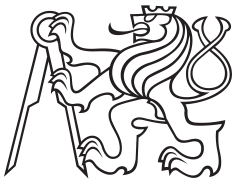
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Kalkulátor na přípravku DE10-Lite v jazyce VHDL

Magdalena Folková

Vedoucí práce: Ing. Pavel Lafata, Ph.D.
Studijní program: Elektronika a komunikace
Leden 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Folková** Jméno: **Magdalena** Osobní číslo: **507270**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Kalkulátor na přípravku DE10-Lite v jazyce VHDL

Název bakalářské práce anglicky:

Calculator Using DE-Lite Kit in VHDL

Pokyny pro vypracování:

Seznamte se s přípravkem DE10-Lite (FPGA MAX10) a jeho obsluhou pomocí jazyka VHDL. K přípravku připojte jednoduchou maticovou klávesnici, využijte přepínače a tlačítka na přípravku a segmentový displej, případně navrhněte připojení dalšího displeje k přípravku. Navrhněte a v jazyce VHDL realizujte komplexní kalkulátor s využitím přípravku DE10-Lite a uvedených periferií. Kalkulátor bude obsahovat funkce a operace: sčítání, odečítání, násobení a dělení (s uvažovanými omezeními), práci se zápornými a desetinnými čísly, převod číselných soustav (desítková, dvojková, šestnáctková), funkce paměti a její vyvolání apod. Výstupem budou vytvořené VHDL kódy a knihovny pro realizaci kalkulátoru na přípravku DE10-Lite.

Seznam doporučené literatury:

[1] Lafata, P. - Hampl, P. - Pravda, M.: Digitální technika. 1. vyd. Praha: Česká technika - nakladatelství ČVUT, 2011. 164 s. ISBN 978-80-01-04914-3.
[2] Pinker, J. - Poupa, M.: Číslicové systémy a jazyk VHDL. Praha : BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.
[3] Ashender, P., J.: The VHDL Cookbook [online]. Dostupné z: https://tams-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf.
[4] Terasic: DE10-Lite User Manual [online]. Dostupné z: https://www.intel.com/content/dam/www/programmable/us/en/portal/dsn/42/doc-us-dsnbk-42-2912030810549-de10-lite-user-manual.pdf

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Lafata, Ph.D. katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Pavel Lafata, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studentky

Poděkování

Tímto bych chtěla poděkovat vedoucímu bakalářské práce Ing. Pavlu Lafatovi, Ph.D. za odborné vedení, ochotu, poskytování cenných rad a zejména za vypůjčení přípravku DE10–Lite, znakového displeje, klávesnice a příslušenství.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 16. května 2024

Abstrakt

Tato bakalářská práce se zabývá návrhem a realizací kalkulátoru na přípravku DE10–Lite v jazyce VHDL. Ve výsledném zapojení se využívá periferií na přípravku pro řízení a připojení maticové klávesnice a znakového LCD displeje. Výstupem je kalkulátor s matematickými operacemi sčítání, odčítání, násobení a dělení, funkcí paměti a režimem převodu soustav. Významná část práce se věnuje architektuře FPGA pole, popisu přípravku DE10–Lite, obsluze maticové klávesnice a obsluze znakového LCD displeje s řadičem HD44780U.

Klíčová slova: DE10–Lite, FPGA, maticová klávesnice, VHDL, znakový LCD displej

Vedoucí práce: Ing. Pavel Lafata, Ph.D.

Abstract

This bachelor thesis designs a math calculator by configuring the DE10–Lite development board. The configuration is accomplished by using the VHDL description language. A matrix keypad and LCD character display module are connected to the development board. Several components of the development board are used for operating the calculator. The final product handles the main arithmetic operations and also includes memory function and number system conversion. Part of this bachelor thesis focuses on describing FPGA architecture, DE10–Lite development board, matrix keypad and LCD character display module that uses the HD44780U controller.

Keywords: DE10–Lite, FPGA, matrix keypad, VHDL, LCD character display module

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Cílem této bakalářské práce je realizovat funkční kalkulátor na přípravku DE10–Lite od firmy Terasic prostřednictvím popisu v jazyce VHDL. Výsledný návrh má mít implementované matematické operace sčítání, odčítání, násobení a dělení, kalkulátor by měl zvládat práci se zápornými a desetinnými čísly a obsahovat převody soustav i funkci paměti. Pro komunikaci s uživatelem bude využitá maticová klávesnice a znakový LDC displej.

V první, teoretické, části práce budou popsány jednotlivé fyzické komponenty – vývojová deska DE10–Lite, znakový LCD displej 20×4 a maticová klávesnice 4×4. Další kapitoly jsou věnovány FPGA polím a jazyku VHDL. Praktická část práce se zabývá zapojením klávesnice a displeje k přípravku a jejich konfigurací. Dále rozebírá implementaci jednotlivých funkcí i související omezení. V praktické části je také zobrazeno celkové zapojení komponent.

Kapitola 2
Teoretická část

2.1 FPGA

2.1.1 Historie

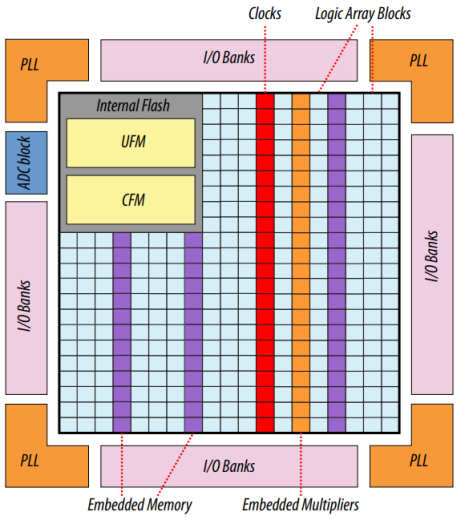
Vývoj technologie pro realizaci logických členů začal reléovými a mechanickými prvky. Po objevu polovodičů vznikla diodová logika, která umožňovala sestavení hradel OR a AND. Následovaly diodově tranzistorová logika, odporově tranzistorová logika a tranzistorově tranzistorová logika – TTL. Technologie TTL používaly bipolární tranzistory a dokázaly realizovat funkce AND, OR a NOT. Tato logika byla použita v integrovaných obvodech. Největší revoluci na poli logických hradel způsobil vynález CMOS technologie, jejímž základem jsou unipolární tranzistory. CMOS logika dovoluje především mnohem větší hustotu integrace v IO [1].

Zcela odlišný přístup řešení realizace logických hradel představují programovatelné obvody PLD (*Programmable Logic Device*). Jejich předchůdcem jsou paměti PROM (*Programmable ROM*), které mají předdefinovaná hradla AND a programovatelná hradla OR [2]. Jakmile byla jednou data do PROM zanesena, již je není možné změnit, paměť dále slouží jen pro čtení. Toto zásadní omezení charakterizuje i první typy PLD, jmenovitě PLA (*Programmable Logic Array*) a PAL (*Programmable Array Logic*). Se vznikem EPROM (*Erasable PROM*) a EEPROM (*Electrically Erasable PROM*) pamětí, které byly navrženy k odstranění problematiky jednorázového programování, se objevil EEPLD (*Electrically Erasable PLD*) obvod GAL (*General Array Logic*). Jedná se v zásadě o PAL s možností opětovného přeprogramování [1] [2].

2.1.2 Architektura

Programovatelné logické obvody FPGA (*Field Programmable Gate Array*) jsou integrované obvody, které slouží k realizaci logických funkcí. Principiálně vycházejí z předchůdce programovatelných polí, jednorázově programovatelných PROM (*Programmable Read-Only Memory*), a využívají pamětí pro definování logických funkcí. Paměti fungují jako vyhledávací tabulky LUT (*Look-Up Table*) [1].

FPGA tvoří matici logických bloků, které jsou vzájemně propojeny. Ačkoliv se jejich uspořádání a modifikace mohou lišit u jednolivých výrobců i vývojových řad, výchozí architektura zůstává stejná. Jediným významnějším rozdílem může být způsob označení konkrétních bloků, které se u každého výrobce liší. Názvosloví používané dále odpovídá terminologii vyprodukované firmou Altera. Základním prvkem jsou LAB (*Logic Array Block*) bloky. Každý LAB obsahuje logické buňky LE (*Logic Element*), které ve vývojové řadě Intel MAX 10 představují nejmenší logickou jednotku [3]. Součástí jednotlivých LE jsou LUT tabulky. Vyhledávací tabulka LUT plní funkci programovatelné pravdivostní tabulky, díky níž lze generovat logické funkce. Samotné LAB bloky jsou vzájemně metalicky spojeny maticí cest a propojů PSM (*Programmable Switch Matrix*). Součástí architektury FPGA jsou dále paměťové bloky volatilní paměti, násobičky, vnitřní Flash paměť pro ukládání nevolatilních informací, síť pro distribuci hodinového signálu, PLL (*Phase-Locked Loops*) závěsy, I/O bloky a ADC (*Analog-to-Digital Converter*) převodníky analogového signálu na číslicový [3].



