



Střední průmyslová škola strojní
a elektrotechnická a Vyšší odborná škola,
Liberec 1, Masarykova 3

TUX-MAN PRO FPGA

Maturitní práce

Autor	Martin Přívozník
Obor	Informační technologie
Vedoucí práce	Ing. Vladimír Prokeš
Konzultant práce	Ing. Petr Socha
Školní rok	2019/2020

Anotace (Resumé)

Tématem práce je návrh arkádové hry na RTL úrovni a její implementace na programovatelném hradlovém poli, tj. FPGA. Uživatelský vstup je zajištěn pomocí PS/2 klávesnice a výstup prostřednictvím VGA. K implementaci je použit jazyk VHDL.

Klíčová slova

RTL, programovatelné hradlové pole, FPGA, PS/2, VGA, VHDL

Summary

The topic of this thesis is designing an arcade game on an RTL level and its implementation on a programmable gate array, i.e. FPGA. User input is provided by PS/2 keyboard and output is shown using VGA. Language called VHDL is used for implementation.

Keywords

RTL, programmable gate array, FPGA, PS/2, VGA, VHDL

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní/ročníkovou práci vypracoval(a) sám(a) a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 13.03.2020

.....

Martin Přívozník

Obsah

Úvod	1
1 Analýza	2
1.1 Číslicový návrh	2
1.1.1 Kombinační obvody	2
1.1.2 Sekvenční obvody	8
1.1.3 Synchronní návrh	12
1.1.4 Jazyk VHDL	14
1.2 Programovatelná hradlová pole FPGA	18
1.2.1 Dostupné prostředky	18
1.2.2 Logická syntéza	19
1.2.3 Vývojová deska Digilent Basys 2	20
1.2.4 Vývojové prostředí Xilinx ISE	21
1.3 Rozhraní	21
1.3.1 7-segmentový displej	21
1.3.2 PS/2	23
1.3.3 VGA	23
1.4 Hra Pac-Man	25
2 Návrh hry	26
2.1 Specifikace hry	26
2.1.1 Cíl a chování hry	26
2.1.2 Herní mapa	27
2.1.3 Postavy a jejich chování	27
2.2 Periferie	27
2.2.1 Sedmsegmentový displej	27
2.2.2 PS/2	28
2.2.3 VGA	30
2.3 Logika hry	30
2.3.1 Herní mapa jako mřížka	32
2.3.2 Ovládání postavy	32
2.3.3 Pohyb duchů	32
2.4 Textury a propsání na výstup	32
2.5 Kompletní propojení	32

3	Implementace	33
3.1	Modul pro čtení z PS/2 klávesnice	33
3.1.1	Zpracování vstupu	33
3.1.2	Generování výstupu	33
3.2	Modul pro výstup na VGA monitor	33
3.2.1	Časování VGA	34
3.2.2	Propsání textur na monitor	34
3.3	Herní logika	34
3.3.1	Herní mapa	34
3.3.2	Ovládání postav	34
3.3.3	Cíle hry	34
4	Testování	35
	Závěr	36

Seznam obrázků

1.1	Schéma poloviční sčítačky a příklad jejího značení ve schématu.	6
1.2	Schéma kompletní sčítačky a příklad jejího značení ve schématu.	6
1.3	Schéma komparátoru a příklad jeho značení ve schématu.	7
1.4	Schématická značka multiplexoru.	7
1.5	Huffmanův model sekvenčního obvodu.	8
1.6	Schéma RS klopného obvodu s využitím logických hradel NOR.	8
1.7	Schéma D klopného obvodu.	9
1.8	Příklad značení D klopného obvodu ve schématu.	10
1.9	Časový diagram D klopného obvodu.	10
1.10	Příklad schématu dvoubitového čítače reagujícího na náběžnou hranu.	11
1.11	Příklad schématické značky dvoubitového čítače reagujícího na náběžnou hranu.	11
1.12	Příklad stavového diagramu konečného stavového automatu.	12
1.13	Příklad stavového diagramu (typ Moore) čítače modulo 4 se vstupem count enable.	13
1.14	Příklad návrhu synchronizéru.	14
1.15	Příklad zakreslení LUT.	19
1.16	Příklad základního bloku FPGA.	19
1.17	Digilent Basys 2.	20
1.18	Základní pohled na Xilinx ISE Project Navigator.	22
1.19	Sedmsegmentový displej.	22
1.20	Příklad zapojení segmentů na sedmsegmentovém displeji.	22
1.21	Časový diagram PS/2 přenosu.	23
1.22	Časový diagram VGA.	24
2.1	Čítač s vyvedeným osmnáctým a devatenáctým bitem.	28
2.2	Připojení sedmsegmentového displeje na čítač.	29
2.3	Značení modulu pro ovládání sedmsegmentového displeje.	29
2.4	Návrh obvodu pro zpracování příchozích dat z klávesnice PS/2.	30
2.5	Návrh obvodu pro detekci speciálních kláves a aktivity.	31
2.6	Stavový diagram FSM pro indikaci zda je klávesa aktuální, speciální, či se jedná o puštění klávesy.	31

Seznam tabulek

1.1	Axiomy a vztahy Booleovy algebry.[1]	3
1.2	Základní pravidla Booleovy algebry.[2]	3
1.3	Pravdivostní tabulka.	3
1.4	Tabulka nejpoužívanějších základních logických funkcí.	4
1.5	Schématické značky některých nejpoužívanějších základních logických funkcí (americká norma ANSI).	5
1.6	Pravdivostní tabulka poloviční sčítáčky.	6
1.7	Pravdivostní tabulka RS klopného obvodu s využitím NOR.	9
1.8	Pravdivostní tabulka D klopného obvodu.	9

Úvod

Cílem práce je navrhnout a implementovat číslicový logický obvod realizující arkádovou hru. Číslicový obvod bude umět zpracovat příchozí signál z PS/2 klávesnice, obsluhovat VGA monitor a řídit chod hry. K implementaci mého návrhu využiji přípravek Basys 2 od firmy Digilent s vestavěným FPGA Spartan-E3. Důvodem k využití tohoto přípravku jsou vestavěné porty PS/2 a VGA a dostatečné splnění požadavků pro implementaci. Motivací ke psaní této práce je obsáhlá teorie potřebná pro návrh a implementaci obvodu, jíž se bude zaobírat značná část práce. Výstup této práce nebude tedy pouze funkční arkádová hra, ale také seznámení s problematikou řešení pomocí číslicových obvodů a průvod tím, jak může probíhat návrh komplexnějšího číslicového obvodu vč. implementace na vybrané FPGA.

Kapitola 1

Analýza

Tato kapitola obsahuje stručný souhrn znalostí a informací potřebných pro následný návrh a implementaci. V sekci 1.1 je vysvětlen číslicový obvod a postup jeho návrhu. V sekci 1.2 stručně vysvětluji programovatelné hradlové pole a dále vybranou vývojovou desku. Sekce 1.3 se zabývá použitými komunikačními rozhraními, které zajišťují uživatelský vstup a výstup. Na závěr kapitoly, v sekci 1.4 vysvětluji princip a funkčnost hry, jenž je vzorem pro můj návrh.

1.1 Číslicový návrh

V této sekci se venuji tomu, co je číslicový obvod a jak jej navrhnout jak ve schématu, tak v jazyce popisujícím hardware. V podsekci 1.1.1 rozeberu logické funkce, prostředky jejich popisu a realizace pomocí logických hradel. Podsekce 1.1.2 je zaměřena na návrh sekvenčních obvodů a synchronních sekvenčních automatů (FSM), na což naváže podsekce 1.1.3, ve které vysvětluji princip hodinových domén a plně sekvenčního návrhu. V podsekci 1.1.4 stručně ukážu, jak převést schéma číslicového obvodu do kódu v jazyce popisujícím hardware (Hardware Description Language, HDL), v mé případě do jazyka Very High Speed Integrated Circuit Hardware Description Language (VHDL).

1.1.1 Kombinační obvody

Booleovská funkce

Booleovská funkce je funkce N vstupů a M výstupů nad množinou $\{0, 1\}$. V případě, kdy má funkce více jak jeden výstup, lze ji rozdělit na M funkcí s jedním výstupem. Uvážíme-li Booleovu algebru, platí pro operace sčítání a násobení pravidla uvedená v tabulce 1.1. Operace se dvěma vstupními hodnotami nazýváme binární operace. Některé binární operace, přestože často používají stejná značení + a * jako v algebře reálných čísel, mají v Booleové algebře stejnou prioritu a jiný význam (zádná operace nemá přednost) [2]. Pro logický součet a logický součin platí základní pravidla v tabulce 1.2. Příkladem re-

$a + b = b + a$	$a * b = b * a$	(komutativita)
$a + (b + c) = (a + b) + c$	$a * (b * c) = (a * b) * c$	(asociativita)
$a + (b * c) = (a + b) * (a + c)$	$a * (b + c) = (a * b) + (a * c)$	(distributivita)
$a + 0 = a$	$a * 1 = a$	(neutralita: 0 a 1)
$a + \bar{a} = 1$	$a * \bar{a} = 0$	(vlastnosti negace)

Tabulka 1.1: Axiomy a vztahy Booleovy algebry.[1]

$$\begin{array}{lll} \text{de Morgan} & \overline{(a+b)} = \bar{a} * \bar{b} & \overline{(a * b)} = \bar{a} + \bar{b} \\ \text{idempotence} & a + a = a & a * a = a \end{array}$$

Tabulka 1.2: Základní pravidla Booleovy algebry.[2]

prezentace Booleovské funkce je pravdivostní tabulka 1.3, kde in_1 a in_2 jsou vstupní hodnoty a out je výstupní hodnota. Pravdivostní tabulka obsahuje vždy N^2 řádků, aby reprezentovala výstupní hodnotu pro všechny možné kombinace vstupních hodnot. Další možností je Booleovská formule [2]. K vyjádření formule a k popisu booleovské funkce používáme nejčastěji základní funkce uvedené v tabulce 1.4

Kombinační obvod

Kombinační logický obvod, je takový obvod, ve kterém jsou výstupní hodnoty dány pouze aktuální kombinací vstupních proměnných. Neobsahuje žádnou paměť předchozích stavů. Jedinou výjimkou je krátký časový interval, za který logický člen (AND, NAND, OR, NOR ...) vyhodnotí výstup na základě vstupních hodnot. Tento časový interval může být zanedbatelný v případě krátkých datových cest. V případě dlouhé datové cesty může být potřeba na tento časový interval brát zřetel a zvážit optimálnější řešení.

