VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁSTROJ PRO SIMULACI VÝPOČTŮ S PEVNOU ŘÁDOVOU ČÁRKOU
SIMULATION TOOL FOR FIXED-POINT ARITHMETIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Vojtěch Grézl
AUTHOR

VEDOUCÍ PRÁCE — Ing. Martin Čala, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Vojtěch Grézl ***ID:*** 220980

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Nástroj pro simulaci výpočtů s pevnou řádovou čárkou

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je zjednodušit implementaci výpočtů s čísly s pevnou řádovou čárkou (fixed-point). Výsledek praktické části bude po zadání schémata výpočtu (např. formou blokového diagramu) umožňovat zobrazení vlivu numerické reprezentace matematických operací na nepřesnost výsledku a dále vizualizaci rozložení chyby na mezivýpočtech. Zadání cílí na implementaci algoritmů v LabVIEW a jeho FPGA modulu.
Zadání lze shrnout do následujících bodů:

1. Popište reprezentaci čísel využívající pevnou řádovou čárku a porovnejte ji s čísly s plovoucí řádovou čárkou.
2. Navrhněte uživatelské rozhraní, ve kterém uživatel zadá algoritmus výpočtu, datové typy vstupů a výstupů,a dále definujte realizaci výstupu programu. Také navrhněte blokové schéma algoritmu nástroje.
3. Implementujte uvedený nástroj, použijte prostředí LabVIEW.
4. Po dohodě s vedoucím práce si vyberte alespoň dva netriviální ukázkové příklady, na kterých demonstrujte funkčnost implementace.
5. Zdokumentujte vytvořené řešení, uveďte jeho možný budoucí vývoj a diskutujte výhody a nevýhody.

DOPORUČENÁ LITERATURA:

[1] KORMANYOS, Christopher. Real-Time C++: Efficient Object-Oriented and Template MicrocontrollerProgramming. Třetí. Reutlingen (Německo): Springer, 2018. ISBN 9783662567173.

[2] PADGETT, Wayne T. a David V. ANDERSON. Fixed-Point Signal Processing. Morgan & Claypool Publishers,2009. ISBN 9781598292596.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 23.5.2022

Vedoucí práce: Ing. Martin Čala, Ph.D.

doc. Ing. Václav Jirsík, CSc.

předseda rady studijního programu

Abstrakt

Tato bakalářská práce se zabývá vytvořením nástroje pro simulaci výpočtů s pevnou řádovou čárkou. Díky tomuto nástroji dojde ke zjednodušení a ke zvýšení efektivity provádění operací s hodnotami různých datových typů. Pomocí vyčíslení a grafického zobrazení absolutních chyb, které vytváří převod hodnot mezi datovými typy a pomocí maximální absolutní chyby, může uživatel zhodnotit, je-li pro něj tento převod optimální. Po obeznámení s touto problematikou v rámci teoretického úvodu následuje návrh praktické části, který shrnuje realizaci a postup při návrhu programu v prostředí LabVIEW 2021. Postup návrhu praktické části je proveden na základě vlastností a podpory prostředí LabVIEW a jeho FPGA modulu a cílí na vytvoření přívětivého a přehledného uživatelského rozhraní. Výstupem praktické části je nástroj, který pracuje s vytvořeným VI, obsahující posloupnost různých operací s různými vstupními a výstupními datovými typy. Program slouží pro převod čísel různých datových typů na typ jiný, převážně se jedná o převod na datový typ fixed point. Uživatel udává hlavní směr vytvořením vzorového VI, jehož operace budou následně provedeny. Další parametry, které může uživatel nastavit a ovlivnit tak chod programu, jsou uvedeny na uživatelském rozhraní.

Klíčová slova

Pevná řádová čárka, Plovoucí řádová čárka, Absolutní chyba, Převod datových typů, LabVIEW, SubVI, VI Server, VI Scripting.

Abstract

This bachelor's thesis is focused on creating tool for simulating calculations with fixed point. This tool simplifies and increases the efficiency of performing operations with values of different data types. By calculating and graphically displaying absolute errors, which creates a conversion of values between data types and conversion through maximum absolute errors, the user can evaluate whether this conversion is optimal or not. After getting acquainted with this issue in the theoretical introduction part, the design of the practical part follows, which summarizes the implementation and procedure of program design in the LabVIEW 2021. Process of draft of the practical part is based on attributes of LabVIEW and it's part FPGA module and aim on creating of transparent and user-friendly user interface. The output of the practical part is a tool that works with the created VI, containing a sequence of different operations with different input and output data types. The program is used to convert numbers of different data types to another type, mainly for a conversion to a fixed point data type. The user gives the main direction by creating a sample VI, the operations of which will then be performed. Other parameters that the user can set and affect the program are listed on the user interface.

Keywords

Fixed point, Floating point, Absolute error, Conversion of data types, LabVIEW, SubVI, VI Server, VI Scripting.

Bibliografická citace

GRÉZL, Vojtěch. *Nástroj pro simulaci výpočtů s pevnou řádovou čárkou.* Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/142645. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Martin Čala.

GRÉZL, Vojtěch. *Nástroj pro simulaci výpočtů s pevnou řádovou čárkou* [online]. Brno, 2022 [cit. 2022-05-17]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/142645. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Martin Čala.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Vojtěch Grézl
VUT ID studenta:	220980
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Nástroj pro simulaci výpočtů s pevnou řádovou čárkou

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 19. května 2022 ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Martinovi Čalovi, Ph.D. za pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 19. května 2022

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Tato bakalářská práce se zabývá teoretickým rozborem a následným návrhem a implementací nástroje pro zjednodušení výpočtů s pevnou řádovou čárkou v prostředí LabVIEW. Protože se v praxi neomezujeme pouze na celá čísla, je potřeba vyjádřit i číslo racionální, objevující se například při dělení dvou celých čísel, kdy vzniká zbytek. Zápis můžeme provést pomocí dvou způsobů, kterými jsou zápis pomocí pevné řádové čárky (fixed point) a pomocí plovoucí řádové čárky (floating point).

Na papíře nemají čísla žádné omezení velikosti ani přesnosti. V počítačích jsou ale čísla uložena v paměti a v registrech, které už mají omezený počet bitů. V rámci digitálního hardwaru jsou čísla ukládána v binárním formátu, tudíž se jedná o sekvenci 1 a 0. Jak hardwarové komponenty nebo softwarové funkce interpretují tuto sekvenci určuje nadefinovaný datový typ. Některé mikroprocesory nemusí mít hardwarovou podporu pro výpočty s plovoucí řádovou čárkou. Počítání s datovými typy float nebo double by bylo velmi pomalé, a proto se využívá aritmetiky s pevnou řádovou čárkou.

V rámci teoretického rozboru je detailněji popsán datový typ fixed point a floating point. Na základě informací uvedených v teoretickém rozboru jsou vybudovány některé části nástroje a slouží k bližšímu pochopení realizace praktické části a postupu, kterým byl nástroj navrhován.

Praktickou část v prostředí LabVIEW představuje vytvořený nástroj sloužící k provedení výpočtů na základě vytvořeného vzorového VI, které obsahuje vzájemně propojené controly a operace, o jejichž vlastnostech a propojení získává informace s využitím funkcí VI Serveru. Controly, konstanty i výstupní datové typy operací, u nichž uživatel sám nastavuje datový typ ve vzorovém VI, jsou nejčastěji reprezentovány pomocí pevné řádové čárky a s hodnotou zapsanou pomocí tohoto zápisu se provádí další operace, vzniká tak určitá odchylka od hodnot, které by byly vždy zapsány pomocí plovoucí řádové čárky. Tento nástroj porovnává a vypočítává odchylku operací provedených s pevnou řádovou čárkou (fixed point) od těch s plovoucí řádovou čárkou (floating point). Dále slouží pro převod čísel různých datových typů na typ jiný, převážně se jedná o převod na datový typ fixed point, spolu s vyobrazením různých dat, které informují uživatele o průběhu výpočtů a převodu datových typů. S využitím tohoto nástroje je možné efektivně a rychle provádět výpočty i v případech, kdy není dostupná podpora výpočtů a zápisů čísel typu double, například u FPGA modulu, jehož vlastnosti byly inspirací pro vývoj kódu. Při volbě dostupných bitů pro výsledný datový typ fixed point může dojít ke špatnému odhadu a při převodu datových typů může docházet ke ztrátě dat a následně mohou být způsobeny značné komplikace při snaze tuto ztrátu zmírnit na přijatelnou hodnotu. Díky možnosti rychlé změny nastavení na uživatelském rozhraní probíhá proces zmírnění ztráty rychleji, než kdyby byla nutnost měnit nastavení vždy přímo u každé operace a následně sledovat chování výpočtů. Vytvořený nástroj je, pro ověření funkčnosti, otestován na ukázkových příkladech.

1. TEORETICKÝ ROZBOR

1.1 Co je to fixed point.

Fixed point je datový typ založený na celých číslech představující racionální číslo se znaménkem. Dvě čísla vyjádřená jako fixed point hodnoty také mohou být použita pro reprezentaci komplexních čísel. Práce s fixed pointem je účinná, protože se prakticky jedná o práci s celými čísly. [1]

Čísla typu fixed point obecně nemají řádovou čárku, proto jsou téměř celočíselné a pro manipulaci jako je sčítání, odčítání, násobení a dělení používají běžné celočíselné algoritmy, které mohou být jednodušší a efektivnější než klasická reprezentace pomocí plovoucí řádové čárky. [1]

Čísla s pevnou řádovou čárkou usnadňují práci hardwaru, protože když jsou čísla reprezentována s plovoucí řádovou čárkou, potom operace s velkým počtem bitů (double= 64, single precision=32) mají za následek obrovská čísla, vyžadující obrovský počet bitů pro zápis výsledku, místa v paměti a času potřebných k provedení těchto operací. [2]

Při reprezentaci čísel pomocí pevné řádové čárky dochází k snadné a rychlé implementaci v rámci vysoké úspory paměti. Je vhodný pro real-time aplikace, které nezvládají pracovat efektivně s plovoucí řádovou čárkou. [3]

1.2 Reprezentace/zápis celých a racionálních čísel.

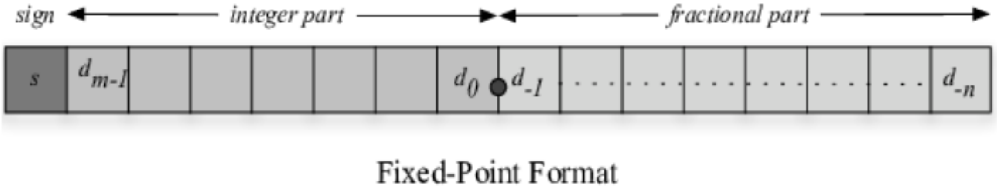
Základ soustavy, ve které se pohybujeme, je označován jako radix. Radix point je označení tečky (čárky), která odděluje desetinnou část od celočíselné. Obecně lze racionální číslo v desítkové soustavě, převáděné z binární, vyjádřit pomocí rovnice (1).

$$x_{(10)} = \sum_{i=-m}^{n-1} b_i * k^i \qquad (1)\ [4]$$

Kde:

b_i – digitální hodnota
k – radix (pro dvojkovou soustavu se jedná o hodnotu 2)
n – počet bitů nalevo od radix pointu
m – počet bitů napravo od radix pointu
$x_{(10)}$ – číslo v desítkové soustavě

[4]



Fixed-Point Format

Obrázek 1:Reprezentace formátu fixed point [5]

Číslo zapsané pomocí fixed pointu není definováno počtem bitů, ale rozsahem a přesností. Při kódování začněte s rozsahem a přesností a potom implementace doporučeného počtu bitů. [6]

Binary value	000	001	010	011	100	101	110	111
Sign and absolute value	+0	+1	+2	+3	-0	-1	-2	-3
One's complement	+0	+1	+2	+3	-3	-2	-1	-0
Two's complement	+0	+1	+2	+3	-4	-3	-2	-1

Obrázek 2: Příklad zápisu celých čísel typu fixed point pro různé vyjádření znaménka [3]

1.3 Q formát

Počítač pracuje s typy celočíselnými, až na několik výjimek. Lze však použít uložené hodnoty jako „změněné" konstanty podle určitého měřítka, které je zadáváno a určuje konstantu bez ohledu na velikost uložené hodnoty. Měřítko udává, kde přesně se nachází radix point. [4]

Jelikož se měřítko může lišit, používá se Q formát zápisu k určení počtu číslic napravo a nalevo od radix pointu. Nejčastěji používaný, pro fixed point aritmetiku, je Q formát pro šestnáctibitová slova. Díky flexibilitě FPGA architektury, a protože často potřebujeme použít mnohem kratší délku, používá se modifikovaný Q zápis, který popisuje obě části, celočíselnou i desetinnou, např formát Q2.2, nebo Q1.15. [4]

Format Type	Examples			
Traditional *Q* format	*Q*15	*Q*14	*Q*31	*Q*3
Modified *Q* format	*Q*1.15	*Q*2.14	*Q*1.31	*Q*1.3
Wordlength, Fractionlength Format	s16,15	s16,14	s32,31	s4,3

Obrázek 3: Způsoby vyjádření pomocí Q formátu [4]

Hlavním účelem Q formátu je umožnit zapsat desetinná čísla (fractional data types) na celočíselný (integer) hardware. [4]

Například Q15.16 popisuje typ s pevnými body s jedním znaménkovým bitem, 15 celočíselnými bity a 16 zlomkovými bity. Reprezentaci Q15.16 lze uložit do 32bitového celého čísla se znaménkem. [1]

Format	Minimum	Maximum	*r*	*n*	*p*	*q*
(UQ16.)	0	$2^{16}-1$	1	16	16	0
(UQ.16)	0	$1-2^{-16}$	2^{-16}	16	0	16
(Q15.)	-2^{15}	$2^{15}-1$	1	16	15	0
(Q.15)	-1	$1-2^{-15}$	2^{-15}	16	0	15
(UQ16.16)	0	$2^{16}-1$	2^{-16}	32	16	16
(Q15.16)	-2^{15}	$2^{15}-2^{-16}$	2^{-16}	32	15	16

Obrázek 4: Příklad parametrů Q formátu [3]

Kde:

r – rozlišení
n – celkový počet bitů
p – počet bitů pro celočíselnou část
q – počet bitů pro desetinnou část

Rozlišení lze vyjádřit pomocí rovnice (2).

$$r = 2^{-q} \qquad (2)\ [3]$$

1.4 Reprezentace čísel s plovoucí řádovou čárkou

Při reprezentaci čísel s plovoucí řádovou čárkou (floating point) se radix point posouvá dynamicky což umožňuje podporu širokého rozsahu zobrazitelných hodnot, na rozdíl od čísel s pevnou řádovou čárkou (fixed point). Struktura čísel s plovoucí řádovou čárkou je taková, že nejvýznamnější bit je znaménkový bit (S), následuje exponent a mantisa. [3]

Obrázek 5: Reprezentace formátu čísel s plovoucí řádovou čárkou [3]

Příklad reprezentace:

Budeme převádět číslo 14.2352. Celou část lze vyjádřit v binární soustavě jako 1110_2 a desetinnou jako 0.00111100001010001111_2

Když provedeme normalizaci čísla $1110.00111100001010001111_2$ na $1.11000111100001010001111_2 * 2^3$ můžeme binárně vyjádřit číslo s plovoucí řádovou čárkou.

- Znaménkový bit (1 bit) bude 0 (jedná se o pozitivní číslo)
- Exponent (8 bitů) bude 130 (127 + 3) zapsaný jako 10000010
- Mantisa (23 bitů) bude vyjádřena jako 11000111100001010001111 [3]

Sign	Exponent	Fraction
0	10000010	11000111100001010001111

Obrázek 6: Finální zápis čísla s plovoucí řádovou čárkou [3]

1.5 Rozsahy a přesnosti datových typů

1.5.1 Rozsahy a přesnost

Čísla zapsaná pomocí pevné řádové čárky (fixed point) mají fixní pozici řádové čárky, na rozdíl od čísel zapsaných pomocí plovoucí řádové čárky (floating pointu), proto u nich dochází ke zmenšení rozsahu i přesnosti. Například minimální reprezentovatelná hodnota u unsigned Q7.8 je 2^{-8} což je přibližně 0,004, pokud tuto hodnotu srovnáme se zápisem čísla s plovoucí řádovou čárkou, ukáže se, že přesnost není příliš vyhovující. A maximální zobrazitelná hodnota pro Q7.8 je +127,996. [1]

Základním důvodem, proč je zde malá přesnost, je téměř celočíselná reprezentace typů s pevnými body. To je však přesně to, co jim propůjčuje jejich vylepšený výkon. Využití této reprezentace s pevnými body snížilo rozsah a přesnost, ale dosáhlo tak vyšší účinnosti, protože při práci lze využívat jednoduché celočíselné algoritmy. [1]

Při definování datových typů fixed point lze měnit buď počet bitů pro zápis celých, nebo desetinných čísel. Taková změna může vést k získání různých výkonů nebo jiných číselných rozsahů a přesností. Reprezentace Q0.8 by mohla být užitečná například v případě, že potřebujeme implementovat pouze takové funkce, jako jsou například trigonometrické výpočty, například funkce sinus a kosinus. Při provádění výpočtů s pevnou řádovou čárkou je potřeba věnovat maximální pozornost tomu, aby byly vždy dodrženy limity (krajní zobrazitelné hodnoty). [1]

1.5.2 Přetečení a podtečení (Overflow a Underflow)

Přetečení v datovém typu s pevnou řádovou čárkou nastane, když hodnota, která se má zapsat, větší než maximální reprezentovatelná hodnota dané konfigurace. [7]

V případě, že dojde k přetečení, je potřeba zvolit režim, který určí, co se provede se zpracovávanou hodnotou. Prvním režimem je režim saturace, kdy se při nastalém přetečení nastaví zobrazená hodnota na zobrazitelné maximum. Druhým režimem je wrap, který vezme binární reprezentaci zadaného čísla a podle počtu dostupných bitů pro zapsání „odsekne“ nejvýznamnější bity které jsou mimo rozsah. [7]

Example FXP Configuration	Overflow Mode	Input Value (Decimal)	Input Value (Binary)	Coerced Value (Binary)	Coerced Value (Decimal)
U4 <2.2>	**Saturate**	6.75	110.11	11.11	3.75
U4 <2.2>	**Wrap**	6.75	110.11	10.11	2.75

Obrázek 7: Příklad výsledků pro úpravy podle overflow módů [7]

Podtečení znamená, že se jedná o situaci, kdy číslo, které chceme zobrazit, není rovno 0, ale dosahuje tak malé hodnoty, že ji není možno zobrazit pro danou přesnost, tudíž se zobrazené číslo bude tvářit jako 0. [8]

1.5.3 Zaokrouhlování

Zaokrouhlování je nejjednodušší postup hardwaru, jak se zbavit nepotřebných bitů. Smazání nejméně významných bitů je operace známá jako krácení. Krácení je výhodné, protože nepotřebuje další hardware ani operace. Problém by mohl nastat v případě automatického krácení nejméně významného bitu u čísla -0.5, což je v binární soustavě vyjádřeno jako 1.1. [4]

Stejně jako v situaci, kdy nastane přetečení, i zde se vybírá režim, jak se úprava (zaokrouhlení) vykoná.

