VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NAČÍTÁNÍ A TISK ASCII ČÍSEL V FPGA

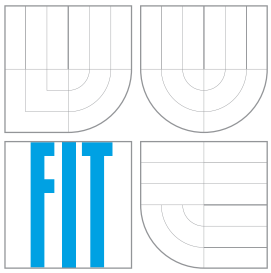
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ ZÁVODNÍK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NAČÍTÁNÍ A TISK ASCII ČÍSEL V FPGA

LOADING AND PRINTING ASCII NUMBERS IN FPGA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ ZÁVODNÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV BARTOŠ
SUPERVISOR

BRNO 2013

Abstrakt

Tématem této práce je otázka zpracování dekadických čísel binárními hardwarovými jednotkami. Použití specializovaného hardware pro tento účel je problematické zejména z důvodu nekompatibility obou číselných soustav. Práce je zaměřena konkrétně na dekadická čísla v pevné řádové čárce předávaná ve formě řetězců ASCII znaků a na technologii FPGA. Navrhovaným řešením je vytvoření hardwarových jednotek umožňujících sekvenční načítání a tisk dekadických čísel ve zmíněné podobě po jednotlivých číslicích. Náplní práce je představení vhodných algoritmů a popis realizace navrhovaných jednotek. Výsledkem je jejich efektivní, konfigurovatelná, přenositelná a znovupoužitelná implementace.

Abstract

The topic of this work is the issue of processing decimal numbers using binary hardware units. Making use of specialized hardware for this purpose is problematic due to both number systems being incompatible. The thesis is focused specifically on fixed point decimal numbers passed in the form of ASCII character strings and on the FPGA technology. The proposed solution lies in creating hardware units that allow sequential loading and printing of decimal numbers in the mentioned form digit by digit. In terms of the content of this work, it introduces suitable algorithms and describes the realization of the proposed units. It results in their efficient, configurable, portable and reusable implementation.

Klíčová slova

sekvenční převod, načítání, tisk, ASCII čísla, ASCII řetězce, dekadická čísla, dekadická reprezentace, binární čísla, binární reprezentace, celá část, desetinná část, pevná řádová čárka, doplňkový kód, double-dabble, hardware, VHDL, FPGA, FITkit

Keywords

sequential conversion, loading, printing, ASCII numbers, ASCII strings, decimal numbers, decimal representation, binary numbers, binary representation, integer part, fractional part, fixed point, complement code, double-dabble, hardware, VHDL, FPGA, FITkit

Citace

Tomáš Závodník: Načítání a tisk ASCII čísel v FPGA, bakalářská práce, Brno, FIT VUT v Brně, 2013

Načítání a tisk ASCII čísel v FPGA

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Václava Bartoše. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Závodník
10. května 2013

Poděkování

Rád bych poděkoval panu Ing. Václavu Bartošovi za znamenité vedení této bakalářské práce, cenné rady při její realizaci a veškerý čas jí věnovaný. Poděkování náleží též všem ostatním, kteří nepřímo přispěli ke zdárnému dokončení této práce.

Obsah

Kapitola 1

Úvod

Desítková soustava je dnes nejpoužívanější číselnou soustavou a její použití je pevně svázáno se všemi odvětvími lidské činnosti, je pro nás přirozená. Dekadická čísla používáme k numerickému popisu různých skutečností, s čímž souvisí i potřeba desítkové aritmetiky a prostředků pro zpracování, přenos a záznam dekadických čísel.

Pro dnešní výpočetní techniku je ale desítková soustava nepřirozená, a běžně ji proto nevyužívá. Existují i výpočetní jednotky založené na desítkové soustavě, ale jejich použití je omezené. Běžné výpočetní systémy jsou namísto desítkové soustavy založeny na vhodnější soustavě dvojkové (binární), pracují s binárními čísly a používají binární aritmetiku.

Rozdílné číselné soustavy používané lidmi a výpočetní technikou znesnadňují zpracování dekadických čísel výpočetními systémy. Dekadická čísla tak nelze zpracovávat přímo, ale je třeba je před zpracováním převést z dekadické reprezentace do binární a po zpracování z binární reprezentace do dekadické.

Na běžných počítačích zajišťuje převody mezi reprezentacemi i vlastní zpracování čísel univerzální procesor na základě vykonání předepsaného programu (software). V některých případech však nemusí být použití univerzálního procesoru a software pro tyto účely vhodné a vhodnější je použít specializovaný hardware. Příkladem zpracování dekadických čísel přímo v hardware mohou být hardwarové síťové sondy analyzující aplikační protokoly, jejichž hlavičky mohou nést dekadická čísla např. v podobě řetězců ASCII znaků. Dalším příkladem může být speciální hardware pro zpracování burzovních transakcí[1], kde mohou být zpracovávaná čísla rovněž reprezentována dekadicky v podobě ASCII řetězců. Takový hardware pak musí zajistit převod čísel mezi dekadickou a binární reprezentací a v uvedených případech umožnit načítání a tisk dekadických čísel ve formě ASCII řetězců, což je náplní této práce. Načítáním a tiskem dekadických čísel přitom v této práci rozumíme jejich sekvenční vstup a výstup po jednotlivých číslicích z/do řetězců osmibitových ASCII znaků.

Následující text je rozdělen na několik kapitol a zahrnuje formulaci účelu a cíle této práce (kapitola 2), vysvětlení použitých algoritmů (kapitola 3), popis návrhu, implementace, jejího vyhodnocení a ověření funkčnosti hardwarové jednotky pro načítání dekadických čísel z ASCII řetězců a jejich převod do binární reprezentace (kapitola 4) a jednotky pro tisk dekadických čísel do ASCII řetězců po jejich převodu z binární reprezentace (kapitola 5), popis testovací a demonstrační aplikace (kapitola 6) a závěrečné zhodnocení práce a dosažených výsledků (kapitola 7).

[1] V oblasti velmi rychlého on-line obchodování je mj. velmi důležitá rychlost reakce na burzovní nabídku či poptávku, proto se v dnešní době začínají objevovat i řešení akcelerovaná pomocí FPGA.

Kapitola 2

Účel a cíl práce

Zpracování pomocí specializovaného hardware je možné realizovat mj. prostřednictvím programovatelných hradlových polí FPGA. Jejich výhodou je možnost implementace hardwarového zpracování bez nutnosti stavby fyzických číslicových obvodů. Čip FPGA je programován na základě zdrojového textu zapsaného v jazyce pro popis hardware, jako je např. jazyk VHDL. Tyto vlastnosti umožňují relativně snadno implementovat nejrůznější výpočetní funkce. Mnohdy je navíc potřebná implementace pro FPGA (nebo jiná) již dostupná k použití nebo ji lze pomocí specializovaných nástrojů automaticky vygenerovat. Takovým nástrojem je např. Core Generator [1], který je součástí Xilinx XST a umožňuje generovat optimální implementaci různorodých výpočetních jednotek. To ale většinou platí pouze pro zpracování binárních čísel. Zpracování dekadických čísel pak vyžaduje převod mezi dekadickou a binární reprezentací, ten však zůstává problematickým.

Cílem této práce je návrh a implementace hardwarových jednotek, které poskytnou prostředky pro obousměrný převod čísel v pevné řádové čárce se znaménkem mezi dekadickou a binární reprezentací a umožní načítání a tisk dekadických čísel jinak binárním hardwarovým jednotkám.

Dekadická čísla, obsahující celou a desetinnou, nebo pouze celou část, budou předávána ve formě řetězců osmibitových ASCII znaků. ASCII znaky '0' až '9' budou představovat odpovídající dekadické číslice, ostatní znaky pak ponesou zvláštní význam, nebo budou mít funkci oddělovačů jednotlivých dekadických čísel. Mezi znaky se zvláštním významem patří znaky '+' a '−', které budou určovat znaménko bezprostředně následujícího dekadického čísla v případě, že nebudou umístěny bezprostředně za předchozím dekadickým číslem. Dalším znakem se zvláštním významem je znak '.', který bude představovat desetinnou tečku/čárku a oddělovat bezprostředně předcházející celou (neprázdnou) a bezprostředně následující desetinnou (i prázdnou) část dekadického čísla. Vyjmenované zvláštní znaky na jiných než definovaných pozicích a ostatní neuvedené znaky budou mít funkci oddělovačů. Dekadická čísla, ukončená vždy nejméně jedním oddělovačem, budou načítána a tisknuta sekvenčně po jednotlivých dekadických číslicích počínaje nejlevější.

Binární čísla, obsahující celou a zlomkovou část, nebo pouze některou z nich, budou reprezentována v doplňkovém kódu a jejich bitová šířka bude konfigurovatelná.

Implementačním prostředkem je jazyk VHDL a cílovou technologií je FPGA. Výsledné hardwarové jednotky jsou považovány především za jednotky podpůrné, které budou součástí většího a složitějšího výpočetního systému. Z tohoto důvodu je vyžadována jejich vysoká rychlost a co nejmenší požadavky na zdroje, převod musí být co nejefektivnější.

Kapitola 3

Použité algoritmy

Všechny uvedené algoritmy pracují pouze s bezznaménkovými čísly v pevné řádové čárce. Společně tvoří teoretický základ potřebný pro realizaci převodu čísel mezi dekadickou a binární reprezentací a pro implementaci hardwarového načítání a tisku dekadických čísel.

3.1 Algoritmus pro načítání celé části dekadického čísla

Při implementaci načítání celé části dekadického čísla lze využít strukturních vlastností čísel desítkové soustavy. Hodnotu A celého dekadického čísla bez znaménka je možné vypočítat pomocí vztahu

$$A = a_k * 10^k + a_{k-1} * 10^{k-1} + \ldots + a_1 * 10 + a_0 = \sum_{i=0}^{k} (a_i * 10^i) \tag{3.1}$$

kde A je hodnota daného čísla a a_i je hodnota dekadické číslice na pozici i v daném čísle, přičemž nejlevější číslici odpovídá pozice k a nejpravější pozice 0. Tuto rovnici lze pak přepsat do tvaru

$$A = ((\ldots(a_k * 10 + a_{k-1}) * 10 + \ldots) * 10 + a_1) * 10 + a_0 \tag{3.2}$$

Tento zápis ukazuje, že hodnotu celého dekadického čísla bez znaménka můžeme získat postupným násobením deseti a přičítáním hodnot obsažených číslic pořadě od té nejlevější. Načítání celé části dekadického čísla je pak možné implementovat s pomocí iterativního výpočtu rovnice

$$A_{nova} = A_{dosavadni} * 10 + a_{aktualni} \tag{3.3}$$

Obsažené násobení deseti lze navíc převést na pouhé sčítání a bitové posuny podle přepisu

$$X * 10 = (X * 8) + (X * 2) = (X \ll 3) + (X \ll 1) \tag{3.4}$$

kde X je násobené číslo a $\ll$ je bitový posun vlevo o určený počet bitů. Iterace pokračují, dokud není načítání celé části dekadického čísla dokončeno.

Popsaný algoritmus je vzhledem ke své jednoduchosti velmi vhodný pro realizaci sekvenčního převodu celé části dekadického čísla do binární reprezentace a pro implementaci jejího hardwarového načítání po jednotlivých dekadických číslicích počínaje nejlevější.

3.2 Algoritmus pro načítání desetinné části dekadického čísla

Za předpokladu, že je znám počet načítaných dekadických číslic, lze pro implementaci načítání desetinné části dekadického čísla použít postup a algoritmus uvedený v sekci 3.1. Jediným rozdílem pak je, že do výsledku nejsou započítávány přímo hodnoty načítaných dekadických číslic, ale hodnoty vynásobené konstantou 10^{-m}, kde m je počet načítaných číslic desetinné části čísla. Tím dojde k posunu desetinné čárky o m pozic doprava a desetinnou část dekadického čísla lze pak načítat stejným způsobem jako část celou.

Při převodu desetinné části dekadického čísla do binární reprezentace se naneštěstí nevyhneme zanesení chyby způsobené změnou číselné soustavy. Důvodem je existence přesně dekadicky reprezentovatelných čísel (např. $\frac{1}{10}$), která však nelze přesně reprezentovat binárně na konečném počtu bitů. Maximální absolutní chybu vzniklou převodem dle uvedeného algoritmu lze předem určit na základě vztahu

$$\Delta = \sum_{i=0}^{m-1} (e * 10^i) \tag{3.5}$$

kde Δ je celková maximální absolutní chyba převodu a e je maximální absolutní chyba převodu hodnot jednotlivých dekadických číslic vynásobených konstantou 10^{-m}, vypočítaná jako maximální absolutní hodnota rozdílu původní a převedené hodnoty. Maximální relativní chybu převodu lze vypočítat dle vztahu

$$\delta = \frac{\Delta}{|A|} \tag{3.6}$$

kde δ je celková maximální relativní chyba převodu a A ($A \neq 0$) je hodnota převáděné desetinné části dekadického čísla.

3.3 Algoritmy pro tisk celé části dekadického čísla

3.3.1 Algoritmus s využitím komparátorů

Jednou z možností implementace tisku celé části dekadického čísla je použití algoritmu založeného na využití komparátorů pro určení hodnoty tisknuté dekadické číslice. Přitom vycházíme opět ze vztahu 3.1 pro výpočet hodnoty celého dekadického čísla bez znaménka, pouze s opačným přístupem. Místo celkové hodnoty čísla je třeba určit postupně hodnoty všech obsažených dekadických číslic počínaje tou nejlevější (nejvyšším řádem). Na základě známého počtu bitů vstupního binárního čísla je možné určit hodnotu parametru k a tím také nejvyšší řád tisknutého dekadického čísla 10^k. Zbývá určit hodnotu a_k, tedy koeficient nejvyššího řádu a zároveň hodnotu nejlevější tisknuté dekadické číslice. Toho lze dosáhnout pomocí devíti komparátorů „větší nebo rovno“, které provedou porovnání hodnoty vstupního binárního čísla s možnými hodnotami nejvyššího řádu ($1*10^k$ až $9*10^k$). Po provedení komparace je hodnota nejvyššího řádu odečtena od vstupního čísla a výsledek je vynásoben deseti. Tím se dosavadní druhý nejvyšší řád stane řádem nejvyšším a může následovat další komparace pro tisk následující dekadické číslice. Výpočet je popsán vztahem

$$A_{nova} = (A_{dosavadni} - a_{aktualni} * 10^k) * 10 \tag{3.7}$$

kde A_{nova} je nová hodnota vstupního binárního čísla, určená pro další zpracování, $A_{dosavadni}$ je dosavadní hodnota vstupního čísla a $a_{aktualni}$ je hodnota aktuálně tisknuté dekadické

vstup (4 bity)	výstup (4 bity)
0000	0000
0001	0001
0010	0010
0011	0011
0100	0100
0101	1000
0110	1001
0111	1010
1000	1011
1001	1100

Tabulka 3.1: Look-up tabulka pro korekci dekadických číslic algoritmu double-dabble

číslice. Použitou operaci násobení deseti je opět možné převést na sčítání a bitové posuny podle přepisu 3.4. Výpočet je iterativně opakován, dokud není tisk celé části dekadického čísla dokončen.

Popsaný algoritmus je opět poměrně jednoduchý, vyšší složitost přináší pouze použití komparátorů, a je vhodnou variantou pro realizaci sekvenčního převodu celé části binárního čísla do dekadické reprezentace a pro implementaci hardwarového tisku celé části dekadického čísla po jednotlivých číslicích počínaje nejlevější. Výhodou tohoto algoritmu je možnost tisku dekadických číslic zleva, aniž by byl převod úplně dokončen, a relativně nízký nárůst složitosti při narůstajícím počtu bitů vstupního binárního čísla.