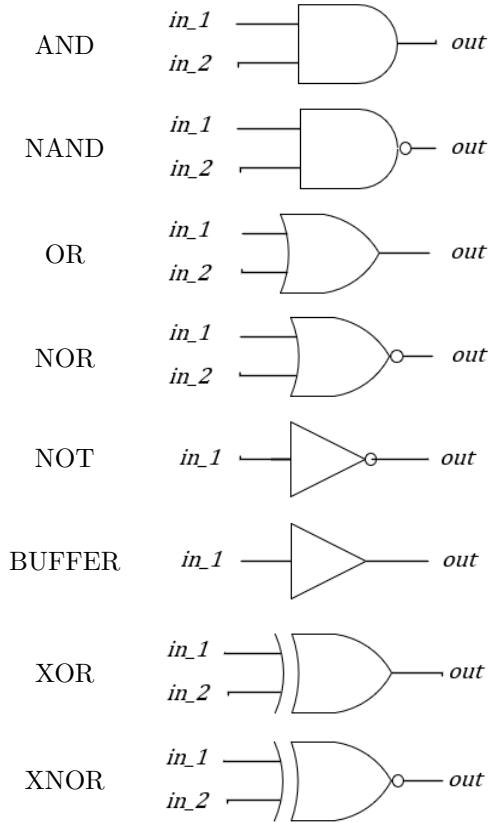
Platí, že u číslicových obvodů každá proměnná v operaci nabývá hodnotu jednoho tzv. bitu. Bit je základní jednotkou dat a může nabývat hodnot 0, nebo 1. Reprezentací číslicového obvodu je schéma číslicového obvodu, kde každá z funkcí je reprezentována tzv. schématickou značkou. Schématické značky mohou být různé, dokud z nich jasně vyplývá, jakou funkci zastupují. Schématické značky jsou propojené signály, které představují jednotlivé bity. Pro minimizaci je možné několik signálů (bitů) zakreslit jedním konektorem, pokud je označený počtem bitů, které reprezentuje. Nejčastěji používané normy značení

in_1	in_2	out
0	0	$f(0, 0)$
0	1	$f(0, 1)$
1	0	$f(1, 0)$
1	1	$f(1, 1)$

Tabulka 1.3: Pravdivostní tabulka.

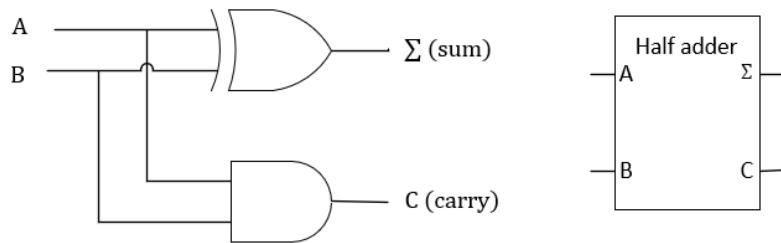
Název	Pravdivostní tabulka			Formule
	in_1	in_2	out	
AND (logický součin)	0	0	0	$out = in_1 * in_2$
	0	1	0	
	1	0	0	
	1	1	1	
NAND (negovaný logický součin)	in_1	in_2	out	$out = \overline{in_1 * in_2}$
	0	0	1	
	0	1	1	
	1	0	1	
	1	1	0	
OR (logický součet)	in_1	in_2	out	$out = in_1 + in_2$
	0	0	0	
	0	1	1	
	1	0	1	
NOR (negovaný logický součet)	in_1	in_2	out	$out = \overline{in_1 + in_2}$
	0	0	1	
	0	1	0	
	1	0	0	
	1	1	0	
NOT (logická negace)	in_1	out		$out = \overline{in_1}$
	0	1		
	1	0		
BUFFER (opakovač)	in_1	out		$out = in_1$
	0	0		
	1	1		
XOR (nonekvivalence)	in_1	in_2	out	$out = in_1 \oplus in_2$
	0	0	0	
	0	1	1	
	1	0	1	
XNOR (ekvivalence)	in_1	in_2	out	$out = \overline{in_1 \oplus in_2}$
	0	0	1	
	0	1	0	
	1	0	0	
	1	1	1	

Tabulka 1.4: Tabulka nejpoužívanějších základních logických funkcí.



Tabulka 1.5: Schématické značky některých nejpoužívanějších základních logických funkcí (americká norma ANSI).

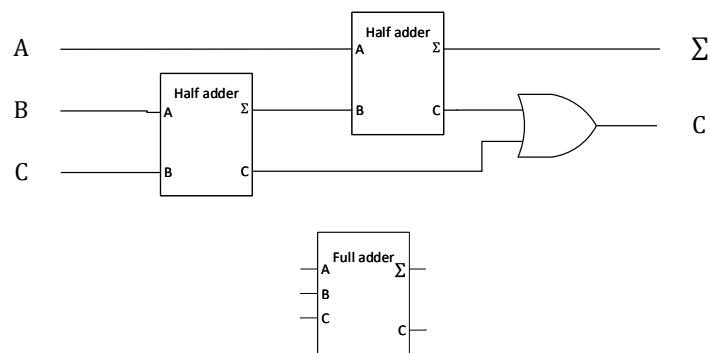
jsou evropská a americká. Příklady schématických značek pro nejpoužívanější logické funkce jsou uvedené v tabulce 1.5. Pro usnadnění práce můžeme využívat logických bloků, které plní danou funkci. Opět platí, že ze značení logických bloků ve schématu musí plně vyplívat, jakou funkci zastupují. Logický blok, který má definovanou funkci může být použit schématu. Při návrhu číslicových obvodů využíváme hierarchie, kde jsou pro každý logický blok popsány vstupy i výstupy a v případě, kdy se nejedná o základní logické bloky, tak je popsána i funkce (formule, pravdivostní tabulka, nebo schéma bloku) a vhodně přiřazena ke schématu. Jedním ze základních logických bloků je poloviční sčítáčka, viz obrázek 1.1. Poloviční sčítáčka umí sečít dvě jednobitová čísla a vygenerovat bit do vyššího rádu (carry) podle pravdivostní tabulky 1.6. Zřetězením dvou polovičních sčítáček a přenesením pomocí funkce OR získáme poté kompletní sčítáčku, viz obrázek 1.2.[2] Kompletní sčítáčka umí sečít jednobitová čísla, vygenerovat bit do vyššího rádu a přjmout bit z nižšího, sčítá tedy tři bity. Sčítá počet jedniček na vstupech.



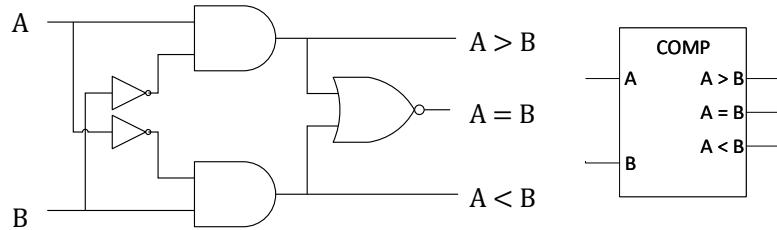
Obrázek 1.1: Schéma poloviční sčítačky a příklad jejího značení ve schématu.

A	B	C	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

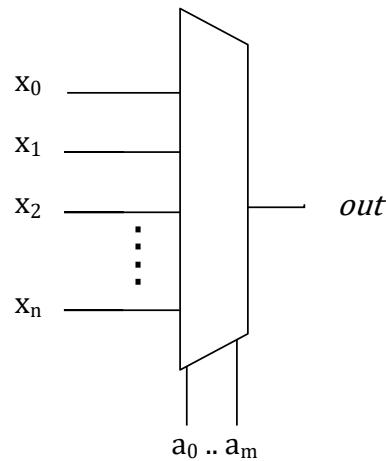
Tabulka 1.6: Pravdivostní tabulka poloviční sčítačky.



Obrázek 1.2: Schéma kompletnej sčítačky a příklad jejího značení ve schématu.



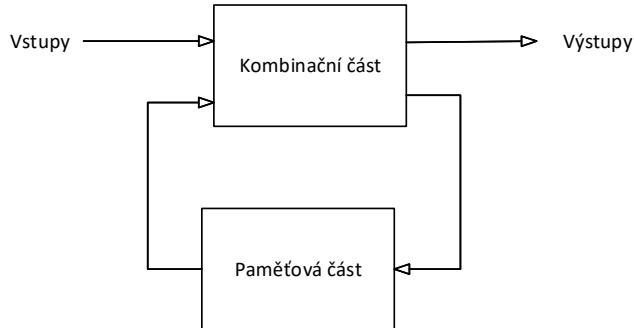
Obrázek 1.3: Schéma komparátoru a příklad jeho značení ve schématu.