Round Half to Even je režim kdy program zaokrouhlí hodnotu na nejbližší hodnotu, která může být na výstupu zobrazena. Pokud je zaokrouhlovaná hodnota přesně mezi dvěma reprezentovatelnými hodnotami, vybere se sudá hodnota, tudíž nejméně významný bit (LSB) po zaokrouhlení bude mít hodnotu nula. Tento režim zaokrouhlení má největší dopad na výkon, ale vytváří nejpřesnější výstupní hodnoty tím, že neutralizuje vychýlení směrem k vyšším hodnotám. [9]

Round Half Up zaokrouhlí hodnotu na nejbližší hodnotu, která může být na výstupu zobrazena. Pokud je hodnota k zaokrouhlení přesně mezi dvěma reprezentovatelnými hodnotami, tento režim zaokrouhlování zaokrouhlí hodnotu nahoru na vyšší ze dvou platných hodnot v kladném směru. Tento režim zaokrouhlování poskytuje přesnější výsledky. [9]

Round Half Down někdy označováno jako truncate (například v FPGA modulu). Zaokrouhlení hodnoty probíhá dolů směrem k zápornému nekonečnu na nejbližší hodnotu, kterou může typ výstupu představovat. Tento režim zaokrouhlení má nejlepší výkon, ale produkuje nejméně přesné výstupní hodnoty. [9]

1.6 Co je to LabVIEW

LabVIEW, což je zkratka pro Laboratory Virtual Instrument Engineering Workbench, je vývojové prostředí, které používá grafické rozhraní. V podstatě se jedná o grafický programovací jazyk vytvořený firmou National Instruments, využívající místo řádků textu ikonky k vytvoření programu. Na rozdíl od běžných programovacích jazyků, kde se vykonávání programu řídí instrukcemi, LabVIEW využívá datového toku, který určuje pořadí provedení funkcí a VI. Uživatelské rozhraní je zde známo jako front panel, zatímco blokový diagram, který může připomínat vývojový diagram, obsahuje kód. [10]

1.6.1 VI Server

VI server je sada různých funkcí v prostředí LabVIEW, které slouží k dynamickému ovládání objektů umístěných na front panelu. Všechna VI mají své vlastnosti, které lze číst nebo nastavovat a metody, které lze zavolat použitím funkcí nabízených VI Serverem. [11]

2. NÁVRH PRAKTICKÉ ČÁSTI

Tato kapitola slouží k seznámení se základní myšlenkou funkčnosti a postupem návrhu programu. Slouží také jako určitá forma návodu k obsluze programu. Jsou zde vyobrazeny tři různé návrhy, pro demonstraci průběhu vývoje a znázornění myšlenkových postupů. Každý návrh obsahuje popis základní myšlenky s důvodem jeho vzniku a následný princip již implementovaného návrhu, který se s každým dalším návrhem rozšiřoval.

2.1 Návrh prvního řešení

Cílem je vytvořit program pro provedení zvolené operace, jejíž výsledek bude zapsán pomocí pevné řádové čárky. Pro vstup je potřeba zadat numerické vstupní hodnoty, zatím se jedná jen o operace se dvěma vstupy, vybrat základní operaci pro zadávané vstupy a vybrat počty dostupných bitů, podle kterých se mají vstupy a výsledek dané operace upravit. Výstupem je výsledek operace, převedený na zapsatelnou hodnotu.
Seznam dostupných operací:

- Sčítání
- Odečítání
- Násobení
- Dělení

2.1.1 Princip prvního řešení

Obrázek 8: Vývojový diagram prvního návrhu

Po zadání potřebných vstupních parametrů (numerické vstupní hodnoty, výběru jedné ze základních operací (sčítání, odčítání, násobení, dělení) a počty dostupných bitů pro vstupní hodnoty a pro výstupní), se upraví vstupní numerické hodnoty na hodnoty zobrazitelné, což je dáno počty dostupných bitů pro vstupní hodnoty a následně se s nimi provede zvolená operace. Výsledek operace projde úpravou, opět na zobrazitelnou hodnotu, což je dáno počtem dostupných bitů pro výstup.

Toto řešení má mnoho nevýhod, jako je nedostatek dostupných operací, volba overflow módu a způsobu zaokrouhlení nebo absence informací o chybě, tudíž bude nutné na tuto práci navázat a přidat různá rozšíření. Výhodou je vytvoření prvních potřebných částí kódu a udání základního směru, kterým se budou další řešení ubírat.

2.2 Návrh druhého řešení

Jelikož je první návrh příliš obecný a nedostatečný co se týče potenciálu a funkcí, vznikl tento rozšířený návrh. Nový program je více benevolentní k požadavkům uživatele. Vstup se nyní neskládá jen z numerických hodnot, operace a počtu dostupných bitů vstupů a výstupu, ale má na vstupu mnohem více kontrolovaných veličin jako je například způsob zaokrouhlení, nebo reprezentace znaménka.

Prvním velkým rozšířením je možnost volby ze dvou režimů (obecného a konkrétního). Výběrem režimu se změní nejen provedení operací a výstup, ale také množství a typ vstupních dat. Hlavní rozdíl mezi těmito režimy je že konkrétní spočívá v provedení operace pouze jednou s jedním nebo dvěma vstupy, což se odvíjí od typu zvolené operace, zatímco obecný pracuje s poli hodnot a ve většině případů se provede operace několikrát. Pole hodnot pro obecný režim uživatel určí pomocí minimální a maximální hodnoty a velikosti kroku, který reprezentuje rozdíl po sobě jdoucích hodnot a operaci provádí pro každou kombinaci vstupních hodnot. Počet výsledků obecného režimu bude roven součinu velikostí vstupních polí v případě dvou vstupů (sčítání, násobení, …) a roven velikosti vstupního pole v případě operace s jedním vstupem (převrácená hodnota, mocnina…). Variantou pro zadání vstupu obecného režimu je výběr předdefinovaného datového typu (U8, I16, …), místo zadávání dostupných bitů, což odpovídá datovému typu fixed point. Oba režimy vyžadují volbu overflow módu, způsob zaokrouhlení a typ prováděné operace. Vstupní parametry pro konkrétní režim jsou numerické hodnoty, dostupné bity pro zápis vstupů a výstupu spolu s reprezentací znaménka. Pro obecný režim uživatel vybírá minimální a maximální hodnotu s velikostí kroku, opět záleží, jestli operace vyžaduje jeden nebo dva vstupy a počet dostupných bitů pro výstupní hodnoty. Výstupní parametry se liší podle zvoleného režimu. Konkrétní režim se zaměřuje více na pomoc uživateli formou informování o vhodnější volbě dostupných bitů pro zápis, zatímco obecný režim graficky vyobrazí průběh absolutní chyby. Pro oba režimy platí nutnost hlídat chybové stavy a v případě jejich výskytu se musí zastavit vykonávání výpočtů.

Seznam dostupných datových typů:

- U32, U16, U8
- I32, I16, I8
- Double, Single
- Fixed point

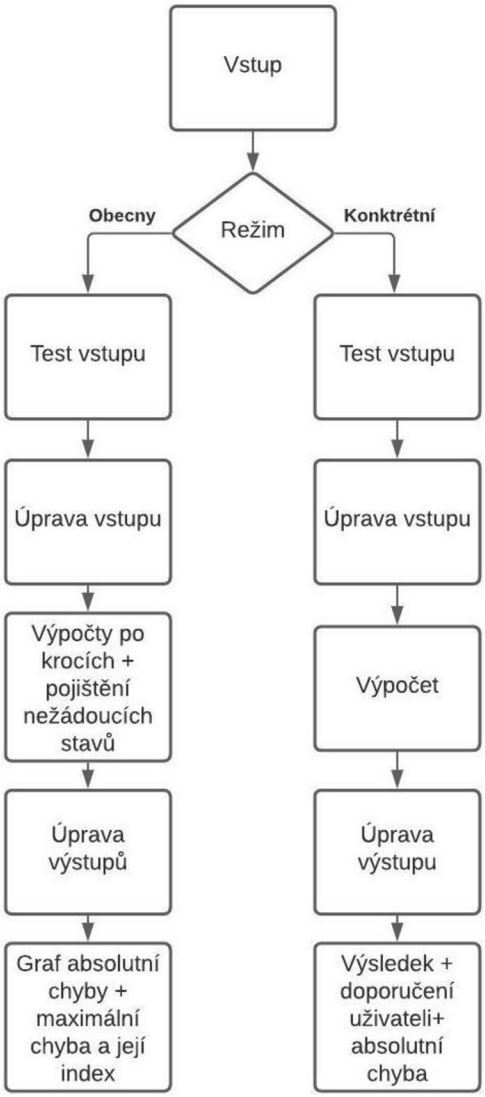
Seznam dostupných operací:

- Základní operace (Sčítání Odečítání Násobení Dělení)
- Kvadratická funkce (Převrácená hodnota, Druhá mocnina, Odmocnina)
- Goniometrické operace (Sinus, Cosinus, Tangens, Cotangens)

Seznam chybových stavů:

- Dělení nulou
- Převrácená hodnota nuly
- tangens π
- cotangens 2*π

2.2.1 Princip druhého řešení



Obrázek 9: Vývojový diagram druhého návrhu

Všechny vstupní parametry, které jsou uvedeny jako první na Obrázek 9, jsou popsány v kapitole 2.2. Po zadání potřebných vstupních parametrů, se vykoná jeden ze dvou režimů (obecný nebo konkrétní), přičemž režim si volí uživatel. Jako první se provede test vstupních parametrů, jestli by při chodu programu nedošlo k chybovému stavu, seznam chybových stavů je opět uveden v kapitole 2.2. Test se provádí pro oba režimy. Následuje úprava vstupních numerických hodnot podle dostupných bitů, opět se jedná o krok pro oba režimy, jen pro každý z nich dochází k úpravě jiným způsobem. Pro konkrétní režim se zvolené numerické vstupní hodnoty upravují na základě zvolených dostupných bitů pro zápis a tato akce se provede pouze jednou. V případě obecného režimu se pracuje s poli hodnot určitého datového typu, tudíž jediná úprava, která se zde provádí je pomocí minimální a maximální hodnoty, která musí odpovídat příslušnému datovému typu, v případě že jsou nastaveny tyto hodnoty mimo jeho rozsah, je uživatel informován pomocí vyskakovacího okna a minimum nebo maximum je upraveno na validní hodnotu. Dále se vykonají operace s numerickými hodnotami. Následuje úprava výstupu nebo výstupů podle vstupních parametrů, které jsou přímo spjaty s výstupem.

U konkrétního režimu je výstup definován počtem bitů, u obecného datovým typem. V posledním kroku dojde k vyobrazení výsledků, opět se jedná o různé typy výsledných hodnot pro oba režimy.

Výstup obecného typu je graf absolutních chyb, hodnota a index maximální chyby, pole absolutních chyb, výsledků typu double a výsledů podle zvoleného nastavení dostupných bitů. Výstup konkrétního režimu tvoří doporučení o nastavení vstupních parametrů, výsledek typu double a výsledek zapsaný pomocí dostupných bitů a hodnota absolutní chyby.

Řešení provedené v kapitole 2.2 je výhodné z důvodu velkého počtu nových implementací, které uživateli nabízí rozšířené možnosti pro vykonávání operací. Nevýhodou je velké a nepřehledné množství vstupních parametrů, které musí uživatel zadávat a nutnost předem zvolit vykonávaný režim. Velkou nevýhodou je také absence možnosti zřetězení operací za sebou, lze vždy vykonat jen jeden typ operace.

2.3 Návrh finálního řešení

Při hledání vhodnější formy zadávání vstupních parametrů a zároveň omezení množství, které musí zadávat uživatel, se jako nejvhodnější jeví využití funkcí VI Serveru. Finální řešení bude mít díky využití funkcí VI Serveru automatizované vyplnění vstupních údajů, neboť k provedení určité posloupnosti výpočtů stačí vytvořit jednoduché vzorové VI, které bude obsahovat controly, konstanty a operace, na základě jejich propojení a nastavení datových typů se budou vykonávat v kódu tyto operace.

Je nutné dodržet povolené nástroje při tvorbě vzorového VI. V případě, kdy uživatel použije ve vzorovém VI nepodporovaný objekt, jako je například for smyčka nebo neimplementovaná operace, nebude vytvářený nástroj funkční a může vést k zacyklení nebo vyhlášení chyby.

Seznam povolených objektů k použití je uveden níže v této kapitole. Kompletní seznam vstupních i výstupních parametrů je vyobrazen a popsán v kapitole 2.4.1 a 2.4.2. Díky omezení počtu uživatelem zadávaných vstupních parametrů se také limituje pravděpodobnost vzniku chyby při jejich vyplňování. Uživateli tak stačí vytvořit toto VI a následně na uživatelském rozhraní zadat pro všechny controly minimální a maximální hodnotu s velikostí kroku, tak jako tomu bylo u předchozího řešení a program spustit. Kód je rozdělen na dvě hlavní části, jedna se soustředí na práci s referencemi za účelem získat potřebná data ze vzorového VI a druhá část se soustředí na matematické výpočty a provedení převodů datových typů. Výstupem je grafické vyobrazení absolutních chyb všech operací, informace o maximální vzniklé absolutní chybě, hodnoty absolutních a relativních chyb pro všechny kombinace vstupních hodnot a hodnoty typu double a zvoleného datového typu ve vzorovém SubVI.

Seznam dostupných datových typů:

- U64, U32, U16, U8
- I64, I32, I16, I8
- Double, Single
- Fixed point

Seznam dostupných operací:

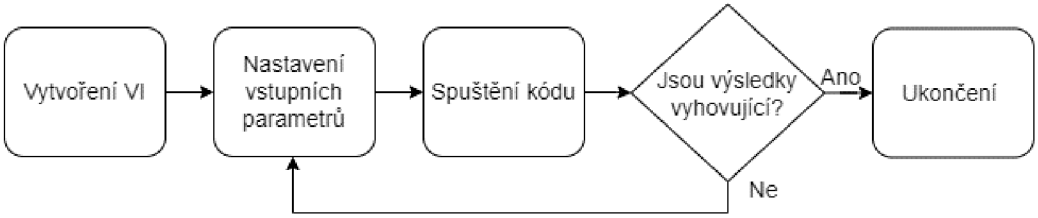
- Základní operace (Sčítání, Odečítání, Násobení, Dělení)
- Kvadratická funkce (Převrácená hodnota, Druhá mocnina, Odmocnina)
- Exponenciální funkce (Exponenciála)
- Goniometrické operace (Sinus, Cosinus, Tangens, Cotangens)
- Přirozený logaritmus
- To Double Precision Float

Seznam chybových stavů:

- Odmocnina ze záporného čísla
- Dělení nulou
- Převrácená hodnota nuly
- Tangens π
- Cotangens $2*\pi$
- Nekladné logaritmované číslo

2.3.1 Princip finálního řešení

- **Postup uživatele:**

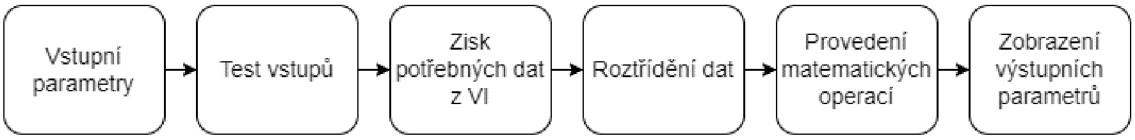


Obrázek 10: Vývojový diagram s postupem uživatele

Prvním krokem by vždy mělo být vytvoření vzorového VI, které se podrobí rozboru. Toto VI může obsahovat kombinace různých matematických operací a controlů či konstant různých datových typů. Příklad, jak může vypadat takové VI je v kapitole 4. Uživatelovo nastavení datových typů pro všechny controly a operace se následně přenáší do hlavního kódu. Seznam dostupných operací a datových typů je uveden v kapitole 2.3 a dále v dokumentu přímo u SubVI *Vypocetni_SubVI.vi*, které tyto operace vykonává. Jakmile je VI vytvořeno, je potřeba na uživatelském rozhraní nastavit vstupní parametry, jenž definují podstatnou část chodu programu.

Vstupními parametry jsou cesta ke vzorovému VI, rozsah vstupních hodnot, určený pomocí minimální a maximální hodnoty jednotlivých controlů spolu s velikostí kroku, která představuje rozdíl mezi po sobě jdoucími hodnotami v poli, které je pomocí těchto parametrů vytvářeno. Dalším, již nepovinným, vstupním parametrem je nastavení overflow módu a způsobu zaokrouhlení, v případě že není uživatelem zvolen, je nastaven na výchozí hodnoty, kterými jsou saturace a Round Half to Even. Posledním nepovinným vstupním parametrem je změna výstupního datového typu operací v případě, že uživatel není spokojen s nastavením těchto typů ve vzorovém VI, ale hlavním účelem je nastavení výstupního datového typu u goniometrických operací (sinus, cosinus, tangens a cotangens), logaritmu a exponenciála, neboť u těchto operací tuto volbu ve vzorovém VI provést nelze. Po vyplnění a následném spuštění programu jsou zobrazeny výstupní parametry, u kterých musí uživatel vyhodnotit, zda jsou přípustné nebo nikoli, v takovém případě musí změnit nastavení vstupních parametrů a kód spustit znovu. Je třeba si uvědomit, že výstupním parametrem může být i hlášení o chybě z důvodu špatného nastavení uživatelem nebo vytvořením nefunkčního vzorového VI. Jakmile je uživatel spokojen s výsledky, tudíž velikost absolutní chyby nepřekročila určitou hodnotu, může program ukončit, nebo pokračovat v testování pomocí jiného VI.

