ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROMECHANIKY A VÝKONOVÉ ELEKTRONIKY

BAKALÁŘSKÁ PRÁCE

Generátor stimulačních VHDL souborů

Ondřej Mareček **2017**

Originál (kopie) zadání BP/DP

Abstrakt

Předkládaná bakalářská práce je zaměřena na vytvoření aplikace pro platformu Windows, která má za cíl zjednodušit vytváření stimulačních VHDL souborů pro účely testování.

Klíčová slova

VHDL, aplikace pro Windows, stimulační soubor

Abstract

The translated bachelor thesis is focused on a creation of an application for windows platform. It's goal is to make the creation of stimulating VHDL files for the testing purpose easier.

Key words

VHDL, an application for windows, a stimulating file

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

Podpis

V Plzni dne 1.6.2017 Ondřej Mareček

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Petrovi Burianovi, Ph.D. za profesionální rady, připomínky a metodické vedení práce.

Obsah

Úvod

Předkládaná bakalářská práce je zaměřena na vytvoření aplikace pro platformu Windows, která má za cíl zjednodušit vytváření stimulačních VHDL souborů pro účely testování.

Text je rozdělen do tří částí. První se zabývá VHDL jazykem, druhá uvádí informace o částech aplikace. Třetí část je věnována návodu aplikace.

Seznam symbolů a zkratek

VHSIC ..... Very High Speed Integrated Circuits, projekt ministerstva obrany U. S.

VHDL ..... VHSIC Hardware Description Language, jazyk pro popis hardware

IEEE ..... Institute of Electrical and Electronic Engineers, standardizační komise

abs ..... Absolutní hodnota

not ..... Invertor

and ..... Logický součet

nand ..... Negace logického součtu

or ..... Logický součet disjunkce

xor ..... Exkluzivní logický součet

xnor ..... Negace exkluzivního součtu

sll ..... Logický posuv vlevo

srl ..... Logický posuv vpravo

sla ..... Aritmetický posuv vlevo

sra ..... Aritmetický posuv vpravo

rol ..... Rotace vlevo

ror ..... Rotace vpravo

mod ..... Operátor dělení modulo

rem ..... Zbytek po dělení

1 VHDL

VHDL je jazyk vysoké úrovně navržený speciálně pro účely popisu a simulace velmi rozsáhlých číslicových obvodů a systémů.

1.1 Historie jazyka VHDL

Vývoj jazyka VHDL byl zahájen v roce 1981 v projektu VHSIC ministerstva obrany Spojených státu amerických. Projekt měl vyřešit efektivní popis velmi rozsáhlých integrovaných obvodů, neboť doposud byli řešeny za pomoci různých vzájemně nekompatibilních jazyků a nástrojů. Vlastní vývoj byl zadán firmám v roce 1983, které se účastnili projektu VHSIC. Jednalo se o firmy Intermetrics, Texas Instruments a IBM, které v letech 1983 až 1985 vytvořili základ definice jazyka VHDL. V roce 1985 byl poprvé veřejně publikován. Syntaxe jazyka VHDL vycházela z jazyka ADA. Označení VHDL vzniklo jako akronym z názvu VHSIC Hardware Description Language. V roce 1987 byl organizací IEEE poprvé publikován standart jazyka VHDL. V literatuře označován jako VHDL – 87.

1.2 Historie jazyka VHDL v letopočtech

- *1981: zahájení vývoje jazyka, projekt VHSIC U.S. DoD*
- *1983 – 1985: vývoj základů jazyka firmami*
- *1986: postoupení jazyka organizaci IEEE*
- *1987: publikování standardu IEEE Std 1076 – 1987 (VHDL – 87)*
- *1994: publikování standardu IEEE Std 1076 – 1993 (VHDL – 93)*
- *1999: publikování standardu IEEE Std 1076.1 – 1999 (VHDL – AMS)*
- *2000: publikování standardu IEEE Std 1076 – 2000 (VHDL – 2000)*
- *2002: publikování standardu IEEE Std 1076 – 2002 (VHDL – 2002)*

1.3 Komentáře a identifikátory

Jazyk VHDL nerozlišuje malá a velká písmena.