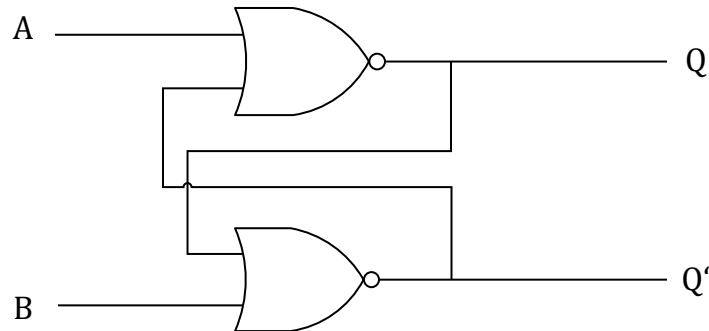
Obrázek 1.4: Schématická značka multiplexoru.

Pro porovnávání dvou hodnot a vyhodnocení jejich nerovnosti používáme tzv. komparátor, viz obrázek 1.3. Dalším důležitým základním logickým blokem je multiplexor, viz obrázek 1.4. Multiplexor na základě řídícího vstupu/řídících vstupů (a) přivádí na výstup (out) jeden ze vstupních signálů (x).

Číslicový obvod lze realizovat například z integrovaných obvodů, v nichž bývají hradla realizována z několika tranzistorů. Logické hodnoty představuje napětí přivedené na obvod. Logická 1 bývá reprezentována napětím kladným a logická 0 napětím nulovým.



Obrázek 1.5: Huffmanův model sekvenčního obvodu.



Obrázek 1.6: Schéma RS klopného obvodu s využitím logických hradel NOR.

1.1.2 Sekvenční obvody

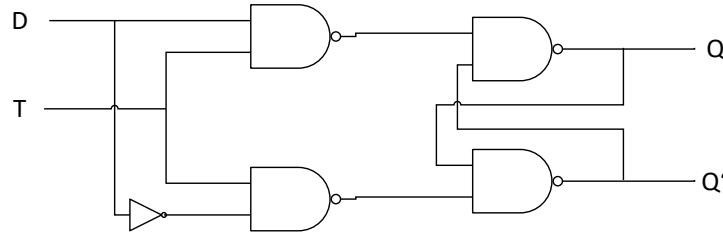
Sekvenční logický obvod

Sekvenční logický obvod je takový obvod, ve kterém výstupní hodnoty nejsou dány pouze aktuální kombinací vstupních proměnných, ale zároveň jeho vnitřním stavem. Sekvenční obvod si tedy musí zapamatovat předchozí hodnoty pomocí paměti, která bývá realizována pomocí zpětné vazby [2]. Sekvenční obvod se dělí na dvě části - kombinační a paměťovou, kde paměťová část je tvořena logickým obvodem, ve kterém bývá zavedena zpětná vazba a kombinační část bývá tvořena kombinačním obvodem. Obecné schéma sekvenčního obvodu je na obrázku 1.5.

Příkladem logického obvodu se zpětnou vazbou může být tzv. RS klopný obvod, viz obrázek 1.6. RS klopný obvod překlápí mezi dvěma mezními hodnotami. Pravdivostní tabulka RS klopného obvodu se může dělit na tři části - zakázaný stav (ZS), překlápecí část a paměť, viz tabulka 1.7. Pokud se tedy

A	B	Q	Q'
0	0	Z.S.	
0	1	0	1
1	0	1	0
1	1	Paměť	

Tabulka 1.7: Pravdivostní tabulka RS klopného obvodu s využitím NOR.



Obrázek 1.7: Schéma D klopného obvodu.

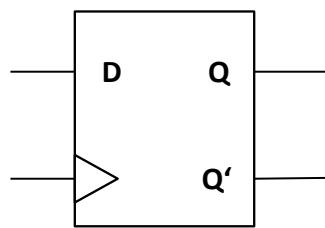
změní vstupní hodnoty z $[0, 1]$, nebo $[1, 0]$ na $[1, 1]$, na výstupu bude předchozí hodnota, dokud neproběhne další změna vstupních hodnot. Jedná se tedy o tzv. hladinový klopný obvod (angl. latch). Častěji využívaným klopným obvodem je tzv. D klopný obvod, viz obrázek 1.7. Výhodou D klopného obvodu je fakt, že narozdíl od RS klopného obvodu nemá zakázaný stav, viz pravdivostní tabulka 1.8. Signál T u D klopného obvodu se dá považovat za tzv. povolovací signál. Při náběžné hraně na T (změna stavu z logické 0 na logickou 1) se na výstup Q zapíše aktuální hodnota na signálu D . V tomto případě se tedy jedná o hranový klopný obvod. Příklad značení D klopného obvodu je na obrázku 1.8. Vstupní signál označený trojúhelníkem je signál T . Trojúhelník značí, že obvod mění výstupní hodnotu v reakci na náběžnou hranu. Tento vstup se jinak nazývá hodinovým vstupem. Pokud má číslicový obvod takový vstup jedná se o sekvenční obvod. V případě hladinového obvodu by ve značení mohl být místo trojúhelníku čtvereček.

Hodnoty na signálech se dají popsat tzv. časovým diagramem. Časový diagram popisuje, jaké logické hodnoty nabývá daný signál v jaký čas. Příklad časového diagramu je na obrázku 1.9.

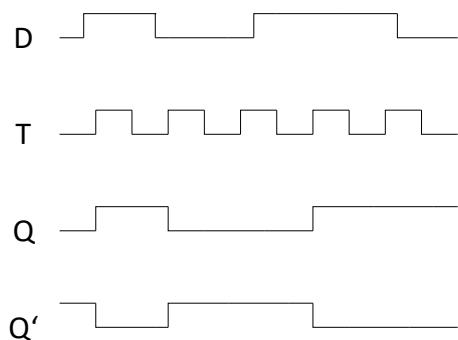
Příkladem sekvenčního logického bloku je čítač. Čítač je takový sekvenční

T	D	Q	Q'
\nearrow	x	x	\bar{x}
Jinak			Paměť

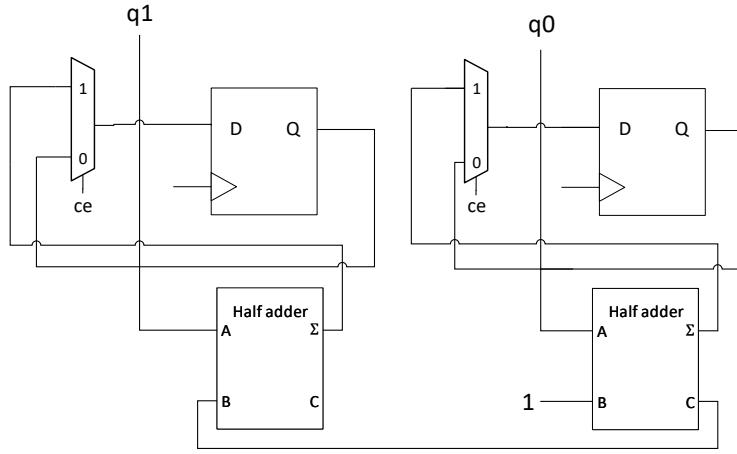
Tabulka 1.8: Pravdivostní tabulka D klopného obvodu.



Obrázek 1.8: Příklad značení D klopného obvodu ve schématu.



Obrázek 1.9: Časový diagram D klopného obvodu.



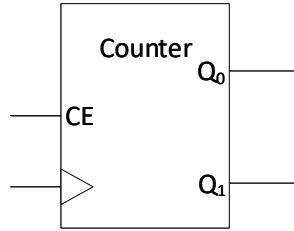
Obrázek 1.10: Příklad schématu dvoubitového čítače reagujícího na náběžnou hranu.

logický obvod, který uchovává v „paměti“ informaci o tom, kolikrát zaznamenal změnu stavu (dle návrhu náběžnou, nebo sestupnou hrana) na hodinovém signálu. Příklad schématu čítače je na obrázku 1.10. Může být ve schématu značen, jako na obrázku 1.11.

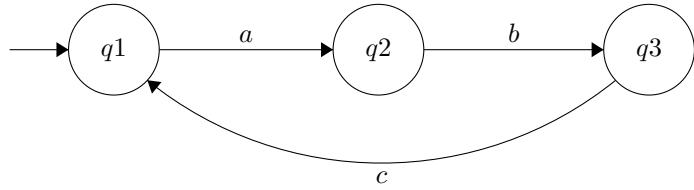
Konečný stavový automat

Konečný stavový automat (angl. Finite State Machine - FSM) M je šestice $M=(Q, T, D, \delta, \lambda, q_0)$, kde [3]:

- Q je konečná množina vnitřních stavů



Obrázek 1.11: Příklad schématické značky dvoubitového čítače reagujícího na náběžnou hranu.



Obrázek 1.12: Příklad stavového diagramu konečného stavového automatu.