- **Princip fungování:**



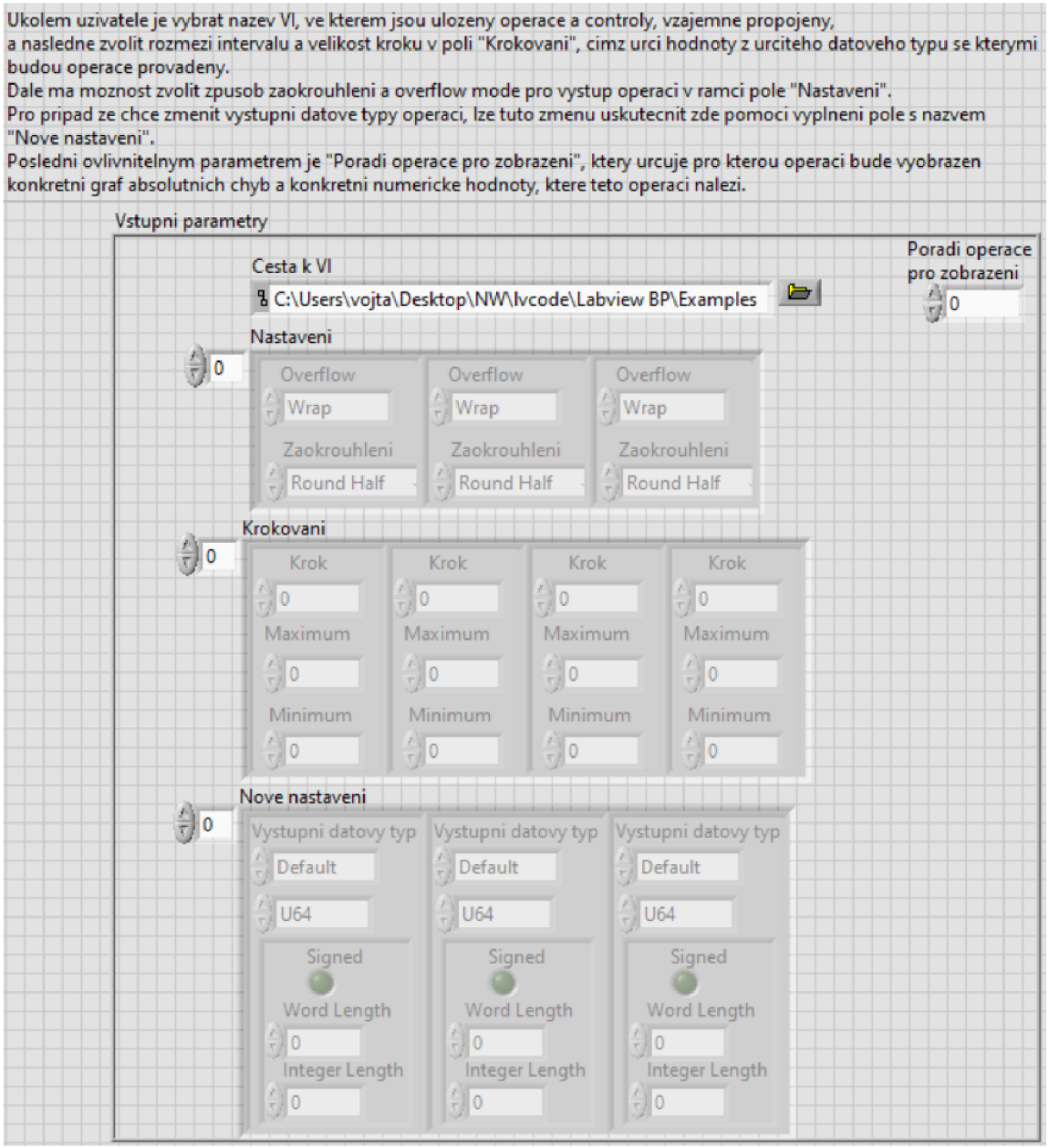
Obrázek 11: Vývojový diagram fungování finálního návrhu

Po zadání všech vstupních parametrů a spuštění programu, dojde k otestování jejich validity. V případě že nejsou zadané parametry v pořádku, kód se neprovede a dojde k vypsání chyby s informací, proč k jejímu vzniku došlo. V opačném případě následuje rozbor zadaného vzorového VI pomocí SubVI, využívajícího funkce VI Serveru za účelem získání všech potřebných dat, se kterými je dále v kódu pracováno, jako jsou například datové typy controlů nebo reference na všechny vstupy a výstupy operací. Díky využití funkcí VI Serveru se značně zjednoduší zadávání operací spolu s datovými typy, nevýhodou ale je, že nelze vyčíst všechny informace. Ze vzorového VI nelze vyčíst informace o způsobu zaokrouhlení a overflow mód. Snaha nalézt způsob, jak vyčíst informace o tomto nastavení nebyla úspěšná, ale aby uživatel měl možnost tyto parametry ovlivnit, je na uživatelském rozhraní alternativní řešení, které když není využito, tak se pro operace nastaví výchozí nastavení.

Tyto informace jsou dále rozděleny a roztřízeny podle jejich významu pro fungování kódu, jak lze vidět na Obrázek 11. Hlavní rozdělení probíhá na dvě části, jedna část obsahuje informace spojené s referencemi a slouží k vytvoření struktury a posloupnosti operací a druhá část obsahuje již informace potřebné pro numerické výpočty a jejich nastavení. Po roztřízení probíhá již vykonání matematických operací, což je ovlivněno vstupními parametry, které zadává uživatel. Jakmile jsou všechny operace vykonány, jsou vyobrazeny všechny výstupní parametry, na základě kterých, probíhá rozhodnutí či diskuze o jejich přijatelnosti v rámci dané situace.

2.4 Uživatelské rozhraní

2.4.1 Vstupní parametry



Obrázek 12: Uživatelské rozhraní – vstupní parametry

Nyní uživatel nemusí zadávat takové množství vstupních parametrů, ani přepínat mezi režimy, pouze si vytvoří vlastní vzorové VI, kde nadefinuje požadované operace a jejich vstupy a následně provede na uživatelském rozhraní nastavení několika vstupních parametrů. Uživatel má povinnost nastavit tři vstupní parametry, ostatní parametry může nastavit podle aktuálních potřeb.

Prvním povinným parametrem je cesta k tomuto VI, ze které se v kódu vyčítá i název tohoto VI, což je vidět v levém horním rohu na Obrázek 12. Dalším povinným parametrem je vybrání rozsahu vstupních hodnot a velikosti kroku, který definuje rozdíl po sobě jdoucích hodnot v poli vstupních hodnot, které se na základě tohoto nastavení vytvoří. Nastavení rozsahu a kroku se provádí za účelem vytvoření pole hodnot, aby se operace provedla více než jednou a zároveň aby se provedla v přijatelném čase, neboť provádět operace se všemi hodnotami, které jsou dostupné pro určité datové typy by bylo nereálné. Uživatel tedy nastavením vybírá část hodnot datového typu, která se použije jako vstup pro operaci. Toto nastavení je nutno provést pro každý zadaný control aby bylo dosaženo správné funkčnosti programu, jinak by byly automaticky nenastavené controly vyplněny nulami místo pole několika hodnot. Jako nápověda pro správné nastavení controlů, je ve výstupních parametrech dostupný seznam jmen controlů ve stejném pořadí, v jakém pro ně bude probíhat krokování.

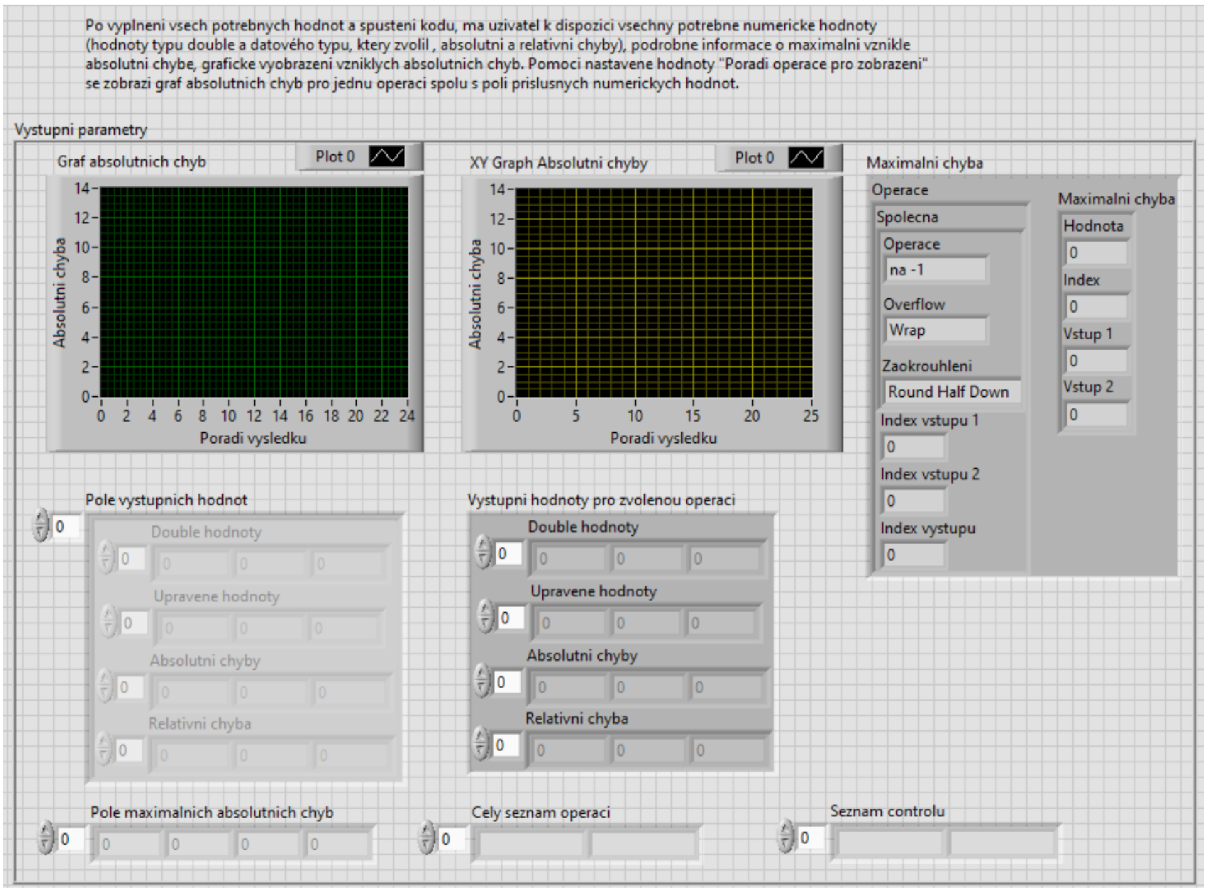
Dalším parametrem, u kterého záleží pouze na uživateli, zda ho využije, je změna výstupního datového typu operace, v případě že nezmění výchozí nastavení na nastavení, které vyvolá změnu, což řídí první enum v clusteru nabízející dva stavy, první ponechá původní datový typ a druhý který přiřadí datový typ nově zvolený, zůstane datový typ u operace stejný jako je ve vzorovém VI. Toto nastavení změní datový typ pouze pro výpočty v tomto programu, nijak nezasahuje do vzorového VI a jeho hlavním účelem je vytvoření způsobu, jak změnit výstupní datový typ goniometrických operací (sinus, cosinus, tangens a cotangens), logaritmu a exponenciály, jejichž výstupní datový typ ve vzorovém VI nelze změnit.

V posledním poli lze nastavit pro každou operaci overflow mód a způsob zaokrouhlování. Overflow mód může být nastaven na běžné typy, kterými jsou wrap a saturace, jejichž význam byl rozebrán v rámci teoretického úvodu. Způsob zaokrouhlení opět nabývá klasických typů Round Half to Even, Round Half Down neboli Truncate a Round Half Up. Způsob zaokrouhlení, ani overflow mód nemusí být zadávány, ale jsou následně nastaveny na výchozí hodnoty, kterými jsou saturace a Round Half to Even. Toto nastavení je zde z důvodu, že nebyl nalezen způsob jak tuto část nastavení vyčíst pomocí funkcí VI Server.

Poslední ovladatelnou hodnotou na uživatelském rozhraní je control, který udává, pro kterou operaci bude zobrazit samostatný graf absolutní chyby a výstupní numerické hodnoty pouze pro její výstup.

Jako nápověda pro nastavení posledních dvou parametrů je ve výstupních parametrech uvedeno pole operací v takovém pořadí, v jakém jsou vykonávány, přičemž zadaná hodnota odpovídá vždy indexu prvku v poli. Také si lze všimnout že nad rámečkem, který ohraničuje vstupní parametry, se pro uživatele nachází krátký návod s instrukcemi o vyplnění vstupních.

2.4.2 Výstupní parametry



Obrázek 13: Uživatelské rozhraní – výstupní parametry

Po vytvoření různých rozšíření v kódu a provedení změn u prvního řešení, jsou nyní k dispozici dva grafy znázorňující vývoj absolutní chyby. První z grafů vyobrazuje vývoj absolutní chyby postupně pro všechny výstupy operací v pořadí, v jakém jsou vykonány. Druhý graf je řízen controlem, který již byl zmíněn při popisu vstupních parametrů. Na základě zvolené hodnoty se zde zobrazí graf absolutní chyby pro výstup jedné operace.

Dále se zobrazuje pole clusterů, přičemž každý cluster obsahuje čtyři další pole hodnot. Jedná se o pole výsledků operace typu double, následně pole s výsledky s datovým typem, který zvolil uživatel ve vzorovém VI, nebo při změně výstupních datových typů. Tímto převodem může vzniknout absolutní a relativní chyba, jejichž velikosti jsou vyobrazeny v posledních dvou polích.

Pomocí zadávaného controlu indexu, který je zobrazen na Obrázek 12, lze znovu, stejně jako u grafu, zobrazit cluster se čtyřmi poli pouze pro konkrétní operaci, zjednodušeně řečeno se jedná o vytažení prvku z pole clusterů na základě tohoto indexu.

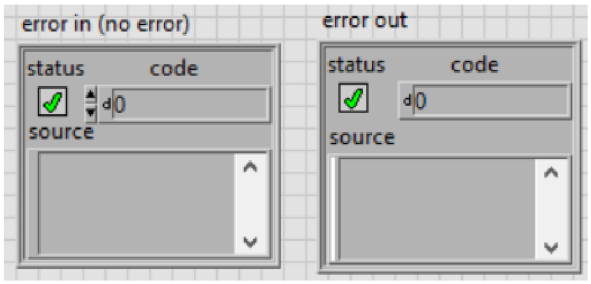
S volbou hodnoty tohoto controlu může pomoci cluster *Maximalni chyba*, kde jsou uloženy veškeré informace o maximální absolutní chybě, která vznikla během provádění operací. Z clusteru se uživatel dozví hodnotu této chyby, jaký je její index v celkové posloupnosti všech výsledků všech operací, o jakou se jednalo operaci včetně jejího nastavení (overflow mód a způsob zaokrouhlení) jaké jsou indexy vstupních hodnot spolu s jejich hodnotami a index výstupu, který určí pořadí výsledku v rámci operace, u které došlo k chybě.

Informace o maximální absolutní chybě u každé operace je uložena v poli Pole maximálních absolutnich chyb, největší z těchto hodnot je pak uváděna v clusteru coby maximální chyba celého výpočtu.

Posledními výstupními prvky jsou dvě pole, seznam prováděných operací a seznam s názvy controlů ze vzorového VI, tyto pole slouží jako asistence při nastavování vstupních parametrů.

Stejně jako u vstupních parametrů, je i zde nad ohraničujícím rámečkem stručný popis o výstupních parametrech pro uživatele.

Mimo vstupní a výstupní parametry jsou ještě uvedeny informace o chybách, kde se uživatel, v případě že se dopustí chyby, o této situaci dozví.



Obrázek 14: Uživatelské rozhraní – hlášení o chybě

3. VYUŽITÍ LABVIEW – REALIZACE PRAKTICKÉ ČÁSTI

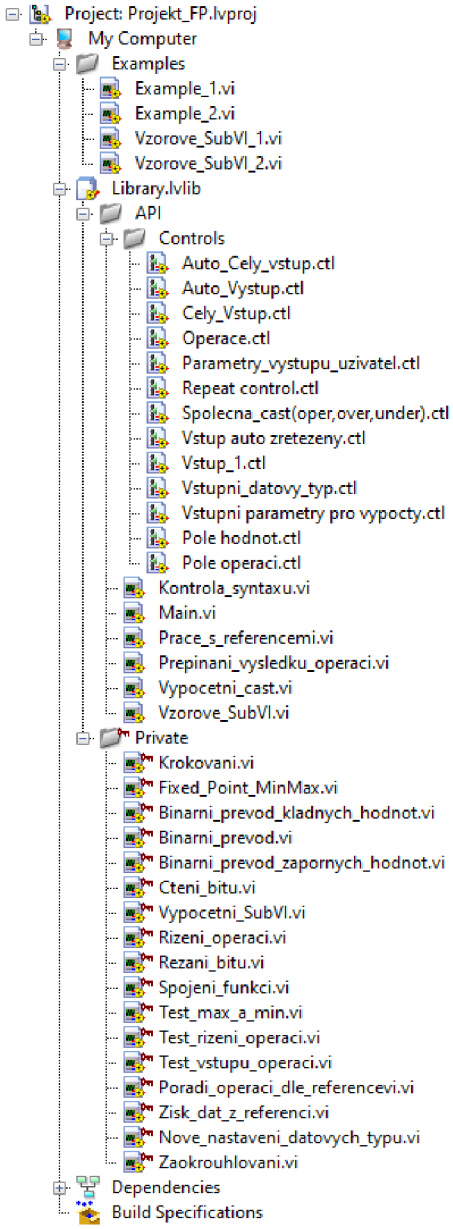
Tato kapitola se věnuje popisu praktické části, konkrétně použitým SubVI, pomocí kterých byl celý program zrealizován, neboť uvádět celý kód včetně všech zapojení by mohlo být nepřehledné a zmatené, rozboru knihovny a popisu testování funkčnosti programu. Některé SubVI pro své fungování využívají funkcí jiných SubVI. V případech, kdy bylo výsledkem zpřehlednění programu, byly použity pro vedení dat clustery. Za účelem získání potřebných informací se využívá funkcí VI Scriptingu, tyto funkce je ale potřeba aktivovat, aby bylo možné jejich použití. Aktivace probíhá v následujících krocích:

- V LabVIEW klikněte na záložku *Tools*
- Klikněte na položku *Options*
- Objeví se vám nové dialogové okno, přičemž vlevo je seznam kategorií, kde kliknete na kategorii *VI Server*
- Nyní v kolonce *VI Scripting* zaškrtnete *Show VI Scripting functions, properties and methods*
- Tímto máte aktivovány funkce VI Scriptingu, jako je například funkce Traverse

