2) Synchronní a asynchronní simulace

Synchronní – zpoždění se neuvažuje
- metoda pevného časového kroku

Asynchronní
- zpoždění se uvažuje:
  - jednotkové - stejné pro všechny obvody
  - násobné - násobek jednotkového zpoždění
  - libovolné
- podrobnější přístup umožňuje:
  - zjistit statické hazardy a dynamické hazardy
  - ověřit dodržení předstihů a přesahů
  - ověřit správnost frekvence hodinových pulsů
- metoda proměnného časového kroku

13) Asynchronní simulace: stručná charakteristika a možnosti implementace.
- zpoždění se uvažuje: jednotkové - stejné pro všechny obvody, násobné - násobek jednotkového zpoždění, libovolné
- podrobnější přístup umožňuje:
  * zjistit statické hazardy a dynamické hazardy
  * ověřit dodržení předstihů a přesahů
  * ověřit správnost frekvence hodinových pulsů
- metoda proměnného časového kroku
- použijeme metodu aktivní cesty neplánujeme soustavně všechny logické členy do seznamu událostí, ale pouze "kandidáty na změnu" (t. j. členy u nichž došlo ke změně hodnoty některého ze vstupních signálů)
- provedení veškerých změn v modelu podmíníme existencí příslušného záznamu v seznamu událostí seznamu událostí
- důsledně oddělíme výpočty dílčích členů a uložení vypočtených hodnot do výstupních signálů

14) Synchronní simulace. Stručná charakteristika a možnosti implementace
- zpoždění se neuvažuje, metoda pevného časového kroku, nevystihuje paralelní šíření signálu

4) Dva typy sil, F (force) a R (resistive) a 4-hodnotový systém 0,1,X,Z rezoluční tabulky pro 2 vodice

X – neznámá hodnota (0 nebo 1), Z – vysoká impedance

	0	1	X	Z
0	0	X	X	0
1	X	1	X	1
X	X	X	X	X
Z	0	1	X	Z

Dva typy sil: - 9-i stavový systém – Z0,Z1.ZX,R0,R1,RX,F0,F1,FX. Příklad: MOS investor, otevřený kolektor: VCC ---R-→ R1, GND --→ F0

16) Možné přístupy k organizaci výpočtu dílčích modelů při simulaci log. obvodu

4-hodnotový systém: 0,1,X,Z
9-hodnotová logika: 0,1,X - sily Z(weak), R(resistive), F(force)
12-stavový systém: 0,1,X - síly Z,R,F,U(unknown)

20) Napiste tabulku vystupu resolucni funkce pro kombinace F0 F1 FX Z RX R1 R0 s RX (R - resistive, F - forcing)
- viz. tabulka [uvod1.pdf 11]

12) F0...RX (9 hodnotová logika) rezoluční funkce s R0

Dva typy sil: - 9-i stavový systém – Z0,Z1.ZX,R0,R1,RX,F0,F1,FX. Příklad: MOS investor, otevřený kolektor: VCC ---R-→ R1, GND --→ F0

1) Reálný objekt, systém, model. souvislosti. pokuste se o formální definici systému
- reálny objekt: posuzovaná skutečnost
- model(systém): je vytvořen z reál. objektu přes filtr rozlišovací úrovně, vymezuje část objektu a vnitřní organizaci
- simulátor: provádí výpočet modelu
- modelování:platné zjednodušení z daného hlediska (zachová důležité prvky)
- simulace: numerické řešení modelu

3) Model, analytický model, simulační model

Model je popis reálného objektu.
Modely se dělí na fyzické (těmi se nezabýváme) a matematické, ty dále na
analytické - nositel teorie, ucelený pohled na reálný objekt.
simulační (numerické) - jde o metodu řešení problémů, nástroj hrubé síly

10) Model, strukt.model...

Model: - nejširší pojetí: metodologický nástroj; př: dílenský výkres, schéma obvodu, mapa terénu, atd;;; - užší pojetí: striktně matematicky specifikovaný systém na určité úrovni abstrakce
- model(systém): je vytvořen z reál. objektu přes filtr rozlišovací úrovně, vymezuje část objektu a vnitřní organizaci

7) Vysvětlete pojmy model, modelováni a simulace, funkce modelů, typy modelů.
- Modelování - činnost sloužící k získávání poznatků o reálném objektu. Jde o platné zjednodušování z hlediska daného záměru; platné = zachování podstatných rysů.
- Model: - nejširší pojetí: metodologický nástroj; př: dílenský výkres, schéma obvodu, mapa terénu, atd.
  - užší pojetí: striktně matematicky specifikovaný systém na určité úrovni abstrakce
- Modely - fyzikální (malá přehrada, čelní náraz auta)
  - matematické: analytické, simulační - numerické
  dle interakce s okolím: otevřené, uzavřené
  dle vnitřní paměti: statické, dynamické
  dle chování: spojité, diskrétní, kombinované, stochastické
- Funkce modelů:
  * nástroj myšlení (př. konečný automat - minimalizace vnitřních stavů)
  * prostředek ke komunikaci : odborné konference, časopisy
  * nástroj výuky: trenažéry, prostředek předvídání a experimentování:
- Simulace - numerické řešení matematických modelů

17) Co je to reálný čas, modelový čas, strojový čas

hodnoty reálného času mapujeme na hodnoty z množiny čísel (real nebo integer) => modelový čas
originál: reálný čas
model: modelový čas
výpočetní prostředek: strojový čas

5) Události, proces, reaktivační bod, fáze procesu

událost - transakce, při níž dochází ke změně hodnoty signálu.
proces - paralelní příkaz, jehož výsledný efekt je definován pomocí sekvenčního algoritmu (dílčí sekvenční část simulačního programu).
reaktivační bod - bod, který definuje místo příští aktivace procesu nebo procedury.
fáze procesu - části procesu oddělené od sebe příkazem wait

11) Pojmy proces, spojitý proces, diskrétní proces
- událost - změna hodnoty atributu
- proces - posloupnost událostí v čase. Popis procesu = zobrazení F z časové množiny do množiny atributů systému.
- spojitý proces - F je určena formou soustavy diferenciálních rovnic
- diskrétní proces - F je určena formou posloupnosti událostí

9) Seznam událostí – event list

položky (záznamy událostí, event notices)
- čas události (KDY?)
- specifikace události - odkaz na objekt (CO?)

struktura:
- sekvenční, nesetříděný - nesnadné vkládání i výběr
- sekvenční, setříděný (dle hodnot model. času) - výběr od začátku
- jednocestně zřetězený, setříděný
- dvoucestně zřetězený, setříděný
- hierarchický apod.
- viz. obrazek [uvod1.pdf 24]

15) Modelovaní struktur číslicových obvodu: možné strategie
- repetiční vyhodnocení neuspořádaných prvků (správná odezva: max. n průchodů)
- vyhodnocení uspořádaných prvků ve směru toku signálu (správná odezva: 1 průchod)

6) Setrvačné zpožděni - co, princip, nástin implementace
- zpoždění pro logické členy. Impuls se neprojeví, protože při plánování události se zruší událost naplánovaná na dřívější dobu.
- nepropouští užší pulzy, než je vlastní zpoždění.
- model viz. Slide – použití pomocného objektu DELAY a podprogramů, - uvažujeme pouze 2-hodnotovou logiku :

