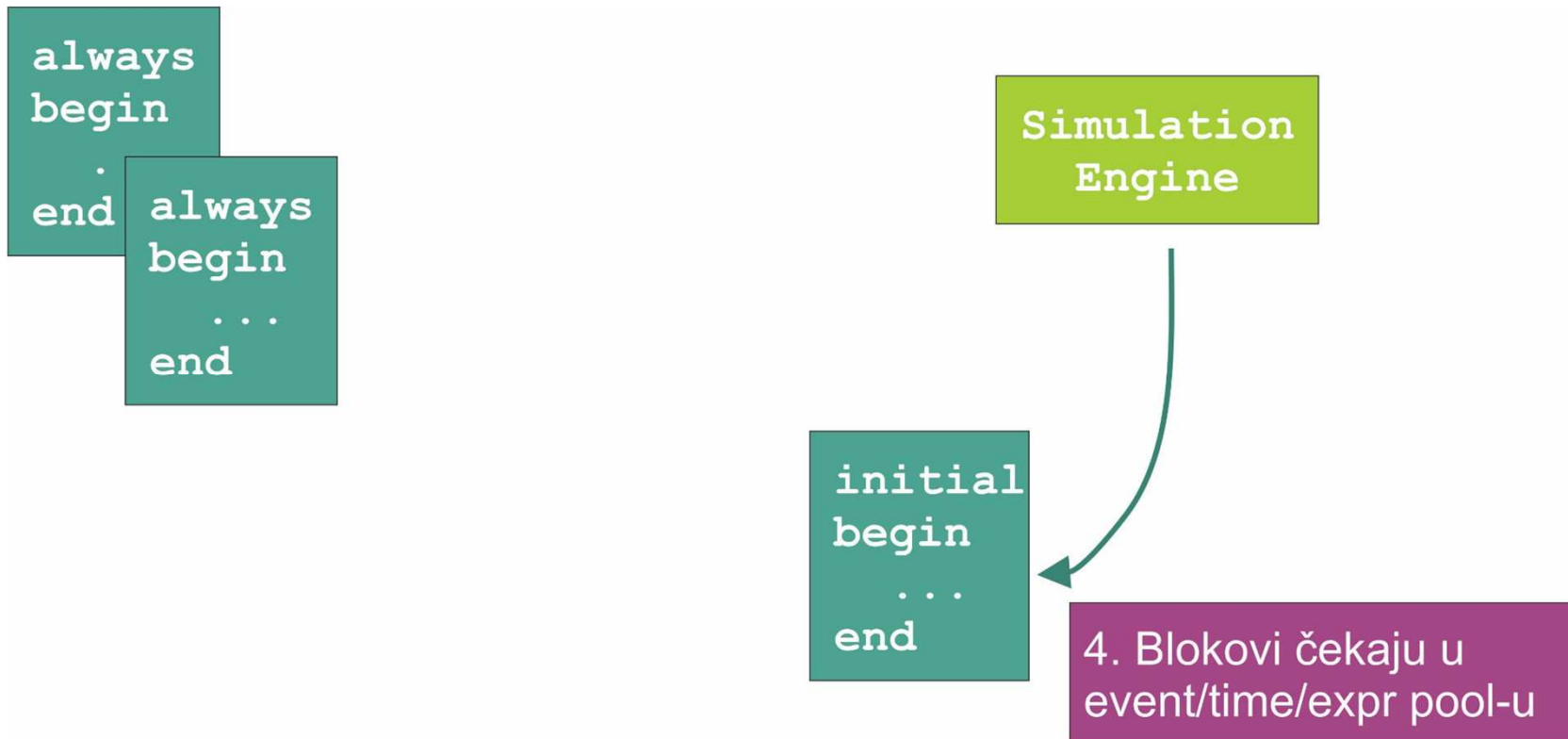
# Proces simulacije



# Proces simulacije



# Proces simulacije



# Proces simulacije

Simulation  
Engine

```
always  
begin  
    initial  
    ...  
end
```

```
always  
begin  
    ...  
end
```

5. Svi blokovi su u  
pool-u

# Proces simulacije

Blokovi u «ready to execute pool-u»

```
always  
begin  
initial  
begin  
always  
begin  
...  
end  
end
```

6. Vreme napreduje

Simulation  
Engine



# Proces simulacije

- Kako se proces inicijalizuje?
- Svi blokovi su stavljeni u „ready to execute pool“
- Svi registri se inicijalizuju na sve x-ove
  - Real registri se inicijalizuju na 0.0
- Još uvek nema vremenske kontrole

# Proces simulacije

- Prvo izvršavanje će se obaviti u istom redosledu kao što je deklarirano u modulu
  - sledeće izvršavanje je random
- Uputstvo: vaš model ne treba da zavisi od redosleda izvršavanja

# Proces simulacije

- Pitanje: koje je stanje simulacije posle 2 takta?

```
module MODEL;
reg CLK;
reg [3:0] CNT, FOO;

initial CNT = 0;

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

initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
...
```

```
...

always @ (negedge CLK)
begin
    FOO = FOO + 1;
end

always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
endmodule
```

# Proces simulacije

- Inicijalizacija:

CNT 

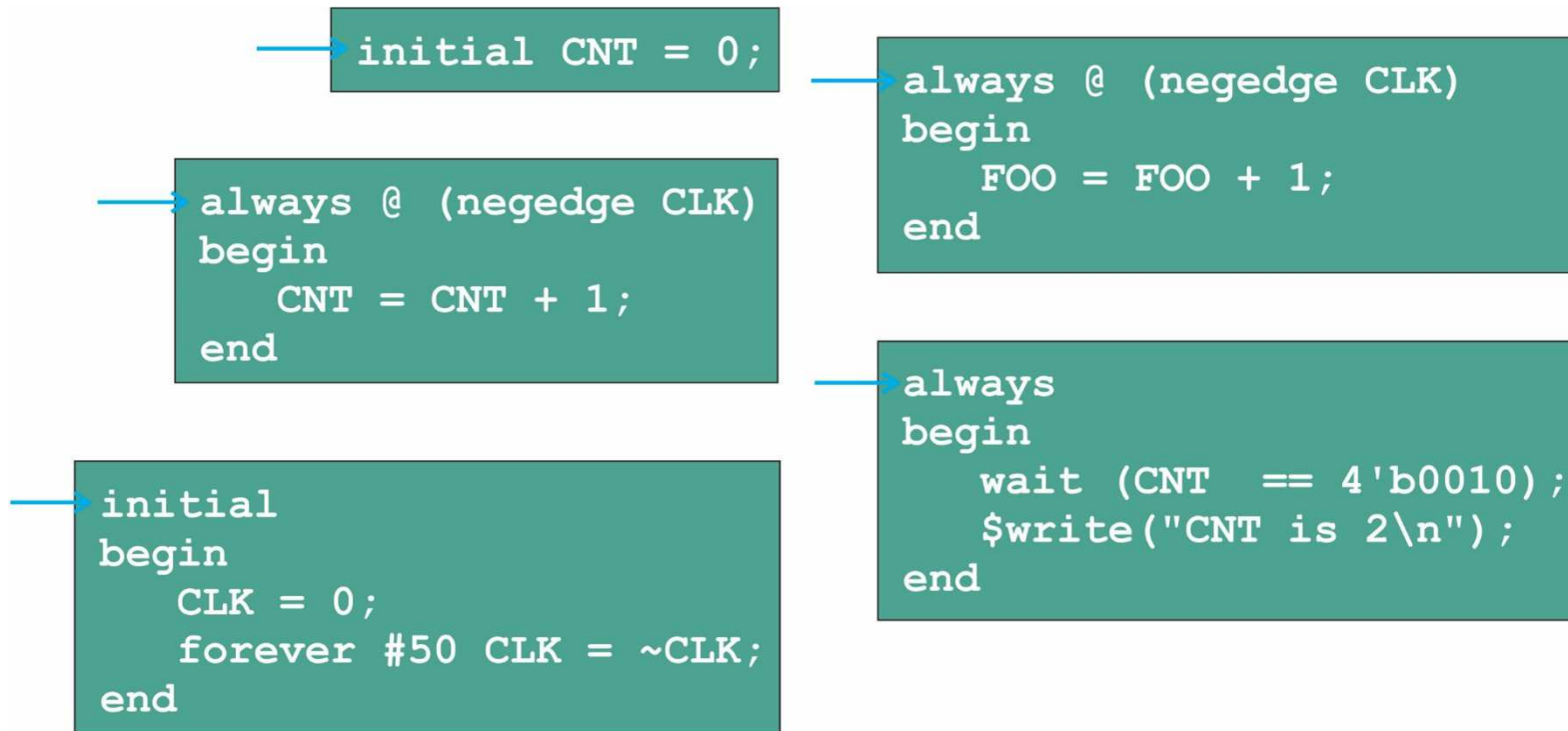
XXXX
------

  
FOO 

XXXX
------

  
CLK 

x
---



time: 0

# Proces simulacije

Ovaj blok se izvršava  
i onda se sklanja

CNT 0000  
FOO xxxx  
CLK x

```
initial CNT = 0;
```

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

```
initial  
begin  
    CLK = 0;  
    forever #50 CLK = ~CLK;  
end
```

```
always @ (negedge CLK)  
begin  
    FOO = FOO + 1;  
end
```

```
always  
begin  
    wait (CNT == 4'b0010);  
    $write("CNT is 2\n");  
end
```

