PROJEKTOVANJE INTEGRISANIH KOLA SA MEŠOVITIM SIGNALIMA

1

Verilog

Doc. dr Miona Andrejević Stošović

Dejan Mirković, asistent

Verilog kao HDL

- Jednostavan za učenje
- Sintaksa je slična *C*-u
- Dizajn može biti opisan na visokom nivou apstrakcije
- Funkcionalna verifikacija dizajna može da se uradi u ranoj fazi projektovanja

Šta možemo da uradimo sa Verilogom?

- Modelovanje digitalnih logičkih kola
- Simulacija i verifikacija digitalnih logičkih kola
- Sinteza digitalnih logičkih kola

Šta možemo da uradimo sa Verilogom?

Modelovanje digitalnih logičkih kola

- Modelovanje na bihevioralnom nivou
- Modelovanje na RTL nivou
- Modelovanje na nivou gejtova
- Modelovanje na mixed nivou

Šta možemo da uradimo sa Verilogom?

Simulacija i verifikacija velikih digitalnih logičkih kola

- Testbenčevi
- Bus-functional modeli (za verifikaciju)
- Full timing simulacija

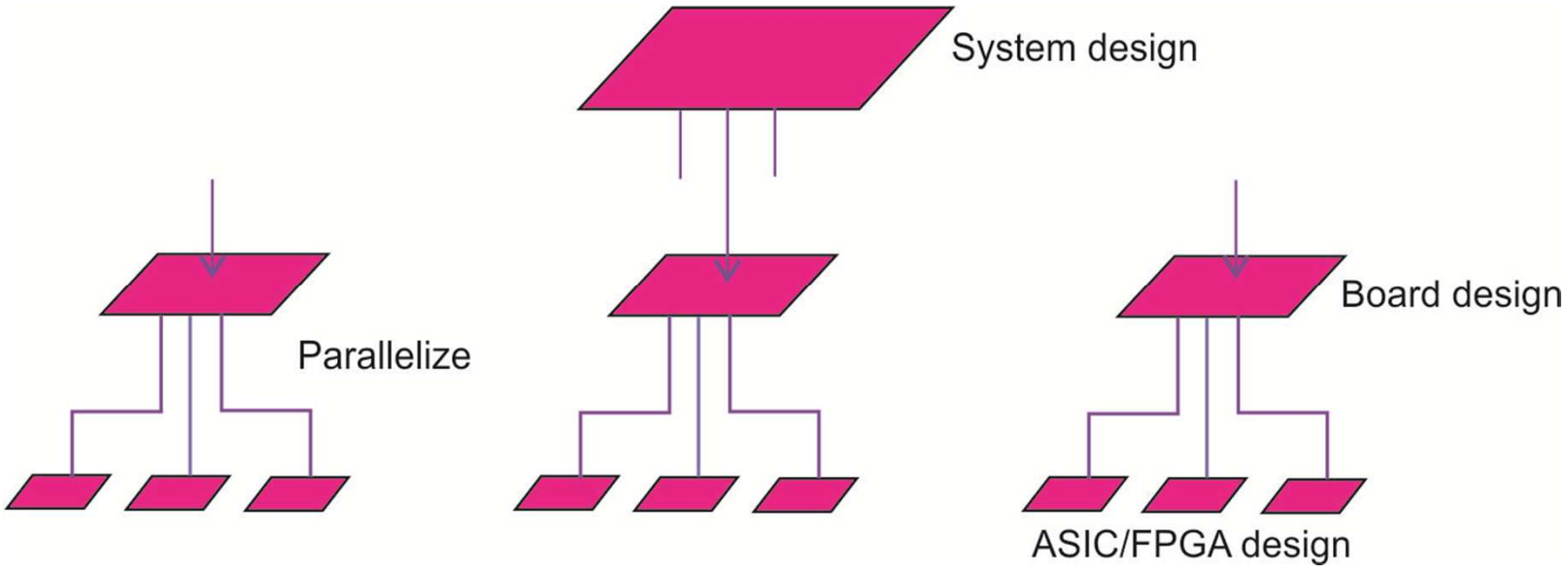
Šta možemo da uradimo sa Verilogom?

Sinteza digitalnih logičkih kola

- Osnove RTL kodovanja- podskup Veriloga
- Kombinaciona kola
- Sekvencijalna kola

