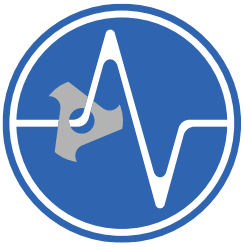


Střední průmyslová škola strojní
a elektrotechnická a Vyšší odborná škola,
Liberec 1, Masarykova 3

TUXMAN - FPGA

Maturitní/Ročníková 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 sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 21. února 2020

Martin Přívozník

Obsah

Úvod

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 věnuji 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 zaměřená na návrh sekvenčních obvodů a synchronních sekvenčních automatů (FSM), na což naváže podsekce 1.1.3, ve které vysvěluji 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 mém 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 Booleově algebře stejnou prioritu a jiný význam (žádná 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)
$a + \overline{a} = 1$	$a * \overline{a} = 0$	(vlastnosti negace)

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

de Morgan	$\overline{(a + b)} = \overline{a} * \overline{b}$	$\overline{(a * b)} = \overline{a} + \overline{b}$
idempotence	$a + a = a$	$a * a = a$

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 vyjímkou 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.

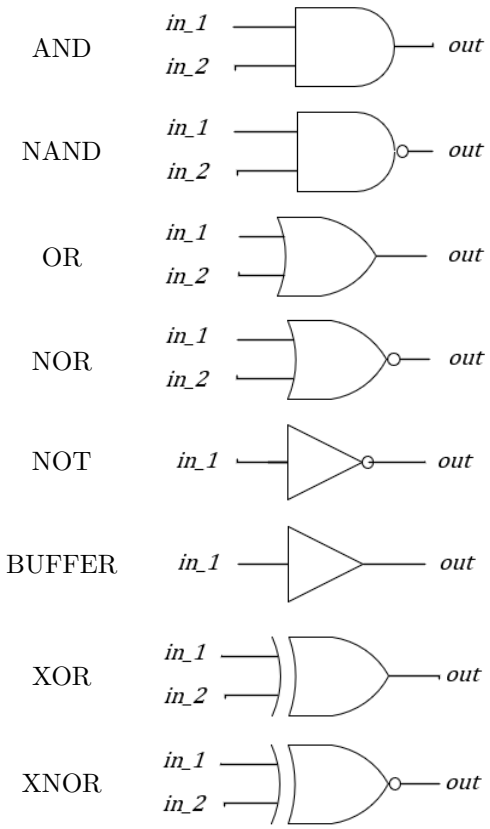
Platí, že u číslicových obvodů každá proměnná v operaci nabývá hodnotu jednoho tzv. bitu. Bit je základní jednotkou dat a může nabývat hodnot 0, nebo 1. Reprezentací číslicového obvodu je schéma číslicového obvodu, kde každá z funkcí je reprezentována tzv. schématickou značkou. Schématické značky mohou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

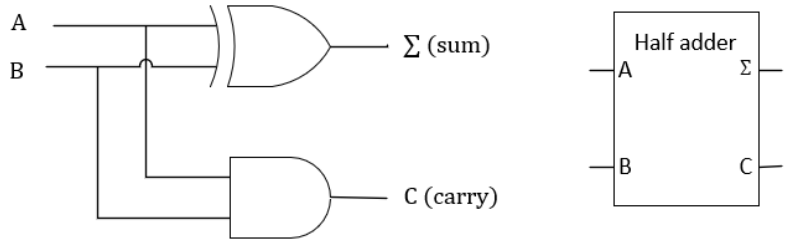
Název	Pravdivostní tabulka	Formule
AND (logický součin)	in_1 in_2 out 0 0 0 0 1 0 1 0 0 1 1 1	$out = in_1 * in_2$
NAND (negovaný logický součin)	in_1 in_2 out 0 0 1 0 1 1 1 0 1 1 1 0	$out = \overline{in_1 * in_2}$
OR (logický součet)	in_1 in_2 out 0 0 0 0 1 1 1 0 1 1 1 1	$out = in_1 + in_2$
NOR (negovaný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

Tabulka 1.4: Tabulka nejpoužívanějších základních logických funkcí.



