České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

Jednoduchý zabezpečovací systém vytvořený z přípravku Nexys

Viktorie Pražáková

Vedoucí práce: Ing. Pavel Lafata, Ph.D.
2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pražáková** Jméno: **Viktorie** Osobní číslo: **459161**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Aplikovaná elektronika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Jednoduchý zabezpečovací systém vytvořený z přípravku Nexys

Název bakalářské práce anglicky:

Simple Security System Based on Nexys Kit

Pokyny pro vypracování:

Využijte přípravek Nexys3/4 od společnosti Digilent k realizaci jednoduchého systému pro zabezpečení budov (objektů). K danému kitu připojte maticovou číselnou klávesnici, čidlo detekce pohybu (PIR) a reproduktor.
Využijte rovněž vestavěné periférie kitu Nexys. Pomocí nich realizujte jednoduchý systém zadávání kódu pomocí klávesnice, detekci pohybu senzorem a případné spuštění alarmu v případě detekce neoprávněného vstupu.
Realizaci proveďte pomocí jazyka VHDL.
Ověřte funkci navrženého systému alarmu.
Zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

[1] Pinker, J.; Poupa, M.: Číslicové systémy a jazyk VHDL (1. vydání). BEN - Technická literatura, Praha 2006. ISBN 80-7300-198-5.
[2] Digilent Xilinx - Manuály a materiály k vývojovým kitům Nexys3/4.

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: **31.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. Pavel Lafata, Ph.D.		prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	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.

14.5.2018
Datum převzetí zadání

Podpis studentky

Poděkování

Tímto bych chtěla poděkovat vedoucímu mé bakalářské práce Ing. Pavlu Lafatovi, Ph.D. za konzultace průběžných výsledků a za jeho rady.

Dále děkuji své rodině a přátelům za podporu a připomínky.

Prohlášení

Prohlašuji, že jsem bakalářskou práci na téma „Jednoduchý zabezpečovací systém vytvořený z přípravku Nexys“ vypracovala samostatně a s použitím uvedené literatury.

V Praze, dne 25. 5. 2018 _____

Abstrakt

Tato bakalářská práce se zabývá návrhem VHDL modulu pro zabezpečovací systém, který je realizován na přípravku Nexys 3. V první polovině práce je teoretická část, která přibližuje pohled na jednotlivé komponenty, se kterými jsem se setkala. V části druhé je samotná realizace a vysvětlení funkčnosti.

- **Klíčová slova:** Zabezpečovací systém, FPGA, VHDL, Nexys 3

Abstract

This bachelor thesis deals with the design of VHDL module, which is used as a security system on Nexys 3 trainer board. The first part of the thesis consists of theory of each individual component with which i have worked. The second part concerns my design a explanation of functionality.

- **Index terms:** Security system, FPGA, VHDL, Nexys 3

Obsah

1 Úvod 1
2 Teoretická část 3
2.1 Použitý hardware 3
2.1.1 Digilent Nexys 3® Spartan®-6 FPGA Trainer Board 3
2.1.2 Vestavěné periferie 4
2.1.3 Externí doplňky 5
2.2 Použitý software 7
2.2.1 Software 7
2.2.2 Jazyk VHDL 7
3 Praktická část 9
3.1 Popis individuální realizace 9
3.2 Popis jednotlivých bloků VHDL kódu 10
3.2.1 Dělička impulzů 10
3.2.2 Generování signálu do klávesnice 10
3.2.3 Vzorkování signálu 11
3.2.4 Čtení z klávesnice 11
3.2.5 Přiřazení vybraných čísel ke znaku na displayi 11
3.2.6 Vybírání pozic 11
3.2.1 Promítání na display 12
3.2.2 Senzor + alarm 12
3.2.3 Změna časovače spuštění alarmu 13
3.2.4 LED indikátory 13
4 Závěr 15
Použitá literatura 17
Přílohy 19
Příloha A: Blokové schéma 20
Příloha B: VHDL modul 21

1 Úvod

V této práci se věnuji návrhu VHDL modulu pro přípravek Nexys 3, ke kterému jsou zapojeny externí periferie. Finální zapojení funguje jako jednoduchý zabezpečovací systém. Realizován je pomocí detektoru pohybu, reproduktoru a klávesnice připojených na I/O porty kitu.

V první části práce se věnuji teoretickému úvodu, kde se zaměřuji na popsání a vysvětlení všech použitých komponentů. Tato kapitola je rozdělena na dvě poloviny, kdy v první z nich se rozebírá použitý hardware (Nexys 3, externí periferie) a ve druhé software (VHDL, ISE Design Suite, Digilent Adept).

V druhé části, tedy části praktické, se zabývám přesným popisem modulu blok po bloku. Vysvětluji funkčnost celého systému i za pomocí vývojových diagramů a blokového schématu modulu. Jde o zpracování signálu z detektoru pohybu a o následnou reakci systému. V případě nečinnosti dojde automaticky ke spuštění alarmu, a naopak při zadání správného kódu na klávesnici se odpočet zruší, popřípadě se přeruší již spuštěný alarm.

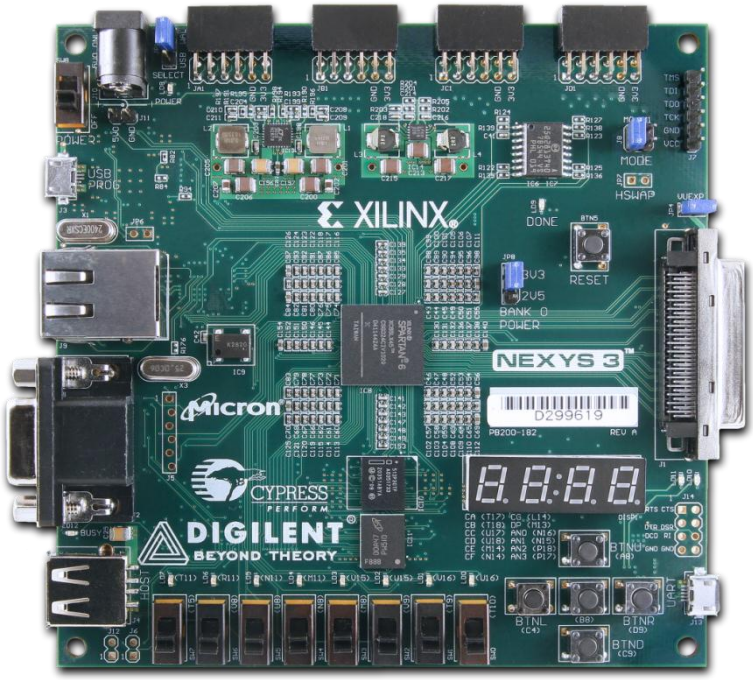
2 Teoretická část

2.1 Použitý hardware

2.1.1 Digilent Nexys 3® Spartan®-6 FPGA Trainer Board

Nexys 3 od společnosti Digilent, Inc. je základním prvkem mé bakalářské práce. Je to digitální vývojový kit, jehož hlavní složkou je FPGA – v tomto případě se jedná o Spartan 6 od společnosti Xilinx, který má až 150 tisíc logických bloků a výrobní technologii o 45 nm (více informací níže viz odstavec FPGA).