time: 0

# Proces simulacije

CNT 0000  
FOO xxxx  
CLK x

Ovaj blok se izvršava  
i onda čeka na  
opadajuću ivicu

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

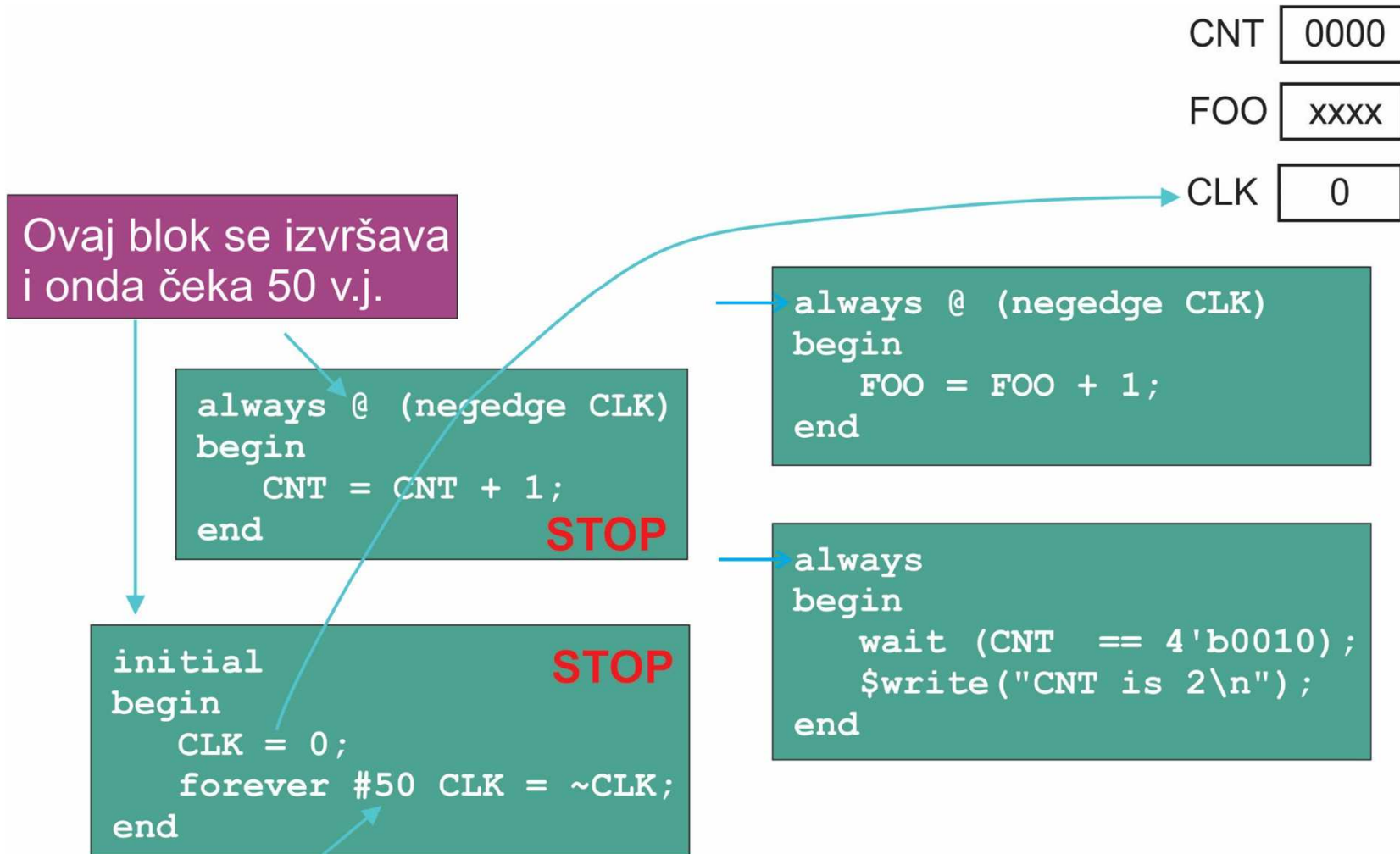
```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

