Osnove Verilog HDL jezika

VREMENSKA KONTROLA NAREDBI I PROCESA

VREMENSKA KONTROLA NAREDBI I PROCESA

Prvi tip vremenske kontrole je oblika:

#<broj>

koji odlaze izvrsavanje naredbe ili procesa za <broj> vremenskih jedinica u simulatoru. Na primer:

```
initial #20
begin
#10 x <= y;
end
```

VREMENSKA KONTROLA NAREDBI I PROCESA

- zapocinje proces tek u 20-toj vremenskoj jedinici od pocetka simulacije.
- Nakon sto proces zapocne, ceka se jos 10 vremenskih jedinica (sto je ukupno 30 jedinica) da bi se izvrsila naredba dodele.

VREMENSKA KONTROLA NAREDBI I PROCESA

Drugi tip vremenske kontrole je zasnovan na promeni vrednosti signala po zelji. Na primer:

```
always @(posedge clk)
begin
  q <= ~q;
end
```

VREMENSKA KONTROLA NAREDBI I PROCESA

Ovaj proces simulira T flip flop koji reaguje na prelazak signala clk sa 0 na 1. Slicno, naredni proces ima identicni efekat:

```
always
begin
 @(posedge clk) q <= ~q;
end
```

VREMENSKA KONTROLA NAREDBI I PROCESA

Ovaj proces simulira T flip flop koji reaguje na prelazak signala clk sa 0 na 1. Slicno, naredni proces ima identicni efekat:

```
always
begin
 @(posedge clk) q <= ~q;
end
```

VREMENSKA KONTROLA NAREDBI I PROCESA

Treci tip vremenske kontrole je wait() naredba. Njena sintaksa je:

wait(<uslov>) <naredba_ili_blok>

Uslov moze biti bilo koji logicki izraz. Izvrsavanje naredbe se odlaze dok se ne ispuni uslov. Na primer:

VREMENSKA KONTROLA NAREDBI I PROCESA

```
integer x;

initial
begin
x <= 0;
wait(clk) x <= x + 1;
end
```

VREMENSKA KONTROLA NAREDBI I PROCESA

- Najpre se x postavlja na nulu, a onda se ceka da se signal sata postavi na 1 (ukoliko vec nije jednak 1).
- Tada se izvrsava naredba uvecanja vrednosti.
- Dakle, @() sintaksa nam omogucava da registrujemo promenu signala, dok wait() omogucava da cekamo odredjenu vrednost ili ispunjenje odredjenog uslova (koji je mozda vec ispunjen, u tom slucaju ne cekamo nista, vec odmah izvrsavamo naredbu).

VREMENSKA KONTROLA NAREDBI I PROCESA

Posmatrajmo i sledeci blok:

```
integer x;
reg clk;
initial
begin
 x <= 0;
 clk <= 0;
end
always #10
 clk <= ~clk;
always wait(clk) x <= x + 1;
```

VREMENSKA KONTROLA NAREDBI I PROCESA

- Ovde imamo problem sa beskonacnom petljom.
- Naime, signal kloka se menja svakih 10 vremenskih jedinica.
- Kada se jednom postavi na 1,tada je uslov wait-a ispunjen i poslednji always proces je, dokle god vazi clk == 1 (a to je narednih 10 vremenskih jedinica) ekvivalentan sa:

```
always
x <= x + 1;
```

VREMENSKA KONTROLA NAREDBI I PROCESA

Sada se kada clk postane jedan ovaj blok svodi na:

always #1 x <= x + 1;

sto za efekat ima da se na svakih #1 vremenskih jedinica uvecava vrednost x za 1.

KONTROLA VREMENA UNUTAR DODELE

Vremenska kontrola se moze zadati i unutar naredbe dodele:

x <= #10 y;

Efekat ovoga je da se desna strana izracunava odmah, ali se upis u levu stranu odlaze za 10 vremenskih jedinica.

KONTROLA VREMENA UNUTAR DODELE

Suprotno tome,

#10 x < = y;

odlaze kompletnu naredbu, tj. i izracunavanje desne strane.

KONTROLA VREMENA UNUTAR DODELE

Slicno,
ako imamo:

x <= #10 y;
y <= z;

obe desne strane ce se izracunati odmah, zatim ce se odmah upisati
z u y, dok ce se upis u x odloziti za 10 vremenskih jedinica.

KONTROLA VREMENA UNUTAR DODELE

Suprotno tome,

```
#10 x <= y;
y <= z;
```

KONTROLA VREMENA UNUTAR DODELE

odlaze obe naredbe kompletno, dok kod:

```
x <= y;
#10 y <= z;
```

odlaze drugu naredbu kompletno, dok se prva izracunava u potpunosti
u tekucoj vremenskoj jedinici.