3.3.2 Algoritmus double-dabble

Alternativou algoritmu pro implementaci tisku celé části dekadického čísla popsaného v sekci 3.3.1 je algoritmus double-dabble [2]. Tento původně sekvenční algoritmus pro převod celých čísel z binární do dekadické reprezentace v podobě BCD kódu je založen na bitových posunech a lze jej adaptovat i na jiná kódování včetně ASCII. Číslice výstupního dekadického čísla jsou určeny čtyřbitovými bloky, které jsou seřazeny zleva doprava dle snižujícího se řádu odpovídajících číslic. Během převodu je vstupní binární číslo bit po bitu vsouváno zprava do bloků výstupních dekadických číslic, přičemž jejich hodnoty jsou zvyšovány o 3 vždy, když dosáhnou 5, aby po dalším bitovém posunu vlevo nedošlo k přesáhnutí hodnoty 9. Vzhledem k lokálnosti změn při korekci není třeba použít operaci sčítání, ale hodnoty lze upravovat na základě look-up tabulky v podobě tabulky 3.1. Převod končí po vsunutí celého vstupního binárního čísla do bloků výstupních dekadických číslic. Příklad převodu osmibitového binárního čísla bez znaménka na tři dekadické číslice v BCD kódu znázorňuje tabulka 3.2.

Popsaný algoritmus je vhodnou alternativou s možností kompromisů pro realizaci sekvenčního převodu celé části binárního čísla do dekadické reprezentace a pro implementaci hardwarového tisku celé části dekadického čísla po jednotlivých číslicích počínaje nejlevější. Nevýhodou tohoto algoritmu je, že před tiskem dekadických číslic je nutné úplné dokončení převodu. Plně sekvenční verze algoritmu double-dabble se vyznačuje velmi nízkou složitostí, ale počet cyklů nutných k dokončení převodu je úměrný počtu bitů vstupního čísla. Algoritmus lze realizovat také plně kombinační cestou, ale tato verze vykazuje velký nárůst složitosti při zvyšování počtu bitů vstupního binárního čísla. Kompromisu mezi složitostí a počtem cyklů nutných pro převod lze dosáhnout hybridní realizací algoritmu, která je čás-

stovky	desítky	jednotky	vstup	provedená operace
0000	0000	0000	11111111 (255)	inicializace
0000	0000	0001	11111110	posun o 1 bit vlevo
0000	0000	0011	11111100	posun o 1 bit vlevo
0000	0000	0111	11111000	posun o 1 bit vlevo
0000	0000	1010	11111000	korekce jednotek
0000	0001	0101	11110000	posun o 1 bit vlevo
0000	0001	1000	11110000	korekce jednotek
0000	0011	0001	11100000	posun o 1 bit vlevo
0000	0110	0011	11000000	posun o 1 bit vlevo
0000	1001	0011	11000000	korekce desítek
0001	0010	0111	10000000	posun o 1 bit vlevo
0001	0010	1010	10000000	korekce jednotek
0010 (2)	0101 (5)	0101 (5)	00000000	posun o 1 bit vlevo

Tabulka 3.2: Příklad převodu osmibitového binárního čísla bez znaménka hodnoty 255 na tři dekadické číslice v BCD kódu algoritmem double-dabble

tečně kombinační a částečně sekvenční. Hybridní verze algoritmu double-dabble umožňuje kombinačně provádět více kroků plně sekvenční verze v jednom cyklu převodu. Tím lze snížit celkový počet cyklů nutných k dokončení převodu za cenu zvýšení složitosti.

3.4 Algoritmus pro tisk desetinné části dekadického čísla

Tisk desetinné části dekadického čísla lze implementovat pomocí prosté operace násobení deseti. Vynásobíme-li desetinnou část bezznaménkového čísla deseti, vznikne číslo, jehož celá část představuje hodnotu nejlevější dekadické číslice původní desetinné části. Násobením desetinné části nově vzniklého čísla deseti lze získat hodnotu následující dekadické číslice a tak dále. Iterativním prováděním násobení vstupního binárního čísla na základě tohoto principu lze provést celý tisk desetinné části dekadického čísla. Operaci násobení deseti je opět možné převést na sčítání a bitové posuny podle přepisu 3.4.

Pro svoji jednoduchost je popsaný algoritmus velmi vhodný pro realizaci sekvenčního převodu zlomkové části binárního čísla do dekadické reprezentace a pro implementaci hardwarového tisku desetinné části dekadického čísla po jednotlivých číslicích počínaje nejlevější.

Stejně jako v případě převodu desetinné části dekadického čísla do binární reprezentace se ani při převodu opačném nevyhneme zanesení chyby způsobené změnou číselné soustavy. V tomto případě závisí velikost maximální absolutní chyby převodu Δ na počtu n tisknutých číslic desetinné části dekadického čísla a zdola se blíží hodnotě 10^{-n}. Maximální relativní chybu převodu δ lze pak vypočítat dle vztahu

$$\delta = \frac{\Delta}{|B|} \tag{3.8}$$

kde δ je maximální relativní chyba převodu a B ($B \neq 0$) je hodnota převáděné zlomkové části binárního čísla.

Kapitola 4

Jednotka pro načítání ASCII čísel

4.1 Návrh

Návrh jednotky pro načítání dekadických čísel z ASCII řetězce a jejich převod do binární reprezentace je koncipován tak, aby byl funkční základ oddělen od řídicí části. Návrh rovněž definuje rozhraní jednotky a určuje její funkci.

4.1.1 Bázová komponenta

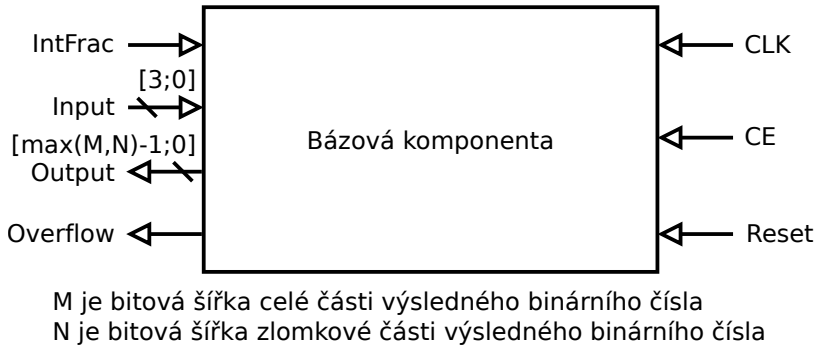
Funkční základ jednotky je realizován formou samostatné bázové komponenty, která představuje výpočetní jádro jednotky a implementuje převodní algoritmy. Bázová komponenta poskytuje jednotce prostředky pro načítání celé i desetinné části dekadického čísla. Každá z částí dekadického čísla je přitom načítána samostatně. Načítání celé části je navíc doplněno o detekci přetečení při načítání příliš dlouhého dekadického čísla a počet načítaných číslic desetinné části je pevně dán konfigurací. Komponenta je navržena pro práci pouze s bezznaménkovými čísly. To umožňuje její jednodušší implementaci a snazší samostatnou přenositelnost v případech, kdy jsou zpracovávána pouze čísla bez znaménka. Návrh rovněž definuje rozhraní a funkci bázové komponenty, vycházející z charakteristik využitých převodních algoritmů uvedených v sekcích 3.1 a 3.2.

Rozhraní

Rozhraní bázové komponenty obsahuje vstup synchronizačního signálu (`CLK`), vstup povolení synchronizačního signálu (`CE`), vstup synchronního resetu (`Reset`), vstup přepínače načítání celé a desetinné části dekadického čísla (`IntFrac`), čtyřbitový vstup načítané dekadické číslice v binární podobě BCD kódu (`Input`), výstup výstupního binárního čísla bez znaménka (`Output`) konfigurovatelné bitové šířky a výstup příznaku přetečení při načítání celé části dekadického čísla (`Overflow`). Rozhraní je znázorněno na obrázku 4.1.

Funkce

Vstup `Input` je načítán s každou nástupnou hranou synchronizačního signálu `CLK`, pokud jsou vstupy `CE` v logické 1 a `Reset` v logické 0. Načítání je možné pozastavit nastavením vstupu `CE` na logickou 0, nebo zrušit nastavením vstupu `Reset` na logickou 1. Během celého načítání musí být vstup `IntFrac` správně nastaven na logickou 0 při načítání celé části dekadického čísla, nebo na logickou 1 při načítání části desetinné. Výstup `Output` je aktualizován s každým načtením vstupu `Input`. Výstupní binární číslo se na výstupu `Output`



Obrázek 4.1: Návrh bázové komponenty pro načítání ASCII čísel a její rozhraní

objeví s načtením poslední dekadické číslice. V případě, že při načítání celé části dekadického čísla došlo k přetečení, je společně s chybným výstupem `Output` také nastaven výstup `Overflow` na logickou 1. Před každým novým načítáním je nutné provést vynulování bázové komponenty nastavením vstupu `Reset` na logickou 1.

4.1.2 Řídicí část

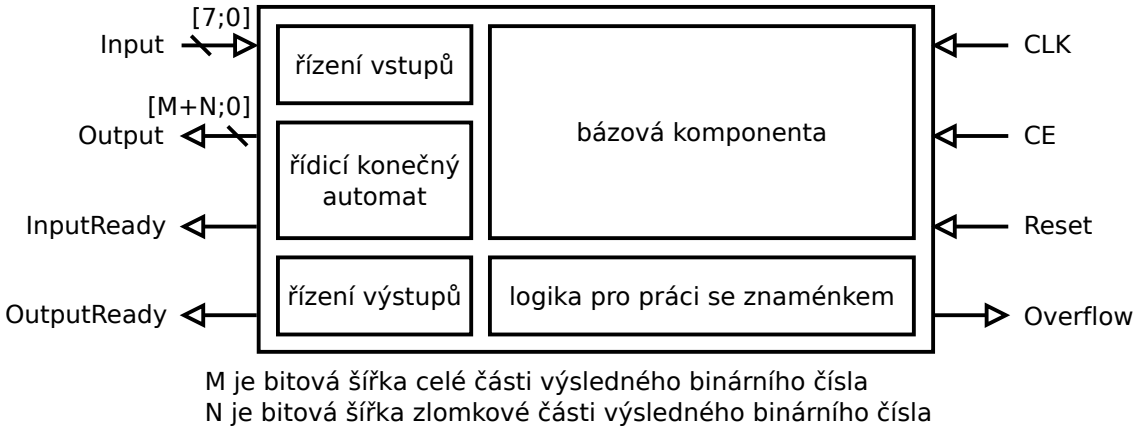
Řídicí část jednotky zapouzdřuje bázovou komponentu, poskytuje jí řízení nezbytné pro převod dekadických čísel do binární reprezentace a využívá jejích prostředků pro načítání celé a desetinné části dekadických čísel. Řízení jednotky je realizováno pomocí konečného automatu ve spojení s podpůrnými prostředky. Vzhledem k tomu, že bázová komponenta je navržena pouze pro bezznaménková čísla, poskytuje řídicí část jednotky navíc prostředky pro práci se znaménkem. Počet převáděných číslic desetinné části dekadického čísla je pevně dán konfigurací jednotky, nadbytečné číslice jsou ignorovány, chybějící nahrazovány nulami.

4.1.3 Rozhraní

Rozhraní navržené jednotky obsahuje vstup synchronizačního signálu (`CLK`), vstup povolení synchronizačního signálu (`CE`), vstup synchronního resetu (`Reset`), osmibitový vstup načítaného ASCII znaku (`Input`), výstup výsledného binárního čísla se znaménkem v doplňkovém kódu (`Output`) konfigurovatelné bitové šířky, výstup příznaku připravenosti přijmout načítaný ASCII znak (`InputReady`), výstup příznaku připravenosti předat výsledné binární číslo (`OutputReady`) a výstup příznaku přetečení (`Overflow`). Rozhraní je znázorněno na obrázku 4.2.

4.1.4 Funkce

Vstup `Input` je načítán s každou nástupnou hranou synchronizačního signálu `CLK`, pokud jsou vstupy `CE` v logické 1, `Reset` v logické 0 a výstup `InputReady` v logické 1. Převod je možné pozastavit nastavením vstupu `CE` na logickou 0, nebo zrušit nastavením vstupu `Reset` na logickou 1. Výsledné binární číslo se na výstupu `Output` objeví po dokončení převodu a zároveň s ním je nastaven výstup `OutputReady` na logickou 1. V případě, že při převodu dojde k přetečení, je nastaven výstup `Overflow` na logickou 1. Navržená jednotka je schopna načítání více dekadických čísel v řadě, oddělených oddělovači.



Obrázek 4.2: Návrh jednotky pro načítání ASCII čísel a její rozhraní

4.2 Implementace

Implementace jednotky pro načítání dekadických čísel z ASCII řetězce a jejich převod do binární reprezentace zahrnuje parametrizaci implementace, implementaci bázové komponenty a implementaci řídicí části včetně podpůrných prostředků.

4.2.1 Parametrizace

Implementace jednotky je realizována tak, aby byla parametrizovatelná a použitelná v různých konfiguracích, přitom jsou používány parametry:

`BIN_WIDTH` určující bitovou šířku celé části výsledného binárního čísla

`BIN_FRAC_WIDTH` určující bitovou šířku zlomkové části výsledného binárního čísla

`DEC_FRAC_WIDTH` určující počet převáděných číslic desetinné části dekadického čísla v případě, že je desetinná část převáděna

`PRECISION_BITS` určující počet dodatečných bitů pro zvýšení přesnosti převodu desetinné části dekadického čísla, přesnostní bity se využívají pouze interně

`FRAC_ROUND` povolující zaokrouhlení výsledného binárního čísla se zlomkovou částí

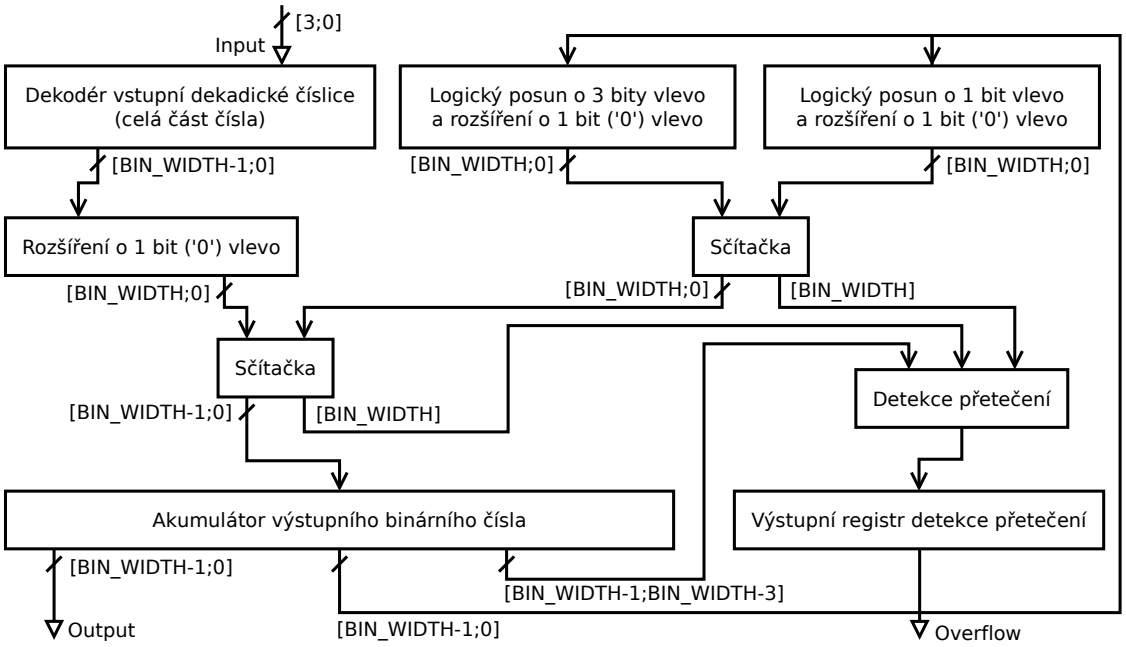
Bitová šířka výsledného binárního čísla je pak určena součtem hodnot parametrů `BIN_WIDTH`, `BIN_FRAC_WIDTH` a 1 bitu vyhrazeného pro znaménko. Zpracovatelné hodnoty jsou navíc omezeny intervalem $(-2^{\texttt{BIN\_WIDTH}}, +2^{\texttt{BIN\_WIDTH}})$.

