# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS


# SYNCHRONIZACE GRAFICKÉ APLIKACE BĚŽÍCÍ NA DVOU FITKITECH SOUČASNĚ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS


AUTOR PRÁCE Tomáš Kaplan
AUTHOR



BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# SYNCHRONIZACE GRAFICKÉ APLIKACE BĚŽÍCÍ NA DVOU FITKITECH SOUČASNĚ

SYNCHRONIZATION OF GRAPHIC APPLICATION RUNNING ON TWO FITKITS SIMULTANEOUSLY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE Tomáš Kaplan
AUTHOR

VEDOUCÍ PRÁCE Ing. Martin Žádník
SUPERVISOR

BRNO 2008

## Abstrakt

Tato bakalářská práce popisuje vývoj aplikace na platformě FITkit. Vývoj na zmíněné platformě obnáší vývoj konfigurace programovatelného hradlového pole (FPGA) od společnosti Xilinx a vývoj programu pro mikrokontrolér od společnosti Texas Instruments. Aplikace má za úkol demonstrovat synchronizaci grafické aplikace běžící na dvou FITkitech současně. Komunikace mezi zařízeními probíhá přes sériovou linku.



## Klíčová slova

FITkit, vestavěný systém, mikrokontrolér, programovatelné hradlové pole, sériová komunikace



## Abstract

This bachelor thesis describes development application on platform FITkit. Development on mentioned platform amounts making configuration for programmable gate field (FPGA) by company Xilinx and program making for micro-controller by company Texas Instruments. Application have as one's task demonstrate synchronization of graphic application running on two FITkits simultaneously. Inter-device communication proceeds over serial line.



## Keywords

FITkit, embedded system, microcontroller, Field-Programmable Gate Array, serial communication


## Citace

Kaplan Tomáš: Synchronizace grafické aplikace běžící na dvou FITkitech současně. Brno, 2008, bakalářská práce, FIT VUT v Brně.

# Synchronizace grafické aplikace běžící na dvou FITkitech současně

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Žádníka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Kaplan
4.5.2008

## Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce Ing.Martinu Žádníkovi za jeho ochotu při řešení problémů, které během práce nastaly.

# Obsah

# Úvod

V dnešní době se staly vestavěné systémy (Embedded systems) nedílnou součástí každodenního života. Jejich použití je natolik rozmanité, že nemohu všechny vyjmenovat, ale uvedu alespoň pár příkladů, jakými jsou například mobilní telefony, MP3 přehrávače, ale také řídící systémy jaderných elektráren.

Jedná se o jednoúčelové počítačové systémy často pracující v reálném čase. Předností těchto systémů je jejich energetická spotřeba, rozměry, hmotnost a pořizovací náklady. Toho lze s výhodou využít při hromadné produkci. Důležitými vlastnostmi těchto systémů je kritičnost, reaktivnost a autonomie. Kritičnost je vlastnost, která vypovídá o bezpečnosti, spolehlivosti a dokončenosti požadované úlohy. Například pokud se bude jednat o systém řídící výtah, tak je určitě požadováno bezpečné dokončení požadované úlohy a není možné, aby se například otevřely dveře dříve, než kabina dojede do požadovaného patra. Další z vlastností, která je reaktivnost, informuje o omezeních práce v reálném čase. Například pokud má systém za úkol mapovat terén při přeletu stíhačky, tak je velmi důležité, podávat informace s velmi krátkým časovým zpožděním. Překročením maximálního možného časového zpoždění může systém ohrozit chod celého letadla a tím i život pilota. Poslední zmíněná vlastnost popisuje míru zásahu člověka do systému. Nutnost častého zakročení člověka do systému se promítá ve spotřebě, spolehlivosti a robustnosti systému. Nejčastěji jsou zabudovány přímo do zařízení, které ovládají a obsahují minimum ovládacích prvků.