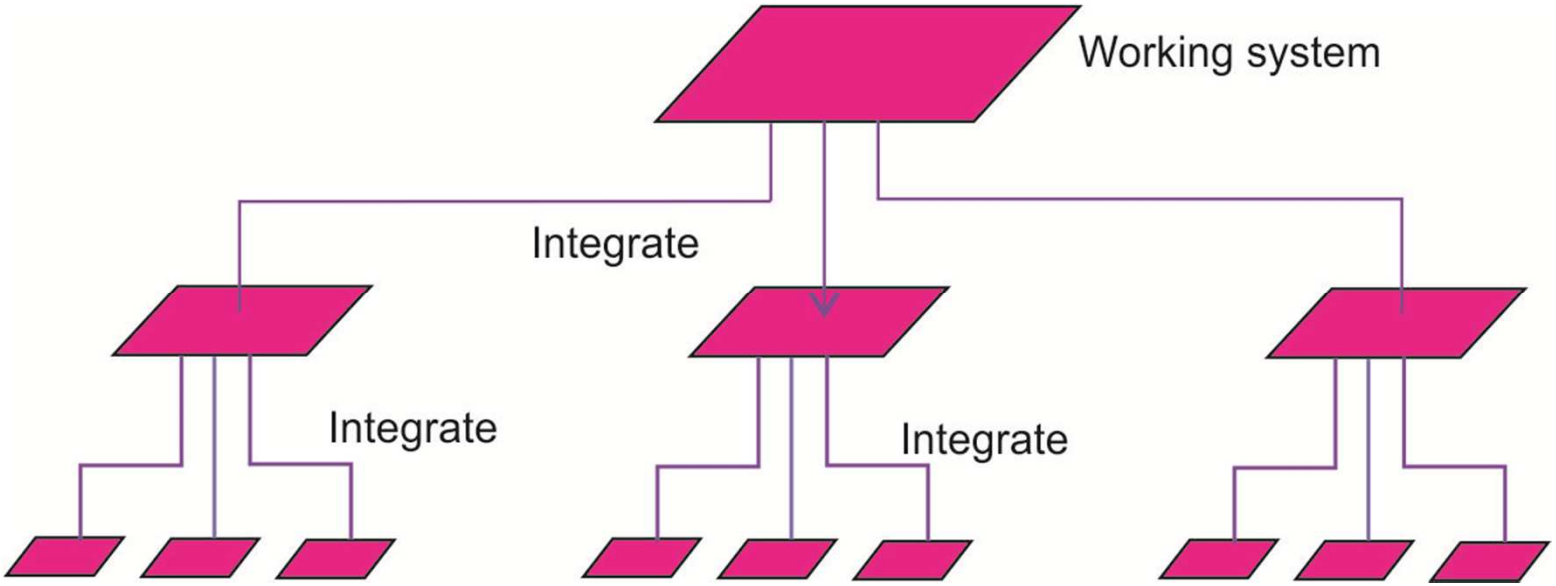
High-level design

Top Down Methodology

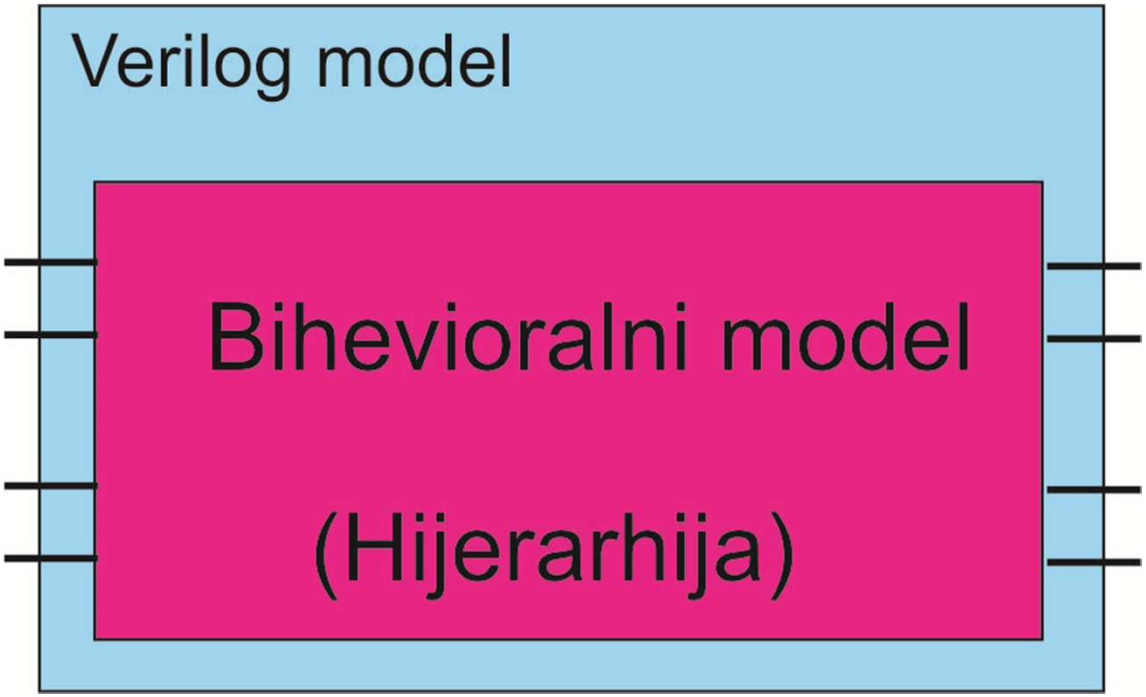


High-level design

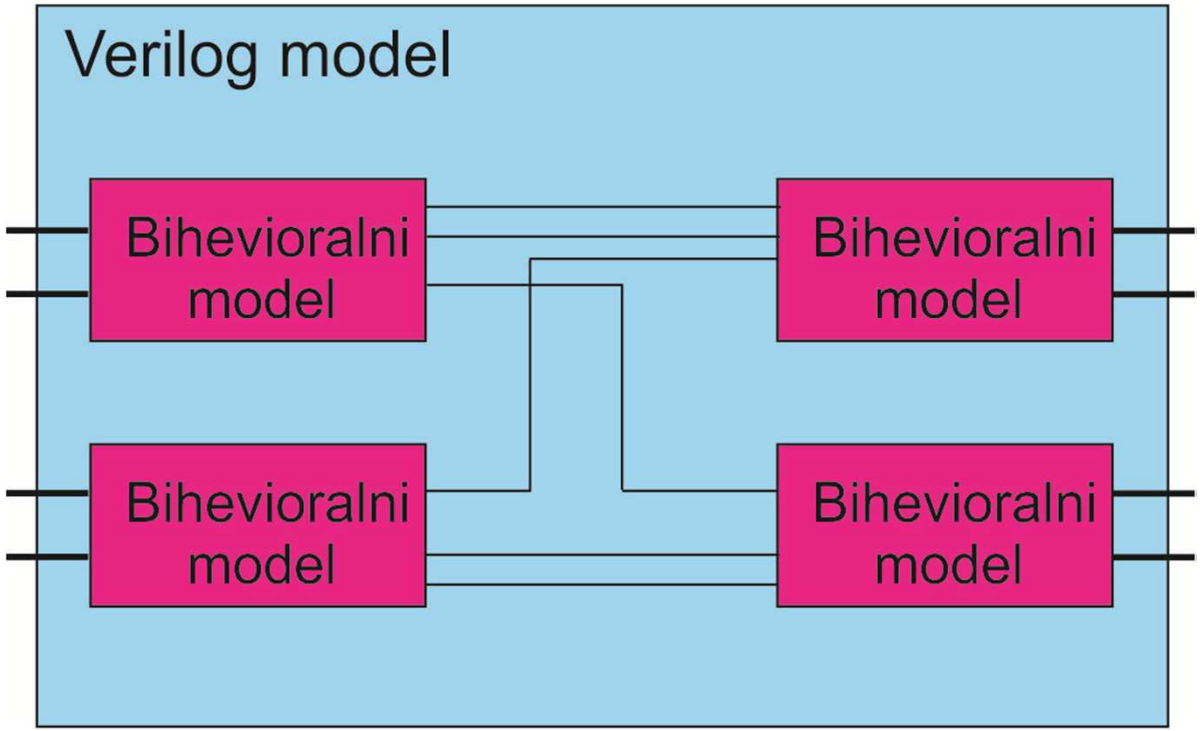
Bottom Up Methodology



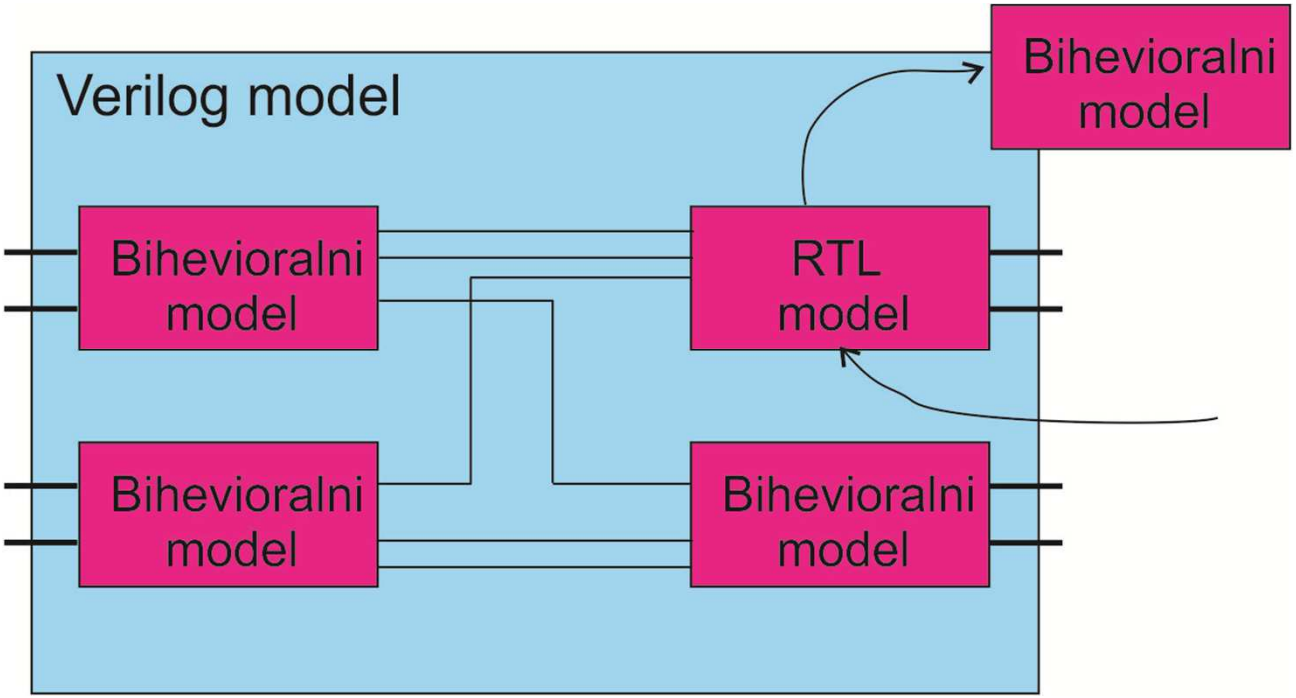
Arhitektura



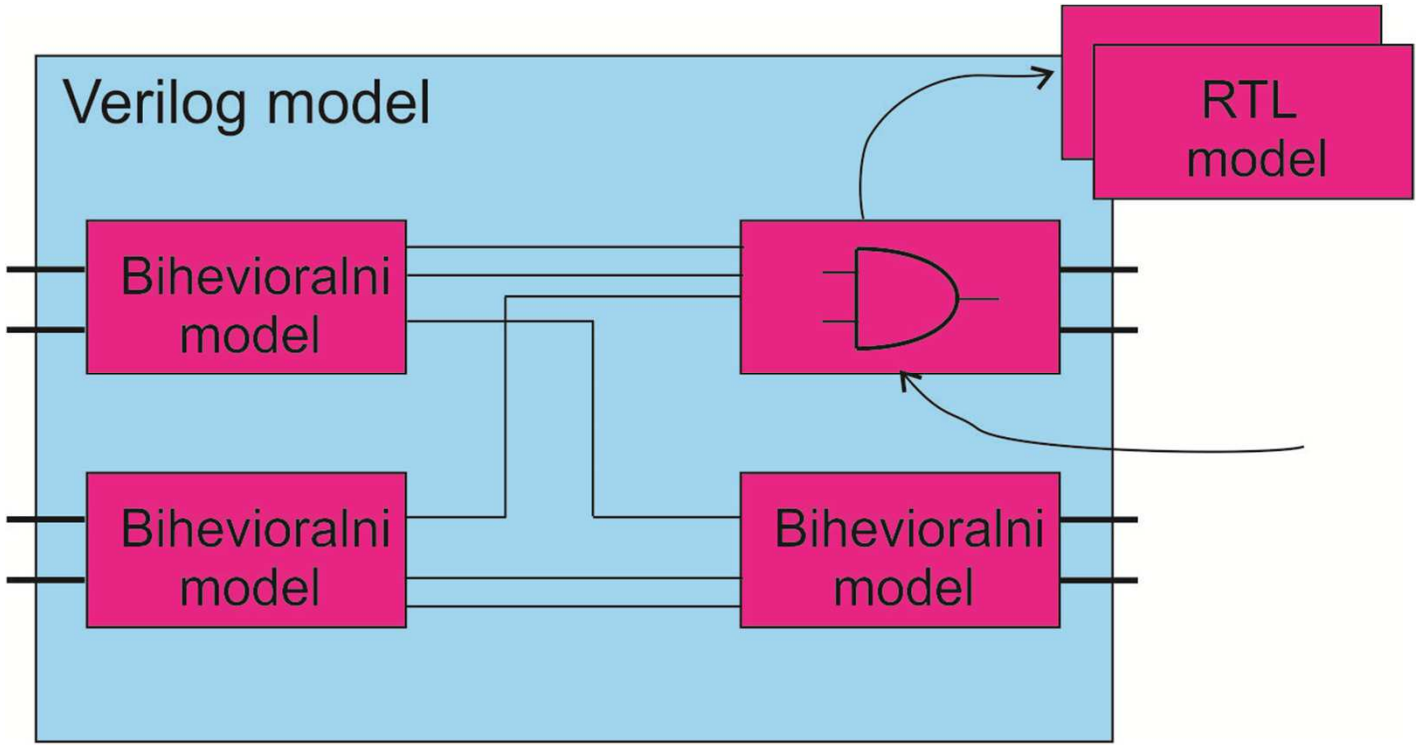
Bihevioralni dizajn



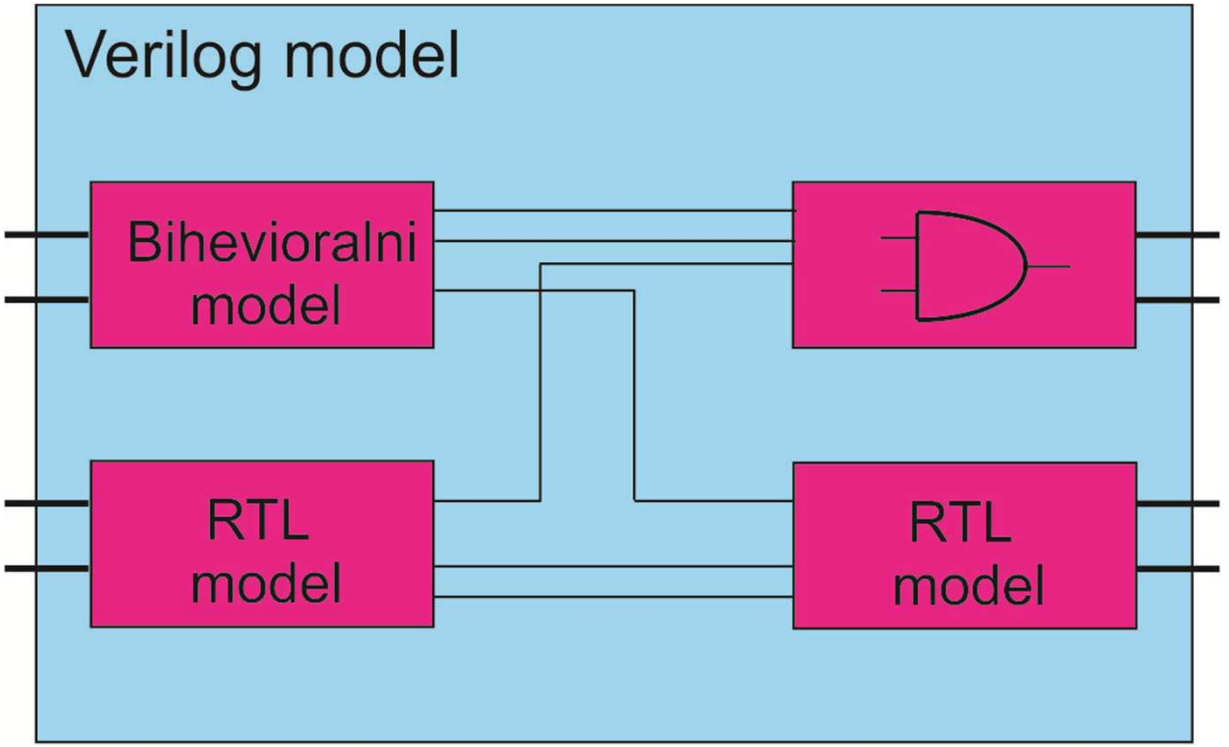
Ubacivanje RTL modela



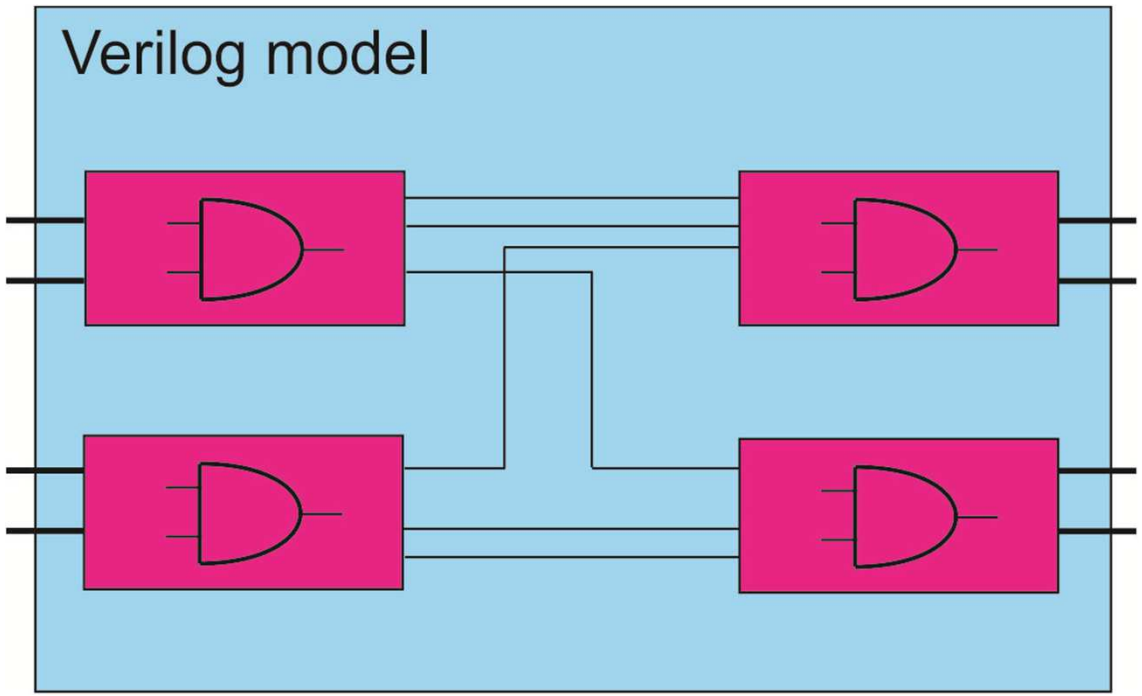
Ubacivanje modela na nivou gejta



Mixed-level modelovanje



Gate-level dizajn



Ubaciti vremenska ograničenja i raditi simulaciju

Uvod u Verilog

Osnovna jedinica

Modul

- Sintaksa:

```
module <name>;
endmodule
```

- Primer:

```
module PIKMS;
endmodule
```

Sintaksna pravila

- Case sensitive- osetljiv na veličinu slova
- Rezervisane reči se pišu malim slovima

Napomena:
Identifikatori koje definišemo mogu da pomognu drugima da prepoznaju šta smo radili. Pravite kôd onako kako biste želeli da ga nasledite od nekog drugog:
Tx_fifo_rd, Tx_fifo_rreg; Tx_fifo_wr...
Umesto TR trr twr

Sintaksna pravila

Rezervisane reči

```
module PIKMS;
endmodule
```

- ne mogu se koristiti kao identifikatori
- Listu rezervisanih reči možete naći u bilo kom priručniku za Verilog

Sintaksna pravila

User-defined identifikatori

```
module PIKMS;
endmodule
```

```
module pikms;
endmodule
```

```
module PikMs;
endmodule
```

3 različita modula

Sintaksna pravila

Identifikatori:

- karakteri A-Z, a-z, 0-9 i ‘_’
- moraju početi slovom

Primeri:

```
module ADD8;
endmodule
```

```
module ATM_SWITCH;
endmodule
```

pogrešno!

```
module INVALID!;
endmodule
```

```
module ONE*TWO;
endmodule
```

Sintaksna pravila

- ‘indentation’- uvlačenje nije značajno
- novi red nije značajan

Napomena:

- uvlačenje radi lakšeg praćenja toka
- novi red gde je to pogodno

```
module PIKMS;  endmodule
```

```
module PIKMS;

endmodule
```

Komentari

- Blok komentari
  - počinju sa “/*”
  - završavaju sa “*/”
  - ne mogu biti ugnježdeni

```
/*
 * U ovom modulu je opisan...
 *

*/
```

- Linijski komentari
  - počinju sa “//”
  - završavaju na kraju linije

```
module PIKMS;
...

endmodule // ovaj modul opisuje...
```

Source fajlovi

- Verilog čita ulazne fajlove kao niz teksta
- Moduli mogu biti u mnogo fajlova
- Više modula može da bude u jednom fajlu

- Napomena:
  - Staviti module u posebne fajlove

Source fajlovi

- Imena fajlova su proizvoljna
- Napomena:
  - Koristiti ".v" kao ekstenziju
  - isto ime kao i modul koji sadrži

mod1.v

```
module MOD1;
endmodule
```

mod2.v

```
module MOD2;
endmodule
```

Kompajliranje Verilog modela

- Kompajliranje se obavlja u tri faze:
  - Parsiranje sors fajla i provera sintakse
  - Hijerarhijsko povezivanje
  - Forward reference resolutions

- Različite vrste greški se prijavljuju u toku različitih faza

“Interpretive” simulacija

- Čitav model se rekompajlira svaki put
- Kompilacija je sakrivena simulacijom

- xxxxxxxxxxxxxxxxxxxxxx

Kompajlirana simulacija

- Model se rekompajlira postepeno
  - rekompajlira se samo ono što je promenjeno
- Kompilacija je odvojena od simulacije

Kompilacija Simulacija

- xxxxxxxxxxxxxxxxxxxxxx

Uvod u modelovanje

- Modelovanje hardvera je samo primena Veriloga
- Samo modelovanje se obavlja pisanjem softvera, tako da smo mi sada softverski inženjeri ☺