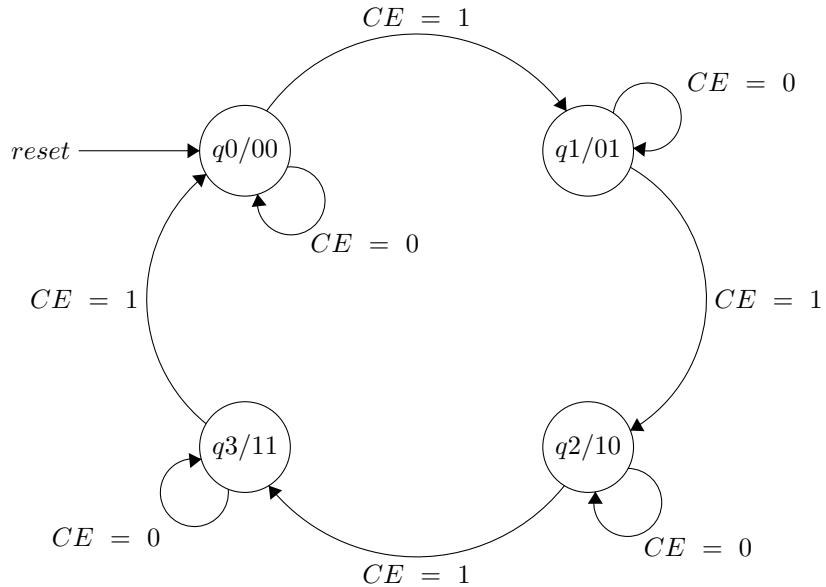
- T je konečná množina vstupních symbolů (signálů)
- D je konečná množina výstupních symbolů (signálů)
- δ je zobrazení z $Q \times T$ do Q (stav x vstup do stav) nazývané přechodová funkce
- λ je zobrazení z $Q \times T$ do D (Mealy, ze stavu a vstupu do výstupu) nebo Q do D (Moore, jen ze stavu do výstupu)
- q_0 je počáteční stav

Má definované stavy, mezi kterými za daných podmínek přepíná a mění svůj výstup. FSM provádí vždy právě jeden přechod s každou následující hranou hodin. Musí popisovat přechodovou funkci (podmínky pro změnu stavu a do kterého), výstupní funkci (jaký signál v danou chvíli udává na výstup) a dále musí mít definovaný počáteční stav. Příklad stavového diagramu FSM je na obrázku 1.12, kde a, b, c jsou podmínky pro změnu stavu a $q1, q2, q3$ jsou stavy FSM. Počáteční stav je značen šipkou, která nevychází z žádného stavu, v tomto případě $q1$. Rozdělujeme dva typy FSM na základě definičního oboru výstupní funkce [2, 3]. Prvním typem je typ Mealy (angl. často input-based), který mění svůj výstup na základě aktuálního stavu a změn na vstupních signálech. Druhý je typ Moore (angl. často state-based), jehož výstupní funkce je závislá pouze na aktuálním stavu (každý stav má definovaný výstup). Příkladem stavového diagramu čítače modulo 4 s povolujícím vstupem je obrázek 1.13

1.1.3 Synchronní návrh

Pravidla návrhu synchronního obvodu

Synchronní obvod je takový obvod, ve kterém všechny sekvenční části obvodu mají na svůj hodinový vstup přivedený stejný signál, aby měnily svůj stav ve stejný čas. Tento signál můžeme nazývat společné hodiny. Společné hodiny mění svůj stav na dané frekvenci (nejčastěji dle krystalu v obvodu). Část obvodu se společným hodinovým signálem nazýváme hodinovou doménou [4]. Pro návrh synchronního obvodu platí několik pravidel, které je nutné dodržet pro zajištění správné funkčnosti. Společné hodiny nesmí být hradlované (signál musí být přiveden přímo k hodinovým vstupům sekvenčních obvodů). V případě

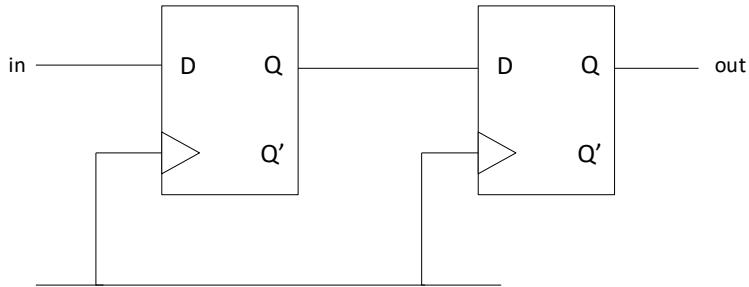


Obrázek 1.13: Příklad stavového diagramu (typ Moore) čítače modulo 4 se vstupem count enable.

hradlování společných hodin může docházet k zákmítům a časovým rezervám v jednotlivých částech obvodu. Dále je nutné, aby obvod obsahoval synchronní reset. Po synchronním resetu obvodu změní stav celý obvod společně a nedojde k časovým nesrovnalostem [4].

Při návrhu se nemusí vždy pracovat pouze s jednou hodinovou doménou. Univerzální řešení přechodu mezi hodinovými doménami je dvouboranové FIFO se dvěma hodinovými vstupy. Pro zamezení nestabilit můžeme využít tzv. synchronizér. Synchronizér se skládá ze dvou D klopných obvodů zapojených v sérii, které zajistí podmínky pro korektní funkci ostatních klopných obvodů v člově časové doméně. Vstupní hodnota D klopného obvodu nesmí být změněna krátce před příchodem náběžné hrany na hodinovém vstupu (tj. předstih). Zároveň nesmí být změněna těsně po příchodu náběžné hrany na hodinovém vstupu (tj. přesah) [4]. V případě porušení těchto podmínek se může stát, že klopný obvod bude mít abnormálně zpožděnou, nebo kolísavou odezvu. Příklad synchronizéru je na obrázku 1.14.

Datové signály jsou řízené řídícími signály. Protokol řídících signálů musí být navržen tak, aby se měnil vždy jen jeden. Příkladem řídících signálů mohou být signály "strobe" nebo "ack". Úkolem tohoto protokolu je indikace, zda jsou data aktuální. V případě tzv. jednosměrného handshake budou data aktuální po dobu, kdy signál "strobe" nabývá logické '1'. U tzv. oboucestného handshake jsou data platná a signál strobe bude nabývat logické '1', dokud nepřijde od protistrany signál ACK.



Obrázek 1.14: Příklad návrhu synchronizéru.

1.1.4 Jazyk VHDL

Charakteristika jazyka VHDL

VHDL (VHSIC (Very-High-Speed Integrated Circuit, česky velmi rychlý integrovaný obvod) Hardware Description Language) je programovací jazyk sloužící pro popis hardware (tj. Hardware Description Language, zkr. HDL, česky jazyk popisující hardware). Používá se pro simulaci a návrh digitálních integrovaných obvodů, např. pro FPGA (Field of Programmable Gate Array, česky programovatelná hradlová pole). VHDL je silně typovaný jazyk a umožňuje popsat obvod na hradlové, RTL (Register Transfer Level, česky přenos na úrovni registrů) i algoritmické úrovni [5].

Datové typy

VHDL je silně typovaný jazyk, při operacích je tedy nutné, aby se typy shodovaly. Jedny z důležitých základních datových typů jsou: bit, integer, std_logic a std_logic_vector.

- Array - pole prvků jiného typu
- Bit - Výčetový typ ('0', '1')
- Integer - Celé číslo
- Std_logic - Popis signálu ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-')
- Std_logic_vector - Pole std_logic
- Signed - Bitový vektor, který nepopisuje stav signálu, ale pouze číselnou hodnotu
- Unsigned - Signed, který může obsahovat pouze kladné hodnoty

Základní konstrukce

Konstrukce jazyka VHDL se dělí na několik částí [5]. Jednou z nich je tzv. entita. Entita popisuje pouze rozhraní, nikoliv chování nebo vnitřní strukturu návrhu. Obsahuje tedy definici vstupních a výstupních portů. Příklad návrhu entity multiplexoru se dvěma datovými vstupy A, B , jedním řídícím vstupem SEL a výstupem Y :

```
entity MULTIPLEXER is
port(  A, B, SEL      : in bit;
        Y           : out bit
      );
end MULTIPLEXER;
```

Další částí je architektura. Architektura obsahuje chování a vnitřní strukturu entity. Architektura tedy musí být definována uvnitř entity a nelze, aby fungovala samostatně. Entita může obsahovat více architektur. Ta se poté dělí na dvě části. Deklarační a sekci paralelních příkazů. Deklarační část se nachází před klíčovým slovem begin a je vyhrazena pro deklaraci signálů, konstant, nebo typů. Součástí sekce paralelních příkazů může být instancování komponent, behaviorální popis (tj. popis chování), nebo tzv. procesy. Sekce paralelních příkazů se nachází v uvnitř architektury (tj. část za klíčovým slovem begin). Příklad architektury výše uvedeného multiplexoru:

```
architecture MUXBODY of MULTIPLEXER is
signal SELNON, ASEL, BSEL : std_logic;
begin
  SELNON <= not SEL;
  ASEL <= A and SELNON;
  BSEL <= B and SEL;
  Y <= ASEL or BSEL;
end MUXBODY;
```

Proměnná označená klíčovým slovem signal značí stejnojmenný signál v obvodu. Klíčové slovo *std_logic* říká, že se jedná o jednobitový signál. V případě vícebitového signálu by bylo využito klíčové slovo *std_logic_vector*. Signál si pamatuje svou hodnotu, dokud není změněna dalším přiřazením. V části paralelních příkazů operátor $<=$ zastupuje operátor přiřazení. Hodnota z pravé strany je tedy po vyhodnocení logické operace přiřazena signálu na levé straně. Všechny logické operace v tomto případě jsou vyhodnocovány paralelně. Tato metoda se většinou využívá pro popis chování kombinačních obvodů. V případě popisu sekvenčního obvodu je nejčastěji využíván tzv. proces. Proces je součástí architektury a přestože popisuje sekvenční obvod, tak běží souběžně. V sekvenčním popisu je možné krom operátorů přiřazení a logických operací využívat

také podmínky, jako je např. *if*, nebo *case*. V případě psaní syntetizovatelného kódu proces obsahuje citlivostní seznam a opět popis chování. Citlivostní seznam je seznam signálů, na který sekvenční obvod reaguje (u sekvenčních obvodů hodinový vstup). Příklad procesu, který popisuje D klopny obvod, kde *D* je vstupní hodnota, *Q* je výstupní, *clk* jsou hodiny a *reset* je synchronní reset:

```
D.FLIP.FLIP : process (clk)
begin
    if (clk'event and clk='1') then
        if (reset = '1') then
            Q <= '0';
        else
            Q <= D;
        end if;
    end if;
end process;
```

Pokud na hodinách obvod zaznamená náběžnou hranu a není aktivní synchronní reset, tak na signál *Q* zapíše aktuální hodnotu na signálu *D*.