Na rozdíl od univerzálního počítače (například PC) uživatel nespouští v operačním systému nejrůznější programy, ale používá jeden program vyvinutý pro specifickou aplikaci. Vzhledem k tomu, že program podle, kterého systém běží, je vytvořen na míru a má za úkol pracovat léta bez chyby, je věnována velká pozornost jejich vývoji a testování.

Na fakultě informačních technologií (FIT) Vysokého učení technického v Brně vzniká pod vedením Dr. Ing. Otto Fučíka výuková platforma nazvaná FITkit [1]. FITkit je vestavěný systém sloužící k prezentaci školy a výuce v řadě kurzů fakulty. Vývojový tým za spolupráce studentů vyvíjí aplikace, které demonstrují možnosti použití těchto systémů. Příkladem mohou být některé již vytvořené: aplikace pro zobrazení textur na monitoru, aplikace ukazující možnosti tvorby vektorových objektů (3D grafika), aplikace hodin reálného času nebo třeba aplikace pro měření teploty.

Důležitou již zmíněnou vlastností je možnost práce v reálném čase. To znamená umět reagovat na určité podněty v určitém časovém intervalu. Toho lze s výhodou využít ke zpracování analogových signálů a ke komunikaci více zařízení mezi sebou.

Komunikace dvou zařízení je hlavním tématem této bakalářské práce. Jako ukázka synchronizace v reálném čase slouží grafická aplikace, která představuje hru „tenis“. Oba vestavěné systémy spolu komunikují po sériové lince, zpracovávají vstupy od uživatele a zobrazují stav hry na monitoru.

Text této bakalářské práce je rozdělen do čtyř hlavních kapitol, které se zabývají vývojovým cyklem projektu. V první kapitole nazvané „Teoretický rozbor“ jsou uvedeny informace potřebné pro realizaci projektu. V druhé kapitole je popsán návrh systému, který je úzce spjat s kapitolou třetí. V té je popsána implementace. Zhodnocení práce a dosažených výsledků se nachází v poslední kapitole.

# 1 Teoretický rozbor

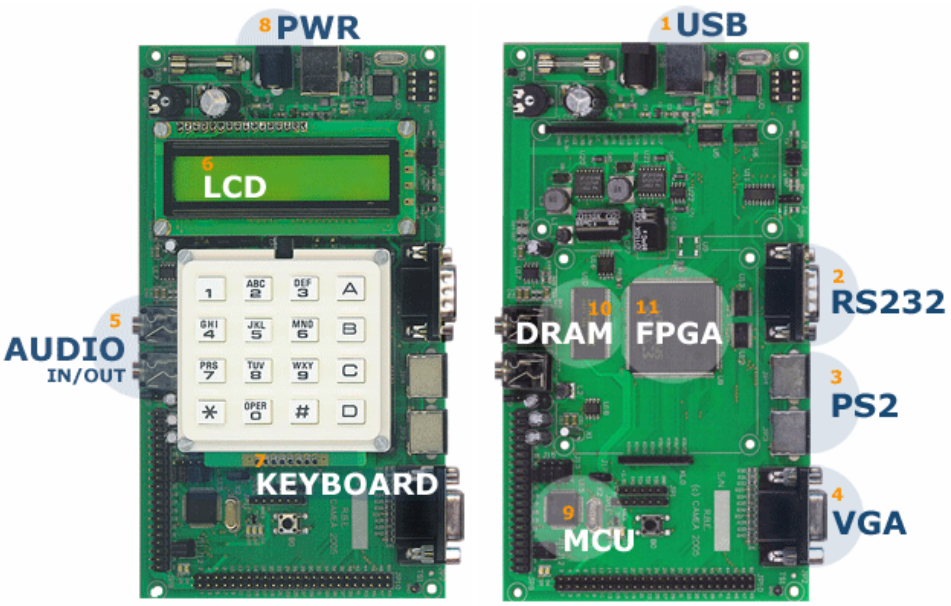
Vývoj aplikace na platformě FITkit je sofistikovaná činnost, která se dělí na dvě části. První částí je vývoj technického vybavení (hardware) v reprogramovatelném obvodu. K tomu účelu slouží programovatelné hradlové pole (FPGA) a případné další podpůrné obvody a periferie. Druhou částí je vývoj programového vybavení (software). U těchto systému se spíše používá označení mikroprogramové vybavení (firmware). Firmware se stará pomocí mikrokontroléru (µC) o řízení běhu systému.