Využité nástroje:

- LabVIEW 2021
- FPGA module
- VI Scripting

3.1 Popis knihovny



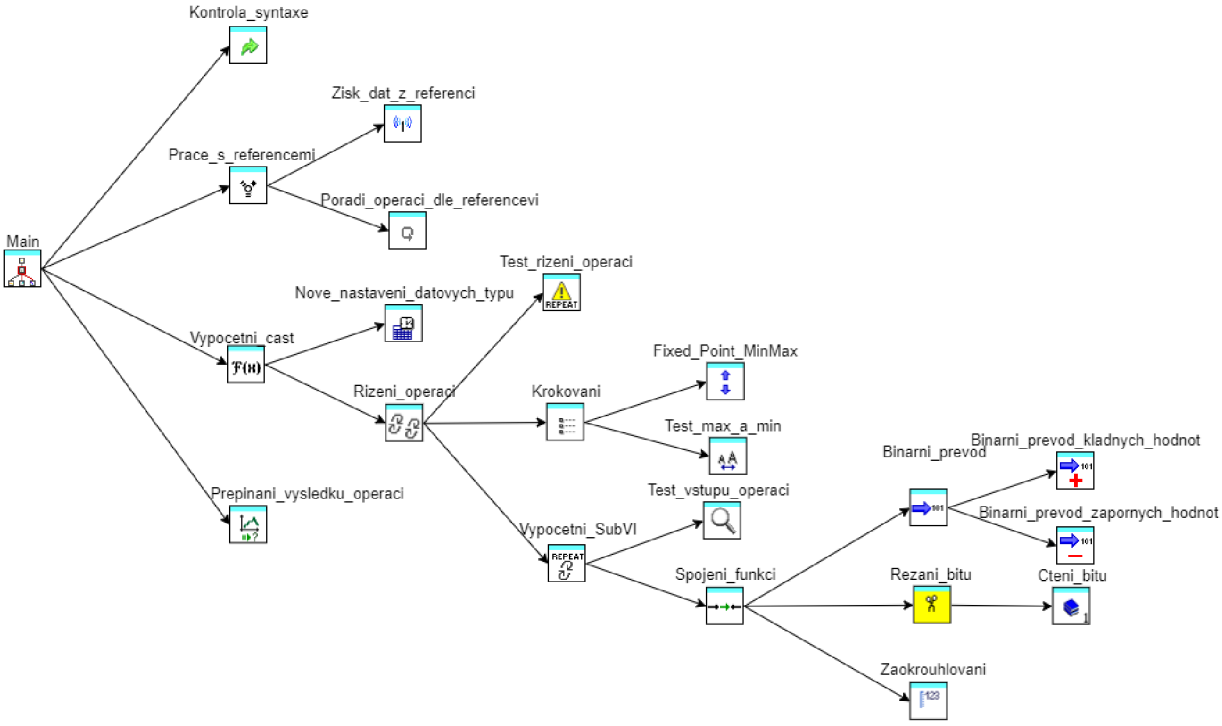
Obrázek 15: Praktická část – Projekt v LabVIEW

Knihovna se skládá ze dvou základních složek, z *API* a *Private*. Soukromá složka *Private* obsahuje VI a SubVI, které představují funkce knihovny, jež jsou využívány ve VI ve veřejné složce *API*. VI a SubVI uloženy ve složce *API* slouží ke spojení a propojení dostupných funkcí za účelem dosažení požadovaného výsledku. Ve složce *API* se nachází podsložka Controls, která obsahuje controly využívané v celém programu.

Mimo knihovnu se nachází složka *Examples*, její hlavní význam tkví v uložení ukázkových příkladů ověřující funkci programu do jejího prostoru. Dále slouží mimo jiné i k ověření přístupu neboli zda jsou všechny VI, které musí být ve složce *Public*, skutečně v ní.

3.2 Princip implementace a fungování programu:

Pro větší přehlednost a snažší pochopení následující kapitoly je zde vyobrazená struktura vytvořených VI a SubVI viz Obrázek 16, která názorně ukazuje vzájemnou závislost jednotlivých částí. Čtenář tak snáze pochopí funkčnost SubVI a způsob předávání jejich vstupních parametrů.

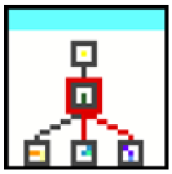


Obrázek 16: Struktura vytvořených VI a SubVI

3.2.1 Popis využitých VI a SubVI

Všemi uvedenými VI a SubVI prochází errorové dráty za účelem odhalení případné chyby a včasného přerušení kódu. Při popisování polí s prvky je míněno 1D pole, pokud není přímo řečeno jinak. Všechny SubVI jsou popsány v pořadí, v jakém jsou v kódu využívány.

- **Main.vi**



Obrázek 17: Ikona pro VI Main.vi

Hlavní částí programu je *Main.vi*, který představuje pomyslný vršek pyramidy a pro dosažení požadované funkce kombinuje funkce jiných VI. Jeho obsah je již popsán v kapitole 2.3 a 2.4. Vstupní parametry zadávané uživatelem, které se zadávají uživatelem v tomto VI, jsou všechny posílány dále do čtyř následujících SubVI, které s těmito vstupy pracují. Všechny vstupní parametry, které uživatel může nebo musí nastavit, byly již popsány v rámci uživatelského rozhraní. Probíhá zde vyčtení názvu vzorového VI, který se posílá jako vstup do využívaných SubVI.

- **Kontrola_syntaxe.vi**

Obrázek 18: Ikona pro SubVI Kontrola_syntaxu.vi

Zde se kontroluje, zda je vzorové VI uživatelem skutečně vytvořeno a zda neobsahuje nějakou chybu, která by znemožnila jeho spuštění. V případě, že nastane chyba, vypíše se o ni informace a zbytek kódu se nevykoná. Příkladem takové chyby je nepřipojení vstupu k operaci, což má za následek, že nemá potřebné hodnoty k jejímu vykonání. Vstupními parametry jsou název vzorového VI spolu s cestou k místu, kde je toto VI uloženo. Výstupním parametrem je informace, zda lze pokračovat v kódu či nikoliv, v podobě errorového výstupu, všechny následující SubVI jsou tedy závislé na tomto SubVI.

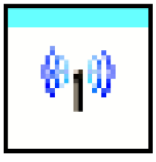
- **Prace_s_referencemi.vi**

Obrázek 19: Ikona pro SubVI Prace_s_referencemi.vi

Toto SubVI slouží ke sjednocení veškeré prováděné práce a manipulace s referencemi na jedno místo. Odlišují se zde od sebe datové typy controlů a indikátorů a sjednocují se k sobě jednotlivé informace, jenž slouží jako vstupní parametry použitým SubVI. Jeho hlavní funkci obstarávají dvě další SubVI, *Poradi_operaci_dle_reference.vi* a *Zisk_dat_z_referenci.vi.*

Vstupními parametry jsou opět název VI spolu s cestou. Výstupními parametry jsou pole řetězců a cluster. Pole řetězců obsahuje kompletní seznam operací, které uživatel použil ve svém vzorovém VI a jsou zde uvedeny v pořadí, v jakém jsou vykonány. Pole řetězců poskytuje důležitou informaci pro uživatele, neboť v rámci *Vypocetni_cast.vi* uživatel má možnost nastavit a ovlivnit výstup jednotlivých operací, což by bez znalosti jejich pořadí nešlo. Cluster obsahuje informace nezbytné k vykonání numerických operací dále v kódu. Obsahem clusteru jsou pole hodnot, kterých nabývají konstanty ve vzorovém VI, jsou-li tam použity, počet controlů, bitová reprezentace určující reprezentaci znamének a počty bitů pro controly, indikátory a výstupy operací. Dalším parametrem obsaženým v clusteru je již zmíněné pole s názvy operací, ale tentokrát se jedná o pole clusterů, přičemž každý cluster obsahuje pole řetězců. Tento přístup je zvolen z důvodu, že se v kódu pracuje s funkcí *array size*. Funkce *array size* určí velikost pole a v případě využití 2D pole řetězců, by měl každý řádek stejný počet sloupců, což by mělo nežádoucí následky, zatímco s využitím clusterů tato situace nenastane. Dále je v něm uložen cluster, který obsahuje dvě 2D pole s indexy, které následně určují vstupní hodnoty operací, pole datových typů controlů a 2D pole datových typů výstupů, přičemž zde může být využito 2D pole, neboť všechny operace mají dva nebo tři datové typy, jeden výstupní a jeden nebo dva vstupní.

- **Zisk_dat_z_referenci.vi**



Obrázek 20: Ikona pro SubVI Zisk_dat_z_referenci.vi

Toto SubVI se využívá k získání potřebných hodnot ze vzorového VI. Pomocí funkce Traverse, která je dostupná díky VI Scriptingu, se zde získávají reference a téměř všechny informace o všech operacích, controlech, konstantách a indikátorech.

Vstupními parametry pro *Zisk_dat_z_referenci.vi* jsou název vzorového VI a cesta k tomuto VI. Výstupními parametry jsou tři clustery. První cluster obsahuje pole s hodnotami využívaných konstant ve vzorovém VI spolu s referencí na tyto konstanty. Druhý cluster, který reprezentuje controly a indikátory, obsahuje reference na tyto prvky, jejich datové typy včetně rozsahu a velikosti kroku, cluster s reprezentací znaménka a počty dostupných bitů, informaci, zda se jedná o control nebo indikátor v podobě boolean indikátoru a pole numerických hodnot, které jsou uloženy v controlech a indikátorech, toto pole se dále v kódu využívá pro určení počtu controlů. Třetí cluster obsahuje informace o prováděných operacích. Součástí clusteru je pole používaných operací, 2D pole datových typů (vstupní a výstupní datové typy operací), cluster s reprezentací znaménka a počty dostupných bitů a cluster s rozsahem a velikostí kroku pro výstupní datový typ a 2D pole referencí na vstupy a výstupy operací.

- **Poradi_operaci_dle_reference.vi**

Obrázek 21: Ikona pro SubVI Poradi_operaci_dle_reference.vi

Toto SubVI funguje jako simulace datového toku a využívá se k určení pořadí operací, v jakém se budou, neboť je důležité, aby se operace provedla až poté, co se provedou operace předcházející v případě, že využívá výstup některé z těchto operací jako svůj vstup, což se děje u všech složitějších výpočtů. Princip určení pořadí a zároveň i určení vstupních parametrů operace spočívá v porovnávání referencí controlů, konstant, vstupů a výstupů operací v kombinaci s využitím informace o potřebném počtu vstupních parametrů operace. Jakmile je určeno pořadí všech operací, je toto SubVI ukončeno. Výstupními parametry jsou čtyři pole, dvě pole, které dále určují indexy vstupních hodnot do operací a dvě pole s názvy všech prováděných operací, v takovém pořadí, v jakém jsou vykonávány, přičemž jedno pole obsahuje přímo názvy operací a jde na výstup uživatelského rozhraní a druhé pole obsahuje clustery s polem, které pokračují dále do kódu a určují pořadí vykonání operací. Dvě pole s názvy a pořadím operací jsou zde proto, že jedna varianta je přehlednější pro uživatele a druhá se lépe uplatní v kódu při vykonávání operací v určitých cyklech.

- **Vypocetni_cast.vi**

Obrázek 22: Ikona pro SubVI Vypocetni_cast.vi

V tomto SubVI se na základě informací z referencí, získaných v rámci SubVI *Prace_s_referencemi.vi*, jehož výstupní cluster je zároveň vstupním parametrem pro toto SubVI, provádí postupně jednotlivé operace. Dalšími vstupními parametry jsou tři pole, které zadává uživatel v rámci *Main*.vi. První pole obsahuje cluster s nastavením režimu zaokrouhlení spolu s overflow módem, v případě, že uživatel toto nastavení nevyplní, kód se provede s nastavením, které mají operace jako defaultní nastavení, což, jak bylo zmíněno v kapitole 2.4.1, je saturace, coby overflow mód a Round Half to Even jako režim zaokrouhlení. Tento způsob nastavení je zvolen z toho důvodu, že s využitím funkcí VI Serveru, nebyl nalezen způsob, jak z VI získat informaci o tomto nastavení operací. Druhým polem je nastavení krokování, kde uživatel nastaví velikost intervalu, ze kterého se budou čerpat numerické hodnoty. Velikost intervalu se definuje pomocí zadání minima a maxima, přičemž se toto nastavení vztahuje na vstupní control s nastaveným datovým typem ve vzorovém VI. V případě, že je nastaveno minimum nebo maximum, které neodpovídá rozsahu příslušného datového typu, je interval omezen na základě vlastností tohoto typu a pro uživatele vyskočí hlášení o této změně. Jedná se tedy o interval, jehož horní i spodní hranice odpovídá příslušnému datovému typu.

Poslední pole obsahuje cluster, který slouží ke změně výstupních datových typů operací přímo z uživatelského rozhraní a nemusí tak nutně měnit nastavení ve vzorovém VI. Tento cluster obsahuje dva controly typu enum a další cluster, první enum nabízí dvě varianty, které určí, zda se toto nastavení má aplikovat na operaci nebo ne, jelikož se jedná o pole, tak je tímto zajištěno, že aplikování může proběhnout jen na zvolené operace, nastavení hodnoty default zařídí ponechání aktuálních výstupních datových typů, zatímco nastavení na parametr zmenit vystupni datovy typ, nastaví vybraný výstupní datový typ podle volby hodnoty v druhém enumu. Druhý enum obsahuje seznam dostupných datových typů, přičemž pokud je zvolen datový typ fixed point, je nutné vyplnit již zmíněný cluster. V tomto clusteru se nastavuje reprezentace znaménka na signed nebo unsigned a počty dostupných bitů celkem a počty bitů pro celou část. V případě že je zvolen jiný datový typ než fixed point, bitové nastavení v clusteru nehraje roli. Je důležité dbát na správné pořadí v poli při vyplňování, neboť každý prvek pole odpovídá jedné operaci v pořadí, v jakém jsou vykonávány. Pomocí dalších SubVI se zde postupně vykonávají operace s numerickými hodnotami a následně se určují výstupní parametry.

Výstupními parametry *Vypocetni_cast.vi*, které jsou předávány interně z jiných SubVI, jsou dva grafy absolutních chyb pro všechny operace, pole maximálních chyb, za každou operaci je zde uložena jedna hodnota, pole výstupních hodnot a cluster s informacemi o maximální chybě. Pole výstupních hodnot obsahuje jeden cluster za každou provedenou operaci. V každém takovém clusteru jsou uloženy čtyři pole. Pole double hodnot, které vyjadřuje výstup z operace zapsaný pomocí typu double. Pole upravených hodnot, které vyjadřuje výstup z operace zapsaný pomocí typu, který byl nastaven ve vzorovém VI nebo byl nastaven uživatelem v uživatelském rozhraní. Zbylé dvě pole vyjadřují absolutní chyby a relativní chyby. Všechny tyto pole mají stejný počet prvků, jejich počet je určen na základě velikostí polí, se kterými jsou operace prováděny. Detailnější popis provádění operací bude v rámci SubVI *Rizeni_operaci.vi*. Cluster s informacemi o maximální chybě poskytuje uživateli podrobné informace o tom, jak k této chybě došlo. Z clusteru lze vyčíst o jakou operaci se jedná, jaká je její numerická hodnota, jaké je nastavení zaokrouhlení a overflow módu, jaké jsou numerické hodnoty vstupů, které vstupují do operace, jaký je index vstupů, které určují, které vstupní datové typy byly použity (všechny datové typy, jak vstupní, tak i výstupní jsou uloženy v poli) a index maximální chyby, který určí o kolikátou kombinaci vstupních hodnot se jedná. Cluster s těmito informacemi slouží pro uživatele jako nápověda, kde by měl provést změny v nastavení nebo ve volbě datových typů. Pole výstupních hodnot a graf absolutních chyb jsou předávány do následujícího SubVI pro další zpracování.

- **Nove_nastaveni_datovych_typu.vi**

Obrázek 23: Ikona pro SubVI Nove_nastaveni_datovych_typu.vi

Funkce tohoto SubVI slouží ke změně výstupních datových typů operací. Uživateli je tak umožněna pohodlnější a rychlejší změna výstupních datových typů za účelem testování. Toto SubVI je také nezbytné z důvodu práce s goniometrickými funkcemi, logaritmy a exponenciálou. V klasické verzi LabVIEW, při použití běžných operací, například sčítání, lze nastavit výstupní datový typ této operace. U již zmíněných problematický operací, jako je například operace sinus, je však absence možnosti nastavit výstupní datové typy. Toto nastavení je ale nyní pokryto pomocí tohoto SubVI. Vstupními parametry jsou součet počtu controlů a numerických konstant, 2D pole datových typů operací, tudíž vstupní a výstupní datové typy, pole clusterů, které již bylo zmíněno v popisu *Vypocetni_cast.vi*, kde každý cluster obsahuje informace o novém nastavení výstupního datového typu a zda k této změně má dojít. Posledním vstupním parametrem je pole s informací o vyčtených výstupních datových typech operací ze vzorového VI, které obsahuje popis celkového počtu bitů a počty bitů určených pro celou část.

Výstupními parametry *Nove_nastaveni_datovych_typu.vi* jsou dvě pole. První pole obsahuje seznam upravených datových typů, přičemž měněn může být pouze výstupní datový typ. Druhé pole obsahuje informace o rozložení počtu dostupných bitů a reprezentaci znaménka, v případě že nedošlo k úpravě neboli při nastavení hodnoty prvního enumu u clusteru, který udává, zda má dojít ke změně, zůstane nastavena hodnota default, zůstává prvek v obou polích vždy s původní variantou, jenž byla nastavena ve vzorovém VI. Toto pole není nutné inicializovat v případě, kdy uživatel nechce měnit výstupní datový typ operací ve vzorovém VI. Celé toto SubVI funguje na základě procházení vstupních parametrů pomocí smyčky, takže se projdou postupně všechny prvky, jelikož má každý své vlastní nastavení o možné změně.

