Osnove Verilog HDL jezika

MODELIRANJE PONASANJA

MODELIRANJE PONASANJA

- Modeliranje ponasanja je najapstraktniji (I najmocniji) nacin za opis hardvera u Verilog-u.
- Najvise lici na klasicno programiranje ali postoje i znacajne razlike, jer se odredjeni

  verilog kod izvrsava drugacije nego sto bi se izvrsavao analogni kod u nekom proceduralnom jeziku.
- Na ovom nivou treba voditi racuna da dobijene konstrukcije ne budu previse komplikovane, jer se tada ne bi mogle sintetizovati u hardveru.

MODELIRANJE PONASANJA

PROCESI

MODELIRANJE PONASANJA

- Da bi protok vremena bio izrazen u nanosekundama, na pocetku fajla, pre definicija modula stavljamo:

  `timescale 1ns/1ns

MODELIRANJE PONASANJA

- Prvo vreme oznacava osnovnu vremensku jedinicu.
- Drugo vreme oznacava preciznost sa kojom se vreme meri (ako se stavi nesto manje, npr 1ps, tada je moguce zadavati i razlomljene delove nanosekunde u specifikacijama kasnjenja i slicnim situacijama, npr. #5.1).

MODELIRANJE PONASANJA

- Initial proces je proces koji se izvrsava samo jednom, pocev od nulte vremenske jedinice u radu simulatora (tj. odmah po pokretanju simulacije).
- Sintaksa ovog procesa je:

initial

<naredba>

MODELIRANJE PONASANJA

- Ako postoji vise od jedne naredbe (sto je tipican slucaj), tada se koriste begin i end:

  initial

  begin

  // naredbe

  end

MODELIRANJE PONASANJA

- Vreme potrebno za izvrsenje ovog procesa zavisi od naredbi
- ukoliko neke od njih sadrze definiciju kasnjenja, tada to produzava rad initial bloka.
- U suprotnom, isti se izvrsava u 0-toj vremenskoj jedinici. Na primer:

MODELIRANJE PONASANJA

```
initial
begin
x <= 0;
y <= 1;
end
```

- Ovaj proces se pokrece samo jednom (u 0-toj vremenskoj jedinici) i postavlja signale x i y (koji moraju biti registarskog tipa) na date vrednosti.

MODELIRANJE PONASANJA

```
initial
begin
x <= 0;
#20 y <= 1;
end
```

- Ovaj proces nakon sto upise 0 u x, mora da saceka 20 vremenskih jedinica, pa tek onda da u y upise 1.
- Ceo proces traje 20 vremenskih jedinica (od pocetka simulacije).

MODELIRANJE PONASANJA

Drugi tip procesa u verilog-u je always.

- Ovaj proces se, za razliku od initial procesa, iznova pokrece cim se zavrsi i tako dokle god traje simulacija.
- Sintaksa je ista kao kod initial procesa, samo sto se koristi kljucna rec always.

MODELIRANJE PONASANJA

Ako bi se ovaj proces izvrsio trenutno, tj. u jednoj vremenskoj jedinici, tada bismo ga odmah pokretali ponovo, sto bi dovelo do beskonacne petlje.

- Zbog toga se kod always procesa uvek na neki nacin kontrolise njegovo pokretanje.
- To se radi na dva nacina:

MODELIRANJE PONASANJA

dodavanjem kasnjenja u naredbe u procesu, npr:

```
initial
clk <= 0;
always
#20 clk <= ~clk;
```

MODELIRANJE PONASANJA

Prvi (initial) proces inicijalizuje (odmah) clk signal na 0.

- Drugi (always) proces se pokrece takodje odmah, ali pre izvrsenja naredbe mora da saceka 20 vremenskih jedinica, nakon cega se invertuje clk signal.
- Odmah zatim se ponovo pokrece always proces koji opet ceka jos 20 vremenskih jedinica pre nego sto invertuje clk signal i td

```
initial
clk <= 0;
always
#20 clk <= ~clk;
```

MODELIRANJE PONASANJA

Drugi nacin kontrolisanja pokretanja always procesa je specifikacijom dogadjaja koji aktivira proces, npr.

always @(p or q)

begin

p <= q;

q <= p;

end

MODELIRANJE PONASANJA

Ovaj proces se aktivira svaki put kada se promeni vrednost bilo p bilo q signala.

- Efekat procesa je zamena vrednosti p i q .

```
always @(p or q)
begin
p <= q;
q <= p;
end
```

MODELIRANJE PONASANJA

Slozene naredbe u okviru begin-end uvek se izvrsavaju sekvencijalno, tj. po redu kako su navedene.

- Alternativno, umesto begin-end moze se koristiti fork-join blok, u kome se sve naredbe izvrsavaju konkurentno.
- Samim tim specifikacija kasnjenja se u begin-end bloku uvek racuna od zavrsetka prethodne naredbe, dok se u fork-join bloku uvek racuna od pocetka izvrsavanja bloka.