1) Start
2) Generuj objekt DELAY
3) Čekej na změnu vstupu a
4) DELAY.pom == a ? Pokud ANO, jdi na 3 . Pokud NE, jdi na 5
5) DELAY.pom := a
6) je DELAY naplánován? Pokud ANO, jdi na 7, jinak jdi na 8.
7) vymaž DELAY z plánu. Jdi na 3)
8) Naplánuj metodu „ulož“ objektu DELAY, na dobu T. Jdi na 3)

- objekt DELAY – obs. proměnnou – slouží jako dočasná paměť výstupní hodnoty

8) Dopravní zpožděni, popis, naznak implementace
- propouští všechny pulsy, vhodné pro simulaci spojů (vodičů) , Y <= transport X after

- předpoklad: existuje jádro pro časovou synchronizaci dílčích elementů simulované struktury
- požití pomocných dynamických objektů DELAY a podprogramů
- objekt DELAY – obs. proměnnou – slouží jako dočasná paměť výstupní hodnoty
1) Čekej na změnu vstupu a
2) Generuj nový objekt DELAY, DELAY.pom := a.
3) Naplánuj metodu „ulož“ objektu DELAY na dobu T. Jdi na 1.

1) Proměnné ve VHDL , použití, kde se deklaruji
- objekt získaný deklarací (pouze v sekvenčním prostředí!) klausulí variable.
- disponuje pouze svou současnou hodnotou
- přiřazení hodnoty – okamžitě při provedení příkazu

2) Rozdělení signálů
- "obyčejný" signál - každý objekt třídy signal, který je nosičem datového typu nedisponujícího žádnou rezoluční fcí. Může být buzen pouze jedním budičem.
- rozhodovaný (resolved) signál - objekt třídy signal, který je nosičem datového typu nesponujícího žádnou rezoluční fcí. Tento typ připouští neomezený počet budičů, které však nelze od daného signálu odpojovat.
  - stražený signál - rozhodovaný signál specifikovaný klíčovým slovem register nebo bus. Budiče lze odpojovat. Odpojení dosáhneme v sekvenčním prostředí tím, že signálu přiřadíme hodnotu null. Typ registr si pamatuje poslední hodnotu i po odpojení budičů, typ bus ji ztrácí.

27) Signály ve VHDL: důvod existence, výskyt deklarace
- reprezentují vodiče modelovaného obvodu a jsou jediným prostředkem pro komunikaci v paralelním prostředí
- výskyt deklarace: pouze paralelní prostředí entity, architektury nebo knihovna.
  - - rozdělení: obyčejný, rozhodovaný, strážený

10) Rozhodované signály, k čemu to je, kde se deklaruji
- rozhodovaný (resolved) signál - objekt třídy signal, který je nosičem datového typu disponujícího žádnou rezoluční fcí. Tento typ připouští neomezený počet budičů, které však nelze od daného signálu odpojovat.
- strážený signál - rozhodovaný signál specifikovaný klíčovým slovem register nebo bus. Budiče lze odpojovat. Odpojení dosáhneme v sekvenčním prostředí tím, že signálu přiřadíme hodnotu null. Typ registr si pamatuje poslední hodnotu i po odpojení budičů, typ bus ji ztrácí.
- deklarace:

```
subtype mont_soucet is wired_or bit; --podtyp datového typu bit, který je vázaný na rezoluční fci
signal S1: mont_soucet; --rozhodovaný signál
signal S2: mont_soucet [register | bus];
```

17) Rozdíl mezi proměnnými a signály, zejména přiřazování v sériovém a paralelním prostředí
- hodnoty můžou být proměnným nebo signálům přiřazeny v procesu, tj. v sekvenčním prostředí : ->
proměnné - objekt získaný deklarací (pouze v sekvenčním prostředí!) klausulí variable.
  - disponuje pouze svou současnou hodnotou
  - přiřazení hodnoty – okamžitě při provedení příkazu

signály - hodnota se přiřadí po provedení všech fází naplánovaných procesů (po uplynutí delta zpoždění). Důvodem je zabránění nedeterministického chování v případě kvaziparalelního provádění více procesů (nesmí záležet na pořadí jejich provádění).
- paralelní prostředí – data flow příkazy - není-li ve výrazu explicitně uvedeno nenulové zpoždění pak skutečné přiřazení hodnoty danému signálu nastane v simulačním cyklu, který následuje za cyklem v němž byl proveden přiřazovací příkaz.
- Objekty: - constant - nemění hodnoty (pro jednoráz. přiřazení :=), výskyt: sekvenční i paralelní prostředí
  - variable - mění hodnoty (pro přiřazení :=), výskyt: sekvenční prostředí
  - signal - mění hodnoty (pro přiřazení <=), výskyt: sekvenční i paralelní prostředí

33) Vyjmenujte všechny případy, kdy dojde k tomu, ze jeden signál budí víc budičů
- k tomu že je signál buzen více budiči může dojít pouze u rozhodovaného signálu (a tím pádem i u stráženého).
- "obyčejný" signál - každý objekt třídy signal, který je nosičem datového typu nedisponujícího žádnou rezoluční fcí. Může být buzen pouze jedním budičem.
- rozhodovaný (resolved) signál - objekt třídy signal, který je nosičem datového typu disponujícího žádnou rezoluční fcí. Tento typ připouští neomezený počet budičů, které však nelze od daného signálu odpojovat.
- strážený signál - rozhodovaný signál specifikovaný klíčovým slovem register nebo bus. Budiče lze odpojovat. Odpojení dosáhneme v sekvenčním prostředí tím, že signálu přiřadíme hodnotu null. Typ registr si pamatuje poslední hodnotu i po odpojení budičů, typ bus ji ztrácí.

22) Strážené signály ve VHDL: charakterizujte jejich odlišnosti, typy, buzeni.
- stražený (guarded) signál - připouští více budičů, které lze dynamicky jednotlivě odpojovat a připojovat
- stražený signál typu register - po případném odpojení všech budičů signál pamatuje svou poslední hodnotu (v takovém případě příslušná resoluční funkce není vyhodnocována)
- stražený signál typu bus - resoluční funkce je vyhodnocována vždy a musí být navržena i pro případ odpojení všech budičů

Buzení stražených signálů: pouze formou stražených signálových přiřazovacích příkazů ve straženém bloku
Odpojení budičů od stražených signálů: nastane v okamžiku, kdy implicitní signál guard nabude hodnoty false (budič nabývá hodnoty tzv. prázdné transakce null).
Deklarace:

```
subtype mont_soucet is wired_or bit; –jde o podtyp datového typu bit, který je vázaný na resoluční funkci
signal S1: mont_soucet;                -- rozhodovaný signál, může být buzen více vodiči
signal S2: mont_soucet register;       -- stražený signál, typu register: může být odpojen od zdroje
signal S3: mont_soucet bus;            --stražený sig. typu bus, může být odpojen od budiče
```

19) Strážené bloky ve VHDL - ucel, specifické znaky
Stráženy blok (guarded block)
- blok doplněný ve své hlavičce o tzv. stražený výraz (guarded expression)
- ve straženém bloku je implicitně deklarován signál "guard", jehož hodnota automaticky sleduje v průběhu simulace hodnotu straženého výrazu
- signál "guard" nelze explicitně budit žádným budičem uvnitř straženého bloku a ani jej nelze připojovat k portům módu in, inout, buffer
- ve straženém bloku lze použít signál guard k podmíněnému provedení tzv. stražených signálových příkazů (označených symbolem guarded); toto provedení nastane:
  - při změně hodnoty straženého výrazu z hodnoty false na hodnotu true
  - v případě, že stražený výraz má hodnotu true a nastala událost na některém signálu vyskytujícím se na pravé straně straženého příkazu

