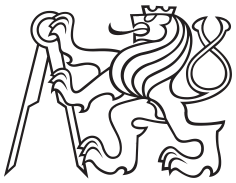
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Regulátor PWM ve VHDL na přípravku Spartan3E

Tomáš Bánok

Školitel: Ing. Pavel Lafata, Ph.D.
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bánok** Jméno: **Tomáš** Osobní číslo: **491843**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Regulátor PWM ve VHDL na přípravku Spartan3E

Název bakalářské práce anglicky:

Digital PWM Control in VHDL Based on Spartan3E Kit

Pokyny pro vypracování:

Seznamte se s přípravkem Xilinx Spartan3E a jeho obsluhou pomocí jazyka VHDL. Navrhněte a realizujte jednoduchý regulátor napájení na principu PWM, otestujte jeho funkčnost nejprve přímo na přípravku Spartan3E (otočné tlačítko, indikační panel LED diod). Využijte navržený PWM regulátor a realizujte regulaci otáček připojeného ventilátoru jednoduchým ručním ovládáním pomocí otočného přepínače. K přípravku rovněž připojte digitální teplotní čidlo (např. DS18B20) a vytvořte ve VHDL obslužný program, který z něho bude vyčítat teplotu. Rozšiřte předchozí program o možnost automatického řízení otáček ventilátoru na základě teploty změřené pomocí čidla. Využijte VGA výstup přípravku Spartan3E, přidejte rozšíření původního VHDL kódu a na připojeném VGA monitoru vypište či zobrazte aktuální nastavení PWM regulátoru.

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.

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: **21.01.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

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

Student bere na vědomí, že je povinen 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 studenta

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Pavel Lafata, Ph.D. za jeho odborné vedení, cenné rady a vypůjčení vývojového kitu Spartan-3E a potřebných periferií. Dále Bych rád poděkoval celé mé rodině, která mě během studia podporovala.

Prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

V Praze, 20. května 2022

.....

Abstrakt

Tato bakalářská práce se zabývá vytvořením VHDL kódu pro přípravek Spartan-3E Starter Board. Měření teploty je zajištěno digitálním senzorem DS18B20, který komunikuje pomocí 1–Wire sběrnice. Získaná teplota je použita pro PWM regulaci, jejíž parametry lze změnit využitím otočného tlačítka a LCD monitoru připojeného přes VGA rozhraní. Displej dále zobrazuje graf historii teplot připojených digitálních senzorů.

Klíčová slova: FPGA, VHDL, Spartan-3E Starter Board, 1-Wire protokol, DS18B20 senzor, PWM řízení, VGA, bitmapové písmo

Školitel: Ing. Pavel Lafata, Ph.D.

Abstract

This bachelor thesis deals with the creation of VHDL code for the Spartan-3E Starter Board. Temperature measurement is provided by a DS18B20 digital sensor that communicates via a 1–Wire bus. The obtained temperature is used for PWM control, the parameters of which can be changed by using a rotary switch and LCD monitor connected via a VGA interface. The display also shows a graph of the temperature history of the connected digital sensors.

Keywords: FPGA, VHDL, Spartan-3E Starter Board, 1-Wire protocol, DS18B20 sensor, PWM control, VGA, bitmap font

Title translation: Digital PWM Control in VHDL Based on Spartan3E Kit

Obsah

1 Úvod **1**

2 Teoretická část **3**

2.1 Programovatelné logické obvody . 3

2.1.1 Vývojový kit Xilinx Spartan-3E 4

2.2 VGA 6

2.3 1-Wire protokol 9

2.3.1 Komunikace se slave zařízením 9

2.4 Uživatelské vstupy 11

2.4.1 Debouncing kontaktů 11

2.4.2 Otočné tlačítko 11

2.5 Řízení ventilátorů 13

2.5.1 Druhy řízení 13

2.5.2 Specifikace PWM řízení PC ventilátorů 13

3 Praktická část **15**

3.1 Přehled VHDL modulů 15

3.2 Vytvořená VHDL knihovna 17

3.3 Generování taktovacích signálů . 18

3.3.1 Rozhraní 18

3.3.2 Využití DCM 18

3.3.3 Nastavení frekvence výstupních signálů 19

3.4 Řízení VGA výstupu 20

3.4.1 Rozhraní 20

3.4.2 Časování 21

3.4.3 Písmo a vykreslování více znaků 22

3.5 Výběr znaku pro VGA řadič . . . 25

3.5.1 Rozhraní 25

3.6 Čtení teploty senzoru 27

3.6.1 Rozhraní 27

3.6.2 Časování a čtení teploty 27

3.6.3 Chybové výstupy 28

3.7 Stavový kód 1–Wire rozhraní . . . 29

3.8 Obsluha otočného tlačítka 30

3.8.1 Rozhraní 30

3.8.2 Debouncing 30

3.9 Řízení ventilátorů 32

3.9.1 Rozhraní 32

3.9.2 PWM signál 32

3.9.3 Změna nastavení 33

3.9.4 Struktura 34

3.10 Znakový teplotní graf 37

3.10.1 Rozhraní 37

3.10.2 Využití blokové RAM 37

3.10.3 Vkládání teplot do RAM . . . 38

3.10.4 Čtení teplot z RAM, výběr znaku 40

4 Závěr **43**

Literatura **45**

Příloha A

Seznam zkratek **47**

Příloha B

Obsah elektronické přílohy **49**

Obrázky

2.1 Fotka vývojového kitu Xilinx Spartan-3E. 4
2.2 Časování CRT monitoru, převrato z [3]. 7
2.3 VGA konektor na vývojové desce Spartan-3E Starter Kit s připojením k FPGA, převrato z [3]. 8
2.4 Možnosti napájení slave zařízení připojeného k 1–Wire sběrnici. Vlevo přes pull-up rezistor a doplňující pull-up tranzistor, vpravo připojením senzoru přímo ke zdroji napětí, převzato z [7]. 9
2.5 Resetovací puls na 1–Wire sběrnici, převzato z [7]. 9
2.6 Průběhy zápisu log. 0/1 a čtení log. 0/1 na 1–Wire sběrnici, převzato z [7]. 10
2.7 Schéma připojení tlačítka s vyobrazeným průběhem zákmitů napětí při jeho stisknutí. 11
2.8 Ilustrace fázově posunutých výstupů A, B. Převzato z [1]. 12
2.9 Výstup otočného tlačítka při neustálém otáčení, včetně vyobrazení zákmitů kontaktů. Převzato z [3]. . 12
2.10 Obrázek čtyřpinového konektoru pro připojení počítačového ventilátoru. Převzato z [6]. 14

3.1 Popis funkce hlavních VHDL modulů. 15
3.2 Přehled vytvořených VHDL modulů v projektu a jejich hierarchie. 16
3.3 Rozhraní modulu CLK_gen. . . . 18
3.4 Příklad využitého procesu pro děličku kmitočtu na 400 kHz. 19
3.5 Rozhraní modulu VGA_driver. . 20
3.6 Blokové schéma VGA řadiče. Výběr pixelu k odeslání displeji je popsán na obr. 3.9. 21
3.7 Vnitřní časování modulu VGA řadiče. 22
3.8 Příklady bitmapy znaků „0“, „A“ a „e“. Barvy invertovány pro lepší čitelnost. 22
3.9 Stavový diagram popisující výběr barev pixelu k odeslání displeji. . . . 23
3.10 Příklad prioritizace znaků a jejich příslušné bitmapy. Pozicí se myslí index ve znakovém vstupu modulu. 23
3.11 Fotka vykreslovaného snímku. Vlevo se nachází konfigurace PWM kanálů, vpravo nahoře znakový teplotní graf zobrazující historii teplot a v pravém dolním rohu stavový kód indikující stav 1–Wire modulů. 24
3.12 Rozhraní modulu Char_selector. 25
3.13 Blokové schéma interakce mezi modulem Char_selector s obsluhovanými moduly. 26
3.14 Rozhraní modulu OneWire_DS18B20. 27
3.15 Formát výstupní teploty s vyznačenou číselnou váhou a indexem bitů. „S“ značí pozici znaménkového bitu, kde „0“ značí kladné číslo a „1“ číslo záporné. . . 28
3.16 Stavový diagram řídícího procesu 1–Wire modulu. 28
3.17 Rozhraní modulu Error_codes. 29
3.18 Rozhraní modulu Debounce_ROT. 30
3.19 Časový průběh výstupů otočného tlačítka při jedné otáčce. 31
3.20 Principiální schéma pro debouncing vstupů otočného tlačítka. 31
3.21 Rozhraní modulu PWM_controller_top. 32
3.22 Nastavení střídy PWM signálu v automatickém režimu. Vyobrazeno nastavení: minimální střída 20 %, PWM konstanta 1,00 %/°C a maximální teplota 90 °C. Červená křivka kopíruje výslednou závislost střídy na teplotě. 33

3.23 Příklad nastavení dvou PWM kanálů. Kurzor se nachází v nastavení střídy prvního kanálu, přičemž otáčením otočného tlačítka se mění hodnota výstupní střídy. . . 34
3.24 Příklad nastavení dvou PWM kanálů. Kvůli přesáhnutí maximální teploty senzoru 2 je výstupní střída druhého kanálu 100%. 35
3.25 Vnitřní znaková paměť PWM modulu, která je adresována selektorem s příkladem zobrazovaných znaků. 35
3.26 Popis znaků využitých pro kurzor. 36
3.27 Rozhraní modulu Temp_graph. 37
3.28 Využití adresního prostoru RAM paměti pro skladování historie teplot. 38
3.29 Přepsání spodní pozice obsahu RAM paměti. Obrázek popisuje jen přepsání teploty z prvního senzoru, ale stejným způsobem se přepisuje i teplota z druhého senzoru, která se nachází na vyšších pozicích bitů v RAM (viditelné na obr. 3.28). . . 39
3.30 Přepsání vyšší pozice obsahu RAM paměti. 39
3.31 Posun vstupů/výstupů přepisovacího bufferu na další adresu RAM. 39
3.32 Opakování předešlých kroků zápisu do RAM paměti, dokud není obsah celé paměti posunut. 40
3.33 Vyobrazení vztahu mezi uloženou sekvencí bitů teploty a výškou vykreslené trojice pixelů s jejich odpovídající teplotou. Vrchní 4 bity jsou využity k rozhodnutí, jestli bude teplota vykreslována do požadovaného znaku. Spodní 3 bity určují pozici vykreslované trojice pixelů jednoho znaku. V levé části znaku je vykreslována teplota na nižší pozici přečtených dat z RAM (novější teplota), v pravé části teplota na vyšší pozici (starší teplota). . . . 41
3.34 Vykreslený průběh teplot dvou senzorů. 41

Tabulky

Kapitola 1
Úvod

Pro implementaci složitějších digitálních obvodů lze v dnešní době použít tři hlavní metody – využití mikrokontroleru, FPGA či ASIC. Výhodou mikrokontrolerů je jejich relativně nízká cena, ale pro rychlé či paralelní úkony mohou být svým výpočetním výkonem nedostačující. Nevýhodou využití ASIC je jejich velmi vysoká cena pro návrh a po výrobě ho již nelze upravovat. Návrh s využitím FPGA je sice dražší než využití mikrokontroleru, ale pro malovýrobu je výhodnější než ASIC a zároveň má možnost úpravy návrhu i po nahrání do FPGA.

Cílem práce bylo vytvořit VHDL kód pro PWM regulaci otáček ventilátoru, nejdříve s manuálním nastavením střídy a poté s automatickým řízení na základě teploty změřené připojeným digitálním teploměrem. Dalším úkolem bylo rozšířit projekt pro využití VGA výstupu přípravku Spartan-3E pro zobrazování aktuálního nastavení PWM regulátoru.

Práce je rozdělena do dvou kapitol. V teoretické části popisuji základní principy nutné pro správnou funkci vytvořených VHDL modulů. V praktické části se poté zabývám popisem funkce a rozhraní těchto modulů, doplněné o blokova schémata a vývojové diagramy pro vysvětlení složitějších funkcí. Sekce popisující moduly, které obsahují znakový výstup, jsou doplněny o fotografie celého displeje nebo jeho relevantní části.

Kapitola 2

Teoretická část

2.1 Programovatelné logické obvody

Programovatelné logické obvody umožňují realizovat požadovanou logickou funkci využitím jednoho integrovaného obvodu, který plní tuto funkci namísto využití diskrétních logických hradel či jejich kombinace v jednom pouzdře. Tyto obvody se dělí na:

- ROM (Read Only Memory)

  Využití nevolatilní paměti pro definování logické funkce. Mezi nevýhody patří nižší rychlost, hazardy při přechodu mezi stavy a vyšší spotřeba.

- PLA (Programmable Logic Array)

  Obsahuje programovatelné matice AND hradel a OR hradel, jejichž propojením je realizována požadovaná logická funkce. Programování je však nevratné a probíhá již při výrobě vypálením příslušných cest a propojů.

- PAL (Programmable Array Logic)

  Podobně jako PLA obsahuje programovatelnou matici AND hradel, které ale vedou do pevné matice OR hradel a invertorů.

- GAL (Generic Array Logic)

  Vylepšení PAL, které je možné přeprogramovat.

- CPLD (Complex Programmable Logic Devices)

  Ekvivalent více PAL či GAL v jednom pouzdře umožňující realizaci složitějších logických funkcí.

