Popis systémů pomocí VHDL

Vývoj VHDL

- **HDL** - **H**ardware **D**escription **L**anguage
- **VHDL** - **V**ery High Speed Integrated Circuits **HDL**
- Vývoj od roku 1983 v rámci projektu **VHSIC**
- **1987 - standard IEEE 1076-1987**
- **1993 - revize IEEE Std 1076-1993**
- 1999 - revize IEEE Std 1076.1-1999
  VHDL-AMS (Analogue & Mixed Signals)
- 2000 - revize IEEE Std 1076-2000
- 2002 – revize IEEE Std 1076-2002
- 2008 – revize IEEE Std 1076-2008
- v návrh. systémech - převážně verze VHDL-87 a VHDL-93

Verilog HDL

- vývoj od osmdesátých let (Verilog–XL)
  (firma Gateway Design Automation)
- v roce 1995 - IEEE Std 1364-1995
- udržován OVI (Open Verilog International)
- rozšířený zejména v Americe a Asii
- jednodušší a omezenější ve srovnání s VHDL
- v roce 2001 – revize IEEE Std 1364-2001
- podobný jazyku C

Další jazyky pro popis HW

Nové jazyky vznikají zejména z důvodu dosažení větší abstrakce od hardwaru, kladou důraz na popis na úrovni algoritmů (nejsou koncipovány pro přímý popis HW – nejsou to HDL jazyky)

SystemVerilog – standard IEEE Std 1800-2005
(vychází z Verilogu)

SystemC – standard IEEE Std 1666-2005
(vychází z objektového jazyka C++)

Handel-C – Celoxica (vychází z jazyka C)

Charakterizace VHDL

- všeobecně přístupný otevřený standard
- jazyk pro popis technických prostředků elektronických systémů
- vhodné pro návrh metodou shora-dolů (top-down)
- nezávislé na budoucí technologii realizace
- důraz na funkci obvodu (oproštění od detailů)
- umožňuje opakované používání modelů (knihovny)
- využití pro dokumentaci a modelování
- paralelní jazyk (ne sekvenční)

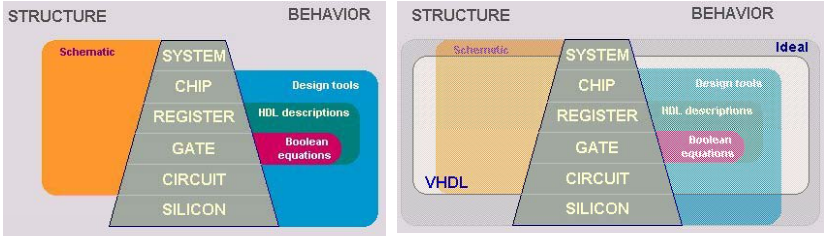
Charakterizace VHDL (pokr.)

- snadná výměna částí návrhů mezi návrháři (IP core)
- libovolná část návrhu může být osamostatněna
- model VHDL může být simulován v různých systémech
- podpora testovatelnosti (Boundary Scan Architecture)
- „upovídaný“ jazyk (opakování bloků, deklarace)
- ne všechny konstrukce jazyka musí být syntetizovatelné (lze pouze simulovat)

VHDL v různých úrovních abstrakce

VHDL lze použít v různých úrovních abstrakce:

- úroveň behaviorální (popis chování obvodu)
- úroveň RTL (Register Transfer Level) – popis toku dat
- úroveň hradel (logická úroveň) - strukturální



Formální vlastnosti VHDL

- při zápisu se nerozlišují malá a velká písmena (není „case sensitive“)
- každý příkaz je ukončen středníkem (;)
- v zápisu jazyka lze pro lepší čitelnost používat libovolný počet mezer („space insensitive“)
- využívá klíčových slov
- žádná pravidla pro jména souborů (doporuč. jméno souboru shodné se jménem nejvyšší entity)

Pravidla jmen identifikátorů

Syntaxe:

```
písmeno{[_]písmeno_nebo_číslice}
```

- nerozlišují se velká a malá písmena
- musí začínat písmenem
- mohou obsahovat písmena, číslice a podtržítka
- jméno nesmí obsahovat mezeru
- nelze použít dvě podtržítka za sebou
- podtržítko nesmí být posledním znakem
- nesmí být totožná s klíčovými slovy
- musí být unikátní
  - nelze použít signál A a současně sběrnici A(7 downto 0)

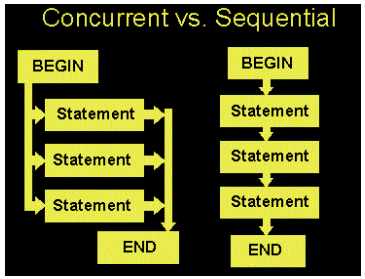
Literály

Zápisy čísel, znaků, soustav, jednotek:

- celočíselné: 725, 10E4, 16#D2#, 2#110110#
  (# změna číselné soustavy, E – exponent (nezáporný))
- reálné: 1.0, -65_971.33_245, 8#43.6#e-4
  (_ slouží pro optické oddělení)
- znakové (character): ‘0’, ‘1’, ‘a’ (použití apostrofů)
- řetězcové (string): ” 1001 ”, ” low” (použití uvozovek)
  string je case sensitive (např.: ”bit” /= ”Bit”)
- bitová pole: b”1100_1001”, X”C3F”, O”754”
- fyzikální: 2.3 us (předdefinováno), 15 kohm
  (povinná mezera mezi hodnotou a jednotkou)

Příkazy

- **Declaration statements**
  - definice konstant, typů, objektů, podprogramů
- **Concurrent statements** – současně probíhající
  - pro popis kombinační logiky
  - např. block, signal assignment, procedure call, ...
- **Sequential statements**
  - spouští se v napsaném v pořadí
  - např. příkazy if, case, loop, next, wait, exit, …
  - obdoba SW programu



Komentář

- začíná dvěma pomlčkami (dash)
- komentář může začínat i za libovolným příkazem
- komentář končí na konci řádku (nelze jiným způsobem ukončit)
- neexistuje víceřádkový komentář (často řešeno až v editorech)
- nedoporučuje se v komentářích používat diakritiku
- v komentářích se někdy objevují i speciální příkazy návrhového systému (např. pro syntezátor)

```
-- toto je komentar
c <= a  AND  b;    -- toto je také komentar
```

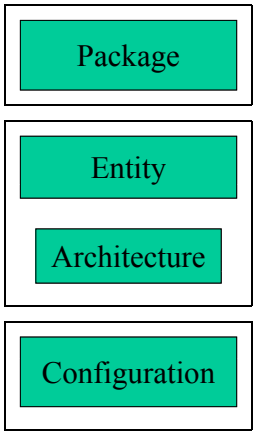
Hlavní komponenty VHDL

dvě povinné komponenty: Entity a Architecture

Příklad - hradlo XOR:

```
ENTITY hr_xor IS
    PORT ( a, b : IN BIT;
           y : OUT BIT );
END hr_xor;

ARCHITECTURE ar_hr_xor OF hr_xor IS
BEGIN
    y <= a XOR b;
END ar_hr_xor;
```



Entita a architektura

- **Entita** - „černá skříňka“ se vstupy a výstupy (obdoba grafického symbolu)
- Entita nepopisuje chování modulů (nedefinuje funkci)

- **Architektura** - určuje chování entit
- tělo architektury má dvě části:
  - deklarační část (např. definice signálů)
  - příkazová část (uzavřeno do ***begin - end***)
- architektura musí být spojena se specifikovanou entitou
- rozdílné architektury definují rozdílné pohledy na entity
- ke každé entitě lze definovat více architektur

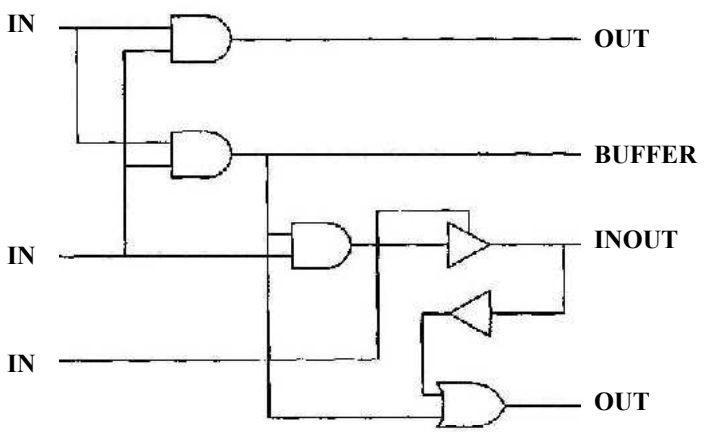
Porty (brány)

- popisují vnější signály entity
- jsou charakterizovány:
  - jménem (libovolná skupina znaků začínající písmenem)
  - módem (určuje směr toku dat) – 5 módů:
    IN, OUT, BUFFER, INOUT, LINKAGE
  - datovým typem (lze spojovat porty stejného typu)

Módy portů

- **IN** – data lze z portu pouze číst
- **OUT** – data vycházejí z portu (výstupní signál nemůže být použit jako vstup uvnitř entity)
- **BUFFER** – výstup se zpětnou vazbou (může být buzen pouze z vnitřku entity - data mohou z entity pouze vystupovat, lze zpětně číst)
- **INOUT** – obousměrný tok (obousměrné vstupy/výstupy)
- **LINKAGE** – neznámý směr datového toku
  - obousměrný (návaznost na jiné než VHDL modely, např. Verilog)

Příklad portů



Porty (pokrač.)