V následující kapitole podrobněji představím právě platformu FITkit. Nejdříve uvedu obecný popis určité součásti a poté uvedu základní parametry součásti použité na fakultním zařízení.

Pro vytvoření aplikace je také zapotřebí znát několik teoretických informací z oblasti sériového rozhraní, které jsou uvedeny v kapitole 1.2. V kapitole 1.3 jsou uvedeny některé modely komunikace a způsoby jejich řízení.

## 1.1 FITkit

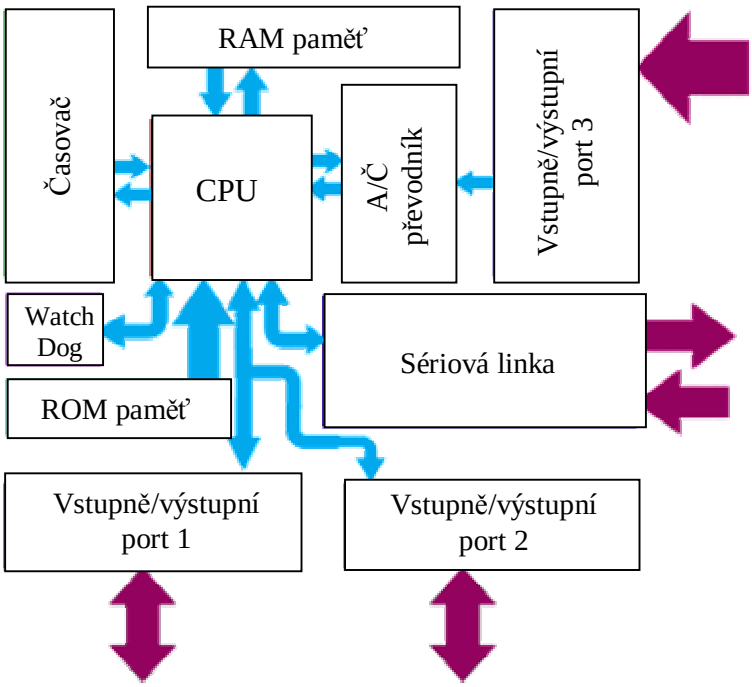
Platforma FITkit obsahuje výkonný mikrokontrolér s nízkým příkonem, FPGA, FLASH paměť, DRAM paměť a řadu periferií. Je napájen pěti volty, které dodává připojený externí adaptér nebo universální sériová sběrnice USB. Pomocí universální sériové sběrnice také probíhá programovaní FPGA a MCU. Konfigurace FPGA i program MCU je možné uložit do FLASH paměti, a tak zajistit při každém restartu aplikace automatické načtení těchto informací. Na obrázku 1.1 jsou znázorněny důležité prvky kitu.




Obrázek 1.1 : Pohled na platformu FITkit

# Mikrokontrolér

Mikrokontrolér neboli MCU (microcontroller unit) je jednočipový integrovaný obvod vysokého stupně integrace s nízkým příkonem. Tyto obvody jsou především určeny pro řízení, regulaci a podobné činnosti různých systémů. Mikrokontrolér v sobě zahrnuje jádro mikroprocesoru s energeticky nezávislou pamětí (nonvolatile memory). To znamená paměť, která dokáže uchovat informaci i bez přísunu elektrické energie (ROM, FLASH, EEPROM). Tato paměť slouží pro uchování mikroinstrukcí programu. Další pamětí je paměť s náhodným přístupem (RAM), která napomáhá při výpočtu jádra. Dále obsahuje řadu periferních obvodů. Mezi důležité periferie patří například analogově číslicové převodníky, čítače, časovače, komunikační linky a také třeba modul pro pulzně-šířkovou modulaci. MCU je možné dělit podle druhu architektury a podle druhu instrukční sady. Na obrázku 1.2 je znázorněno základní jednoduché schéma jednočipového mikrokontroléru.