Parametry `BIN_WIDTH` a `DEC_FRAC_WIDTH` jsou beze změny předávány bázové komponentě skrze stejnojmenné protějšky. Hodnoty parametrů `BIN_FRAC_WIDTH` a `PRECISION_BITS` jsou bázové komponentě předávány skrze její parametr `BIN_FRAC_WIDTH` jako součet zvýšený o 1 odpovídající zaokrouhlovacímu bitu. Parametr `FRAC_ROUND` bázové komponentě předáván není.

4.2.2 Bázová komponenta

Parametrizace

Implementace bázové komponenty je rovněž parametrizovatelná vlastními parametry:



Obrázek 4.3: Blokové schéma načítání celé části dekadického čísla

`BIN_WIDTH` určující bitovou šířku celé části výsledného binárního čísla

`BIN_FRAC_WIDTH` určující bitovou šířku zlomkové části výsledného binárního čísla

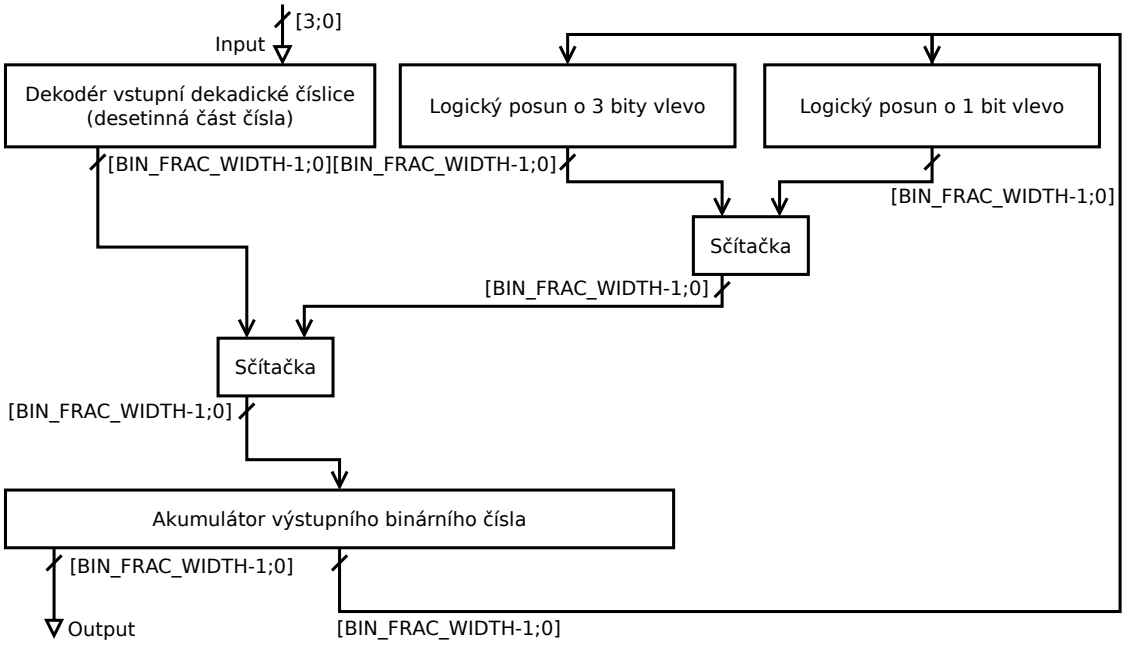
`DEC_FRAC_WIDTH` určující počet načítaných číslic desetinné části dekadického čísla

Bitová šířka výstupního binárního čísla je určena maximem z hodnot parametrů `BIN_WIDTH` a `BIN_FRAC_WIDTH`.

Načítání celé části dekadického čísla

Implementace načítání celé části dekadického čísla bázové komponenty je založena na algoritmu popsaném v sekci 3.1 ve spojení s detekcí přetečení při načítání příliš dlouhého dekadického čísla. Logická struktura implementovaného algoritmu je znázorněna na obrázku 4.3. Znázorněné schéma je platné pouze v případě, že šířka výstupního binárního čísla (zde parametr `BIN_WIDTH`) je nejméně 4 bity. V opačném případě je jakýkoli výpočet zbytečný, hodnota výstupního čísla je rovna načítané hodnotě a detekce přetečení je odvozena přímo od hodnoty akumulátoru výstupního binárního čísla a načítané hodnoty.

V každém taktu synchronizačního signálu `CLK` během načítání je zpočátku nulová hodnota v akumulátoru výstupního binárního čísla vynásobena deseti za použití bitových posunů vlevo a sčítačky podle přepisu 3.4. K výsledku je za použití další sčítačky přičtena aktuálně načítaná hodnota, odvozená pomocí dekodéru vstupní dekadické číslice ze vstupu `Input`. S následující nástupnou hranou signálu `CLK` je výsledek výpočtu uložen do akumulátoru, čímž je zároveň vystaven na výstup `Output`. Po načtení poslední dekadické číslice je hodnota uložená v akumulátoru hodnotou výslednou. Během výpočtu je navíc od nejvyšších tří bitů akumulátoru a nejvyššího kontrolního bitu obou součtů odvozena detekce přetečení. Výsledný příznak je s následující nástupnou hranou signálu `CLK` uložen do výstupního registru detekce přetečení a tím zároveň vystaven na výstup `Overflow`.



Obrázek 4.4: Blokové schéma načítání desetinné části dekadického čísla

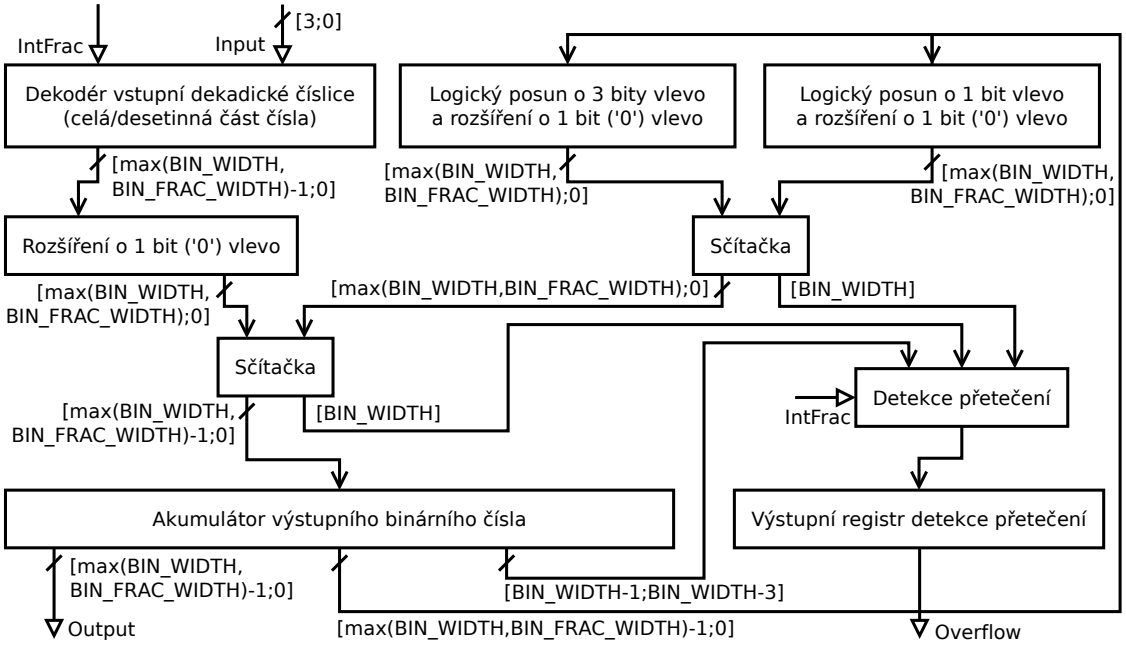
Načítání desetinné části dekadického čísla

Implementace načítání desetinné části dekadického čísla bázové komponenty je založena na algoritmu popsaném v sekci 3.2. Logická struktura implementovaného algoritmu je znázorněna na obrázku 4.4. Znázorněné schéma je platné pouze v případě, že šířka výstupního binárního čísla (zde parametr `BIN_FRAC_WIDTH`) je nejméně 4 bity. V opačném případě je jakýkoli výpočet zbytečný a hodnota výstupního čísla je rovna načítané hodnotě.

V každém taktu synchronizačního signálu `CLK` během načítání je zpočátku nulová hodnota v akumulátoru výstupního binárního čísla vynásobena deseti za použití bitových posunů vlevo a sčítačky podle přepisu 3.4. K výsledku je za použití další sčítačky přičtena aktuálně načítaná hodnota, odvozená pomocí dekodéru vstupní dekadické číslice ze vstupu `Input`. S následující nástupnou hranou signálu `CLK` je výsledek výpočtu uložen do akumulátoru, čímž je zároveň vystaven na výstup `Output`. Po načtení poslední dekadické číslice je hodnota uložená v akumulátoru hodnotou výslednou.

Kombinace načítání obou částí dekadického čísla

Vzhledem k použití, kdy je v jedné fázi načítání převáděna vždy pouze celá, nebo pouze desetinná část dekadického čísla, je bázová komponenta implementována s maximálním možným sdílením zdrojů mezi implementacemi načítání obou částí dekadického čísla. To umožňuje maximální využití implementovaného hardwaru a redukci nároků na zdroje za cenu mírného snížení rychlosti. Vzhledem k podobnosti implementací obou převodních algoritmů je pak redukce nároků na zdroje vysoká a snížení rychlosti minimální. Kombinované logické schéma implementace bázové komponenty je znázorněno na obrázku 4.5. Znázorněné schéma je platné pouze v případě, že hodnota parametru `BIN_WIDTH` je nejméně 4. V opačném případě je detekce přetečení odvozena přímo od načítané hodnoty a hodnoty akumulátoru výstupního binárního čísla. Pokud jsou hodnoty obou parametrů `BIN_WIDTH` a `BIN_FRAC_WIDTH` menší než 4, je navíc hodnota výstupního čísla rovna načítané hodnotě.



Obrázek 4.5: Blokové schéma implementace bázové komponenty pro načítání ASCII čísel

Oproti samostatným implementacím načítání celé a desetinné části dekadickáho čísla je navíc využit vstup `IntFrac`. Ten ovlivňuje výstup dekodéru vstupní dekadické číslice a detekce přetečení tak, aby bylo možné načítat obě části dekadického čísla pomocí sdílené implementace. Vlastní postup načítání přitom zůstává nezměněn.

I přes preferované sdílení zdrojů je bázová komponenta implementována tak, že v případě jejího použití pro načítání pouze celé, nebo pouze desetinné části dekadického čísla je maximum přebytečné logiky, typické pro nepoužívanou převodní část, při optimalizaci odstraněno. Výsledná implementace načítání dané části dekadického čísla pak má vlastnosti implementace samostatné.

4.2.3 Řídicí část

Podpůrné prostředky

Nejdůležitějším podpůrným prostředkem implementované jednotky je logika pro práci se znaménkem. Bázová komponenta pracuje pouze s bezznaménkovými čísly, a její výstup je proto na základě načteného znaménka nutné upravit na správnou hodnotu. V případě záporného znaménka je pak výsledné binární číslo před svým výstupem upraveno operací dvojkového doplňku do správné znaménkové reprezentace doplňkového kódu.

Druhým nejdůležitějším podpůrným prostředkem je čítač počtu převáděných číslic desetinné části dekadického čísla `CntReg` používaný v případě, že je desetinná část převáděna. Bázová komponenta je implementována pro načítání pevného počtu dekadických číslic desetinné části čísla, daného parametrem `DEC_FRAC_WIDTH`. Daný počet je nutné dodržet a s pomocí čítače případně počet načítaných dekadických číslic omezit, nebo naopak doplnit.

Třetím podpůrným prostředkem je logika pro zaokrouhlování výsledného binárního čísla se zlomkovou částí na základě dodatečného zaokrouhlovacího bitu. Tato volitelná součást jednotky, daná parametrem `FRAC_ROUND`, upřesňuje výsledek převodu dekadického čísla v případě, že je převáděna i jeho desetinná část.

Konečný automat

Řízení jednotky je implementováno pomocí konečného automatu, který na základě aktuálního stavu načítání a aktuálně načítaného ASCII znaku ze vstupu `Input` nastavuje příslušné řídicí signály. Používány jsou řídicí signály[1]:

`BaseReset` vynulování bázové komponenty

`CntReset` vynulování čítače počtu převáděných číslic desetinné části dekadického čísla

`SignReset` vynulování znaménkového registru (nastavení kladného znaménka)

`BaseEn` povolení synchronizačního signálu bázové komponenty

`CntEn` povolení inkrementace čítače počtu převáděných číslic desetinné části čísla

`SignEn` povolení záznamu znaménkového registru (nastavení záporného znaménka)

`IntEn` povolení záznamu registru výsledku převodu celé části dekadického čísla

`OutputEn` povolení záznamu výstupního registru výsledného binárního čísla

`InInputReady` vnitřní příznak připravenosti přijmout načítaný ASCII znak

`InOutputReady` vnitřní příznak připravenosti předat výsledné binární číslo

`BaseInputSel` výběr načítané dekadické číslice bázové komponenty (nejnižší čtyři bity vstupního ASCII znaku v logické 0, nebo nula v logické 1)

`IntOutputSel` výběr výsledku převodu celé části dekadického čísla (uložený v logické 0, nebo aktuální v logické 1)

`FracOutputSel` výběr výsledku převodu desetinné části dekadického čísla (nulový v logické 0, nebo aktuální v logické 1)

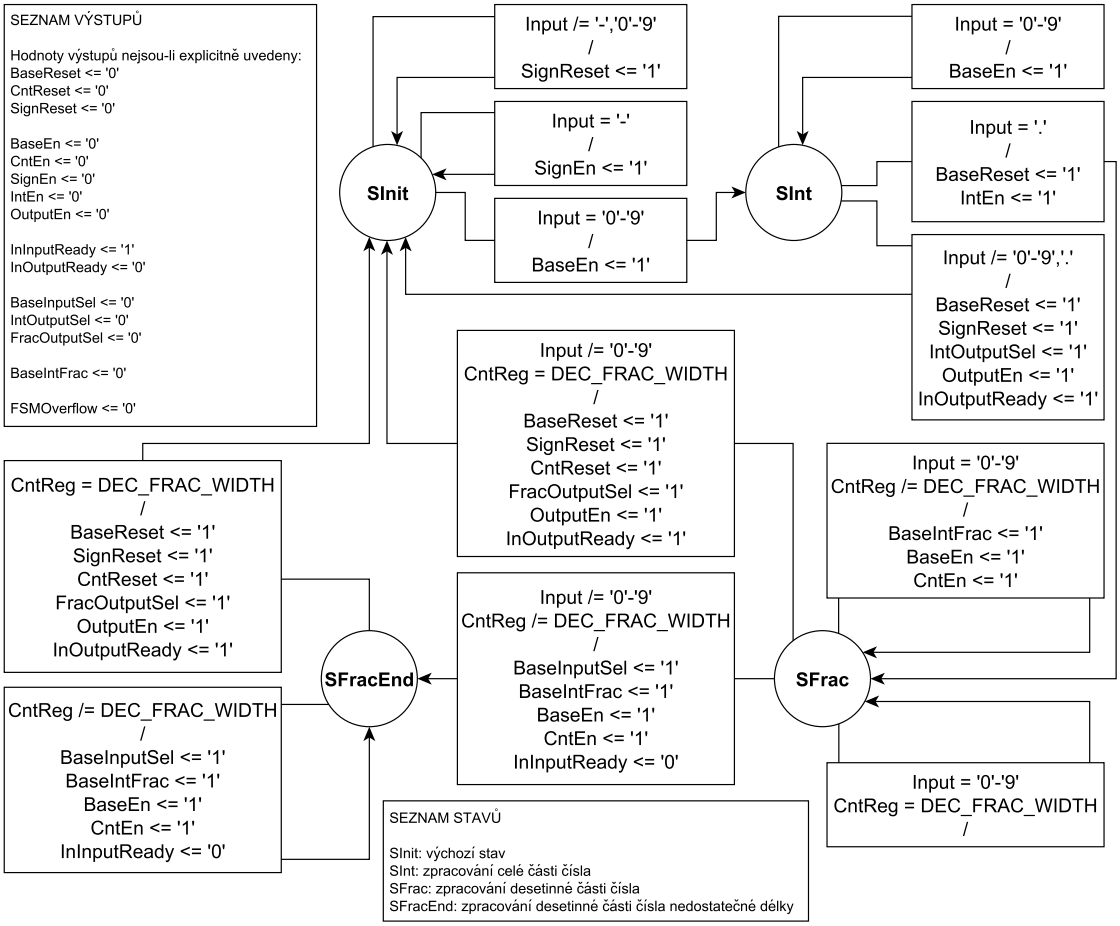
`BaseIntFrac` přepínač načítání celé a desetinné části dekadického čísla bázové komponenty (celá část v logické 0, nebo desetinná část v logické 1)

`FSMOverflow` dílčí příznak přetečení řídicího automatu

Řídicí konečný automat je implementován v jedné ze tří verzí. Konkrétní verze je vybrána podle toho, jestli je součástí výsledného binárního čísla celá i zlomková část (jsou-li parametry `BIN_WIDTH` a `BIN_FRAC_WIDTH` oba nenulové), nebo pouze část celá (je-li parametr `BIN_FRAC_WIDTH` nulový), nebo pouze část zlomková (je-li parametr `BIN_WIDTH` nulový). Jednotlivé verze se od sebe liší především tím, že nemusejí převádět příslušnou část dekadického čísla. S tím souvisí také způsob využití bázové komponenty a ostatních prostředků při načítání dekadických čísel. Stavový diagram verze, kdy je součástí výsledného binárního čísla celá i zlomková část, je znázorněn na obrázku 4.6. Diagramy ostatních verzí řídicího automatu jsou pak součástí příloh této práce (obrázky A.1 a A.2).