Kit obsahuje mnoho vestavěných periferií, mezi které patří například čtyřmístný sedmisegmentový display, tlačítka, LED či vnitřní oscilátor o 100 MHz. Z konektorů bych uvedla USB, UART, VGA, VHDC a 10/100 Ethernet (programování a napájení je realizováno přes microUSB). Co ovšem v mém případě stojí za zmínku je 32 I/O portů pro zapojení libovolných vstupně/výstupních prvků kompatibilních s úrovněmi TTL logiky.



Obrázek 1 - Nexys 3

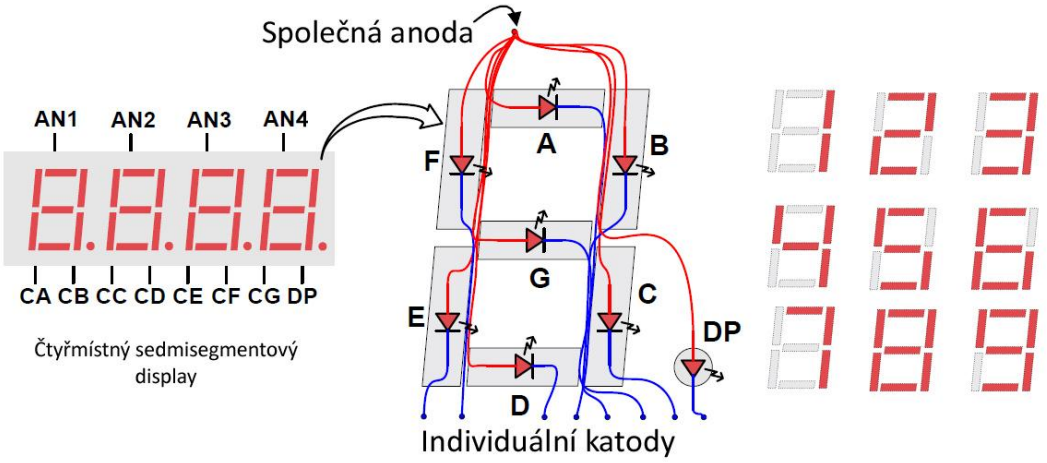
FPGA

Field Programmable Gate Array, tedy v překladu programovatelné hradlové pole, je logický integrovaný obvod specifický svou všestranností. Naprogramuje se až u cílového zákazníka a může se z něj vytvořit jakékoliv digitální zařízení. Skládá se z velké matice logických bloků, kde je možné vše propojit se vším. Na rozdíl od mikrokontrolérů se zde pracuje s volatilní konfigurací, tedy že čip FPGA si nepamatuje předchozí nastavení a pokaždé když se znovu zapne, tak se musí konfigurační soubor nahrát z externí paměti.

2.1.2 Vestavěné periferie

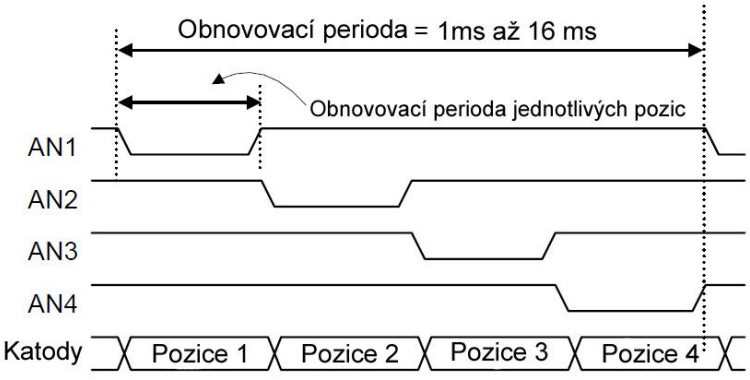
Display

Jedna z vestavěných periferií na kitu Nexys 3 je čtyřmístný sedmisegmentový display, který jsem využila pro zobrazení čísel stisknutých na externí klávesnici. Display se skládá ze čtyř anod a osmi katod a pomocí jejich kombinací vznikají čísla, popřípadě jiné znaky. Každá anoda reprezentuje jednu ze čtyř číselných pozic a každá katoda jeden ze sedmi segmentů u všech pozic (viz Obrázek 2). Takže například pro zobrazení čísla jedna na první pozici, bude napájena katoda B a C a anoda 1.



Obrázek 2 - Princip fungování displaye

Jelikož display vždy může zobrazovat najednou pouze jeden znak (ať už na jedné nebo více pozicích), tak jsem musela naprogramovat kód, který umožní na každé pozici zobrazit jiné číslo. Princip je takový, že postupně prostřídá zobrazení čísel pro individuální příslušné anody (pozice). To musí proběhnout dostatečně rychle, aby to lidské oko nebylo schopné rozpoznat jako blikání. Já si vybrala obnovovací frekvenci celého displaye 100 Hz, což je 2,5 ms na jednu pozici. Toto je zobrazeno v Obrázku 3.



Obrázek 3 – Obnovovací systém

LED diody

Další součást vývojového kitu je 8 LED, tedy elektroluminescenčních diod. Pokud se na diodu přivede požadované napětí – rozsvítí se. v digitálním světě tedy na diodu pošleme signál o velikosti '1'.

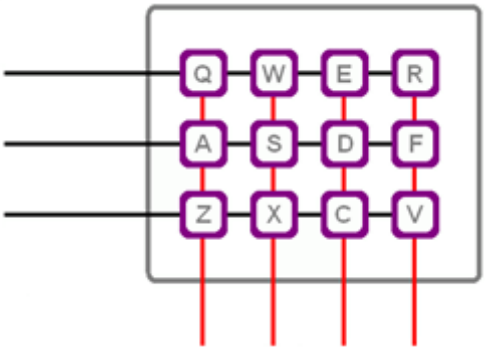
2.1.3 Externí doplňky

Klávesnice

V této práci jsem použila tlačítkovou maticovou klávesnici 4x4 s osmi výstupy (piny). Každý výstup představuje jeden řádek či sloupec viz Obrázek 5. Princip čtení z klávesnice spočívá v rozdělení pinů na 4 vstupy (např. řádky) a 4 výstupy (sloupce) a následném posílání impulsů do vstupů. Tyto impulsy se vždy posílají pouze do jednoho řádku najednou a s hodinovým taktem se mění řádek za další.

Obrázek 4 - Maticová klávesnice

Pokud by bylo stisknuté tlačítko, tak by se impulsy zobrazily i na výstupu u příslušného sloupce. Pokud by k tomu došlo, tak by program porovnal sloupec, na kterém se objevily impulsy, s časem vyslání impulsů a určil by přesné tlačítko – pozici propojení sloupce a řádku v matici.



Obrázek 5 – Princip klávesnice

Detektor pohybu (PIR)

Další z použitých nástrojů je detektor pohybu PIR s třemi piny: napájením, zemí a logickým výstupem. Funguje na principu pyroelektrického jevu, a tedy snímá změnu teploty. Samotný prvek se skládá z polovodiče, který je citlivý na infračervené záření.

Když senzor zaznamená již zmiňovanou změnu teploty, tak sepne a na jeho výstupu se objeví logická jednička.

Na desce plošných spojů se spolu s PIR detektorem nachází také dva potenciometry. Jeden slouží k regulaci citlivosti a druhý k regulaci doby odezvy.

Obrázek 6 - PIR detektor

Reproduktor

Malý reproduktor pro mou potřebu slouží jako zdroj alarmu. Ze zařízení vedou dva výstupy, jeden pin je uzemněný a na druhý se přivádí signál o určité frekvenci – záleží na požadovaném tónu.