21) Charakterizujte atributy signalu ve VHDL.
Atributy – funkce, které poskytují informace o událostech na signálu
s'active - true pokud je v daném simulačním cyklu signál s aktivní
s'last_active – vrací časový interval, který uplynul od poslední transakce na signálu s
s'event – true pokud nastala v simulačním cyklu změna na signálu
s'last_value – vrací hodnotu signálu před poslední událostí
- atributy signálu: druhotně odvozené od událostí či transakcí zvolených signálů; jsou použitelné opět jako signály (s´stable, s'delayed)
s'delayed – vytvoří nový signál stejného typu, ale zpožděný o delta zpoždění
s'stable – nový signál typu boolean, který má hodnotu true, pokud v daném simulačním cyklu nenastala na signálu žádná událost

7) Porty - jaké jsou, použiti v sekv. a paralel. prostředí
Definice vstupně a výstupních portů jsou možné pouze v entitě. Lze je použít jako lokální signály v architektuře.(in,out,inout,buffer).
- chovají se jako signály
- slouží pro přístup k entitě zvenčí
- in: vstupní – lze jen zapisovat
- out: výstupní – lze jen číst
- buffer: bufferované – lze číst i zapisovat
- inout: obousměrný – lze číst i zapisovat

15) Architektura ve VHDL
Komponenta se skládá z entity a architektury.
Entita specifikuje vnější vstupní a výstupní porty. Tyto porty jsou jediným prostředkem pro komunikaci modelu s jeho okolím.
Architektura definuje chování modelu nebo jeho strukturní implementaci s použitím dalších komponent z nichž každý je definován zase dvojicí entita+architektura. Pro jednu entitu můžeme definovat víc architektur
Prostředí architektury:
a) paralelní
   1. data flow popis – chování komponenty formou paralelních příkazů
   2. strukturní popis – hierarchie dílčích komponent

sekvenční – procedurální popis chování komponenty pomocí procesu, procedury nebo funkce

16) Komponenty ve VHDL - co to je, syntax, použití
- viz. předchozí – 15

20) Stručně popište VHDL z hlediska osazovaní architektur
- porty – specifikovány v entitě
- přiřazení entity a architektury dílčí komponentě: lze v architektuře vynechat v případě, že:
  komponenta a entita mají stejná rozhraní (tj. jména & porty) - default configuration
  osazení bude provedeno později (configuration)
- ve VHDL-93: přímá instalace komponent

- mapování portů entity na porty komponenty: při shodě jmen odpovídajících si portů entity a komponenty lze vynechat, různá jména nutno mapovat explicitně

18) Entity ve VHDL
- samostatně překládaná programová jednotka, která definuje rozhraní navrhované části. Pro specifikaci struktury je nutná přidružená architektura.
- specifikuje především vnější vstupní a výstupní porty (případně i parametry) modelu; tyto porty jsou jediným prostředkem pro komunikaci modelu s jeho okolím.

11) Třída Configuration ve VHDL
4) Význam následujících ve VHDL: entity, architecture, package, package body a configuration
- entity - samostatně překládaná programová jednotka, která definuje rozhraní navrhované části. Pro specifikaci struktury je nutná přidružená architektura. Specifikuje především vnější vstupní a výstupní porty (případně i parametry) modelu; tyto porty jsou jediným prostředkem pro komunikaci modelu s jeho okolím.
- architecture - Samostatně překládaná programová jednotka, která definuje interní strukturu a chování modelu.Definuje hodnoty výstupů jako reakce na vstupy.Deklarace: architecture A of E is begin end A;
- package –
- package body –
- configuration – přiřazení entity a architektury dílčí komponentě: lze v architektuře vynechat v případě, že: komponenta a entita mají stejná rozhraní (tj. jména & porty) - default configuration; osazení bude provedeno později (configuration)
- konfigurace vně architektury (**configuration declaration**) – samostatná část zdrojového programu. Umožňuje odložit osazení soklů a toto soustředit do jednoho místa => možnost rychlé změny integrovaných obvodů bez nutnosti nového překladu osazované strukturní architektury
5) Vývojový diagram simulačního cyklu VHDL
6) Nástroje VHDL pro strukturní simulaci
deklarace komponent:
syntax: component <jméno typu komponenty>
generic...; -- formální parametry
port (...); -- formální porty
end component;
specifikace komponent:
syntax: for <label>:<jméno typu komponenty> use entity<jméno knihovny>.<jméno entity> [(<jméno architektury>)];
instalace a zapojení komponent:
<label> : <seznam parametrů>
[generic map (<seznam parametrů>)] --aktuální parametry port map (<mapování portů>) ;
8) Simulační systémy
a) jaký je VHDL
b) typ simul. systému, který nepoužívá seznam udalostí
c) existuje ve VHDL podpora pro koprogramy?

2) Typy paralelních příkazů
- nepodmíněné signálové přiřazovací příkazy (<= after), generate, podmíněné signálové přiřazovací příkazy (when, with), assert (paralelní), odložený příkaz assert, paralelní příkaz procedury, příkaz bloku (block), příkaz stráženého bloku (guarded)

28) Typy sekvenčních příkazů ve VHDL
- kterýkoliv příkaz v operační části procesu, procedury či fce, jeho provedení podléhá pravidlům procedurálních programovacích jazyků.
- příkaz assert (sekvenční), přiřazovací příkaz (sekvenční) pro signály, přiřazovací příkaz pro proměnné, volání podprogramů (sekvenční), sekvenční řídící příkazy (wait, if-then-end if, case, loop, exit, next, return, null)

28) Citlivostní seznam: důvod zavedeni, omezeni
- seznam v hlavičce procesu určující signály, jejichž změny mají za následek automatickou aktivaci daného procesu od jeho začátku.
- procesy obsahující citlivostní seznam nesmí obsahovat wait.
- použitím citlivostního seznamu dojde ke zpřehlednění procesu.

9) Generic ve VHDL
- mechanismus pro jednorázové předávání hodnot entitám (konstanty, které mohou ovlivnit simulaci ale nemohou být modifikovány)
- použití: zpoždění, zátěž obvodu,počet vstupů
- předání hodnot parametrů:
1) inicializací generic constant v entitě - platí pro všechny architektury této entity
2) explicitně při instalaci komponenty v architektuře nebo v konfiguraci; toto explicitní předání lze vynechat, byla-li definována implicitní hodnota dle 1)

12) Generate ve VHDL
Paralelní příkaz, který umožňuje opakované provádění příkazů data-flow. Použití je užitečné při popisech "pravidelně propojených elementů" (tzn. množiny paralelních příkazů, jejichž výrazy se liší pouze indexy vstupních či výstupních signálů).
Dvě formy příkazu:
1) Iterační typ - obdoba cyklu v sekvenčním prostředí.
<label> for <identifikator> in <rozsah> generate <paralelni prikaz>...; end generate;
2) Podmíněný typ - obdoba příkazu if ze sekvenčního prostředí bez možnosti else. Smysluplné použití je v iteračním generate.
<label> if <podminka> generate <paralelni prikaz>...; end generate;