- FPGA (Field Programmable Gate Array)

  V současnosti nejpokročilejší typ hradlových polí. Jsou založena na principu look up tabulek, které generují výstupní hodnoty na základě

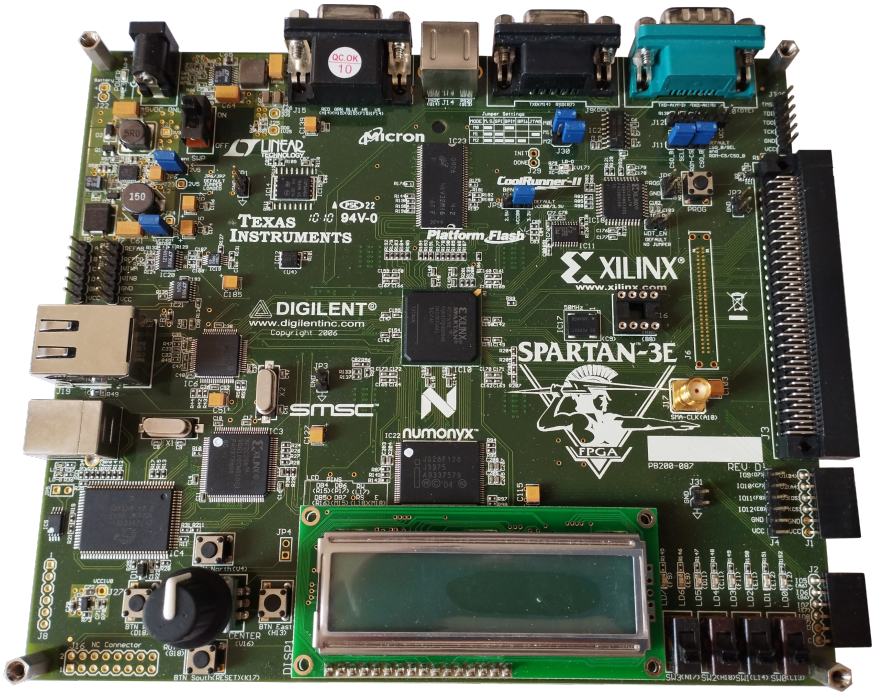
svých vstupů. Navíc obsahují řadu specializovaných obvodů, jako jsou komparátory, registry, A/D převodníky, RAM paměti a další. Počet a druhy zahrnutých specializovaných obvodů se liší mezi výrobci. Konfigurace FPGA je ukládána buď interně, nebo pomocí externí paměti.

Pro implementování logických funkcí do FPGA se využívá dvou jazyků – – VHDL a Verilog. V této práci využívám VHDL, neboť je jednodušší pro začátečníky. VHDL dále umožňuje snadno vytvořit modulární design a umožňuje popis ve třech vrstvách abstrakce – Behavioral (popis algoritmy), RTL (přenos dat mezi registry) a Gate Level (popis pomocí logických hradel). V práci využívám popis pomocí algoritmů.

2.1.1 Vývojový kit Xilinx Spartan-3E

Vývojový kit Xilinx Spartan-3E (obr. 2.1) umožňuje snadný vývoj aplikací využívající FPGA. Pro připojení k počítači je využito USB konektoru, pomocí kterého je možné nahrát konfiguraci FPGA a obsah dvou nevolatilních pamětí umístěných na kitu. Vývojový kit dále obsahuje velké množství periferií, které mohou být obsluhovány (např. LCD displej, DDR SDRAM, RS-232 sériové porty a další).

V této práci využívám VGA konektoru pro přenos obrazu na displej, pin header pro připojení digitálních senzorů teploty a výstup PWM signálů pro řízení ventilátorů, přepínačů pro resetování některých VHDL modulů a diskrétní LED pro zobrazování důležitých signálů pro debugging.



Obrázek 2.1: Fotka vývojového kitu Xilinx Spartan-3E.

Architektura Spartan-3E FPGA

Využitý FPGA čip z rodiny Spartan-3E obsahuje tyto bloky [2], s pomocí kterých se poté syntetizují a implementují logické funkce:

- IOB - Input/Output Blocks (Vstupně/Výstupní bloky)

  Slouží pro propojení interní logiky FPGA s vnějšími periferiemi. Každý IOB umožňuje obousměrný tok dat, třístavový provoz a diferenciální provoz. IOB dále umožňují nastavení pull-up / pull-down a napěťových úrovní vstupů/výstupů.

- CLB - Configurable Logic Blocks (Konfigurovatelné logické bloky)

  Hlavní část FPGA obsahující LUT, paměťové prvky a další podpůrné logické funkce. Slouží pro implementaci logických funkcí a pro skladování dat.

- BRAM - Block RAM (Bloková RAM)

  Synchronní paměť, která umožňuje skladovat velké množství dat v 18Kbitových dual-port blocích. Dual-port umožňuje přístup k jednomu paměťovému prostoru pomocí dvou nezávislých rozhraní.

- Multiplier Blocks (Násobičky)

  Hardwarové násobičky, které umožňují mezi sebou vynásobit dva 18bitové vstupy.

- DCM - Digital Clock Manager

  Bloky pro distribuci, zpožďování, násobení, dělení a fázové posouvání taktovacích signálů.

- Switch Matrix - Programovatelná matice propojů

  Umožňuje propojení výše zmíněných bloků mezi sebou.

2.2 VGA

VGA rozhraní je svou signalizací utvořeno k řízení CRT monitorů, které vykreslují obraz pomocí paprsků elektronů vychylovaných magnetickým polem vychylovacích cívek. Na obr. 2.2 je vidět, že se tento paprsek pohybuje zleva doprava po jednotlivých řádcích. Řádky (snímky) se od sebe poté odlišují pomocí pulsů horizontální (vertikální) synchronizace. LCD monitory tento způsob řízení s jejich nástupem převzaly.

Intenzita každé barevné složky (R,G,B) se přenáší analogovým signálem o napětí 0 - 0,7 V po vyhrazených vodičích. Vzhledem k připojení VGA konektoru k výstupům FPGA (obr. 2.3) je možné přenášet jen 2 intenzity každé barevné složky – nulovou či maximální.

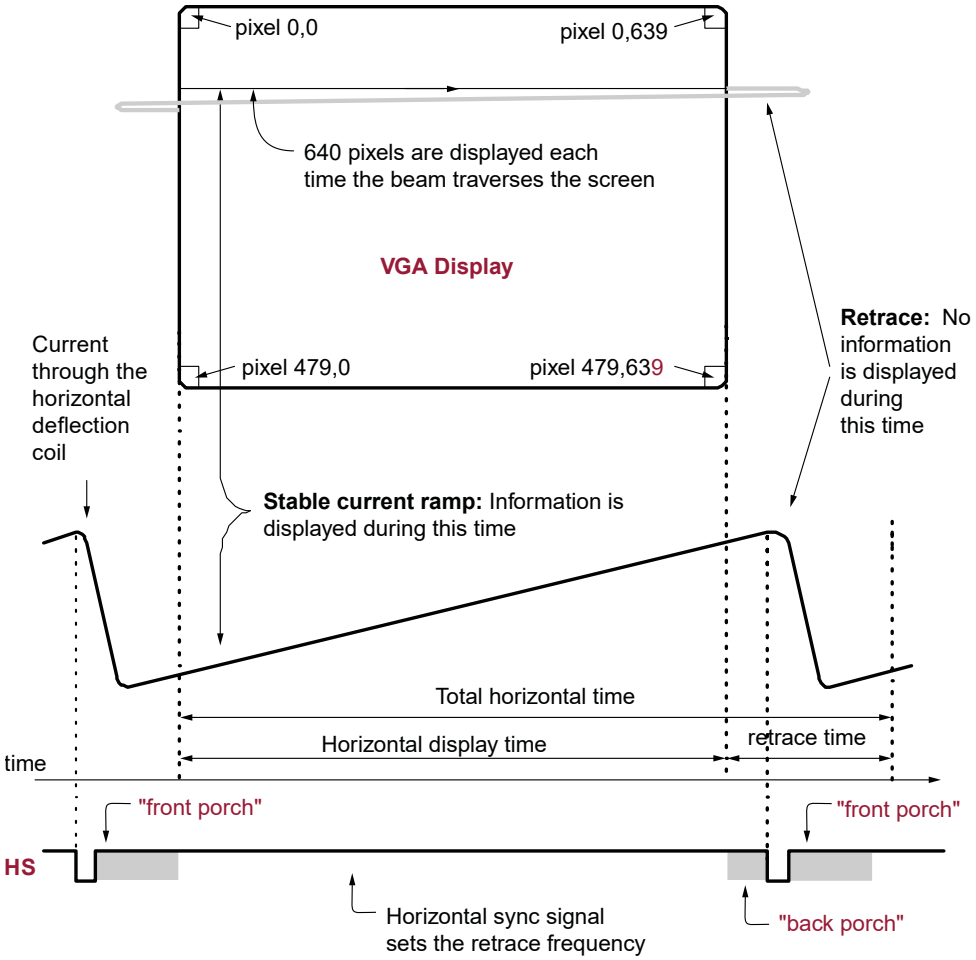
Vykreslování snímků lze rozdělit na dva druhy – vykreslování pixelů a vykreslování znaků.

Vykreslování pixelů

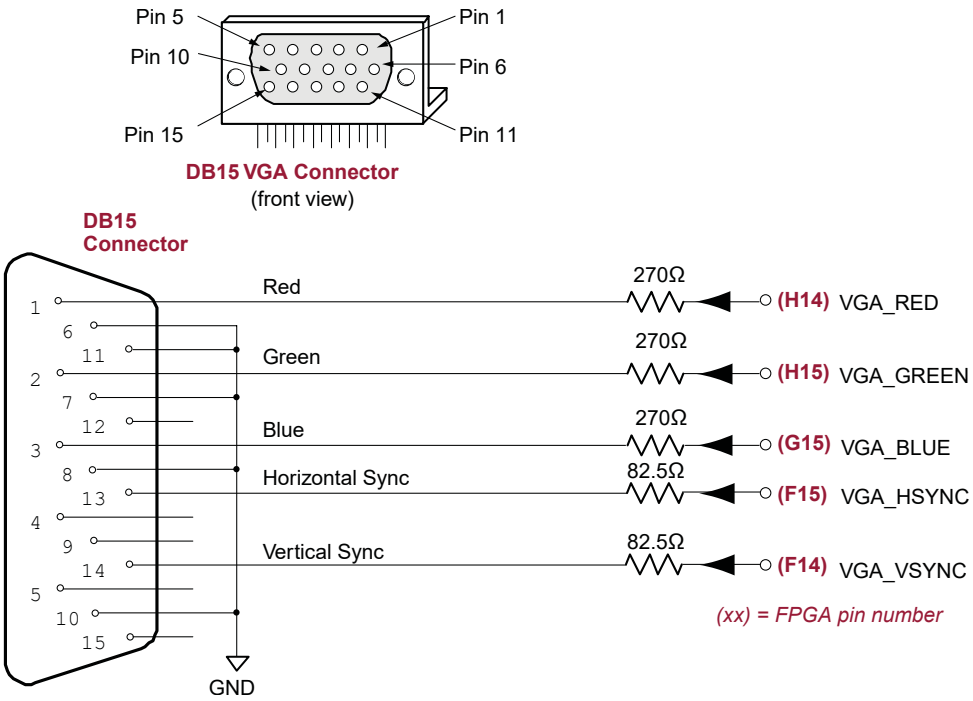
Grafický řadič ukládá informace o každém pixelu, které poté posílá do monitoru k vykreslení. Náročnější na prostředky a řízení, ale umožňuje vykreslit libovolný obraz.

Vykreslování znaků

Namísto definování každého pixelu nezávisle na druhém se vykreslovaný snímek rozdělí na pole znaků o fixní velikosti. Pixel k odeslání se poté vybírá grafickým řadičem z LUT, jejímž vstupem je kód znaku, který se má zobrazit. Hlavní nevýhodou je neschopnost vykreslit libovolný obraz a omezený počet znaků, které musí být uloženy v LUT. Výhodou je snažší řízení a ušetření prostředků, převážně paměti.



Obrázek 2.2: Časování CRT monitoru, převrato z [3].

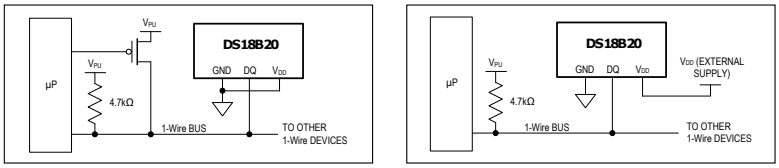


Obrázek 2.3: VGA konektor na vývojové desce Spartan-3E Starter Kit s připojením k FPGA, převrato z [3].

2.3 1-Wire protokol

1–Wire sběrnice umožňuje komunikaci mezi jedním master zařízením a jedním či více slave zařízeními po společném vodiči. 1–Wire protokol umožňuje komunikovat buď s jedním slave zařízením vybraným jeho 64bitovou adresou, se všemi slave zařízeními najednou či se všemi slave podporující jiný výběr[1]. Kolize na sběrnici jsou řešené použitím výstupů s otevřeným kolektorem a externím pull-up rezistorem.