Je to však pouze reproduktor pro malé výkony, takže výsledný tón slouží jen jako nevýrazná zvuková indikace, vhodná pro ověření správnosti signálu na výstupu. Není tedy vhodný jako plnohodnotná zvuková siréna.

Obrázek 7 - Reproduktor

2.2 Použitý software

2.2.1 Software

Každý výrobce FPGA vytvořil vlastní vývojový program pro programování hradlových polí z důvodu rozdílných technologií každého z nich. Jelikož já používám hardware Spartan-6 od firmy Xilinx, tak jsem si stáhla příslušný program ISE Design Suite. Xilinx nabízí také další vývojové prostředí Vivaldo Design Suite, které se doporučuje pro FPGA s novější a výkonnější technologií.

ISE® Design Suite 14.7

ISE (Integrated Synthesis Environment) tedy prostředí pro integrovanou syntézu je vývojový nástroj, který slouží k vytvoření souboru s popisem celého hradlového pole a pro následnou konfiguraci čipu FPGA. To je možné vytvořit buďto schematickým návrhem pro jednodušší obvody, nebo modulem programovacího jazyka Verilog či VHDL (viz kapitola 2.2.2 Jazyk VHDL).

Takový návrh musí nejdříve projít syntézou, poté implementací a nakonec se vytvoří soubor pro samotnou konfiguraci. U syntézy se pomocí unikátního softwaru analyzuje naprogramovaný modul a vytvoří se příslušné prvky logického obvodu. Ty se dále implementují dle potřeb specifického FPGA – jednotlivé části popisu obvodu se přidělí do logických bloků hradlového pole. Pak už jen zbývá vygenerovat konfigurační soubor.

Digilent Adept

Pro nahrávání vygenerovaného souboru *.vhd na kit slouží program Adept od společnosti Digilent. Je to rychlý a přehledný způsob pro konfiguraci FPGA a celého kitu, proto ho upřednostňuji, i když proces nahrávání je možný i ze samotného vývojového prostředí ISE Design Suite. Tento program také dokáže otestovat správnou funkčnost vývojového kitu připojeného k počítači.

2.2.2 Jazyk VHDL

VHDL je zkratka pro VHSIC (Very High Speed Integrated Circuits) Hardware Description Language, do češtiny lze přeložit jako programovací jazyk k popisu hardware pro velmi rychlé integrované obvody. Jak napovídá název – používá se pro návrh digitálních integrovaných obvodů a také pro jejich simulaci.

Na rozdíl od většiny programovacích jazyků se u VHDL používá paralelní (nikoli sekvenční) programování. To znamená, že všechny příkazy a procesy se provádějí najednou a ne postupně podle toho jak jsou zapsány.

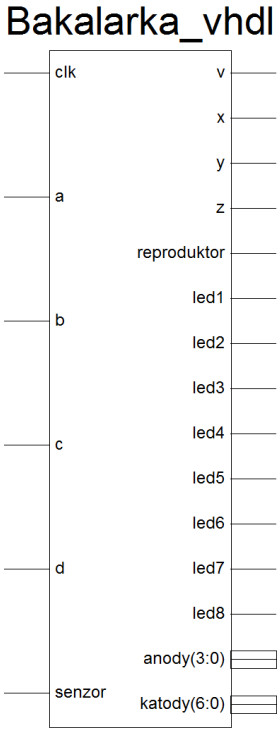
Samotný naprogramovaný modul se skládá ze dvou částí: entita a architektura. V entitě se vypíší všechny signály, se kterými má samotný program pracovat. Přidělí se k nim status portu, jako který mají fungovat, např. IN (vstup), OUT (výstup), nebo třeba INOUT (obousměrný). Architektura se potom věnuje z části deklarování dalších pomocných signálů a proměnných, a dále také samotným procesům a příkazům.

3 Praktická část

3.1 Popis individuální realizace

Zadání jsem se rozhodla interpretovat následujícím způsobem. Celý systém přípravku Nexys 3 s externími periferiemi je nečinný, dokud čidlo PIR nedetekuje pohyb. Poté se začne odpočítávat nastavený čas a na konci se spustí alarm v podobě zvukového výstupu z reproduktoru. Odpočítávání je zřejmé z LED diod, které po vteřině zhasínají až do té doby, kdy odpočet dojde k nule, pak jsou diody všechny zhaslé a spustí se již zmíněný alarm. Během celé této doby je možné psát na klávesnici – psané znaky se vyobrazují na displayi, nebo vykonávají svou specifickou funkci. Pokud se na displayi pomocí klávesnice zobrazí správný autentizační kód – alarm se přeruší. Pokud se kód zadá v průběhu alarmu, tak reproduktor přestane vydávat zvuk. Pokud se zadá v průběhu odpočítávání, tak se celý proces zastaví a pokud se zadá ještě před detekováním pohybu, tak systém do odvolání bude impulzy z PIR ignorovat.

Na klávesnici jsou také naprogramované další funkce kromě čtení číslic. V pravém sloupci jsou také písmena ***A***, ***B***, ***C***, ***D***, kdy zmáčknutí každého z nich představuje spuštění jiného procesu. ***A*** slouží pro opětovnou inicializaci celého procesu, kdy může dojít k nové detekci. Po zmáčknutí ***B*** se hodnota na první pozici displaye uloží jako nový čas pro odpočítávání po detekci. Po zmáčknutí ***C*** se kombinace čísel ze všech pozic displaye uloží jako nový autentizační kód. Písmeno ***D*** slouží k vymazání celého displaye a je tak možné opětovné psaní na klávesnici.



Obrázek 8 - Vstupy a výstupy celého modulu

3.2 Popis jednotlivých bloků VHDL kódu

3.2.1 Dělička impulzů

Ze 100 MHz oscilátoru je pro různé použití potřeba vytvořit nižší kmitočty. Já si zvolila několik děliček fungujících na principu jednoduchých čítačů. Jde tedy o funkci *if*, ze které během jedné smyčky mohou vyjít dva různé výsledky: buďto čítač nenabyl požadované hodnoty a tedy výstup zůstane na logické nule, nebo čítač dosáhne nastavené hodnoty a na výstup se pro tento jeden impuls pošle logická jednička. Poté se také čítač vynuluje a začne nový cyklus.

V ukázce viz Obrázek 9 je nastavená hodnota čítače 100, tedy až původní oscilátor nabyde hodnoty 100 – nový oscilátor za tu dobu udělá pouze jeden impulz. Frekvence nově vzniklého signálu je tím pádem stokrát menší.

```
process(clk)
variable pocet : integer :=0;
begin
if clk'event and clk = '1' then
   if pocet = 100 then
   impuls1 <= '1';
   pocet :=0;
   else
   impuls1 <= '0';
   pocet := pocet+1;
   end if;
end if;
end process;
```

Obrázek 9 - Dělička frekvence

3.2.2 Generování signálu do klávesnice

Princip, na kterém funguje klávesnice, je již vysvětlen v teoretickém úvodu a funkce, kterou je vysílání realizováno vysvětlím zde. Do klávesnice vede 8 vodičů, 4 pro řádky a 4 pro sloupce, nyní nás zajímají 4 řádky, na které jsem se rozhodla připojit 4 výstupy z desky (*v, x, y, z*). Na ty se poté střídavě přivádí logická jednička. Logická jednička se vždy „přesune“ z řádku na řádek s náběžnou hranou *impulsu3*, tedy všechny řádky se prostřídají za 0,16 s. To celé řídí funkce *if*.