Obrázek 2.1: Architektura FPGA řady Intel MAX10 [3, strana 4]

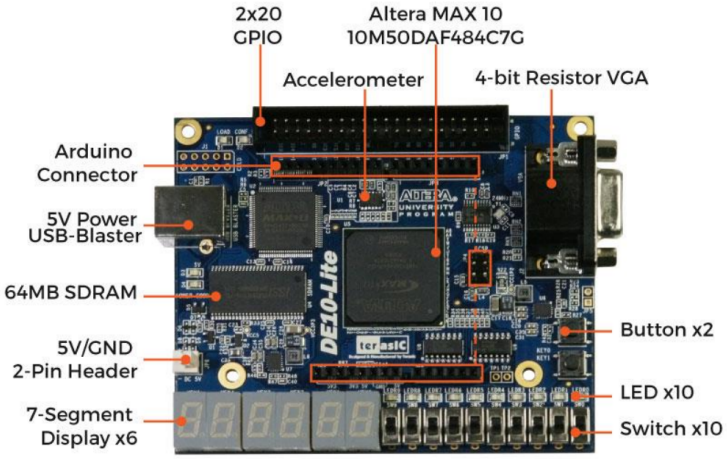
Značnou výhodou FPGA jsou nízké náklady a s tím související dostupnost koncovým uživatelům. Díky programovacím jazykům sloužícím pro návrh obvodu lze obvody opětovně přeprogramovat a přizpůsobit konkrétním zákaznickým obvodům a aplikacím.

FPGA je s nejvyšší hustotou integrace nejpokročilejším typem programovatelného obvodu [1]. Mezi přední výrobce FPGA polí se řadí firmy Xilinx, kterou koupila firma AMD, a Altera, kterou vlastní společnost Intel.

2.2 Přípravek DE10–Lite

Vývojová deska DE10–Lite od firmy Terasic je postavená kolem FPGA MAX 10. Obsahuje 2×20 GPIO (*General-Purpose Input/Output*) kontaktů, USB

konektor, 10 uživatelem nastavitelných LED diod a 4 indikační LED diody, 10 přepínačů, 2 tlačítka, 6 sedmisegmentových displejů a další komponenty. Veškeré spoje jsou vedeny přes hradlové pole MAX 10 [4]. Celkový layout přípravku ilustruje Obrázek 2.2. Součástí kitu je také generátor hodin. Uživatel má k dispozici dva hodinové signály s frekvencí 50 MHz a jeden hodinový signál s frekvencí 10 MHz [4].



Obrázek 2.2: Layout desky DE10–Lite [4]

Hlavní složkou přípravku DE10–Lite je již zmíněné FPGA MAX 10 od firmy Altera s označením 10M50DAF484C7G [5]. Tento kód specifický pro každý typ produktu označuje základní charakteristiky konkrétního FPGA. Význam kódu přípravku DE10–Lite je rozepsán v následující tabulce [5]:

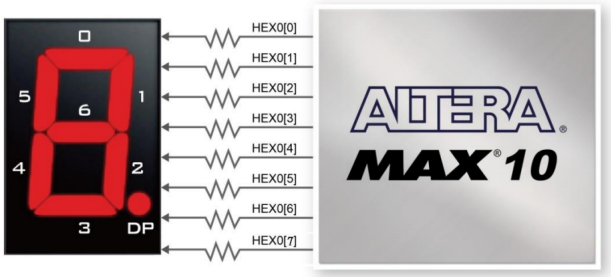
10M	rodina Intel MAX 10
50	50 000 logických buněk (LE)
DA	duální konfigurace, integrované ADC
F	malé vzdálenosti mezi jednotlivými piny
484	484 pinů
C	provozní teplota od 0 °C do 85 °C
7	stupeň rychlosti - střední
G	indikuje možnosti zařízení

Tabulka 2.1: Význam kódu FPGA na přípravku DE10–Lite

2.2.1 Sedmisegmentový displej

Na přípravku DE10–Lite je pro zobrazení čísel a jednoduchých znaků k dispozici 6 sedmisegmentových displejů se společnou anodou [4, strana 28]. To znamená, že po přivedení logické nuly se příslušný segment rozsvítí, zatímco při logické jedničce bude zhasnutý. Jednotlivé segmenty jsou systematicky

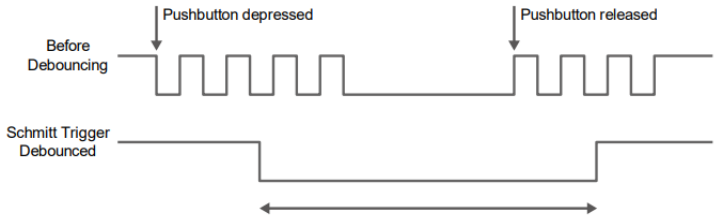
očíslovány, jak uvádí Obrázek 2.3. Pro znázornění desetinné tečky slouží segment DP (*Decimal Point*) [4, strana 28].



Obrázek 2.3: Sedmisegmentový displej [4, strana 28]

2.2.2 Tlačítka, přepínače, LED diody

Pro komunikaci s uživatelem či indikaci stavu slouží na přípravku DE10–Lite tlačítka, přepínače a LED diody. Tlačítka jsou k dispozici dvě a v případě stisknutí se jejich stav změní z logické 1 na logickou 0 [4, strana 25]. Tlačítka využívají Schmittův klopný obvod, aby se na výstupu neprojevily zákmity způsobené jejich stisknutím [4, strana 25].



Obrázek 2.4: Potlačení zákmitů [4, strana 25]

Přepínače, obdobně jako tlačítka, představují uživatelem nastavitelnou vstupní hodnotu FPGA pole. Horní poloha přepínače odpovídá logické 1, zatímco v dolní poloze se odesílá logická 0 [4, strana 26].

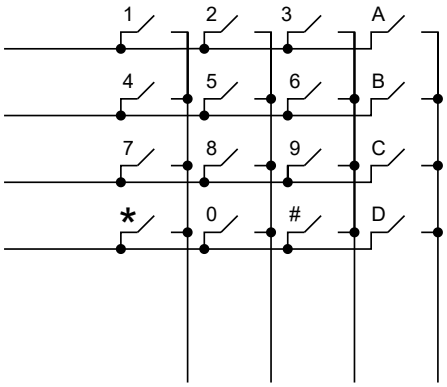
LED diody jsou narozdíl od tlačítek a přepínačů výstupním prvkem. Při logické 1 je LED dioda rozsvícená, zatímco při logické 0 zhasnutá [4, strana 27]. Jednotlivá tlačítka, přepínače a LED diody jsou připojeny k příslušným pinům na FPGA poli [4, strany 6, 25, 26, 27].

2.2.3 GPIO

Port 2×20 GPIO umístěný na desce DE10–Lite disponuje 40 piny, z nichž jeden reprezentuje 5V napájení, jeden 3,3V napájení a dva piny představují kontakt na zem [4, strana 30]. Zbývajících 36 pinů tvořících rozhraní FPGA pole lze využít pro vstupní i výstupní digitální signály.

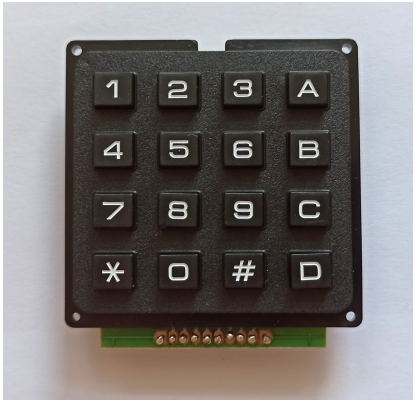
2.3 Maticová klávesnice

Maticová klávesnice představuje na počet kontaktů úsporné řešení pro zapojení většího množství tlačítek [6]. Tlačítka jsou uspořádána v matici s vyvedenými kontakty na jednotlivé řádky a sloupce [6]. Při stisknutí konkrétní klávesy se vodivě propojí řádek a sloupec, ve kterém je umístěna [6]. V práci je použita klávesnice 4×4, která má celkem 8 I/O kontaktů. Princip činnosti znázorňuje Obrázek 2.5.



Obrázek 2.5: Maticová klávesnice [vlastní obrázek, [6]]

Stisknutou klávesu lze rozpoznat následujícím způsobem: Na jednotlivé řádky se postupně vysílá zvolená logická hodnota a současně se zkoumá, jestli nebyla zaznamenána na některém ze sloupců [6]. V případě detekce logické hodnoty na konkrétním sloupci se použitá klávesa určí snadno, protože známe řádek, který právě hodnotu vyslal. Tento postup lze aplikovat i obráceně, a to posíláním logických hodnot na sloupce a jejich detekováním na řádcích.