Každý diagram obsahuje seznam používaných řídicích signálů, které jsou podmnožinou výše uvedeného výčtu. Řídicí signály, které v seznamu nejsou uvedeny, nejsou danou verzí řídicího automatu využívány a logika s nimi související není implementována. Znázorněné stavové diagramy přesně a do detailu demonstrují proces zpracování načítaných ASCII

[1] Řídicí signály jsou aktivní v logické 1, není-li uvedeno jinak.



Obrázek 4.6: Stavový diagram řídicího konečného automatu pro načítání ASCII čísel, když je součástí výsledného binárního čísla celá i zlomková část

znaků jednotlivými verzemi řídicího automatu v závislosti na hodnotách parametrů, strukturu zdrojového ASCII řetězce a způsob využití bázové komponety při načítání jednotlivých částí dekadického čísla.

Během své činnosti řídicí automat načítá ASCII znaky ze vstupu `Input` a všechny načtené znaky zahazuje, dokud nenarazí na dekadickou číslici ('0' až '9'). Načtení záporného znaménka ('−') bezprostředně před dekadickou číslicí vede k nahrazení výchozího kladného znaménka ('+'). Nastavené znaménko je následně použito pro závěrečnou znaménkovou úpravu výsledného binárního čísla. Načtení dekadické číslice pak zahajuje fázi načítání celé části dekadického čísla, která je ukončena až načtením znaku odlišného od dekadické číslice. Je-li posledním načteným znakem desetinná tečka ('.'), následuje fáze načítání desetinné části dekadického čísla, jinak je načítání čísla ukončeno. Fáze načítání desetinné části dekadického čísla je ukončena až načtením znaku odlišného od dekadické číslice a celý proces se může od začátku opakovat.

4.3 Vyhodnocení implementace

Vzhledem k charakteru použití a souvisejícím požadavkům je jednotka pro načítání ASCII čísel implementována s důrazem na jednoduchost. Konstrukce použité při implementaci jsou založeny na co nejjednodušších operacích a umožňují tak jednotce dosáhnout vysokých rychlostí a nízkých požadavků na zdroje. Konkrétní výsledky dosažené různými konfiguracemi implementované jednotky jsou uvedeny v tabulce B.1, která je součástí příloh této práce.

S narůstajícím počtem bitů výsledného binárního čísla dle očekávání klesá maximální frekvence jednotky a stoupá množství využitých zdrojů. Tento trend však není nijak extrémní a jednotka dosahuje dobrých výsledků i při zpracování velkého počtu bitů.

Počet taktů synchronizačního signálu nutných k načtení dekadického čísla je s jedinou výjimkou roven počtu jemu příslušejících ASCII znaků včetně ukončujícího oddělovače. Výjimkou je případ, kdy desetinná část načítaného čísla obsahuje méně dekadických číslic, než je vyžadováno konfigurací, a počet potřebných taktů synchronizačního signálu je pak navýšen o počet chybějících dekadických číslic.

4.4 Ověření funkčnosti

Ověření funkčnosti jednotky pro načítání ASCII čísel je založeno na sérii testů prováděných v simulačním i reálném prostředí. Testování v reálném prostředí je umožněno s využitím testovací a demonstrační aplikace, která je popsána v kapitole 6. Vzorky prováděných testů, zahrnující testovací vstupy a výstupy pro různé konfigurace implementované jednotky, jsou uvedeny v tabulkách B.3, B.4, B.5 a B.6, které jsou součástí příloh této práce.

Kapitola 5

Jednotka pro tisk ASCII čísel

5.1 Návrh

Návrh jednotky pro tisk dekadických čísel do ASCII řetězce po jejich převodu z binární reprezentace je koncipován tak, aby byl funkční základ oddělen od řídicí části. Návrh rovněž definuje rozhraní jednotky a určuje její funkci.

5.1.1 Bázová komponenta

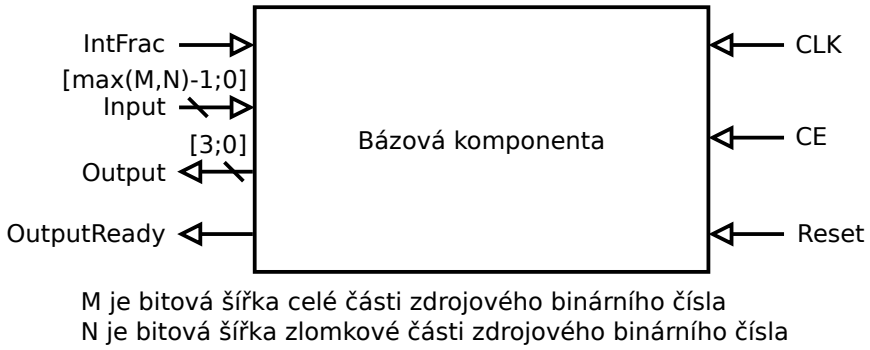
Funkční základ jednotky je realizován formou samostatné bázové komponenty, která představuje výpočetní jádro jednotky a implementuje převodní algoritmy. Bázová komponenta poskytuje jednotce prostředky pro tisk celé i desetinné části dekadického čísla. Každá z částí dekadického čísla je přitom tisknuta samostatně. Komponenta je navržena pro práci pouze s bezznaménkovými čísly. To umožňuje její jednodušší implementaci a snazší samostatnou přenositelnost v případech, kdy jsou zpracovávána pouze čísla bez znaménka. Návrh rovněž definuje rozhraní a funkci bázové komponenty, vycházející z charakteristik využitých převodních algoritmů uvedených v sekcích 3.3 a 3.4.

Rozhraní

Rozhraní bázové komponenty obsahuje vstup synchronizačního signálu (`CLK`), vstup povolení synchronizačního signálu (`CE`), vstup synchronního resetu (`Reset`), vstup přepínače tisku celé a desetinné části dekadického čísla (`IntFrac`), vstup vstupního binárního čísla bez znaménka (`Input`) konfigurovatelné bitové šířky, čtyřbitový výstup tisknuté dekadické číslice v binární podobě BCD kódu (`Output`) a výstup příznaku připravenosti předat tisknutou dekadickou číslici (`OutputReady`). Rozhraní je znázorněno na obrázku 5.1.

Funkce

Vstup `Input` je načten s první nástupnou hranou synchronizačního signálu `CLK`, pokud jsou vstupy `CE` v logické 1 a `Reset` v logické 0. Následně je za stejných podmínek prováděn převod a tisk. Převod a tisk je možné pozastavit nastavením vstupu `CE` na logickou 0, nebo zrušit nastavením vstupu `Reset` na logickou 1. Během celého převodu a tisku musí být vstup `IntFrac` správně nastaven na logickou 0 při tisku celé části dekadického čísla, nebo na logickou 1 při tisku části desetinné. Tisknuté dekadické číslice se na výstupu `Output` začnou objevovat po dokončení pro tisk vyžadované fáze převodu. Zároveň s nimi je nastavován



Obrázek 5.1: Návrh bázové komponenty pro tisk ASCII čísel a její rozhraní

výstup `OutputReady` na logickou 1. Před každým novým tiskem je nutné provést vynulování bázové komponenty nastavením vstupu `Reset` na logickou 1.

5.1.2 Řídicí část

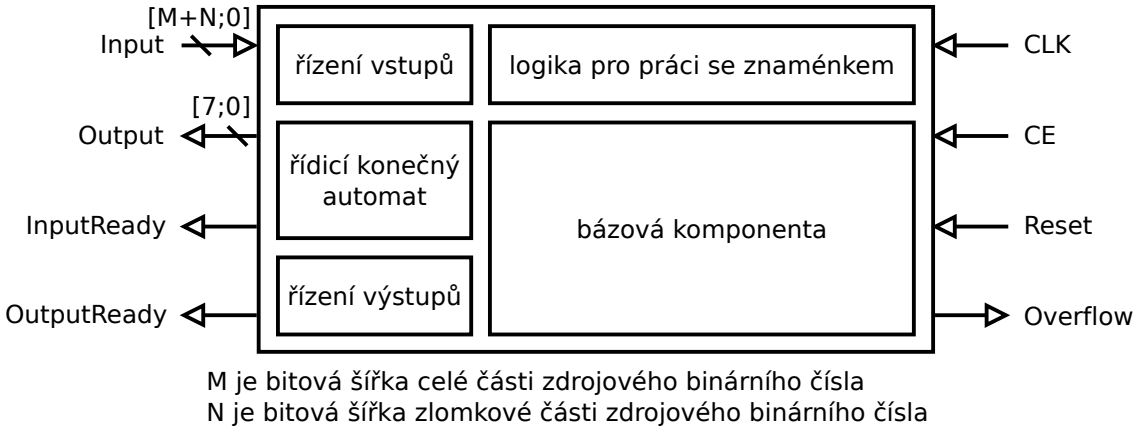
Řídicí část jednotky zapouzdřuje bázovou komponentu, poskytuje jí řízení nezbytné pro převod binárních čísel do dekadické reprezentace a využívá jejích prostředků pro tisk celé a desetinné části dekadických čísel. Řízení jednotky je realizováno pomocí konečného automatu ve spojení s podpůrnými prostředky. Vzhledem k tomu, že bázová komponenta je navržena pouze pro bezznaménková čísla, poskytuje řídicí část jednotky navíc prostředky pro práci se znaménkem. Počet tisknutých číslic desetinné části dekadického čísla je pevně dán konfigurací jednotky.

5.1.3 Rozhraní

Rozhraní navržené jednotky obsahuje vstup synchronizačního signálu (`CLK`), vstup povolení synchronizačního signálu (`CE`), vstup synchronního resetu (`Reset`), vstup zdrojového binárního čísla se znaménkem v doplňkovém kódu (`Input`) konfigurovatelné bitové šířky, osmibitový výstup tisknutého ASCII znaku (`Output`), výstup příznaku připravenosti přijmout zdrojové binární číslo (`InputReady`), výstup příznaku připravenosti předat tisknutý ASCII znak (`OutputReady`) a výstup příznaku přetečení (`Overflow`). Rozhraní je znázorněno na obrázku 5.2.

5.1.4 Funkce

Vstup `Input` je načten s první nástupnou hranou synchronizačního signálu `CLK`, pokud jsou vstupy `CE` v logické 1, `Reset` v logické 0 a výstup `InputReady` v logické 1. Následně je za stejných podmínek prováděn převod a tisk. Převod a tisk je možné pozastavit nastavením vstupu `CE` na logickou 0, nebo zrušit nastavením vstupu `Reset` na logickou 1. Tisknuté ASCII znaky se na výstupu `Output` objevují během převodu a zároveň s nimi je nastavován výstup `OutputReady` na logickou 1. V případě, že při převodu dojde k přetečení, je nastaven výstup `Overflow` na logickou 1. Navržená jednotka je schopna tisku více dekadických čísel v řadě, oddělených konfigurovatelným oddělovačem. Jednotka umožňuje tisk v přizpůsobivém, nebo v pevném formátu, který zahrnuje i tisk kladného znaménka a vedoucích nul.



Obrázek 5.2: Návrh jednotky pro tisk ASCII čísel a její rozhraní

5.2 Implementace

Implementace jednotky pro tisk dekadických čísel do ASCII řetězce po jejich převodu z binární reprezentace zahrnuje parametrizaci implementace, implementaci bázové komponenty a implementaci řídicí části včetně podpůrných prostředků. Implementace bázové komponenty je založena na dvou architekturách využívajících různé algoritmy pro tisk celé části dekadického čísla s odlišnými vlastnostmi pro zvýšení variability implementované jednotky.

5.2.1 Parametrizace

Implementace jednotky je realizována tak, aby byla parametrizovatelná a použitelná v různých konfiguracích, přitom jsou používány parametry:

`BIN_WIDTH` určující bitovou šířku celé části zdrojového binárního čísla

`BIN_FRAC_WIDTH` určující bitovou šířku zlomkové části zdrojového binárního čísla

`DEC_FRAC_WIDTH` určující počet tisknutých číslic desetinné části dekadického čísla

`END_CHAR` určující ASCII znak používaný jako oddělovač

`FIXED_OUTPUT` povolující tisk čísel v pevném formátu včetně znaménka a vedoucích nul

Bitová šířka zdrojového binárního čísla je pak určena součtem hodnot parametrů `BIN_WIDTH`, `BIN_FRAC_WIDTH` a 1 bitu vyhrazeného pro znaménko. Zpracovatelné hodnoty jsou navíc omezeny intervalem $(-2^{\texttt{BIN\_WIDTH}}, +2^{\texttt{BIN\_WIDTH}})$.

