Predmet PNRS2 – UVM Lista tema koje dolaze ne teoretsku proveru znanja

Finalna verzija.

HDL teme za proveru znanja:

Uvođenje HDL jezika VHDL, Verilog, svrha, značaj.

1. Modelovanje sekvencijalnih procesa
2. Modelovanje kombinacionih procesa
    a. Problem – nepotpuna lista osetljivosti,
    b. Lečevi
    c. Multiple driver
3. Tipične kombinacione mreže:
    a. Mux
    b. Demux
    c. Koder
    d. Decoder
    e. Prioritetni koder
4. Sekvencijalne mreže
    a. Brojač
5. Milijev automat
6. HDL snippeti koji pokrivaju date teme

SV teme za proveru znanja:

7. 4-value logic
8. Definisanje vremenskih okvira simulacije - direktiva `timescale 1ns/100ps
   razlikovati time unit, time step – rezolucija, korišćenje istih. #1 #1step #1us
9. Initial i always blokovi, vremenska kontrola istih.
10. Razumevanje redosleda izvršavanja paralelnih procesa/threadova – snippeti te vrste.
11. Blokirajuća i neblokirajuća dodela vrednosti.
12. Instanciranje i konektovanje modula

a. Poziciono
b. Po imenu
c. Implicitno po imenu
d. Implicitno .*

13. Fork – join mehanizam
14. SV scheduler / raspoređivač

e. Koncept vremena u SV raspoređivaču – dve ose
f. Redosled aktivnosti po kojima scheduler radi

15. Taskovi I funkcije

g. Kada se koriste taskovi, a kada funkcije – osnovna razlika
h. Kako se pozivaju, kakvi mogu biti argumenti

16. Interfejsi
i. Šta oni kapsuliraju, šta predstavljaju
j. Kada se koriste
k. Prednosti
l. Modport unutar intefejsa
17. Race condition – šta je i kada predstavlja problem?
m. Race na liniji DUT-TB
(primer bloking dodela signalu modelovana u DUT-u na rastuću ivicu takta, u TB-u se isti signal čita u bloking režimu na istu ivicu takta)
n. Clocking blok kao rešenje
18. Clocking blok
o. Šta on sistemski realizuje
p. Kako se implementira
19. Nizovi i Queue (redovi izvršenja)
q. Značaj Queue kao naprednije objektne tehnike u odnosu na nizove, zašto nam je to potrebno? Zbog transakcija kao apstraktnih objekata koje ćemo prometati kroz Queue.

20. OOP
a. Značaj uvođenja OOP koncepata u verifikaciju HW
b. Veza između klase koja predstavlja transakciju i signalnog telegrama (frame-a) nekog komunikacionog protokola tipa Ethernet.
c. Linija razdvajanja / spajanja OOP sveta I siglanog time-accurate sveta.
d. Osnovni OOP koncepti: klase, njena polja I metode, referenciranje na sopstvena polja i metode
e. Nasleđivanje,
f. Enkapsulacija local / protected
g. Apstraktna / konkretna klasa
h. Parametrizovana klasa
i. Polimorfizam
21. Randomizacija
a. Cilj randomizacije
b. Ograničavanje randomizacije
c. Randomizacija podataka naspram randomizacije komandi
d. Kako se deklarišu slučajne varijable
e. Nasleđivanje ograničenja
f. dist, implikacija u okviru ograničavanja
g. In-line ograničenja,
h. Rand_mode() metode – task i funkcija istog imena
i. Constraint_mode – task i funkcija istog imena
j. Random stability – šta predstavlja i koji mu je značaj.