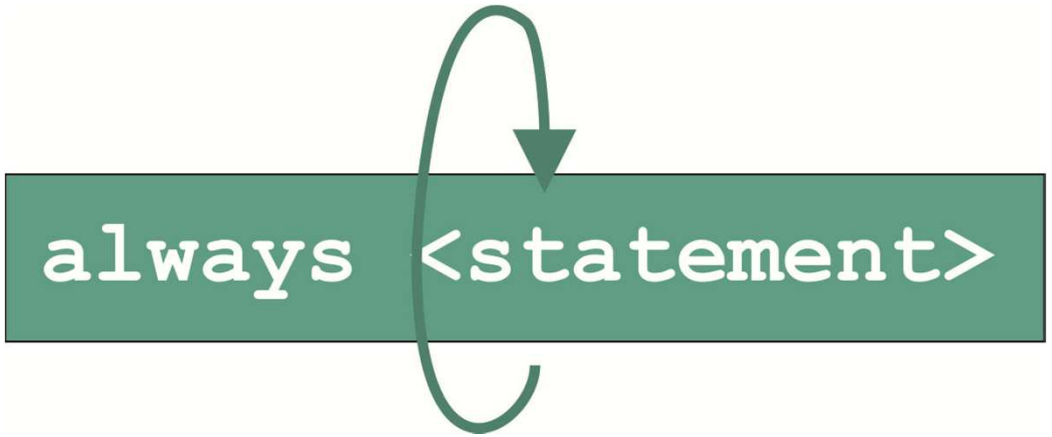
Sekvencijalni Verilog

always blok

- ***always* blok** je program koji je zaglavljen u beskonačnoj petlji
- Izvršava se sekvencijalno, od početka prema kraju
- Jednom kada stigne do kraja, izvršavanje počinje ponovo od početka
- Sintaksa:

```
always <statement>
```

always blok



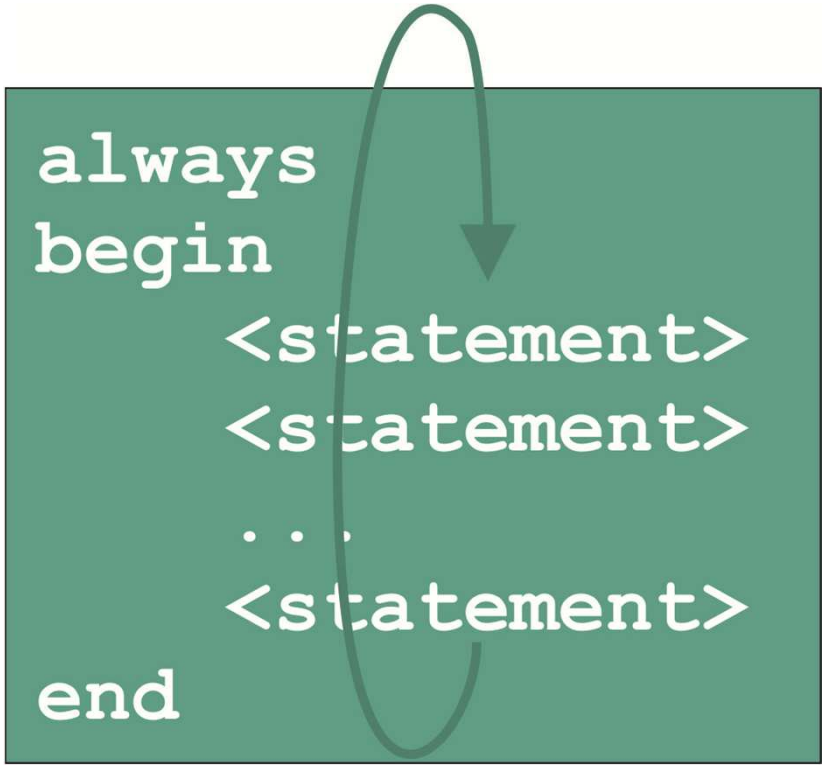
begin-end blok

- Izvršavanje jednog iskaza beskonačno nije vrlo korisno
- ***begin-end* blok** čini da nekoliko iskaza kompajleru „izgledaju“ kao jedan iskaz

- Sintaksa:

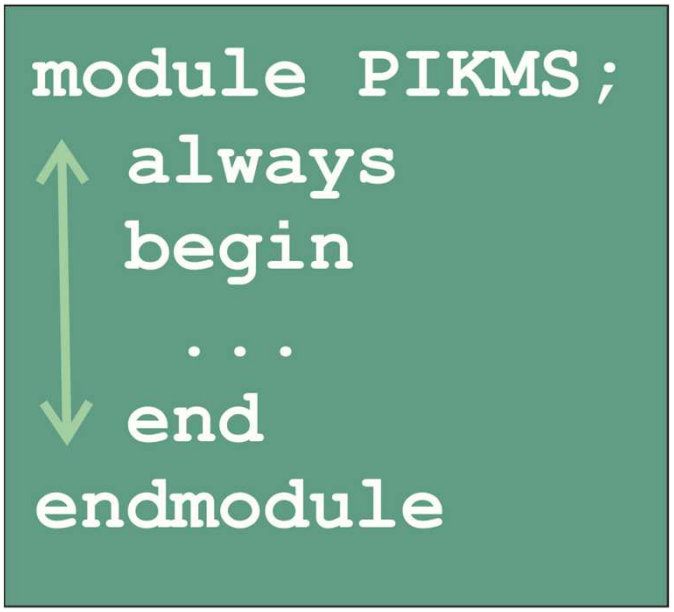
```
begin
    (<statement>)
end
```

always blok



always blok

***always* blok** ide unutar modula



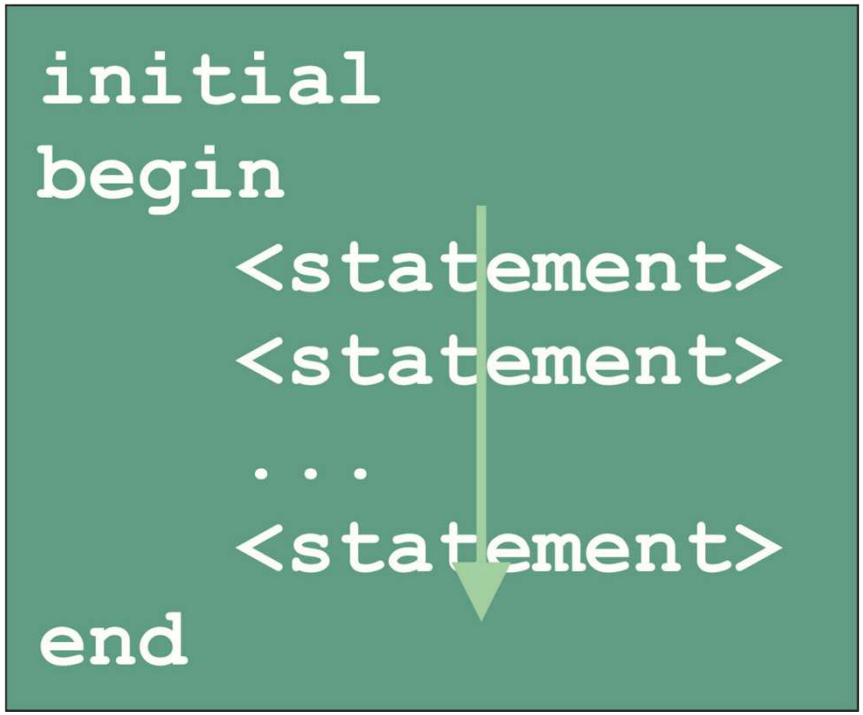
initial blok

- ***initial* blok** se izvršava samo jednom
- izvršava se sekvencijalno, od početka do kraja
- kad se stigne do kraja, izvršavanje se prekida

- Sintaksa:

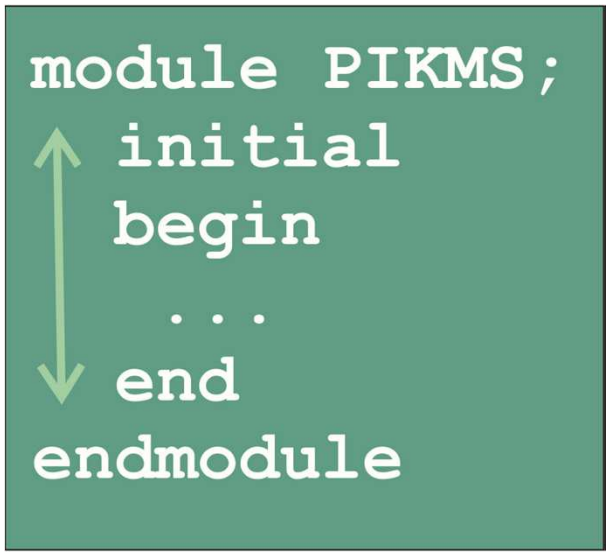
```
initial <statement>
```

initial blok



initial blok

***inital* blok** takođe ide unutar modula



$write iskaz

- Prikazuje poruku na standardnom izlazu
- Poruka mora biti u jednoj liniji
- Nova linija mora biti dodata eksplicitno
  - koristiti “\n” da bi se ubacila nova linija

- Sintaksa:

```
$write (<string>);
```

- Primer:

```
$write(”Hello World\n”);
```

$finish iskaz

- Prekida simulaciju
- Simulacija se sama zaustavlja ako nema šta da radi

- Sintaksa:

```
$finish;
```

Logičke vrednosti

- Verilog ima logički sistem sa 4 vrednosti
  - 0
  - 1
  - Z, ? visoka impedansa
  - X nepoznata

Binarne vrednosti

- Validne vrednosti: 0, 1, z, Z, ?, x, X, _
- Jedan bit po vrednosti
- Popunjava se nulama do 32 bita
  - popunjava se sa X ili Z ako je to bit najveće težine
  - nije signed-extended

ignoriše se
služi kao razmak

- Sintaksa:

```
'b{<digit>}
'B{<digit>}
```

- Primeri:

```
'b0      00..0000000
'B10xz   00..00010xz
'bx      xx..xxxxxxx
```

Decimalne vrednosti

- Validne vrednosti: 0-9
- Ne postoji X ni Z
- Popunjava se nulama do 32 bita
  - nije signed-extended

- Sintaksa:

```
'd{<digit>}
'D{<digit>}
```

- Primeri:

```
'd7      00..0000111
'D12     00..0001100
'd1      00..0000001
```

Heksadecimalne vrednosti

- Validne vrednosti: 0-9, a-f, A-F, z, Z, ?, x, X
- Četiri bita po cifri
- Popunjava se nulama do 32 bita
  - popunjava se sa X ili Z ako je to bit najveće težine
  - nije signed-extended

- Sintaksa:

```
'h{<digit>}
'H{<digit>}
```

- Primeri:

```
'h1f      00..00000000011111
'H17xz    00..010111xxxxzzzz
'Hx       xx..xxxxxxxxxxxxxx
```

Vrednosti određene dužine

- Ako želimo da broj bitova bude različit od 32
- Moguće je za bilo koju brojnu osnovu
- Suvišni bitovi se odsecaju
- Popunjava se nulama do odredjenog broja bitova
  - popunjava se sa X ili Z ako je to bit najveće težine
  - nije signed-extended