Obrázek 2.6: Maticová klávesnice [vlastní obrázek]

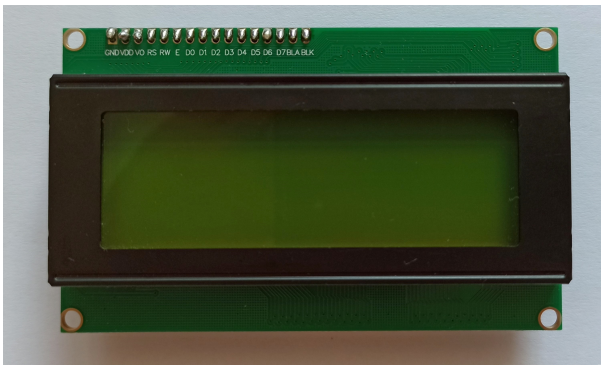
Nevýhodou maticových klávesnic je, že nelze najednou detekovat více

stisknutých kláves ve stejném sloupci (pro variantu odesílání logických hodnot na řádky).

2.4 LCD displej

Znakový LCD (*Liquid Crystal Display*) s řadičem HD44780U od firmy Hitachi dokáže zobrazit celou řadu znaků včetně latinské abecedy, arabských číslic či japonského písma. K dispozici je navíc paměť pro definování omezeného množství vlastních znaků [7, strana 1]. Kromě toho je poskytnuta možnost nastavení kontrastu displeje [8]. Do znakového dipleje lze data narozdíl od sedmisegmentového dipleje nejen zapisovat, ale také je z dipleje číst.

Značnou výhodou znakových displejů je jejich dostupnost a díky 5V napájení také nízká energetická náročnost. Lze zakoupit řadu znakových LCD, které se liší velikostí i barvou podsvícení. V této práci je využit LCD 20×4.



Obrázek 2.7: Znakový LCD displej [vlastní obrázek]

2.4.1 DDRAM, CGROM, CGRAM

O vypisování konkrétních znaků na požadované pozice na displeji se starají tři paměti:

- DDRAM
- CGROM
- CGRAM

DDRAM (*Display Data RAM*) paměť uchovává až 80 adres, které odpovídají příslušným pozicím na dipleji. Adresy mohou být v DDRAM uspořádány dvěma způsoby [7, strany 10, 11]. První způsob představuje seřazení všech 80 lokací do jednoho souvislého bloku. Druhý způsob rozdělí adresy na dva bloky (řádky) po 40 adresách, jak znázorňuje Obrázek 2.8. Adresy jsou zapsány hexadecimálně [7, strany 10, 11].

Hlavní rozdíl oproti jednořádkovému uspořádání spočívá ve zdánlivé nenávaznosti adres při přechodu z jednoho řádku na druhý [7, strana 11]. Důvodem

je odlišení adres prvního a druhého bloku paměti. Pokud by se adresy přepsaly do binárního tvaru, první a druhý řádek se vždy bude lišit hodnotou v bitu s indexem šest [9].

Display position	1	2	3	4	5	.....	39	40
DDRAM address (hexadecimal)	00	01	02	03	04	.....	26	27
	40	41	42	43	44	.....	66	67

Obrázek 2.8: DDRAM – dvouřádkový displej [7, strana 11]

Konkrétní uspořádání adres v DDRAM paměti je dáno rozlišením a konfigurací znakového dipleje a nastavuje se během procesu inicializace. Například rozdělení adres do dvou bloků je vhodnější pro víceřádkové dipleje.

Paměť CGROM (*Character Generator ROM*) na základě osmibitového znakového kódu generuje konkrétní formu symbolu, ve které bude zobrazen na displeji [7, strany 13, 17]. Všechny použitelné symboly a jim odpovídající kódy jsou zobrazeny na Obrázku 2.9. Při inicializaci displeje se volí velikost zobrazovaných znaků 5×8 nebo 5×10 pixelů. Větší rozlišení poskytuje dodatečných 32 symbolů, ale současně s sebou přináší i jistá omezení. Ve formátu 5×10 pixelů například není možné vypisovat současně na dva řádky bezprostředně pod sebou [7, strany 13, 20, 29].

CGRAM (*Character Generator RAM*) paměť ponechává prostor pro definování až osmi vlastních znaků. Nevyužitá část DDRAM a CGRAM funguje jako standardní operační paměť RAM [7, strana 13].

2.4.2 Registry

K dočasnému ukládání dat slouží dva registry s kapacitou 8 bitů – IR (*Instruction Register*) a DR (*Data Register*). Každý z nich slouží k uchování odlišného typu dat [7, strana 9].

IR ukládá přesně definované kódy pro instrukce, které probíhají například při počáteční inicializaci displeje, nebo pro nastavení adres. V režimu čtení z displeje se na nejvyšší pozici IR zaznamená informace o probíhajících operacích (*busy flag*) a na nižší pozice adresa, na které se nachází kurzor [7, strany 9, 24].

Do DR se ukládají data, která budou následně zapsána do DDRAM. V případě čtení z displeje se do DR zkopírují data uložená do aktuální adresy (pozice na displeji) [7, strany 9, 24, 25].

Výběr konkrétního registru řídí signál RS (*Register Selector*). Pokud je hodnota RS rovná logické 0, použije se IR, zatímco pro logickou 1 je zvolen DR [7, strana 9]. Obdobně se volí mezi režimem zapisování a čtení dat pomocí signálu R/W (*Read / Write*). Stav zapisování dat odpovídá nastavení R/W na logickou 0, čtecí režim je aktivován v opačném případě [7, strana 9].

Obrázek 2.9: CGROM – tabulka [7, strana 17]

2.4.3 Inicializace

Pro správnou funkčnost je nejprve nutné nastavit chování displeje a způsob komunikace v procesu inicializace. Musí se definovat velikost sběrnice, přes kterou bude komunikace probíhat, rozměry znaků, směr, kterým se bude posouvat kurzor, aj. Inicializační příkazy mají jednoznačně definovanou strukturu a jsou určeny signály RS, R/W a datovou sběrnicí DB0 až DB7 [7, strany 23, 24].

Příkazem *Function set* se nastavuje velikost datové sběrnice (DL), počet řádků displeje (N) a rozměr vypisovaných znaků (F) [7, strana 27]. Mezi displejem a kitem může probíhat 4bitová nebo 8bitová komunikace. Při 8bitové komunikaci se využije celá datová sběrnice. Pro 4bitovou komunikaci se připojí pouze piny DB4 až DB7, čímž se sníží počet využívaných portů na DE10–Lite. Hlavní rozdíl mezi těmito dvěma konfiguracemi spočívá ve způsobu přenosu dat, kdy pro 4bitovou komunikaci musí být data odeslána dvakrát po sobě. První se posílají horní 4 bity následovány 4 spodními bity [7, strany 22, 27].

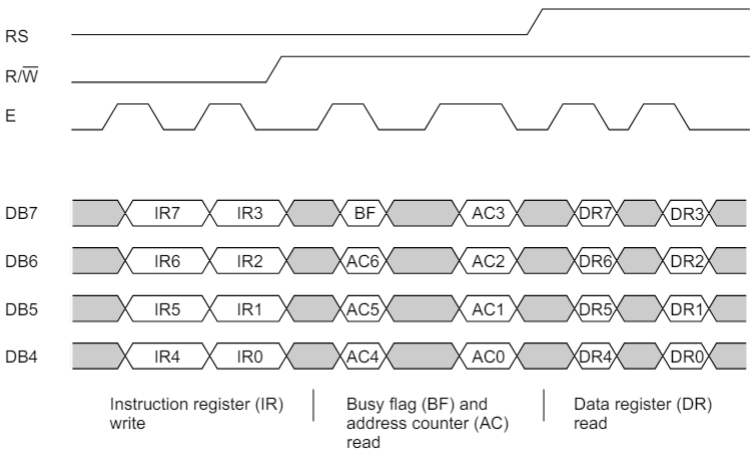
Situaci popisuje Obrázek 2.10. 8bitová komunikace odpovídá stavu DL = 1, zatímco pro zvolení druhého způsobu komunikace se signál DL nastaví na logickou 0 [7, strana 25].

Volba počtu řádků displeje rozhodne o způsobu uspořádání adres v DDRAM, viz kapitola 2.4.1. Pro N = 0 jsou adresy uloženy do jednoho řádku, naopak pro N = 1 jsou rozděleny do dvou řádků [7, strany 10, 11]. Rozměr znaků na dipleji může být 5×8 (F = 0) nebo 5×10 pixelů (F = 1). Více viz kapitola 2.4.1. Po odeslání instrukce *Function set* již není možné nastavení počtu řádků a rozměru znaků měnit [7, strany 40, 42].