Obrázek 1.2 : Základní schéma jednočipového mikrokontroléru

Architektury:

- **Von Neumannova** - Jedná se o architekturu, pro kterou je typická společná paměť pro data i program. Toto uspořádání má výhodu v tom, že nepotřebujeme rozlišovat instrukce přístupu k datům a k programu, což vede ke zjednodušení čipu. Zapotřebí je pouze jedna sběrnice. Nevýhodou je ovšem pomalejší přenos obou typů dat po jedné sběrnici, než by tomu bylo u oddělených sběrnicí. Na této architektuře je postaven osobní počítač (PC).
- **Harvardská** - Je architekturou s oddělenou pamětí dat a programu. Návrh této architektury je náročnější o vytvoření druhé sběrnice. Mezi její výhody patří možnost vytvořit různé šířky

sběrnicí. Této možnosti se často využívá. Mezi další výhody určitě patří rychlost vykonávání instrukcí, protože instrukci i potřebná data je možné číst ve stejný okamžik.

Instrukční sady:

- **CISC** – označuje procesor s úplnou instrukční sadou. To znamená, že sada obsahuje instrukce, které se snaží provést co nejvíce operací během jedné instrukce. Vede to sice k úspoře programové paměti, ale na druhé straně se jedná o složitější dekodér instrukcí a tím pomalejší zpracování instrukcí.
- **RISC** – označuje procesor s redukovanou instrukční sadou. Hlavní myšlenkou této architektury je vykonání raději více jednoduchých instrukcí, než jedné komplikované.

Program, který bude mikrokontrolér provádět, je možné psát ve vyšších programovacích jazycích, které jsou později přeloženy na program v mikroinstrukcích. Nejčastěji se používají jazyky assembler a C. Kód psaný pro jeden mikrokontrolér většinou není možné přenést na jiný mikrokontrolér, protože bývá psaný na míru určité architektury. Tím je myšleno například jiné adresovaní periferií, velikost paměti, ….

V sestavě je použit mikrokontrolér Harvardské architektury s označením MSP430F168IPM od společnosti Texas Instruments, který má tyto základní parametry:

- Frekvence 8 MHz
- Nízké napájecí napětí (rozmezí 1.8 V až 3.6 V)
- Nízký příkon
    - 330 μA v aktivním režimu při 1 MHz a 2.2 V
    - 1.1 μA ve stand-by režimu
    - 0.2 μA v režimu vypnuto
- 16-bitová RISC architektura s instrukčním cyklem 125 ns
- Flash Paměť 48 kB + 256 B
- RAM paměť 2 kB
- Moduly na čipu:
    - 3-kanálové DMA
    - Dva 16-bitové časovače
    - Komparátor
    - Sériové komunikační rozhraní USART0 (asynchronní UART, synchronní SPI, I2C),
    - Sériové komunikační rozhraní USART1 (asynchronní UART, synchronní SPI)
    - 12-bitové A/D převodník
    - 12-bitový D/A převodník
    - Přerušení