22. Pokrivenost u verifikaciji
a. Koje vrste pokrivenosti razlikujemo
b. Pokrivenost HDL koda
c. Pokrivenost podataka
d. Pokrivenost komandi odnosno testnih slučajeva
e. Funkcionalna pokrivenost
    i. Hijerarhija: Covergroup / coverpoint / bins
f. Automatske korpe/bins
g. Eksplicitne korpe/bins
h. Umeti tumačiti covergroup primere sa automatskim I eksplicitnim korpama/bins koje uključuju međupokrivenost (cross coverage) između coverpoints i/ili varijabli
i. Korpe/bins za praćenje tranzicija (sekvence tranzicija)
23. Statička verifikacija
a. Šta proveravaju statički alati
    i. Naming konvencije
    ii. Lečevi
    iii. Liste osetljivosti
    iv. Nesintetizibilne konstrukcije
    v. Klok domen krosing
    vi. Kombinacione petlje
    vii. Multiple drivers
    viii. Mrtav kod
b. Automatizovane formalne provere
    i. Logic Equivalence Checking
    ii. Static Timing Analysis
24. Problem potrošnje energije
a. Statička I dinamička potrošnja CMOS-a sa naptetkom tehnologije.
25. Events / događaji
a. Deklarisanje tipa podatka event
b. Okidanje događaja sa -> blokirajuće (opisati simulacioni mehanizam)
c. Odnosno sa ->> ne blokirajuće (opisati simulacioni mehanizam)
d. Čekanje na događaj if(posedge clk)
e. Čekanje na događaj @(posedge clk)
f. Čekanje na događaj wait(event_identifier.triggered)
26. Snippeti koji pokrivaju coverage + eventove.

---

27. Assertion-i
    a. Svrha, značaj, kada i gde se postavljaju.

b. Pokrivaju boolean/kombinacionu i sekvencijalnu logiku.
c. Trenutni (immediate) assertion
    i. Kako se postavljaju, kako funkcionišu
d. Konkurentni (concurrent) assertion
    i. Kako se postavljaju, kako funkcionišu (oni mogu inkorporirati temporal logiku)
e. Sekvence unutar assertiona
f. Specificiranje kašnjenja unutar sekvenci ##broj
g. Logički uslovi sa sekvencama I, ILI,
h. Preklapanje sekvenci intersect, within, throughout.
i. Mehanizam za "reset" sekvence disable iff
j. Hijerarhija boolean izraz/sequence/property
k. Implikacija u istom ciklusu
l. Implikacija u sledećem ciklusu

28. Assume
29. UVM osnovna pitanja i tipični blokovi
    a. Šta nam UVM nudi u odnosu na čist SV, bez UVM biblioteka
    b. Koji blokovi čine osnovni koncept – osnovni šablom UVM-a
    c. UVM testbench – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski
    d. UVM test – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski u TB-u, šta sve tipično sadrži.
    e. UVM Environment – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski u TB-u, šta sve tipično sadrži.
    f. UVM Scoreboard – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski, šta sve tipično sadrži.
    g. UVM Agent– čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski, šta sve tipično sadrži.
    h. UVM Sequencer – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski u TB-u.
    i. UVM Sequence – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski u TB-u.
    j. UVM Driver – čemu služi, kada se instancira, kada se izvršava, gde se nalazi hijerarhijski u TB-u.
    k. UVM Monitor u TB-u.
    l. Zašto se insistira na UVM šablonu
30. UVM simulacione faze
    a. build_phase(), čemu služi, odakle se nasleđuje, da li troši simulaciono vreme
    b. connect_phase(),čemu služi, odakle se nasleđuje, da li troši simulaciono vreme
    c. run_phase(),čemu služi, odakle se nasleđuje, da li troši simulaciono vreme
31. UVM transaction
    a. Šta sve obuhvata UVM transakcija

b. Putem kojih interfejsa možemo da prenosimo transakcije iz jedne UVM komponente u drugu UVM komponentu
c. TLM konekcija port-export (producer kao inicijator poziva task put())
d. TLM konekcija port-export (consumer kao inicijator poziva task get())
e. Problem komunikacije putem pozivanja blokirajućih taskova u okviru TLM konekcije
f. uvm_tlm_fifo kao elastična veza između producer-a i consumer-a
g. Kolika je dubina uvm_tlm_fifo sprege