time: 0

# Proces simulacije



time: 0

# Proces simulacije

CNT 

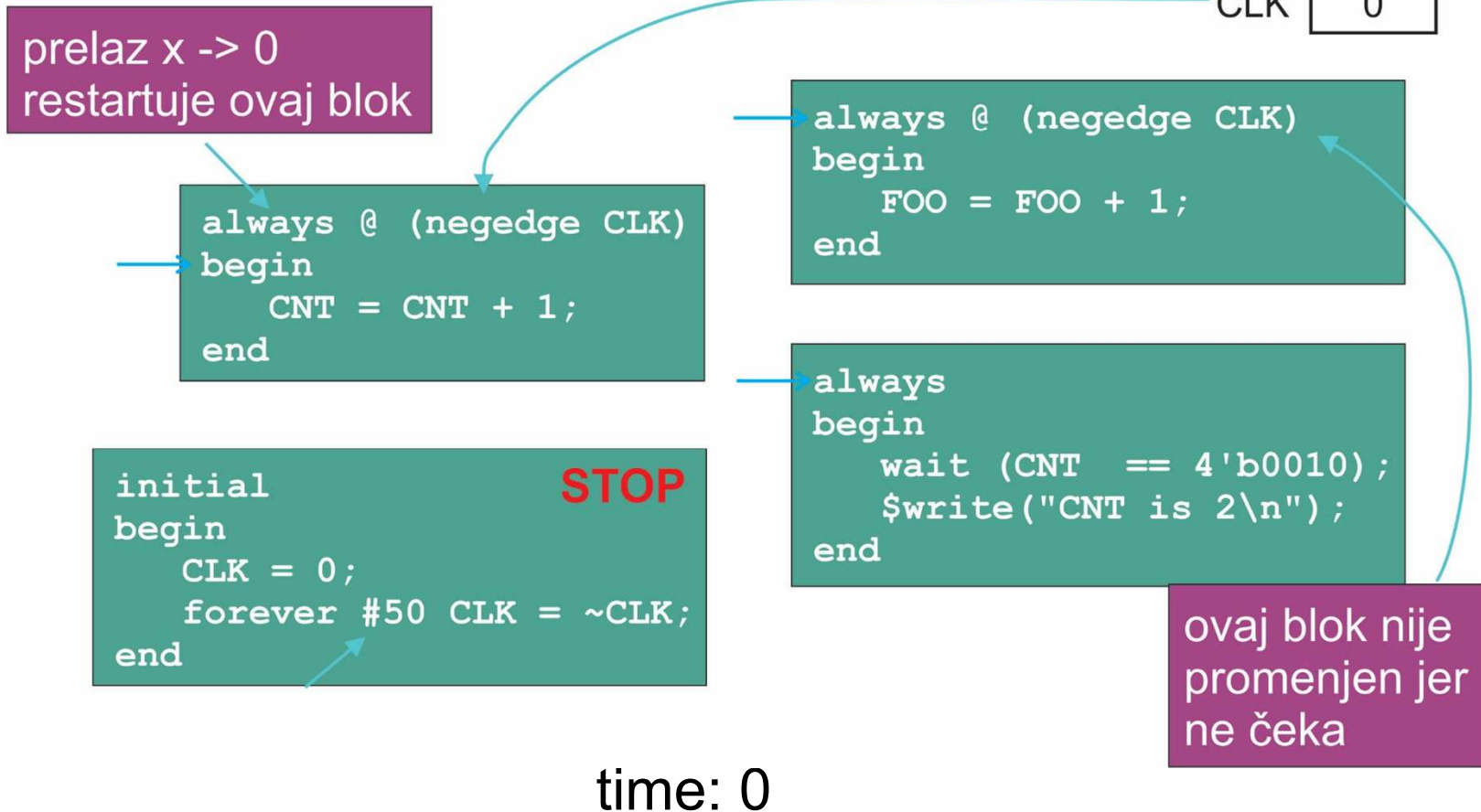
0000
------

  
FOO 

xxxx
------

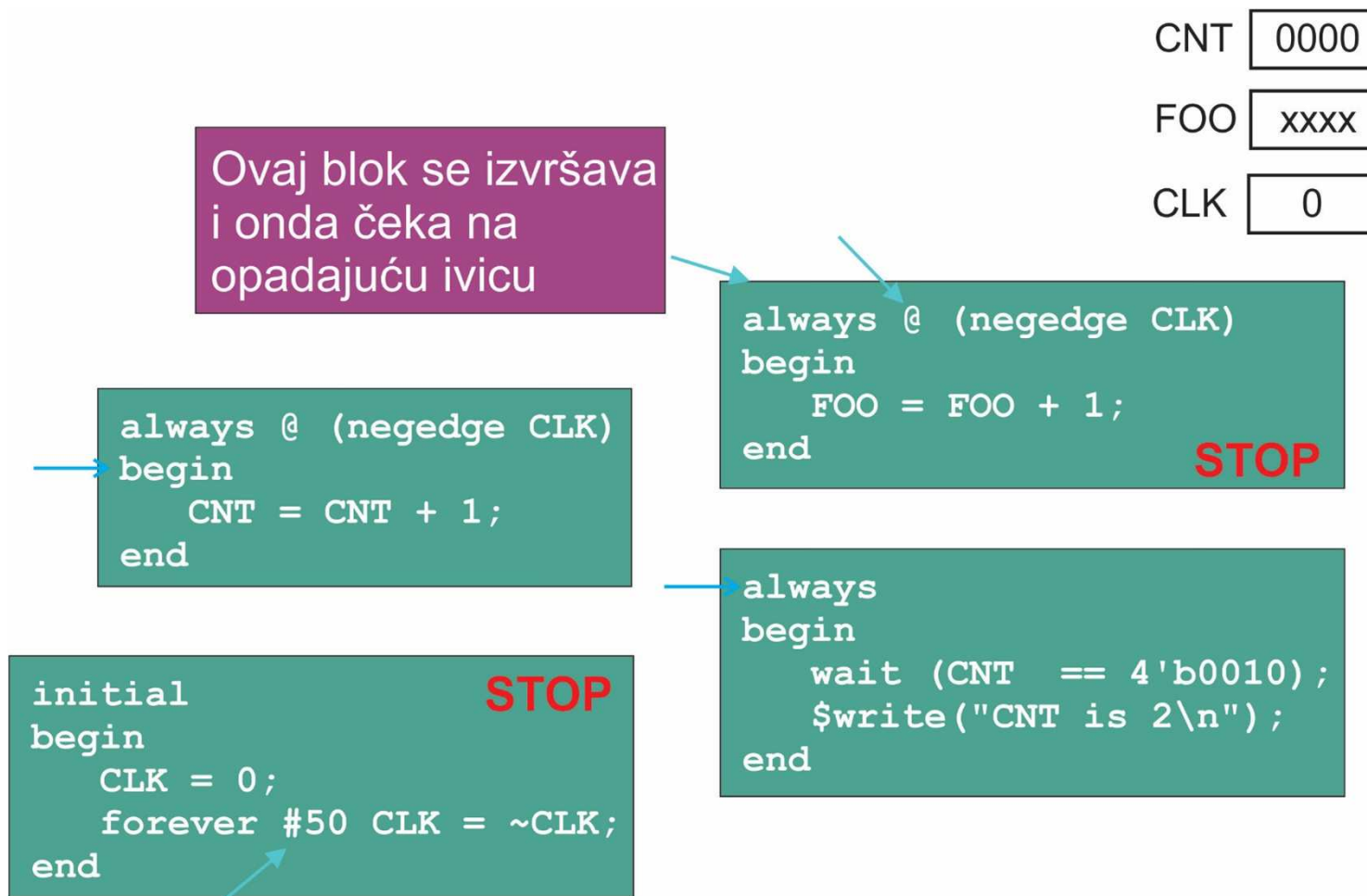
  
CLK 

0
---





# Proces simulacije



time: 0