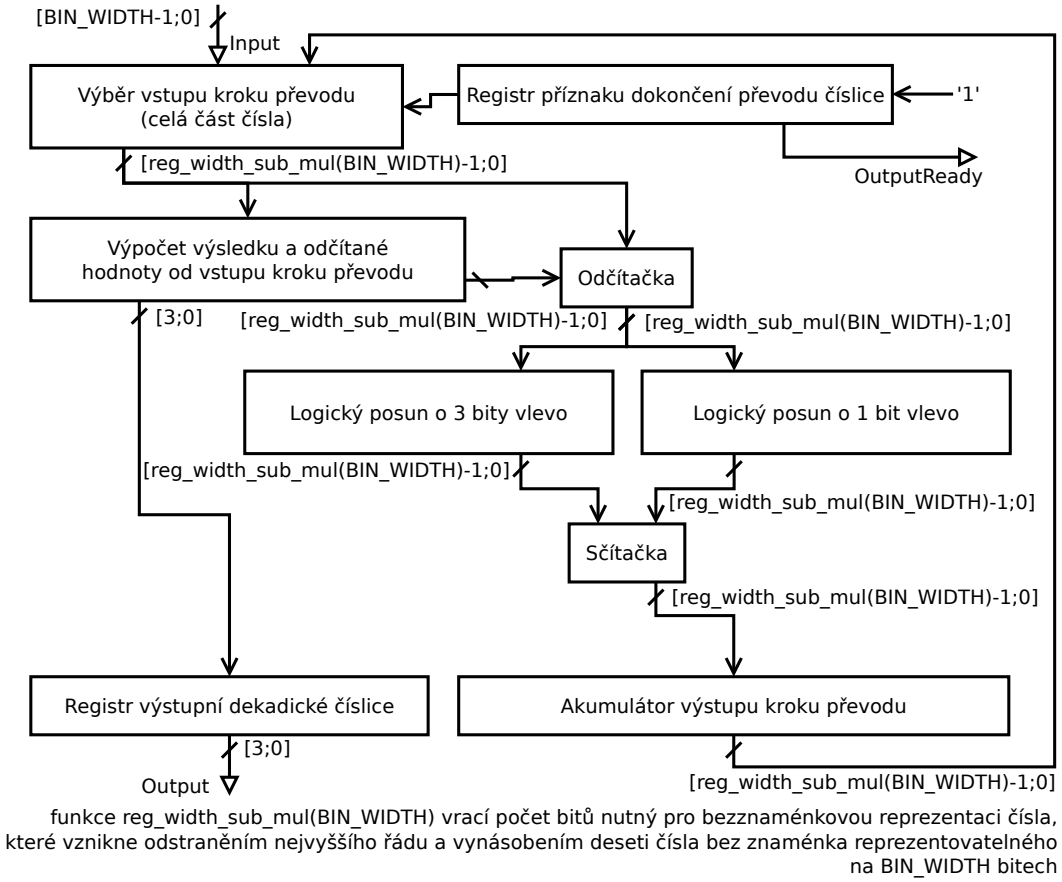
Parametry `BIN_WIDTH` a `BIN_FRAC_WIDTH` jsou beze změny předávány bázové komponentě skrze stejnojmenné protějšky. Parametry `DEC_FRAC_WIDTH`, `END_CHAR` a `FIXED_OUTPUT` bázové komponentě předávány nejsou.

5.2.2 Bázová komponenta

Parametrizace

Implementace bázové komponenty je rovněž parametrizovatelná vlastními parametry:

`BIN_WIDTH` určující bitovou šířku celé části zdrojového binárního čísla



Obrázek 5.3: Blokové schéma tisku celé části dekadického čísla algoritmem s komparátory

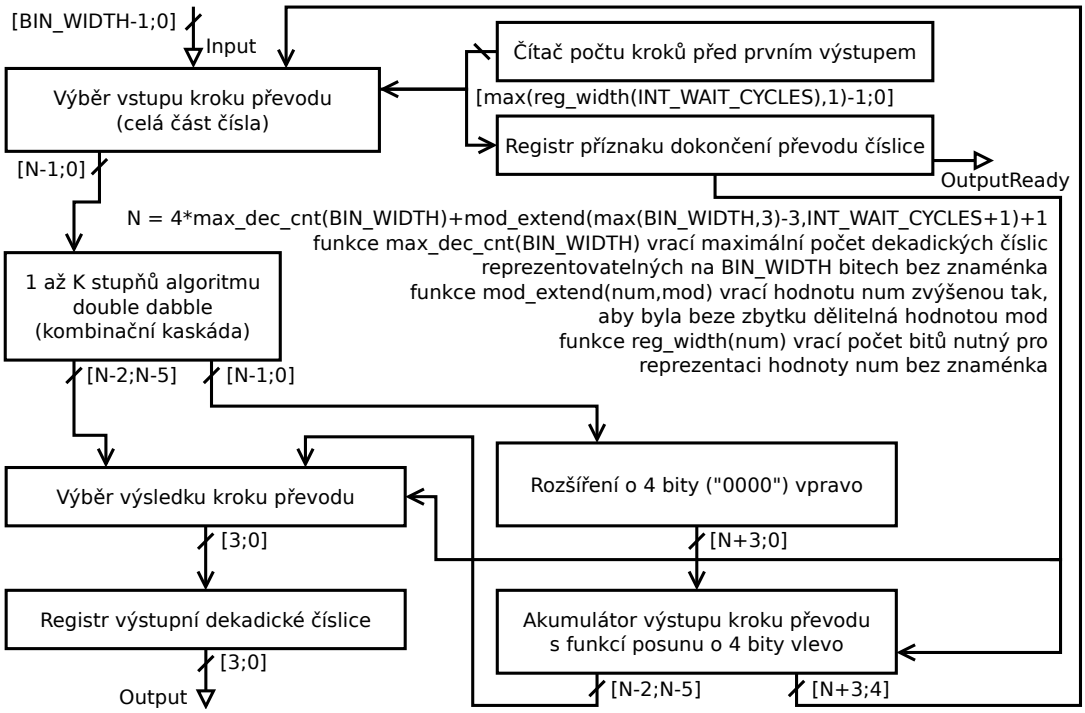
`BIN_FRAC_WIDTH` určující bitovou šířku zlomkové části zdrojového binárního čísla

Bitová šířka vstupního binárního čísla je určena maximem z hodnot parametrů `BIN_WIDTH` a `BIN_FRAC_WIDTH`.

Tisk celé části dekadického čísla algoritmem s využitím komparátorů

Implementace tisku celé části dekadického čísla algoritmem s využitím komparátorů bázové komponenty je založena na algoritmu popsaném v sekci 3.3.1. Logická struktura implementovaného algoritmu je znázorněna na obrázku 5.3.

V každém taktu synchronizačního signálu `CLK` během tisku je hodnota vstupu kroku převodu, kterou je v prvním taktu hodnota vstupu `Input`, podrobena výpočtu výsledku a odčítané hodnoty kroku převodu. Prvním výstupem tohoto výpočtu je hodnota aktuálně tisknuté dekadické číslice. Ta je s následující nástupnou hranou signálu `CLK` uložena do registru výstupní dekadické číslice a vystavena na výstup `Output`. Druhým výstupem je hodnota, která je s pomocí odčítačky odečtena od hodnoty vstupu kroku převodu. Rozdíl je pak za použití bitových posunů vlevo a sčítačky podle přepisu 3.4 vynásoben deseti. Výsledek výpočtu je zároveň s nastavením výstupu `Output` uložen do akumulátoru výstupu kroku převodu. Výstup aktuálního kroku převodu se pak stane vstupem kroku následujícího. Tisknuté dekadické čislice jsou dostupné od druhého taktu signálu `CLK` spolu s logickou 1 v registru příznaku dokončení převodu číslice a na výstupu `OutputReady`.



Obrázek 5.4: Blokové schéma tisku celé části dekadického čísla algoritmem double-dabble

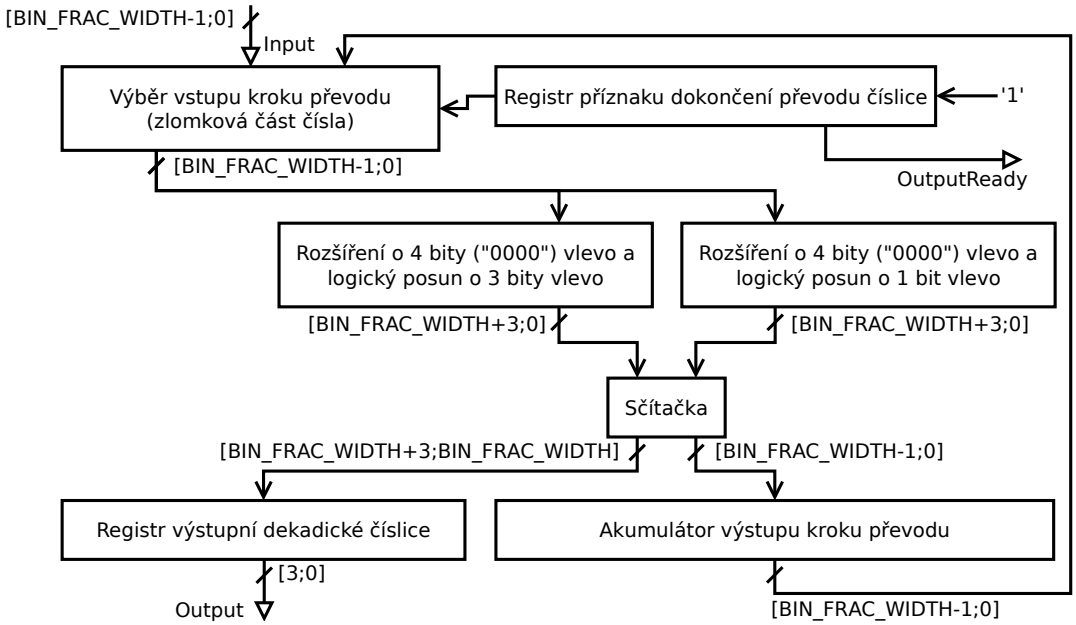
Tisk celé části dekadického čísla algoritmem double-dabble

Implementace tisku celé části dekadického čísla algoritmem double-dabble bázové komponenty je založena na hybridní verzi algoritmu popsané v sekci 3.3.2. Využíván je dodatečný parametr `INT_WAIT_CYCLES` určující počet taktů synchronizačního signálu `CLK` před tiskem první dekadické číslice. Logická struktura implementovaného algoritmu je znázorněna na obrázku 5.4.

V každém taktu synchronizačního signálu `CLK` před dokončením převodu je hodnota vstupu kroku převodu, kterou je v prvním taktu hodnota vstupu `Input`, zpracována 1 až k kombinačními stupni algoritmu double-dabble. Výstup zpracování je rozšířen o čtyři nulové bity vpravo a výsledek je s následující nástupnou hranou signálu `CLK` uložen do akumulátoru výstupu kroku převodu. Výstup aktuálního kroku převodu se pak stane vstupem kroku následujícího. Jakmile jsou dokončeny všechny kroky převodu, dané čítačem počtu kroků před prvním výstupem, je z výstupu kombinační kaskády algoritmu double-dabble vybrána hodnota první tisknuté dekadické číslice. Ta je s následující nástupnou hranou signálu `CLK` uložena do registru výstupní dekadické číslice a vystavena na výstup `Output`. Zároveň je do akumulátoru uložen výsledek výpočtu a do registru příznaku dokončení převodu číslice a na výstup `OutputReady` je vložena logická 1. V každém následujícím taktu signálu `CLK` je obsah akumulátoru posunut o čtyři bity vlevo a je z něj vybrána hodnota aktuálně tisknuté číslice, která se v příštím taktu objeví na výstupu.

Tisk desetinné části dekadického čísla

Implementace tisku desetinné části dekadického čísla bázové komponenty je založena na algoritmu popsaném v sekci 3.4. Logická struktura implementovaného algoritmu je znázorněna na obrázku 5.5.



Obrázek 5.5: Blokové schéma tisku desetinné části dekadického čísla

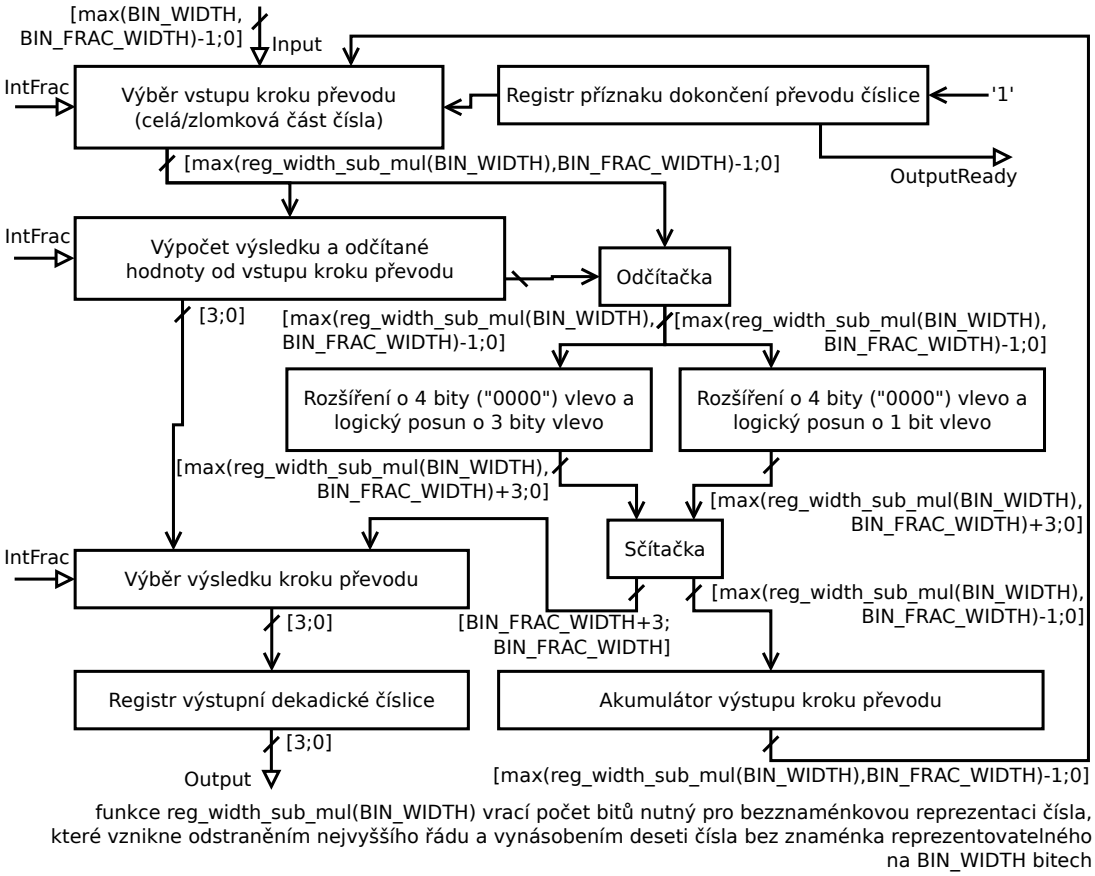
V každém taktu synchronizačního signálu `CLK` během tisku je hodnota vstupu kroku převodu, kterou je v prvním taktu hodnota vstupu `Input`, rozšířena o čtyři nulové bity vlevo a vynásobena deseti za použití bitových posunů vlevo a sčítačky podle přepisu 3.4. Hodnota nejvyšších čtyř bitů výsledku výpočtu představuje hodnotu aktuálně tisknuté dekadické číslice. Ta je s následující nástupnou hranou signálu `CLK` uložena do registru výstupní dekadické číslice a vystavena na výstup `Output`. Zároveň je zbytek výsledku výpočtu uložen do akumulátoru výstupu kroku převodu a stane se vstupem následujícího kroku převodu. Tisknuté dekadické čislice jsou dostupné od druhého taktu signálu `CLK` spolu s logickou 1 v registru příznaku dokončení převodu číslice a na výstupu `OutputReady`.