- **Rizeni_operaci.vi**

Obrázek 24: Ikona pro SubVI Rizeni_operaci.vi

Toto SubVI řídí postupné vykonávání všech numerických operací, k čemuž využívá funkcí dalších SubVI, také zde probíhá výběr maximální vzniklé chyby pomocí vyhledávání v poli absolutních chyb. Vstupními parametry jsou 3 pole a cluster obsahující pole s hodnotami numerických konstant a počet controlů ve vzorovém VI. První pole obsahuje clustery, které popisují využívané datové typy controlů a výstupní datové typy operací ve vzorovém VI, součástí clusteru je název datového typu spolu s reprezentací znaménka a počty dostupných bitů celkem a pro celou část, reprezentace znaménka a počty bitů mají však význam jen v případě kdy se jedná o datový typ fixed point, v opačném případě jsou tyto hodnoty interně předvoleny a následně přiřazeny podle názvu datového typu. Druhé pole obsahuje clustery s informacemi o prováděných operacích, jaké má každá z nich zvolený typ zaokrouhlení a overflow mód a indexy vstupů a výstupů, které pak vybírají z pole nakrokovaných hodnot datových typů vstupní hodnoty. Třetí pole obsahuje clustery řídící krokování controlů vzorového SubVI. Tyto clustery udávají rozsah používaných hodnot příslušných datových typů, spolu s velikostí kroku, což určí kolikátý prvek se vždy bude vybírat pro operace ze zvoleného rozsahu. Rozsah vstupních hodnot musí být nastaven pro všechny použité controly, v případě že nejsou nastaveny, dojde k vypsání chyby a chod kódu se ukončí. Dochází zde k práci s výstupem *Krokovani.vi*, jenž je popsáno níže, což je pole nakrokovaných hodnot. Výstupem je vždy pole pro jeden control, proto je toto SubVI obehnáno smyčkou, která toto SubVI vykoná několikrát za účelem získání hodnot všech controlů. Tato smyčka ale nevytváří 2D pole hodnot, kde by každý řádek představoval jeden control. Smyčka vždy převádí toto pole do cluster do clusteru, aby byl zachován správný počet sloupců každého controlu. Ze smyčky tedy vystupuje pole clusterů, jenž je dále využíváno. Výstupní parametry jsou totožné jako u *Vypocetni_cast.vi*.

- **Test_rizeni_operaci.vi**

Obrázek 25: Ikona pro SubVI Test_rizeni_operaci.vi

Toto SubVI kontroluje validitu uživatelem zadávaných vstupních dat, konkrétně zda je správně zadáno pole s krokováním. Kontrolují se zde velikosti kroků, minim a maxim. Je-li zvolena nekladná velikost kroku, neceločíselný krok pro celočíselný datový typ, malý počet nastavených rozsahů pro controly, záporný nebo nulový počet dostupných bitů, nebo je zvoleno minimum větší než maximum, dojde k vypsání chyby a kód se přeruší. V případě, kdy uživatel zadá minimum nebo maximum, které je mimo rozsah příslušného datového typu, je tento rozsah upraven na přípustné hodnoty daného datového typu a uživatel je o této akci informován v podobě modálního okna s upozorněním.

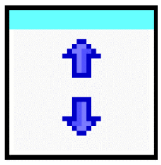
Vstupními parametry jsou dvě pole, jedno s clustery popisující datové typy a druhé popisující nastavení krokování, tyto vstupy jsou zároveň i vstupy *Rizeni_operaci.vi*, byly tedy již popsány. Výstupem je pouze errorový drát, který přenáší informaci o případné chybě.

- **Krokovani.vi**

Obrázek 26: Ikona pro SubVI Krokovani.vi

Toto SubVI slouží k nakrokování datových typů controlů na základě zvoleného nastavení krokování. Vstupní parametry jsou totožné jako u *Test_rizeni_operaci.vi*. Princip fungování je jednoduchý, u každého datového typu se pomocí velikosti kroku nastaveného v clusteru, který řídí krokování, přičítá velikost kroku k nastavenému minimu až do chvíle, kdy tato hodnota nepřesáhne maximum. Všechny tyto nakrokované hodnoty jsou uloženy do pole, které představuje výstupní parametr. V případě nejasností, jaké je přesné pořadí controlů, které jsou krokovány, lze tuto informaci vyčíst ve výstupních parametrech vyobrazených na uživatelském rozhraní.

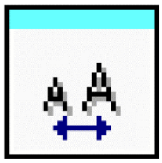
- **Fixed_point_MinMax.vi**



Obrázek 27: Ikona pro SubVI Fixed_point_MinMax.vi

V případě že je zvolen ve vzorovém VI datový typ controlu fixed point, musí se ze zadaných počtů dostupných bitů a pomocí reprezentace znaménka určit zapsatelné minimum a maximum, k čemuž je určeno toto SubVI, při volbě jiného datového typu pro control, toto SubVI není využíváno. Vstupem je tedy způsob reprezentace znaménka a počty dostupných bitů pro celou a desetinnou část. Informace o reprezentaci znaménka a počtech bitů je získána ze vzorového VI a předávána pomocí různých SubVI. Výstupem jsou minimální a maximální zapsatelné hodnoty pro nastavený fixed point.

- **Test_max_a_min.vi**

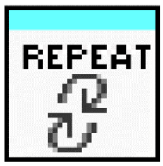


Obrázek 28: Ikona pro SubVI Test_max_a_min.vi

Toto SubVI slouží ke kontrole, zda nejsou nastavené minima a maxima pro datové typy controlů mimo jejich rozsah. V případě, že rozsah volený uživatelem v rámci krokování je mimo rozsah datového typu, vyskočí upozornění a dojde k nastavení hodnot minim a maxim na hodnoty zapsatelné pomocí zvoleného datového typu.

Vstupními parametry jsou nastavené minimum a maximum zvolené uživatelem a minimum a maximum odpovídající rozsahu daného datového typu controlu, jenž jsou určovány v nadřazeném VI. Výstupem je hodnota minima a maxima neboli krajní hodnoty intervalu pro který budou následně prováděny operace.

- **Vypocetni_SubVI.vi**



Obrázek 29: Ikona pro SubVI Vypocetni_SubVI.vi

Toto SuBVI tvoří samotné jádro výpočetní části. Vždy vykoná jednou operaci s poli vstupních hodnot.

Vstupními parametry pro *Vypocetni_SubVI.vi* jsou dvě pole s numerickými hodnotami, se kterými je operace prováděna, nastavení výstupního datového typu, opět včetně reprezentace znaménka a počtů dostupných bitů pro celou a desetinnou část (bity pro desetinnou část lze snadno získat odečtením počtů bitů pro celou část od celkového počtu bitů) a cluster, jenž obsahuje typ prováděné operace, typ zaokrouhlení a overflow mód. V případě že vykonávaná operace vyžaduje pouze jeden vstup, využívá se jen jedno vstupní pole a druhé pole operaci nijak neovlivní, toto ovládání je zajištěno externě mimo toto SubVI a řídí ho již dříve zmíněné indexy vstupních hodnot. Kromě provedení zadané operace se zde také vypočítává velikost absolutní a relativní chyby. Dále se zde na základě datových typů controlů určuje jejich minimální a maximální zapsatelná hodnota, které jsou využívány v *Test_max_a_min.vi*.

Seznam dostupných operací:

- Základní operace (Sčítání, Odečítání, Násobení, Dělení)
- Kvadratická funkce (Převrácená hodnota, Druhá mocnina, Odmocnina)
- Exponenciální funkce (Exponenciála)
- Goniometrické operace (Sinus, Cosinus, Tangens, Cotangens)
- Přirozený logaritmus
- To Double Precision Float

To Double Precision Float operace je zde uvedena, protože je-li při vytváření vzorového VI využito goniometrických, exponenciálních a logaritmických operací, potom v klasické verzi LabVIEW lze na jeho vstup připojit všechny ostatní datové typy, kromě fixed pointu, v tomto případě je nutné využít funkci To Double Precision Float, aby bylo možné připojit tento datový typ k této operaci. Hodnota nebude nijak pozměněna, pouze dojde k zajištění žádoucího bezchybového stavu vzorového VI.

V seznamu operací na uživatelském rozhraní se objeví všechny vykonané operace včetně této, její výstupní numerické hodnoty budou totožné jako hodnoty vstupní, tudíž by je uživatel měl přeskočit a brát tuto situaci jen jako informaci že proběhla zvolená operace.

Výstupními parametry je cluster obsahující pole hodnot absolutních a relativních chyb, výsledky operace typu double a typu, který byl navolen uživatelem při vytvoření vzorového VI nebo ho na uživatelském rozhraní nastavil jinak, jedná se o pole hodnot, neboť i vstup operace jsou pole. Dalšími výstupy jsou velikost maximální chyby a její index a graf zobrazující vývoj absolutní chyby v závislosti na kombinaci vstupních hodnot. Počet výsledných hodnot operace závisí na počtu jejich vstupů.

Vyžaduje-li operace pouze jeden vstup, počet jejího provedení bude roven velikosti vstupujícího pole numerických hodnot, jedná-li se o operaci se dvěma vstupy, počet provedení bude roven součinu velikostí obou vstupních polí. V případě že ve vzorovém VI je použita místo controlu konstanta, pracuje se s touto hodnotou jako s polem o jednom prvku a počet operací je roven velikosti druhého pole. Pro situaci, kdy se jedná o operaci, která má na vstup přivedeny dvě konstanty, operace se provede pouze jednou. Tyto pravidla platí pouze v případě, že se během výpočtů nenarazí na kombinaci operace a vstupních hodnot, která způsobí vyvolání chyby.

- **Test_vstupu_operaci.vi**

Obrázek 30: Ikona pro SubVI Test_vstupu_operaci.vi

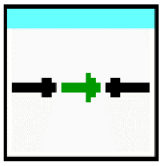
Toto SubVI slouží k zajištění situace, kdy část vstupních numerických hodnot, se kterými bude prováděna operace, bude splňovat podmínky a jiná část jejich hodnot bude vést k chybovým stavům. Například při práci v Labview na FPGA targetu při použití operace odmocnina, nelze připojit na vstup této operace žádný datový typ, který je signed.

Chybové stavy jsou:

- odmocnina ze záporných čísel
- dělení nulou
- převrácená hodnota nuly
- tangens π
- cotangens 2*π
- Nekladné logaritmované číslo

Vstupními parametry jsou vždy dvě hodnoty spolu s prováděnou operací. Výstupem je pouze errorový drát, který v případě vzniku chyby přeruší chod programu a informuje uživatele o důvodu jejího vzniku. Na základě informace o chybě, je uživatel schopný tuto chybu opravit.

- **Spojeni_funkci.vi**



Obrázek 31: Ikona pro SubVI Spojeni_funkci.vi

Toto SubVI slouží jako prostor pro sloučení tří SubVI (*Zaokrouhlovani.vi, Binarni_prevod.vi a Rezani_bitu.vi*), aby byla zvýšena přehlednost v hlavní části blokového diagramu. Jeho funkce je čistě estetická.

- **Binarni_prevod.vi**

Obrázek 32: Ikona pro SubVI Binarni_prevod.vi

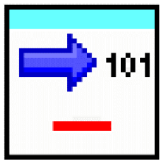
Toto SubVI opět využívá jiných SubVI pro svou funkci. Konkrétně se používají *Binarni_prevod_kladnych_hodnot*.vi a *Binarni_prevod_kladnych_hodnot.vi*. Vstupním parametrem je numerická hodnota a způsob reprezentace znaménka. Výstupními parametry jsou celočíselná a desetinná část vstupní numerické hodnoty, pole binárních hodnot pro vstupní numerickou hodnotu, dvě pole binárních hodnot pro jeho celočíselnou a desetinnou část a potřebné bity pro zápis obou částí.

- **Binarni_prevod_kladnych_hodnot.vi**

Obrázek 33: Ikona pro SubVI Binarni_prevod_kladnych_hodnot.vi

Toto SubVI realizuje první část funkce nadřazeného *Binarni_prevod.vi* a používá se pro převod nezáporných hodnot z desítkové soustavy do binární. Vstupní i výstupní parametry jsou totožné s nadřazeným SubVI.

- **Binarni_prevod_zapornych_hodnot.vi**



Obrázek 34: Ikona pro SubVI Binarni_prevod_zapornych_hodnot.vi

Toto SubVI realizuje druhou část funkce nadřazeného *Binarni_prevod.vi* a používá se pro převod záporných hodnot z desítkové soustavy do binární. Vstupní i výstupní parametry jsou opět totožné s nadřazeným SubVI.

- **Rezani_bitu.vi**

Obrázek 35: Ikona pro SubVI Rezani_bitu.vi

V tomto SubVI se upravují obě části (celá i desetinná) zadané číselné hodnoty na hodnotu zapsatelnou pro nastavený rozsah. Pro úpravu při zvoleném overflow módu na parametr wrap se využívá *Cteni_bitu.vi*. Vstupními parametry jsou celá část hodnoty zadaného čísla a pole binárních hodnot celé části zadaného čísla, pole binárních hodnot desetinné části, počet potřebných a dostupných bitů pro celou část a overflow mód. Výstupními parametry jsou upravená hodnota, zapsatelná pro daný rozsah signalizace saturace, práce se zápornými hodnotami a overflowu v podobě booleanu, s výsledky signalizace pracuje dále *Zaokrouhlovani.vi*, které na základě nich mění svůj průběh.

- **Cteni_bitu.vi**

Obrázek 36: Ikona pro SubVI Cteni_bitu.vi

Toto SubVI slouží k převedení již upraveného pole binárních hodnot reprezentujících celou část čísla, které je dáno výstupem *Rezani_bitu.vi*, na celé číslo v desítkové soustavě. Vstupními parametry jsou pole binárních hodnot a způsob reprezentace znaménka. Výstupním parametrem je pak převedená hodnota zapsaná v desítkové soustavě.

- **Zaokrouhlovani.vi**

Obrázek 37: Ikona pro SubVI Zaokrouhlovani.vi

Funkce tohoto SubVI je důležitá pro úpravu numerických hodnot, neboť se zde upravuje desetinná část zadané číselné hodnoty na hodnotu zapsatelnou pro nastavený rozsah. Vstupními parametry jsou desetinná část zadaného čísla, počet dostupných bitů pro desetinnou část, typ zaokrouhlení a signalizace saturace, práce se zápornými hodnotami a overflowu v podobě booleanu, které jsou poskytovány ze *Rezani_bitu.vi*, jehož výstup slouží jako vstupní hodnoty pro toto SubVI. Výstupním parametrem je zaokrouhlená hodnota.

- **Prepinani_vysledku_operaci.vi**

Obrázek 38: Ikona pro SubVI Prepinani_vysledku_operaci.vi

Toto SubVI slouží k detailnějšímu prozkoumání výsledných hodnot z *Vypocetni_cast.vi*. Vstupními parametry jsou pole výstupních hodnot a graf absolutních chyb z již zmíněného předcházejícího SubVI a numerický control zadávaný uživatelem na uživatelském rozhraní, kterým určí pořadí operace, kterou chce detailněji prozkoumat. Indexace operací probíhá od nuly. Po vyplnění hodnoty pořadí, je vyobrazen graf absolutních chyb a pole výstupních hodnot pouze pro zvolenou operaci, těmito prvky jsou reprezentovány výstupní parametry tohoto SubVI. Díky izolování této části od části výpočetní, lze jednou spustit program, za účelem načtení dat a následně otevřít toto SubVI a pomocí přepínání hodnoty pořadí operace pro zobrazení si libovolně projít všechny výpočetní výsledky bez nutnosti znovu spouštět celý program a načítat tak data znovu, v případě že nedojde ke změně vstupních dat.

4. UKÁZKOVÉ PŘÍKLADY (EXAMPLES)

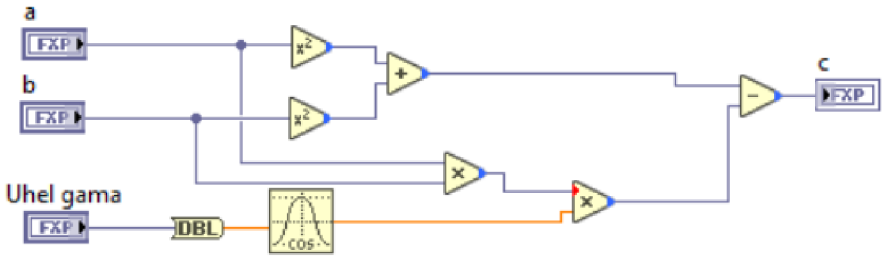
Tato kapitola slouží pro demonstraci a ověření funkčnosti vytvořeného nástroje. Po vytvoření vzorového VI, vyplnění vstupních parametrů a spuštění programu se uživateli zobrazí výstupní parametry, kde se zaměří na informaci o maximální vzniklé absolutní chybě a porovná ji s pro něj přijatelnou hodnotou, v případě, že je velikost chyby menší nebo rovna přijatelné hodnotě, je nastavení provedeno správně. V momentě, kdy absolutní chyba některé operace přesáhne stanovenou hranici, musí být provedena změna nastavení parametrů ve vzorovém VI nebo na uživatelském rozhraní. Proces se následně opakuje, dokud nejsou splněny uživatelovy podmínky.

V případě, že uživatel neví, jak se nastavuje výstupní datový typ operace, stačí kliknout pravým tlačítkem myši na příslušnou operaci a zvolit *Properties*. Po kliknutí se se objeví okno *Object Properties,* kde uživatel zvolí *Output Configuration* a následně se mu zobrazí nastavení výstupního datového typu operace. Toto nastavení je dále vyobrazeno v kapitole 4.2.2.

4.1 Example 1 – Cosinova věta

První ukázkový příklad demonstruje simulaci výpočtu pomocí cosinovy věty. Krajní hodnota absolutní chyby požadována uživatelem má velikost 0.07.