3.2.3 Vzorkování signálu

Jelikož mačkání tlačítek na klávesnici způsobovalo záchvěvy a tím pádem falešná zmáčknutí, bylo zapotřebí to ošetřit. U vzorkování jsou důležité vstupy z klávesnice do desky (*a, b, c, d*). Pro každý vstup je zvlášť proměnná *vzorek* (logický vektor), kdy se vektor postupně s vzestupnou hranou hodinového impulzu zaplňuje aktuálními hodnotami ze vstupů na desce. Jakmile je vektor zaplněn požadovaným počtem logických jedniček – tlačítko se považuje za zmáčknuté.

Další roli zde hraje proměnná *vzorek_boolean*, která pouze slouží k určení, zdali je tlačítko zrovna stisknuté či nikoliv (slouží pro posouvání zapisované pozice na displayi).

3.2.4 Čtení z klávesnice

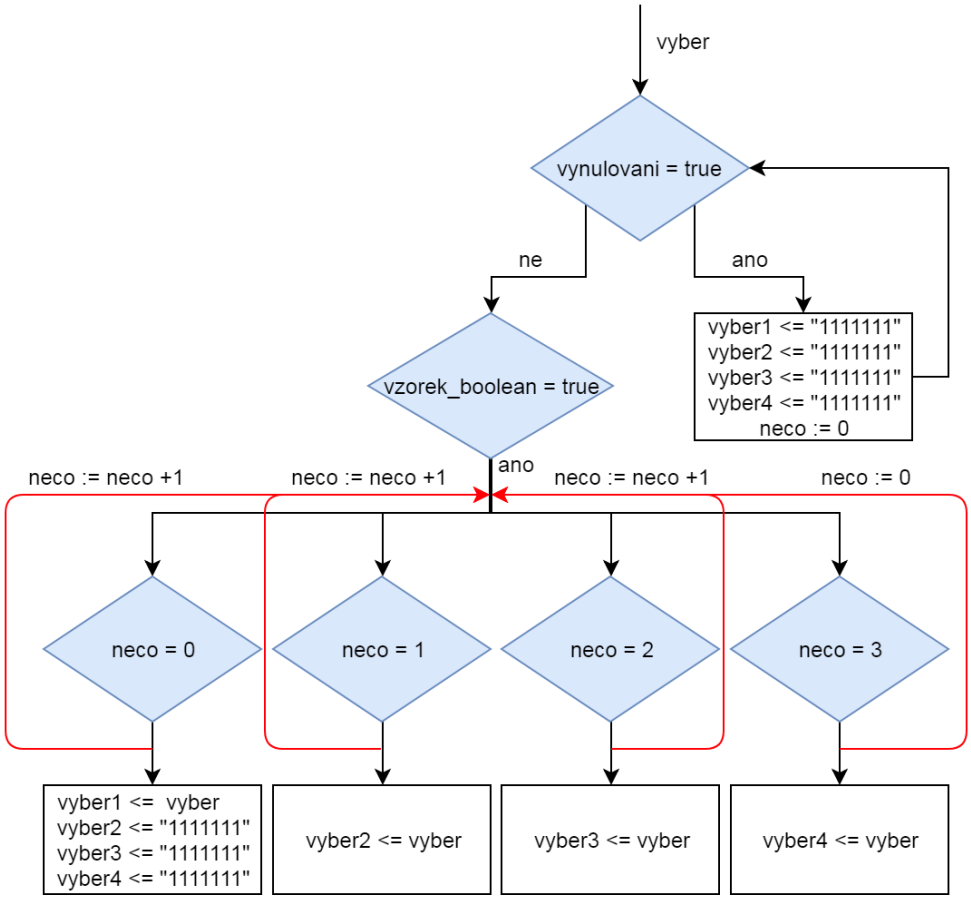
Pro úspěšné snímání správného tlačítka je důležité zjistit, na který řádek byla v tu chvíli nastavena logická jednička a zároveň z jakého sloupce byla logická jednička přečtena. Jeden z procesů tedy vyhodnocuje pomocí funkce *if* 4 možnosti (na který řádek se zrovna vysílalo). Každá z těchto možností má ještě podsekci vyhodnocení vzorků pro sloupce. Jakmile se zaznamená plně zaplněný vektor *vzorek* pro nějaký ze sloupců, tak se zároveň okamžitě ví i řádek, a tedy lze určit místo propojení řádku a sloupce na maticové klávesnici – hodnotu stisknuté klávesy. Podle toho se to také vyhodnotí – příslušné číslo se zapíše do proměnné *volba*.

3.2.5 Přiřazení vybraných čísel ke znaku na displayi

Proměnná *volba* se přiřadí k příslušnému numerickému kódu a uloží se jako *vyber*.

3.2.6 Vybírání pozic

Slouží k postupnému zapisování čísel na display o čtyřech pozicích. Nejdříve funkce *if* zjistí, jestli proměnná *vynulovani* nabývá hodnoty *true*, pokud ano, tak celý proces vynuluje a všechny pozice displaye zhasne. To se stane za předpokladu, že je zmáčknuto tlačítko ***D***. v opačném případě se po zmáčknutí libovolného tlačítka (*vzorek_boolean*) do čítače připočítá jednička a zmáčknuté číslo se zapíše na danou pozici. Při dalším zmáčknutí předchozí hodnota zůstane uložena na dané pozici a aktuální hodnota se zapíše na pozici následující. Pokud jsou již všechny pozice obsazené, tak display tzv. „přeteče“ a číslo se zapíše znovu na první pozici, zatímco zbytek displaye zhasne. Tento proces je znázorněn v Obrázku 10.