Kombinace tisku obou částí dekadického čísla

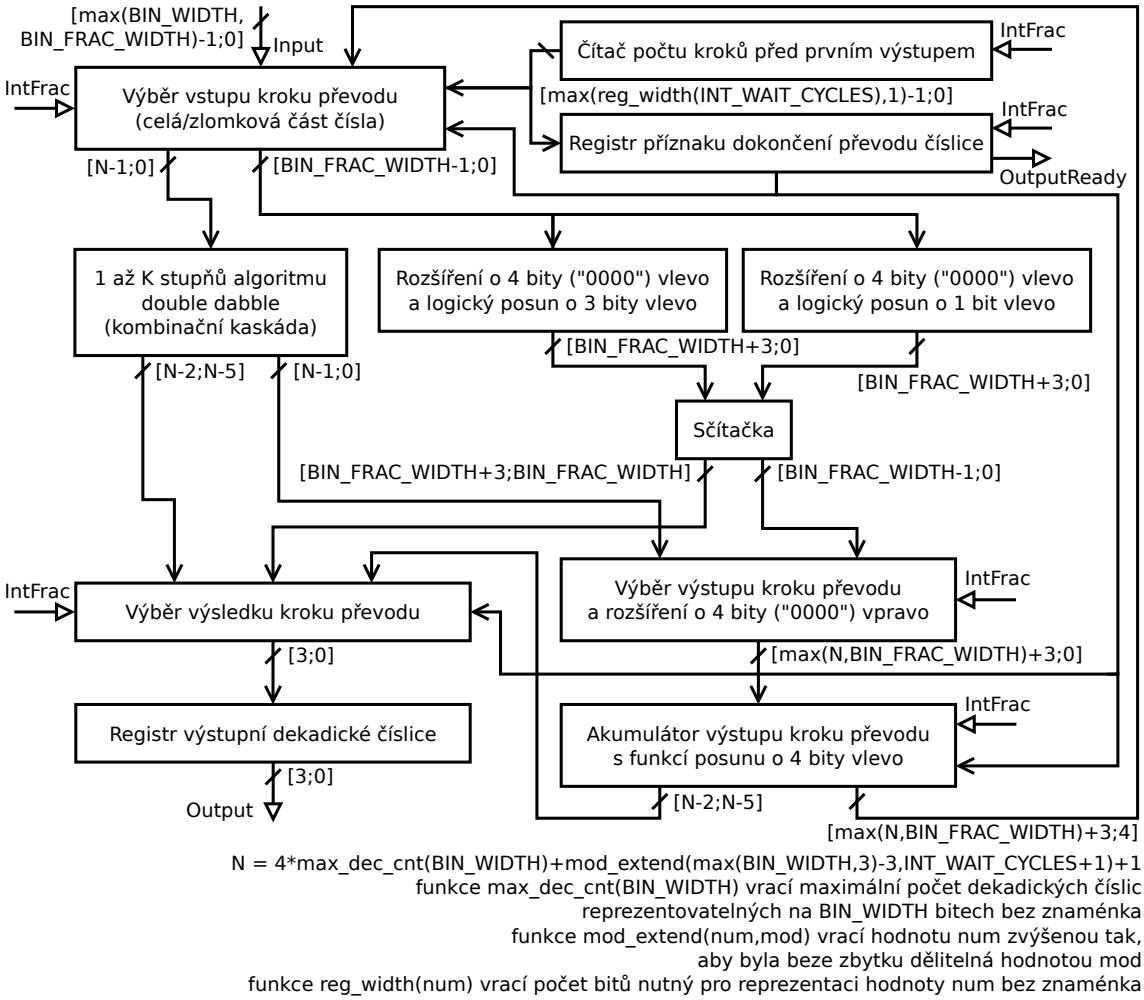
Vzhledem k použití, kdy je v jedné fázi tisku převáděna vždy pouze celá, nebo pouze zlomková část zdrojového binárního čísla, je bázová komponenta implementována s maximálním možným sdílením zdrojů mezi implementacemi tisku obou částí dekadického čísla. To umožňuje maximální využití implementovaného hardwaru a redukci nároků na zdroje za cenu mírného snížení rychlosti. Kombinované logické schéma implementace bázové komponenty je znázorněno na obrázcích 5.6 (architektura *main* s využitím komparátorů) a 5.7 (architektura *double_dabble*).

Oproti samostatným implementacím tisku celé a desetinné části dekadického čísla je navíc využit vstup `IntFrac`. Ten ovlivňuje činnost jednotky tak, aby bylo možné tisknout obě části dekadického čísla pomocí sdílené implementace. Vlastní postup tisku přitom zůstává nezměněn.

I přes preferované sdílení zdrojů je bázová komponenta implementována tak, že v případě jejího použití pro tisk pouze celé, nebo pouze desetinné části dekadického čísla je maximum přebytečné logiky, typické pro nepoužívanou převodní část, při optimalizaci odstraněno. Výsledná implementace tisku dané části dekadického čísla pak má vlastnosti implementace samostatné.



Obrázek 5.6: Blokové schéma implementace bázové komponenty pro tisk ASCII čísel – architektura *main* s využitím komparátorů



Obrázek 5.7: Blokové schéma implementace bázové komponenty pro tisk ASCII čísel – architektura *double_dabble*

5.2.3 Řídicí část

Podpůrné prostředky

Nejdůležitějším podpůrným prostředkem implementované jednotky je logika pro práci se znaménkem. Bázová komponenta pracuje pouze s bezznaménkovými čísly, a její vstup je proto na základě znaménka zdrojového binárního čísla nutné upravit na vhodnou hodnotu. V případě záporného znaménka je pak zdrojové binární číslo před vstupem do bázové komponenty upraveno operací dvojkového doplňku do vhodné bezznaménkové reprezentace.

Druhým nejdůležitějším podpůrným prostředkem je čítač počtu tisknutých číslic celé a desetinné části dekadického čísla `CntReg`. Bázová komponenta neimplementuje žádné prostředky pro řízení počtu tisknutých číslic dekadického čísla, který v případě jeho celé části vychází z hodnoty parametru `BIN_WIDTH` a v případě části desetinné je určen parametrem `DEC_FRAC_WIDTH`, a je třeba použití dodatečného čítače.

Konečný automat

Řízení jednotky je implementováno pomocí konečného automatu, který na základě aktuálního stavu tisku a aktuální hodnoty příznaku připravenosti předat tisknutý ASCII znak bázové komponenty `BaseOutputReady` nastavuje příslušné řídicí signály. Používány jsou řídicí signály[1]:

`BaseReset` vynulování bázové komponenty

`CntReset` vynulování čítače počtu tisknutých číslic celé a desetinné části čísla

`LeadZerosReset` nastavení příznaku tisku znaménka a vedoucích nul dekadického čísla

`BaseEn` povolení synchronizačního signálu bázové komponenty

`CntEn` povolení inkrementace čítače počtu tisknutých číslic celé a desetinné části čísla

`LeadZerosEn` zrušení příznaku tisku znaménka a vedoucích nul dekadického čísla

`FracEn` povolení záznamu registru zlomkové části zdrojového binárního čísla

`InInputReady` vnitřní příznak připravenosti přijmout zdrojové binární číslo

`InOutputReady` vnitřní příznak připravenosti předat tisknutý ASCII znak

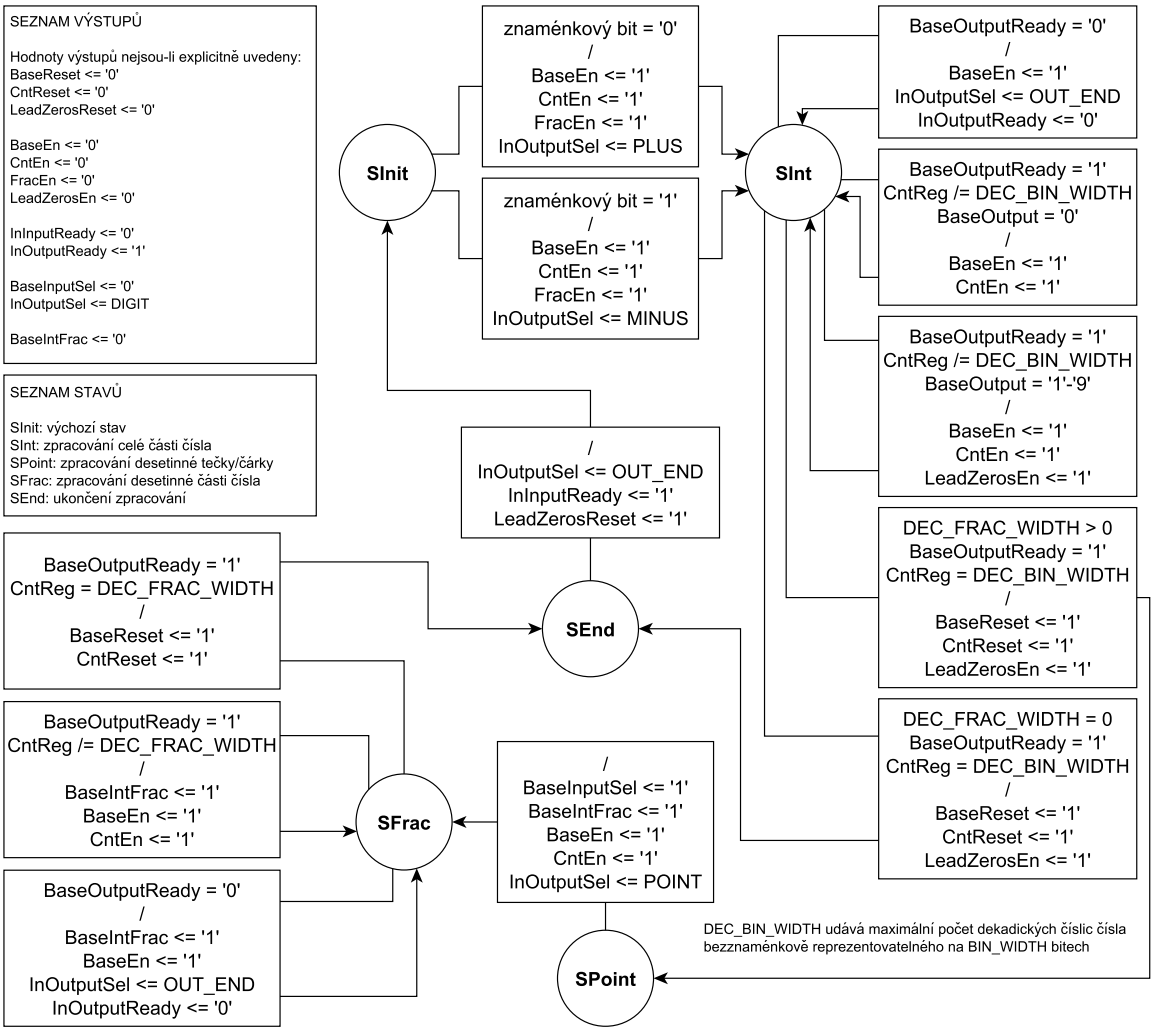
`BaseInputSel` výběr vstupního binárního čísla bázové komponenty (celá část zdrojového binárního čísla v logické 0, nebo uložená zlomková část zdrojového čísla v logické 1)

`InOutputSel` výběr tisknutého ASCII znaku ('+'/`PLUS`, '−'/`MINUS`, '.'/`POINT`, '0'/`ZERO`, tisknutá dekadická číslice bázové komponenty/`DIGIT`, oddělovač/`OUT_END`)

`BaseIntFrac` přepínač tisku celé a desetinné části dekadického čísla bázové komponenty (celá část v logické 0, nebo desetinná část v logické 1)

Řídicí konečný automat je implementován v jedné ze tří verzí. Konkrétní verze je vybrána podle toho, jestli je součástí zdrojového binárního čísla celá i zlomková část (jsou-li parametry `BIN_WIDTH` a `BIN_FRAC_WIDTH` oba nenulové), nebo pouze část celá (je-li parametr

[1] Řídicí signály jsou aktivní v logické 1, není-li uvedeno jinak.



Obrázek 5.8: Stavový diagram řídicího konečného automatu pro tisk ASCII čísel, když je součástí zdrojového binárního čísla celá i zlomková část

`BIN_FRAC_WIDTH` nulový), nebo pouze část zlomková (je-li parametr `BIN_WIDTH` nulový). Jednotlivé verze se od sebe liší především tím, že nemusejí převádět příslušnou část zdrojového binárního čísla. S tím souvisí také způsob využití bázové komponenty a ostatních prostředků při tisku dekadických čísel. Stavový diagram verze, kdy je součástí zdrojového binárního čísla celá i zlomková část, je znázorněn na obrázku 5.8. Diagramy ostatních verzí řídicího automatu jsou pak součástí příloh této práce (obrázky A.3 a A.4).

Každý diagram obsahuje seznam používaných řídicích signálů, které jsou podmnožinou výše uvedeného výčtu. Řídicí signály, které v seznamu nejsou uvedeny, nejsou danou verzí řídicího automatu využívány a logika s nimi související není implementována. Znázorněné stavové diagramy přesně a do detailu demonstrují proces tisku ASCII znaků jednotlivých verzí řídicího automatu v závislosti na hodnotách parametrů, strukturu výsledného ASCII řetězce a způsob využití bázové komponety při tisku jednotlivých částí dekadického čísla.

Během své činnosti řídicí automat tiskne ASCII znaky na výstup `Output`. Nejprve je vytisknuto znaménko[2] ('+' nebo '−') odpovídající hodnotě zdrojového binárního čísla. Poté je zahájena fáze tisku celé části dekadického čísla, která končí vytisknutím všech potřebných dekadických číslic[3] ('0' až '9'). Následuje fáze tisku desetinné části, je-li požadován, jinak je tisk dekadického čísla ukončen. V rámci fáze tisku desetinné části dekadického čísla je nejdříve vytisknuta desetinná tečka ('.') a poté všechny požadované dekadické číslice. Nakonec je vytisknut oddělovač a celý proces se může od začátku opakovat.

5.3 Vyhodnocení implementace

Vzhledem k charakteru použití a souvisejícím požadavkům je jednotka pro tisk ASCII čísel implementována s důrazem na jednoduchost. Konstrukce použité při implementaci jsou založeny na co nejjednodušších operacích a umožňují tak jednotce dosáhnout vysokých rychlostí a nízkých požadavků na zdroje. Konkrétní výsledky dosažené různými konfiguracemi implementované jednotky jsou uvedeny v tabulce B.2, která je součástí příloh této práce.

S narůstajícím počtem bitů zdrojového binárního čísla dle očekávání klesá maximální frekvence jednotky a stoupá množství využitých zdrojů. V případě převodu pouze zlomkové části zdrojového binárního čísla tento trend není nijak extrémní a jednotka dosahuje dobrých výsledků i při zpracování velkého počtu bitů. V případě převodu celé části zdrojového čísla s využitím architektury *main* bázové komponenty s komparátory je pak popsaný trend strmější a výsledky při zpracování velkého počtu bitů jsou horší, nicméně stále v mezích použitelnosti. V případě využití architektury *double_dabble* jsou výsledky výrazně horší, ale lze je ovlivnit pomocí parametru `INT_WAIT_CYCLES` a dosáhnout tak i výsledků srovnatelných s převodem pouze zlomkové části zdrojového binárního čísla.

Počet taktů synchronizačního signálu nutných k vytisknutí dekadického čísla je dán součtem maximálního počtu dekadických číslic jeho celé části (vycházejícího z hodnoty parametru `BIN_WIDTH`), hodnoty parametru `DEC_FRAC_WIDTH` a počtu pomocných ASCII znaků, zahrnujících znaménko, případnou desetinnou tečku a ukončující oddělovač. V případě převodu celé části zdrojového binárního čísla s využitím architektury *double_dabble* bázové komponenty je počet potřebných taktů synchronizačního signálu navýšen o hodnotu parametru `INT_WAIT_CYCLES`.

5.4 Ověření funkčnosti

Ověření funkčnosti jednotky pro tisk ASCII čísel je založeno na sérii testů prováděných v simulačním i reálném prostředí. Testování v reálném prostředí je umožněno s využitím testovací a demonstrační aplikace, která je popsána v kapitole 6. Vzorky prováděných testů, zahrnující testovací vstupy a výstupy pro různé konfigurace implementované jednotky, jsou uvedeny v tabulkách B.7, B.8 a B.9, které jsou součástí příloh této práce.

[2] Kladné znaménko je skutečně vytisknuto pouze v případě tisku v pevném formátu (`FIXED_OUTPUT`).
[3] Vedoucí nuly jsou skutečně vytisknuty pouze v případě tisku v pevném formátu (`FIXED_OUTPUT`).

Kapitola 6

Testovací a demonstrační aplikace

Testovací a demonstrační aplikace implementovaných hardwarových jednotek je realizována s využitím výukové platformy FITkit 2.0 [3]. FITkit 2.0 je samostatný hardware, který obsahuje mikrokontroler, hradlové pole FPGA a řadu periferií včetně dvouřádkového LCD displeje a klávesnice s tlačítky **0** až **9**, **A** až **D**, $*$ a #. Součástí FITkitu je rovněž USB rozhraní schopné komunikace s počítačem pomocí softwarového terminálu. To vše jsou prostředky potřebné pro realizaci dané aplikace.