- Watch dog
- JTAG

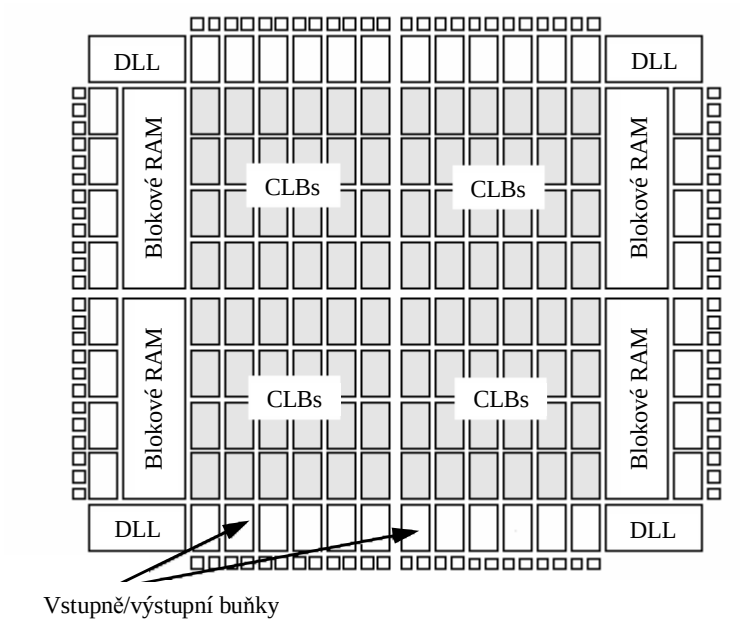
Další informace je možné najít na stránkách výrobce [2].

## Hradlové pole

Nese také označení programovatelné hradlové pole (PLD - programmable logic device). Je to elektronická součástka používaná pro vytváření digitálních obvodů. Na rozdíl od hradel, klopných obvodů a obdobných součástek nemá předem stanovenou funkčnost. Funkčnosti se dosahuje pomocí nahrání konfigurace obvodu do součástky. Popis se provádí pomocí HDL jazyků určených pro popis hardwaru (VHDL, VERILOG, …). Pomocí specializovaných syntetizačních programů je vygenerována konfigurace obvodu, která je následně nahrána do obvodu.

Programovatelný hardware rozdělujeme do několika skupin:

- **GAL** a **PAL** – jednoduché obvody, které se často programují v patici speciálního programátoru.
- **CPLD** – komplexní programovatelná hradlová pole jsou obvody, které jsou zapájeny do desky plošných spojů a až poté jsou například pomocí rozhraní JTAG naprogramovány. Jedná se v podstatě o spojení několika tisíc GAL nebo PAL obvodů spolu s podpůrnými obvody.
- **FPGA** – jsou na místě programovatelné logické obvody (on Field Programmable Gate Arrays). Tyto obvody se od CPLD liší v tom, že po zapnutí napájení musí být vždy jejich konfigurace nahrána například z paměti FLASH nebo EEPROM. Pokud je tato paměť jejich součástí, programují se podobně jako CPLD. Struktura FPGA čipu je zobrazena na obrázku 1.3.
- **ASIC** – je alternativa hradlových polí, používající se u výroby velkých sérií. U tohoto obvodu není možné měnit hardwarovou konfiguraci. Je totiž speciálně navržen pro danou aplikaci. Výhodou ovšem je, že je levnější a má menší spotřebu.




CLBs - konfigurovatelné logické bloky

DLL (Delay Locked Loop) - slouží pro rekonstrukci, případné dělení či násobení vnějších taktovacích signálů

Obrázek 1.3 : Obecná struktura FPGA čipu

Nedílnou součástí FITkitu je FPGA s označením XC3S50-4PQ208C řady Spartan 3 od společnosti Xilinx.

Čip má k dispozici:

- Rozhraní: až 124 uživatelských vstupů/výstupů (I/O), podpora až 23 různých I/O standardů
- 192 konfigurovatelných logických bloků (CLBs) uspořádaných do matice o 16 řádcích a 12 sloupcích
- 1728 logických buněk
- 50000 logických hradel
- 4 dvoukanálové blokové paměti RAM o celkové velikosti 72 kb
- 2 jednotky pro správu hodin (DCMs)
- 4 násobičky 18x18 bitů

Další informace je možné najít na stránkách výrobce [3].

## Periferie

FITkit dále disponuje klávesnicí, pamětí, LCD displejem a mnoha konektory k propojení nejrůznějších periférií. Pomocí FPGA dochází k propojení těchto periférií s mikrokontrolérem.