Dalším příkazem realizovaným v rámci inicializace displeje je *Display on/off control*. Řídí vypisování dat na diplej (D), zobrazení kurzoru (C) a aktivuje blikání (B) [7, strana 26]. Při nastavení signálu D na logickou 1 se budou data z DDRAM vypisovat na displej. V případě D = 0 data zůstanou v paměti DDRAM [7, strana 26]. Pro označení aktuální pozice na dipleji, která odpovídá právě používané adrese v DDRAM, je možné zobrazit kurzor (C = 1), nebo aktivovat blikání (B = 1) všech pixelů příslušících této adrese.

Entry mode set konkretizuje autoinkrementaci (I/D) a posun displeje (S) [7, strana 26]. Po zápisu nebo čtení ze stanovené DDRAM adresy se kurzor automaticky posune na sousední adresu. I/D specifikuje, jestli se adresa bude inkrementovat (I/D = 1) nebo dekrementovat (I/D = 0) [7, strana 26]. Řadič HD44780U nabízí možnost namísto posouvání kurzoru posunout celý displej volbou S = 1 [7, strana 26]. V takovém případě se v definovaném směru posunují adresy DDRAM [7, strany 10, 11, 12, 26].

Velmi užitečný je příkaz *Clear display*, který do všech adres DDRAM uloží znak pro mezeru a na první pozici na displeji nastaví adresu 00, čímž se anuluje jakýkoliv posun displeje, a současně na tuto adresu přemístí kurzor [7, strany 24, 26]. Pro vrácení pozice displeje a kurzoru do výchozího stavu lze také použít příkaz *Return home*, který ale nemění data uložená v DDRAM paměti [7, strany 24, 26].



Obrázek 2.10: Komunikace přes 4bitovou sběrnici [7, strana 22]

2.4.4 Vypisování znaků

Po procesu inicializace je možné na displej vypisovat znaky. Příkazem *Set DDRAM address* se kurzor posune na adresu, na které má být daný znak zobrazen [7, strana 24]. Díky autoinkrementaci není nutné nastavovat adresu před každým vypsáním znaku, ale jen v případech nestandardního posunu kurzoru na vzdálené pozice [7, strana 24]. Po nastavení správné adresy do ní lze uložit data reprezentující požadovaný znak, který následně bude na odpovídající pozici na displeji zobrazen. Bezprostředně po operaci zápisu dat do DDRAM se adresa automaticky inkrementuje (případně dekrementuje) [7, strana 25].

Každá poslaná instrukce spustí vnitřní operaci řadiče, během které nemohou být přijímána další data. Informaci o probíhající vnitřní operaci v sobě uchovává signál BF (*Busy Flag*), přičemž jediný realizovatelný příkaz při běhu vnitřní instrukce je právě *Read busy flag & address*. Pokud je BF = 1, řadič se nachází v režimu vnitřní operace, když se BF přepne do stavu logické 0, je možné provést další instrukce. Před provedením dalšího příkazu musí být buď stav BF zkontrolován, nebo je zapotřebí implementovat dostatečně velké zpoždění.

Čtení nebo zápis dat zahajuje signál *Enable* (E), jak je znázorněno na Obrázku 2.10. Instrukce jsou vždy provedeny po změně signálu E z logické 1 na logickou 0 [7, strany 8, 33] [8].

2.5 VHDL

VHDL (VHSIC Hardware Description Language) je univerzální jazyk určený pro popis hardwarového zapojení elektronických systémů včetně FPGA polí [1, 10]. Zkratka VHSIC pochází z *Very High Speed Integrated Circuits Program*, což byl výzkumný projekt spuštěný v 80. letech vládou Spojených států [10]. Jedná se o velmi komplexní jazyk, který využívá známých postupů programovacích jazyků (např. jazyka C) [10].

VHDL jazyk umožňuje modelovat logické obvody ve třech rovinách při různých úrovních abstrakce [11]. První způsob s nejvyšší úrovní abstrakce představuje behaviorální popis, který specifikuje chování logického obvodu. Pro popis chování obvodu se využívá například pravdivostní tabulka. Dataflow neboli RTL (*Register Transfer Level*) popis obvodu se soustředí na charakterizování jeho činnosti a využívá Booleovské rovnice. Třetí možností je strukturální popis s nejnižší mírou abstrakce. Při strukturálním popisu logického obvodu je narozdíl od předchozích dvou způsobů známé konkrétní zapojení jednotlivých hradel [11].

Při popisu chování a činnosti se pro převod na výsledný obvod používá syntezátor. Po kontrole syntaxe kódu syntezátor v rámci procesu kompilace generuje na základě popsaného modelu konkrétní zapojení obvodu. Následně syntezátor jednotlivým částem FPGA pole přidělí určité bloky sestaveného logického obvodu. Výhodou jazyka VHDL je možnost simulace návrhu logického obvodu ve kterékoliv fázi [1]. Jazyk VHDL je definován standardem IEEE

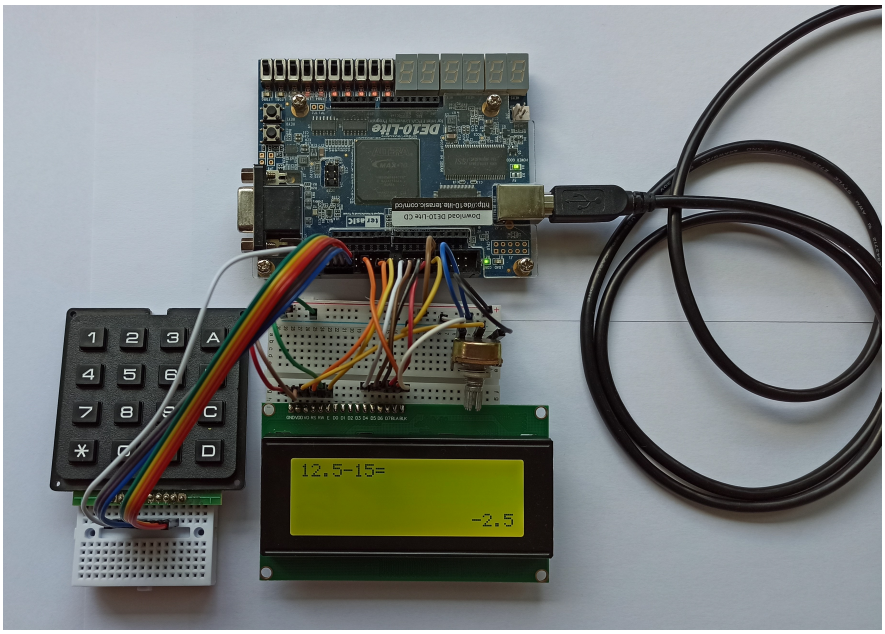
(*Institute of Electrical and Electronics Engineers*), a mělo by proto být možné jej aplikovat ve všech elektronických systémech nezávisle na výrobci [10].

Jazyk VHDL poskytuje paralelní i sekvenční prostředí. V paralelním prostředí jsou všechny příkazy vykonávány najednou, zatímco v sekvenčním prostředí proběhnou v pořadí zápisu. Prostředí VHDL jazyka je defaultně paralelní, a z toho důvodu musí být sekvenční prostředí uzavřené v procesu. Proces je spuštěn při změně hodnoty kterékoliv z proměnných uvedených v tzv. citlivostním seznamu. Pokud je v kódu implementováno více procesů, mohou být vykonávány současně [10]. Paralelní a sekvenční prostředí lze v návrhu kombinovat [11].

Kapitola 3

Praktická část

Pro návrh kalkulátoru v jazyce VHDL byl použit program Quartus Prime Lite Edition od firmy Intel. Výsledné zapojení je zobrazeno na Obrázku 3.1. Následující část se zabývá konfigurací každé z komponent a jejich vzájemným propojením.



Obrázek 3.1: Celkové zapojení [vlastní obrázek]

3.1 VHDL struktury

Výsledný návrh zapojení popsaný v jazyce VHDL může být pro větší přehlednost rozdělen do více komponent, neboli dílčích bloků. Rozhraní každé komponenty je určeno porty (tj. vstupy a výstupy), které se definují při deklaraci komponenty [11]. Porty mohou být vstupní (*in*), výstupní (*out*), vstupně výstupní (*inout*) nebo výstupní se zpětnou vazbou (*buffer*) [12].

Datové objekty ukládají či přenášejí nějaké hodnoty. Mezi datové objekty se řadí signály (*signal*), proměnné (*variable*), nebo konstanty (*constant*) [12].

Signál má fyzickou reprezentaci a představuje vodič. Používá se například k propojení komponent. Signály mohou být jakéhokoliv datového typu a jsou globální, což znamená, že existují v celé architektuře. Oproti tomu proměnná nemá fyzickou reprezentaci, jedná se o místo v paměti, na které lze uložit konkrétní hodnotu. Proměnná je lokální, existuje jen v procesu, ve kterém byla deklarovaná. Zásadní rozdíl mezi signálem a proměnnou spočívá v rychlosti obnovení hodnoty datového objektu. Změna hodnoty proměnné se provede okamžitě po zadání příkazu, zatímco hodnota uložená do signálu se aktualizuje až po skončení cyklu, procesu [12].