Celý obvod je možné popsat v rámci jedné entity, pro přehlednost je možné ale využívat komponenty. Při návrhu většího obvodu je vhodné logicky oddělit části obvodů od sebe a vytvořit z nich entity, které je možné poté propojit v nadřazené entitě pomocí klíčového slova component a namapování vstupních a výstupních signálů jednotlivých entit pomocí příkazu port map. Při mapování je poté možné pouze přiřadit vstupní a výstupní signály komponenty k signálům v nadřazené entitě [5].

Popis FSM

Jazyk VHDL nezná pojem automatu a existuje mnoho způsobů, jak jej popsat ve VHDL. Často používaný popis a zároveň takový, který usnadní práci logické syntéze programu je metoda se třemi procesy [6]. Prvním procesem je přechodová funkce a jedná se o kombinační proces, který na základě vstupních hodnot vyhodnocuje, jaký bude příští stav FSM. Vstupy přechodové funkce jsou stavy a vstupy FSM a jejím výstupem je příští stav. Druhý proces je registr stavu, který je sekvenčním procesem. Jeho vstupem je příští stav a výstupem je nový aktuální stav. Příklad registru stavu, kde *initial_state* je počáteční stav automatu, *CURRENT_STATE* je aktuální stav a *NEXT_STATE* je následující stav:

```
CLKP : process (clk)
begin
    if (clk'event and clk ='1') then
        if (reset = '1') then
            CURRENT_STATE <= initial_state;
        else
            CURRENT_STATE <= NEXT_STATE;
        end if;
    end if;
```

```

    end if;
end process;

```

Poslední proces je kombinační proces zvaný výstupní funkce, který na základě aktuálního stavu a v případě návrhu Mealyho FSM i vstupních hodnot vyhodnocuje výstup FSM. Výstupní funkce má jako vstupy stav a dle návrhu i vstup FSM. Výstupem je výstup FSM [6]. Příklad přechodové a výstupní funkce FSM s funkcí čítače modulo 4, kde T_STATE je výčet stavů čítače, TRANSP je přechodová funkce a OUTP je výstupní funkce (Mealyho návrh):

```

type T_STATE is (Q1, Q2, Q3, Q4);
signal CURRENT_STATE, NEXT_STATE : T_STATE;
begin
TRANSP : process (CURRENT_STATE, ce)
begin
    case CURRENT_STATE is
        when Q1 => if ce='1' then
                    NEXT_STATE <= Q2;
                else
                    NEXT_STATE <= Q1;
                end if;
        when Q2 => if ce='1' then
                    NEXT_STATE <= Q3;
                else
                    NEXT_STATE <= Q2;
                end if;
        when Q3 => if ce='1' then
                    NEXT_STATE <= Q4;
                else
                    NEXT_STATE <= Q3;
                end if;
        when Q4 => if ce='1' then
                    NEXT_STATE <= Q1;
                else
                    NEXT_STATE <= Q4;
                end if;

    end case;
end process;

OUTP : process (CURRENT_STATE, ce)
begin
    case CURRENT_STATE is
        when Q1 => if ce='1' then
                    q <= "01";
                else
                    q <= "00";
                end if;
    end case;
end process;

```

```

        end if;
when Q2 => if ce='1' then
    q <= "10";
else
    q <="01";
end if;
when Q3 => if ce='1' then
    q <= "11";
else
    q <="10";
end if;
when Q4 => if ce='1' then
    q <= "00";
else
    q <="11";
end if;
end case;
end process;

```

1.2 Programovatelná hradlová pole FPGA

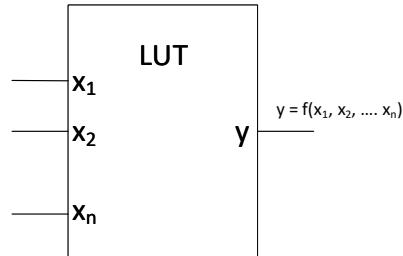
Tato sekce se zabývá tím, co jsou programovatelná hradlová pole (Field of programmable gate array, FPGA) a jak probíhá práce s těmito hradlovými poli na vývojové desce. V podsekci 1.2.1 popisují části FPGA, které jsou použity pro implementaci číslicového obvodu. Podsekce 1.2.2 stručně vysvětluje kroky nezbytné pro implementaci samotného číslicového obvodu na základě jeho popisu VHDL kódem. V podsekcích 1.2.3 a 1.2.4 je stručně popsána použitá vývojová deska a vývojové prostředí, které se váže k FPGA, které deska obsahuje.

1.2.1 Dostupné prostředky

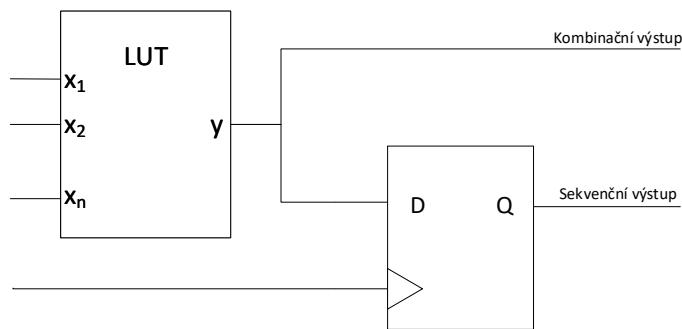
Základní stavební prvky

Technologie FPGA využívá tří základních stavebních prvků. Jedním z nich je propojení. Propojení tvoří síť vodičů, které jsou propojeny MOS tranzistory nebo tzv. antipojistkami (nevodivý prvek, který je možné vnějším působením napětí prorazit a poté přejde do vodivého stavu). Dle nastavení buňky v konfigurační paměti jsou vodiče pomocí tranzistorů spojeny. Stejně jako u logických hradel pro tyto spínací prvky platí, že jejich odpor působí přídavné zpoždění. Při konfiguraci je tedy potřeba, aby FPGA vytvořilo specializovaný rozvod hodinového signálu, který zajistí, že signály budou správně načasovány [7]. Hodinový signál mívá stromovou topologii k zajištění co největší časové přesnosti v celém obvodu.

Druhým základním stavebním prvkem jsou tzv. Look-up Tables (zkr. LUT). Pomocí programového propojení lze nakonfigurovat síť kombinačních prvků.



Obrázek 1.15: Příklad zakreslení LUT.



Obrázek 1.16: Příklad základního bloku FPGA.

FPGA využije části konfigurační paměti jako pravdivostní tabulky k implementaci logické funkce. LUT může být zakreslen jako na obrázku 1.15.

Posledním stavebním prvkem jsou registry. Typicky ke každé logické funkcii přísluší jeden registr. V FPGA převládají hranové D klopné obvody, které mají typicky vstup pro povolení hodin [7]. Základní blok může vypadat jako na obrázku 1.16.

1.2.2 Logická syntéza

Kroky logické syntézy

Logická syntéza je proces, při kterém je popis hardware převeden do schématu, který užívá konkrétní dostupné prostředky daného FPGA. Proto se dá říct, že k dosažení nejlepších výsledků je potřeba navrhovat obvod s ohledem na cílovou technologii. Logická syntéza se skládá z několika kroků. Mezi důležité kroky patří syntéza, mapování a „place and route“ [8]. Úkolem syntézy je rozpoznání charakteristických bloků použitých v popisu hardware a vytvoření tzv. netlistu (tj. schéma z logických hradel). Nástroj musí umět rozpozнат aritmetické operátory,



Obrázek 1.17: Digilent Basys 2.

paměti, čítače, FSM, multiplexory a další bloky. V rámci syntézy probíhá také proces předoptimalizace, který např. vymáže nevyužité bloky v popisu a vysokovýrovnové datové typy, jako je například integer (označení celočíselného 32-bitového datového typu) přeloží do bitové reprezentace. Mapování řeší realizaci obvodu pomocí dostupných bloků FPGA. Jedná se o překlad z netlistu na dostupné stavební bloky. Úkolem place and route je napojit přeložené stavební bloky na ty, které obsahuje konkrétní čip a rozvést spoje v čipu tak, aby správně fungovalo např. časování obvodu. Po provedení těchto kroků je vytvořen tzv. bitstream, který je možné nahrát do paměti konkrétního FPGA.

1.2.3 Vývojová deska Digilent Basys 2

Specifikace

Vývojová deska Digilent Basys 2 [9] viz obrázek 1.17. je platforma pro návrh a implementaci logických obvodů. Deska obsahuje Spartan-3E FPGA od firmy Xilinx. Může pracovat s vestavěným oscilátorem, který lze nastavit na frekvenci 25, 50, nebo $100MHz$, nebo má připravený socket pro druhý externí oscilátor. Využívá XCF02 Flash ROM (Read Only Memory), ve které ukládá FPGA konfigurace na dobu neurčitou. Pracuje také s tzv. Pmods (analogové a digitální I/O (vstupní i výstupní) moduly, které dovolují například převod z analogového signálu na digitální a naopak). Má tři vestavěné stabilizátory napětí ($1.2V$, $2.5V$, $3,3V$), které dovolují využívat $3.5V - 5.5V$ externí napájecí zařízení [9]. Typicky využívá napájení z USB, přes který může mít FPGA také naprogramováno, ale deska obsahuje také konektor pro baterii. Vstupní napájení je vedeno přes tzv.

power switch. Deska musí být nakonfigurována pro vykonávání dané činnosti. K vygenerování bitstreamu pro Spartan-3E FPGA je nutné použít vývojové prostřední od firmy Xilinx. Software od firmy Digilent zvaný Adept poté může být použit ke konfiguraci FPGA přes USB [9].