Obrázek 10 - Diagram bloku vybírání pozic

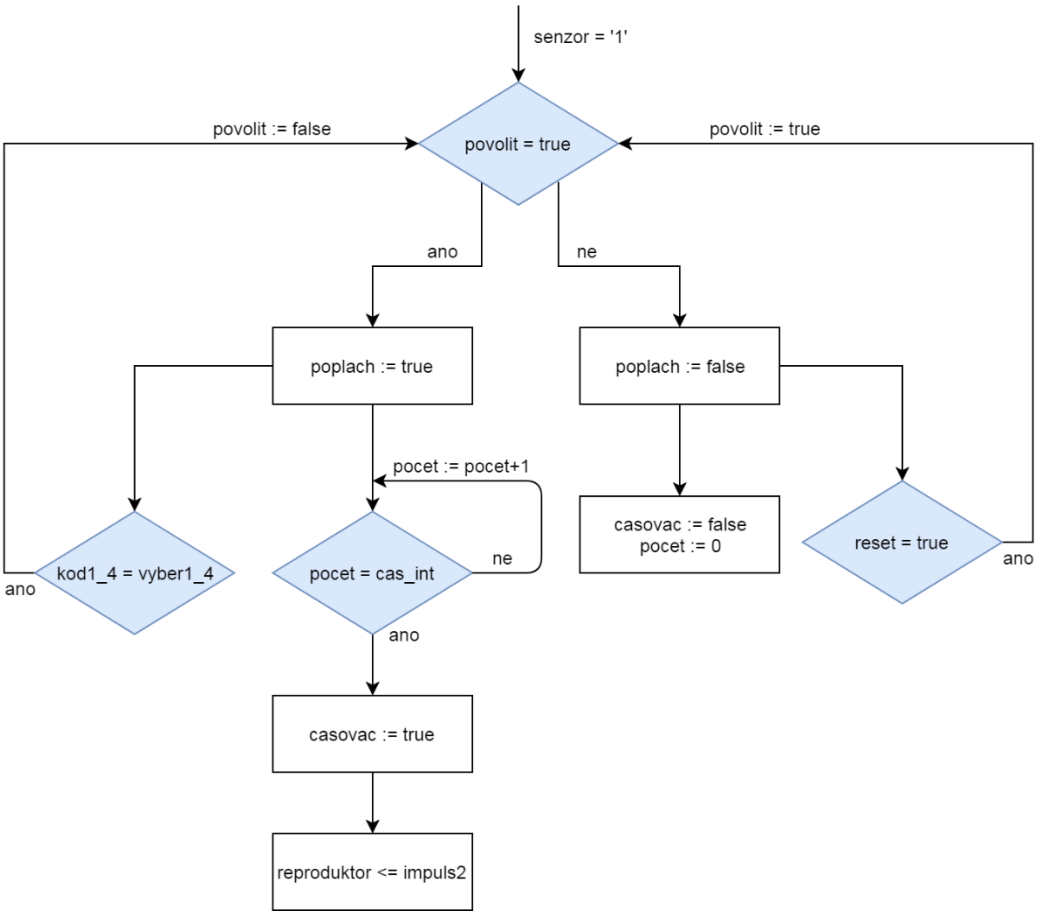
3.2.1 Promítání na display

U promítání na display je vytvořen cyklus, kdy se pokaždé s *impulsem2* vymění zobrazovaná anoda, tedy pozice na displayi. K tomu se pro každou anodu přidá hodnota, která na ní má být zobrazena, tedy příslušná katoda. Do proměnné *katody* se uloží hodnota přečtená z klávesnice, původně jako proměnná *volba*, potom jako *vyber* a v bloku Vybírání pozic přiřazená k proměnné jedinečné pro každou z anod. Tedy k první pozici na displayi bude vždy přiřazena proměnná *vyber1*, k druhé pozici *vyber2* atd.

3.2.2 Senzor + alarm

Tento blok se skládá ze čtyř procesů. První zajišťuje, že pokud je sepnut senzor, tak se spustí vnitřní poplach. Ve druhém procesu se se spuštěním poplachu zapne časovač, který počítá do zadané hodnoty cas_int, která v přepočtu odpovídá hodnotám 1 až 8 sekund. Třetí proces přivádí signál do reproduktoru pod podmínkou, že je spuštěn poplach a že časovač nabyl požadované hodnoty. V posledním procesu se kontroluje, zdali byl na klávesnici zadán správný kód, pokud je tomu tak – poplach se zruší. Diagram funkce bloku je vyobrazen v Obrázku 11.

Po proběhnutí detekce a ověření správným kódem zůstává proměnná *povolit* ve stavu *false* do té doby, než se pomocí tlačítka ***A*** neaktivuje proměnná *reset*. Potom v tomto bloku proměnná *povolit* znovu nabývá hodnoty *true a* přístroj je připraven na novou detekci.



Obrázek 11 - Diagram bloku Senzor + Alarm

3.2.3 Změna časovače spuštění alarmu

Proces v tomto bloku má na starosti provedení změny času, pokud dojde ke zmáčknutí tlačítka ***B***. V bloku Čtení z klávesnice se aktuální zapsaná hodnota ve formě proměnné *volba* zapíše do proměnné *cas_vektor a* právě v tomto bloku tato hodnota dostane formu čísla, které se přiřadí k odpovídající hodnotě *cas_int*. S touto hodnotou dále počítají čítače pro alarm a pro LED.

3.2.4 LED indikátory

Blok pro indikaci diodami se skládá ze dvou procesů. První od chvíle spuštění vnitřního poplachu zajišťuje impulzy po jedné sekundě, s každým impulzem se tedy proměnná *led o* jeden zvětší. Druhý v závislosti na zadané hodnotě *cas_int* rozsvítí odpovídající počet diod a postupně je po sekundě zhasíná až do poslední – to signalizuje spuštění alarmu.

4 Závěr

V této práci jsem naprogramovala VHDL modul, který byl implementován do FPGA na vývojovém kitu Nexys 3. Tento čip ovládá celý systém, který funguje jako zabezpečovací zařízení. Systém se skládá z již zmíněného kitu, na který je připojena externí klávesnice, kam se zadává autentizační kód. Odpovídající znaky mačkaných tlačítek jsou současně zobrazovány na vestavěný sedmisegmentový display. V případě sepnutí senzoru se spustí odpočítávání, které je znázorněno LED diodami. V případě skončení odpočtu (pokud nedojde k interferenci správně zadaným kódem) se spustí alarm v podobně zvuku z reproduktoru.

Systém byl opakovaně testován a funguje dle požadovaného zadání. Navíc jsem přidala například funkce resetu, či možnost změny autentizačního kódu a změnu délky odpočtu při sepnutí detektoru.

Programování této práci mi bylo velmi nápomocné při porozumívání funkčnosti hradlových polí a vytváření návrhů za použití VHDL.

Na přiloženém CD se nacházejí veškeré soubory, které jsem při vypracovávání práce vygenerovala, včetně souboru *.vhd. A dále také elektronická verze této práce.

Použitá literatura

[1] Uživatelská příručka Digilent, https://reference.digilentinc.com/_media/nexys:nexys3:nexys3_rm.pdf

[2] Zdeněk Vašíček (20. 5. 2009), Návody / Popis HW pomocí VHDL, http://merlin.fit.vutbr.cz/FITkit/docs/navody/synth_templates.html

[3] Xilinx, Spartan-6 Family Overview (25. 10. 2011), https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[4] Programovatelné hradlové pole (5 .10. 2017), https://cs.wikipedia.org/wiki/Programovatelné_hradlové_pole

[5] Obrázek 1 – Nexys 3 Reference Manual, https://reference.digilentinc.com/reference/programmable-logic/nexys-3/reference-manual

[6] Obrázek 4 – Arduino 4x4 Maticová tlačítková klávesnice Plastová tlačítka FZ0840, https://laskarduino.cz/ovladaci-prvky/132005-4x4-maticova-tlacitkova-klavesnice-plastova-tlacitka-fz0840.html