MODELIRANJE PONASANJA

Konkurentnost se u verilog-u, naravno, ispoljava i u tome sto se naredbe iz razlicitih procesa konkurentno izvrsavaju.

MODELIRANJE PONASANJA

NAREDBA PROCEDURALNE DODELE

MODELIRANJE PONASANJA

Postoje dve vrste naredbe proceduralne dodele:

blokirajuce I neblokirajuce.

- Blokirajuce se izvrsavaju tako sto se izracuna izraz na desnoj strani, a zatim se azurira vrednost promenljive na levoj strani.
- Ovo ponasanje je najslicnije naredbi dodele u proceduralnim programskim jezicima.

MODELIRANJE PONASANJA

Sintaksa ove naredbe je:

<promenljiva> = <izraz>;

- Naziv blokirajuca potice od toga sto se naredbe koje slede u bloku naredbi iza te naredbe nece izvrsavati pre nego sto se azuriranje vrednosti leve strane ne zavrsi (upravo ovako radi i dodela u proceduralnim programskim jezicima).

MODELIRANJE PONASANJA

Na primer:

<initial

begin

p = q;

q = p;

end

- Ovde ce se vrednost q upisati u p, pa ce se tek onda zapoceti izvrsavanje druge dodele kojom se vrednost p upisuje u q.
- Efekat je da se u q upisuje nova vrednost podatka p, a to je upravo stara vrednost q.

MODELIRANJE PONASANJA

Neblokirajuca naredba proceduralne dodele ima sintaksu:

<promenljiva> <= <izraz>;

Ova naredba se izvrsava u dve etape:

- najpre se izracuna izraz na desnoj strani, ali se izracunata vrednost ne upisuje odmah u promenljivu na levoj strani.

MODELIRANJE PONASANJA

Umesto toga, vrednost izraza se zapamti, a nastavlja se dalje sa sledecom naredbom u bloku.

- Na kraju tekuce vremenske jedinice, kada se aktivni red isprazni, vrsi se azuriranje promenljive sa leve strane.
- Ova naredba ne blokira izvrsavanje narednih naredbi u bloku, jer se one izvrsavaju pre nego sto se izvrsi upis u promenljivu.

MODELIRANJE PONASANJA

Na primer:

```
initial
begin
p <= q;
q <= p;
end
```

MODELIRANJE PONASANJA

Sada se najpre izracuna desna strana prve naredbe (q), ali se ta vrednost ne upisuje u p.

- Zatim se izracuna desna strana druge naredbe (to je vrednost p pre promene, jer nova vrednost jos nije upisana) i ta vrednost se takodje zapamti interno za kasnije.

```
initial
begin
p <= q;
q <= p;
end
```

MODELIRANJE PONASANJA

Na kraju vremenske jedinice, kada se svi ostali dogadjaji obrade (ukljucujuci i one iz drugih procesa), vrsi se upisivanje zapamcenih vrednosti u p odnosno q.

- Efekat ce biti da ce q dobiti staru vrednost od p, a p ce dobiti staru vrednost od q, tj. imacemo zamenu vrednosti ovih promenljivih.

```
initial
begin
p <= q;
q <= p;
end
```

MODELIRANJE PONASANJA

Promenljiva na levoj strani proceduralne dodele (i blokirajuce i neblokirajuce) mora biti registarskog tipa, za razliku od ranije uvedene naredbe kontinuirane dodele kod koje promenljiva na levoj strani mora biti zicanog (wire) tipa.

MODELIRANJE PONASANJA

Dobra preporuka je da se blokirajuce dodele koriste pri dizajnu kombinatornih kola (jer se izlazi nekih kola moraju najpre izracunati pre nego sto se dovedu na ulaze narednih kola u lancu), dok se pri dizajnu sekvencijalnih kola koriste ne-blokirajuce dodele (jer se tipicno na uzlaznoj putanji sata uzimaju zatecene vrednosti i one se koriste za izracunavanje novih vrednosti, tj. novog stanja).

MODELIRANJE PONASANJA

Preporuka je da se u istom bloku naredbi ne mesaju

- blokirajuce I neblokirajuce naredbe dodele , iako sam

  programski jezik to ne zabranjuje.
- ovakva praksa obicno dovodi do loseg dizajna kola.

MODELIRANJE PONASANJA

Proceduralna dodela (always, initial)

```
reg a;
always @(posedge clk)
    a <= b;
```

leva strana (a) mora biti reg može se koristiti:

- = (blokirajuća)
- <= (neblokirajuća)

Zato se kaže: „promenljiva mora biti registarskog tipa“

MODELIRANJE PONASANJA

Kontinuirana dodela (assign)

wire y;

assign y = a & b;

leva strana (y) mora biti wire

dodela je stalna (uvek aktivna)

Zato se kaže: „promenljiva mora biti wire tipa“

MODELIRANJE PONASANJA

Najvažnija razlika:

Proceduralna (always),sekvencijalna logika,reg

Kontinuirana (assign),kombinaciona logika, wire