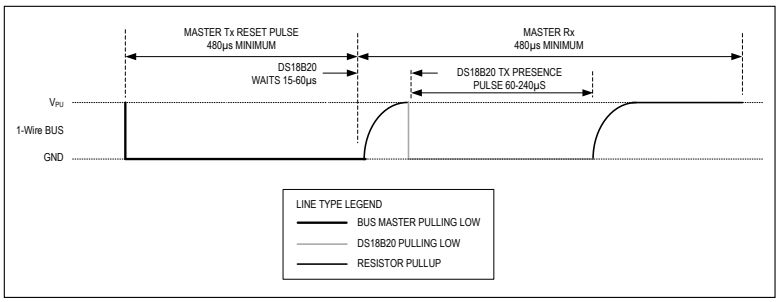
Napájení zařízení na sběrnici může být zařízeno buď přivedením vodiče pro napájení, nebo přes pull-up rezistor na sběrnici po které se komunikuje. Toto umožňuje přivedení pouze dvou vodičů ke koncovému zařízení, ale v některých případech je nutné pro operace vyžadující vyšší odběr připojit i pull-up tranzistor, viz obr. 2.4.



Obrázek 2.4: Možnosti napájení slave zařízení připojeného k 1–Wire sběrnici. Vlevo přes pull-up rezistor a doplňující pull-up tranzistor, vpravo připojením senzoru přímo ke zdroji napětí, převzato z [7].

2.3.1 Komunikace se slave zařízením

Komunikaci na sběrnici vždy zahajuje master vysláním RESET pulsu (obr. 2.5), na který všechna připojená slave zařízení reagují prezenčním pulsem (vyznačeno šedě). Přenos bitů poté probíhá od LSB v časových slotech o délce 60 až 120 μs s odstupem minimálně 1 μs mezi sloty. Přenos bitu vždy zahajuje master stažením sběrnice k 0 V a uvolněním v průběhu časového slotu, viz obr. 2.6.



Obrázek 2.5: Resetovací puls na 1–Wire sběrnici, převzato z [7].

[1] Např. se všemi slave zařízeními, u kterých nastala tzv. alarm událost, nebo se všemi slave zařízeními podporující rychlou komunikaci. Podpora záleží na připojeném slave zařízení.

Po RESET pulsu následuje ROM příkaz, který vybírá, s jakým slave zařízením bude pokračovat komunikace. Senzor DS18B20 podporuje:

- Search Rom (0xF0)

  Iterativní čtení ROM adres všech připojených zařízení.

- Read Rom (0x33)

  Čtení ROM adresy připojeného zařízení (nutno zajistit pouze 1 zařízení na sběrnici).

- Match Rom (0x55)

  Následováno unikátní ROM adresou vybíraného zařízení.

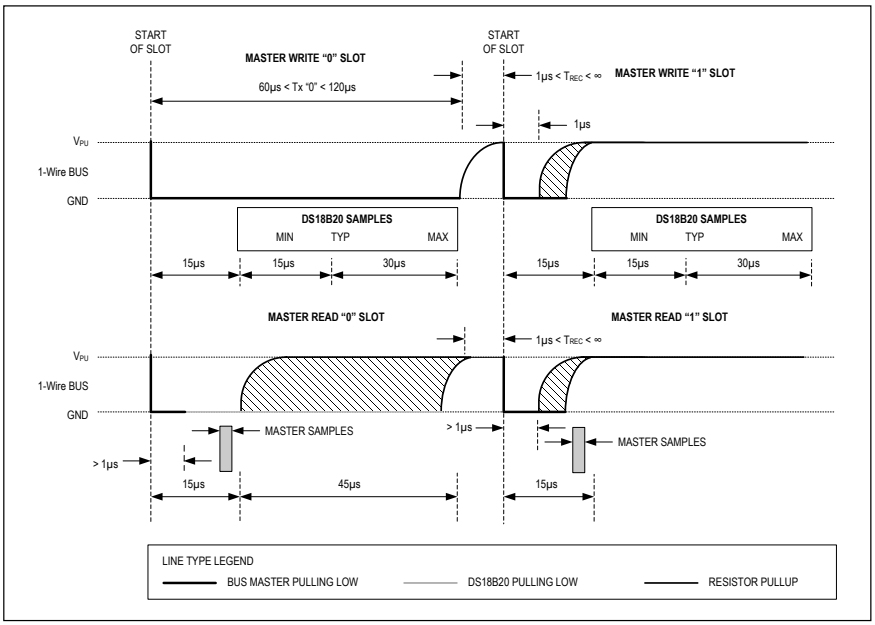
- Skip Rom (0xCC)

  Komunikace se všemi zařízeními.

- Alarm Search (0xEC)

  Komunikace se zařízeními, u kterých nastala alarm událost (překročení nastavené teploty).

Po výběru slave zařízení je možné přenášet data mezi master/slave, jako je např. čtení teploty senzoru či zápis do paměti zařízení.

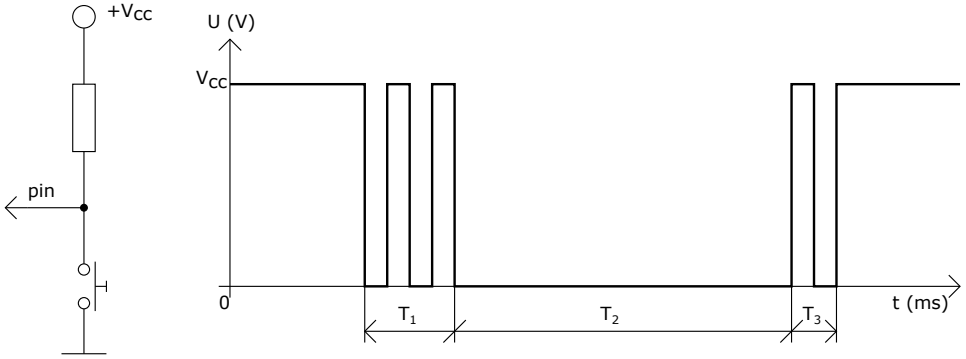


Obrázek 2.6: Průběhy zápisu log. 0/1 a čtení log. 0/1 na 1–Wire sběrnici, převzato z [7].

2.4 Uživatelské vstupy

2.4.1 Debouncing kontaktů

Ideální spínač (či jiný kontakt) má schopnost okamžitě spojit či rozpojit připojený obvod a v tomto stavu setrvat, dokud jej uživatel (či jiný faktor) nezmění. U reálných kontaktů však dochází během změny stavu k zákmitům, angl. *bouncing*. Toto je ilustrováno na obr. 2.7, kde na začátku doby T_1 uživatel stiskne tlačítko a na konci doby T_2 jej uvolní. Během T_1 dochází k mechanickým zákmitům kontaktů, při kterém tlačítko generuje další nežádoucí sestupné a vzestupné hrany a jeho opravdový stav lze s jistotou zjistit až během doby T_2. K zákmitům také dochází i po uvolnění tlačítka, vyznačené dobou T_3.

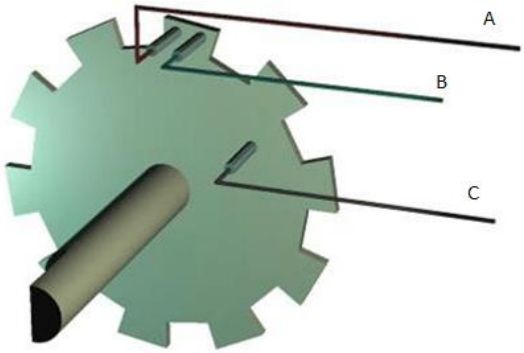


Obrázek 2.7: Schéma připojení tlačítka s vyobrazeným průběhem zákmitů napětí při jeho stisknutí.

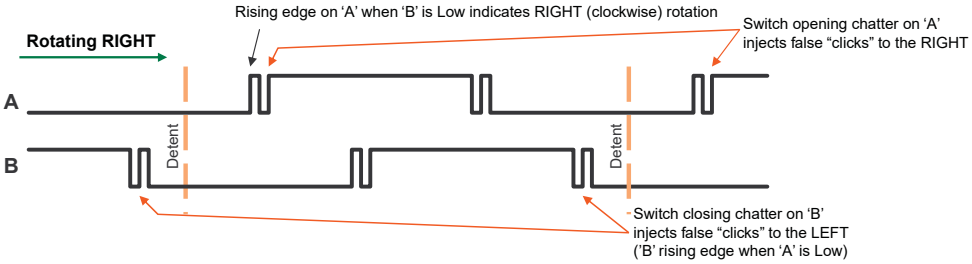
2.4.2 Otočné tlačítko

Otočné tlačítko, také známé jako rotační enkodér, umožňuje detekovat otáčení i jeho směr pomocí dvou fázově posunutých výstupů. Na obr. 2.8 je vidět princip těchto fázově posunutých výstupů, kdy je jeden spojen/rozpojen se společným kontaktem dříve než druhý. Výstupní signály při neustálém otáčení jsou poté vyobrazeny na obr. 2.9.

Nevýhodou využití otočného tlačítka je složitější debouncing jeho kontaktů. Při neúspěšném provedení může nastat nejen nedetekování otáčky, ale i detekce otáčení špatným směrem.



Obrázek 2.8: Ilustrace fázově posunutých výstupů A, B. Převzato z [1].



Obrázek 2.9: Výstup otočného tlačítka při neustálém otáčení, včetně vyobrazení zákmitů kontaktů. Převzato z [3].

2.5 Řízení ventilátorů

2.5.1 Druhy řízení

Využité řízení otáček ventilátoru lze zjednodušit na dvě metody:

- Řízení příkonu

  Nejjednodušší metoda na implementaci, kde střída PWM signálu je úměrná příkonu ventilátoru. Podle druhu ventilátoru je možné zanedbat vyhlazení napájecího napětí.

  Mezi nevýhody patří nelinearita řízení, minimální nutná střída pro běh a rozběh ventilátoru a podle druhu ventilátoru případné proudové špičky při sepnutí řídícího tranzistoru.

- Komunikace s řídící elektronikou

  Ventilátory využívané pro počítače, servery apod. využívají převážně BLDC motory, které vyžadují řídící elektroniku. Tato elektronika může být dle Intel specifikace vybavena možností regulace otáček PWM signálem, jehož střída určuje relativní hodnotu otáček [5].

  Hlavní nevýhodou je nutnost podpory této regulace ventilátorem. není-li ventilátor vybaven vodičem pro tento druh řízení, tak je nutné regulovat příkon.

2.5.2 Specifikace PWM řízení PC ventilátorů

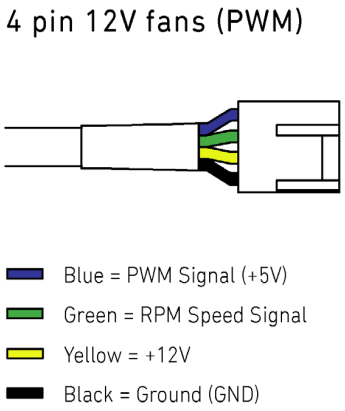
Počítačové ventilátory (a některá jiná zařízení) využívají pro své připojení nejčastěji tří či čtyřpinový konektor (obr. 2.10), přičemž třípinový neobsahuje vodič pro řídící PWM signál [5] [6].

PWM signál na vyhrazeném vodiči musí splňovat tyto požadavky [5]:

- Frekvenci 25 kHz, s tolerovaným rozsahem 21 - 28 kHz
- Maximální napětí 0,8 V pro log. 0
- Maximální napětí 5,25 V
- Absence pull-up rezistoru[2] a využití výstupu s otevřeným kolektorem
- Maximální odebíraný proud 5 mA

Součástí specifikace je také obsažení tzv. *sense*[3] vodiče, který přenáší informaci o počtu otáček připojeného ventilátoru. V této práci není využit, ale kromě zobrazení aktuálních otáček může také detekovat zastavení ventilátoru.

[2] Pull-up rezistor je zajištěn připojeným ventilátorem.
[3] Také i RPM sense, na obr. 2.10 RPM Speed Signal apod.



Obrázek 2.10: Obrázek čtyřpinového konektoru pro připojení počítačového ventilátoru. Převzato z [6].

Kapitola 3

Praktická část

3.1 Přehled VHDL modulů

Vytvořený projekt je rozdělen do VHDL modulů, které mohou pracovat samostatně (za předpokladu zajištění potřebných vstupů, jako je např. taktovací signál o správné frekvenci) a obsluhují jen jednu část projektu pro zajištění snadné záměny jiným modulem, který plní stejnou či podobnou funkci.

Dále jsem se rozhodl obsluhovat každý z digitálních senzorů teploty vlastní instancí 1–Wire modulu. Toto přináší jistou redundanci a větší využití prostředků FPGA, ale umožňuje snadné rozšíření pro obsluhu více čidel a jednoduchou úpravu v případě záměny senzoru za jiný, který nekomunikuje pomocí 1–Wire protokolu.