13) Procesy ve VHDL - komunikace, synchronizace
Část programu, kterou lze dynamicky aktivovat v závislosti na vnější události. Provádění procesu lze přerušit a později lze pokračovat od místa přerušení. Všechny procesy ve VHDL jsou implicitně periodické. Princip aktivace je v kooperativním multitaskingu. Na začátku simulace je automaticky zajištěna aktivace všech procesů. Proces má deklarativní a příkazovou část. Rozlišujeme dva typy procesů:
1) s citlivostním seznamem – seznam signálů, jejichž změna způsobí provedení procesu od začátku. Zde nelze použít příkaz **wait**. Po skončení procesu se odevzdá řízení jádru.
2) s příkazem **wait** – stanoví podmínky pro příští pokračování procesu, ukončí jeho provádění a vytvoří reaktivní bod (např. **wait for** 10 ns – čeká po dobu 10 ns, **wait on** a – čeká na změnu signálu a)

Odložený proces – účelem je definovat poslední události pro určitou hodnotu modelovacího času, tj. operační část je spuštěna až po ustálení hodnot všech signálů v daném časovém okamžiku. Deklarace:
postponed process(.....)
begin ..... end process;
Odložený proces může být pouze jeden. V operační části nelze použít příkaz wait s nulovým zpožděním.

24) Procedury ve VHDL. Charakteristika, příklady, použití.
Procedura může být deklarována:
- v architektuře; příkazy takové procedury lze použít jak v paralelním prostředí (kdy hraje podobnou roli jako proces a je aktivována pomocí signálů) tak i uvnitř procesu (kdy její aktivace podléhá veškerým pravidlům sekvenčního prostředí),
- v procesu; v takovém případě je procedura v daném procesu lokální a její příkaz bude prováděn pouze sekvenčním způsobem.
parametry: signály, proměnné, konstanty
priklad:
procedure NAS(signal a,b: in bit_vector(3 downto 0); signal soucin: out bit_vector(7 downto 0)) is
variable pom: bit_vector(7 downto 0); begin .... end procedure;

14) Typy Dat ve VHDL
1) skalární typy (lze uspořádat, nelze dekomponovat)
(diskrétní: integer,výčtový ; real; fyzikální)
2) složené typy (pole, záznam)
3) přístupový typ (ukazatel)
4) soubor
- existují jako předdefinované typy (lze definovat i nové typy)
- pro přesun hodnot mezi typy jsou nutné explicitní typové konverze

29) Datové typy VHDL
1) skalární typy (lze uspořádat, nelze dekomponovat)
(diskrétní: integer,výčtový ; real; fyzikální)
2) složené typy (pol, záznam)
3) přístupový typ (ukazatel)
4) soubor
- existují jako předdefinované typy (lze definovat i nové typy)
- pro přesun hodnot mezi typy jsou nutné explicitní typové konverze

31) Delta zpožděni, k čemu to je, proč je
- delta zpoždění - používá se jako malé zpoždění např. při přiřazování hodnot v procesu
- vliv na simulaci - bez existence delta zpoždění dochází před uložením vypočtených hodnot signálů a naplánování příslušných citlivých procesů k posunu modelového času

25) Příkaz assert ve VHDL - důvod zavedeni, možnosti výskytu
Použití: kontrola nepřípustných situací: zakázané kombinace vstupních hodnot, kontrola parity, porušení časových omezení (předstih, přesah)
Účel: test splnění nějaké podmínky (v okamžiku provedení příkazu)
Syntax: assert <podmínka> report <zpráva> severity <error level>;
4 úrovně chyb rozlišují vážnost situace - lze nastavit na kteroukoliv : note | warning | error | failure, efektem je že pokud není splněná podmínka, dojde k výstupu zprávy, případně zastavení simulace
příklady použití:
1) assert now<=100 ms report "vyčerpání času - konec simulace" severity failure;

2) assert (not (clk´delayed(presah)=´1´ and clk'delayed(presah)'event) or D´stable(presah)) report "nedodrzen přesah" severity error ;

26) Příkaz wait ve VHDL - charakterizujte jednotlivé typy
wait on <seznam signálů> - proces je aktivován po změně některého ze signálů
wait until <podmínka> - proces je aktivován po splnění podmínky
wait for <časový údaj> - proces je aktivován za časový interval
30) Možnosti VHDL z hlediska implementace hiearch. struktur
- více konkrétní než model chování
- vyjadřuje konkrétní vazby na jiné (dílčí) jednotky
- zahrnuje modely dílčích jednotek
- v každé architektuře lze mixovat strukturní popis, data-flow popis a sekvenční popis
- umožňuje hierarchické uspořádání modelu struktury mohou být zahnízděny do hloubky

23) Operátor sjednoceni ve VHDL. Charakteristika, příklady, použití.
& - binární operátor, vyžaduje operandy stejného typu
- nejčastější použití: paralelní sjednocení vodičů, rozšíření operandů, sjednocení stringů, posuvy

32) Použití bloku ve VHDL
- mechanismus pro vnitřní členění paralelního prostředí v architektuře (příkaz bloku = paralelní příkaz)
- bloky lze vzájemně vnořovat
- v bloku lze deklarovat vše co v architektuře
- definice jsou pouze lokální pro daný blok
- bloky mohou obsahovat porty, které umožňují mapovat signály z nadřazeného bloku do vnitřních signálů bloku
- strážené bloky umožňují specifikovat podmínku pro synchronizaci paralelních příkazů

34) Konfigurace strukturních schémat ve VHDL: důvod zavedeni, charakteristika, jejich možnosti
mechanismus pro:
a) vzájemné přiřazení architektury a entity (při vynechání: automatický výběr posledně překládané architektury),
b) přiřazení dvojice entita+architektura komponentě
- obě výše zmíněná přiřazení lze provést na různých úrovních vývoje modelu:
a) konfigurace v deklarační části architektury (component specification)
for K1: S16 use entity work.E193(A193); --individuální osazení komponenty typu S16 dvojicí E193 + A193
for K1: S16 use entity work.E193; --přiřadí se "nejmladší architektura", tj. přeložená jako poslední
b) konfigurace vně architektury (cofiguration declaration)
-> samostatná část zdrojového programu: umožňuje odložit osazení soklů a toto soustředit do jednoho místa -> možnost rychlé záměny integrovaných obvodů bez nutnosti nového překladu osazované strukturní architektury
příklad: explicitní výběr architektury A8255 pro entitu E8255
configuration Konf_1 of E8255 is
for A8255 end for ; --volba architektury A8255 (pro E8255)
end Konf_1;