# Proces simulacije

CNT 0000  
FOO xxxx  
CLK 0

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

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

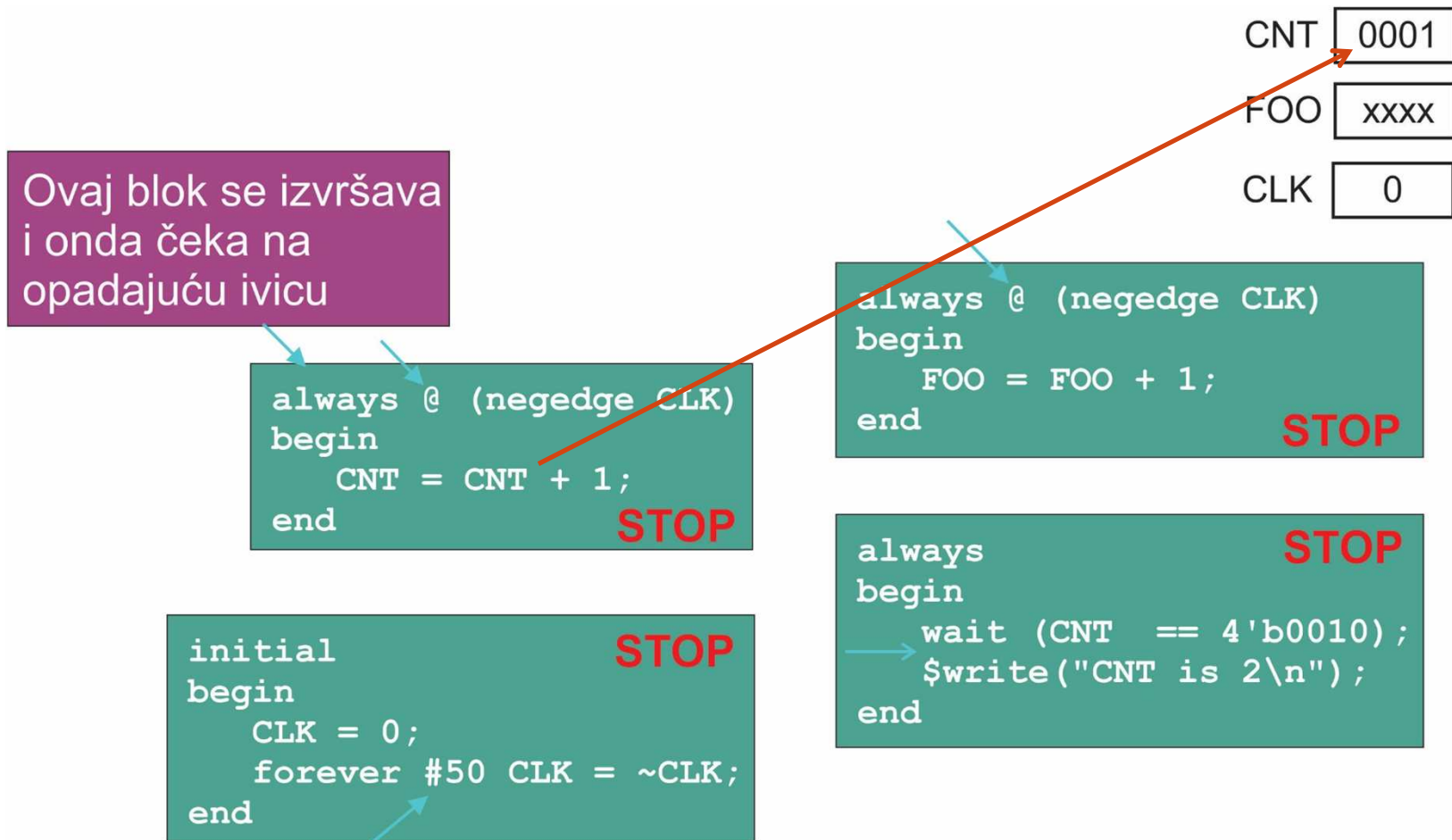
```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