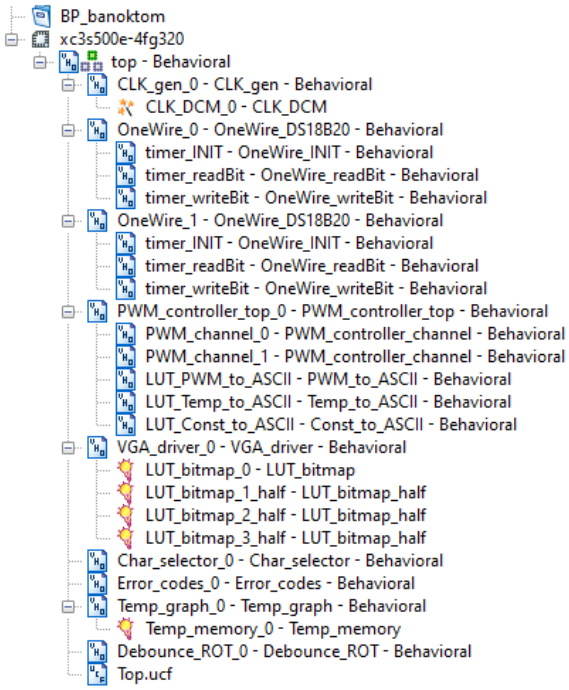
Popis funkce hlavních VHDL modulů je popsán na obr. 3.1 a jejich hierarchie, včetně dalších podřízených modulů, je zobrazena na obr. 3.2.

top	Zajišťuje propojení modulů mezi sebou a s periferiemi na vývojovém kitu
CLK_gen	Generování taktovacích signálů
OneWire	Čtení teplot z digitálního senzoru DS18B20
PWM_controller_top	Generování parametrizovaného PWM signálu
VGA_driver	Řadič pro řízení displeje
Char_selector	Výběr znaku pro zobrazení VGA řadičem
Error_codes	Převádí chybové stavy z 1-Wire modulů na znaky
Temp_graph	Znakový teplotní graf pro zobrazení historie teplot senzorů
Debounce_ROT	Debouncing výstupů otočného tlačítka

Obrázek 3.1: Popis funkce hlavních VHDL modulů.

Výše zmíněný modul top zajišťuje jen definování instancí podřazených modulů, jejich vzájemně propojení a propojení s periferiemi na vývojovém kitu.

Soubor *top.ucf* (na obr. 3.2 dole) popisuje přiřazení vstupních a výstupních signálů pinům, společně s výběrem využitých napěťových úrovní. V tomto souboru jsou zároveň definovány tzv. *timing constraints*, pomocí kterých se při implementaci definuje maximální akceptovatelný skluz taktovacích signálů.



Obrázek 3.2: Přehled vytvořených VHDL modulů v projektu a jejich hierarchie.

3.2 Vytvořená VHDL knihovna

Během tvorby projektu jsem také i vytvořil VHDL knihovnu, ve které jsem definoval vlastní datové typy a dvě funkce pro přístup k jednomu z nich. Tato knihovna slouží převážně ke zjednodušení kódu a jednoduššímu rozšíření (např. o další zdroj teploty, další PWM výstup apod.). Tyto datové typy jsou:

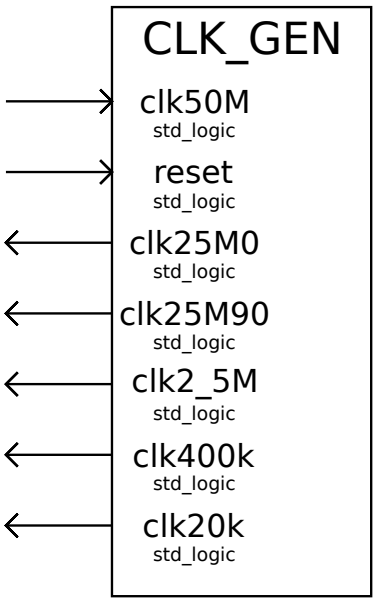
cstring	-	pole std_logic_vector(7 downto 0), používané pro 8bitový kód znaků,
ccolor	-	pole std_logic_vector(2 downto 0), používané pro barevnou masku znaků,
array#	-	pole std_logic_vector(# downto 0), kde # je celé číslo větší než 0,
bitmap	-	pole std_logic_vector(47 downto 0), které obsahuje bitmapu znaku.

Dále jsou definované dvě funkce se společným jménem to_cstring, které jsou určeny pro převod z datového typu string či bitové posloupnosti na cstring.

Zakódované znaky v cstring jsou ve spodních 7 bitech (hodnoty 0 až 127) kompatibilní s ASCII kódováním. Horní bit značí využití některého ze speciálních znaků, které slouží např. k vykreslení teplotního grafu (viz sekce Znakový teplotní graf) či indikaci pozice kurzoru při konfiguraci PWM signálu (viz sekce Změna nastavení). Výjimku tvoří znak ° (stupeň), který byl posunut na pozici 254, neboť jeho původní pozice v ASCII kódování byla obsazena.

3.3 Generování taktovacích signálů

3.3.1 Rozhraní



Obrázek 3.3: Rozhraní modulu CLK_gen.

3.3.2 Využití DCM

Do modulu vstupuje 50MHz taktovací signál z externího oscilátoru, který je připojen na vstup DCM. Tato komponenta umožňuje manipulovat s frekvencí a fází taktovacích signálů a do jisté míry i kompenzovat zkreslení[1] výstupních taktovacích signálů. Výstupem DCM je také signál LOCKED, který udává, zdali jsou výstupní taktovací signály stabilní (s žádanou frekvencí/fází). Při spuštění či po restartu potřebuje DCM navzorkovat až několik tisíc cyklů vstupního taktovacího signálu[2] než lze tuto stabilitu zaručit.

Výstupem použitého DCM jsou taktovací signály 25MHz (s fází 0° a 90°), 50MHz signál pro použití v děličkách kmitočtu a signál DCM_LOCKED pro jejich zastavení.

V práci využívám pro označování taktovacích signálů zápis clk[frekvence][fáze], tedy např. clk25M90 je taktovací signál o frekvenci 25 MHz s fázovým posunem 90° vůči clk25M0. Frekvenci v tomto značení zaokrouhluji, tedy např. pro frekvenci 396,8 kHz využívám značení clk400k. Dále jsem zvolil pro 2,5MHz taktovací signál označení clk2_5M.

[1] Angl.: clock skew.

[2] Dle dokumentace ve vývojomém prostředí Xilinx ISE - Clocking Wizard - General Setup

3.3.3 Nastavení frekvence výstupních signálů

Každý výstupní signál (kromě clk25M0 a clk25M90) je obsluhován vlastním procesem. Tyto procesy využívají stejné struktury, ve které vstupní taktovací signál inkrementuje čítač a jednotlivé děličky se liší pouze maximální hodnotou čítače *maxI* a obsluhovaným taktovacím signálem. Na obr. 3.4 je příklad procesu pro děličku na 400 kHz.

Čítače se inkrementují se vzestupnou hranou 50MHz taktovacího signálu a zastavují při ztrátě stability DCM. Dosáhne-li čítač hodnoty *maxI*, tak dojde k jeho vynulování a změně stavu obsluhovaného signálu (z log. '0' na log. '1' či obráceně). Hodnotou *maxI* tedy nastavujeme dobu jedné půlperiody, kterou lze získat dle vztahu

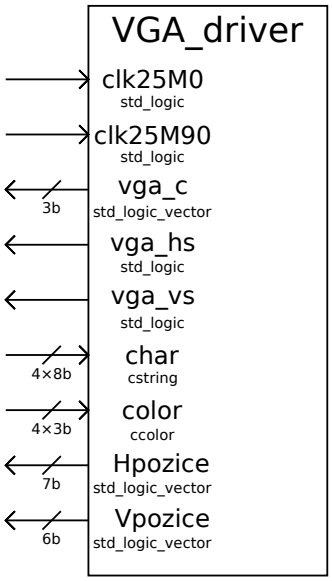
$$\mathrm{maxI} = \frac{f_{\mathrm{in}}}{2 \cdot f_{\mathrm{out}}} - 1 = \frac{50 \cdot 10^6}{2 \cdot f_{\mathrm{out}}} - 1. \tag{3.1}$$

```
-- delicka na 400 kHz
process(DCM_locked, s_clk50M)
constant maxI : integer := 62;
variable i : integer range 0 to maxI;
begin
   if(DCM_locked='0') then
     s_clk400k <= '0';
   elsif(rising_edge(s_clk50M)) then
     if(i=maxI) then
       i := 0;
       s_clk400k <= not s_clk400k;
     else
       i := i+1;
     end if;
   end if;
end process;
```

Obrázek 3.4: Příklad využitého procesu pro děličku kmitočtu na 400 kHz.

3.4 Řízení VGA výstupu

3.4.1 Rozhraní

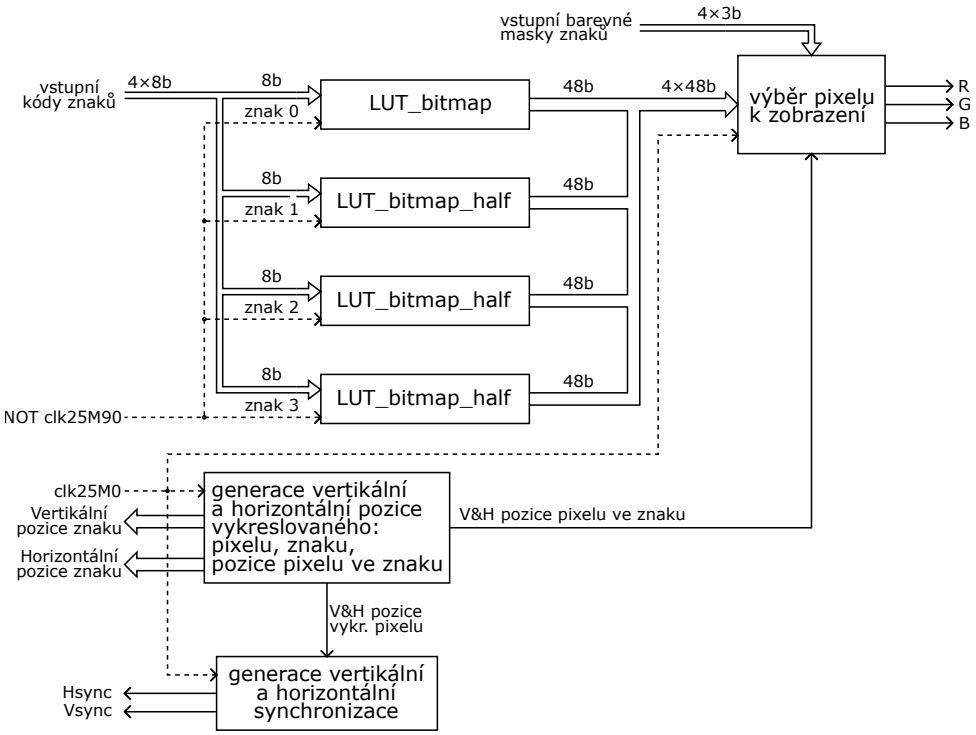


Obrázek 3.5: Rozhraní modulu VGA_driver.

Pro zjednodušenou obsluhu a menším požadavkům na paměť jsem se rozhodl využít princip znakového grafického řadiče (viz sekce VGA). Vykreslovaný obraz o velikosti 640×480 je rozdělen na pole 106×60 znaků o velikosti 6×8 pixelů s každým znakem popsaným jeho 8bitovým kódem a tříbitovou barevnou maskou.

Výstupem řadiče (kromě externích signálů, viz sekce VGA) je vertikální a horizontální pozice znaku k vykreslení a vstupem je pole kódu čtyř znaků a pole čtyř barevných masek, které umožňují vykreslení až čtyř barevných znaků přes sebe s prioritizací. Prioritizace umožňuje správné vykreslování teplotního grafu (více v sekci Písmo a vykreslování více znaků), ale také i možnost vyplnit pozadí znaku a zachovat jeho čitelnost (příklad na obr. 3.10). Vytvořený řadič je také popsán blokovým schématem na obr. 3.6.

Vzhledem k tomu, že znakové vstupy 1 až 3 nejsou v práci využity pro text, tak jsou jejich LUT jen poloviční – zahrnují jen kódy znaků 128 až 255 pro ušetření využité paměti. Znaky využité pro vykreslování teplotního grafu byly vygenerovány pomocí programu napsaného v C++, jehož zdrojový kód je zahrnut v elektronické příloze (graph_bitmap_generator.cpp), ale ve finální implementaci byla vygenerovaná data upravena do formátu, který je vhodný pro inicializaci ROM LUT.



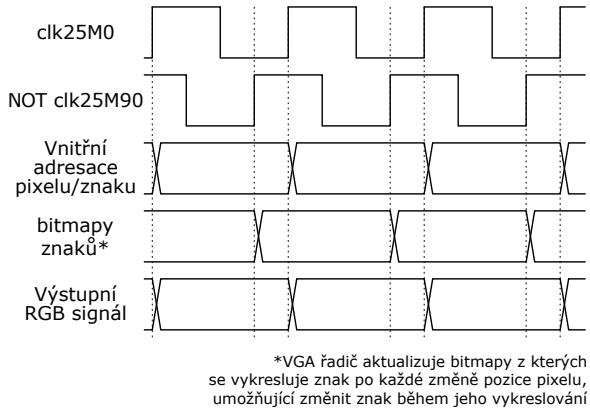
Obrázek 3.6: Blokové schéma VGA řadiče. Výběr pixelu k odeslání displeji je popsán na obr. 3.9.

3.4.2 Časování

Pro přenos obrazu o velikosti 640 × 480 pixelů s obnovovací frekvencí 60 Hz je definován tzv. *Pixel clock* o frekvenci 25,175 MHz ±5 % [4]. Převrácenou hodnotou této frekvence je doba vyhrazená pro přenos jednoho pixelu obrazu. Vzhledem k 50MHz oscilátoru na vývojovém kitu je využito této tolerance nastavením frekvence na 25 MHz.

Řadič pracuje ve dvou fázích, které jsou časovány pomocí taktů signálů clk25M0 a negace clk25M90, tedy efektivně clk25M270. Při vzestupné hraně clk25M0 se aktualizuje vnitřní adresace pixelu (a znaku, jsme-li na jeho konci). Pokud se vykreslovaný snímek nachází ve vykreslované oblasti, tak se zároveň pošle příslušný pixel z bitmapy znaku s aplikovanou barevnou maskou. V opačném případě dojde k vyslání černého pixelu. Časování řadiče a jeho dílčích funkcí je vidět na obr. 3.7.