Komentář začíná dvěma pomlčkami a končí na konci řádku, nelze pokračovat na dalším řádku.

Identifikátory slouží k pojmenování typů, proměnných, signálů a dalších objektů. Identifikátory se mohou skládat pouze z písmen, číslic a podtržítka.

1.4 Zápis čísel, znaků a řetězců

Čísla lze zapisovat v několika soustavách např. desítkové, dvojkové, osmičkové a šestnáctkové. Rozlišujeme čísla celá a čísla reálná. Podtržítko mezi čísly nemá žádný vliv na hodnotu čísla, proto je vhodné ho použít např. pro oddělení tisíců. Tečka označuje desetinou čárku a E označuje exponent. Exponent nesmí být záporný.

Znaky se zapisují mezi apostrofy.

Textové řetězce zapisujeme mezi uvozovky. Řetězce lze spojovat operátorem **&**.

Příklad: “VHDL je urceny“ & “pro popis hardware“ -- řetězec o 33 znacích

1.5 Entita, architektura

Entita popisuje rozhraní (vstupy a výstupy) objektu. Entitu lze přirovnat ke schematické značce pojmenovávající vstupy a výstupy, definuje jejich typ a směr přenosu dat.

Entita obsahuje sekci generic, která definuje generické konstanty entity. Dále sekci port, ta definuje porty (vstupy a výstupy). Port musí mít nadefinováno své jméno, režim přenosu a typ dat. Režimy přenosu jsou: in, out, inout a buffer.

- *In – vstupní port, data mohou pouze vstupovat*
- *Out – výstupní port, data mohou pouze vystupovat a port je buzen z vnitřku entity*
- *Inout – vstupně výstupní port, požívá se nejčastěji pro obousměrné datové sběrnice*
- *Buffer – výstupní port, vlastní entita může číst data i zpětně, požívá se pro výstupy čítačů*

Architektura popisuje vlastní chování a funkci entity. Architektura definuje vnitřek entity. Každá entita musí mít alespoň jednu architekturu.

1.6 Základní datové typy

Skalární typy: výčtový, celočíselný, fyzický, s plovoucí řádovou čárkou

Kompozitní typy: pole, záznam

Výčtový typ patří mezi diskrétní skalární typy. Protože je to skalární typ, jsou pro něj definovány veškeré relační operátory. Díky diskrétnímu typu můžeme určit předchozí i následující hodnotu.

Celočíselný typ musí mít minimálně 32bitovu implementaci. Celočíselný typ vždy musí podporovat minimální rozsah $-(2^{31}-1)$ až $+(2^{31}-1)$.

Fyzický typ je určen k vyjádření množství vztaženého k základní jednotce. Hodnota je celočíselný násobek základní jednotky fyzického typu. Musí podporovat minimální rozsah $-(2^{31}-1)$ až $+(2^{31}-1)$.

Typ s plovoucí řádovou čárkou definuje rozsah hodnot, které může tento typ nabývat. Vlastní implementace je závislá na výrobci simulátoru nebo syntezátoru. Současné syntezátory většinou neumí provést syntézu matematických operací s plovoucí řádovou čárkou do hardware.

Typ pole patří mezi kompozitní datové typy (skládá se z více prvků). Skládá se z více stejných prvků. VHDL podporuje vícerozměrná pole.

Typ záznam patří stejně jako pole mezi kompozitní datové typy. Na rozdíl od typu pole může záznam obsahovat prvky různých typů (záznam je heterogenní datový typ).

1.7 Operátory

Operátory rozdělujeme do několika tříd. Nejnižší prioritu mají logické operátory a nejvyšší prioritu pak má operátor mocniny (absolutní hodnota **abs** a logický operátor **not**). Znak L značí levý operand, znak P značí pravý operand.

Logické operátory (L **and** P, L **or** P, L **nand** P, L **not** P, L **xor** P, L **xnor** P).

Relační operátory (L = P, L /= P, L < P, L <= P, L > P, L >= P). Operátor rovnosti a další operátory vrací booleovskou hodnotu **true**, pokud jsou oba shodné, jinak vrací hodnotu **false**.