- Alfanumerická klávesnice typu *AK-1604-A-WWB* zapojená do matice 4x4
- Dynamická paměť s náhodným přístupem (DRAM) o velikosti 8x8 Mb od společnosti Micron
- Šestnácti znakový jednořádkový LCD displej s označením CM1610NR-j2
- Konektor USB – komunikaci přes tento konektor zajišťuje čip FT2232C od společnosti FTDI Chip
- Konektory pro vstupní a výstupní audio signál zpracovávaný čipem TPA6111A2 od společnosti Texas Instruments
- Konektory PS2
- Konektor VGA
- Konektor RS232
- Kolíkové lišty

# 1.2 Rozhraní RS232

V této kapitole je poměrně podrobně popsáno rozhraní RS232 a jeho komunikační protokol, protože se jedná o jednoduchý komunikační prostředek s externími zařízeními, který je použit dále v práci.

Jedná se o sériové rozhraní pro přenos informací dvou zařízení do vzdálenosti přibližně 20 m. Pro generování informace se standardně používá napětí 5V. Hodnoty logických hodnot se za použití násobiče a invertoru transformují na hodnoty -3V až -15V pro logickou „1“ a +3V až +15V pro logickou „0“. Na větší vzdálenosti nebo pro zvýšení odolnosti proti rušení se používají vyšší napětí. Přenos informací probíhá asynchronně a na pevně domluvené přenosové rychlosti (Baud rate). Informace je předávána v tak zvaném přenosovém rámci uvedeném na obrázku 1.4.

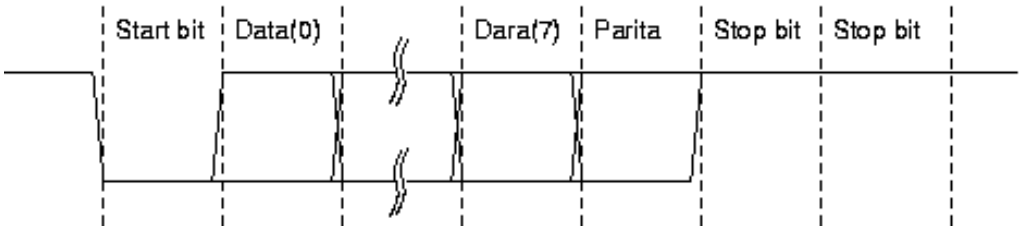
## Synchronní a asynchronní přenos dat

Synchronního přenos dat je způsob datového přenosu, u kterého je zapotřebí zvláštního synchronizačního vodiče. Na tento vodič vysílá jedno ze zařízení (toto zařízení musí být jednoznačně určeno) signál, pomocí kterého přijímající zařízení pozná, že v daný okamžik jsou na datovém vodiči správná data. Toho se s výhodou používá například při větším objemu dat přenášených po více vodičích nebo při přizpůsobení rychlosti přenosu míře chybovosti.

Asynchronní přenos dat je realizován pomocí již zmíněných přenosových rámců. Každé zařízení je vybaveno přesným krystalovým oscilátorem, který udává přesnou pracovní frekvenci.

Tento přenos dat lze s výhodou použít pro komunikaci více zařízení, které se synchronizují pomocí prvního bitu (start bit) přenosového rámce.

## Přenosový rámec




Obrázek 1.4 : Přenosový rámec 8. bitového slova.

- **Start bit** –Slouží pro zahájení komunikace, to znamená, že odesílání může být zahájeno v kterémkoliv okamžiku. Pomocí tohoto bitu, který je definován na úroveň logické „0“, pozná přijímací strana, že bude následovat přenášené slovo.
- **Datové bity** - Hned za start bitem následují po sobě jdoucí datové bity. Nejméně významný bit (LSB) je vysílán jako první. Počet datových bitů může být 5 až 8.
- **Paritní bit** – Pro detekci případného přeslechu na lince se používá kontrolní (paritní) bit. Jeho hodnota se vypočítá z přenášených dat. Přijímající strana potom nad obdrženými daty provede tentýž výpočet a porovná výsledek. Pro výpočet paritního bitu existují dva algoritmy.
    - **Sudá parita** - U sudé parity je vždy součet všech jedniček z datových bitů a paritního bitu sudé číslo. Součet se provádí funkcí XOR a paritní bit se doplní tak, aby jeho výsledek byl roven 0.
    - **Lichá parita** - U liché parity je vždy součet všech jedniček z datových bitů a paritního bitu liché číslo. Součet se provádí funkcí XOR a paritní bit se doplní tak, aby jeho výsledek byl roven 1.
- **Stop bit(y)** – Definuje ukončení rámce a zároveň zajišťuje určitou časovou rezervu pro zpracování dat přijímající stranou. Druhý stop bit se používá u pomalejších zařízení.