Jednotky pro načítání a tisk ASCII čísel jsou implementovány v čipu FPGA a řízeny programem mikrokontroleru, který rovněž ovládá LCD displej a obsluhuje klávesnici. Načítaná dekadická čísla jsou zadávána prostřednictvím klávesnice, kdy jednotlivá její tlačítka reprezentují ASCII znaky odpovídající jejich potisku. Výjimkou jsou tlačítka **A**, **B** a $*$, které reprezentují znaky kladného znaménka '+', záporného znaménka '−' a desetinné tečky '.'. Každý vstupní znak je ihned zpracován, předán jednotce pro načítání a vypsán na displej a terminál. Výsledky načítání jsou rovněž vypisovány na terminál a displej, jehož první řádek pak představuje celou část výsledného binárního čísla a druhý řádek část zlomkovou. Načtená čísla jsou ihned předávána jednotce pro tisk a tisknuta zpět do ASCII řetězce, přičemž výstupní znaky jsou postupně vypisovány na terminál. Takto lze načítat a tisknout i více dekadických čísel v řadě, oddělených oddělovači. Podrobný průběh načítání a tisku je vypisován na terminál v podobě hodnot výstupů převodních jednotek před a po každém taktu převodu.

Testovací a demonstrační aplikace pracuje s binárními čísly o pevné šířce 32 bitů, z toho 1 bit je určen pro znaménko, 15 bitů pro celou část čísla a 16 bitů pro zlomkovou část čísla. Za oddělovač při tisku dekadického čísla (parametr `END_CHAR`) je pevně zvolen znak '#'. Architektury a hodnoty ostatních parametrů jsou plně volitelné.

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat hardwarové jednotky pro převod čísel v pevné řádové čárce mezi dekadickou a binární reprezentací a umožnit načítání a tisk dekadických čísel ve formě ASCII řetězců binárním hardwarovým jednotkám. Implementaci navržených jednotek bylo třeba vyhodnotit s ohledem na jejich podpůrný charakter v rámci většího výpočetního systému a řádně ověřit jejich funkčnost.

K dosažení tohoto cíle bylo nutné nastudovat dostupné prostředky a možnosti řešení dané problematiky, případně sestavit postupy vlastní. Vlastními jsou pak všechny uvedené algoritmy kromě algoritmu double-dabble. Vhodné algoritmy byly přizpůsobeny a optimalizovány pro daný účel a použity při návrhu a implementaci výsledných hardwarových jednotek. Návrh jednotek byl pro jejich vyšší variabilitu koncipován tak, aby byl kromě celku samostatně konfigurovatelný, přenositelný a znovupoužitelný i jejich výpočetní základ. Implementace jednotek byla vedena tak, aby byla maximalizována jejich rychlost a minimalizováno množství využitých zdrojů. V případě více alternativ implementace s různými vlastnostmi, jako je tomu u algoritmu s využitím komparátorů a algoritmu double-dabble pro tisk celé části dekadického čísla, byly implementovány všechny možnosti a výběr konkrétní z nich byl ponechán na výsledném použití jednotky. Úspěšnou realizaci výsledných hardwarových jednotek a jejich vhodnost pro daný účel pak potvrdilo závěrečné vyhodnocení implementace a testování v simulačním i reálném prostředí.

Smyslem realizace jednotek pro načítání a tisk ASCII čísel je jejich široké využití v přítomnosti i budoucnosti. Vzniklé hardwarové jednotky nyní umožňují efektivní zpracování dekadických čísel v pevné řádové čárce binárním výpočetním systémům. Po případném budoucím rozšíření by mohly být schopny umožnit i zpracování dekadických čísel v plovoucí řádové čárce nebo eventuálně přispět k realizaci jiných příbuzných aplikací.

Literatura

[1] Xilinx CORE Generator System [online]. `http://www.xilinx.com/tools/coregen.htm`, 2013 [cit. 2013-04-29].

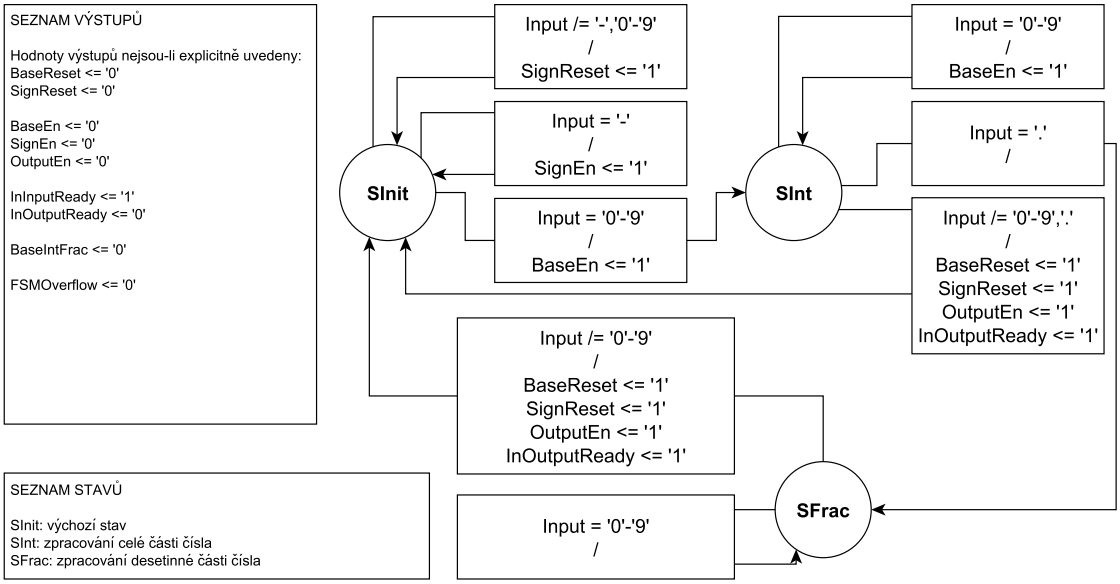
[2] Falconer, C. B.: An Explanation of the Double-Dabble Bin-BCD Conversion Algorithm [online]. `http://www.classiccmp.org/cpmarchives/cpm/mirrors/cbfalconer.home.att.net/download/dubldabl.txt`, 2004 [cit. 2013-04-25].

[3] Vašíček, Z.: FITkit [online]. `http://merlin.fit.vutbr.cz/FITkit/`, 2012 [cit. 2013-04-21].

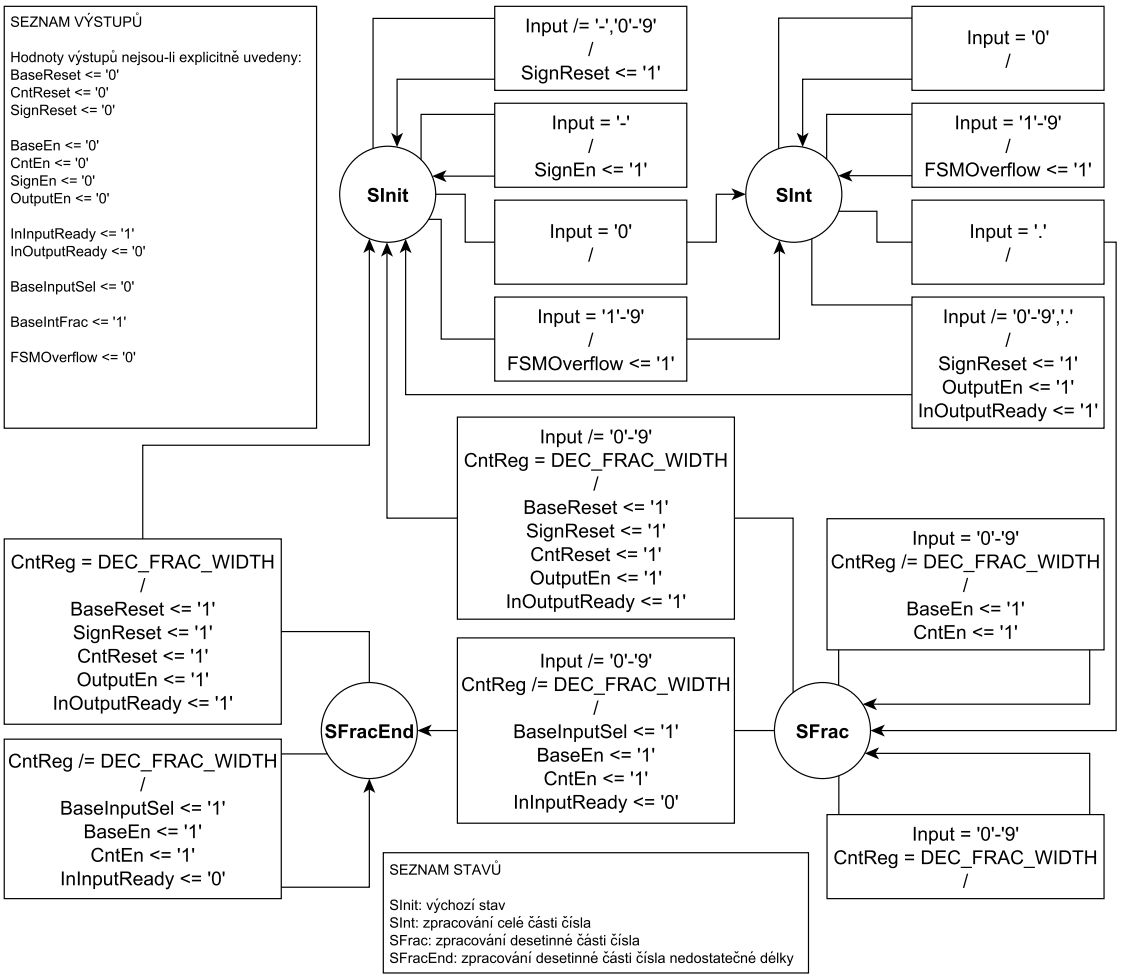
Příloha A

Obrázky

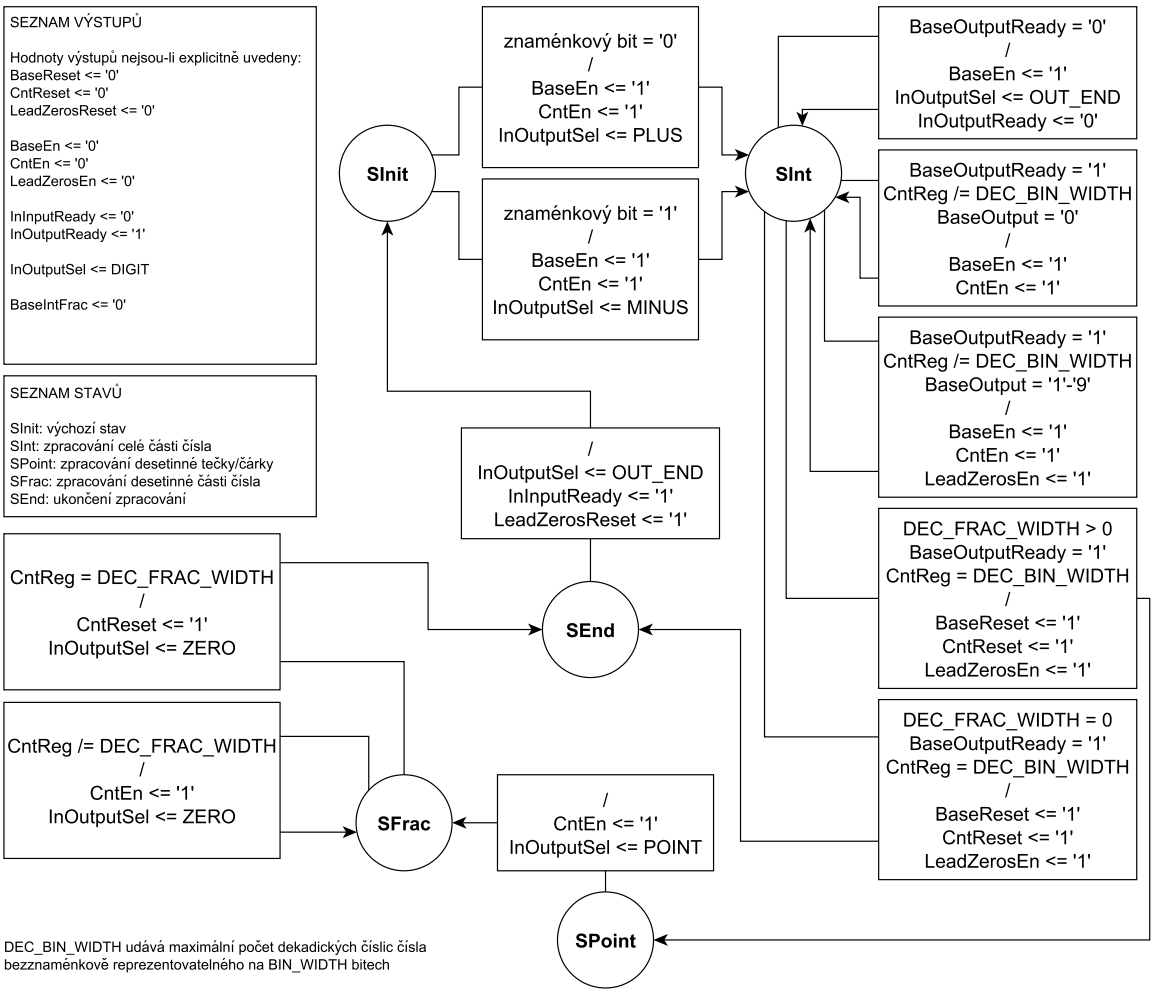
Seznam obrázků



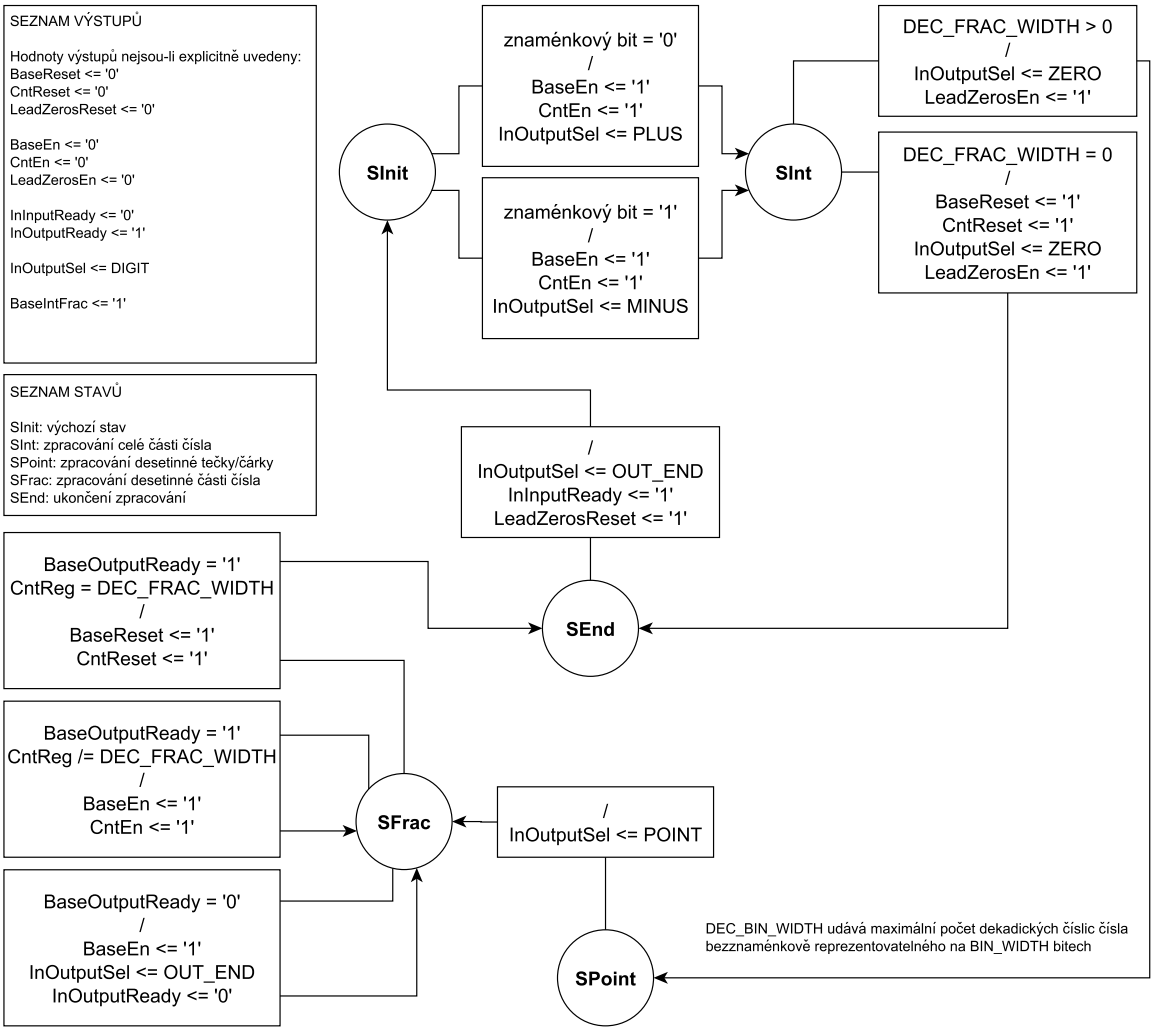
Obrázek A.1: Stavový diagram řídicího konečného automatu pro načítání ASCII čísel, když je součástí výsledného binárního čísla pouze celá část