V sekvenčním prostředí, procesu, lze pro větvení kódu využít zejména dvou podmínkových struktur: case–when a if–else [13]. Struktura if–else rozhoduje, která část kódu se vykoná na základě uvedené podmínky. Tyto podmínky se postupně prochází a pokud se některá vyhodnotí jako pravdivá, provede se příslušná sekvence příkazů [10] [13]. Struktura case–when vykonává určitou sekvenci příkladů na základě hodnoty uložené v testovaném datovém objektu. Pro stejnou hodnotu nelze vyhodnotit více větví kódu a zároveň musí existovat sekvence příkazů pro každou variantu hodnoty datového objektu. Z toho důvodu struktura case–when často končí sekcí „*when others*", která zahrnuje zbývající možnosti [10] [13]. Příklady case–when a if–else struktur jsou uvedeny na Obrázku 3.3.

3.2 Připojení maticové klávesnice

Na maticové klávesnici použité v zapojení jsou příhodně zobrazeny číslice 0 – 9, nicméně klávesnice neobsahuje symboly pro matematické operace ani desetinnou čárku. Z toho důvodu v dalších částech návrhu klávesy s písmeny *A, B, C, D* reprezentují po řadě operace sčítání, odečítání, násobení a dělení. Klávesa se znakem hvězdičky nahrazuje symbol pro desetinnou tečku a pod křížkem se skrývá značka pro vyhodnocení operace (=).

Maticová klávesnice 4×4 disponuje osmi I/O kontakty, které jsou k přípravku DE10–Lite připojeny prostřednictvím pinů ze slotu GPIO. Kontakty na řádky jsou v této implementaci použity jako výstupní porty a sloupce jako vstupní. Na řádky, *rows*, se v pravidelných intervalech odesílají logické hodnoty a současně se detekují na vstupních sloupcích, *cols*. Aby z důvodu nadměrného šumu nedošlo k vyhodnocení chybného sloupce, přiřazuje se právě testovanému řádku logická 0.

Přípravek DE10–Lite disponuje hodinovými signály o dvou frekvencích: 50 MHz a 10 MHz. Pro řízení odesílání hodnot na řádky postačí pomalejší signál v řádu desítek či nižších stovek Hz. Pro zpomalení běhu hodinového signálu se využije dělička kmitočtu, jejímž výstupem je signál *clk_slow*. VHDL kód děličky zobrazuje Obrázek 3.2.

S každou náběžnou hranou zpomaleného hodinového signálu *clk_slow* se spustí proces pro obsluhu klávesnice, v němž se na základě hodnoty uložené v proměnné *cyc_count* na příslušný řádek odešle logická 0. Proměnná *cyc_count* představuje počítadlo cyklů a na konci každého spuštění procesu se její obsah inkrementuje o 1. V procesu je pro řízení přiřazování hodnot

```
--zpomalení 10MHz hodinového signálu
process(clk)
   variable clk_count : unsigned(17 downto 0):=(others=>'0');
begin

   if clk='1' and clk'event then
      clk_count:= clk_count + 1;
   end if;

   clk_slow <= std_logic(clk_count(17));
end process;
```

Obrázek 3.2: Dělička hodinového signálu [vlastní obrázek]

implementováno větvení case–when, do kterého je pro účely čtení ze sloupců, *cols*, vnořená druhá struktura case–when. Konkrétní podoba VHDL kódu pro část prvního řádku klávesnice je zobrazena na Obrázku 3.3, pro zpracování stisku dalších kláves se postupuje analogicky. Při detekci stisknuté klávesy rovnou dojde k přiřazení příslušného čísla do proměnné u, na základě které se provádí další vyhodnocení, viz kapitola 3.6.

```
case cyc_count is
when "00" =>
   rows<="1011";

   case cols is
      when "0111" =>                          --1
         if curs<2 then vystup:= null;
         elsif detect_0='0' then u:=1;
         end if;

      when "1011" =>                          --2
         if curs<2 then vystup:= null;
         elsif detect_0='0' then u:=2;
         end if;

      when "1101" =>                          --3
         if curs<2 then vystup:= null;
         elsif detect_0='0' then u:=3;
         end if;
```

Obrázek 3.3: Detekce stisku kláves 1, 2, 3 [vlastní obrázek]

Proměnné *detect_0, detect_1, detect_2* a *detect_3* zajišťují, že při držení kterékoliv klávesy nedojde k neustálému vypisování příslušného znaku na LCD displej. Dokud neproběhne pro dotčený řádek jeden cyklus aniž by byl detekován stisk klávesy na jakémkoliv sloupci (tj. klávesa byla uvolněna), znak se na displej ani do proměnné pro výpočet nevypíše. Tím se zajistí, že delší držení klávesy neovlivní ani výsledek operace.

3.3 Připojení LCD

3.3.1 Fyzické zapojení

Znakový LCD displej disponuje celkem 16 piny, ale protože komunikace s přípravkem bude probíhat přes 4bitovou sběrnici, piny DB0 až DB3 zůstanou neaktivní [7, strany 8, 22]. Zbývající piny jsou přes nepájivé pole připojeny k portu 2×20 GPIO, přičemž kromě běžných pinů jsou využity i kontakty na zem (GPIO piny 12 a 30), 5 V (GPIO pin 11) a 3,3 V (GPIO pin 29). Jednotlivé piny jsou popsány označením indikujícím jejich funkci, viz

Obrázek 2.7.

Napěťové úrovně pro napájení (VDD a GND) a podsvícení displeje (BLA a BLK), ke kterým mají být odpovídající piny připojeny, jsou vypsány v tabulce 3.1 [8]. Pin V0 řídí kontrast LCD displeje a připojením k prostřednímu kontaktu 10kΩ potenciometru je možné jas kdykoliv upravit [8] [14]. Krajní vývody potenciometru jsou vodivě spojeny se zemí a napěťovou úrovní 5 V.

GND	zem
VDD	5 V
BLA	3,3 V
BLK	zem

Tabulka 3.1: Připojení pinů LCD [8]

Vývody s označením RS, R/W, E řídí komunikaci a datová sběrnice DB přenáší konkrétní data. Vzhledem k tomu, že se na LCD budou znaky pouze vypisovat, R/W bude konstantně odpovídat úrovni logická 0, a proto se pin připojí přímo na zem. Chování portů RS a E podrobně popisují kapitoly 2.4.2 a 2.4.3.

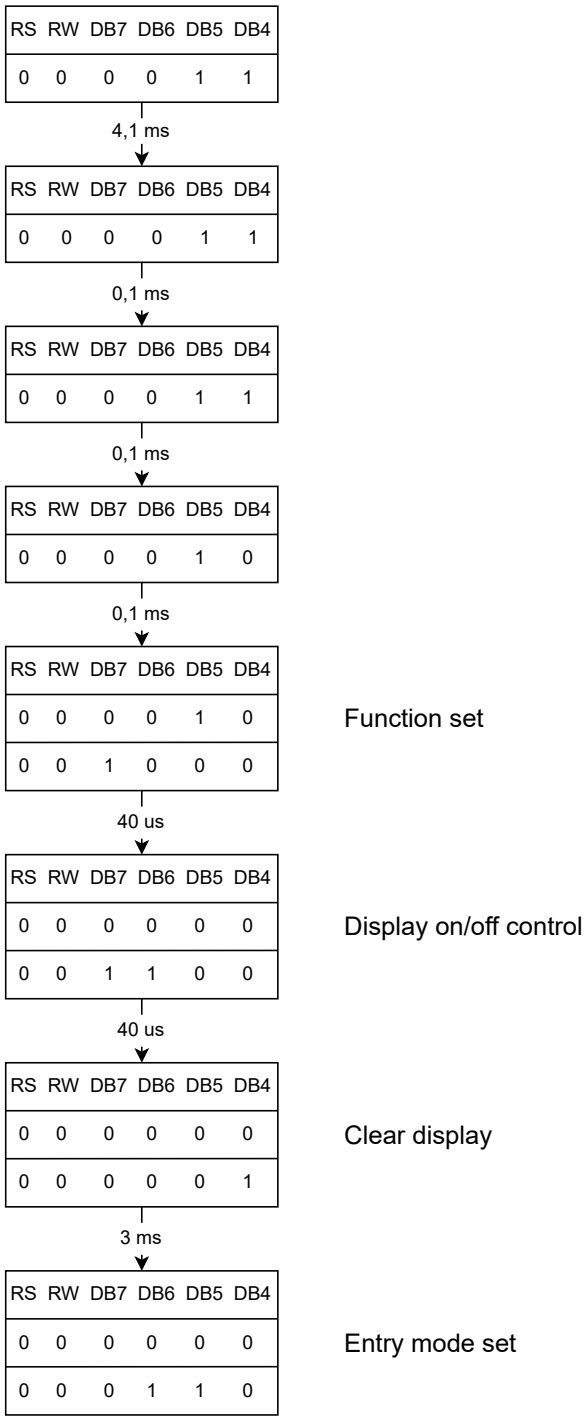
3.3.2 Inicializace

V rámci procesu inicializace se nastaví chování LCD displeje a způsob posílání dat prostřednictvím příkazů popsaných v kapitole 2.4.3. Z důvodu úspory pinů je zvolena komunikace přes 4bitovou sběrnici. Řadič LCD displeje defaultně očekává, že se data budou posílat po 8bitové sběrnici. Pro zavedení 4bitové komunikace je nutné postupovat dle přesně definovaných kroků uvedených na Obrázku 3.4 [7, strana 46] [15]. První čtyři příkazy jsou speciální formou *Function set*. Proces inicializace zahajuje hned třikrát zopakovaný kód 0×3 hexadecimálně. Tato úvodní část inicializace je identická s nastavením 8bitového rozhraní. Následující příkaz, 0×2 hexadecimálně, indikuje zavedení 4bitové komunikace a řadič od této chvíle uvažuje pro přenos dat pouze piny DB7 až DB4 [15].