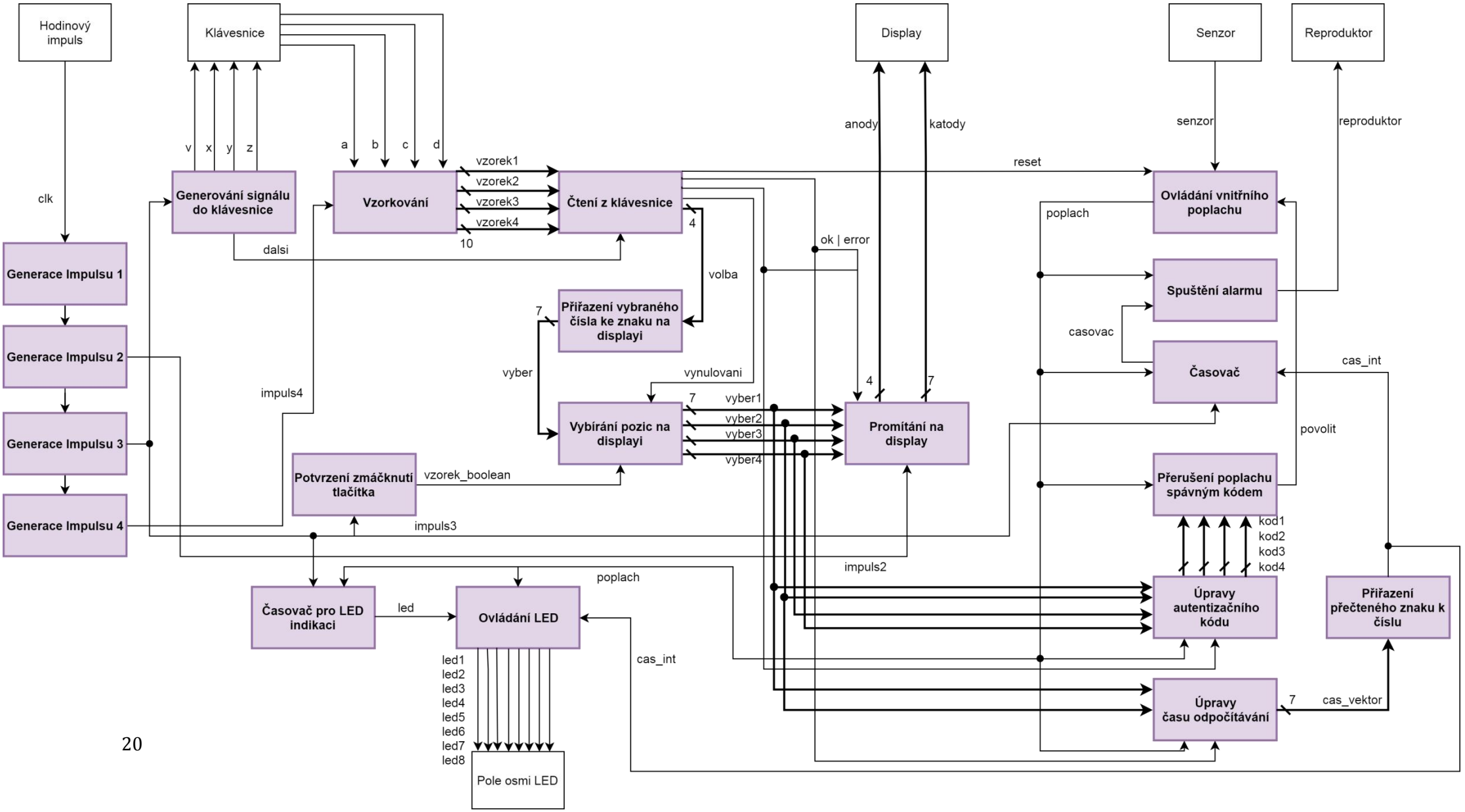
[7] Obrázek 5 – Raspberry Pi, https://astromik.org/raspi/25.htm

[8] Obrázek 6 – PIR modul miniaturní SB00622A-2, https://www.tipa.eu/cz/pir-modul-miniaturni-sb00622a-2/d-131518

[9] Obrázek 7 – 36CS16FN-50BD VANSONIC, https://www.maritex.com.pl/akustyka/glosniki/vansonic/vec-36cs16fn-50bd.html

Přílohy

Příloha A: Blokové schéma



Příloha B: VHDL modul

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_UNSIGNED.ALL;
4   use IEEE.NUMERIC_STD.ALL;
5
6   entity Bakalarka_vhdl is
7       Port (  clk : in  STD_LOGIC;
8               anody : out STD_LOGIC_VECTOR (3 downto 0);
9               katody : out STD_LOGIC_VECTOR (6 downto 0);
10              a : in  STD_LOGIC;
11              b : in  STD_LOGIC;
12              c : in  STD_LOGIC;
13              d : in  STD_LOGIC;
14              v : out  STD_LOGIC;
15              x : out  STD_LOGIC;
16              y : out  STD_LOGIC;
17              z : out  STD_LOGIC;
18              reproduktor : out STD_LOGIC;
19              senzor : in STD_LOGIC;
20              led1 : out STD_LOGIC := '0';
21              led2 : out STD_LOGIC := '0';
22              led3 : out STD_LOGIC := '0';
23              led4 : out STD_LOGIC := '0';
24              led5 : out STD_LOGIC := '0';
25              led6 : out STD_LOGIC := '0';
26              led7 : out STD_LOGIC := '0';
27              led8 : out STD_LOGIC := '0');
28  end Bakalarka_vhdl;
29
30  architecture Behavioral of Bakalarka_vhdl is
31
32  signal impuls1 : STD_LOGIC;
33  signal impuls2 : STD_LOGIC;
34  signal impuls3 : STD_LOGIC;
35  signal impuls4 : STD_LOGIC;
36  signal vyber : STD_LOGIC_VECTOR (6 downto 0);
37  signal vyber1 : STD_LOGIC_VECTOR (6 downto 0):="1111111";
38  signal vyber2 : STD_LOGIC_VECTOR (6 downto 0):="1111111";
39  signal vyber3 : STD_LOGIC_VECTOR (6 downto 0):="1111111";
40  signal vyber4 : STD_LOGIC_VECTOR (6 downto 0):="1111111";
41  signal kod1 : STD_LOGIC_VECTOR (6 downto 0):="1111001";
42  signal kod2 : STD_LOGIC_VECTOR (6 downto 0):="0100100";
43  signal kod3 : STD_LOGIC_VECTOR (6 downto 0):="0110000";
44  signal kod4 : STD_LOGIC_VECTOR (6 downto 0):="0011001";
45  signal cas_vektor : STD_LOGIC_VECTOR (3 downto 0):= "0101";
46  signal volba : STD_LOGIC_VECTOR (3 downto 0):="1111";
47  signal vzorek1 : STD_LOGIC_VECTOR (9 downto 0);
48  signal vzorek2 : STD_LOGIC_VECTOR (9 downto 0);
49  signal vzorek3 : STD_LOGIC_VECTOR (9 downto 0);
50  signal vzorek4 : STD_LOGIC_VECTOR (9 downto 0);
51  shared variable dalsi : integer range 0 to 3 :=0;
52  shared variable vzorek_boolean : boolean:= false;
53  shared variable vynulovani : boolean := false;
54  shared variable povolit  : boolean := true;
55  shared variable poplach : boolean := false;
56  shared variable casovac : boolean := false;
57  shared variable reset : boolean := false;
58  shared variable led : integer range 0 to 9 :=0;
```

```
59  shared variable cas_int : integer range 25 to 200 :=125;
60  shared variable error : boolean := false;
61  shared variable ok : boolean := false;
62
63
64  begin
65
66  ----------------------------------------------------Dělička impulzů
67
68  process(clk)
69  variable pocet : integer range 0 to 2500 :=0;
70  begin
71  if clk'event and clk = '1' then
72      if pocet = 2500 then
73      impuls1 <= '1';
74      pocet :=0;
75      else
76      impuls1 <= '0';
77      pocet := pocet+1;
78      end if;
79  end if;
80  end process;
81
82  process(impuls1)
83  variable pocet : integer range 0 to 100 :=0;
84  begin
85  if impuls1'event and impuls1 = '1' then
86      if pocet = 100 then
87      impuls2 <= '1';
88      pocet :=0;
89      else
90      impuls2 <= '0';
91      pocet := pocet+1;
92      end if;
93  end if;
94  end process;
95
96  process(impuls2)
97  variable pocet : integer range 0 to 16 :=0;
98  begin
99  if impuls2'event and impuls2 = '1' then
100     if pocet = 16   then
101     impuls3 <= '1';
102     pocet :=0;
103     else
104     impuls3 <= '0';
105     pocet := pocet+1;
106     end if;
107 end if;
108 end process;
109
110 process(impuls3)
111 variable pocet : integer range 0 to 2 :=0;
112 begin
113 if impuls3'event and impuls3 = '1' then
114     if pocet = 2    then
115     impuls4 <= '1';
116     pocet :=0;
117     else
118     impuls4 <= '0';
119     pocet := pocet+1;
```