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

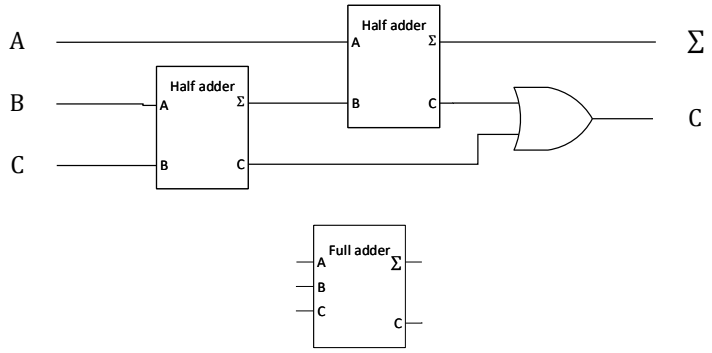
jsou evropská a americká. Příklady schématických značek pro nejpoužívanější logické funkce jsou uvedené v tabulce 1.5. Pro usnadnění práce můžeme využívat logických bloků, které plní danou funkci. Opět platí, že ze značení logických bloků ve schématu musí plně vyplívat, jakou funkci zastupují. Logický 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 sčítačka, viz obrázek 1.1. Poloviční sčítačka umí sečíst dvě jednobitová čísla a vygenerovat bit do vyššího řádu (carry) podle pravdivostní tabulky 1.6. Zřetězením dvou polovičních sčítaček a přenesením pomocí funkce OR získáme poté kompletní sčítačku, viz obrázek 1.2.[2] Kompletní sčítačka umí sečíst jednobitová čísla, vygenerovat bit do vyššího řádu a přijmout bit z nižšího, sčítá tedy tři bity. Sčítá počet jedniček na vstupech.



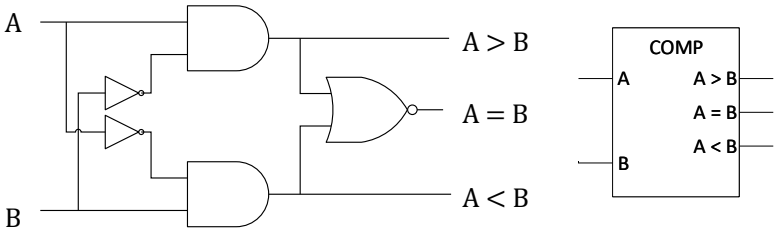
Obrázek 1.1: Schéma poloviční sčítačky a příklad jejího značení ve schématu.

A	B	C	$\sum$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

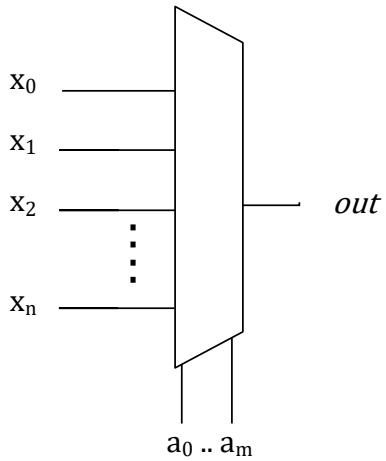
Tabulka 1.6: Pravdivostní tabulka poloviční sčítačky.



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



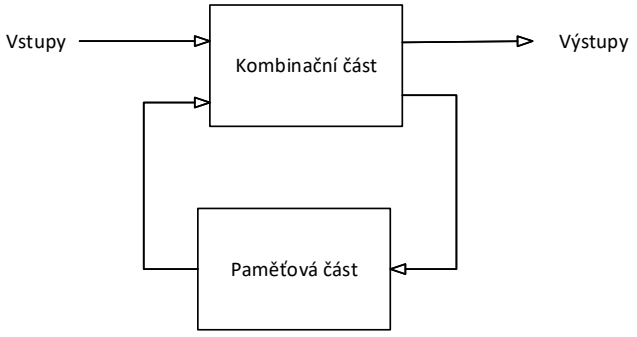
Obrázek 1.3: Schéma komparátoru a příklad jeho značení ve schématu.



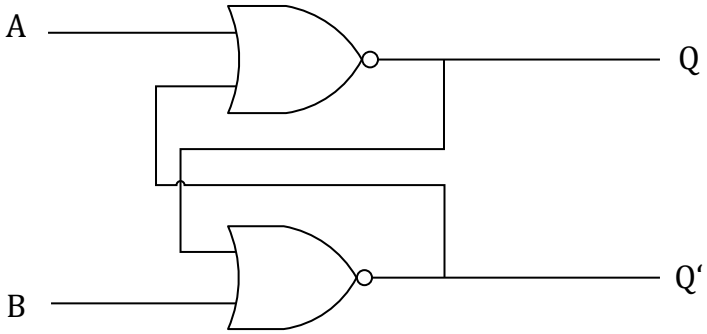
Obrázek 1.4: Schématická značka multiplexoru.