4.1.1 Vytvoření vzorového VI

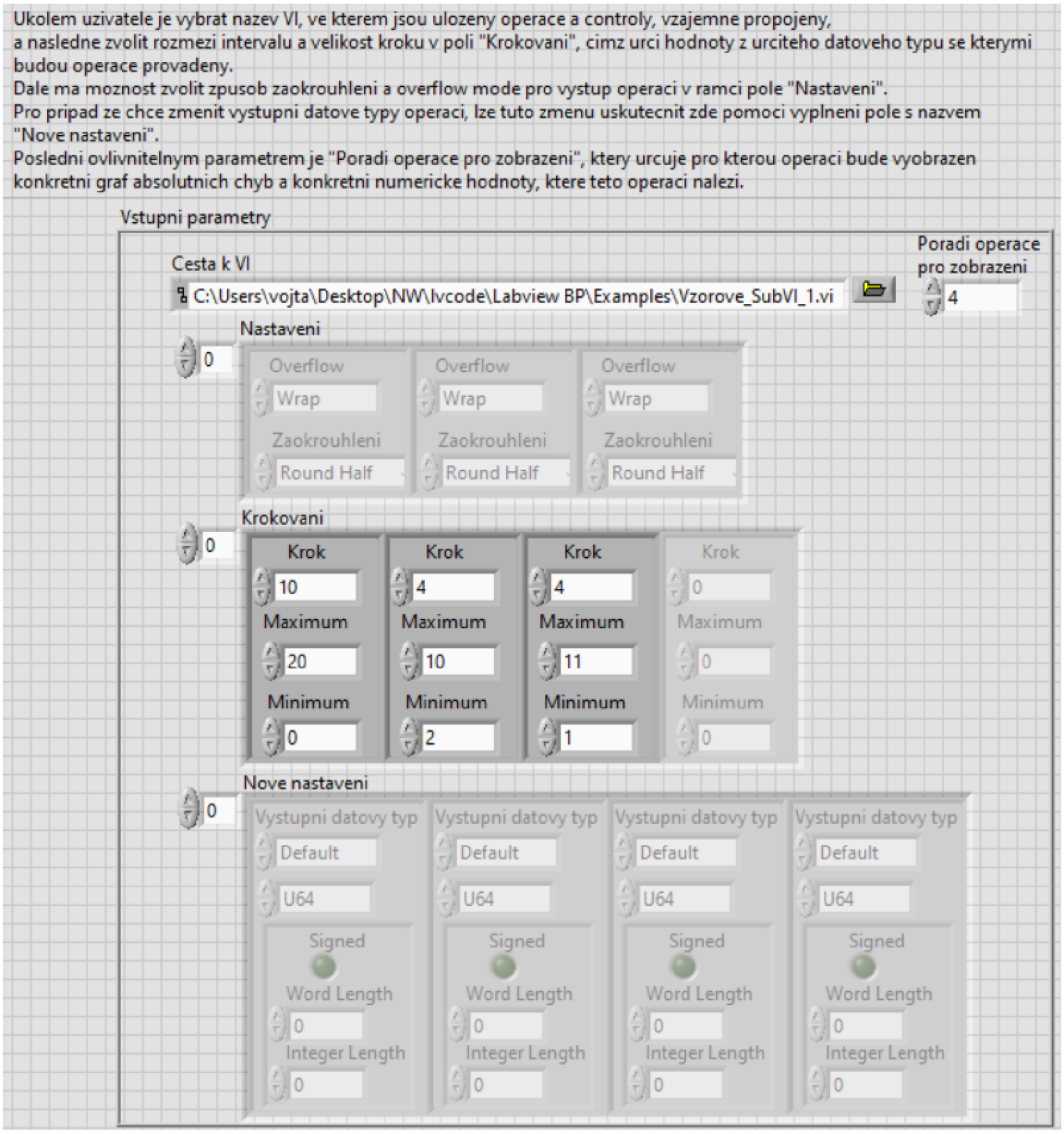


Obrázek 39: Blokový diagram – Vzorové VI – Cosinova věta

Prvním krokem je vytvoření samostatného vzorového VI, které bude obsahovat pouze povolené operace a datové typy, jejichž kompletní seznam je uveden v kapitole 2.3. V uvedeném seznamu jsou všechny operace, které jsou potřeba k simulaci výpočtu cosinovy věty, tímto je splněna uvedená podmínka. Uživatel tedy pomocí vytvoření potřebného počtu numerických controlů a následného nastavení jejich datového typu určuje první parametry, který využívá vytvořený nástroj.

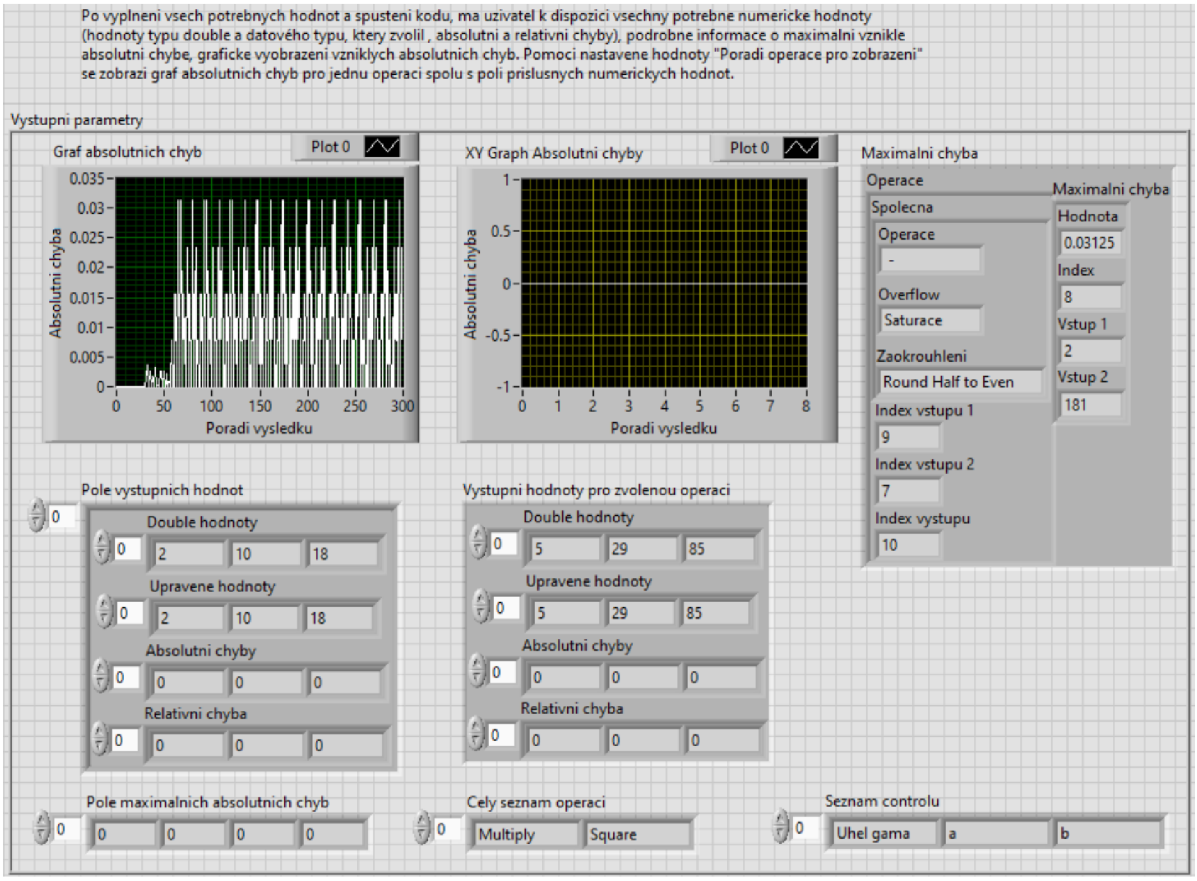
Po vytvoření controlů je potřeba umístit na blokový diagram všechny potřebné matematické operace a ve správném pořadí je propojit, jak je vyobrazeno na Obrázek 39. Jakmile je vše správně nastaveno a propojeno, vzorové VI se uloží a uživatel již pracuje s uživatelským rozhraním. VI vytvořeno pro tento příklad je uloženo pod názvem *Vzorove_SubVI_1*.

4.1.2 Nastavení uživatelského rozhraní



Obrázek 40: Uživatelské rozhraní – Vstup – Cosinova věta

V případě uživatelského rozhraní, je prvním krokem zvolit cestu k vytvořenému vzorovému VI. Následně je důležité správné vyplnění krokování numerických controlů, čímž se vytvoří pole hodnot, se kterými budou operace pracovat. Pokud uživatel zvolil pro výpočty pouze konstantní hodnoty v rámci vzorového VI, nastavení krokování neprovádí a program může být bez jakýchkoliv změn spuštěn.
Nastavení pole *Krokování* uživatel provádí pro každý zvolený control, v případě, kdy si není jistý, který control odpovídá, kterému prvku pole, může zadat pár náhodných hodnot a program spustit, následně se mu mezi výstupními parametry vypíše seznam controlů, viz Obrázek 41, ve stejném pořadí, v jakém jsou krokovány.



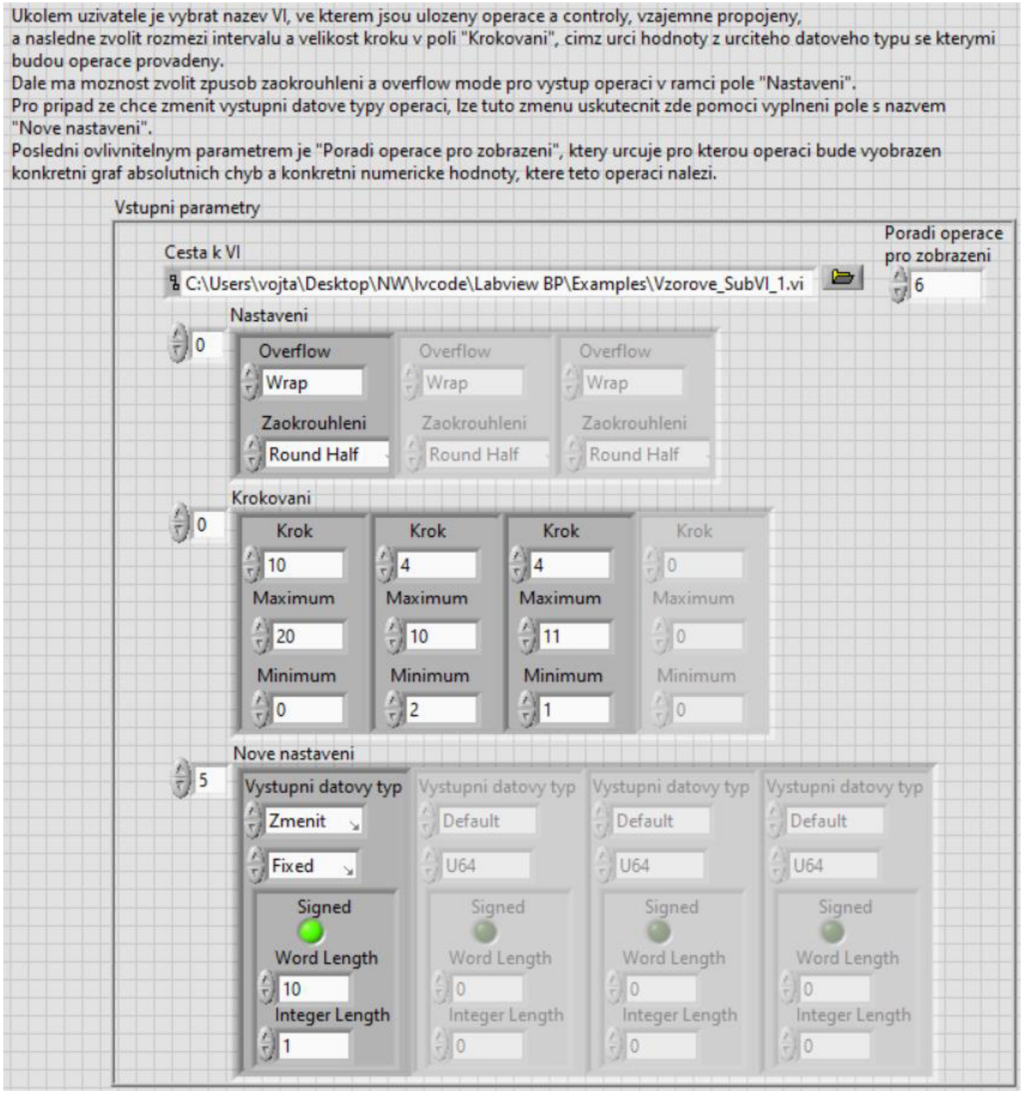
Obrázek 41: Uživatelské rozhraní – Výstup – Cosinova věta

Na Obrázek 41 lze v pravém dolním rohu vidět pole řetězců, které představují již zmíněné názvy control v pořadí v jakém probíhá jejich krokování. Nyní je známo jejich pořadí a uživatel může provést změnu rozsahu či velikosti kroku. Aktuální nastavení je pro příklad vyhovující. Dle zvoleného nastavení se tedy vytvoří pro *Uhel gama* pole s hodnotami 0,10 a 20. Pro control *a* vznikne pole o hodnotách 2,6 a 10. Pro control *b* vznikne pole s hodnotami 1, 5 a 9.

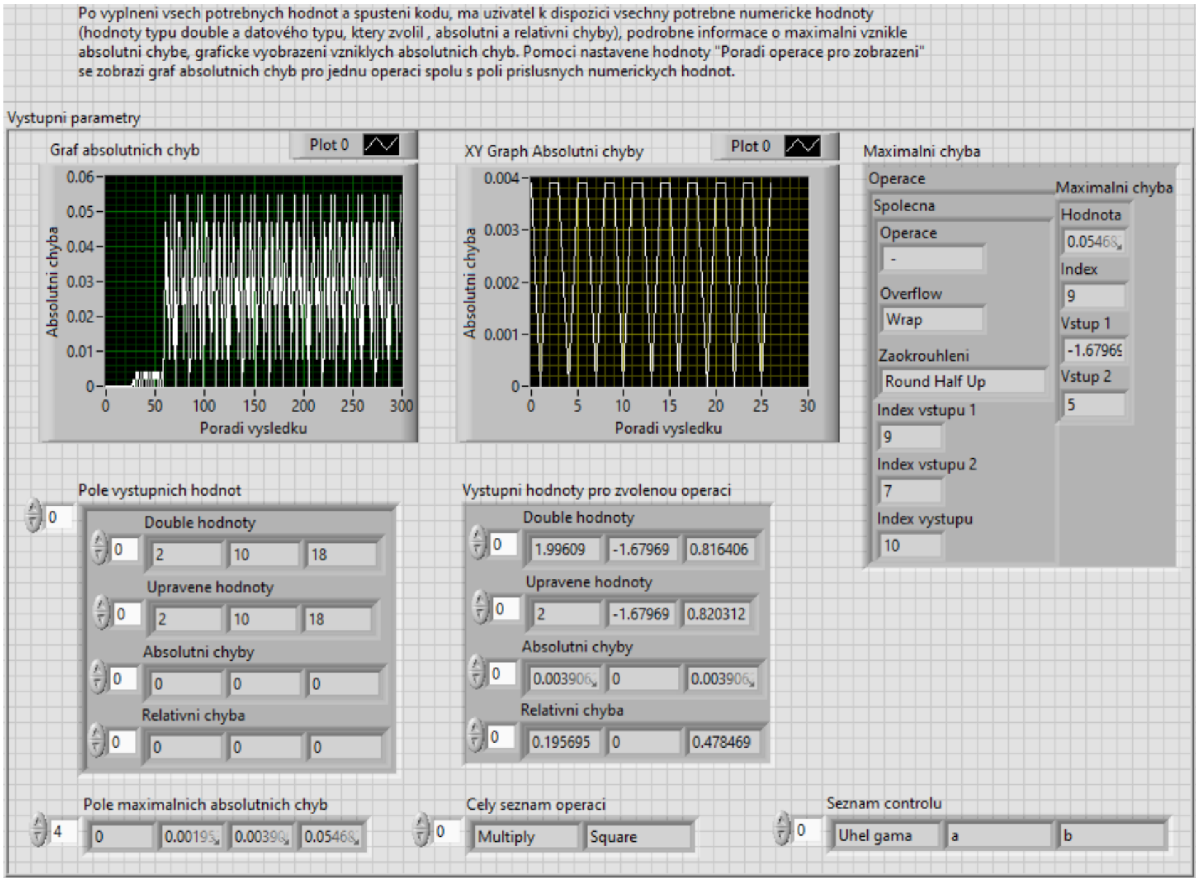
Dalším krokem je volba již nepovinných parametrů, kterými je pole *Nastavení* a pole *Nove nastavení.* Pole Nastavení představuje nastavení typu zaokrouhlování a overflow módu pro jednotlivé operace, přičemž princip je stejný jako u controlu. Pokud chce uživatel nastavit jeden z těchto parametrů pro konkrétní operaci, stačí se podívat do pole *Cely seznam operaci,* kde jsou uvedeny všechny operace v pořadí, v jakém jsou vykonávány. Pořadí vykonání operace je shodné s pořadím v poli *Nastavení* i v poli *Nové nastavení.* Uživateli tedy stačí najít příslušnou operaci v poli a podle jejího indexu provést příslušné nastavení. Druhé pole představuje změnu výstupního datového typu operace, což se uplatňuje hlavně u operací, které byly několikrát zmíněny výše v dokumentu, například v kapitole 2.4.1, neboť ve vzorovém VI pro tyto operace tuto volbu uživatel neprovede.

Operace jsou poté prováděny se všemi aplikovanými změnami. Pokud není provedena změna ani v jednom z polí, zůstávají pro všechny operace výchozí hodnoty. Výchozími hodnotami jsou datový typ nastavený ve vzorovém VI, saturace coby overlow mód a round half even jakožto způsob zaokrouhlení.

Pro tento příklad je potřeba změnit výstupní datový typ operace cosinus, neboť v opačném případě by operace měla výstupní datový typ double. Pořadí této operace vyčteme z pole *Cely seznam operaci*. Operace cosinus je v poli umístěna pod indexem pět, jak je i dále ukázáno na Obrázek 43. Ze stejného obrázku si lze všimnout, že operace násobení následuje hned za operací cosinus, vezmeme tedy její index a vložíme ho do controlu *Poradi operace pro zobrazeni,* abychom o této operaci získali detailnější informace. Jako poslední zvolíme pro operaci násobení s indexem nula, což lze vyčíst z Obrázek 41, nastavení overflow módu na wrap a způsob zaokrouhlení na round half up. Jakmile jsou nastaveny všechny požadované údaje, uživatel spustí program.