```
120         end if;
121     end if;
122     end process;
123 
124     ----------------------------------Generování signálu do klávesnice
125 
126     process(impuls3)
127     begin
128     if impuls3'event and impuls3 = '1' then
129      if dalsi = 0 then
130         z <= '0';
131         v <= '1';
132         dalsi := dalsi + 1;
133      elsif dalsi = 1 then
134         v <= '0';
135         x <= '1';
136         dalsi := dalsi + 1;
137      elsif dalsi = 2 then
138         x <= '0';
139         y <= '1';
140         dalsi := dalsi + 1;
141      elsif dalsi = 3 then
142         y <= '0';
143         z <= '1';
144         dalsi :=0;
145      end if;
146     end if;
147     end process;
148 
149     ------------------------------------------------Vzorkování signálu
150 
151     process(clk)
152     begin
153 
154     if rising_edge(clk) then
155         if impuls4 = '1' then
156                     vzorek1(9 downto 1) <= vzorek1(8 downto 0);
157                     vzorek1(0) <= a;
158                     vzorek2(9 downto 1) <= vzorek2(8 downto 0);
159                     vzorek2(0) <= b;
160                     vzorek3(9 downto 1) <= vzorek3(8 downto 0);
161                     vzorek3(0) <= c;
162                     vzorek4(9 downto 1) <= vzorek4(8 downto 0);
163                     vzorek4(0) <= d;
164         end if;
165     end if;
166     end process;
167 
168     process(clk)
169     begin
170     if rising_edge(clk) then
171     if impuls3 = '1' then
172     if vzorek1 = "111111111" then
173         vzorek_boolean := true;
174     elsif vzorek2 = "111111111" then
175         vzorek_boolean := true;
176     elsif vzorek3 = "111111111" then
177         vzorek_boolean := true;
178     elsif vzorek4 = "111111111" then
179         vzorek_boolean := true;
180     else
```

```
181         vzorek_boolean := false;
182     end if;
183     end if;
184     end if;
185     end process;
186 
187     ------------------------------------------------Čtení z klávesnice
188 
189     process(clk,a,b,c,d)
190     begin
191     if rising_edge(clk) then
192     reset := false;
193     vynulovani := false;
194     if dalsi = 1 then
195         if vzorek1 = "111111111" then
196             volba <= "0001";
197         elsif vzorek2 = "111111111" then
198             volba <= "0010";
199         elsif vzorek3 = "111111111" then
200             volba <= "0011";
201         elsif vzorek4 = "111111111" then
202             vynulovani := true;
203             reset := true;
204     --> (Reset celého programu - možná nová detekce)
205         end if;
206     elsif dalsi = 2 then
207         if vzorek1 = "111111111" then
208             volba <= "0100";
209         elsif vzorek2 = "111111111" then
210             volba <= "0101";
211         elsif vzorek3 = "111111111" then
212             volba <= "0110";
213         elsif vzorek4 = "111111111" and poplach = false then
214             if vyber1 /= "1111111" and vyber2 = "1111111" then
215             cas_vektor <= volba;
216             ok := true;
217             else
218             error := true;
219     --> (Změna časovače spouštění alarmu)
220             end if;
221         end if;
222     elsif dalsi = 3 then
223         if vzorek1 = "111111111" then
224             volba <= "0111";
225         elsif vzorek2 = "111111111" then
226             volba <= "1000";
227         elsif vzorek3 = "111111111" then
228             volba <= "1001";
229         elsif vzorek4 = "111111111" then
230             if vyber4 /= "1111111" and poplach = false then
231                 kod1 <= vyber1;
232                 kod2 <= vyber2;
233                 kod3 <= vyber3;
234                 kod4 <= vyber4;
235             ok := true;
236             else
237             error := true;
238             end if;
239     --> Přepsání stávajícího kódu na kód z displaye
240         end if;
241     elsif dalsi = 0 then
```

```
242         if vzorek2 = "111111111" then
243             volba <= "0000";
244         elsif vzorek3 = "111111111" then
245             volba <= "1010";
246         elsif vzorek1 = "111111111" then
247             volba <= "1010";
248         elsif vzorek4 = "111111111" then
249             vynulovani := true;
250             error := false;
251             ok := false;
252     --> (Vymazání displaye)
253         end if;
254     end if;
255     end if;
256     end process;
257 
258     --------------------Přiřazení vybraných čísel ke znaku na displayi
259 
260     with volba select
261        vyber <=     "1000000" when "0000",
262                     "1111001" when "0001",
263                     "0100100" when "0010",
264                     "0110000" when "0011",
265                     "0011001" when "0100",
266                     "0010010" when "0101",
267                     "0000010" when "0110",
268                     "1111000" when "0111",
269                     "0000000" when "1000",
270                     "0010000" when "1001",
271                     "0000110" when "1010",
272                     "1111111" when others;
273 
274     ------------------------------------------------------Vybírání pozic
275 
276     process(clk)
277     variable neco : integer range 0 to 3 :=0;
278 
279     begin
280     if rising_edge(clk) then
281     if vynulovani = true then
282         neco := 0;
283         vyber1 <= "1111111";
284         vyber2 <= "1111111";
285         vyber3 <= "1111111";
286         vyber4 <= "1111111";
287     else
288      if vzorek_boolean = true then
289       if neco = 0 then
290         vyber1 <= vyber;
291         vyber2 <= "1111111";
292         vyber3 <= "1111111";
293         vyber4 <= "1111111";
294         neco := neco + 1;
295       elsif neco = 1 then
296         vyber2 <= vyber;
297         neco := neco + 1;
298       elsif neco = 2 then
299         vyber3 <= vyber;
300         neco := neco + 1;
301       elsif neco = 3 then
302         vyber4 <= vyber;
```

```
303         neco :=0;
304       end if;
305     end if;
306     end if;
307     end if;
308     end process;
309
310     ------------------------------------------------Promítání na display
311
312     process(impuls2)
313     variable dalsi2 : integer range 0 to 3 :=0;
314     begin
315     if impuls2'event and impuls2 = '1' then
316     if error = false and ok = false then
317      if dalsi2 = 0 then
318         anody <= "0111";
319         katody <= vyber1;
320         dalsi2 := dalsi2 + 1;
321      elsif dalsi2 = 1 then
322         anody <= "1011";
323         katody <= vyber2;
324         dalsi2 := dalsi2 + 1;
325      elsif dalsi2 = 2 then
326         anody <= "1101";
327         katody <= vyber3;
328         dalsi2 := dalsi2 + 1;
329      elsif dalsi2 = 3 then
330         anody <= "1110";
331         katody <= vyber4;
332         dalsi2 :=0;
333      end if;
334     elsif error = true and ok = false then
335       if dalsi2 = 0 then
336         anody <= "0111";
337         katody <= "0000110";
338         dalsi2 := dalsi2 + 1;
339      elsif dalsi2 = 1 then
340         anody <= "1011";
341         katody <= "0101111";
342         dalsi2 := dalsi2 + 1;
343      elsif dalsi2 = 2 then
344         anody <= "1101";
345         katody <= "0100011";
346         dalsi2 := dalsi2 + 1;
347      elsif dalsi2 = 3 then
348         anody <= "1110";
349         katody <= "0101111";
350         dalsi2 :=0;
351      end if;
352     elsif ok = true and error = false then
353       if dalsi2 = 0 then
354         anody <= "0111";
355         katody <= "0001000";
356         dalsi2 := dalsi2 + 1;
357      elsif dalsi2 = 1 then
358         anody <= "1011";
359         katody <= "0101011";
360         dalsi2 := dalsi2 + 1;
361      elsif dalsi2 = 2 then
362         anody <= "1101";
363         katody <= "0100011";
```