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

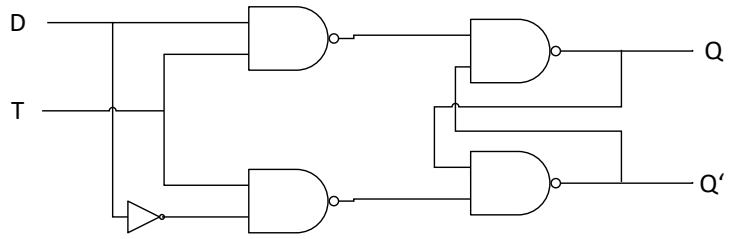
Sekvenční logický obvod

Sekvenční logický obvod je takový obvod, ve kterém výstupní hodnoty nejsou dány pouze aktuální kombinací vstupních proměnných, ale zároveň předchozím 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), překlápěcí část a paměť, viz tabulka 1.7. Pokud se tedy změní vstupní

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.

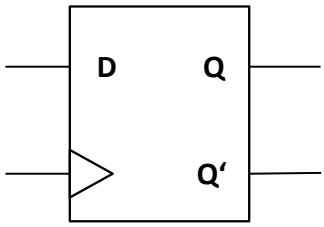
hodnoty z $[0, 1]$, nebo $[1, 0]$ na $[1, 1]$, na výstupu bude předchozí hodnota, dokud neproběhne další změna vstupních hodnot. Jedná se tedy o tzv. hladinový klopný obvod (angl. latch). Častěji využívaným klopným obvodem je tzv. D klopný obvod, viz obrázek 1.7. Výhodou D klopného obvodu je fakt, že 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 byl místo trojúhelníku čtverec.

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.

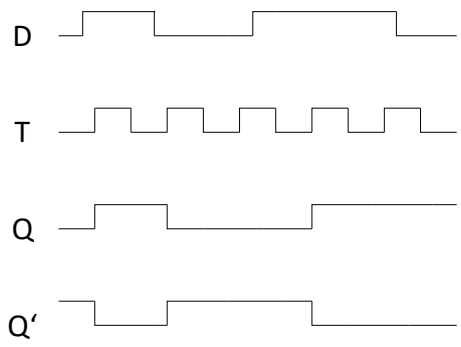
Příkladem sekvenčního logického bloku je čítač. Čítač je takový sekvenční logický obvod, který uchovává v ,,paměti"informaci o tom, kolikrát zaznameno-

T	D	Q	Q'
$\_\!\!\sqcap$	x	x	$\overline{x}$
Jinak		Paměť	

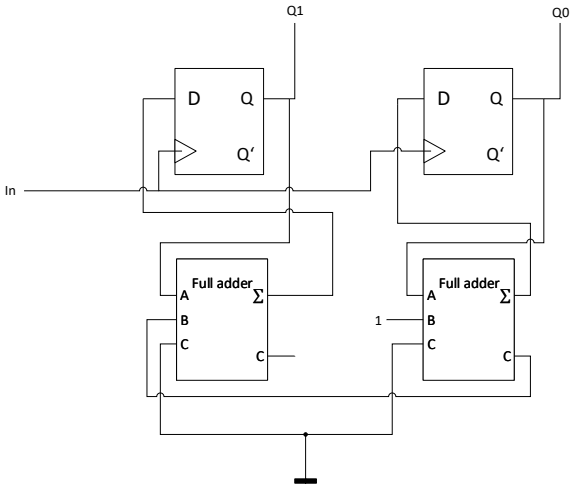
Tabulka 1.8: Pravdivostní tabulka D klopného obvodu.



Obrázek 1.8: Příklad značení D klopného obvodu ve schématu.



Obrázek 1.9: Časový diagram D klopného obvodu.



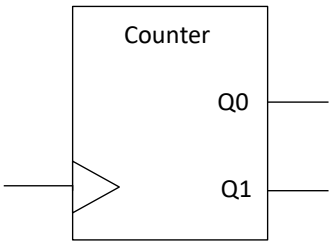
Obrázek 1.10: Příklad schématu dvoubitového čítače reagujícího na náběžnou hranu.

nal 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.