## Přenosová rychlost

Přenosová rychlost (také označovaná jako baud rate) udává počet změn signálu za sekundu. Jednotkou je baud (Bd). Vzhledem k tomu, že je možné do jedné signálové změny zakódovat i více než pouze jeden bit, nemůžeme obecně slučovat bity za sekundu (bps) s pojmem baud. Ovšem nejčastěji se přenáší právě jeden bit v jedné signálové změně. Často používané rychlosti jsou 921600 Bd, 460800Bd a další. Na přenosové rychlosti také závisí dálka možného vedení. Se snižující se rychlostí se zvyšuje vzdálenost. Avšak většinou se problém délky vedení řeší jinými technikami, jakými jsou například volba jiného výhodnějšího rozhraní nebo použití proudové smyčky.

## Zapojení RS232

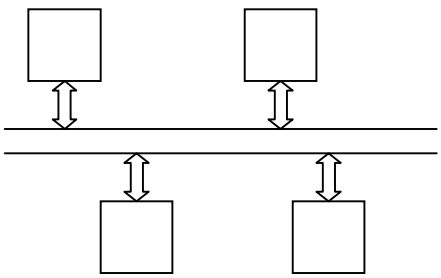
Nejčastěji používaný konektor pro rozhraní RS232 je devíti pinový konektor CANNON označovaný také jako DE-9. V dnešní době se převážně používá plně duplexní komunikace. To znamená, že je možné data jak přijímat, tak vysílat zároveň. Dříve probíhala komunikace pouze pomocí poloduplexního kanálu, a proto bylo zapotřebí několika signálů pro řízení. V tabulce 1.5 je uveden přehled signálů a popis jejich použití.

| **signál** | **popis** |
|---|---|
| DCD - Data Carrier Detect | Signál pro oznámení detekce nosného kmitočtu |
| RXD - Receive Data | Signál pro příjem informace |
| TXD - Transmit Data | Signál pro vysílání informace |
| SGND - Signal Ground | Signálová zem |
| RI - Ring Indicator | Indikátor zvonění použitý například u modemů. |
| DSR - Data Set Ready | Signály pro řízení již zmíněné poloduplexní komunikace, jinak také označované jako hardwarový handshaking. Existuje více variant hardwarového řízení toku, například “hardware (RTS/CTS)”, “hardware (DTR/DSR)” apod. |
| RTS - Request to Send | |
| CTS - Clear to Send | |
| DTR - Data Terminal Ready | |

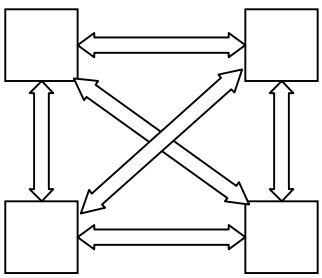
Tabulka 1.5 : Popis signálů

# 1.3 Modely komunikace

Důležitou roli v modelu komunikace hraje způsob propojení komunikujících zařízení. Jedná se například o následující topologie: sběrnice, kostka, hvězda, kruh a peer to peer. Cílem této práce není popsat všechny tyto způsoby, ale přiblížit si hlavně ty nejpoužívanější. Prvním popisovaným je propojení za použití sběrnice. Tím druhým je propojení každý s každým (point to point).