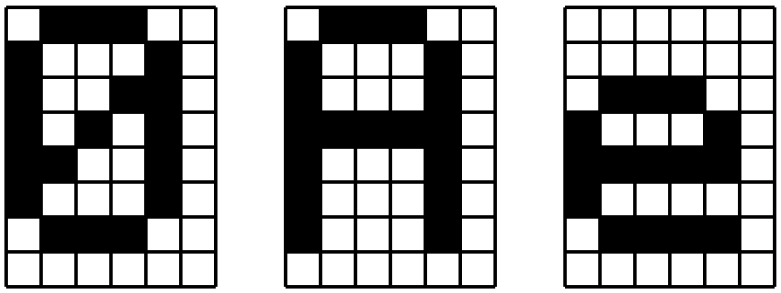
Nutnou podmínkou k zajištění stabilního výstupu je, aby vstup ROM LUT byl včas správně nastaven, a tudíž se vybrala bitmapa správného znaku. Vzhledem k využitému taktu 25 MHz a fázovému posuvu 270° to znamená, že od změny pozice vykreslovaného znaku po dekódování na jeho bitmapu může uplynout nejvýše 30 µs.



Obrázek 3.7: Vnitřní časování modulu VGA řadiče.

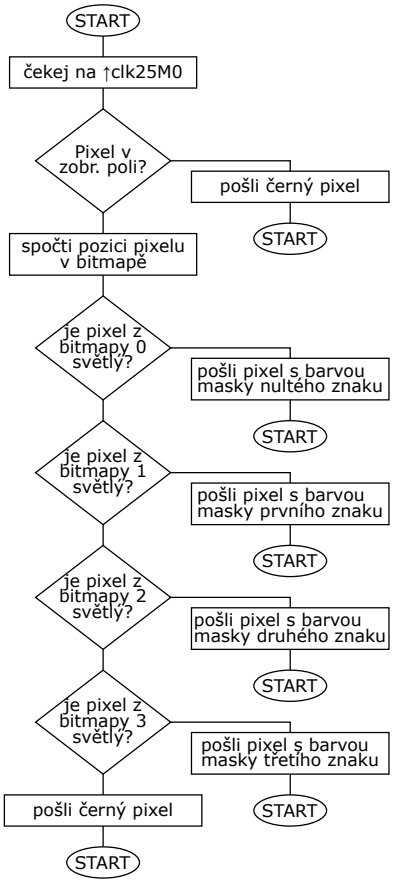
3.4.3 Písmo a vykreslování více znaků

Velikost písma jsem zvolil 6×8 pixelů včetně odsazení (obr. 3.8), neboť umožňuje dobrou čitelnost a relativně velké množství znaků k vykreslení na obrazovku. Vzhledem k množství dat nutných pro uložení bitmap (10 752 bitů) jsem se rozhodl vložit bitmapy do ROM paměti k uvolnění LUT umístěných v CLB za cenu využití BRAM bloku. Příklad těchto bitmap je vidět na obr. 3.8.

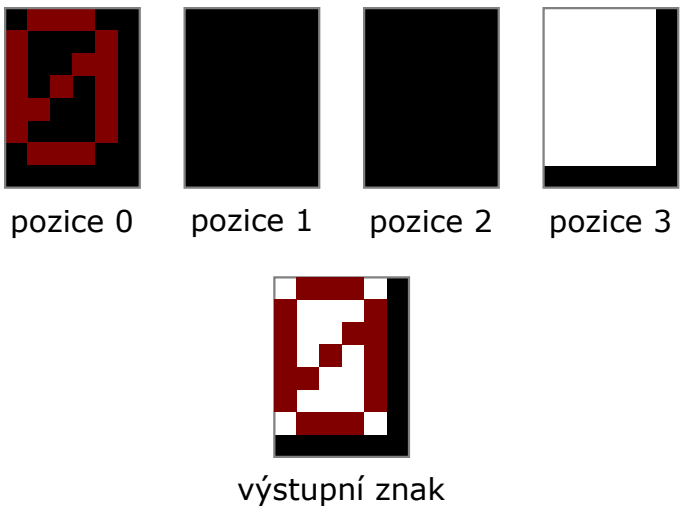


Obrázek 3.8: Příklady bitmapy znaků „0“, „A“ a „e“. Barvy invertovány pro lepší čitelnost.

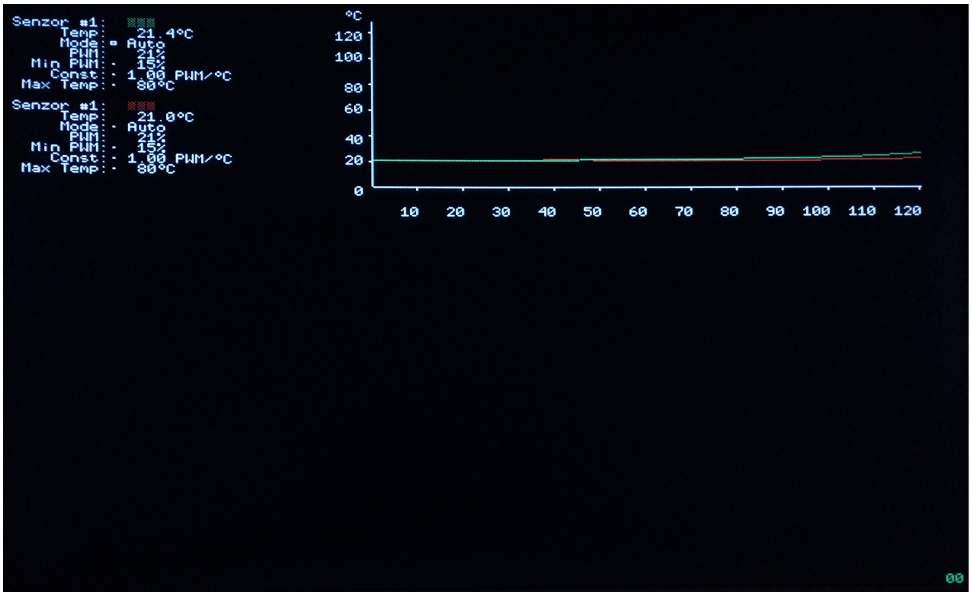
Prioritizace vykreslování znaků je popsána na obr.3.9. Příklad této prioritizace je poté ukázán na obr. 3.10. Celkový vykreslovaný obraz je vidět na obr. 3.11.



Obrázek 3.9: Stavový diagram popisující výběr barev pixelu k odeslání displeji.



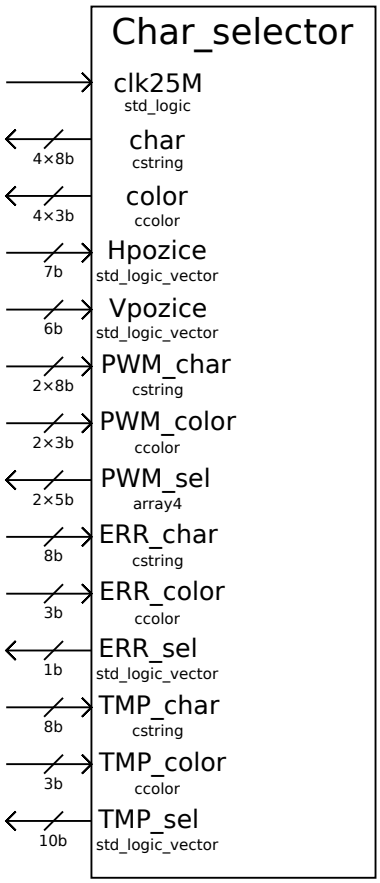
Obrázek 3.10: Příklad prioritizace znaků a jejich příslušné bitmapy. Pozicí se myslí index ve znakovém vstupu modulu.



Obrázek 3.11: Fotka vykreslovaného snímku. Vlevo se nachází konfigurace PWM kanálů, vpravo nahoře znakový teplotní graf zobrazující historii teplot a v pravém dolním rohu stavový kód indikující stav 1–Wire modulů.

3.5 Výběr znaku pro VGA řadič

3.5.1 Rozhraní

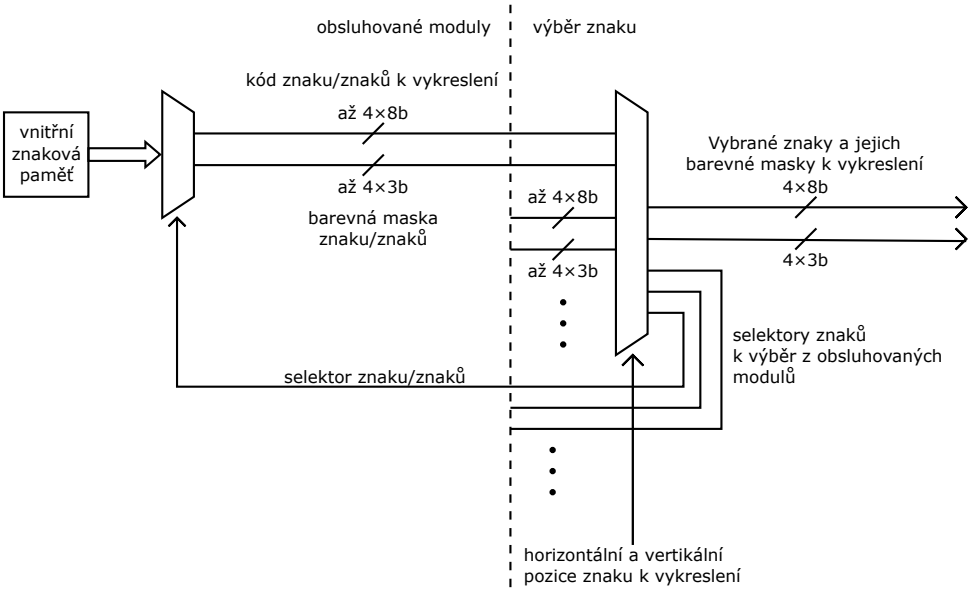


Obrázek 3.12: Rozhraní modulu Char_selector.

Modul pro výběr znaku se chová jako multiplexor, který dále řídí multiplexory dalších modulů, jejichž výstupem je znak k vykreslení. Vstupem modulu je 25MHz clock signál, vertikální a horizontální pozice znaku k vykreslení a pole znaků a jejich barevných masek z modulů, které vyžadují vykreslení znaku.

Výstupem je vybrané pole znaků a barevných masek k odeslání VGA řadiči a pozice vybíraného znaku z výstupu modulů (dále jen selektor). Žádný z výstupů však není synchronní. Interakce s obsluhovanými moduly je popsána blokovým schématem na obr. 3.13.

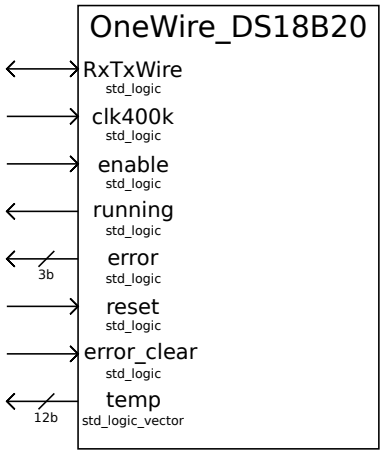
Modul dále obsahuje neměnný text, který je zobrazován na displeji, včetně doplňujících znaků pro graf (vykreslení os a vyznačení měřítka) a barevného označení PWM kanálů pro odlišení vykreslovaných průběhů grafu (viditelné na obr. 3.11 a obr. 3.23 a 3.24). Vstupní taktovací signál je vyhrazen pro LUT, která měla obsahovat tento text, ale nebyla implementována.



Obrázek 3.13: Blokové schéma interakce mezi modulem Char_selector s obsluhovanými moduly.

3.6 Čtení teploty senzoru

3.6.1 Rozhraní



Obrázek 3.14: Rozhraní modulu OneWire_DS18B20.

3.6.2 Časování a čtení teploty

Vstupem modulu je 400kHz taktovací signál pro vypínatelnou děličku kmitočtu a zapínací vstup, na který je nutné přivést impuls o délce alespoň 2 taktů 400kHz signálu pro zapnutí modulu. Tato dělička zprostředkovává kromě výstupní frekvence 200 kHz také i zapnutí a následné samovolné vypnutí modulu po skončení čtení teploty ze senzoru. Celý modul lze rozdělit na tři části – děličku kmitočtu, hlavní řídící proces a podřízené časovací moduly.

Hlavní řídící proces určuje sled operací (viz obr. 3.16), které řídí zapínáním tří podřízených modulů – INIT, readBit a writeBit. Podřízené moduly pracují s o 180° fázově posunutým taktovacím signálem pro zajištění ustálených stavů při komunikaci mezi podřízenými moduly a hlavním řídícím procesem. Výstupem podřízených modulů je žádaný stav 1–Wire sběrnice (Z[3] či log. 0), impuls indikující ukončení práce podmodulu a v případě INIT a readBit impuls pro přečtení aktuálního stavu 1–Wire sběrnice hlavním řídícím procesem.

