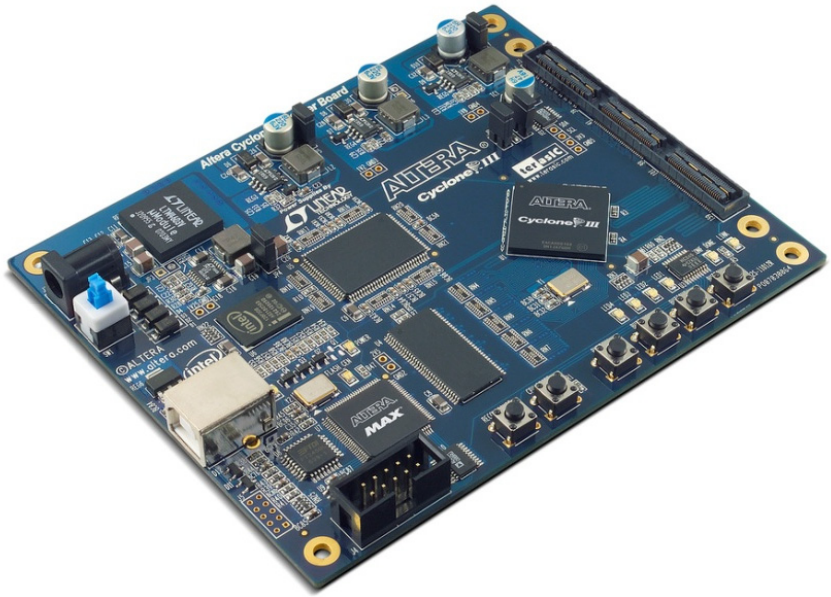
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

Ing. Radek Sedláček, Ph.D. , katedra měření K13138



Jazyk VHDL – zápis čísel, znaků a řetězců

Jazyk VHDL – základní datové typy a operátory

Kurz A0B38FPGA – Aplikace hradlových polí

OPERAČNÍ PROGRAM PRAHA ADAPTABILITA

PRAHA PRAGUE PRAGA PRAG

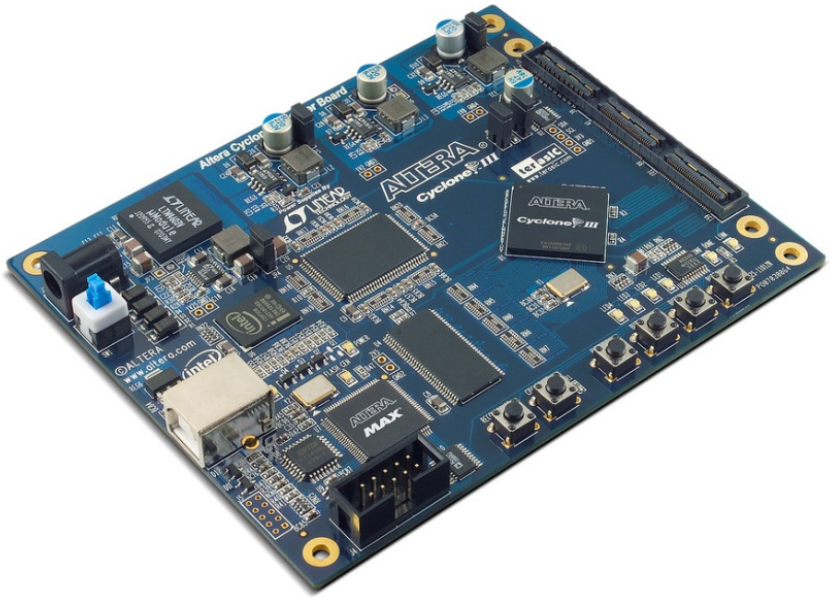
EVROPSKÁ UNIE

Praha & EU: Investujeme do vaší budoucnosti

Jazyk VHDL – zápis čísel, znaků a řetězců

1. část přednášky

Kurz A0B38FPGA – Aplikace hradlových polí



Zápis čísel v jazyku VHDL

- **Defaultní** reprezentace čísel je **dekadická (základ 10)**
- Existují tyto dva číselné formáty
  - **Integer** (příklad: 12 3 10E6 1e2) – celočíselný bez desetinných míst , tzv. fixpoint formát
  - **Real** (příklad: 1.2 25.1 3.14e-2) – reálný s desetinou řádkou, tzv. floating point formát
- Záporná čísla (např.: -5) – kombinace **operátoru negace** a daného celočíselné, resp. reálného **literálu**
- Existuje možnost vyjádřit čísla v jiné číselné soustavě – musí se použít následující syntaxe:

základ#číslo#

Příklady:

2#111010110#

16#FD#

16#0FD#

8#0235#

Příklady zápisu čísel v jiných číselných soustavách než-li desítkových

Příklady:

základ 2 : 2#001001# dekadický ekvivalent = 9

základ 16 : 16#10# dekadický ekvivalent =16

Zápis čísla 0,5 při základu deset v různých číselných soustavách:

2#0.100 8#0.4 12#0.6#

Pro vyjádření zejména dlouhých „čísel“ je možné využít symbol **podtržítko** pro zlepšení čitelnosti kódu

Příklady:

2#0010_0010_1110_0000_1122#

236_789

Znaky, řetězce, bitové řetězce v jazyku VHDL

- Zápis **znakového literálu** ve VHDL jazyce:
  - vložit znak do apostrofů (příklad: ‘a’, ‘H’, ‘;’)

- Zápis **stringového literálu** ve VHDL jazyce:
  - vložit string do uvozovek (examples: ”I am string”, “H”, “,”)

- **Bitový string –** reprezentuje posloupnost hodnot bitů – opět možnost definovat **základ** číselného stringu

Příklady:

B”110011_110011” - binární číslo
X”A0FF0C” - hexadecimální číslo
O”123” - oktalové číslo

Znaky, řetězce, bitové řetězce v jazyku VHDL

- **Bitový string s pevně definovanou délkou** - exaktní definice počtu bitů

Příklady:

7X“3C“ …. Ekvivalentní zápisu B“0111100“
8O“5“ ….. Ekvivalentní zápisu B“00000101“

- **Bitový string se znaménkem** – suffix SB, SO, SX

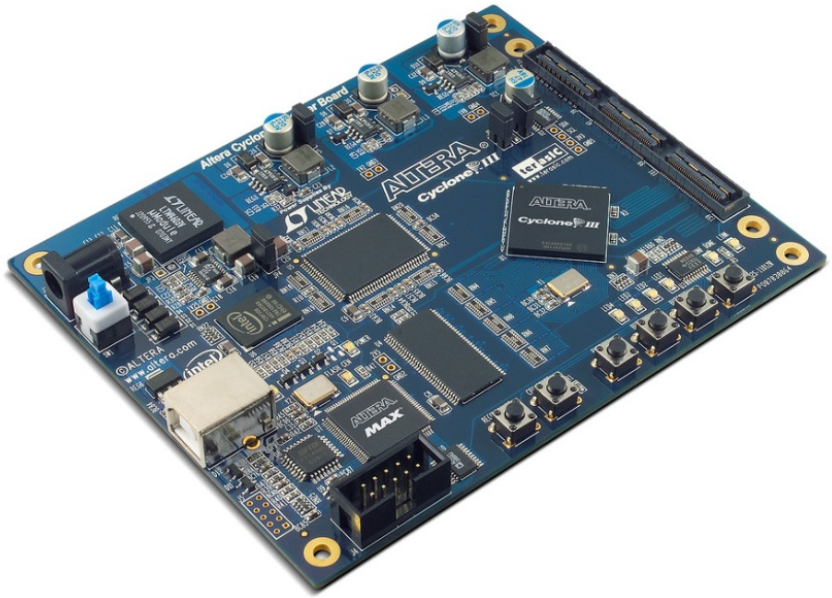
Příklady:

6SX“16“ …. Ekvivalentní zápisu B“010110“
6SX“E7“ ….. Ekvivalentní zápisu B“100111“ – pozor, dojde k odříznutí dvou nejvyšší bitů !!!

Jazyk VHDL - základní datové typy a operátory

2. část přednášky

Kurz A0B38FPGA – Aplikace hradlových polí



Datové typy

- **Skalární** (výčtové, celočíselné, s pohyblivou řádovou čárkou)
- **Složené** (pole jednorozměrné, vícerozměrné, typ záznam)

- Řada datových typů - definována v rámci **knihoven STD** či **STD_LOGIC_1164**
- Součástí definice daného datového typu jsou i **základní logické** či **relační operace** nad těmito dat. typy

Definované datové typy – knihovna STD

- **bit** - hodnoty ‘0’ a ‘1’
  Příklad: constant A:bit:=1;

- **bit_vector** - pole, kde každý element představuje prvek typu bit
  Příklad: signal IN: bit_vector(8 downto 0);

- **boolean** - hodnoty FALSE,TRUE
  Příklad: variable TEST: boolean:=FALSE;

- **character** –jakýkoliv povolené znaky v jazyce VHDL
  Příklad: variable chr: character :=’Y’;