Operátory posuvu (L **sll** P, L **srl** P, L **sla** P, L **sra** P, L **rol** P, L **ror** P), jsou definovány pro jednorozměrná pole. Levý operátor musí být jednorozměrné pole typu **bit** nebo **boolean** a pravý operátor musí být celočíselný typ **integer**.

- *L **sll** P – provádí logický posuv L vlevo o P pozic*
- *L **srl** P – provádí logický posuv L vpravo o P pozic*
- *L **sla** P – provádí aritmetický posul L vlevo o P pozic*
- *L **sra** P – provádí aritmetický posuv L vpravo o P pozic*
- *L **rol** P – provádí rotaci L vlevo o P pozic*
- *L **ror** P – provádí rotaci L vpravo o P pozic*

Sčítací operátory a operátor spojení (L + P, L – P, L **&** P) musí být stejného typu.

Znaménkové operátory (+P, –P) jsou definované pro celočíselné typy, mají normální matematický význam (funkci identity a negace).

Násobící operátory (L * P, L / P, L **mod** P, L **rem** P). Operátory násobení a dělení jsou definovány pro celočíselný typ a typ s plovoucí řádovou čárkou. Operátory dělení modulo (mod) a zbytek po dělení (rem) jsou definovány pouze pro celočíselné typy. Nemůžeme násobit dva fyzické typy, jelikož by nám vznikal nový typ fyzický na druhou a to nelze. Při dělení musí být použit levý operand fyzického typu.

Různé operátory (L ** P, **abs** P, **not** P). Operátor mocniny vyžaduje jako levý operand celočíselný typ nebo typ s plovoucí řádovou čárkou, pravý musí být typu **integer**.
Exponent může nabývat záporné hodnoty pouze, když je levý operand typem s plovoucí řádovou čárkou.

1.8 Základní objekty

Základní objekty jsou konstanty, signály, proměnné, aliasy. Objekty definujeme v deklaračních částech architektury. Každému objektu lze rovnou při deklaraci přiřadit implicitní hodnotu.

Konstantu nelze dále v kódu měnit. Musí se vyskytovat pouze na pravé straně.

Signály slouží k přenosu dat mezi jednotlivými komponenty, jejich funkci lze přirovnat k funkci vodičů v číslicovém systému. Přiřazení do signálu se provádí pomocí symbolu <= za nímž následuje výraz. Implicitní přiřazení do signálu se provádí pomocí symbolu := podobně jako u proměnné.

Proměnné máme sdílené a nesdílené, liší se v deklaraci. Nesdílená se deklaruje uvnitř procesu a sdílené v deklarační části architektury. Přiřazovat do proměnné lze pouze uvnitř procesu a při jejich deklaraci.

Alias není objekt, ale pouze alternativní identifikátor již existujícího objektu nebo jeho části. Umožňuje zjednodušit zápis kódu a zpřehlednit ho.

1.9 Paralelní příkazy

Všechny paralelní příkazy se provádějí současně, na pořadí příkazů v kódu nezáleží. Mezi paralelní příkazy patří:

- *Nepodmíněné přiřazení*
- *Podmíněné přiřazení*
- *Výběrové přiřazení*
- *Příkaz proces*
- *Generující příkaz*
- *Použití komponenty*
- *Volání procedury*
- *Příkaz bloku*
- *Příkaz assert*

Nepodmíněné přiřazení udává přímou hodnotu nebo výraz do signálu. Můžeme požít volitelně prefixové a postfixové příkazy, pro určení charakteru a velikosti zpoždění. Pokud není požitá žádná volitelná část nebo je prefixová část interval, tak má vždy charakter inerčního (setrvačného) zpoždění.

Podmíněným přiřazením můžeme přiřadit přímou hodnotu nebo výrazu do signálu při splnění booleovské podmínky (má-li podmínka hodnotu **true**). Podmíněná přiřazení lze řetězit. Stejně jako u nepodmíněného přiřazení můžeme použit volitelné příkazy pro zpoždění. Pokud je nepoužijeme, pak je podmíněné přiřazení syntetizovatelné. Podmíněné přiřazení lze výhodně použít pro jednoduché povolání signálu nebo skupiny signálů a dalších jednoduchých obvodů.