---

1) Konzervativní metody paralelní simulace
Jedná se o konzervativní synchronizační algoritmus.
princip: v průběhu simulace jsou postupně vymezovány tzv. bezpečné zóny, to jsou časové úseky, ve kterých jednotlivé procesy nemohou být vzájemně ovlivňovány.
Synchronizace procesů v jednotlivých krocích se provádí pomocí centrální nebo distribuované bariéry
2) Synchronní metoda (popsat , kam patří – konzervativni)
Jedná se o konzervativní synchronizační algoritmus. Princip: v průběhu simulace jsou postupně vymezovány tzv. bezpečné zóny, to jsou časové úseky, ve kterých jednotlivé procesy nemohou být vzájemně ovlivňovány.;;
Synchronizace procesů v jednotlivých krocích se provádí pomocí centrální nebo distribuované bariéry.
10) Princip metody uváznuti a zotaveni procesu v paralelní simulaci
- jedná se o konzervativní synchronizační algoritmus (respektuje příčinné závislosti, algoritmus je založen na rozlišení bezpečných událostí)
- princip: v průběhu simulace dochází ke střídání dvou fází:
provádění simulace s průběžnou detekcí uváznutí procesů (dynamické vytváření stromu zotavených procesů)
zotavení z uváznutí (uvolnění nejpomalejšího procesu)
6) Paralelní komunikace, ten nejednodušší konzervativní algoritmus
Algoritmus 1: Chandy-Misra-Bryant
- Start
- Čekej dokud každá ze vstupních front neobsahuje aspoň jednu zprávu.
- Nalezni vstupní (vnější) zprávu s minimální hodnotou časové známky.
- Proveď všechny bezpečné vnitřní události procesu (tj. události ze seznamu SQS, jejichž čas je menší než hodnota časové známky kterékoliv vnější události)
- Vyjmi z komunikačního kanálu zprávu s minimální časovou známkou, nastav modelový čas na hodnotu její časové známky a proveď příslušnou událost.
- goto Start
11) popsat konzervativní algoritmy synchronizace procesu
- respektují příčinné závislosti
- algoritmus je založen na rozlišení bezpečných událostí
- např. metoda nulových zpráv, Chandy-Misra-Bryant, metoda uváznutí a zotavení procesů, synchronní metoda
4) Nulové zprávy – metoda nulových zpráv
- jedná se o konzervativní synchronizační algoritmus, konkrétně o opatření zabraňující zablokování Algoritmu 1.
- mechanismus nulových zpráv se používá pro šíření hodnot předstihů mezi procesory. Nulové zprávy nerepresentují žádnou skutečnou událost; definují horní mez modelového času, do níž může pokročit simulace procesů na daném procesoru aniž by riskovala přijetí dalších zpráv a v důsledku toho narušení neklesající posloupnosti přidružených časů.
- možnosti odesílání výstupních zpráv: do všech výstupních směrů po zpracování každé vstupní události v daném procesoru (velký počet zpráv), blokovanému procesoru na jeho žádost (delší časové zdržení)
- hodnota předstihu procesu se může měnit v průběhu simulace a má vliv na efektivitu simulačního algoritmu
- tato metoda selhává v případě výskytu cyklu, ve kterém všechny objekty vykazují nulovou hodnotu předstihu.
3) Co udělá proces po přijetí antizprávy?
1) událost odvozená od sdružené vstupní zprávy nebyla dosud provedena: příslušná antizpráva způsobí pouze odstranění jejího záznamu.
2) událost odvozená od příslušné antizprávy byla již provedena: dojde ke zpětnému běhu z důvodu eliminace vlivu této události (nastavení stavu, odeslání antizpráv)
3) původní sdružená zpráva nebyla (z důvodu nějakého zpoždění) procesem dosud přijata: antizpráva se zařadí do seznamu událostí a po přijetí původní zprávy bude pouze odstraněna
5) Antizprávy - k čemu jsou dobry a čeho se týkají
Jedná se o základní charakteristiku metody Time Warp. Používají se k eliminaci vlivu odeslaných zpráv.
princip: kromě každé odesílané výstupní zprávy proces automaticky vytváří druhou identickou zprávu s nastaveným příznakem anti. V okamžiku odeslání původní zprávy je příslušná antizpráva vložena spolu s její časovou známkou do fronty antizpráv. Antizprávy jsou odeslány procesem při zpětném běhu za účelem eliminace jejich sdružených původních zpráv.
reakce procesu na přijetí antizprávy:
1) událost odvozená od sdružené vstupní zprávy nebyla dosud provedena: příslušná antizpráva způsobí pouze odstranění jejího záznamu.
2) událost odvozená od příslušné antizprávy byla již provedena: dojde ke zpětnému běhu z důvodu eliminace vlivu této události (nastavení stavu, odeslání antizpráv)
3) původní sdružená zpráva nebyla (z důvodu nějakého zpoždění) procesem dosud přijata: antizpráva se zařadí do seznamu událostí a po přijetí původní zprávy bude pouze odstraněna
7) Optimistické metody-charka
- nerespektují závislosti typu příčina-následek
- procesy nerozlišují bezpečné či jiné události; jsou vybavené mechanismy pro eliminaci vlivu chybně provedených událostí (pomocí zpětných běhů (tj. návratů k menším hodnotám modelového času) anulují efekty všech předčasně provedených událostí, pak pokračují novým dopředným během)
8) Role globálního virtuálního času u optimistických metod paralelní simulace.
Optimistické metody (metody Time Warp) pomocí návratů k menším hodnotám modelového času anulují efekty všech předčasně provedených událostí,
pak pokračují novým dopředným během.
9) Agresivní a zpožděna eliminace vlivu odeslaných zprav
1) Agresivní rušení
- zpětný běh ruší záznamy všech provedených událostí v předešlém dopředném běhu (vyjma událostí naplánovaných v menším čase než je dolní mez zpětného běhu a událostí ve vstupních frontách) a všechny zprávy které byly odeslány jiným procesům.
- následující dopředný běh generuje znovu všechny výstupní zprávy.
2) Opožděné (líné) rušení
- vychází z předpokladu, že většina zpráv odeslaných v předchozím běhu zůstane v platnosti a jejich rušení ve zpětném běhu je zbytečné.
- všechny antizprávy v průběhu zpětného běhu zůstávají zachovány
- v průběhu následujícího opravného dopředného běhu se antizprávy odesílají pouze v případě, že by nedošlo k opakovanému vyslání původních sdružených zpráv; v opačném případě se antizprávy zachovávají pro případ dalších zpětných běhů.
- výhody: odpadá násilné rušení událostí, které budou pravděpodobně obnoveny
- nevýhody: opožděné vyslání antizpráv umožňuje větší rozšíření chyb

3) Poruchy kanálu obsluhy vlivem opotřebení
mohou nastat pouze během obsluhy; možná koncepce:
* proces požadavek: beze změn
* proces obsluha:
- při každém začátku nějaké obsluhy aktivuje proces porucha
- při každém ukončení obsluhy volá funkci Interrupted() (zda jde o řádné ukončení obsluhy či nikoliv):
-> při řádném ukončení pokračuje obsluhou dalšího požadavku
-> při poruše se proces pasivuje a čeká na aktivaci od poruchy; pak pokračuje obsluhou přerušeného požadavku
* proces porucha:
- generuje dobu bezporuchového provozu
- testuje, zda porucha nastane při právě probíhající obsluze;
pokud ne tak provede modifikaci atributu "do_poruchy" a pasivaci.
pokud ano přeruší obsluhu pomocí funkce Interrupt()
- čeká na odstranění poruchy, pak aktivuje obsluhu (s prioritou)

11) Nezávislé poruchy kanálu obsluhy u SHO - vlastnosti, nástin simulace
- mohou nastat kdykoliv; možná koncepce:
proces požadavek - beze změny
proces porucha - běží nezávisle na procesu obsluhy (dle střední doby bezporuchového provozu a dle střední doby trvání poruchy)
- v případě poruchy volá funkci Interrupt a po ukončení poruchy naplánuje obsluhu pro daný čas
proces obsluha - přerušení obsluhy - podobně jako u silných priorit pomocí funkce Interrupt; proces však nepokračuje obsluhou požadavku z jiné fronty. ale čeká na aktivaci od procesu porucha
- reakce procesu obsluha na přerušení: předpokládáme, že v obsluze lze pokračovat (jiná možnost: je-li obsluha nepřerušitelná vyřadit obsluhovaný požadavek jako zmetek)