- **integer –** rozsah čísel je dán konkrétní implementací, přesto minimální rozsah musí být 32 bitů: $-(2^{31}-1)$ to $+(2^{31}-1)$
  Příklad: constant CONST1: integer :=129;

Definované datové typy – knihovna STD

- **natural** - integer začínající hodnotou 0 až po max. hodnotou danou při deklaraci
  Příklad: variable VAR1: natural :=2;

- **positive** - integer začínající hodnotou 1 až max. hodnotou danou při deklaraci
  Příklad: variable VAR2: positive :=2;

- **real** – floating point číslo v rozsahu -1.0×10^{38} to $+1.0 \times 10^{38}$ (rozsah opět může být závislý na konkrétní implementaci)
  Příklad: variable VAR3: real :=+64.2E12;

- **string** – pole, kde každý prvek představuje 1 znak
  Příklad: variable VAR4: string(1 to 6):= “@$#ABC”;

- **time** – celočíselný formát, kde kromě hodnoty jsou definovány též i jednotky vyjádřené např. sec, ms, us, ns, ps, fs, min, hr.
  Příklad: variable DELAY: time :=5 ns;

Definované datové typ STD_ULOGIC – knihovna STD_logic_1164

- Díky této knihovně VHDL obsahuje datový typ **std_ulogic** (standard unresolved logic)
- Může nabývat celkem devíti hodnot v tomto pořadí:
  - 'U' – uninitialized*
  - 'X' - strong drive, unknown logic value*
  - '0' - strong drive, logic zero
  - '1' - strong drive, logic one
  - 'Z' - high impedance *
  - 'W' - weak drive, unknown logic value*
  - 'L' - weak drive, logic zero
  - 'H' - weak drive, logic one
  - '-' - don't care*

*lze použít pouze pro simulační účely, vše ostatní též i pro syntézu

- Zavádí též datové typy **std_logic**, resp. **std_logic_vector** – jsou podtypem **std_ulogic**
- Obsahuje též **dvě konverzní funce: To_bit, To_StdUlogic**

Použití knihovny v projektu

- Pro použití knihovny **STD** do projektu - nutno přidat tyto dva následují řádky

```
library std;
use std.standard.all;
```

- Pro použití knihovny **STD_logic_1164** do projektu - nutno přidat tyto dva následují řádky

```
library ieee;
use ieee.std_logic_1164.all;
```

Uživatelem definované datové typy

Syntaxe:

type identifikátor **is** datový typ identifikátoru;

Příklady:

type small_int **is range** 0 **to** 1024**;**

type my_word_length **is range** 31 **downto** 0**;**

subtype data_word **is** my_word_length **range** 7 **downto** 0;

subtype int_small **is** integer **range** -1024 **to** +1024;

Výčtové typy

Syntaxe:

type *jméno typu* **is** (*seznam identifikátorů či znakových literálů*);

Příklady:

type my_3values **is** (‘0’, ‘1’, ‘Z’);
type PC_OPER **is** (load, store, add, sub, div, mult, shiftl, shiftr);
type hex_digit **is** (‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, 8’, ‘9’, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’);
type state_type **is** (S0, S1, S2, S3);

Deklarace typu package

Package - umožňuje vytvořit vlastní knihovnu, která definuje uživatelské datové typy

Příklady:

```
package my_types is
    type small_int is range 0 to 1024;
    type my_word_length is range 31 downto 0;
    subtype data_word is my_word_length is range 7 downto 0;
    type cmos_level is range 0.0 to 3.3;
end package my_types;
```

Jednorozměrné pole

složený datový typ – kromě **rozsahu** je nutné přidat informaci o **typu jednotlivých prvků**

Syntaxe:

type *jméno pole* **is array** (*schéma indexace)* **of** *typ elementu pole*;

Příklady:

type MY_WORD **is array** (15 **downto** 0) **of** std_logic;
type YOUR_WORD **is array** (0 **to** 15) **of** std_logic;
type VAR **is array (**0 **to** 7) **of** integer**;**
type STD_LOGIC_1D **is array (**std_ulogic) **of** std_logic**;**

Vícerozměrná pole

```
type MY_MATRIX_3X2 is array (1 to 3, 1 to 2) of natural;
type YOUR_MATRIX_4X2 is array (1 to 4, 1 to 2) of integer;
type STD_LOGIC_2D is array (std_ulogic, std_ulogic) of std_logic;
```

Příklady použití výše definovaného typu pole:

```
variable DATA_ARR: MY_MATRIX_3X2 :=((0,2), (4,6), (5,7));
variable DATA_ARR: YOUR_MATRIX_4X2 :=((0,2), (1,3), (4,6), (5,7));
```