Dostupné prostředky

Vývojová deska obsahuje čtyři tlačítka, osm přepínačů, které lze využít jako vstupy obvodu. Jako výstup lze využít osmi LED diod, nebo sedmsegmentového displeje. Sedmsegmentový displej na desce obsahuje čtyři číslice, které jsou připojeny na společné anodě (tzn. jsou aktivní při logické '0'). Krom těchto prvků obsahuje deska také PS/2 a VGA konektor (tříbitvé D/A převodníky pro červenou a zelenou barvu a dvoubitový převodník pro modrou) a konektory pro přídavné moduly.

1.2.4 Vývojové prostředí Xilinx ISE

Uživatelské rozhraní

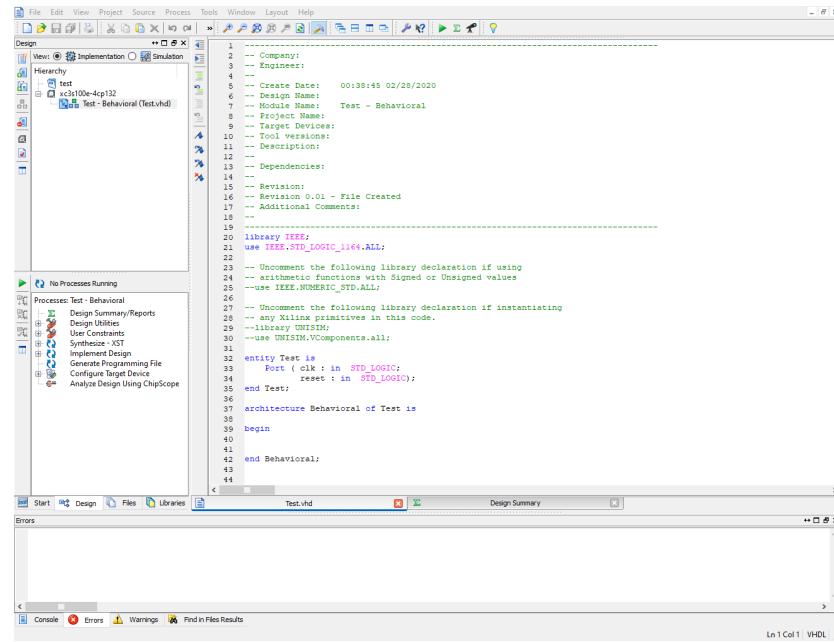
Xilinx ISE je softwarový nástroj od firmy Xilinx pro syntézu a analýzu HDL návrhů [10]. Umožňuje syntézu a simulaci návrhu, vytváření návrhů v jazycích VHDL nebo Verilog a konfiguraci FPGA (FPGA od firmy Xilinx, nikoliv od jiné). Primárním rozhraním je Project Navigator, ve kterém lze editovat zdrojový kód, sledovat výstupní konzoli a probíhající kroky syntézy viz obrázek 1.18. Okno hierarchie zobrazuje návrhové soubory (moduly), jejichž závislosti mají stromovou strukturu. Okno procesů popisuje operace, které bude ISE provádět na aktivním modulu (často syntéza, mapování, ...). Pro simulaci návrhu lze využít ISIM nebo ModelSim logického simulátoru, jejichž programy musí být také psané v HDL. Tyto simulátory zvládají primárně tyto typy simulací: Logické ověření (kontrola výstupů modulu), ověření časování a simulace post-place and route (ověření chování po naprogramování FPGA) [10].

1.3 Rozhraní

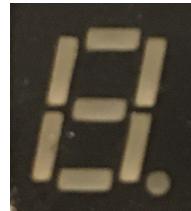
Tato sekce popisuje komunikační rozhraní použitá v návrhu a implementaci, která zajistují uživatelský vstup a výstup. V podsekci 1.3.1 vysvětlují rozhraní 7-segmentového displeje. Sekce 1.3.2 a 1.3.3 popisují protokoly PS/2 a VGA.

1.3.1 7-segmentový displej

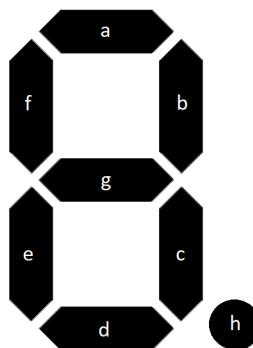
Sedmsegmentový displej se skládá ze sedmi uspořádaných LED diod viz obrázek 1.19. Většina sedmsegmentových displejů obsahuje také osmou LED diodu, kterou využívá jako tečku. Jednotlivé segmenty je možné rozsvítit přivedením daného signálu (při zapojení na společnou katodu logickou 1 a při zapojení na společnou anodu na logickou 0). Častý způsob zapojení segmentů je na obrázku 1.20. Kombinacemi rozvíjených segmentů lze vytvořit obrasy až 128 znaků.



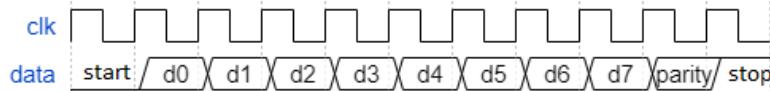
Obrázek 1.18: Základní pohled na Xilinx ISE Project Navigator.



Obrázek 1.19: Sedmisegmentový displej.



Obrázek 1.20: Příklad zapojení segmentů na sedmisegmentovém displeji.



Obrázek 1.21: Časový diagram PS/2 přenosu.

1.3.2 PS/2

Charakteristika

Rozhraní PS/2 využívá dvoudrátové sériové komunikace (data a hodiny) většinou pro ovládání myši, nebo klávesnice. Při přenosu dat využívá jedenáctibitové slovo, které obsahuje tzv. start bit, stop bit, paritní bit a osmibitovou hodnotu (1 bajt), která představuje přenesenou informaci [11]. Pokud nejsou přenášena data, signály rozhraní PS/2 jsou v klidovém stavu v logické 1. Při přenosu datový kabel sériově vysílá informace na frekvenci hodinového signálu. Časový diagram PS/2 přenosu je na obrázku 1.21.

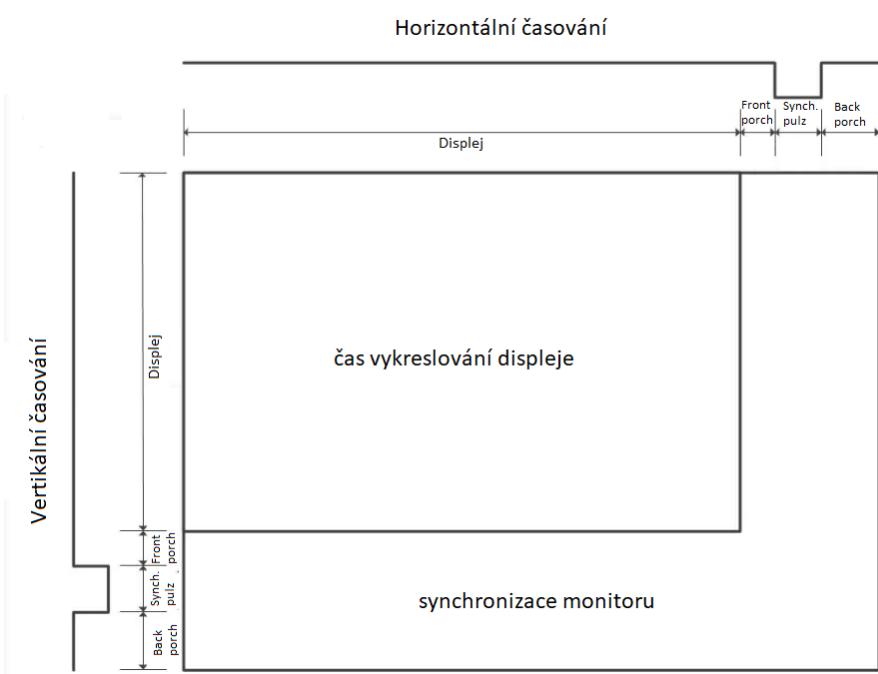
PS/2 klávesnice

PS/2 klávesnice využívá skenovací kódy ke komunikaci se zařízením. Každá z kláves má přiřazený kód, který je odeslán při stisknutí, nebo puštění klávesy. V případě, že se jedná o speciální klávesu, nebo o puštění klávesy, tak může klávesnice přenést více jedenáctibitových slov v sérii. Při stisknutí speciální klávesy bývá odeslána informace '11100000' (tj. 'E0' v hexadecimální soustavě) a po ní kód příslušné klávesy. Při puštění klávesy je odeslána hodnota F0. Například při puštění šipky směřující doleva (kód 6B) jsou v sérii odeslány informace E0, F0, 6B.

1.3.3 VGA

Charakteristika

VGA je standard pro řadič zobrazující grafický výstup [11]. Využívá paralelního přenosu k předávání informací řadiči grafického výstupu. Přenáší synchronizační signály (HS - Horizontální synchronizace, VS - Vertikální synchronizace) a signály RGB (tj. signály definující vykreslovanou barvu). HS a VS dle své frekvence nastavují rozlišení monitoru a umožňují tedy časovou synchronizaci s monitorem, který vykresluje pixel po pixelu. Po příchodu synchronizačního signálu je třeba čekat daný čas, než se monitor sesynchronizuje (tj. časy back porch a front porch, viz obrázek 1.22). RGB jsou analogové signály a hloubka vykreslované barvy je dána příslušným napětím (často 0 – 7V) [11].



Obrázek 1.22: Časový diagram VGA.