27) Modelováni SHO – možné přístupy a jejich srovnáni
1) Metoda plánování událostí
- aktivovat události lze pouze na základě hodnot modelového času
- neex. možnost aktivovat události v závislosti na splnění nějaké podmínky
- popis událostí je formou podprogramů
- změny atributů modelu jsou vázány výhradně na provedení zmíněných podprogramů
- podprogramy událostí proběhnou bez přerušení
- podprogramy jsou plánovány do seznamu událostí dle hodnot modelového času
2) Metoda sledování aktivit
- netypický přístup: chybí plánování.
- metoda umožňuje aktivovat události na základě modelového času nebo nějaké podmínky
- simulační program je rozdělen na části, které simulují jednotlivé události v sytému, tento program je periodicky prováděn. V závislosti na podmínkách se jednotlivé části buď provedou nebo ne.
3) Metoda interakce procesů
- aktivace dle času nebo podmínky
- lze použít princip plánování jako u plánování událostí
- je nutná podpora pro koprogramy
- z důvodu podmíněných zpoždění je komplikovanější synchronizace

4) Způsoby generování náhodných čísel, srovnáni metod
Kongruenční metody - vychází z rekurentních algoritmů - rychlý výpočet, minimalní nároky na paměť, reprodukovatelnost generování, konečná délka slova, konečná perioda.
- Smíšená kongruenční metoda:
volba parametru m: délka periody (zpravidla m=2^beta ; m=2^beta +1; m=prvočíslo)
volba parametrů a, c: pro dosažení plné periody 1) c,m nesoudělná 2) (a=1) mod q (q je prvočinitelem m)
- Multiplikativní kongruenční metoda: (MLCG) $x_{n+1}=(a*x_n) \bmod m$
rychlejší generování, nelze dosáhnout plné periody
- Multiplikativní rekurzivní generátory: složitější závislost (na více číslech jedné posloupnosti)
- Kombinované lineární kongruentní generátory: založeny na kombinaci hodnot několika nezávislých MLCG
- Míchající generátory - posloupnost generovaná jedním generátorem je promíchávána pomocí druhého
- Generátory s využitím přenosu - velmi dlouhé periody, vyžadují více minulých hodnot

10) Generátor náhodných čísel - co je to perioda, max. perioda a které parametry ovlivni periodu?
- (kongruenční metody) vychází z rekurentních algoritmů - rychlý výpočet, minimalní nároky na paměť, reprodukovatelnost generování, konečná délka slova, konečná perioda.
- Smíšená kongruenční metoda:
volba parametru m: délka periody (zpravidla m=2^beta ; m=2^beta +1; m=prvočíslo)
volba parametrů a, c: pro dosažení plné periody 1) c,m nesoudělná 2) (a=1) mod q (q je prvočinitelem m)
- Multiplikativní kongruenční metoda: (MLCG) $x_{n+1}=(a*x_n) \bmod m$
rychlejší generování, nelze dosáhnout plné periody
- Multiplikativní rekursivní generátory: složitější závislost (na více číslech jedné posloupnosti)
- Kombinované lineární kongruentní generátory: založeny na kombinaci hodnot několika nezávislých MLCG
- Míchající generátory - posloupnost generovaná jedním generátorem je promíchávána pomocí druhého
- Generátory s využitím přenosu - velmi dlouhé periody, vyžadují více minulých hodnot

7) Vylučovací metoda (zamítací metoda)
Vylučovací (zamítací) metoda transformace náhodných čísel (von Neumann)
1) Generuj dvojici rovnoměrně rozložených čísel u1,u2: u1=U(a,b) a u2=(0,max(f(x)).
2) Pokud u2>f(u1), ignoruj obě čísla a pokračuj od bodu 1.
3) Pokud u2*f(u1), pak použij číslo u1; číslo u2 ignoruj.
Použití: Gaussovo (normální) rozložení, nelze použít inversní metoda, uvažovaný obor hodnot normovaného rozložení pro malé serie: (- 3s , +3s), pro velké série (-5s , + 5s)

20) SHO: Výukový systém v C++, stavy procesu a přechody mezi nimi
aktivní - jde o 1. proces v seznamu událostí
naplánovaný (potlačený) - není aktivní, ale má záznam v SQS
pasivní - proces nemá záznam v seznamu událostí
ukončený - nemá záznam v seznamu událostí a proběhly všechny fáze, do tohoto stavu lze přejít pouze z aktivního
Prechody: - z aktivního do pasivniho: void Passivate();
5) Vyjmenovat všechny metody pro přechod procesu ze stavu
a) potlačeného do aktivního - void Cancel(CProcess* p); //převede proces p do stavu pasivní - pokud jde o aktivní proces, pak je funkce stejná jako v případě Passivate()
b) ze stavu aktivniho do stavu potlačeného - // private: void ResumeCurrent() // umožní spuštění 1. procesu dle SQS a suspenduje právě běžící proces; void Passivate(); // převede aktivní proces do stavu pasivní (zruší příslušný záznam v SQS) a volá funkci ResumeCurrent(); void Wait(CHead* S); // převede aktivní proces do stavu pasivní - zařadí jej do seznamu S a provede funkci jako Passivate(); void Cancel(CProcess* p); //převede proces p do stavu pasivní - pokud jde o aktivní proces, pak je funkce stejná jako v případě Passivate(), pokud jde o potlačený proces, pak jde o pouhé vyjmutí příslušného záznamu ze SQS
18) Všechny funkce na převod procesu z aktiv. do pasiv. (Passivate, Cancel, Wait)
// private: void ResumeCurrent() // umožní spuštění 1. procesu dle SQS a suspenduje právě běžící proces
void Passivate(); // převede aktivní proces do stavu pasivní (zruší příslušný záznam v SQS) a volá funkci ResumeCurrent()
void Wait(CHead* S); // převede aktivní proces do stavu pasivní - zařadí jej do seznamu S a provede funkci jako Passivate()
void Cancel(CProcess* p); //převede proces p do stavu pasivní - pokud jde o aktivní proces, pak je funkce stejná jako v případě Passivate(), pokud jde o potlačený proces, pak jde o pouhé vyjmutí příslušného záznamu ze SQS
35) C++ metody pro prechod mezi aktivní a potlačený
// private: void ResumeCurrent(); // umožní spuštění 1. procesu dle SQS a suspenduje právě běžící proces
void Hold(TIME T); // převede aktivní proces do stavu potlačený, naplánuje jeho příští fázi bez priority a volá fci ResumeCurrent()
void Cancel(CProcess* p); //převede proces p do stavu pasivní - pokud jde o aktivní proces, pak je funkce stejná jako v případě Passivate(),

15) Slabé priority
(priority se slabou předností)
- požadavek s vyšší prioritou nechá dokončit právě probíhající obsluhu.
- každý požadavek může aktivovat obsluhu, to ale nemá žádný význam, pokud je obsluha právě v činnosti.
- priorita se realizuje buď více frontami (pro každou prioritu jedna) nebo atributem priorita, pak je ale nutná změna algoritmu zařazování do front.