Obrázek 42: Uživatelské rozhraní – Vstup 2 – Cosinova věta

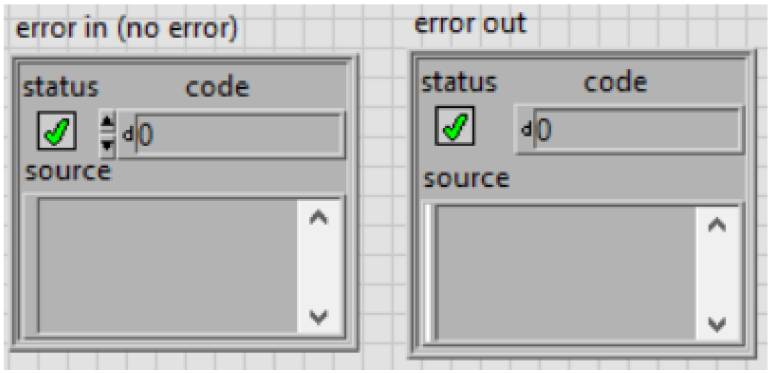


Obrázek 43: Uživatelské rozhraní – Výstup 2 – Cosinova věta

Jakmile se celý program úspěšně vykoná, vypíšou se na uživatelském rozhraní upravené výstupní parametry. Lze si všimnout mírné změny v *Grafu absolutnich chyb*, která bude způsobena změnou výstupního datového typu operace cosinus. Změny overflow módu a typu zaokrouhlení pro operaci prvního násobení se na velikosti absolutní chyby neprojeví. Pro každou operaci vzniká maximální absolutní chyba o určité hodnotě, tyto hodnoty jsou uvedeny v poli Pole maximálních absolutnich chyb. Maximální hodnota absolutní chyby pro první operaci, kterou je násobení, je nulová, tudíž změna nastavení velikost absolutní chyby neovlivnila, ikdyž tato změna proběhla. Na základě hodnot v *Poli maximálních absolutnich chyb* na Obrázek 43, absolutní chyba vzniká až pro operaci s indexem pět, kterou je operace cosinus. Kromě již zmíněných polí jsou zde dále všechny výstupní hodnoty pro každou operaci, které jsou uvedeny v *Poli vystupnich hodnot*. Z tohoto pole se pomocí controlu *Poradi pro zobrazeni operace vybere* jedna operace a pro ni jsou vypsány výstupní hodnoty odděleně od pole s výsledky všech operací. Vybraná operace pro zobrazení je druhé násobení s indexem šest, jejíž hodnoty jsou v poli *Vystupni hodnoty pro zvolenou operaci*. Pro zvolenou operaci je také vyobrazen graf absolutních chyb *XY Graph Absolutni chyby*, oddělený od grafu, který vyobrazuje všechny hodnoty absolutních chyb v průběhu vykonávání operací.

Posledním výstupním parametrem je cluster *Maximalni chyba*, který uživatele informuje, pro jakou operaci, s jakými vstupy a s jakým nastavením vznikla největší absolutní chyba. Pro tento příklad největší chyba nastává pro operaci odečítání. I tato velikost je však stále v normě a na základě této hodnoty a grafického vývoje absolutní chyby lze prohlásit, že nastavení bylo provedeno správně a uživatelův požadavek na velikost absolutních chyb byl splněn. V případě, kdy je maximální absolutní chyba menší než požadovaná mez, je možné zmenšit počet dostupných bitů výstupního datového typu operací, pokud tato změna povede na překročení meze, lze vrátit původní nastavení a změnu provést jinde.

Po celou dobu byl hlídán bezchybový stav, který je vyobrazen na Obrázek 44.

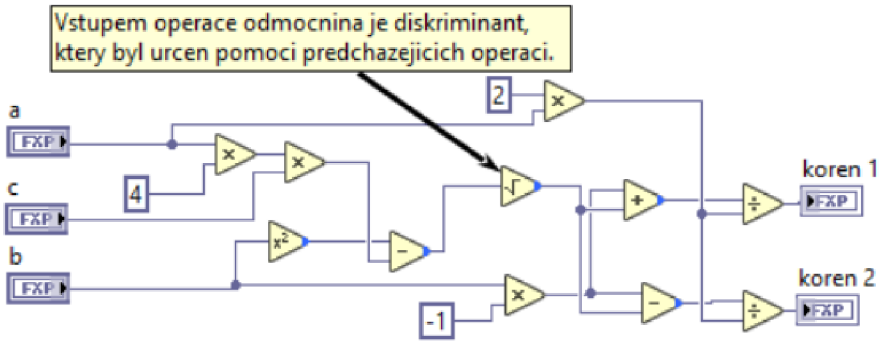


Obrázek 44: Uživatelské rozhraní – hlášení o chybě – Cosinova věta

4.2 Example 2 – Výpočet kořenů kvadratické rovnice

Druhý ukázkový příklad demonstruje simulaci výpočtu pomocí výpočtů kořenů kvadratické rovnice. Popis některých částí uživatelského rozhraní není tak detailní jako v příkladě 4.1. Kapitola se detailněji zabývá metodou snížení absolutní chyby. Postup limitace absolutní chyby lze provést rychleji, provedením několika úprav najednou, z důvodu lepšího pochopení čtenářem budou ale kroky prováděny postupně. Krajní hodnota absolutní chyby požadována uživatelem má velikost 0.15.

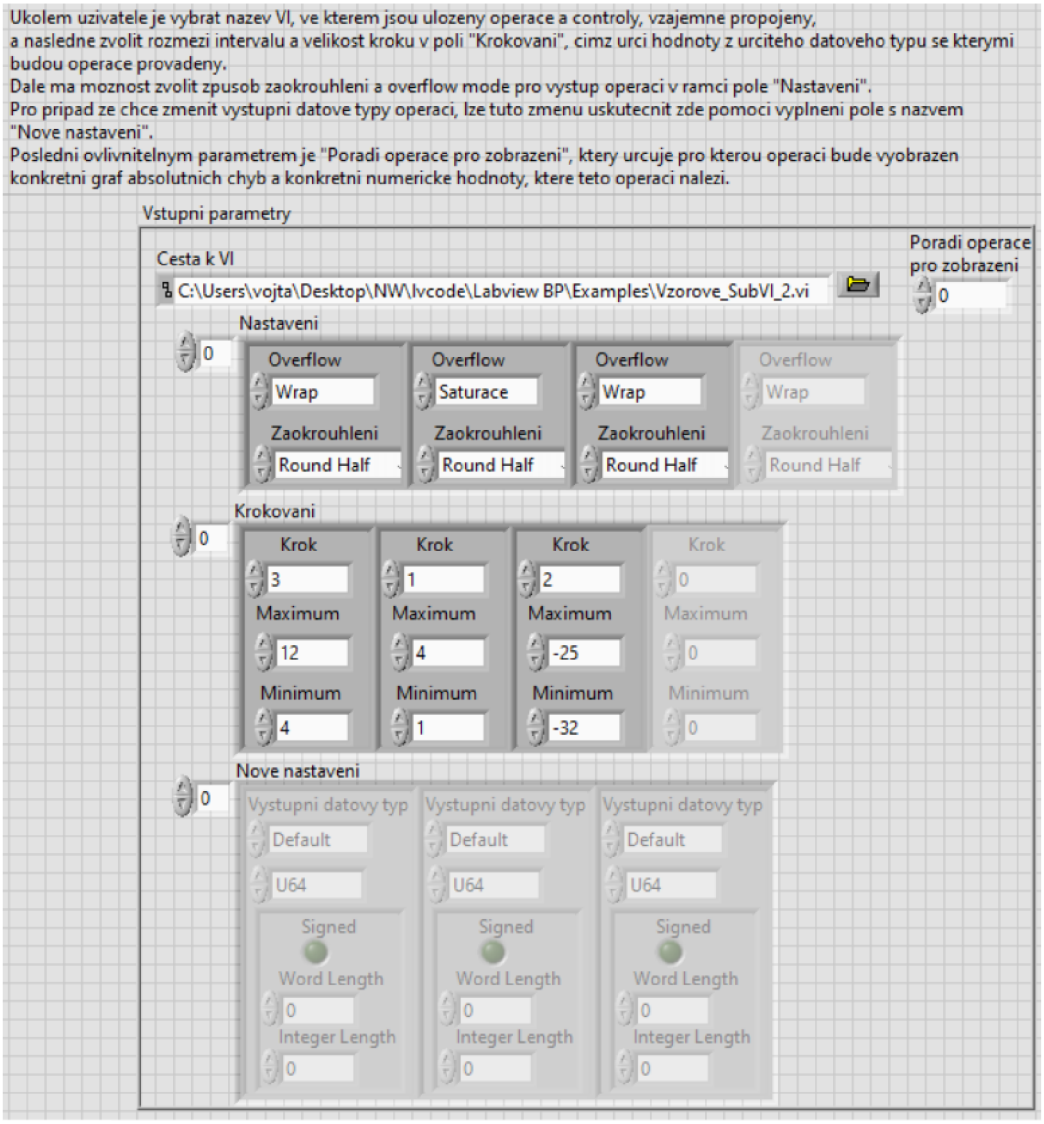
4.2.1 Vytvoření vzorového VI



Obrázek 45: Blokový diagram – Vzorové VI – Kořeny kvadratické rovnice

Uživatelovým prvním krokem je opět vytvoření samostatného vzorového VI uvedeného na Obrázek 45, které bude obsahovat pouze povolené operace a datové typy. Použité operace i datové typy splňují seznam uvedený v kapitole 2.3. Na rozdíl od předchozího vzorového VI ukázkového příkladu v kapitole 4.1.1, se zde objevují kromě controlů i konstanty. Konstantní hodnoty jsou však povoleny také, pokud splňují požadovaný datový typ, což v aktuálním případě splňují, neboť jsou všechny datového typu fixed point. Po vytvoření controlů, konstant a operací, jejich vzájemném propojení a nastavení všech datových typů, následuje přesun pozornosti na uživatelské rozhraní. VI vytvořeno pro tento příklad je uloženo pod názvem *Vzorove_SubVI_2*.

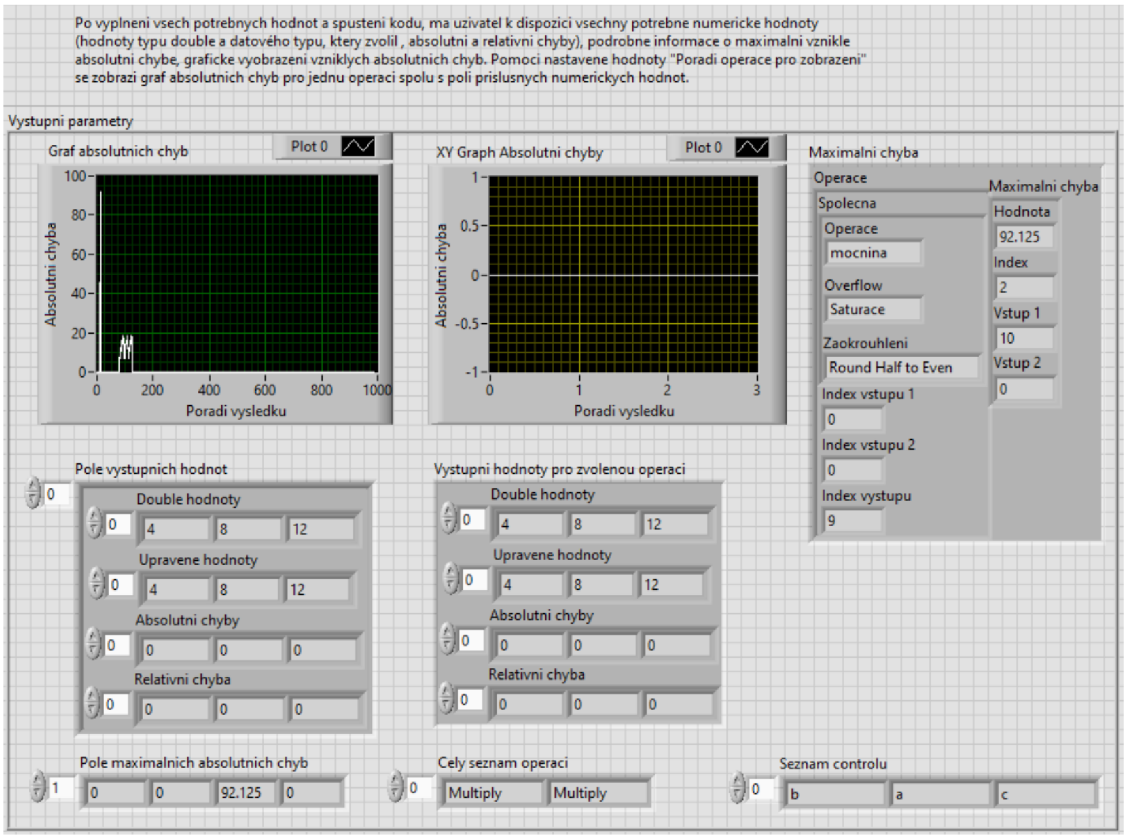
4.2.2 Nastavení uživatelského rozhraní



Obrázek 46: Uživatelské rozhraní – Vstup – Kořeny kvadratické rovnice

Pro aktuální příklad je nutné být na pozoru při volbě vstupních parametrů. V případě nevhodné volby parametrů v *Krokování*, by mohla by nastat situace, kdy by se při určité kombinaci vstupních parametrů následně objevil diskriminant se zápornou hodnotou. Záporná hodnota je však automaticky převedena na unsigned, z důvodu že LabVIEW nepovoluje signed hodnoty datového typu fixed point jako vstup operace odmocnina. Při pokusu o připojení signed hodnoty datového typu fixed point, by bylo výsledkem vypsání chyby. Převodem na unsigned by tak vznikala absolutní chyba, kterou by nemuselo být možné eliminovat. Aktuálně jsou zvoleny hodnoty pro všechny controly v omezeném rozsahu, které jsou však vhodnými vstupními parametry a nevzniká tak záporný diskriminant. Z důvodu omezeného množství vstupních parametrů by bylo potřeba provést ke komplexnější analýze více simulací s různými rozsahy.

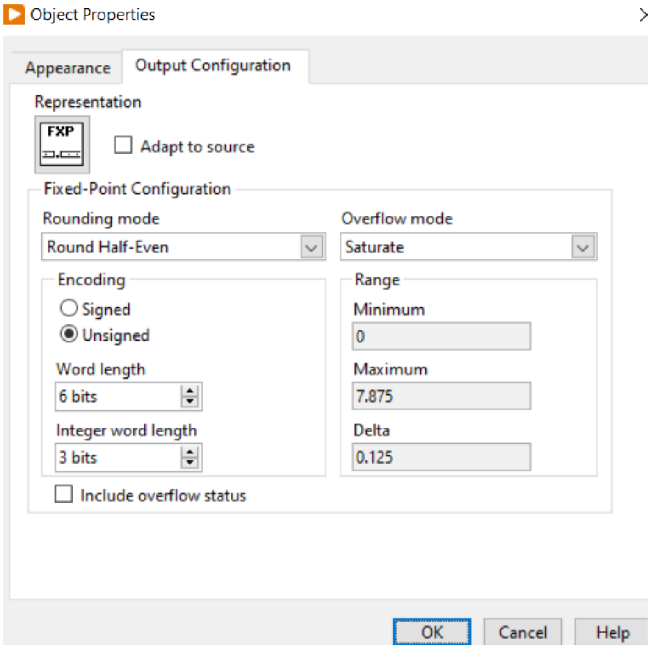
Na Obrázek 46 je vidět uživatelské rozhraní s již nastavenými vstupními parametry. Parametry byly nastaveny stejným principem jako v kapitole 4.1.2. Nyní zbývá program spustit a následně prozkoumat výstupní parametry.



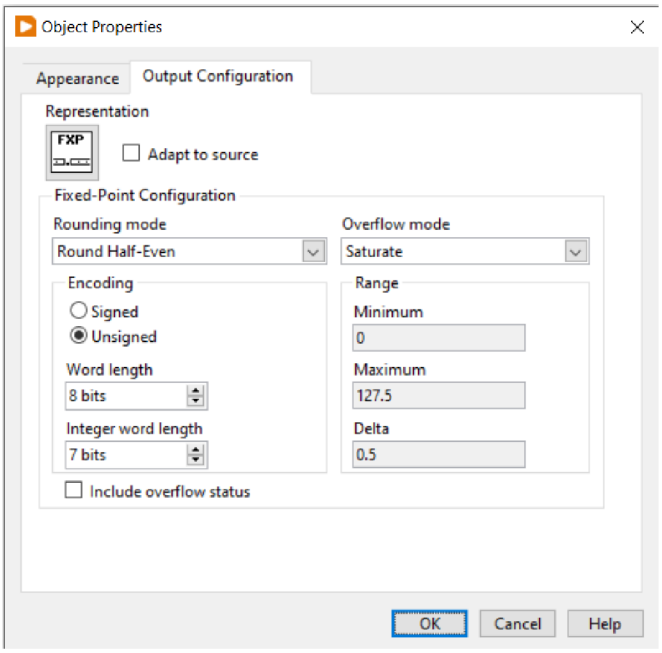
Obrázek 47: Uživatelské rozhraní – Výstup – Kořeny kvadratické rovnice

Po spuštění programu se opět na uživatelském rozhraní objeví výstupní parametry pro zvolené nastavení viz Obrázek 47. V *Grafu absolutnich chyb* jsou vidět dva hlavní nárusty absolutní chyby. Z clusteru *Maximalni chyba* plyne, že větší velikost absolutní chyby způsobuje operace mocnina, proto je nutné změnit její výstupní datový typ a zvětšit množství dostupných bitů.

Nyní jsou dvě možnosti změny výstupního datového typu. První možností je změnit tento typ na uživatelském rozhraní v rámci vstupních parametrů. Druhou možností je návrat do vzorového VI a změnu provést v něm. Jelikož změna provedená na uživatelském rozhraní není permanentní, ale jen v rámci tohoto rozhraní, a protože se toto nastavení primárně používá u operací, u kterých nelze měnit jejich datový typ přímo ve vzorovém VI, zvolíme provést změnu přímo ve VI, které je vyobrazeno na Obrázek 45. Změna výstupního datového typu operace probíhá stejně jako když ji uživatel nastavuje prvně, jen aktuálně změníme rozložení počtu dostupných bitů pro celou a pro desetinnou část.