1.4 Hra Pac-Man

Pac-Man je plošinová arkádová hra, ve které jde o ovládání žluté kuličky s ústy (Pac-Mana) v bludišti vyplněném tečkami, které má za úkol sníst. Krom Pac-Mana se v bludišti nachází také čtyři duchové, kteří se snaží Pac-Mana dohonit před tím, než stihne sníst všechny tečky v bludišti. Pokud některý z nich Pac-Mana chytí, tak je hráči odebrán jeden ze tří životů a hra začíná odznova uprostřed bludiště bez teček, které jsou již snězeny a duchové jsou vráceni zpět do tzv. domečku. Aby se Pac-Man mohl bránit duchům, tak se na mapě nachází čtyři větší tečky, po jejichž snězení může sníst i duchy, které se následně na chvíli zastaví v domečku a čekají na vyprchání efektu. Hra může být ovládána čtyřmi tlačítky, kde každé nastaví Pac-Manovi nový směr (dle tlačítka nahoru, dolu, doleva, doprava).

Kapitola 2

Návrh hry

V této kapitole se zabývám návrhem obvodu pro finální implementaci. V sekci 2.1 rozeberu návrh hry jako celek. Sekce 2.2 rozebírá návrh obvodu pro ovládání periferií (PS/2, VGA, sedmisegmentový displej). V sekci 2.3 vysvětluji návrh vnitřní logiky samotné hry a v sekci 2.4 poté propisování textur na výstup. V sekci 2.5 popisuji propojení všech navržených částí v jeden celek.

2.1 Specifikace hry

V této sekci rozebírám konkrétní prvky hry a vysvětluji návrhy pro jejich řešení. V podsekci 2.1.1 popisují, jak má hra fungovat jako celek. Podsekce 2.1.2 vysvětluje optimální návrh herního pole, na kterém se hra bude odehrávat a podsekce 2.1.3 popisuje chování postav na navrženém herním poli.

2.1.1 Cíl a chování hry

Cílem práce je navrhnout obvod realizující klona známé 2D arkádové hry a implementovat jej na vývojové desce Digilent Basys 2. Hráč bude moci hrnu ovládat pomocí šipek na PS/2 klávesnici a stav hry bude moci sledovat na VGA monitoru. Na začátku hry se hráčova postava (Tux-Man) objeví v bludišti, ve kterém se budou nacházet také čtyři duchové. Hráč bude muset pomocí šipek na klávesnici ovládat Tux-Mana a sbírat body rozpoložené na mapě meziklou, co nesmí přijít do styku s duchy. Body budou počítány v hexadecimální soustavě na sedmisegmentový displej dostupný na vývojové desce. Tux-Man má tři životy na to, aby posbíral všechny body v bludišti. Životy bude možné sledovat na LED diodách na vývojové desce. Pokud hráče duchové trikrát chytí, tak bude hra zastavena a bude nutné resetovat obvod pro novou hru. Ve hře bude jedna hratelná úroveň, kde se duchové budou pohybovat stejnou rychlostí, jako Tux-Man.

2.1.2 Herní mapa

Herní mapa bude navržena jako mřížka. Navrhoji rozdělit rozlišení VGA na matice, kde každá buňka má rozlišení 16×16 pixelů. Z těchto buněk bude možné jednoduše vytvořit herní mapu s rozlišením 21×19 buňek dle originální mapy. Výhodou tohoto řešení je možnost jednoduchého uchovávání informací o jednotlivých políčkách mapy v paměti a přistupování k těmto informacím dle souřadnic x a y . Každá souřadnice bude moci mít několik informací, jimiž jsou, zda se na nich nachází Tux-Man, duch, bod nebo zed'.

2.1.3 Postavy a jejich chování

Každá z postav bude mít nastavený směr dle vnitřního signálu (u Tux-Mana ovládáno šipkami na klávesnici, u duchů řízeno automatem), na základě kterého se bude posouvat o jednu souřadnici na dané frekvenci. Těsně před posunem postavy proběhne kontrola, co se nachází na souřadnicích před danou postavou dle nastaveného směru. Pokud se na souřadnicích před postavou nic nenachází, tak je posunuta. V případě Tux-Mana, pokud se bude jednat o bod, tak se daný bod smaže z paměti a Tux-Man je přesunut na jeho pozici a pokud se bude jednat o zed', tak jeho souřadnice nebudou změněny. Duchové body budou ignorovat a pokud se před nimi bude nacházet zed', tak pomocí generátoru náhodných čísel vyhodnotí nový směr. Duchové budou tedy náhodně měnit směr pokaždě, když dojdou ke zdi.

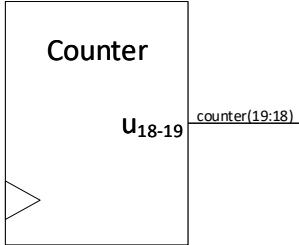
2.2 Periferie

V této sekci rozebírám návrhy číslicových logických obvodů pro komunikaci s periferiem. V podsekci 2.2.1 popisuji návrh pro ovládání sedmsegmentového displeje. Podsekce 2.2.2 popisuje navržený obvod pro zpracování signálu z PS/2 klávesnice. a v sekci 2.2.3 poté vysvětlují obvod generující signály pro ovládání VGA rozhraní.

2.2.1 Sedmsegmentový displej

Analýza

Pro počítání bodů na sedmsegmentový displej jsem zvolil hexadecimální soustavu, každá číslice tedy bude mít čtyřbitový vstup. Na sedmsegmentovém displeji vývojové desky Digilent Basys 2 je možné na všech čtyřech číslicích zobrazovat v jednu chvíli pouze jednu hodnotu, ale každá z číslic má vlastní povolovací vstup (při logické '0' svítí a při logické '1' nesvítí). Navrhoji tedy postupně přepínat aktivní číslice tak, aby byla v jednu chvíli aktivní pouze jedna a zároveň přepínat hodnotu, která je zobrazena na displeji. Pokud tak bude provedeno na vysoké frekvenci, tak lidské oko nebude schopno zachytit problikávání displejů a bude vytvořena iluze plynulého obrazu zobrazujícího jinou hodnotu na každé



Obrázek 2.1: Čítač s vyvedeným osmnáctým a devatenáctým bitem.

číslici. Problémem tohoto řešení je obnovovací frekvence sedmisegmentového displeje (při problikávání číslice na vysoké frekvenci nemusí vestavěné LED diody stíhat měnit svůj stav). Pokusím se tedy návrhem co nejvíce přiblížit minimální frekvenci nerozeznatelné lidským okem (zhruba tříčet snímků za vteřinu).

Návrh obvodu

K docílení chtěné frekvence navrhoji využít čítač, který čítá v reakci na náběžnou hranu hodinového signálu ($50MHz$), jehož x -tý bit bude této frekvenci dosahovat. Uvážíme-li čtyřčíslicový displej, frekvenci hodinového signálu $50MHz$ a pro každou číslici chtěnou frekvenci zhruba $30Hz$, tak bude platit následující:

$$(50 * 10^6) / 2^x = 4 * 30$$

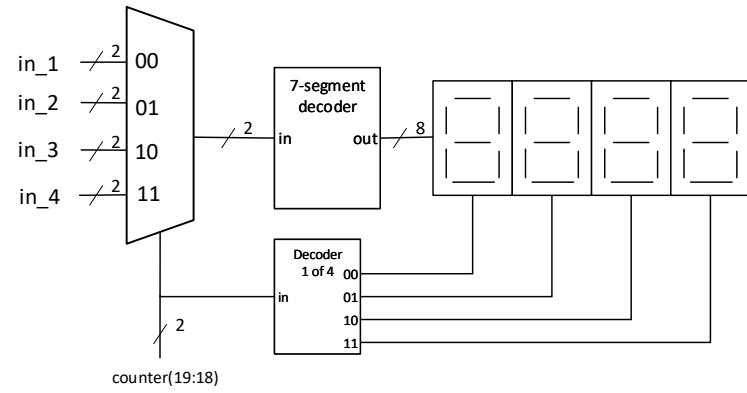
$$\log_2((50 * 10^6) / 120) = x$$

$$x = 18.67$$

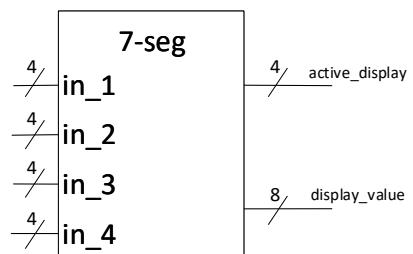
Po zaokrouhlení na celé číslo dolů vyjde osmnáctý bit čítače, který se nejvíce přibližuje chtěné frekvenci. Z důvodu přepínání mezi čtyřmi číslicemi potřebujeme z čítače dva byty. Čítač tedy může vypadat jako na obrázku 2.1. Úkolem tohoto čítače bude paralelně přepínat aktivní vstupy a aktivní číslice na sedmisegmentovém displeji. K přepínání vstupů využiji multiplexor a k přepínání aktivní číslice dekodér 1 z 4. Ke správnému propsání číslice na displej musí být hodnota dekódována dekodérem na sedmisegmentový displej (pro přehlednost dle ASCII tabulky). Celý obvod může vypadat jako na obrázku 2.2. Modul budu dále značit jako na obrázku 2.3.

2.2.2 PS/2

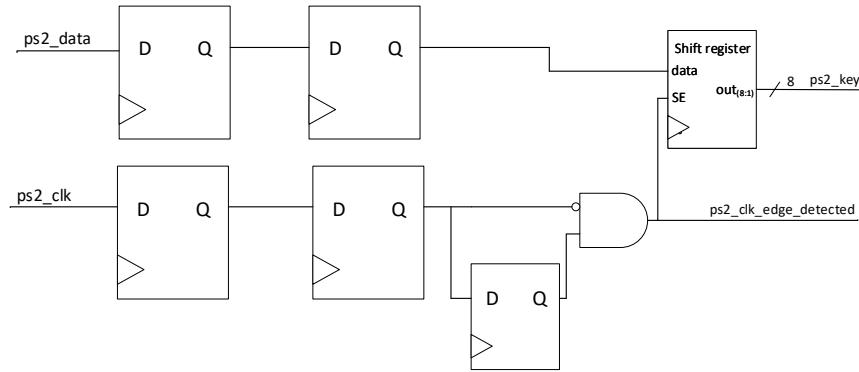
Cílem je navrhnout modul, který zpracuje příchozí signál z klávesnice a na výstup bude posílat kód stisknuté klávesy, indikovat zda se jedná o speciální klávesu a zda je klávesa aktuální (strobe). Pro hru nebude nutné sledovat puštění kláves, ale pouze jejich stisknutí. Tento modul lze pro usnadnění navrhnout tak,



Obrázek 2.2: Připojení sedmisegmentového displeje na čítač.