30) Silné priority požadavků, nástin simulace
(priority se silnou předností)
- požadavek s vyšší prioritou okamžitě přeruší právě probíhající obsluhu požadavku s nižší prioritou
- předpoklad: případné přerušení nemá vliv na kvalitu obsluhy a v přerušené obsluze lze později pokračovat
- proces požadavek opatříme atributem doba_ob, pro evidenci dosud vykonané obsluhy. V případě příchodu nového požadavku obsluha ukončí provádění požadavku, vyhodnotí prioritu a zahájí obsluhu požadavku s vyšší prioritou.

12) Popište funkce metod waituntil(...) a signal()

- metody třídy TCondq, Q...fronta procesů čekajících na určitou podmínku
waituntil(P); - metoda pro testování podmínky, v případě nesplněné podmínky pozdrží se Current() proces ve frontě Q, v případě splněné podmínky jde o prázdný příkaz
signal(); - metoda pro aktivaci procesů z fronty Q

6) Co dělají (a jak?) coopt(), waitq(), find()

coopt() - metoda pro sloučení dominantního procesu s procesem doprovodným
waitq() - metoda pro sloučení doprovodného procesu s procesem dominantním
find() - metoda pro sloučení dominantního procesu s procesem doprovodným, který navíc vykazuje určité vlastnosti (definované pomocí parametru této funkce)

23) Popište stručně vnitřní princip funkci pro slučováni a rozlučováni procesu v simulačních systémech diskrétního typu.

- 2 typy procesů: dominantní - určuje společný děj, doprovodný - ve společném úseku bez vlastního děje
- implementace: pomocí objektů třídy TCoopt - vlastnosti:
  master_q - fronta dominantních procesů
  slave_q - fronta doprovodných procesů
  coopt() - metoda pro sloučení dominantního procesu s procesem doprovodným
  waitq() - metoda pro sloučení doprovodného procesu s procesem dominantním
  find() - metoda pro sloučení dominantního procesu s procesem doprovodným, který navíc vykazuje určité vlastnosti (definované pomocí parametru této funkce)

příklady použití: převoz aut přes mořskou úžinu, "seznamka": nezávislé životy před sňatkem, společný život po sňatku

26) Funkce acquire() a release() v simulačních systémech diskrétního typu

acquire() - nejsou-li zdroje pozdrží aktivní proces; release() - uvolní zdroje a aktivuje čekající procesy (použití u semaforů, např. pro řešení problému producent – konzument).

8) Diskrétní simul. systémy se seznamem událostí - principy, vlastnosti

Simulační systémy diskrétního typu nabízí kromě běžných konstrukcí univerzálních programovacích jazyků další prostředky pro podporu následujících činností:
- generování pseudonáhodných čísel pro často používaná rozložení
- generování a rušení přechodných objektů (nebo garbage collector)
- ovládání seznamu událostí
- provádění často používaných operací nad seznamy objektů
- statistická vyhodnocení sledovaných veličin (výpočet středních hodnot, histogramů, odchylek)

14) Metoda Monte Carlo

- převádí úlohu na stochastický proces, tento simuluje na počítači a statisticky vyhodnotí výsledky
- jde o napodobení velkého počtu skutečných situací; v těchto situacích jsou vyhodnocovány sledované veličiny a nakonec statisticky vyhodnoceny
- na rozdíl od analytických modelů lze metodu Monte Carlo vždy aplikovat
- přesnost: P[|chyba| <= 3*sigma/sqrt(n)]=0,997

24) Systém M/M/1. Stručná char., výsledky anal. modelu, podmínka stability

-příchody požadavků do SHO - Poissonův proces
-počet obsloužených požadavků - Poissonův proces
-počet kanálů obsluhy = 1
Cílem je zjistit E(Nq), D(Nq), Nq...celkový počet požadavků v SHO
E(Tq), D(Tq), Tq...celková doba strávená požadavkem v SHO
Stabilní při intenzitě provozu r<1 v SHO s neomezenou frontou

1) Co je potřeba k modelováni M/M/1

M/M/1 - příchod požadavků do SHO i počet obsloužených požadavků se řídí Poissonovým procesem, počet kanálů obsluhy =1
a) vstupní informace - Nq...celkový počet požadavků v SHO
- Tq...celková doba strávená požadavkem v SHO
b) vyst.inf ktere muzeme ziskat analyticky a simulaci - Cílem je zjistit E(Nq), D(Nq)
- E(Tq), D(Tq)

9) Analytické a simulační metody – rodily ???

mate M/M/1 - E(Nq)=lambda/(lamdba-mi), vytizenost obsluhy 50%, jak dlouha bude fronta?

28) Poissonův proces: charakteristika, použití M/M/1

Poissonův proces - modelování výskytu nezávislých událostí. Předpoklady: příchod požadavku v intervalu dt->0 nezávisí na příchodech požadavků v předcházejících intervalech, pravděpodobnost počtu příchodů odpovídá Poissonovu rozdělení.
M/M/1 - příchod požadavků do SHO i počet obsloužených požadavků se řídí Poissonovým procesem, počet kanálů obsluhy =1

33) Simulacni systemy diskretnoho typu orientovane na procesy

Simula 67:
- možné hierarchie tříd, dědičnost atributů z nadřazených tříd
- virtuální procedury
- chráněné atributy
- vyjma metod třídy disponují i tzv. operační částí třídy:
  * rozštěpená operační část - možnost vkládání operačních částí odvozených pod tříd do operační části nadřazené třídy
  * charakter koprogramu (možnost osamostatnění objektu s vlastním zásobníkem)

25) Simula 67: celkový přínos, charakteristika podpory pro sim. diskr. systémů

- možné hierarchie tříd, dědičnost atributů z nadřazených tříd
- virtuální procedury
- chráněné atributy
- vyjma metod třídy disponují i tzv. operační částí třídy:
  * rozštěpená operační část - možnost vkládání operačních částí odvozených pod tříd do operační části nadřazené třídy
  * charakter koprogramu (možnost osamostatnění objektu s vlastním zásobníkem)

1) Test Kolmogorov-Smirnof

- test náhodných čisel založený na porovnání teoretické a empirické distribuční fce.
  Kn+ = n^1/2 * max(Fn(x)-F(x)), Kn- = n^1/2 * max(F(x)-Fn(x)), rozložení hodnot Kn+ a Kn- je tabelováno
- pouze pro spojité náhodné veličiny

19) Test dobré shody

1) získání kontrolovaného souboru hodnot
- generování kontrolní serie n hodnot
- rozdělení vygenerovaných hodnot do k disjunktních kategorií
- nalezení počtu hodnot v jednotlivých kategoriích
2) získání srovnávacího (kontrolního) souboru hodnot
  použití: - distribuční funkce F (x) :
- pro odhad počtu hodnot v kategorii definované intervalem (r, s> platí: n*pi=n*(F(s)-F(r))
  n...celkový počet hodnot testované serie, pi...pravděpodobnost hodnoty z intervalu (r, s>, požadavek: n*pi > 5
- hustota pravděpodobnosti f(x): n*pi=n*Int(f(x)dx), frekvenční funkce p(x): n*pi=n*(pk + pk+1 + ... + pk+q)
3) srovnání obou souborů
- vyhodnocení výrazu V=X^2=Suma[i=1..k]((Yi-n*pi)^2/ n*pi)
  Yi...počet hodnot v kategorii i (kontrolovaný soubor), n*pi...teoretický počet hodnot z kontrolního souboru
4) zhodnocení odchylky obou souborů
  V...hodnota náhodné veličiny s rozložením chi-kvadrat, která má s stupňů volnosti, kde s=k-1
  zhodnocení na základě tabulky pro chi-kvadrat rozložení