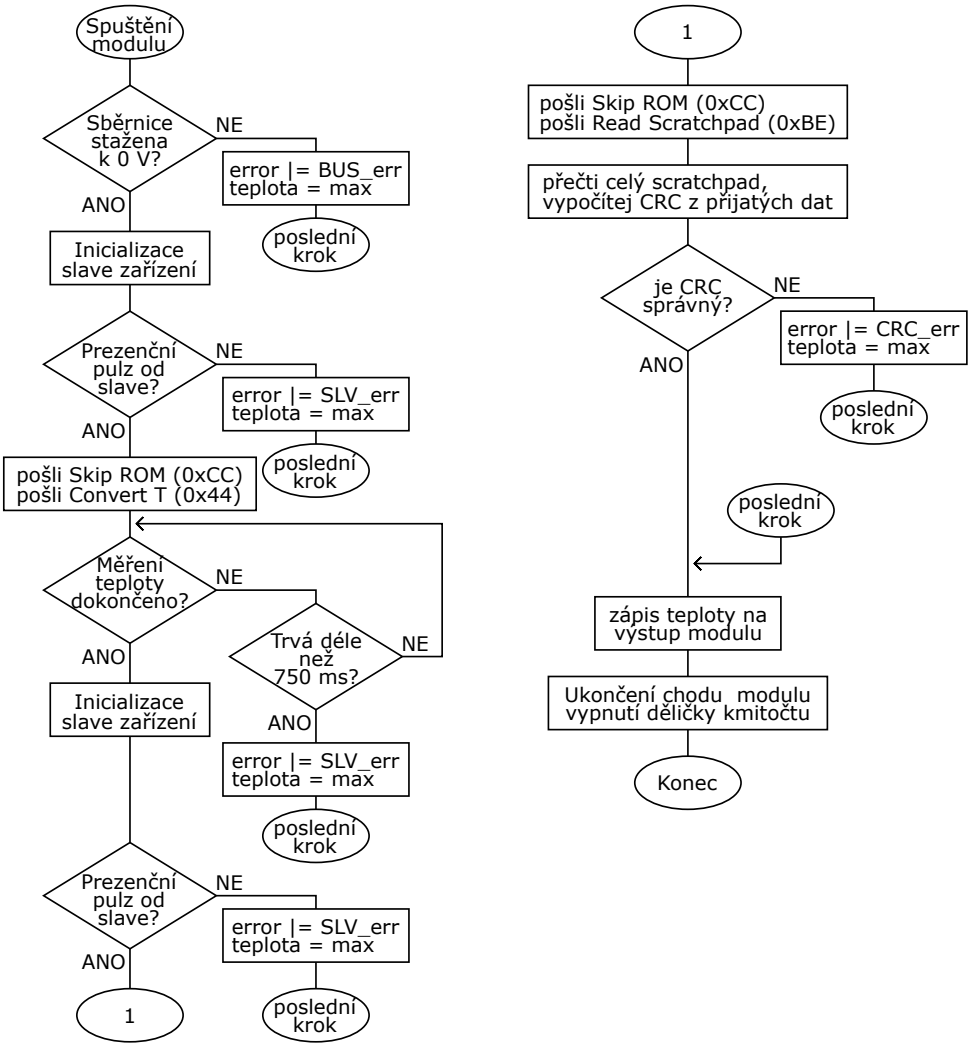
Frekvence 200 kHz byla zvolena pro snažší nastavení časování komunikace mezi master a slave, neboť jí odpovídá perioda 5 µs a dílčí kroky komunikace na 1–Wire sběrnici lze popsat v násobcích této periody (viz sekce Komunikace se slave zařízením).

Digitální senzor DS18B20 používá k zakódování teploty dvojkový doplněk, ale pro zjednodušení logiky navazujících modulů je teplota převáděna na celočíselný kód se znaménkovým bitem na pozici MSB (obr. 3.15).

[3]Stav vysoké impedance

S	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
11	10	9	8	7	6	5	4	3	2	1	0
MSB											LSB

Obrázek 3.15: Formát výstupní teploty s vyznačenou číselnou váhou a indexem bitů. „S“ značí pozici znaménkového bitu, kde „0“ značí kladné číslo a „1“ číslo záporné.



Obrázek 3.16: Stavový diagram řídícího procesu 1–Wire modulu.

3.6.3 Chybové výstupy

Kromě samotné komunikace s digitálním senzorem DS18B20 je modul doplněn o detekci chyb během komunikace. Dojde-li k detekci chyby, tak modul zapíše nejvyšší teplotu na svůj výstup (0b011111111111; 127,9375 °C), nastaví příslušný error bit na log. 1 a ukončí svůj chod. Detekce chyb je popsána

ve stavovém diagramu řídícího procesu (obr. 3.16) a chyby se dělí do tří kategorií:

- BUS_err (0b--1)

  Chyba sběrnice (sběrnice stažena k 0 V).

- SLV_err (0b-1-)

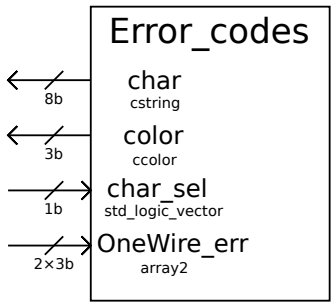
  Chyba slave zařízení. Na sběrnici se buď nenachází či neodpovídá slave zařízení, nebo konverze teploty trvá příliš dlouho.

- CRC_err (0b1--)

  Chyba CRC. Data vyslaná slave zařízením jsou poškozena, nelze zaručit správnost přijaté teploty.

Během jednoho běhu modulu může nastat jen jeden z výše popsaných chybových stavů, ale ve tříbitovém výstupním kódu zůstane záznam i o předešlých chybách. Chybový výstup lze vynulovat jen resetováním celého modulu. Např. výstupní kód 0b011 značí, že od minulého resetu nastala alespoň jednou chyba SLV_err a alespoň jednou chyba BUS_err.

3.7 Stavový kód 1–Wire rozhraní



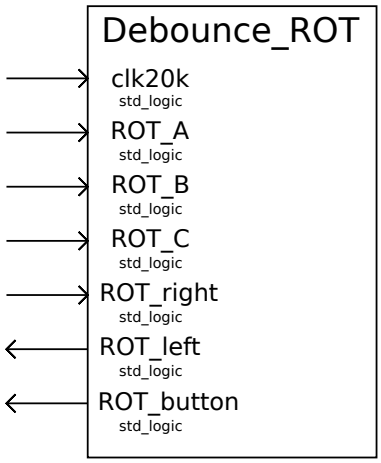
Obrázek 3.17: Rozhraní modulu Error_codes.

Pro zjištění chybových stavů 1–Wire sběrnice je využita dvojice znaků, z nichž každý zobrazuje chybový výstup jednoho z 1–Wire modulů. Jejich tříbitový kód je jen posunut o hodnotu 42, aby kódu 0b000 odpovídal znak „0“. Pro rychlejší zjištění chyb při přenosu jsou znaky zabarveny – červeně při BUS_ERR či SLV_ERR, žlutě při CRC_ERR a zeleně pokud nenastala žádná chyba. Při výskytu BUS_ERR (a/nebo SLV_ERR) a CRC_ERR zároveň je prioritizována červená barva.

Na obr. 3.11 je výstup modulu vidět v pravém dolním rohu.

3.8 Obsluha otočného tlačítka

3.8.1 Rozhraní



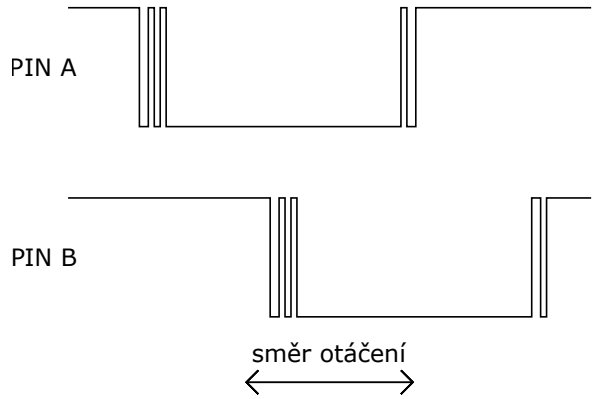
Obrázek 3.18: Rozhraní modulu Debounce_ROT.

3.8.2 Debouncing

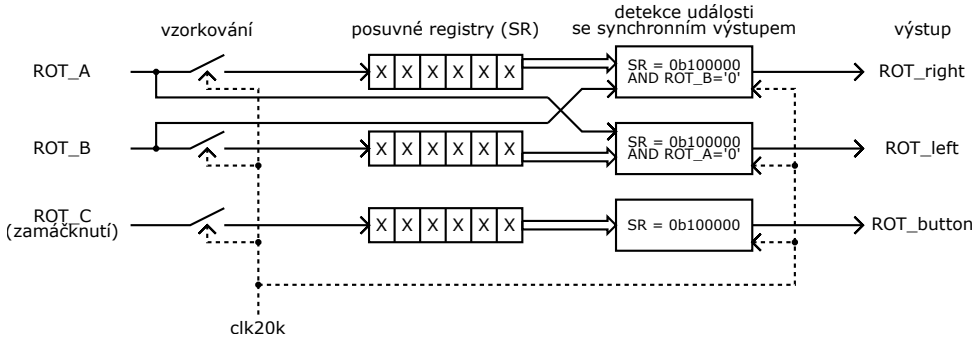
Debouncing je proveden vzorkováním výstupů otočného tlačítka s frekvencí 20 kHz a vkládáním vzorků na MSB[4] příslušných 6bitových posuvných registrů. S pomocí těchto registrů je provedena detekce vzestupné hrany výstupů, která naznačuje konec jedné otáčky či zamáčknutí otočného tlačítka. Při detekci vzestupné hrany na vstupech ROT_A či ROT_B je ještě nutné přečíst stav druhého z těchto vstupů. Pokud je druhý vstup ve stavu log. 0, tak došlo k úspěšné detekci jedné otáčky a jejího směru. Celý debouncing je principiálně popsán na obr. 3.20.

Důvod čtení stavu druhého vstupu je vidět na obr. 3.19, kde při otáčení vpravo bude detekována vzestupná hrana na vstupech obou pinů, ale podmínka stažení druhého vstupu bude splněna jen při vzestupné hraně pinu A. Vzorkovací frekvence a délka posuvných registrů byly určeny experimentálně. Při nižší frekvenci / delších posuvných registrech se snižuje schopnost detekovat rychlé otáčení, ale s vyšší frekvencí / kratšími posuvnými registry se zvyšuje šance detekce zákmitu.

[4]MBS vybrán kvůli popisu, který umožňuje parametrizovat velikost posuvného registru



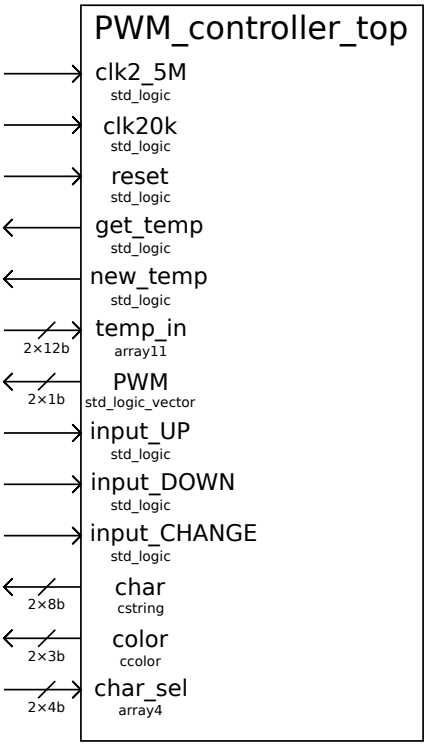
Obrázek 3.19: Časový průběh výstupů otočného tlačítka při jedné otáčce.



Obrázek 3.20: Principiální schéma pro debouncing vstupů otočného tlačítka.

3.9 Řízení ventilátorů

3.9.1 Rozhraní



Obrázek 3.21: Rozhraní modulu PWM_controller_top.

3.9.2 PWM signál

Pro zajištění kompatibility s druhy řízení popsanými v sekci Řízení ventilátorů byla zvolena frekvence PWM signálu 25 kHz, umožňující regulaci napájecího napětí či pomocí řídícího signálu dle Intel specifikace [5]. Zároveň byly i ošetřeny stavy při nastavení 0% a 100% střídy signálu, při kterých je výstupní signál neustále stažen k 0 V či 3,3 V. Vzhledem k omezené možnosti FPGA dodávat či odebírat proud je nutné využít tranzistor ke spínání napájecího vodiče či PWM vodiče ventilátoru.

Každý z PWM výstupů (dále kanálů) je konfigurovatelný uživatelem s pomocí VGA výstupu za chodu modulu. Každý kanál může fungovat ve dvou režimech řízení – manuálním (konstantní střída) a automatickém.

Manuální řízení

Nejjednodušší režim, při kterém je výstupní střída konstantní bez ohledu na teplotě příslušného senzoru. Uživatel může pouze měnit nastavenou střídu

a režim kanálu (manuální či automatický).

Automatické řízení

Automatickou regulaci lze rozdělit do tří dílčích požadavků, jejichž výstupem je požadovaná střída. Z těchto stříd se poté vybere ta s nejvyšší hodnotou (na obr. 3.22 vyobrazeno červeně).

- Minimální střída

  Nejnižší možná střída PWM signálu, zabraňuje zastavení ventilátoru. Nastavitelné od 0 do 100 % s krokem 1 %. Na obr. 3.22 zeleně.