Po počáteční inicializační sekvenci se odešle standardní *Function set*, kde se kromě 4bitového rozhraní nastaví font 5×8 pixelů (F = 0) a dvouřádkové uspořádání (N = 1). Rozdělení paměti DDRAM do dvou bloků je vhodné použít z důvodu zapojení LCD displeje 20×4 [9].

Dále následují příkazy *Display on/off control*, kde se funkce zobrazení kurzoru a blikání ponechají vypnuté (C = 0, D = 0), *Clear display* a *Entry mode set*. V posledním z uvedených příkazů se nastaví inkrementace pozice kurzoru o 1 (I/D = 1) a posunutí displeje se deaktivuje (S = 0).

Po odeslání inicializačního příkazu 0×2 je možné zjistit hodnotu BF. Protože je ale pin LCD displeje R/W konstantně přiveden na zem, implementuje se namísto čtení BF dostatečně dlouhé zpoždění. Po spuštění většiny instrukcí (např. *Function set*, *Set DDRAM address* nebo *Entry mode set*) vnitřní operace trvá 37 μs [7, strana 24]. Ve výsledné implementaci je mimo jiné kvůli řízení řadiče odlišnou frekvencí uvedeno zpoždění 40 μs. Provedení



Obrázek 3.4: Inicializace [vlastní obrázek, [7, strana 46]]

příkazu *Clear display* je časově náročnější, protože se do každé z adres zapíše znakový kód pro mezeru, a tak jsou pro zpracování vyhrazeny 3 ms [15].

Při komunikaci přes 4bitovou sběrnici se data musí odesílat dvakrát po sobě, aby byl příkaz kompletní a mohla být spuštěna vnitřní operace, viz Obrázek 2.10. Odesílání dat z přípravku DE10–Lite probíhá v následujících krocích:

1. přiřazení horních 4 bitů datové sběrnici
2. odeslání dat (E = 1)
3. deaktivace *Enable* (E = 0)
4. přiřazení dolních 4 bitů datové sběrnici
5. odeslání dat (E = 1)
6. deaktivace *Enable* (E = 0)
7. zpracování příkazu, zahájení vnitřní operace (zpoždění)

Čekání na vykonání vnitřní operace ale není jediné implementované zpoždění. Například puls *Enable* má stanovenou minimální dobu trvání 230 ns, která musí být pro správnou funkčnost dodržena. Zároveň další puls *Enable* nesmí být vyslán dříve než po uplynutí 500 ns od předchozího nastavení *Enable* na logickou 1 [7, strany 52, 58]. Současně bezprostředně po změně registru (hodnoty RS) nemůže být aktivováno odeslání dat, protože náběžná doba signálu odpovídá 40 ns [7, strany 52, 58].

Inicializace a řízení komunikace s řadičem LCD displeje je realizováno v samostatné komponentě *display* s výstupními porty *lcd_e*, *lcd_rs* a *lcd_db*. Konkrétní data komponenta získává z komponenty *keyboard*, která obsluhuje klávesnici a provádí vyhodnocení.

3.4 Operace s čísly

Ve výsledném zapojení kalkulačky jsou implementovány funkce sčítání (+), odčítání (–), násobení (∗) a dělení (:). Současně se pro operace sčítání a odčítání zavedla i desetinná čísla, ačkoliv jsou pro zjednodušení omezena pouze na jedno desetinné místo. Při počítání s desetinnými čísly dojde pro účely zpřesnění výpočtu k posunutí desetinné tečky (tj. vynásobení deseti) a při vypsání výsledku se desetinná tečka zapíše na příslušnou pozici. V případě, že po počítání s desetinnými čísly vyjde celočíselný výsledek, výstup se převede na celé číslo a nebude obsahovat desetinnou tečku.

Operace násobení a dělení nemají pro zjednodušení implementována desetinná čísla. Pro neceločíselné dělení je výsledek oříznut na jedno desetinné místo. Vzhledem k principu vyhodnocování vstupních dat (viz kapitola 3.6) by byly jen obtížně realizovatelné přednosti matematických operací. Z toho důvodu nelze vyhodnotit operaci násobení, pokud byla dříve realizována jiná operace. Protože dělení je svým způsobem specifické tím, že může vyjít desetinné číslo zadáním dvou celých čísel, není na této kalkulačce možné

operaci dělení kombinovat s jinými operacemi (včetně násobení). Pro případy komplexnějších zadání lze využít funkci paměti.

Obdobně kalkulačka nedokáže zpracovat vstup s bezprostředně po sobě vypsanými znaménky operací (např. +−, nebo ∗−) a jsou tudíž považována za syntaktickou chybu. Z toho současně plyne, že není možné vzájemně násobit nebo dělit dvě záporná čísla. Nicméně lze využít funkci paměti, uložit do ní záporné číslo a následně s takto zapsanou hodnotou provést matematickou operaci.

Při detekci stisku klávesy s jednou z matematických operací se do proměnné *op* uloží hodnota odpovídající dané operaci. Proměnná *op* je znovu využitá ve chvíli, kdy je známé druhé číslo, se kterým má být operace provedena, tzn. v případě volby další operace nebo stisknutím klávesy pro vyhodnocení.

3.4.1 Omezení a ošetření

Jedno z nejpodstatnějších omezení představuje zákaz dělení nulou. V případě, že se uživatel rozhodne tuto operaci realizovat, na displej se vypíše chybový výstup. Další omezení se týkají desetinných čísel. Vzhledem k tomu, že desetinná čísla jsou implementována jen na jedno desetinné místo, kalkulátor uživateli nepovolí vypsat čísla na další desetinná místa. Obdobným způsobem nelze nikdy zobrazit dvě desetinné tečky bezprostředně za sebou, aby zbytečně nedocházelo k neplatným vstupům. Chybovým výstupem končí veškeré neplatné vstupy, kterými jsou dělení a násobení s desetinnými čísly, syntaktické chyby, nebo složitější operace, které zahrnují uplatnění matematických předností. Všechna omezení jsou v kódu implementována prostřednictvím podmínkové struktury if–else.

3.5 Převody soustav

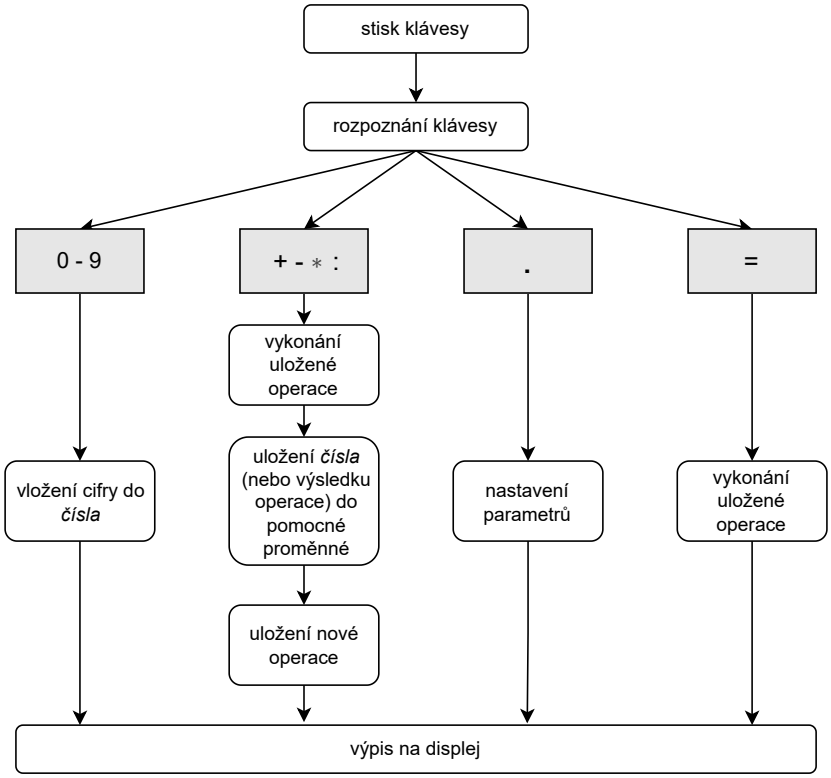
Prostřednictvím jednoho z přepínačů na přípravku DE10–Lite je možné přejít do módu převodu soustav. Převod je realizován pouze z desítkové do dvojkové a šestnáctkové soustavy. Toto omezení bylo vynuceno z důvodu kladení příliš vysokých nároků na kompilátor, což způsobovalo nepřiměřeně dlouhou dobu kompilace. Ukázalo se, že hlavním zdrojem tohoto problému je použití smyčky for v návrhu, více kapitola 3.8. Protože prostřednictvím smyčky je řízen již výpis výsledku na displej, nebylo udržitelné implementovat další smyčky pro převody mezi soustavami.