Předem nespecifikované (unconstrained) pole

Syntaxe:

type *jméno pole* **is array** (*typ* **range <>) of** *typ elementu*;

Příklady:

type MATRIX **is array** (integer **range** <>) of integer;
type VECTOR_INT **is array** (natural **range** <>) of integer;
type VECTOR2 **is array** (natural **range** <>, natural **range** <>) of std_logic;

Datový typ Record

Syntaxe:

```
type name is record
        identifikátor : dat. typ;
                        :
        identifikátor : dat. typ;
          end record;
```

Příklady:

```
type MY_MODULE  is
      record
         RISE_TIME          : time;
         FALL_TIME          : time;
         SIZE               : integer range 0 to 200;
         DATA               : bit_vector (15 downto 0);
      end record;

      signal modul: MY_MODUL;
      modul.RISE_TIME<= 20ns;    --přístup na položku záznamu
```

Typová konverze

- VHDL silně **typově** orientovaný jazyk !!!
- není možné přiřadit hodnotu signálu jednoho datového typu signálu jiného datového typu
- Proto existují tzv. konverzní funkce – viz. např. knihovna **std_logic_1164** - umožňuje realizovat následující konverze datových typů **std_ulogic**, **std_logic**, **std_logic_vector** na bit, bit_vektor apod., případně opačnou konverzi.

Konverzní funkce v knihovně std_logic_1164

- function **to_bit**(arg: std_ulogic) ret. to_bit
- function **to_bitvector**(arg: std_logic_vector) ret. bit_vector
- function **to_bitvector**(*arg:* std_ulogic_vector) ret. to bit_vector
- function **to_StdULogic**(*arg:* bit*) ret.* std_ulogic
- function **to_StdLogicVector**(arg: bit_vector) ret. std_logic_vector
- function **to_StdUlogicVector**(*arg:* bit_vector) ret. std_logic_vector
- function **to_StdLogicVector**(arg: std_ulogic) ret. std_logic_vector
- function **to_StdUlogicVector**(*arg:* std_logic) ret. std_ulogic_vector

Konverzní funkce v knihovně std_logic_arith

- function **CONV_INTEGER** (ARG: INTEGER) ret. INTEGER;
- function **CONV_INTEGER** (ARG: UNSIGNED) ret. INTEGER;
- function **CONV_INTEGER** (ARG: SIGNED) ret. INTEGER;
- function **CONV_INTEGER** (ARG: STD_ULOGIC) return SMALL_INT;

- function **CONV_UNSIGNED** (ARG: INTEGER; SIZE: INTEGER) ret. UNSIGNED;
- function **CONV_UNSIGNED** (ARG: UNSIGNED; SIZE: INTEGER) ret. UNSIGNED;
- function **CONV_UNSIGNED** (ARG: SIGNED; SIZE: INTEGER) ret. UNSIGNED;
- function **CONV_UNSIGNED** (ARG: STD_ULOGIC; SIZE: INTEGER) ret. UNSIGNED;

- function **CONV_SIGNED** (ARG: INTEGER; SIZE: INTEGER) ret. SIGNED;
- function **CONV_SIGNED** (ARG: UNSIGNED; SIZE: INTEGER) ret. SIGNED;
- function **CONV_SIGNED** (ARG: SIGNED; SIZE: INTEGER) ret. SIGNED;
- function **CONV_SIGNED** (ARG: STD_ULOGIC; SIZE: INTEGER) ret. SIGNED;

Konverzní funkce v knihovně std_logic_arith

- function **CONV_STD_LOGIC_VECTOR**(ARG: INTEGER; SIZE: INTEGER)
  ret. STD_LOGIC_VECTOR;

- function **CONV_STD_LOGIC_VECTOR**(ARG: UNSIGNED; SIZE: INTEGER)
  ret. STD_LOGIC_VECTOR;

- function **CONV_STD_LOGIC_VECTOR**(ARG: SIGNED; SIZE: INTEGER)
  ret. STD_LOGIC_VECTOR;

- function **CONV_STD_LOGIC_VECTOR**(ARG: STD_ULOGIC; SIZE: INTEGER)
  ret. STD_LOGIC_VECTOR;

Operátory v jazyce VHDL

- Logické
- Relační
- Posuny
- Doplňkové
- Unární
- Násobící
- Různé jiné …

Logické operátory

- **AND, OR, NAND, NOR, XOR, XNOR**
  - Všechny definovány pro dat. typ : bit, boolean, std_logic a std_ulogic
  - Lze aplikovat pro signály, proměnné i konstanty

- **NAND** a **NOR** – nejsou asociativní – pozor na vyhodnocení výrazů – nutno korektně „závorkovat“