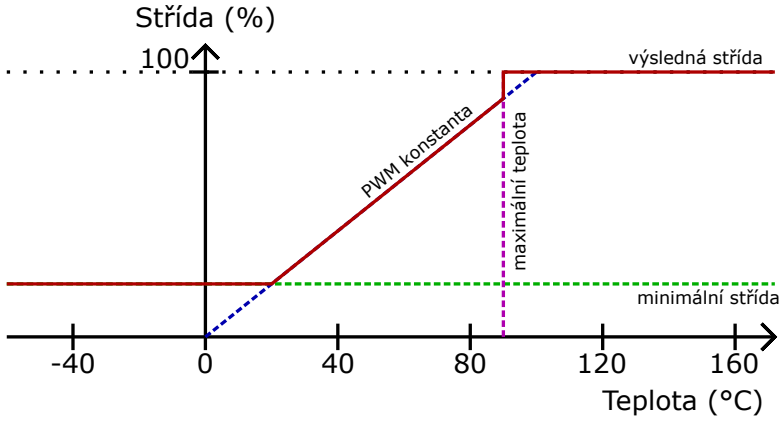
- PWM konstanta (%/°C)

  Popisuje strmost PWM regulace. Nastavitelné od 0,00 do 3,75 s krokem 0,25 %/°C. Na obr. 3.22 modře.

- Maximální teplota

  Při dosažení této či vyšší teploty se nastaví střída na 100 %. Kofigurovatelné od 0 do 127 °C s krokem 1 °C. Na obr. 3.22 purpurově.

Uživatel v tomto módu může měnit výše popsanou konfiguraci a režim kanálu.



Obrázek 3.22: Nastavení střídy PWM signálu v automatickém režimu. Vyobrazeno nastavení: minimální střída 20 %, PWM konstanta 1,00 %/°C a maximální teplota 90 °C. Červená křivka kopíruje výslednou závislost střídy na teplotě.

3.9.3 Změna nastavení

Změna parametrů PWM řízení je prováděna pomocí otočného tlačítka a grafického rozhraní, které je vykreslováno na připojený displej. Vnitřní paměť pro znakový výstup obsahuje teplotu senzoru, režim řízení PWM kanálu, aktuální střídu, minimální střídu, PWM konstantu, maximální teplotu a kurzor pro výběr a změnu parametrů ve formátu na obr. 3.25. Zamáčknutím otočného

tlačítka dochází k přepnutí kurzoru mezi změnou polohy a změnou hodnoty na kterou kurzor ukazuje, přičemž poloha či hodnota se mění otáčením tlačítka. Pozice a mód kurzoru je indikován znaky na obr. 3.26.

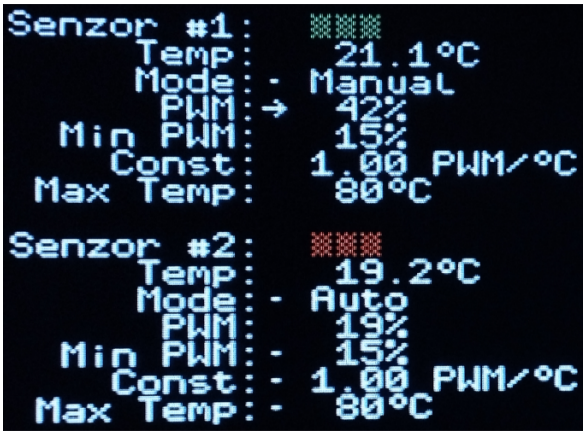
Jednotlivé PWM kanály jsou řazeny „za sebou“, kdy při přechodu z poslední pozice prvního kanálu přechází kurzor na první pozici dalšího kanálu a při nastavování parametrů lze využít přetečení z nejvyšší hodnoty na nejnižší (a obráceně) nastavovaného parametru. Grafické rozhraní je vidět na obrázcích 3.23 a 3.24.

3.9.4 Struktura

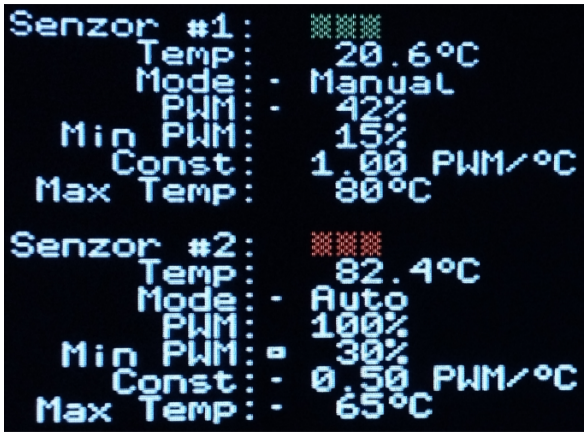
Pro zmenšení využitých prostředků module je modul rozdělen na tři části:

- Hlavní modul, který obsluhuje změnu parametrů, jejich skladování, generaci znaků z parametrů / teplot a vnitřní znakovou paměť
- Moduly obsluhující jednotlivé PWM kanály, které z teploty a parametrů vygenerují PWM signál o požadované střídě
- LUT pro převod z parametrů a teploty na 8bitové kódy znaků. Pro snížení využitých prostředků jsou LUT přepínány mezi jednotlivými PWM kanály.

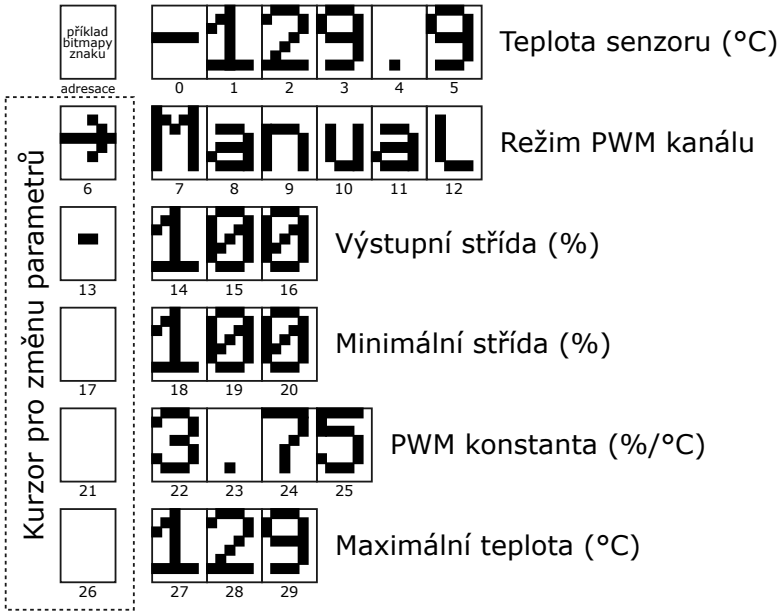
Obsah LUT pro převod z teplot na kód znaku a LUT pro převod ze střídy PWM signálu na kód znaku byly vygenerovány pomocí dvou programů psaných v jazyce C++. Zdrojové kódy těchto programů jsou zahrnuty v elektronické příloze (temp_to_ASCII.cpp a pwm_to_ASCII.cpp).



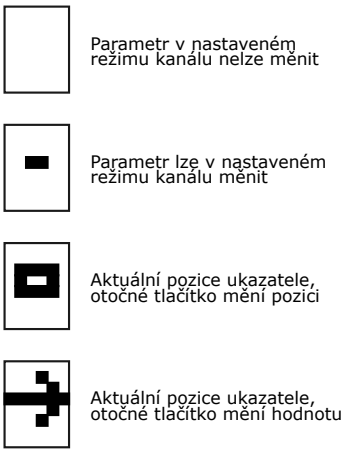
Obrázek 3.23: Příklad nastavení dvou PWM kanálů. Kurzor se nachází v nastavení střídy prvního kanálu, přičemž otáčením otočného tlačítka se mění hodnota výstupní střídy.



Obrázek 3.24: Příklad nastavení dvou PWM kanálů. Kvůli přesáhnutí maximální teploty senzoru 2 je výstupní střída druhého kanálu 100%.



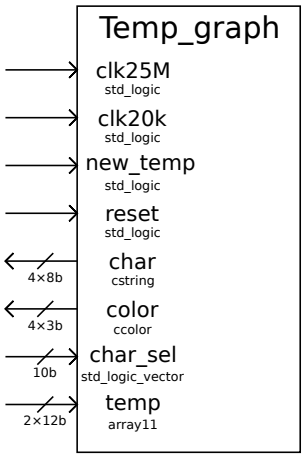
Obrázek 3.25: Vnitřní znaková paměť PWM modulu, která je adresována selektorem s příkladem zobrazovaných znaků.



Obrázek 3.26: Popis znaků využitých pro kurzor.

3.10 Znakový teplotní graf

3.10.1 Rozhraní



Obrázek 3.27: Rozhraní modulu Temp_graph.

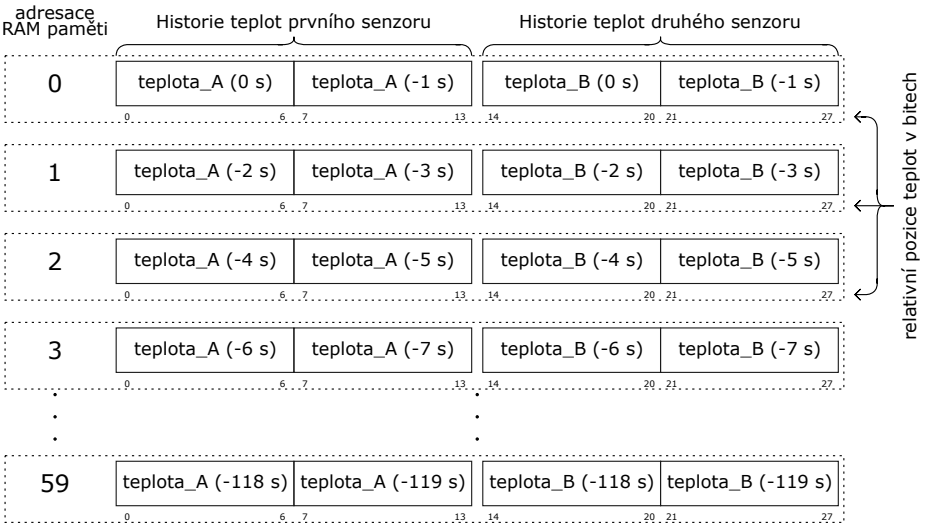
Vytvořený modul umožňuje vykreslovat historii teplot připojených senzorů na pole 16 × 60 znaků s rozlišením časové osy 1 s / 3 pixely a rozlišením teplotní osy 1 °C / 1 pixel. Modul tedy dokáže vykreslit data v rozsahu 1 – 127 °C za posledních 120 s, přičemž teploty pod 1 °C se vykreslují stejně jako 1 °C pro zachování spojitosti grafu. Každé vykreslované teplotě je pevně definována barevná maska a v případě překrytí teplot senzorů se vykreslí jen barva, jejíž výstup se nachází na nižším indexu pro zachování čitelnosti (viz sekce Písmo a vykreslování více znaků).

Vstupem modulu je selektor znaku, taktovací signály 25 MHz a 20 kHz a signál pro posun teplot grafu o jednu sekundu. Výstupem je pole znaků a barevných masek, jejichž velikost je rovna počtu vykreslovaných teplot. Vstupní 10bitový selektor ze sekce Výběr znaku pro VGA řadič je rozdělen na dvě části: bity na pozici 0 – 5 obsahují horizontální pozici vykreslovaného znaku a bity na pozici 6 – 9 obsahují vertikální pozici vykreslovaného znaku. Toto rozdělení je nutné pro snazší výběr teplot k vykreslení.

3.10.2 Využití blokové RAM

Pro ukládání historie teplot byla zvolena RAM paměť z BRAM bloků, aby nedošlo k využití velkého počtu CLB. Vzhledem k časovému rozlišení 1 s / 3 pixely neboli vykreslování dvou teplot na jeden znak je nutné přečíst dvě hodnoty každé vykreslované teploty najednou. Využitá RAM pracuje v tzv. *True Dual-Port* módu, ve kterém dvě plnohodnotná a na sobě nezávislá rozhraní přistupují ke společnému paměťovému prostoru. V tomto modulu využívám jedno rozhraní pro čtení teplot, zatímco druhé zprostředkovává zápis nových a posun stávajících dat.

Pro zjednodušení obsluhy RAM paměti se využívá vertikální pozice vykreslovaného znaku ze selektoru pro adresaci čtení z RAM paměti, díky čemuž je nutné skladovat dvě po sobě jdoucí teploty na společné adrese pro každý vykreslovaný průběh. Tento způsob skladování dat je vyobrazen na obr. 3.28.



Obrázek 3.28: Využití adresního prostoru RAM paměti pro skladování historie teplot.

3.10.3 Vkládání teplot do RAM

Zápis nových teplot a posouvání se stává těžší ze dvou důvodů – dvě teploty jednoho průběhu sdílejí svou adresu v RAM, přičemž na nižší pozici musí být vždy novější z těchto dvou teplot.