Syntaxe:

PORT (jméno_signálu : mód datový_typ) ;

Příklad:

```
PORT ( vstup : IN  BIT ;
       a1, b1 : IN  BIT_VECTOR (3 downto 0) ;
       vystup : OUT  BIT ) ;

soucet : OUT BIT_VECTOR (0 TO 7) ;          -- 0 je MSB, 7 je LSB
operand : IN BIT_VECTOR (4 DOWNTO 0) ; -- 4 je MSB, 0 je LSB

-- nelze BIT_VECTOR (4 TO 1) nebo BIT_VECTOR (0 DOWNTO 5)
```

Položka Generic

- obdoba portů, ale nepředstavuje žádný signál
- obdoba konstanty (viditelná v entitě a přiřazených architekt.)
- používá se jako parametr (neměnící se v čase)
- použití pro lepší čitelnost, správu a konfiguraci

Syntaxe:

GENERIC (generic_jméno : datový_typ [:= hodnota] ;

Příklad:

```
GENERIC (Delay : integer := 5) ; -- časový parametr
GENERIC (BusWidth : integer := 8 ); -- velikost objektu
GENERIC (Loop : integer := 3); -- proměnný počet smyček
```

Datové objekty

Ve VHDL jsou 3 třídy datových objektů:

- **Constants** (konstanty) – mají neměnnou hodnotu (nelze dále měnit ⇒ lze psát pouze na pravé straně přiřazení)
- **Variables** (proměnné) – používají se jako pomocné objekty (nepředstavují skutečné signály, nelze je použít jako porty, jsou lokální v procesech a procedurách)
- **Signals** (signály) – většinou jsou fyzicky přítomné ve formě elektrických signálů

Datové objekty (pokrač.)

Syntaxe:

CONSTANT jméno_konstanty : jméno_typu [:= hodnota] ;
VARIABLE jméno_proměnné : jméno_typu [:= hodnota] ;
SIGNAL jméno_signálu : jméno_typu [:= hodnota] ;

- **Jméno_typu** použijeme standardní nebo je nutné jej definovat příkazem TYPE ;
- nastavení hodnoty není podporováno syntézou (slouží jen k simulaci)

Příklady:

```
CONSTANT pi : REAL := 3.14 ;
CONSTANT rychlost : INTEGER ;          -- defaultní hodnota: 0
VARIABLE suma : BIT_VECTOR (0 TO 3) := "0010" ;
SIGNAL select : STD_LOGIC ;
```

Datové typy

Deklarace portů, signálů, proměnných musí obsahovat typ (nebo podtyp):

- Skalární (Scalar) typy
  - **INTEGER** (např. 11; ne 7.5) – standardně rozsah $\pm(2^{31}-1)$
  - **REAL** (např. 1.7E3 nebo -5.4; ne 1 nebo 9.1ns) – rozsah ±1.0E38
  - **ENUMERATED** (užívá předem specifikované hodnoty) - výčet
  - **PHYSICAL** (požaduje připojení fyzikální jednotky)
- Složené (Composite) typy
  - **ARRAY** (skupina elementů stejného typu spojených do jednoho objektu) - pole
  - **RECORD** (skupina elementů různých typů v jednom objektu) - záznam
- Přístupové (Access) typy
- Souborové (File) typy

Datové objekty a typy

Příklady deklarací různých typů:

```
TYPE byte  IS RANGE 10 DOWNTO -10 ;  -- celočíselný typ
SIGNAL c1 : byte ;

TYPE prepinac IS ( ON, OFF ) ;          -- výčtový typ
VARIABLE  a: prepinac ;
a := ON ;

TYPE resistance IS RANGE 0 TO 100000 ;  -- fyzikální typ
   UNITS Ohm ;                          -- jméno základní jednotky
      KOhm = 1000 Ohm ;                 -- definování dalších jednotek
      Mohm = 1000 Kohm ;
   END UNITS resistance ;
```

Vícehodnotová logika

Definována v package „std_logic_1164“

std_logic a **std_ulogic** – výčtové typy definující 9 hodnot – všechny hodnoty jsou typu „character“ – ‘U’, ‘X’, ‘Z’, ‘W’, ‘L’, ‘H’, ‘-’ (nutno psát velkými písmeny)

```
TYPE std_ulogic IS (
   'U',   -- neinicializováno (signál nebyl dosud buzen), implicitní
   'X',   -- neznámá hodnota (vzniká při konfliktu '0' a '1')
   '0',   -- log. 0 z tvrdého zdroje
   '1',   -- log. 1 z tvrdého zdroje
   'Z',   -- vysoká impedance
   'W',   -- neznámá hodnota (vzniká při konfliktu H a L)
   'L',   -- log. 0 z měkkého zdroje
   'H',   -- log. 1 z měkkého zdroje
   '-' ) ; -- neurčená hodnota (don't care), na hodnotě nezáleží
```