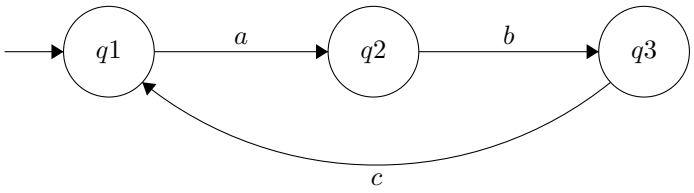
Konečný stavový automat

Konečný stavový automat (angl. Finite State Machine - FSM) M je šestice M=(Q, T, D, delta, lambda, q0), kde:

- Q je konečná množina vnitřních stavů
- T je konečná množina vstupních symbolů (signálů)



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



Obrázek 1.12: Příklad stavového diagramu konečného stavového automatu.

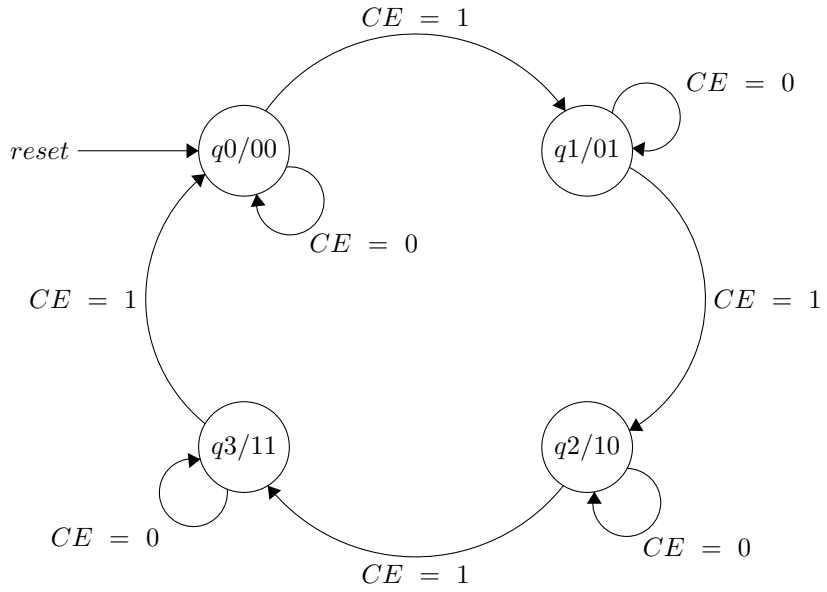
- D je konečná množina výstupních symbolů (signálů)
- delta je zobrazení z Q x T do Q (stav x vstup do stav) nazývané přechodová funkce
- lambda je zobrazení z Q x T do D (Mealy, ze stavu a vstupu do výstupu) nebo Q do D (Moore, jen ze stavu do výstupu)
- q0 je počáteční stav

[3] Má definované stavy, mezi kterými za daných podmínek přepíná a mění svůj výstup. FSM provádí vždy právě jeden přechod s každou náběžnou 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 počátečný 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 $q1$. Rozeznáváme dva typy FSM na základě definičního oboru výstupní funkce [2]. 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). Tuto část obvodu můžeme nazývat hodinová doména. Pro návrh synchronního návrhu platí několik pravidel, které je nutné dodržet pro správnost návrhu. Společné hodiny nesmí být hradlované (signál musí být přiveden přímo k hodinovým vstupům sekvenčních obvodů). V případě hradlování společných hodin může docházet k

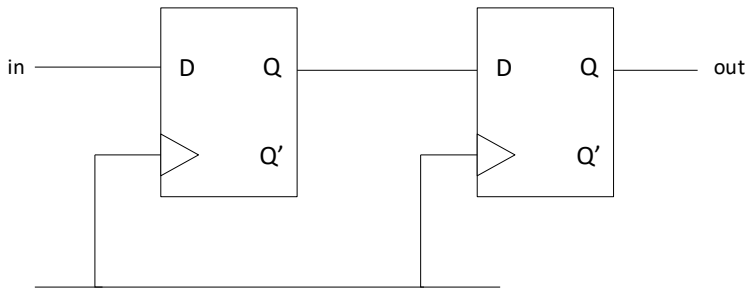


Obrázek 1.13: Příklad stavového diagramu (typ Moore) čítače modulo 4 se vstupem count enable.