Výběrové přiřazení umožňuje přiřazení přímé hodnoty nebo výrazu do signálu na základě hodnoty jiného signálu. Můžeme použít volitelné příkazy pro zpoždění.

Příkaz proces se chová jako paralelní příkaz. Procesy lze použít pro realizaci klopných obvodů a mnoha dalších sekvenčních obvodů. Každý proces je spouštěn automaticky na začátku simulace. Další spuštění je možné dvěma způsoby:

- *Proces lze spustit na základě signálu uvedených v **citlivostním seznamu**. Citlivostní seznam signálů je uveden v závorkách za klíčovým slovem proces. Pokud dojde ke změně hodnoty signálu, bude proces spuštěn. Nesmí obsahovat příkaz **wait.***
- *Pokud není proces v citlivostním seznamu, tak lze řídit pomocí příkazů **wait.** Jakmile proces bez citlivostního seznamu skončí, okamžitě se znovu spouští. Pokud nechceme proces znovu spouštět, musíme použít příkaz **wait** (bez parametru).*

Generující příkaz lze použít pro úsporný popis rovnic v rekurentním zápisu. Pro popis struktur, které se opakují a jsou mezi sebou propojeny tak, že je lze popsat pomocí cyklů (např. hradla s mnoha vstupy, sčítačky, násobičky, atd.). Bez požití **generate** by musel VHDL kód při šíře slova 64 bitu obsahovat 63 rovnic, v každé rovnici jeden **and**, nebo jednu dlouhou rovnici s 63 operátory **and**. Díky příkazu **generate** je kód univerzální pro libovolně široký vstupní vektor dat (musí být alespoň dvoubitový).

Jednotlivé entity lze použít jako komponenty v dalších entitách a tím lze vytvořit hierarchické struktury. Před použitím entity jako komponenty je třeba komponentu

deklarovat, deklaraci provedeme pomocí příkazu **component**. Komponentu lze použít pomocí příkazu **port map**, který může připojit na vstupy a výstup komponenty signály.

Procedury můžeme volat za pomoci identifikátoru procedury, případné parametry uvedeme v závorce. Procedura se vykonává sekvenčně.

Blok slouží k dekompozici návrhu a tím lepší čitelnosti zdrojového kódu. V bloku můžeme definovat generické konstanty, mapovat generické konstanty, definovat porty a mapovat porty.

Příkaz **assert** vyhodnotí podmínku a pokud má hodnotu **false**, provede příkaz report, který zajistí výpis výrazu na textový výstup (okno simulátoru). Příkaz lze použít jako paralelní nebo jako sekvenční příkaz.

1.10 Knihovny

Jazyk VHDL definuje dvě třídy knihoven, a to pracovní – **work** a knihovny zdrojové. Nejdůležitější knihovní balíky jsou standardizovány organizací IEEE. Všechny tyto balíky jsou sdruženy v knihovně ieee, které připojujeme pomocí příkazu „**library** ieee;“.

- *Knihovní balík std_logic_1164*
- *Knihovní balík numeric_bit a numeric_std*
- *Knihovní balík math_real*

2 Aplikace

2.1 Aplikace

Aplikaci se nepodařilo zrealizovat do funkčního stavu.

3 Návod aplikace

3.1 Návod Aplikace

Návod aplikace při funkčním stavu by měl být:

- *Nahraj soubor VHDL*
- *Určení vstupů clc (hodin) a resetů*
- *Vygenerování stimulačního VHDL souboru*

Závěr

Při výběru daného tématu bakalářské práce jsem podcenil své znalosti programování, a proto jsem danou práci nezvládl a prosím o zadání nového tématu.

Seznam literatury a informačních zdrojů

[1] PINKER, Jiří a POUPA, Martin. *Číslicové systémy a jazyk VHDL*. 1. vyd. Praha: BEN – technická literatura, 2006. ISBN 80-7300-198-5.