Režim převodu soustav signalizuje rozsvícená LED dioda bezprostředně nad příslušným přepínačem. Po stisknutí klávesy pro vyhodnocení se v případě platného vstupu na třetím řádku displeje zobrazí zadané číslo v hexadecimální soustavě a na čtvrtém řádku v binární soustavě v maximální délce šestnácti bitů. Viz Obrázek 3.8.

V módu převodu soustav není dovoleno používat jakékoliv operace, desetinná či záporná čísla. Pokud by uživatel chtěl převést do jiné soustavy výsledek operace, může využít funkci paměti. Při zadání neplatného vstupu se na posledním řádku displeje zobrazí chybový výstup.

3.6 Vyhodnocení

V komponentě k obsluze klávesnice současně dochází i k vyhodnocení výstupu. Princip vyhodnocování je systematicky zobrazen na Obrázku 3.5. Detekováním stisku klávesy se do proměnné u uloží odpovídající číslo z množiny pole *digits*, které hraje významnou roli při analýze stisknuté klávesy a výpisu na LCD displej. Dle typu vstupního znaku se postup řešení rozpadá na čtyři části.



Obrázek 3.5: Princip vyhodnocování [vlastní obrázek]

Nejjednodušším případem je zpracování stisku čísel 0 až 9. Zadávaná čísla se po cifrách ukládají do proměnné *num* tak, že s každou další cifrou se *num* vynásobí deseti a navíc se přičte hodnota zobrazená na právě stisknuté klávese. Následně je cifra vypsána na příslušnou pozici na displeji.

V případě použití klávesy s desetinnou tečkou se pouze nastaví parametry pro práci s desetinnými čísly, konkrétně se přiřadí logická 1 proměnným *dec* a *dec_cur*. První jmenovaná proměnná indikuje použití desetinného čísla kdykoliv v průběhu aktuálního vstupu. Pokud je tedy $dec = 1$, znamená to, že alespoň jedno ze zadaných čísel na vstupu je desetinné. Oproti tomu proměnná *dec_cur* značí, jestli je právě vypisované číslo desetinné ($dec\_cur = 1$) či nikoliv ($dec\_cur = 0$). V případě, že nastane například situace $dec = 1$ a zároveň $dec\_cur = 0$, znamená to, že někde na vstupu již bylo zadáno desetinné číslo, ale v místě kurzoru se nachází celé číslo. Systém na takovou situaci musí reagovat a patřičně upravit celá čísla na vstupu, aby výsledek operace odpovídal skutečnosti. Po stisku tlačítka s funkcí reset jsou obě

proměnné opět vynulovány.

Další variantou vstupního znaku je jedna z implementovaných matematických operací. Pokud se jedná o první znaménko operace na současném vstupu ($op = 0$), uloží se obsah proměnné *num* do dočasné proměnné *temp* a následně se *num* přiřadí nula. V dalším kroku se do proměnné *op* zapíše hodnota odpovídající příslušné matematické operaci.

V případě, že proměnná *op* není nulová, se nejprve provede předchozí matematická operace a teprve poté se do ní uloží nová hodnota. Následuje vynulování proměnné *num*. Viz Obrázek 3.6.

```
-- proměnná op ukládá operaci, která
-- bude provedena s dalším číslem
case op is
   when 1 => temp:=temp+num; -- +
   when 2 => temp:=temp-num; -- -
   when 3 => temp:=temp*num; -- *
   when 4 =>                 -- :
      temp:=temp/num;
      if num=0 then err_ind:='1';
      end if;
   when others => temp:=num;

end case;
```

Obrázek 3.6: Kód vyhodnocení operace [vlastní obrázek]

Nejspecifičtějším znakem, který se na vstupu může objevit, je jednoznačně symbol „rovná se", protože se kromě znaku příslušícímu stisknuté klávese vypisuje také celkový výstup. Pokud je prostřednictvím přepínače zvolen režim počítání, dojde po stisknutí této klávesy k vyhodnocení uložené operace obdobně jako v předchozím případě. V módu převodu soustav se proměnná *num* zkopíruje do proměnné *numb* a případně je část uložena také do *numc*, více viz kapitola 3.8. Po vyhodnocení se výsledek vypíše na displej, přičemž se pro přehlednost výstup zobrazuje na jiný řádek než vstup.

3.7 Reset a funkce paměti

Jedno z tlačítek (KEY0) je implementováno jako funkce reset. To znamená, že po jeho stisknutí se vymaže displej, kurzor se nastaví na začátek prvního řádku a proměnným se přiřadí výchozí hodnota (nejčastěji 0). Tlačítko s funkcí reset je vhodné použít před každým zadáním nového vstupu.

Druhé tlačítko (KEY1) vyvolá výsledek poslední provedené operace, funguje jako funkce paměti. Na displeji se hodnota paměti zobrazí jako „X". S takto získaným číslem lze dále provádět operace jako se standardně zadaným číslem prostřednictvím klávesnice a to včetně převodů do číselných soustav. Do paměti je možné uložit také záporná i desetinná čísla.

V případě, že předchozí operací byl převod do číselných soustav, funkce paměti samozřejmě výsledek neukládá. Hodnota paměti se aktualizuje na 0, tak jako tomu bylo před první zadanou operací.

3.8 Výstup

Původně byl výstup realizován na sedmisegmentovém displeji, ale z důvodu značně omezených možností se v průběhu práce přešlo na znakový LCD displej 20×4, který nabízí nesrovnatelně více alternativ pro zobrazení vstupů i výstupů.

Na displeji je pro přehlednost důležité rozlišení zadání (tj. vstupu uživatele) a výsledku operace. Vstup se vždy vypisuje na první řádek displeje a aby nedošlo k jeho přetečení, je použita podmínka pro minimální hodnotu kurzoru (viz if–else podmínka na Obrázku 3.3). V případě dosažení nejvzdálenější povolené pozice na displeji již nelze vypsat žádné další znaky kromě „=". Pro výstup slouží třetí a čtvrtý řádek LCD displeje.

Pro vypsání konkrétního znaku na LCD displej je nutné znát odpovídající znakový kód. Přehled znaků a příslušných kódu je uveden na Obrázku 2.9. Pro zjednodušení převodu cifry, písmena, nebo symbolu na znakový kód jsou všechny používané znakové kódy uspořádané v poli *digits* tak, aby pozicím 0 – 9 a 10 – 15 odpovídaly po řadě kódy pro čísla 0 až 9 a písmena A až F. Vyšší pozice v poli reprezentují znakové kódy pro symboly matematických operací, desetinnou tečku, rovná se a „X", viz Obrázek 3.7.

```
--každá pozice v poli reprezentuje konkrétní znak zapsaný v hexadecimálním
-- tvaru odpovídajícím znakovému kódu pro výpis na LCD
type digits_t is array(0 to 22) of std_logic_vector(7 downto 0);
constant digits : digits_t :=
    (0  => x"30", 1  => x"31", 2  => x"32", 3  => x"33", 4  => x"34",
     5  => x"35", 6  => x"36", 7  => x"37", 8  => x"38", 9  => x"39",
     10 => x"41", 11 => x"42", 12 => x"43", 13 => x"44", 14 => x"45",
     15 => x"46", 16 => x"2b", 17 => x"2d", 18 => x"2a", 19 => x"3a",
     20 => x"3d", 21 => x"2e", 22 => x"58");
```