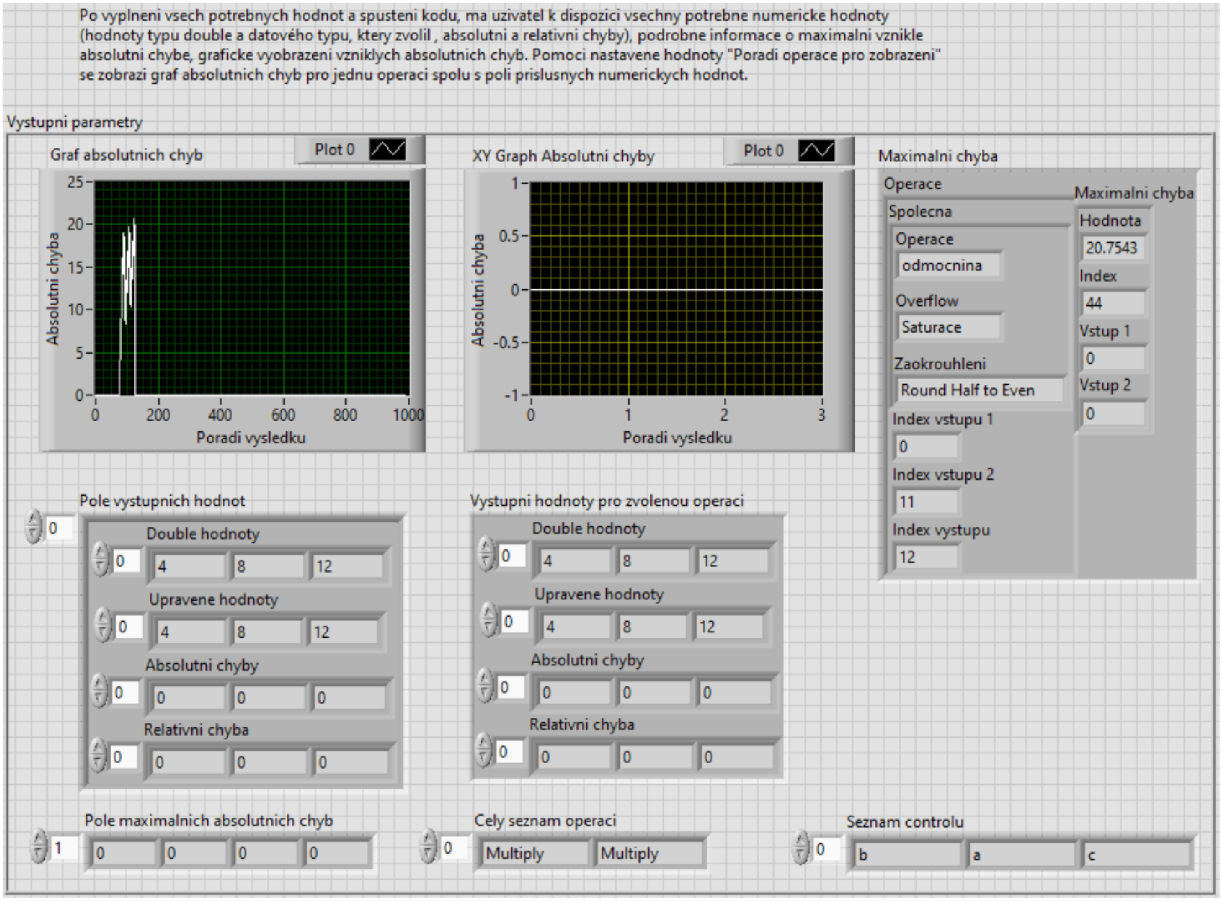


Obrázek 48: Původní nastavení bitů



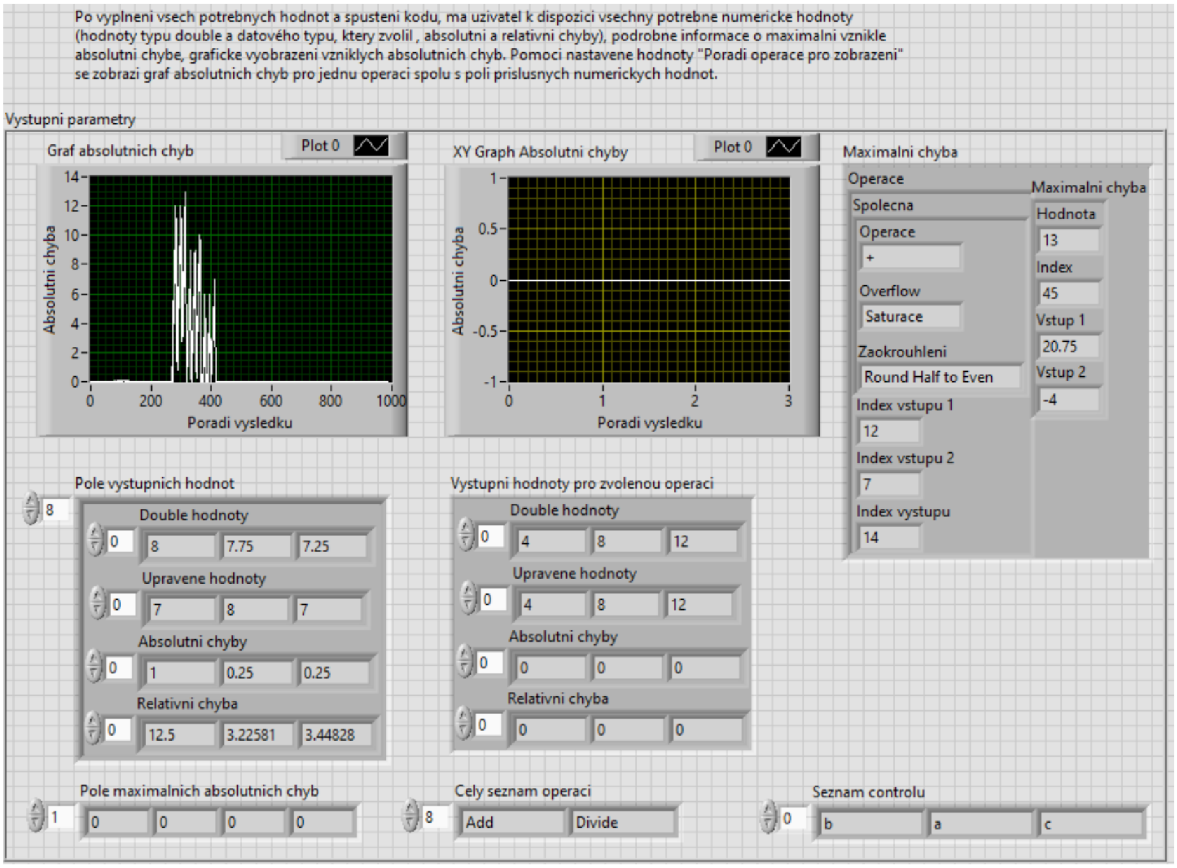
Obrázek 49: Upravené nastavení bitů

Nyní když je provedena změna, spustíme program znovu. Výstupní parametry získané po úpravě jsou zobrazeny na Obrázek 50.



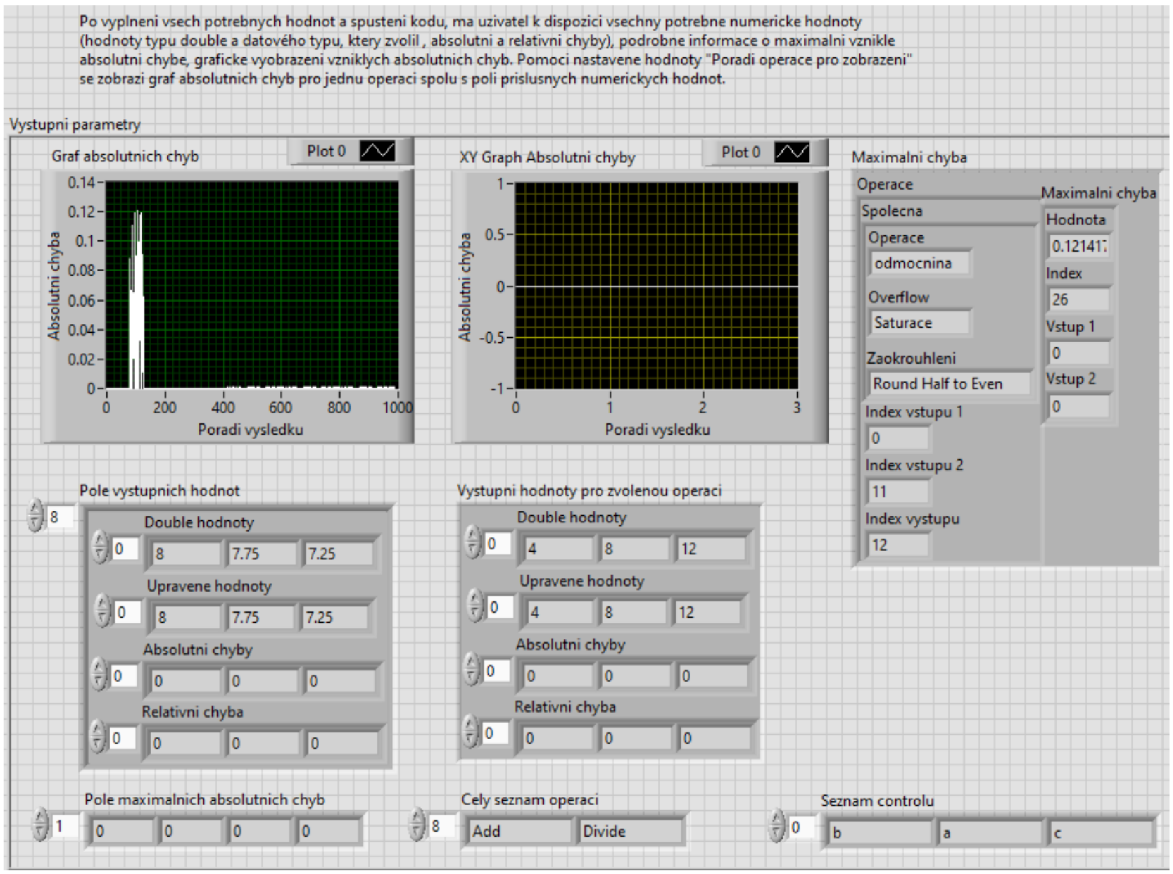
Obrázek 50: Uživatelské rozhraní – Výstup 2 – Kořeny kvadratické rovnice

Díky průběhu vyobrazenému v *Grafu absolutnich chyb* je vidět, že změna výstupního datového typu operace proběhla úspěšně. Na základě vyčtení pořadí operace mocnina v poli *Cely seznam operaci* v kombinaci s polem *Pole maximálních absolutnich chyb*, lze zjistit, že maximální velikost absolutní chyby této operace je nyní nulová, což je ideální stav a nastavení této operace již nebude nutné opravovat. Nyní největší absolutní chybu způsobuje operace odmocnina. Provedeme tedy znovu stejnou změnu výstupního datového typu ve vzorovém VI a celé části navýšíme počet dostupných bitů. Po provedení potřebné změny program znovu spustíme.



Obrázek 51: Uživatelské rozhraní – Výstup 3 – Kořeny kvadratické rovnice

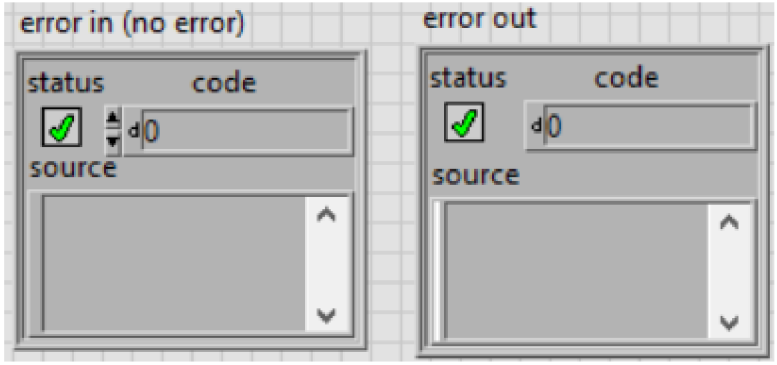
Opět se změna výstupního datového typu projevila a na Obrázek 51 je vidět, že se ale nově objevil problém s operací sčítání, který v předchozím případě nebyl. Vznik této absolutní chyby je následkem změny výstupního datového typu operace odmocnina. Tyto dvě operace jsou propojeny a jedna ovlivňuje svým výstupem tu druhou, v tomto případě výstup odmocniny ovlivňuje operaci sčítání, která nyní dosahuje vysoké maximální absolutní chyby, která je větší než stanovená hranice. Opět tedy provedeme změnu výstupního datového typu ve vzorovém VI. Následně znovu spustíme program.



Obrázek 52: Uživatelské rozhraní – Výstup 4 – Kořeny kvadratické rovnice

Jak je vidět v clusteru *Maximalni chyba*, nyní vznikající maximální absolutní chyba již splňuje požadovanou velikost a operace jsou tedy nastaveny správně. Proces simulace výpočtů je tímto ukončen a výpočetní algoritmus ve vzorovém VI lze použít a bude splněna požadovaná podmínka.

Opět byl po celou dobu hlídán bezchybový stav, který je vyobrazen na Obrázek 53.



Obrázek 53: Uživatelské rozhraní – hlášení o chybě – Kořeny kvadratické rovnice

5. ZÁVĚR

Cílem bakalářské práce je vytvoření nástroje pro simulaci výpočtů s pevnou řádovou čárkou. V úvodu byl detailněji rozepsán důvod vypracování tohoto tématu.

Teoretický rozbor poskytne čtenáři informace, které jsou nutné k pochopení zvoleného postupu a přístupu k řešení této problematiky. Slouží k získání určitého povědomí o rozdílných vlastnostech datových typů floating pointu a fixed pointu. Tímto byl také splněn první bod zadání bakalářské práce.

Po získání teoretických znalostí následuje část s blokovými diagramy. Jsou zde umístěny za účelem získání představy o průběhu vývoje tohoto nástroje od samotného počátku až do vytvoření jeho aktuální finální podoby. V této části jsou vyobrazeny a popsány celkem čtyři vývojové diagramy. Jeden z diagramů slouží jako určité shrnutí kroků, které by měl uživatel podniknout, při využívání vytvořeného nástroje. Na zbylých třech diagramech je vidět, jaké změny se odehrávaly ve snaze dosáhnout co nejlepších výsledků. Dále byl proveden návrh uživatelského rozhraní, kde jsou umístěny všechny vstupní a výstupní parametry. Všechny náležitosti druhého bodu zadání jsou splněny.

Třetí bod zadání je splněn v praktické části v kapitole 3, kde je detailně popsán postup dosažení stanoveného cíle. Jsou zde principiálně popsány a vysvětleny všechny vytvořené SubVI, pomocí jejichž propojení se dosahuje požadované funkce. Využívaných SubVI je velké množství, proto je za účelem zamezení možného zmatení uživatele zobrazena struktura všech SubVI, což představuje určitou mapu toku informací a vzájemného propojení.

Tvorba nástroje se neobešla bez komplikací, například jednou z komplikací bylo zjištění, že nelze pomocí funkcí VI Serveru získat informace o nastavení způsobu zaokrouhlení a overflow módu. Následně byla tato problematika vyřešena nastavením těchto hodnot na výchozí hodnoty, kterých obecně nabývají všechny operace, pokud nejsou nastaveny jinak. V případě, kdy je nutné toto nastavení pro vytvořený nástroj změnit, má uživatel připraveno provést tuto změnu na uživatelském rozhraní. V některých případech se tedy musí zadávat vstupní parametry navíc, stále je ale tento nástroj přínosný pro uživatele časově i funkčně.

Čtvrtým bodem zadání je provést otestování funkčnosti implementace. Vytvořený nástroj byl testován na několika ukázkových příkladech, přičemž dva z nich jsou zdokumentované. Na základě správnosti výsledků testů lze prohlásit, že implementace proběhla úspěšně. Jednotlivé části nástroje byly testovány a ověřovány v průběhu implementace. Celý postup testování je zdokumentován a uveden v kapitole 4. Nástroj tak uživateli usnadní a urychlí optimální převod datových typů a provádění operací.

Aktuální forma nástroje není dokonalá, ale stále je efektivní. Možným budoucím zlepšením je například vytvoření nebo nalezení způsobu vyčítání způsobu zaokrouhlení a overflow módu, aby byla efektivita nástroje zvýšena. Vytvořením dokumentace spolu s popisem výhod a nevýhod byl splněn i poslední bod zadání bakalářské práce.

LITERATURA

[1] KORMANYOS, Christopher. Real-Time C++. 3. Berlin: Springer, 2018. ISBN 978-3-662-56718-0.

[2] Floating-point to Fixed-point conversion. Sharif University of Technology: Electrical Engineering Department [online]. [cit. 2021-12-27]. Dostupné z: https://ee.sharif.edu/~asic/Tutorials/Fixed-Point.pdf

[3] Introductory Overview Week2: MSP430 Teaching Materials. Hacettepe University: Department of Electrical and Electronics Engineering [online]. 2009 [cit. 2021-12-27]. Dostupné z: http://www.ee.hacettepe.edu.tr/~alkar/ELE417/Week2_hacettepe_introduction.pdf

[4] PADGETT, Wayne T. a David V. ANDERSON. Fixed-Point Signal Processing. Morgan & Claypool, 2009. ISBN 9781598292596.

[5] Fixed-Point Format. ResearchGate [online]. 2008, 2009 [cit. 2021-12-27]. Dostupné z: https://www.researchgate.net/figure/Representation-of-the-floating-point-and-fixed-point-formats_fig1_225139564

[6] A Fixed-Point Introduction by Example. DSP Related [online]. 25.5.2011 [cit. 2021-12-27]. Dostupné z: https://www.dsprelated.com/showarticle/139.php

[7] Overflow in the Fixed-Point Data Type. NI [online]. [cit. 2021-12-27]. Dostupné z: https://www.ni.com/documentation/en/labview-comms/latest/data-types/overflow-fixed-point-data/

[8] Architektury počítačů: České vysoké učení technické, Fakulta elektrotechnická. ČVUT FEL [online]. 2014 [cit. 2021-12-27]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a0b36apo/tutorials/02/apo2.pdf

[9] Rounding in the Fixed-Point Data Type. NI [online]. 9.8.2019 [cit. 2021-12-27]. Dostupné z: https://www.ni.com/documentation/en/labview-comms/5.0/data-types/rounding-fixed-point-data/

[10] LabVIEW Fundamentals. David Kleinfeld Laboratory at UC San Diego [online]. 2005 [cit. 2021-12-27]. Dostupné z: https://neurophysics.ucsd.edu/Manuals/National%20Instruments/LV_Fundamentals.pdf

[11] VI Server. LabVIEW Wiki [online]. 2022 [cit. 2022-05-13]. Dostupné z: https://labviewwiki.org/wiki/VI_Server

[12] GRÉZL, Vojtěch. Nástroj pro simulaci výpočtů s pevnou řádovou čárkou [online]. Brno, 2022 [cit. 2022-05-14]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/137969. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Martin Čala.

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

LabVIEW	Laboratory Virtual Instrument Engineering Workbench, vývojové prostředí
VI	Virtual Instrument, nástroj v prostředí LabVIEW
SubVI	Standartní VI, jenž může využívat jiné VI

Symboly:

b_i	digitální hodnota
k	radix (pro dvojkovou soustavu se jedná o hodnotu 2)
n	počet bitů nalevo od radix pointu
m	počet bitů napravo od radix pointu
$x_{(10)}$	číslo v desítkové soustavě
r	rozlišení
n	celkový počet bitů
p	počet bitů pro celočíselnou část
q	počet bitů pro desetinnou část

SEZNAM PŘÍLOH

Příloha A – Přiložené CD obsahující elektronickou formu bakalářské práce a zdrojový kód programu v prostředí LabVIEW.