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. V tomto případě je nutné společné hodiny synchronizovat specializovaným obvodem, který můžeme nazývat synchronizér. Synchronizér se může skládat z D klopných obvodů, které mají dané podmínky pro korektní funkci. 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 obrázku1.14.

Datové signály jsou řízené řídícími signály synchronizéru. Protokol řídících signálů musí být navržen tak, aby se měnil vždy jen jeden. Řídící signál můžeme nazývat strobe. Úkolem tohoto protokolu je indikace, zda jsou data aktuální (tj. na signálu strobe bude logická 1, dokud nezačne opět probíhat příchod nových dat).



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 návrh digitálních integrovaných obvodů, např. u 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 - Výčtový 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. 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 instance koponent, behaviorální popis (tj. popis chování), nebo tzv. procesy a nachází se v tělě architektury (tj. část za klíčovým slovem begin). Příklad architektury výše uvedeného multiplexoru:

```
architecture MUX_BODY 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 MUX_BODY;
```

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. [5] 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 i přes to, že 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 a synchronní reset). Příklad procesu, který popisuje D klopný obvod, kde D je vstupní hodnota, Q je výstupní, *clk* jsou hodiny a *reset* je synchronní reset:

```
D_FLIP_FLOP : process (clk, reset)
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 componenty. 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 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. 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. 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]

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 popisuji části FPGA, které jsou použity pro

Obrázek 1.15: Příklad zakreslení LUT.

Obrázek 1.16: Příklad základního bloku FPGA

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 se skládá ze tří základních stavebních prvků. Jedním z nich je propojení. Propojení tvoří síť vodičů, které jsou propojeny MOS tranzistory a 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 nebo pomocí antipojistek rozpojeny. 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. 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. 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 funkci přísluší jeden registr. V FPGA převládají hranové D klopné obvody, které mají typicky vstup pro povolení hodin. Základní blok FPGA může vypadat jako na obrázku 1.16.

1.2.2 Logická syntéza

1.2.3 Vývojová deska Digilent Basys 2

1.2.4 Prostředí Xilinx ISE

1.3 Rozhraní

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

1.3.1 7-segmentový displej

1.3.2 PS/2

1.3.3 VGA

1.4 Hra Pacman

Tato sekce vysvětluje hru, již má můj návrh klonovat. Podsekce 1.4.1 popisuje cíl a pravidla hry. V podsekci 1.4.2 vysvětluji ovládání hry.

1.4.1 Princip

1.4.2 Ovládání

Kapitola 2

Návrh hry

V této kapitole se zabývám návrhem herní logiky, na základě které je navržen číslicový obvod pro finální implementaci. Sekce 2.1 rozebírá konkrétní prvky hry a návrhy pro jejich řešení. V sekci 2.2 vysvětluji, jak jsou navržené textury, které se zobrazují na výstup.

2.1 Specifikace hry

V této sekci rozebírám konkrétní prvky hry a vysvětluji návrhy pro jejich řešení. V podsekci 2.1.1 popisuji, jak má hra fungovat jako celek a jak dosáhnout cíle hry. Podsekce 2.1.2 vysvětluje návrh herního pole, na kterém se hra odehrává a sekce 2.1.3 poté popisuje, jak se chovají jednotlivé postavy na navrženém herním poli.

2.1.1 Cíl a chování hry

2.1.2 Herní mapa

2.1.3 Postavy a jejich chování

2.2 Herní textury

Tato sekce popisuje návrh textur, které se zobrazují na výstup. V podsekci 2.2.1 vysvětluji způsob použití barev pro výstup. Sekce 2.2.2 se zabývá tím, v jaké podobě jsou textury, které se mají propisovat.

2.2.1 RGB

2.2.2 Textury jako matice barev

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 propsání výstupu na něj. V podsekci 3.2.1 popisuji obvod, který zajišťuje funkčnost a v podsekci 3.2.2 vysvětluji, jak propisuji textury na monitor.

3.2.1 Časování VGA

3.2.2 Propsání textur na monitor

3.3 Herní logika

V této sekci popisuji 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.

[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.

[3] B. MELICHAR, *Jazyky a překlady.* České vysoké učení technické v Praze.

[4] Bečvář, Daněk, Schmidt, Novotný, "Přechod mezi hodinovými doménami," 2006.

[5] Martin Novotný, "VHDL - Processes, signals and data types," 2006.

[6] Bečvář, Daněk, Schmidt, Novotný, "VHDL II - Automaty, typy, generický popis," 2006.