Obrázek A.2: Stavový diagram řídicího konečného automatu pro načítání ASCII čísel, když je součástí výsledného binárního čísla pouze zlomková část



Obrázek A.3: Stavový diagram řídicího konečného automatu pro tisk ASCII čísel, když je součástí zdrojového binárního čísla pouze celá část



Obrázek A.4: Stavový diagram řídicího konečného automatu pro tisk ASCII čísel, když je součástí zdrojového binárního čísla pouze zlomková část

Příloha B

Tabulky

Seznam tabulek

BIN_WIDTH	BIN_FRAC_WIDTH	DEC_FRAC_WIDTH	Maximální frekvence [MHz]	Počet využitých LUT
4	0	0	341.828	30
8	0	0	321.043	44
16	0	0	299.702	80
32	0	0	270.223	144
0	4	3	356.843	22
0	8	3	328.283	67
0	16	3	289.549	114
0	32	3	248.583	198
4	4	3	308.447	85
4	8	3	238.462	105
4	16	3	226.875	146
4	32	3	206.630	264
8	4	3	238.371	96
8	8	3	231.865	127
8	16	3	222.285	156
8	32	3	202.817	276
16	4	3	226.524	130
16	8	3	222.285	152
16	16	3	213.585	180
16	32	3	197.885	292
32	4	3	207.603	210
32	8	3	205.540	232
32	16	3	198.116	259
32	32	3	183.222	351

Tabulka B.1: Tabulka maximální frekvence a využitých zdrojů různých konfigurací jednotky pro načítání ASCII čísel u FPGA Xilinx Virtex-5 (`PRECISION_BITS` = 0, `FRAC_ROUND` = true)

BIN_WIDTH	BIN_FRAC_WIDTH	DEC_FRAC_WIDTH	architektura	INT_WAIT_CYCLES	Maximální frekvence [MHz]	Počet využitých LUT
4	0	3	main	0	407.515	38
8	0	3	main	0	212.454	76
16	0	3	main	0	136.110	168
32	0	3	main	0	116.999	240
4	0	3	double_dabble	0	407.515	37
8	0	3	double_dabble	0	321.177	56
16	0	3	double_dabble	0	133.473	192
16	0	3	double_dabble	5	301.432	191
32	0	3	double_dabble	0	65.441	582
32	0	3	double_dabble	5	203.969	425
32	0	3	double_dabble	10	266.585	367
0	4	3	main	0	507.395	29
0	8	3	main	0	361.193	48
0	16	3	main	0	313.563	93
0	32	3	main	0	281.441	176
4	4	3	main	0	203.434	84
4	8	3	main	0	144.418	146
4	16	3	main	0	122.098	186
8	4	3	main	0	169.203	117
8	8	3	main	0	164.125	135
8	16	3	main	0	120.274	200
8	32	3	main	0	108.119	290
16	4	3	main	0	125.916	213
16	8	3	main	0	116.204	239
16	16	3	main	0	115.634	240
16	32	3	main	0	104.810	327
32	8	3	main	0	115.488	286
32	16	3	main	0	110.587	318
32	32	3	main	0	104.725	409
4	4	3	double_dabble	0	340.948	52
4	8	3	double_dabble	0	211.684	126
4	16	3	double_dabble	0	229.177	138
8	4	3	double_dabble	0	245.598	101
8	8	3	double_dabble	0	212.685	143
8	16	3	double_dabble	0	218.598	181
8	32	3	double_dabble	0	194.668	249
16	4	3	double_dabble	5	273.209	244
16	8	3	double_dabble	5	238.237	252
16	16	3	double_dabble	5	218.724	317
16	32	3	double_dabble	5	196.769	359
32	8	3	double_dabble	10	210.713	415
32	16	3	double_dabble	10	229.669	431
32	32	3	double_dabble	10	196.769	544

Tabulka B.2: Tabulka maximální frekvence a využitých zdrojů různých konfigurací jednotky pro tisk ASCII čísel u FPGA Xilinx Virtex-5 (`END_CHAR` = 0, `FIXED_OUTPUT` = false)

vstup	výstup
”000123;”	123
”abc123;”	123
”−+123;”	123
”123+”	123
”−o123;”	123
”−.123;”	123
”123.;”	123
”123.456;”	123.456
”−0;”	0

Tabulka B.3: Tabulka testovacích vstupů a výstupů jednotky pro načítání ASCII čísel – testování interpretace vstupního ASCII řetězce

vstup	BIN_WIDTH	BIN_FRAC_WIDTH	DEC_FRAC_WIDTH	PRECISION_BITS	FRAC_ROUND	výstup	výstup dekadicky
1	1	0	0	0	nedef.	0 1	1
−1	1	0	0	0	nedef.	1 1	−1
+/−2	1	0	0	0	nedef.	overflow	overflow
3	2	0	0	0	nedef.	0 11	3
−3	2	0	0	0	nedef.	1 01	−3
+/−4	2	0	0	0	nedef.	overflow	overflow
15	4	0	0	0	nedef.	0 1111	15
−15	4	0	0	0	nedef.	1 0001	−15
+/−16	4	0	0	0	nedef.	overflow	overflow

Tabulka B.4: Tabulka testovacích vstupů a výstupů jednotky pro načítání ASCII čísel – testování převodu dekadického čísla obsahujícího pouze celou část

vstup	BIN _WIDTH	BIN _FRAC _WIDTH	DEC _FRAC _WIDTH	PRECISION _BITS	FRAC _ROUND	výstup	výstup dekadicky
+/−1.500	0	nedef.	nedef.	nedef.	nedef.	overflow	overflow
+/−0.500	0	1	0	0	nedef.	0 . 0	0
0.999	0	1	1	10	false	0 . 1	0.5
0.999	0	1	1	10	true	overflow	overflow
−0.999	0	1	1	10	false	1 . 1	−0.5
−0.999	0	1	1	10	true	overflow	overflow
0.500	0	1	3	10	false	0 . 0	0
0.500	0	1	3	10	true	0 . 1	0.5
−0.500	0	1	3	10	false	0 . 0	0
−0.500	0	1	3	10	true	1 . 1	−0.5
0.500	0	2	3	10	false	0 . 01	0.25
0.500	0	2	3	10	true	0 . 10	0.5
−0.500	0	2	3	10	false	1 . 11	−0.25
−0.500	0	2	3	10	true	1 . 10	−1.5
0.500	0	4	3	10	false	0 . 0111	0.4375
0.500	0	4	3	10	true	0 . 1000	0.5
−0.500	0	4	3	10	false	1 . 1001	−0.4375
−0.500	0	4	3	10	true	1 . 1000	−0.5

Tabulka B.5: Tabulka testovacích vstupů a výstupů jednotky pro načítání ASCII čísel – testování převodu dekadického čísla obsahujícího pouze desetinnou část

vstup	BIN_WIDTH	BIN_FRAC_WIDTH	DEC_FRAC_WIDTH	PRECISION_BITS	FRAC_ROUND	výstup	výstup dekadicky
1.500	1	1	0	0	nedef.	0 1 . 0	1.0
−1.500	1	1	0	0	nedef.	1 1 . 0	−1.0
0.999	1	1	1	10	false	0 0 . 1	0.5
0.999	1	1	1	10	true	0 1 . 0	1
−0.999	1	1	1	10	false	1 1 . 1	−0.5
−0.999	1	1	1	10	true	1 1 . 0	−1
1.999	1	1	1	10	false	0 1 . 1	1.5
1.999	1	1	1	10	true	overflow	overflow
−1.999	1	1	1	10	false	1 0 . 1	−1.5
−1.999	1	1	1	10	true	overflow	overflow
1.500	1	1	3	10	false	0 1 . 0	1.0
1.500	1	1	3	10	true	0 1 . 1	1.5
−1.500	1	1	3	10	false	1 1 . 0	−1.0
−1.500	1	1	3	10	true	1 0 . 1	−1.5
3.500	2	2	3	10	false	0 11 . 01	3.25
3.500	2	2	3	10	true	0 11 . 10	3.5
−3.500	2	2	3	10	false	1 00 . 11	−3.25
−3.500	2	2	3	10	true	1 00 . 10	−3.5
15.500	4	4	3	10	false	0 1111 . 0111	15.4375
15.500	4	4	3	10	true	0 1111 . 1000	15.5
−15.500	4	4	3	10	false	1 0000 . 1001	−15.4375
−15.500	4	4	3	10	true	1 0000 . 1000	−15.5

Tabulka B.6: Tabulka testovacích vstupů a výstupů jednotky pro načítání ASCII čísel – testování převodu dekadického čísla obsahujícího celou i desetinnou část

vstup dekadicky	vstup	BIN _WIDTH	BIN _FRAC _WIDTH	DEC _FRAC _WIDTH	FIXED _OUPUT	výstup
1	0 1	1	0	0	true	"+1;"
1	0 1	1	0	0	false	"1;"
−1	1 1	1	0	0	true	"−1;"
−1	1 1	1	0	0	false	"−1;"
−2	1 0	1	0	nedef.	nedef.	overflow
3	0 11	2	0	0	true	"+3;"
3	0 11	2	0	0	false	"3;"
−3	1 01	2	0	0	true	"−3;"
−3	1 01	2	0	0	false	"−3;"
−4	1 00	2	0	nedef.	nedef.	overflow
15	0 1111	4	0	0	true	"+15;"
15	0 1111	4	0	0	false	"15;"
−15	1 0001	4	0	0	true	"−15;"
−15	1 0001	4	0	0	false	"−15;"
−16	1 0000	4	0	nedef.	nedef.	overflow
1	0 0001	4	0	3	true	"+01.000;"
1	0 0001	4	0	3	false	"1.000;"
−1	1 1111	4	0	3	true	"−01.000;"
−1	1 1111	4	0	3	false	"−1.000;"

Tabulka B.7: Tabulka testovacích vstupů a výstupů jednotky pro tisk ASCII čísel – testování převodu binárního čísla obsahujícího pouze celou část (END_CHAR = ';')

vstup dekadicky	vstup	BIN _WIDTH	BIN _FRAC _WIDTH	DEC _FRAC _WIDTH	FIXED _OUPUT	výstup
0.500	0 . 1	0	1	0	true	"+0;"
0.500	0 . 1	0	1	0	false	"0;"
0.500	0 . 1	0	1	3	true	"+0.500;"
0.500	0 . 1	0	1	3	false	"0.500;"
−0.500	1 . 1	0	1	3	true	"−0.500;"
−0.500	1 . 1	0	1	3	false	"−0.500;"
0.750	0 . 11	0	2	3	true	"+0.750;"
0.750	0 . 11	0	2	3	false	"0.750;"
−0.750	1 . 01	0	2	3	true	"−0.750;"
−0.750	1 . 01	0	2	3	false	"−0.750;"
0.9375	0 . 1111	0	4	3	true	"+0.937;"
0.9375	0 . 1111	0	4	3	false	"0.937;"
−0.9375	1 . 0001	0	4	3	true	"−0.937;"
−0.9375	1 . 0001	0	4	3	false	"−0.937;"

Tabulka B.8: Tabulka testovacích vstupů a výstupů jednotky pro tisk ASCII čísel – testování převodu binárního čísla obsahujícího pouze zlomkovou část (END_CHAR = ';')

vstup dekadicky	vstup	BIN_WIDTH	BIN_FRAC_WIDTH	DEC_FRAC_WIDTH	FIXED_OUPUT	výstup
1.500	0 1 . 1	1	1	0	true	"+1;"
1.500	0 1 . 1	1	1	0	false	"1;"
−2.000	1 0 . 0	1	1	nedef.	nedef.	overflow
1.500	0 1 . 1	1	1	3	true	"+1.500;"
1.500	0 1 . 1	1	1	3	false	"1.500;"
−1.500	1 0 . 1	1	1	3	true	"−1.500;"
−1.500	1 0 . 1	1	1	3	false	"−1.500;"
3.750	0 11 . 11	2	2	3	true	"+3.750;"
3.750	0 11 . 11	2	2	3	false	"3.750;"
−3.750	1 00 . 01	2	2	3	true	"−3.750;"
−3.750	1 00 . 01	2	2	3	false	"−3.750;"
15.9375	0 1111 . 1111	4	4	3	true	"+15.937;"
15.9375	0 1111 . 1111	4	4	3	false	"15.937;"
−15.9375	1 0000 . 0001	4	4	3	true	"−15.937;"
−15.9375	1 0000 . 0001	4	4	3	false	"−15.937;"
1.000	0 0001 . 0000	4	4	3	true	"+01.000;"
1.000	0 0001 . 0000	4	4	3	false	"1.000;"
−1.000	1 1111 . 0000	4	4	3	true	"−01.000;"
−1.000	1 1111 . 0000	4	4	3	false	"−1.000;"

Tabulka B.9: Tabulka testovacích vstupů a výstupů jednotky pro tisk ASCII čísel – testování převodu binárního čísla obsahujícího celou i zlomkovou část (END_CHAR = ';')

Příloha C

Obsah DVD

Adresářová struktura přiloženého DVD:

```
demonstracni_aplikace
technicka_zprava
    zdrojove_soubory
    technicka_zprava.pdf
zdrojove_texty
    abc_bac_sup_pkg.vhd
    ABC.vhd
    ABCBase.vhd
    BAC.vhd
    BACBase.vhd
licence.txt
manual.txt
```

Adresář `demonstracni_aplikace` – demonstrační aplikace ABC BAC DEMO (kompletní projekt aplikace QDevKit včetně binárních souborů, které stačí nahrát do FITkitu)

Adresář `technicka_zprava` – technická zpráva ve zdrojové formě a ve formátu pdf

- **Adresář** `zdrojove_soubory` – zdrojové soubory
- **Soubor** `technicka_zprava.pdf` – formát pdf

Adresář `zdrojove_texty` – zdrojové texty implementovaných hardwarových jednotek

- **Soubor** `abc_bac_sup_pkg.vhd` – podpůrný balíček
- **Soubor** `ABC.vhd` – jednotka pro načítání ASCII čísel
- **Soubor** `ABCBase.vhd` – bázová komponenta pro načítání ASCII čísel
- **Soubor** `BAC.vhd` – jednotka pro tisk ASCII čísel
- **Soubor** `BACBase.vhd` – bázová komponenta pro tisk ASCII čísel

Soubor `licence.txt` – licenční podmínky k užití implementovaných jednotek

Soubor `manual.txt` – manuál k implementovaným jednotkám a demonstrační aplikaci