29) Charakterizujte: frekvenční test, test mezer a dvojic a poker test

Jedná se o testování náhodných čísel (zda jsou čísla opravdu náhodná).
- frekvenční test: testuje rovnoměrnost rozložení určitého intervalu (aplikace K-S testu, chi-kvadrát testu)
- test dvojic (test serií pro k=2): testuje k-tice po sobě jdoucích následujících celých čísel
- test mezer: určen pro rovnoměrně rozložená čísla z intervalu <0,1>, testuje se délka posloupnosti mezi 2 hodnotami které leží v určitém podintervalu
- poker test: test po sobě následujících k-tic celých čísel, který zjišťuje počet různých číslic v každé k-tici.

17) Popsat metodu skládání (asi kompoziční) jako jednu z transformací náh. čísel.

Kompozice inverzní a vylučovací metody
Postup: Generuj hodnotu i <1,……,k> dle frekvenční funkce f=<S1, S2,…….Sk>, kde S1+ S2+…….+Sk = 1
  S použitím hustoty fi(x) generuj hodnotu x (dle některé z výše uvedených metod) v intervalu na kterém je hustota definována
  fi(x)...hustoty pravděpodobností; k...počet dílčích hustot fi; S...plochy oblastí

13) Inverzní metoda transformace náh. čísel + příklad

Generování hodnot náhodných veličin s požadovaným rozložením:
1. generování rovnoměrných rozložených celých čísel (pomocí kongruenčních metod)
2. transformace celých čísel xi na reálná čísla ui z intervalu <0,1) (ui = xi / 2^β)
3. transformace reálných čísel ui na požadované rozložení

Inversní metoda – F(x) = u, u ∈ U(0,1) => x = 1 / F(u), kde F(x) je distribuční funkce

---

a) inverzní metoda
- nejjednodušší algoritmus, ne vždy použitelný
F(x)=u, u=U(0,1) => x=F^-1(u)
b) rovnoměrné rozložení na intervalu (a,b)
F(x)=(x-a)/(b-a)
c) exponenciální rozložení s posunutím o x0

F(x)=1-exp(-lambda(x-x0))
- viz. obrazky [nah_cisla.pdf 15,16]

21) SHO: Popište ideový návrh objektu pro representaci vodičů v sim. systému C++

```
class TSignal              // pro reprezentaci vodičů
{ int branch_num;          // počet připojených součástek
  char val;              // skutečná (efektivní) hodnota signálu
  TComponent* P[5];        //pointry na připojené součástky
  CHead* list_of_dr;       //ptr na seznam budičů tohoto signálu
  void resolve(); void schedule();  //resoluč. f-ce;  plánování následníků
  public: TSignal();
   void connect(TComponent* elem ); //připojení signálu na součástku
   void set1();  void set0();       // set a reset signálu
   char value();                  // vrací hodnotu val signálu
   CHead* drivers();              // vrací odkaz na seznam budičů signálu
   friend class TDelay; };  //pomocný proces realizující zpoždění
```

31) Úrovně abstrakce modelu

úrovně abstrakce číslicových zařízení
funkční úroveň - násobička: y=a*b
úroveň registrů - násobička: 3 registry, sčítačka, posouvací obvod,
úroveň hradel - schema ze základních logických obvodů
fyzikální úroveň - schema s transistory apod.

36) Vyvoj. sim. syst. v C++, popiste fci a vnitrni strikturu seznamu udalosti

- viz. obrazek [vyuk_simul_system.pdf 4]
- stavy procesů: aktivní, naplánovaný (potlačený), pasivní, ukončený
- hlavní metody třídy CLink (operace nad prvky seznamu): Suc(), Pred(), Out(), Follow(CLinkage* X), Precede(CLinkage* X), Into(CHead* F)

22) Výukový simulační systém v C++: charakterizujte koncepci podpůrných tříd včetně jejich funkce.

1) Modul Simset - operace pro práci se seznamy
Třídy CLinkage - obecné vlastnosti prvku seznamu, CLink - operace nad prvky senzamu [Suc(), Pred(), Out(), Follow(), Precede(), Into()], CHead - operace nad seznamem [Clear(), Cardinal(), Empty(), First(), Last()]
2) Modul Simulation - služby kvaziparalelního prostředí (modelový čas+seznam událostí, operace synchronizačního jádra)
Operační část třídy nahrazena virtuální metodou Run().
Hlavní proces nahrazen metodou Run () objektu CSimulation.
Třídy CLink, CEventNotice - definuje objekty sloužící jako záznamy událostí v seznamu událostí, CThread - operace s vlákny, CProcess - základní vlastnosti pro prototypy uživatelových procesů, [Current(), FirstEv(), Time(), NextEv(), Idle(), Terminated(), EvTime(), _activate(), Hold(), Passivate(), Wait(), Cancel(), Run()], CSimulation - analogie hlavního programu

16) Jak v SHO simulovat kolize na Ethernetu

CSMA/CD: Algoritmus procesu „transmitting“:
1) Pokud je sběrnice obsazena (busy = true), zařad právě aktivni proces do seznamu „ready“ a pasivuj tento proces.
2) Generuj kopii prvního rámce ze seznamu „output queue“ právě vysílací stanice a zařaď ji do seznamu „bus“
3) Čekej po dobu nezbytnou pro šíření el. signálu podél celého segmentu.
4) Pokud je počet rámců v seznamu „bus“ roven 1, přejdi na bod 8, jinak pokračuj bodem 5.
5) Čekej po dobu trvání kolizního slotu.
6) Vyjmi objekt reprezentujíc právě přenášený rámec ze seznamu „bus“.
7) Čekej po náhodnou dobu potřebnou pro pozdržení dalšího pokusu o získání sběrnice. Pokračuj od bodu 1.
8) Nastav atributy „busy“ na true a čekej dokud nebude dokončen přenos právě vysílaného paketu – záleží na délce paketu.
9) Vyjmi první rámec ze seznamu „output queue“ a předej ho procesu „transfer“.
10) Aktivuj proces „transfer“ pro okamžitou hodnotu simulovaného času a s prioritou.
11) Je-li seunam „output queue“ prázdný“, pasivuj tento proces.
12) Pokračuj od bodu 1.

32) C++: Ideový návrh stanice v Ethernetu

ptr_segment...pointer na objekt reprezentující segment jako celek,
parameters...parametry stanice (aktivita: četnost generování paketů, charakteristika délky paketů, charakteristika provozu stanice, atd)
station address...adresa stanice v síti (pro směrování v mostu)
initiate...inicializační metoda pro generování a aktivaci všech přidružených procesů a generování přidružených front
- viz. obrazek [aplikace_vyuk_simul_systemu.pdf 47]

34) C++ model pro Ethernet sitovy segment

-cíl: umožnit šíření paketu v celém segmentu, odpojování a připojování jednotlivých stanic
- viz. obrazek [aplikace_vuyk_simul_systemu.pdf 48]