X nand Y nand Z - špatná syntaxe

(X nand Y) nand Z - správná syntaxe

Relační operátory

=	rovno (pro jakýkoliv dat. typ)
/=	nerovná se (pro jakýkoliv dat. typ)
<	menší než (pro skalární dat.typ či diskrétní pole)
<=	menší než nebo rovno (pro skalární dat.typ či diskrétní pole)
>	větší než (pro skalární dat.typ či diskrétní pole)
<=	menší než nebo rovno (pro skalární dat.typ či diskrétní pole)

Výstupní hodnota těchto operací : **std_logic** (L nebo H)

Použití relačních operátorů

Příklady:

```
variable STS                : Boolean;
constant A                  : integer :=24;
constant B_COUNT            : integer :=32;
constant C                  : integer :=14;

STS <= (A < B_COUNT) ;
        -- bude přiřazena hodnota “TRUE” do proměnné STS

STS <=  ((A >= B_COUNT) or (A > C));
        -- bude přiřazena hodnota “TRUE” do proměnné STS

STS <=  (std_logic (‘1’, ‘0’, ‘1’) < std_logic(‘0’, ‘1’,’1’));
        -- bude přiřazena hodnota “FALSE” do proměnné STS
```

Operátory posunu

- Logické posuny
  - **SLL** - **S**hift **L**eft **L**ogical (zprava se doplňují nuly)
  - **SRL** - Shift **R**ight **L**ogical (zleva se doplňují nuly)

- Aritmetické posuny
  - **SLA** - **S**hift **L**eft **A**rithmetic (zprava se kopíruje hodnota nejnižšího bitu)
  - **SRA** - **S**hift **R**ight **A**rithmetic (zleva se kopíruje hodnota nejnižšího bitu)

- Kruhové posuny (rotace)
  - **ROL** - **RO**tate **L**eft (nejvyšší bit se přesouvá na pozici nejnižšího bitu)
  - **ROR** - **RO**tate **R**ight (nejnižší bit se přesouvá na pozici nejvyššího bitu)

Použití operátorů posunu

Příklady:

variable A: bit_vector :=”101001”;

A **sll** 2 výsledek operace: “100100”

A **srl** 2 výsledek operace: “001010”

A **sla** 2 výsledek operace: “100111”

A **sra** 2 výsledek operace: “111010”

A **rol** 2 výsledek operace: “100110”

A **ror** 2 výsledek operace: “011010”

- Tyto operace nejsou definovány nad datovými typy **std_logic_vector, std_ulogic_vector !!!**
- Lze je provést například s použitím operátoru **konketanace** neboli zřetězení– symbol &

Příklady:

variable A: std_logic_vector(6 downto 0 :=”101001”;

A := A(0) & A(6 downto 1); --cykliká rotace vlevo

Doplňkové operátory a jejich použití

+ Operátor sčítání

- Operátor odčítání

& Operátor zřetězení

Příklady:

```
signal MYBUS          : std_logic_vector (15 downto 0);
signal STATUS         : std_logic_vector (2 downto 0);
signal RW, CS1, CS2   : std_logic;
signal MDATA          : std_logic_vector ( 0 to 9);

MYBUS <= STATUS  &  RW  &  CS1  &  SC2 & MDATA;
```

Unární operátory

- Unární operátor slouží pro definici znaménka daného čísla

symbol + značí operátor tzv. identity

symbol - značí operátor tzv. negace

Operátory pro násobení

- Zajišťují provádějí operací násobení a dělení nad číselnými formáty **integer** a **real**

  symbol * - operátor násobení

  symbol / - operátor dělení

  rem - operátor remainder – zbytek po celočíselném dělení

  mod - operátor modulo – zbytek po celočíselném dělení

  A **rem** B = A - (A/B)*B (A/B je celé číslo - integer) – znaménko podle A, abs.hodnota je menší než B

  A **mod** B = A - (B*N) (N je celé číslo - integer) – znaménko podle B

Příklad:

11 **rem** 5	Výsledek : 1
(-11) **rem** 5	Výsledek : -1
9 **mod** 4	Výsledek : 1 (9-4*2)
7 **mod** (-4)	Výsledek : - 1 (7 – 4*2 = -1)

Další aritmetické operátory

** exponenciální (jakýkoliv numerický typ)

abs absolutní hodnota (jakýkoliv numerický typ)

not logická negace (jakýkoliv bit nebo typ boolean)

Příklady:

```
signal a,b :std_logic;
signal Y, Z: integer;

Y <= 2**3;    -- výsledek 8
Z<= abs (Y)
a= not b;
```

Děkuji za pozornost, máte nějaké otázky?