- Sintaksa:

```
<size>'b{<digit>}
<size>'d{<digit>}
<size>'h{<digit>}
```

Primeri za vrednosti određene dužine

```
1'b1                                1
'b1          00..00000000000001
8'hF0F                      00001111
7'bx                         xxxxxxx
7'o37                        0011111
12'Oz00                  zzzzzz000000
8'd256                      00000000
```

Integer vrednosti

- Validne vrednosti: 0-9
- Dvojični komplement 32-bitnog integer-a

- Sintaksa:

```
{-}{<digit>}
```

- Primeri:

```
  0    00..000000000
 -1    11..111111111
 35    00..000100011
256    00..100000000
```

Realne vrednosti

- Validne vrednosti: 0-9
- Dvostruka preciznost (64-bita) floating point

- Sintaksa:

```
[-]{<digit>}.{<digit>}[E[-]{<digit>}]
```

- Primeri:

```
 0.0
 1.0
-1.0
 1.0E-9
```

Logičke vrednosti

- Svaka vrednost različita od nule je TRUE
- Sve ostalo je FALSE
  - uključujući i ‘x’ i ‘z’

- Primeri:

```
1'b0        FALSE
1'b1        TRUE
1'bx        FALSE
3'b000      FALSE
3'b100      TRUE
3'b10x      TRUE
3'b00x      FALSE
```

Operatori sa bitovima

- ~<expr> Bitwise *not*
- <expr>&<expr> Bitwise *and*
- <expr> | <expr> Bitwise *or*
- <expr> ^ <expr> Bitwise *xor*
- <expr> ~^ <expr>
  <expr> ^~ <expr> Bitwise *xnor*

- Primeri:

```
~1'b0                    1'b1
~4'b01xz                 4'b10xx
3'b01x & 3'b111          3'b01x
3'b01x | 3'b000          3'b01x
```

Operatori redukcije

- &<expr> — *and*-redukcija
- |<expr> — *or*-redukcija
- ^<expr> — *xor*-redukcija

- Primeri:

```
&3'b001     1'b0
&3'b111     1'b1
&3'b11x     1'bx
&3'b1x0     1'b0
```

```
|3'b001     1'b1
|3'b000     1'b0
|3'b10x     1'b1
|3'b0x0     1'bx
```

```
^3'b001     1'b1
^3'b111     1'b1
^3'b11x     1'bx
```

Vektorski operatori

- <expr> << <expr> Logičko pomeranje ulevo
- <expr> >> <expr> Logičko pomeranje udesno
- {<expr>, <expr>} Konkatenacija
- {<const-expr>{<expr>}} Replikacija

- Primeri:

```
3'b101 << 1        3'b010
3'b101 >> 2        3'b001
{3'b10x, 1'b0}     4'b10x0
{3{2'bx1}}         6'bx1x1x1
```

Aritmetički operatori

- - <expr> negacija- dvojični komplement
- <expr> + <expr> sabiranje
- <expr> - <expr> oduzimanje
- <expr> * <expr> množenje
- <expr> / <expr> deljenje
- <expr> % <expr> moduo

- Primeri:

```
-3'b001          3'b111
3'b010*3'b010    3'b100
7%4              3
```

Relacioni operatori

- <expr> == <expr> jednako
- <expr> != <expr> različito
- <expr> === <expr> identično

  ‘x’ i ‘z’ bitovi moraju biti identični
- <expr> !== <expr> nije identično

- Primeri: (rezultat je 1 bit!!!)

```
3'b001 == 3'b010      1'b0   False
3'b001 != 3'b010      1'b1   True
3'b0x1 == 3'b0x1      1'bx   False
3'b0x1 != 3'b0x1      1'bx   False
3'b0x1 === 3'b0x1     1'b1   True
3'b0x1 !== 3'b0x1     1'b0   False
```

Relacioni operatori

- <expr> < <expr> manje od
- <expr> <= <expr> manje ili jednako
- <expr> > <expr> veće od
  <expr> >= <expr> veće ili jednako

- Primeri:

```
3'b000 < 3'b000      1'b1
3'b111 < 3'b001      1'b0
-1 < 1               1'b1
```

Logički operatori

- !<expr> logičko NOT
- <expr> && <expr> logičko AND
- <expr> || <expr> logičko OR

- Primeri:

```
!3'b001                 1'b0      F
!1'bx                   1'bx      F
3'b001 && 3'b100        1'b1      T   logičko I
3'b001 &  3'b100        3'b000    F   bitwise I
3'b10x && 3'bx01        1'b1      T
3'b10x &  3'bx01        3'bx0x    F
3'b000 || 3'b101        1'b1      T
3'b000 |  3'b101        3'b101    T
```

Kondicionalni (uslovni) operator

- <expr1> ? <expr2> : <expr3>
- Ako je *expr1* TRUE, onda je rezultat *expr2*
- Ako je *expr1* nepoznata, onda je rezultat X
- Inače, rezultat je *expr3*

- Primeri:

```
    1 ? 1 : 2          1
    0 ? 1 : 2          2
1'bx ? 1 : 2      32'bx
```

Prednost operatora

• Unarni	!~ -
• Množenje	*/%
• Sabiranje	+-
• Pomeranje	<< >>
• Odnosni	< <= > >=
• Jednakost	== != === !==
• Bitwise redukciono AND	&
• Bitwise redukciono XOR	^ ~^ ^~
• Bitwise redukciono OR	|
• Logičko AND	&&
• Logičko OR	||
• Kondicionalni	?:

Najviši

Najniži