h. put() metoda za postavljanje transakcije na port. Šta zahteva kao argument, ko je poziva, da li je blokirajuća.
i. try_put() metoda za postavljanje transakcije na port. Šta zahteva kao argument, ko je poziva, da li je blokirajuća.
j. get() metoda za preuzimanje transakcije sa porta. Šta zahteva kao argument, ko je poziva, da li je blokirajuća.
k. try_get() metoda za preuzimanje transakcije sa porta. Šta zahteva kao argument, ko je poziva, da li je blokirajuća.

32. Analysys port
    a. Objasniti namenu analysys_port-a u UVM-u
    b. Objasniti šta se događa kada komponenta pozove analysys_port.write() metodu.
    c. Da li metoda analysys_port.write() troši simulaciono vreme.
    d. Šta predstavlja uvm_tlm_analysys_fifo ?
33. Sequence
    a. Iz koje klase je izvedena uvm_sequence_item klasa
    b. uvm_sequence_item klasa je proširene u odnosno na svoju izvornu klasu između ostalog sa identifikacijom sekvence . Zbog čega je to urađeno, šta je time dobijeno?
    c. Iz izvorne klasa uvm_sequence se nasleđuju se objekti tipa sequence. Objasniti razliku između sequence objekta i sequence_item objekta.
34. Sequencer
    a. Objekat sequencer se nasleđuje iz klase uvm_sequencer. Objasniti njegovu ulogu.
    b. Veza između driver-a i sequencer-a se ostvaruje povezivanjem seq_item_port-a I seq_item_export-a. Objasniti ovaj mehanizam. Ko generiše zahtev za podacima, kog tipa su podaci koji se prenose. Koji blok predstavlja izvor podataka.
    c. Objasniti razliku između sequencer-a i sequence.
    d. Tipično u okviru driver-a se u petlji postavlja sledeći zahtev za novim podacima koji su mu potrebni:
       seq_item_port.get_next_item(T);
       Kog tipa je argument T?
       Da li je ova metoda blokirajuća?

e. U okviru driver-a moguće je zahtevati nove podatke sa seq_item_port-a pozivanjem sledeće metode
   seq_item_port.try_next_item(T);
   Kog tipa je argument T?
   Da li je ova metoda blokirajuća?
f. U okviru driver-a moguće je podatke zahtevati putem donjih metoda
   seq_item_port.get_next_item(T);
   seq_item_port.try_next_item(T);
   Objasniti razliku između njih.

35. Monitor
    a. Objekat monitor se nasleđuje iz klase uvm_monitor. Objasniti njegovu ulogu.
    b. Kako se realizuje komunikacija monitora sa ostalim UVM blokovima višeg nivoa za praćenje i proveru transakcija?
    c. Na koje druge UVM objekte je monitor tipično povezan?
36. Instanciranje komponenti
    a. Kako možemo instancirati komponente (objekte) u UVM-u.
    b. Koji metod instanciranja objekata u UVM je preporučljiv.
    c. Šta podrazumeva koncept prepisivanja (override) komponenti/objekata u UVM-u?
37. Agenti
    a. Objekat agent se nasleđuje iz klase uvm_agent. Objasniti njegovu ulogu.
    b. Koje UVM komponente tipično instancira jedan UVM agent.
    c. U kojim modovima (režimima) može da radi agent.
38. Environment
    a. Objekat env se nasleđuje iz klase uvm_env. Objasniti njegovu ulogu.
    b. Šta predstavlja environment iz perspektive UVM-a?
    c. Gde se hijerarhijski nalazi environment u okviru tipičnog UVM verifikacionog projekta?
39. Sequence
    a. Objekat sequence se nasleđuje iz klase uvm_sequence. Objasniti njegovu ulogu.
    b. Šta nam omogućava objekat sequence?