Obrázek 2.3: Značení modulu pro ovládání sedmisegmentového displeje.



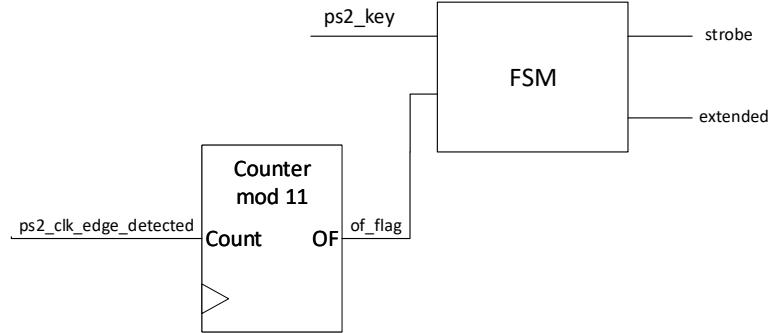
Obrázek 2.4: Návrh obvodu pro zpracování příchozích dat z klávesnice PS/2.

aby přehlížel puštění klávesy. Kvůli o mnoho vyšší frekvenci oscilátoru na desce než je frekvence hodinového signálu PS/2 navrhoji využít synchronizér na oba signály PS/2 a pomocí detektoru sestupné hrany dle protokolu povolovat zápis dat do registrů. Pro uložení jedenáctibitového slova využiji jedenáctibitového posuvného registru s povolovacím vstupem (pokud je aktivní povolovací vstup, tak s náběžnou hranou hodinového vstupu všechny hodnoty předá o řád výš a zapíše vstupní hodnotu na nejnižší bit). Připojením detektoru sestupné hrany z hodinového signálu PS/2 na povolovací vstup posuvného registru docílíme zápisu jedenáctibitového slova do paměti. Obvod může být zakreslen jako na obrázku 2.4. Výstupem druhého až devátého bitu posuvného registru (v obrázku jako *ps2_key*) je informace o klávese. Pro detekci zda je klávesa aktuální nebo se jedná o speciální klávesu nebo o puštění klávesy jsem se rozhodl využít konečný stavový automat (FSM), který bude měnit hodnoty výstupů *strobe* a *extended* (speciální klávesa) na základě příchozích kláves. Pro kontrolu je potřeba, aby automat věděl, kdy je v registru hodnota klávesy aktuální. Výstupu indikujícímu, že je informace v posuvném registru aktuální lze docílit připojením signálu *ps2_clk_edge_detected* k čítači modulo 11. Ze signálu přetečení čítače lze poté vyčíst, zda jsou všechna data načtena. Návrh obvodu je na obrázku 2.5, kde *OF* je indikátor přetečení čítače. FSM lze poté navrhnout se třemi stavami. Rozhodl jsem se využít Mealyho návrh FSM, viz obrázek 2.6.

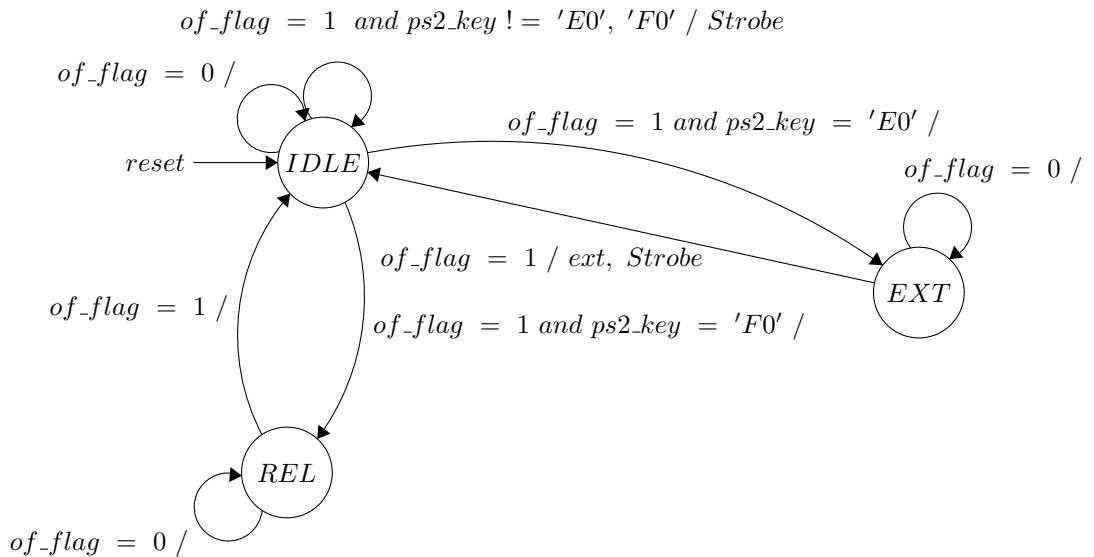
2.2.3 VGA

2.3 Logika hry

Tato sekce popisuje návrh obvodu pro vnitřní logiku hry. V podsekci 2.3.1 vysvětlují způsob navržení herní mapy. Sekce 2.3.2 popisuje návrh obvodu pro ovládání postavy na mapě a sekce 2.3.3 se zabývá návrhem pro samovolný pohyb



Obrázek 2.5: Návrh obvodu pro detekci speciálních kláves a aktivity.



Obrázek 2.6: Stavový diagram FSM pro indikaci zda je klávesa aktuální, speciální, či se jedná o puštění klávesy.

duchů po mapě.

2.3.1 Herní mapa jako mřížka

2.3.2 Ovládání postavy

2.3.3 Pohyb duchů

2.4 Textury a propsání na výstup

2.5 Kompletní propojení

Kapitola 3

Implementace

Tato kapitola se zabývá návrhem číslicových obvodů pro jednotlivé logicky oddělené bloky na základě herní logiky a jejich implementací. Sekce 3.1 a 3.2 popisují číslicové obvody navržené pro čtení vstupních dat z klávesnice a generování výstupu na monitor. Sekce 3.3 poté vysvětluje vnitřní zapojení jednotlivých bloků zajišťujících logickou funkčnost hry. V sekci ?? je poté popsáno kompletní zapojení všech částí do funkčního celku.

3.1 Modul pro čtení z PS/2 klávesnice

V této sekci vysvětluji číslicový obvod, který jako celek zpracovává vstupní signál z klávesnice a generuje daný výstup. Podsekce 3.1.1 rozebírá číslicový obvod pro zpracování vstupního signálu a podsekce 3.1.2 řeší číslicový obvod pro generování výstupního signálu.

3.1.1 Zpracování vstupu

3.1.2 Generování výstupu

3.2 Modul pro výstup na VGA monitor

Tato sekce popisuje číslicový obvod, který zařizuje funkčnost monitoru a možnost propisání výstupu na něj. V podsekci 3.2.1 popisují obvod, který zajišťuje funkčnost a v podsekci 3.2.2 vysvětluji, jak propisují textury na monitor.

3.2.1 Časování VGA

3.2.2 Propsání textur na monitor

3.3 Herní logika

V této sekci popisují číslicový obvod, který zajišťuje vnitřní funkcionalitu samotné hry. V podsekci 3.3.1 je vysvětlený obvod, který řeší herní plochu, na které se postavy pohybují. Podsekce 3.3.2 popisuje obvod ovládající postavy ve hře a v podsekci 3.3.3 je vysvětlen obvod, který řeší splnění cílů hry.

3.3.1 Herní mapa

3.3.2 Ovládání postav

3.3.3 Cíle hry

Kapitola 4

Testování

Závěr

Literatura

- [1] G. Boole, *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities.* Dover Publications, 1854, ISBN: 0486600289.
- [2] doc. Ing. Hana Kubátová CSc., *Struktura a architektura počítačů s řešenými příklady*, 2nd ed. Thákurova 1, 160 41, Praha 6: České vysoké učení technické v Praze, 2018, ISBN: 9788001064108.
- [3] B. MELICHAR, *Jazyky a překlady.* České vysoké učení technické v Praze, ISBN: 80-01-01511-4.
- [4] Bečvář, Daněk, Schmidt, Novotný, “Přechod mezi hodinovými doménami,” Přednáška předmětu BI-PNO, 2006.
- [5] Martin Novotný, “VHDL - Processes, signals and data types,” Přednáška předmětu BI-PNO, 2006.
- [6] Bečvář, Daněk, Schmidt, Novotný, “VHDL II - Automaty, typy, generický popis,” Přednáška předmětu BI-PNO, 2006.
- [7] ——, “Technologie návrhu a realizace číslicových obvodů,” Přednáška předmětu BI-PNO, 2006.
- [8] ——, “Struktura FPGA, RTL Syntéza, optimalizace,” Přednáška předmětu BI-PNO, 2006.
- [9] *Basys 2TM FPGA Board Reference Manual*, Digilent.
- [10] *ISE Design Suite Software Manuals and Help*, Xilinx.
- [11] B. Govindarajalu, *IBM PC AND CLONES.* Mc Graw Hill India, ISBN: 9780070483118.