Zápis probíhá ve čtyřech krocích:

1. Úprava vstupních teplot

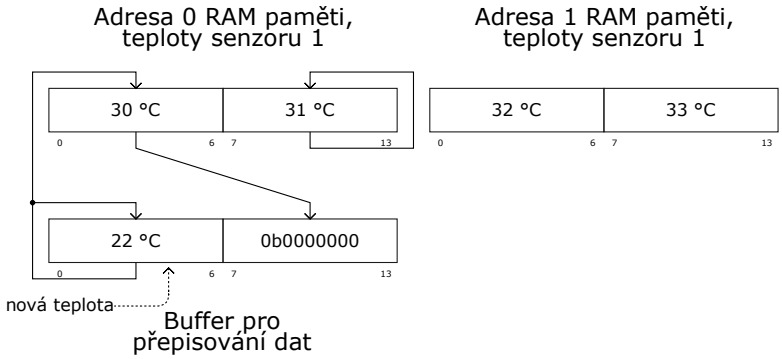
   Teplotní graf zobrazuje jen teploty od 1 °C s rozlišením 1 °C, není tedy nutné do RAM ukládat znaménkový bit a bity s nižší váhou než 1. Od vstupní teploty je také odečtno 0b1, aby ukládané sekvenci 0b0000000 odpovídala teplota 1 °C. V případě záporné teploty je do RAM ukládána teplota 1 °C.

2. Přepsání spodní části RAM (obr. 3.29)

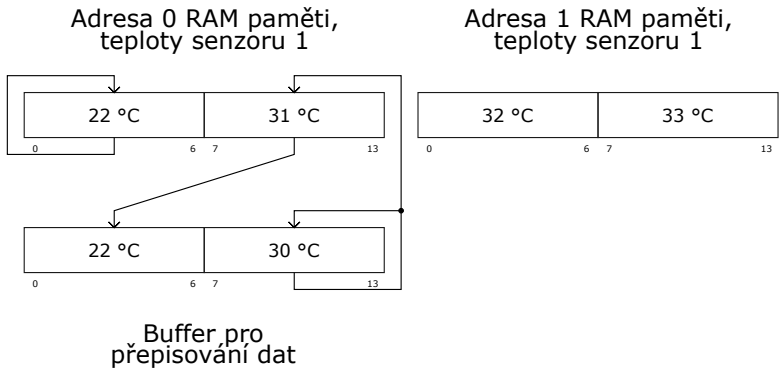
3. Přepsání horní části RAM (obr. 3.30)

4. Posun na další pozici RAM (obr. 3.31)

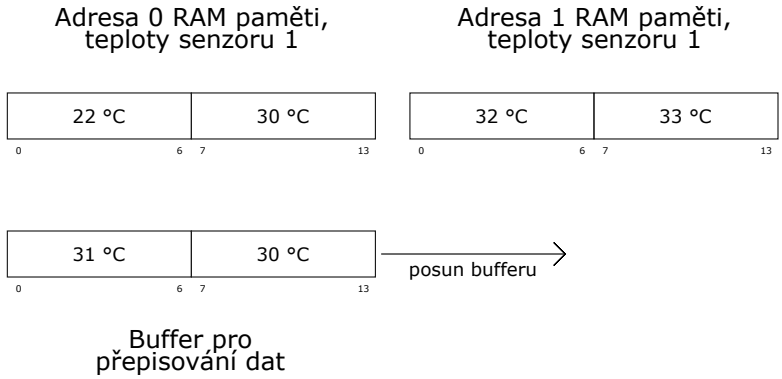
5. Opakování kroků 2, 3 a 4 dokud nejsou přepsány všechny pozice RAM (obr. 3.32)



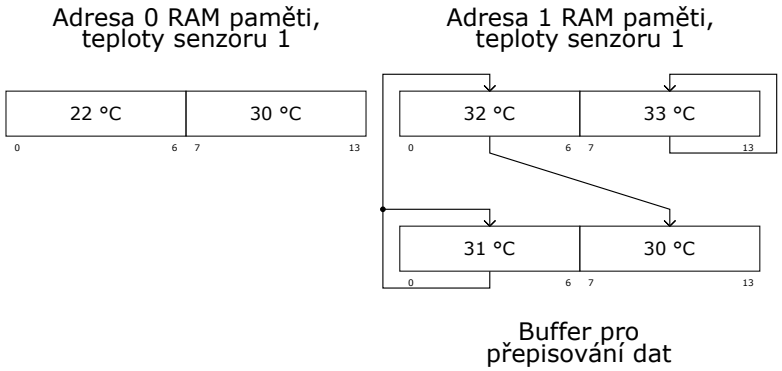
Obrázek 3.29: Přepsání spodní pozice obsahu RAM paměti. Obrázek popisuje jen přepsání teploty z prvního senzoru, ale stejným způsobem se přepisuje i teplota z druhého senzoru, která se nachází na vyšších pozicích bitů v RAM (viditelné na obr. 3.28).



Obrázek 3.30: Přepsání vyšší pozice obsahu RAM paměti.



Obrázek 3.31: Posun vstupů/výstupů přepisovacího bufferu na další adresu RAM.



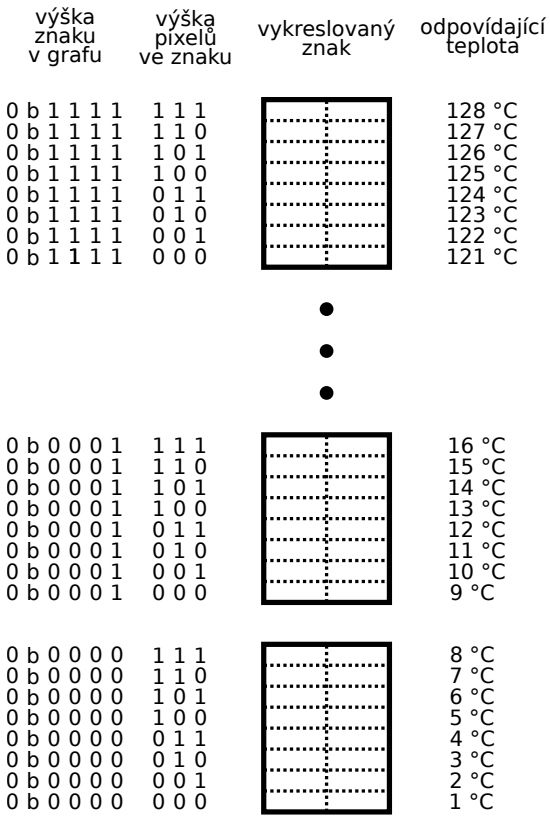
Obrázek 3.32: Opakování předešlých kroků zápisu do RAM paměti, dokud není obsah celé paměti posunut.

3.10.4 Čtení teplot z RAM, výběr znaku

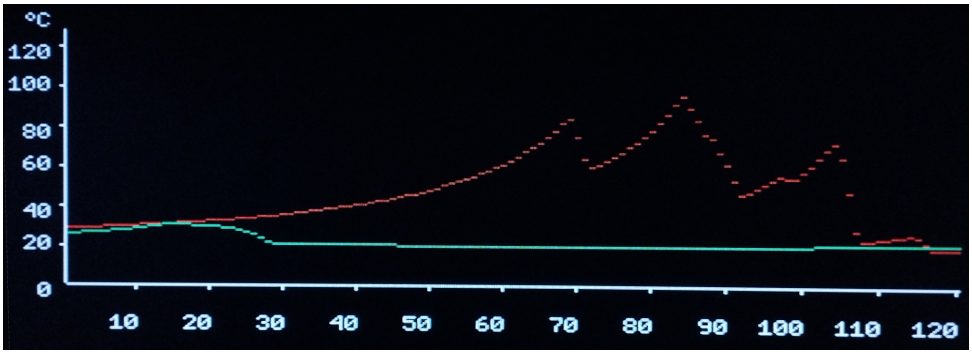
Adresace rozhraní RAM paměti pro čtení využívá bity 0 – 5 selektoru znaku, které udávají sloupec požadovaného znaku grafu. Z RAM paměti jsou poté vyčteny dvě teploty pro každý vykreslovaný průběh.

Pokud horní čtyři bity přečtené teploty odpovídají bitům na pozici 6 – 9 selektoru (udává požadovaný řádek znaku grafu, na obr. 3.33 „výška znaku v grafu“), tak dojde k vykreslení této teploty na jednu polovinu znaku. Výška trojice pixelů ve znaku je poté určena ze spodních tří bitů teploty (na obr. 3.33 „výška pixelů ve znaku“).

Vykreslený průběh dvou teplot je vidět na obr. 3.34.



Obrázek 3.33: Vyobrazení vztahu mezi uloženou sekvencí bitů teploty a výškou vykreslené trojice pixelů s jejich odpovídající teplotou. Vrchní 4 bity jsou využity k rozhodnutí, jestli bude teplota vykreslována do požadovaného znaku. Spodní 3 bity určují pozici vykreslované trojice pixelů jednoho znaku. V levé části znaku je vykreslována teplota na nižší pozici přečtených dat z RAM (novější teplota), v pravé části teplota na vyšší pozici (starší teplota).



Obrázek 3.34: Vykreslený průběh teplot dvou senzorů.

Kapitola 4

Závěr

Dle zadání byl vytvořen PWM regulátor pro regulací otáček ventilátoru. Regulátor je rozdělen do dvou separátních kanálů, které využívají k nastavení střídy svého výstupu teplotu z přiřazených digitálních teploměrů. Aktuální nastavení kanálů je dále zobrazováno na připojený VGA monitor a s pomocí otočného tlačítka je možné parametry PWM regulace měnit za běhu modulu. Připojený displej je také využit pro zobrazování historie teplot senzorů pomocí znakového grafu, který vykresluje teploty v rozsahu 1 – 127 °C staré až 119 sekund a kód pro zjištění stavu modulů obsluhující digitální teploměry.

Praktická část práce popisuje funkci a rozhraní jednotlivých modulů, pro které jsou využité blokové schémata a vývojové diagramy. Bloky, které obsahují znakový výstup jsou doplněny o fotografie celého displeje nebo jeho relevantní části. Vytvořený projekt je obsažen v elektronické příloze a kromě zdrojových kódů VHDL souborů obsahuje také i inicializační soubory pro BRAM bloky.

V projektu jsem také využil tří nástrojů psaných v jazyce C++ pro generování částí LUT. Zdrojové kódy těchto programů jsou zahrnuty v elektronické příloze.

Projekt může být dále rozšířen pro obsluhu dalších digitálních senzorů a PWM kanálů, přičemž teploty mohou být získávány z různorodých senzorů využívající jiná rozhraní pro komunikaci. Největším vylepšením projektu by byla implementace BRAM pro ukládání měněnných znaků, která by vedla ke značnému poklesu počtu využitých CLB.

Práce mi umožnila se blíže seznámit s jazykem VHDL a prohloubit své znalosti nejen o jazyku samotném, ale i o nízkoúrovňové obsluze periferií.

Literatura

[1] Handson Technology, *Rotary Encoder for Arduino/Raspberry* [online]. [cit. 2022-05-14]. Dostupné z: `https://www.handsontec.com/dataspecs/module/Rotary%20Encoder.pdf`

[2] Xilinx, *Spartan-3E FPGA Family Data Sheet* [online]. [cit. 2022-05-13]. Dostupné z: `https://datasheetspdf.com/pdf-file/1409860/Xilinx/XC3S500E/1`

[3] Diligent, *Spartan-3E Starter Kit Board User Guide* [online]. [cit. 2021-12-05]. Dostupné z: `https://digilent.com/reference/_media/reference/programmable-logic/spartan-3e/s3estarter_ug.pdf`

[4] VESA, *VESA and Industry Standards and Guidelines for Computer Display Monitor Timing (DTM)* [online]. [cit. 2021-12-05]. Dostupné z `https://glenwing.github.io/docs/VESA-DMT-1.13.pdf`

[5] Intel Corporation, *4-Wire Pulse Width Modulation (PWM) Controlled Fans Specification* [online]. [cit. 2022-01-03]. Dostupné z `https://www.glkinst.com/cables/cable_pics/4_Wire_PWM_Spec.pdf`

[6] Noctua, *Noctua PWM specifications white paper* [online]. [cit. 2022-01--03]. Dostupné z `https://noctua.at/pub/media/wysiwyg/Noctua_PWM_specifications_white_paper.pdf`

[7] Maxim Integrated, *DS18B20 Programmable Resolution 1-Wire Digital Thermometer* [online]. [cit. 2022-01-03]. Dostupné z `http://www.brrr.cz/specifikace/DS18B20.pdf`

[8] Chris. *Basic Arduino VGA - Signal Theory*. In: *PyroElectro.com Projects & Tutorials*. [cit. 2022-01-06]. Dostupné z: `http://www.pyroelectro.com/tutorials/arduino_basic_vga/theory.html`

Příloha A
Seznam zkratek

angl.	anglicky
apod.	a podobně
log.	logická hodnota
obr.	obrázek
tzv.	takzvaně
ASIC	Application-Specific Integrated Circuit
ASCII	American Standard Code for Information Interchange
BLDC	Brushless DC
BRAM	Block RAM
CLB	Configurable Logic Blocks
CRC	Cyclic Redundancy Check
CRT	Cathode Ray Tube
DCM	Digical Clock Manager
ERR	Error
FPGA	Field Programmable Gate Array
INIT	Inicializace
IOB	Input/Output Blocks
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
LUT	Lookup Table
MSB	Most Significant Bit
PC	Personal computer
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RPM	Revolutions Per Minute
SLV	Slave (zařízení)
USB	Universal Serial Bus
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Příloha B
Obsah elektronické přílohy

Obsahem této přílohy je vytvořený projekt se všemi vytvořenými moduly a soubory pro inicializaci obsahu BRAM bloků (použitých jako ROM LUT), materiály z kterých bylo v průběhu tvorby projektu a psaní práce čerpáno a zdrojové kódy tří programů psaných v jazyce C++, které byly využity pro generaci části obsahu LUT.