```
364         dalsi2 := dalsi2 + 1;
365      elsif dalsi2 = 3 then
366         anody <= "1110";
367         katody <= "1111111";
368         dalsi2 :=0;
369      end if;
370    end if;
371    end if;
372    end process;
373
374    ----------------------------------------------------Senzor + alarm
375
376    process(clk)
377    begin
378    if clk'event and clk = '1' then
379        if senzor = '1' and povolit = true then
380        poplach := true;
381        elsif povolit = false then
382        poplach := false;
383        end if;
384    end if;
385    end process;
386
387    process(clk)
388    begin
389    if clk'event and clk = '1' then
390        if poplach = true and casovac = true then
391        reproduktor <= impuls2;
392        end if;
393    end if;
394    end process;
395
396    process(impuls3)
397    variable pocet : integer range 0 to 200 :=0;
398    begin
399    if impuls3'event and impuls3 = '1' then
400        if poplach = true then
401            if pocet = cas_int then
402            casovac := true;
403            else
404            pocet := pocet+1;
405            end if;
406        elsif poplach = false then
407        casovac := false;
408        pocet := 0;
409        end if;
410    end if;
411    end process;
412
413    process(clk)
414    begin
415    if clk'event and clk = '1' then
416    if poplach = true then
417        if vyber1 = kod1 then
418            if vyber2 = kod2 then
419                if vyber3 = kod3 then
420                    if vyber4 = kod4 then
421                    povolit := false;
422                    end if;
423                end if;
424            end if;
```

```
425         end if;
426     else
427         if reset = true then
428         povolit := true;
429         end if;
430     end if;
431     end if;
432     end process;
433 
434     ------------------------------------------------LED indikátory
435 
436     process(impuls3)
437     variable pocet : integer range 0 to 40 :=0;
438     begin
439     if impuls3'event and impuls3 = '1' then
440         if poplach = true then
441             if pocet = 25 and led <= 7 then
442             led := led + 1;
443             pocet := 0;
444             else
445             pocet := pocet+1;
446             end if;
447         elsif poplach = false then
448         pocet := 0;
449         led := 0;
450         end if;
451     end if;
452     end process;
453 
454     process(clk)
455     begin
456     if clk'event and clk = '1' then
457     if poplach = true then
458      if cas_int = 25 then
459         if led = 0 then
460             led8 <= '1';
461         elsif led = 1 then
462             led8 <= '0';
463         end if;
464      elsif cas_int = 50 then
465         if led = 0 then
466             led7 <= '1';
467             led8 <= '1';
468         elsif led = 1 then
469             led7 <= '0';
470         elsif led = 2 then
471             led8 <= '0';
472         end if;
473      elsif cas_int = 75 then
474         if led = 0 then
475             led6 <= '1';
476             led7 <= '1';
477             led8 <= '1';
478         elsif led = 1 then
479             led6 <= '0';
480         elsif led = 2 then
481             led7 <= '0';
482         elsif led = 3 then
483             led8 <= '0';
484         end if;
485      elsif cas_int = 100 then
```

```
486         if led = 0 then
487             led5 <= '1';
488             led6 <= '1';
489             led7 <= '1';
490             led8 <= '1';
491         elsif led = 1 then
492             led5 <= '0';
493         elsif led = 2 then
494             led6 <= '0';
495         elsif led = 3 then
496             led7 <= '0';
497         elsif led = 4 then
498             led8 <= '0';
499         end if;
500     elsif cas_int = 125 then
501         if led = 0 then
502             led4 <= '1';
503             led5 <= '1';
504             led6 <= '1';
505             led7 <= '1';
506             led8 <= '1';
507         elsif led = 1 then
508             led4 <= '0';
509         elsif led = 2 then
510             led5 <= '0';
511         elsif led = 3 then
512             led6 <= '0';
513         elsif led = 4 then
514             led7 <= '0';
515         elsif led = 5 then
516             led8 <= '0';
517         end if;
518     elsif cas_int = 150 then
519         if led = 0 then
520             led3 <= '1';
521             led4 <= '1';
522             led5 <= '1';
523             led6 <= '1';
524             led7 <= '1';
525             led8 <= '1';
526         elsif led = 1 then
527             led3 <= '0';
528         elsif led = 2 then
529             led4 <= '0';
530         elsif led = 3 then
531             led5 <= '0';
532         elsif led = 4 then
533             led6 <= '0';
534         elsif led = 5 then
535             led7 <= '0';
536         elsif led = 6 then
537             led8 <= '0';
538         end if;
539     elsif cas_int = 175 then
540         if led = 0 then
541             led2 <= '1';
542             led3 <= '1';
543             led4 <= '1';
544             led5 <= '1';
545             led6 <= '1';
546             led7 <= '1';
```

```
547             led8 <= '1';
548         elsif led = 1 then
549             led2 <= '0';
550         elsif led = 2 then
551             led3 <= '0';
552         elsif led = 3 then
553             led4 <= '0';
554         elsif led = 4 then
555             led5 <= '0';
556         elsif led = 5 then
557             led6 <= '0';
558         elsif led = 6 then
559             led7 <= '0';
560         elsif led = 7 then
561             led8 <= '0';
562         end if;
563      elsif cas_int = 200 then
564         if led = 0 then
565             led1 <= '1';
566             led2 <= '1';
567             led3 <= '1';
568             led4 <= '1';
569             led5 <= '1';
570             led6 <= '1';
571             led7 <= '1';
572             led8 <= '1';
573         elsif led = 1 then
574             led1 <= '0';
575         elsif led = 2 then
576             led2 <= '0';
577         elsif led = 3 then
578             led3 <= '0';
579         elsif led = 4 then
580             led4 <= '0';
581         elsif led = 5 then
582             led5 <= '0';
583         elsif led = 6 then
584             led6 <= '0';
585         elsif led = 7 then
586             led7 <= '0';
587         elsif led = 8 then
588             led8 <= '0';
589         end if;
590      end if;
591    elsif poplach = false and reset = true then
592         led1 <= '0';
593         led2 <= '0';
594         led3 <= '0';
595         led4 <= '0';
596         led5 <= '0';
597         led6 <= '0';
598         led7 <= '0';
599         led8 <= '0';
600    end if;
601    end if;
602    end process;
603
604    -----------------------------------Změna časovače spuštění alarmu
605
606    process(clk)
607    begin
```

```
608   if rising_edge(clk) then
609   case cas_vektor is
610       when "0001" => cas_int := 25;
611       when "0010" => cas_int := 50;
612       when "0011" => cas_int := 75;
613       when "0100" => cas_int := 100;
614       when "0101" => cas_int := 125;
615       when "0110" => cas_int := 150;
616       when "0111" => cas_int := 175;
617       when others => cas_int := 200;
618   end case;
619   end if;
620   end process;
621   end Behavioral;
```