**STOP**

Ovaj blok se izvršava i onda čeka uslov

time: 0

# Proces simulacije



time: 0

# Proces simulacije

CNT 

0001
------

FOO 

xxxx
------

CLK 

0
---

Svi blokovi čekaju:  
Vreme treba da  
napreduje

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always
begin
    → wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

**STOP**

time: 50

# Proces simulacije

CNT 

0001
------

  
FOO 

xxxx
------

  
CLK 

0
---

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

```
always
begin
    → wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

**STOP**

kašnjenje je proteklo  
pa se blok restartuje

time: 50

# Proces simulacije

Ovaj blok se izvršava i onda čeka 50 v.j.

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

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

**STOP**

CNT 0001

FOO xxxx

CLK 1

time: 50

# Proces simulacije

CNT 0001  
FOO xxxx  
CLK 1

Svi blokovi čekaju:  
Vreme treba da  
napreduje

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

**STOP**

time: 100

# Proces simulacije

CNT 0001  
FOO xxxx  
CLK 1

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

```
always
begin
    → wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

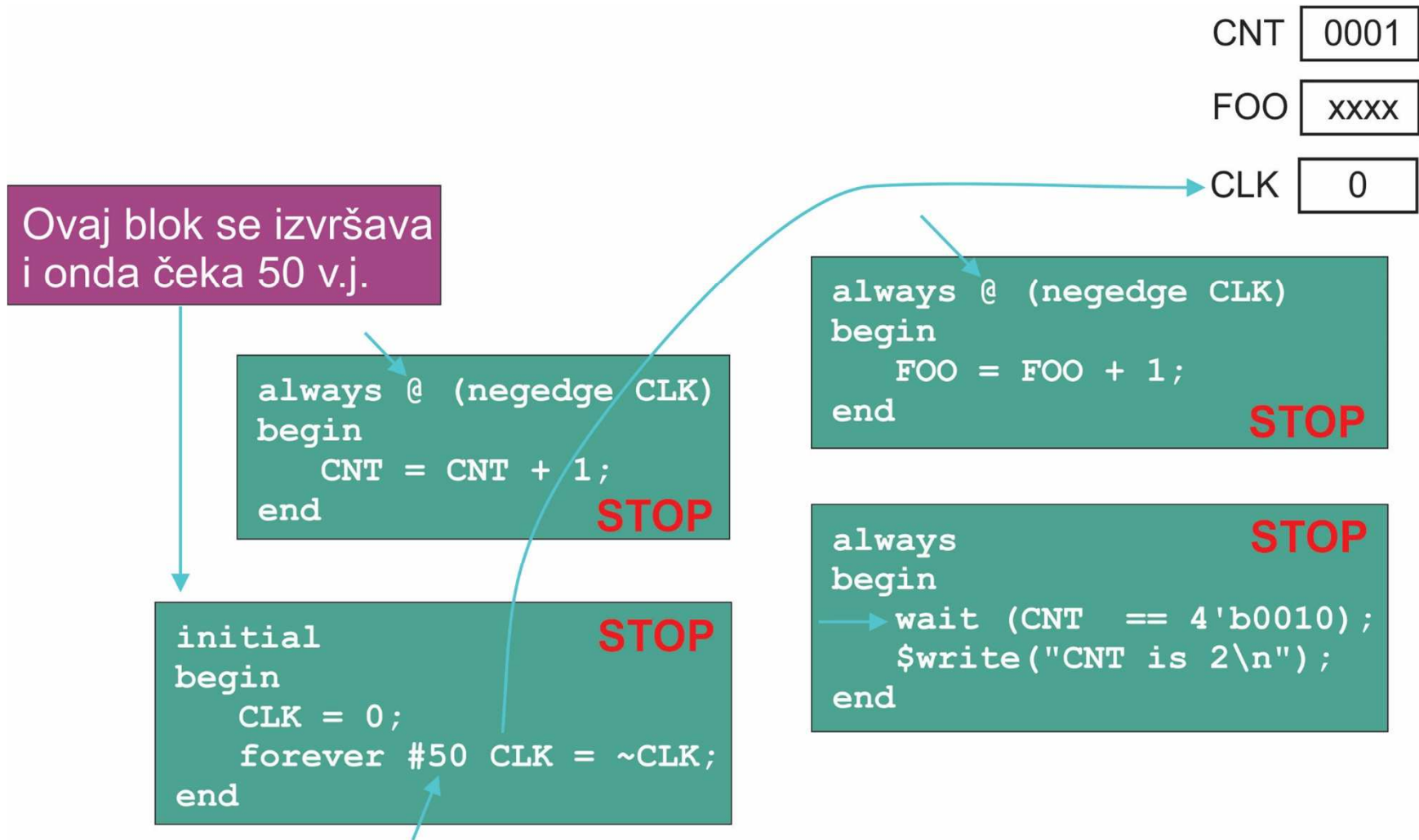
**STOP**

kašnjenje je proteklo  
pa se blok restartuje

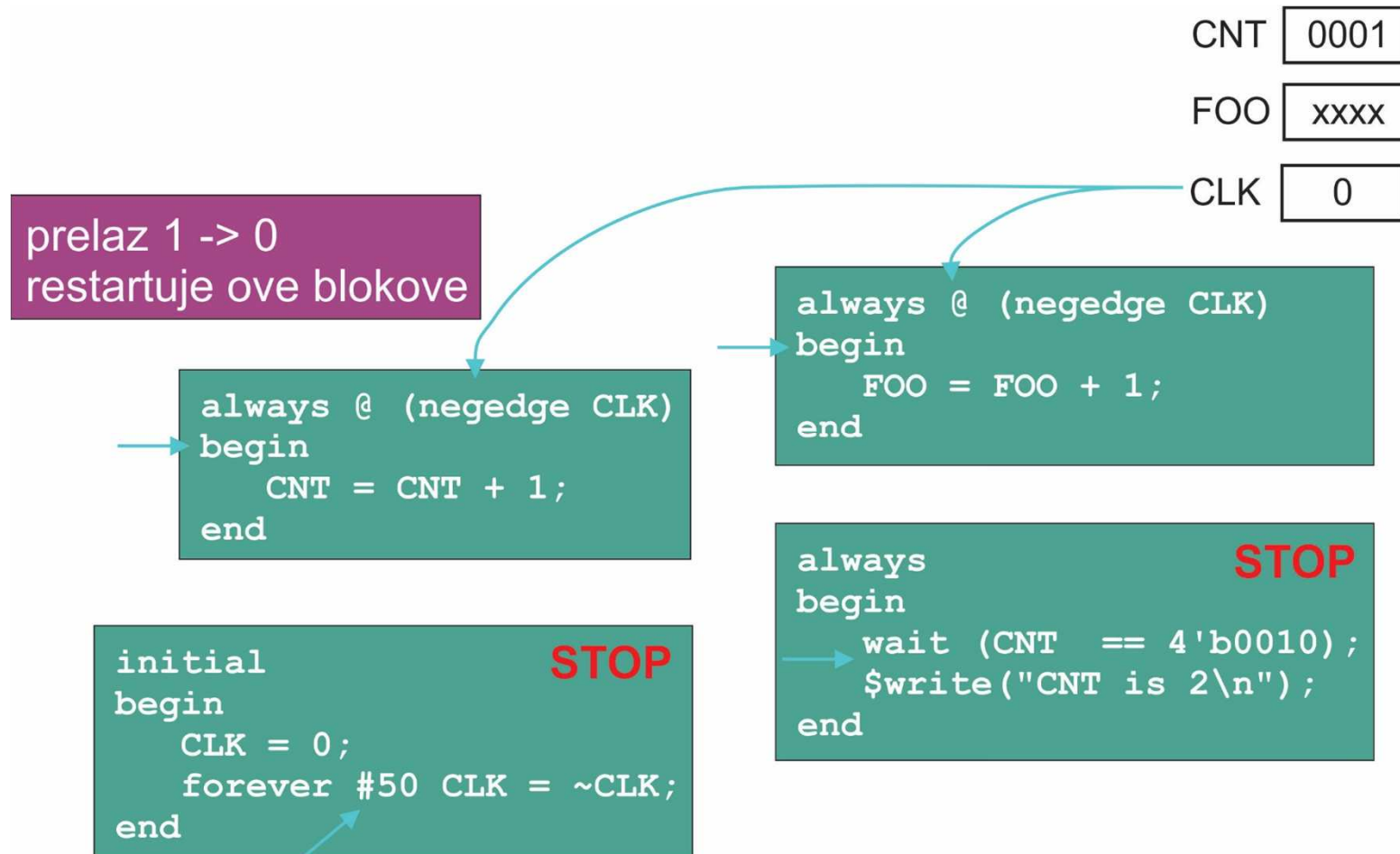
time: 100



# Proces simulacije

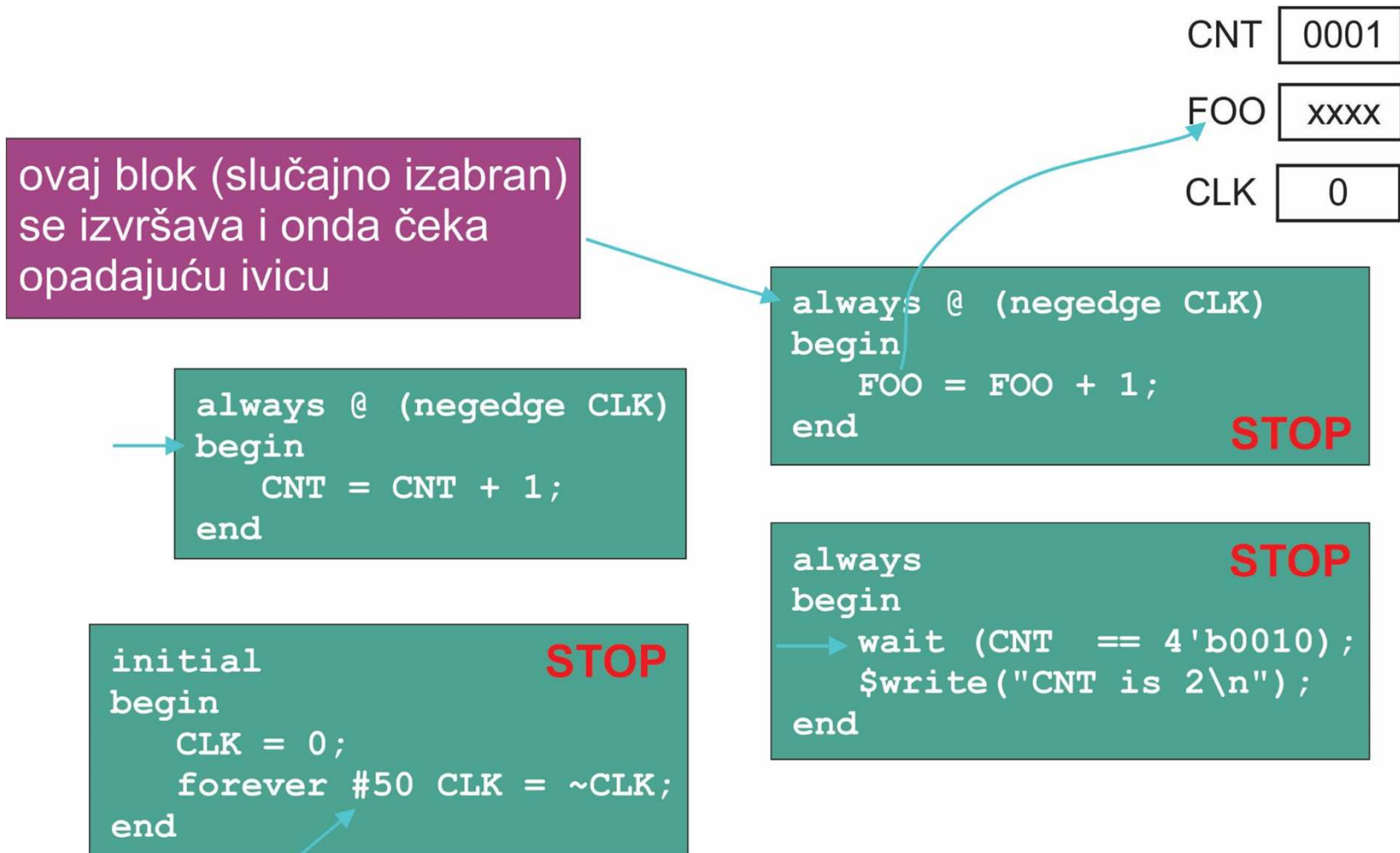


# Proces simulacije



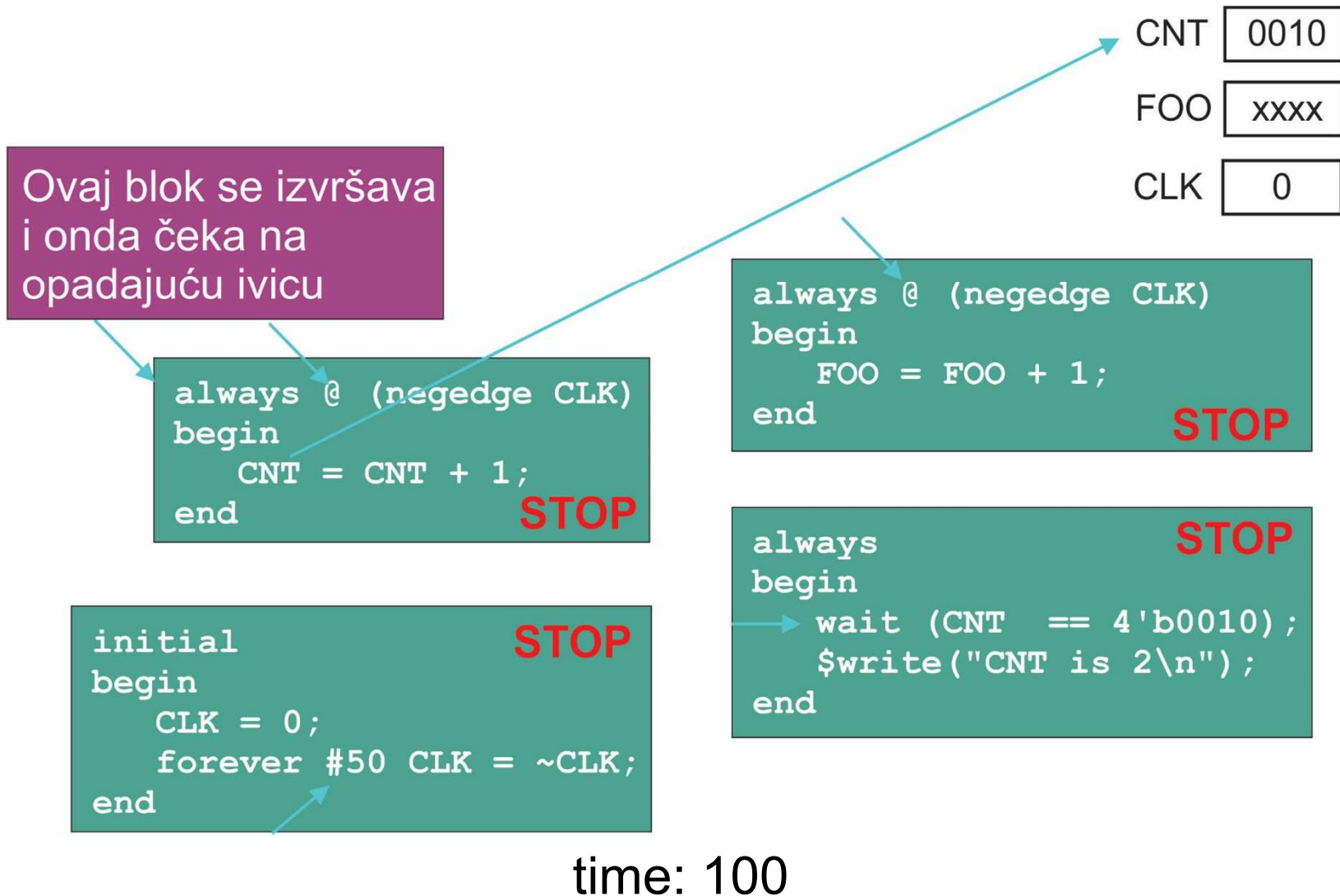
time: 100

# Proces simulacije



time: 100

# Proces simulacije



# Proces simulacije

CNT 0010  
FOO xxxx  
CLK 0

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

uslov je ispunjen pa se ovaj blok restartuje

time: 100

# Proces simulacije

CNT 0010

FOO xxxx

CLK 0

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

ovaj blok se izvršava  
zauvek jer je uslov  
uvek ispunjen

time: 100

# Proces simulacije

CNT 

0010
------

FOO 

xxxx
------

CLK 

0
---

Odgovor na pitanje:  
Simulacija nikada ne  
ulazi u drugi ciklus!

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

**STOP**

```
always @ (negedge CLK)
begin
    FOO = FOO + 1;
end
```

**STOP**

```
initial
begin
    CLK = 0;
    forever #50 CLK = ~CLK;
end
```

**STOP**

```
always
begin
    wait (CNT == 4'b0010);
    $write("CNT is 2\n");
end
```

ovaj blok se izvršava  
zauvek jer je uslov  
uvek ispunjen

time: 100