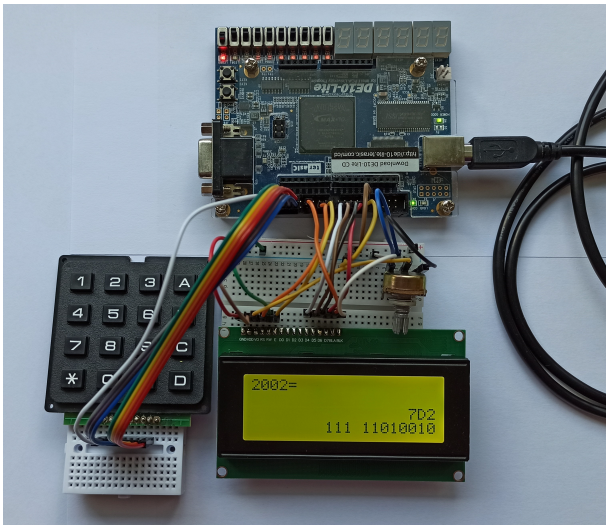
Obrázek 3.7: Pole znakových kódů [vlastní obrázek]

Výsledek je na výstup odesílán po jednotlivých cifrách prostřednictvím smyčky for. Později vyšlo najevo, že smyčka nadměrně zatěžuje kompilátor a značně navyšuje dobu kompilace. Z toho důvodu se upustilo od převodů z dvojkové a šestnáctkové soustavy a výstup se omezil pouze na 8 cifer. Jedinou výjimkou je převod do dvojkové soustavy, kde by maximum osmi bitů velmi limitovalo možnosti této funkce. V případě, že je převáděné číslo větší než 255, rozdělí se do dvou proměnných (num a $numc = num/256$). Obě proměnné se ve smyčce for vyhodnocují současně, pouze výstup se vypisuje na jiné pozice. Pokud výsledek nedosahuje osmi míst, smyčka končí ve chvíli, kdy jsou vypsány všechny nenulové cifry.

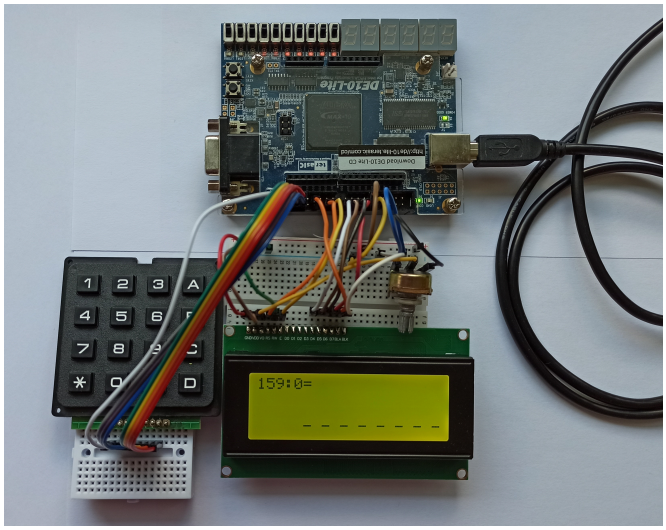
Po doběhnutí for smyčky se na výstup případně dopíše záporné znaménko, nebo se pro desetinné výsledky menší než jedna doplní nula. Kromě standardního výstupu je implementován také již zmiňovaný chybový výstup, který se zobrazí na posledním řádku LCD displeje v případě neplatného vstupu. Chybový výstup se z důvodu omezení na 8 cifer objeví i v případě příliš dlouhého výsledku. Ukázka chybového výstupu pro případ dělení nulou je na Obrázku 3.9.

Ve výsledném návrhu jsou využité tři LED diody přípravku DE10–Lite

k indikaci stisknutých tlačítek či přepnutí módu přepínačem (mezi počítáním a převody soustav). Na Obrázcích 3.8 a 3.9 je mimo jiné vidět rozdílná poloha krajního přepínače a různé stavy LED diody bezprostředně pod ním. Pro mód převodu soustav je přepínač situován v horní poloze a signalizační LED dioda je rozsvícená.



Obrázek 3.8: Příklad výstupu v módu převodu do soustav [vlastní obrázek]



Obrázek 3.9: Ukázka chybového výstupu [vlastní obrázek]

Závěr

V rámci bakalářské práce byl v jazyce VHDL navržen kalkulátor na přípravku DE10–Lite. Pro komunikaci s uživatelem a zobrazování výstupu byla k přípravku připojena maticová klávesnice 4×4 a znakový LCD displej 20×4. Dále se k obsluze kalkulátoru využila tlačítka, přepínač a LED diody na vývojové desce DE10–Lite.

Kalkulátor má implementované matematické operace sčítání, odčítání, násobení a dělení s uvedenými omezeními. Pro operace sčítání a odčítání lze zadávat desetinná čísla na jedno desetinné místo a obdobně je na jedno desetinné místo useknutý také výstup neceločíselného dělení. Přepínač na přípravku DE10–Lite zprostředkovává volbu mezi režimem počítání a převodu soustav. Z důvodu náročnosti vyhodnocování na výstupu implementované smyčky for je převod soustav realizován jen z desítkové do šestnáctkové a dvojkové soustavy. Ze stejného důvodu je výsledek v módu počítání zobrazen na maximálně osm cifer.

Tlačítka na přípravku DE10–Lite reprezentují funkci paměti a reset. S hodnotou uloženou v paměti lze nadále pracovat jako s jakýmkoliv jiným zadaným číslem. LED diody indikují stisknutí tlačítek či přepnutí režimu kalkulátoru. Kromě standardního výstupu a pro převody soustav je implementován také chybový výstup pro neplatné vstupy.

Do budoucna by kalkulátor mohl být rozšířen například o další funkce a matematické operace. Rovněž by bylo možné zabývat se jinými možnostmi vyhodnocování a vypisování dat na displej, aby výstup nemusel být omezen.

Literatura

[1] LAFATA, Pavel, Petr HAMPL a Michal PRAVDA. *Digitální technika*. ČVUT, 2011. ISBN 978-80-01-04914-3.

[2] NDJOUNTCHE, Tertulien. *Digital Electronics 2 : Sequential and Arithmetic Logic Circuits* [online]. Wiley-Blackwell, 2016 [cit. 2024-05-05]. ISBN 9781119329763. Dostupné z: https://ebookcentral.proquest.com/lib/cvut/reader.action?docID=4648716&ppg=277&pq-origsite=summon

[3] *Intel MAX 10 FPGA Device Architecture* [online]. 2022 [cit. 2024-01-09]. Dostupné také z: https://www.intel.com/content/www/us/en/docs/programmable/683105/current/fpga-device-architecture.html

[4] *DE10–Lite User Manual* [online]. 2020 [cit. 2024-01-09]. Dostupné také z: https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=China&No=1021&FID=a13a2782811152b477e60203d34b1baa

[5] *Intel MAX 10 FPGA Device Overview* [online]. 2021 [cit. 2024-01-09]. Dostupné také z: https://cdrdv2-public.intel.com/666776/m10_overview-683658-666776.pdf

[6] *Realizace a obsluha klávesnice* [online]. [cit. 2024-01-28]. Dostupné z: https://eluc.ikap.cz/verejne/lekce/924

[7] *Hitachi HD44780U (LCD-II), Dot Matrix Liquid Crystal Display Controller/Driver* [online]. 1998 [cit. 2024-05-01]. Dostupné z: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[8] *LCD 2004 Module.* Sunfounder [online]. 2018 [cit. 2024-05-12]. Dostupné z: http://wiki.sunfounder.cc/index.php?title=LCD2004_Module

[9] WEIMAN, Donald. *LCD Addressing* [online]. Alfred State College, 2012 [cit. 2024-05-01]. Dostupné z: https://web.alfredstate.edu/faculty/weimandn/lcd/lcd_addressing/lcd_addressing_index.html

[10] ASHENDEN, Peter J. *The VHDL Cookbook* [online]. University of Adelaide, South Australia, 1990 [cit. 2024-05-14]. Dostupné z: https://tams-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf

[11] LAFATA, Pavel. *Úvod do jazyka VHDL I.* Portál inovace vyššího odborného vzdělávání [online]. 2020 [cit. 2024-05-14]. Dostupné z: https://www.vovcr.cz/odz/tech/561/page00.html

[12] LAFATA, Pavel. Úvod do jazyka VHDL II. Portál inovace vyššího odborného vzdělávání [online]. 2020 [cit. 2024-05-14]. Dostupné z: https://www.vovcr.cz/odz/tech/562/page00.html

[13] LAFATA, Pavel. *Úvod do jazyka VHDL III.* Portál inovace vyššího odborného vzdělávání [online]. 2020 [cit. 2024-05-14]. Dostupné z: https://www.vovcr.cz/odz/tech/566/page00.html

[14] HAY, Paul. *LCD Voltage Inputs for LCD Displays Explained.* Focus LCD's [online]. ©2024 [cit. 2024-05-14]. Dostupné z: https://focuslcds.com/journals/lcd-voltage-inputs-for-lcd-displays-explained/

[15] WEIMAN, Donald. *LCD Initialization* [online]. Alfred State College, 2012 [cit. 2024-05-01]. Dostupné z: https://web.alfredstate.edu/faculty/weimandn/lcd/lcd_initialization/lcd_initialization_index.html