Obrázek 1.6 : Sběrnice



Obrázek 1.7 : Point to point

Použití některého způsobu propojení závisí na počtu připojených zařízení, na požadované propustnosti, na pořizovací ceně a dalších aspektech. Například propojení point to point se hodí při

menším počtu zařízení a má vysokou propustnost, naproti tomu je zapotřebí poměrně velké množství vodičů a tím je jeho pořizovací cena vyšší. Častěji používaná je sběrnice, která je tvořena relativně nízkým počtem sdílených vodičů. Zde je možné uplatnit způsob komunikace, kdy jedna komponenta vysílá informace a všechny ostatní mohou přijímat. Parametry sběrnicí podstatným způsobem ovlivňují výkonnost celého systému.

Rozdělení sběrnicí:

- Podle typu přenášených dat
    - Adresová
    - Datová
    - Řídící
- Podle umístění v systému
    - Interní
    - Externí
- Podle způsobu přenosu
    - Sériové
    - Paralelní
- Podle způsobu synchronizace
    - Synchronní
    - Asynchronní
- Další možné rozdělení
    - Multiplexovaná
    - Dedikovaná

Další důležitou vlastností sběrnice je způsob jejího přidělovaní (arbitrace) takovému zařízení, které bude v daný okamžik navazovat komunikaci. Pokud víme, kdo bude komunikovat, musíme také vědět s kým. Toho se docílí pomocí adresace. Každé zařízení je obvykle připojeno ke sběrnici pomocí řadiče. Ten zajišťuje spolu s rozpoznáním adresy také převod dat na reprezentaci srozumitelnou zařízení. Například převod ze sériového přenosu dat na paralelní nebo naopak. V následujících kapitolách jsou vedeny některé typy komunikace.

## Master – slave

Model komunikace master - slave (pán - otrok) je častým zástupcem sběrnicové topologie. Master slouží k řízení sběrnice a začíná komunikaci. To znamená, že si vybírá s kým a o čem bude komunikovat (dotaz - odpověď). Naproti tomu slave pouze čeká na splnění požadavku zadaného masterem.

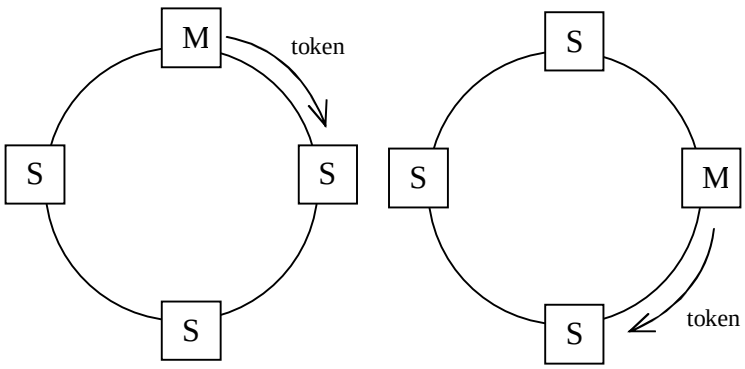
Mezi nejjednodušší možnosti patří taková architektura, ve které se nachází pouze jeden prvek typu master a jeden typu slave. U této architektury není zapotřebí žádného adresování ani arbitra (dedikovaná sběrnice). Tento způsob zapojení se vyznačuje vysokou rychlostí akceptace požadavku. O něco složitější je architektura, která má v systému pouze jeden prvek typu master a ostatní prvky typu slave. Protože je možných cílových prvků více, musí master zajistit adresaci svého cíle. Pokud je více prvků, které by chtěly řídit sběrnici, nastupuje na řadu tak zvaný arbitr (soudce). Ten podle jistých pravidel určí, které zařízení bude v daný okamžik moci řídit sběrnici. Funkce arbitra bude podrobněji vysvětlena později.

## Peer to peer

„Peer to peer“ doslovně přeloženo znamená „rovný s rovným“. Patří mezi představitele topologie „point to point“. Jedná se o způsob komunikace, která je často používaná především v oblasti počítačových sítí. Každý prvek je schopen zároveň zadávat i plnit požadované úkony. To znamená je schopen plnit funkci master i slave z předchozí kapitoly.

## Token ring

Token ring je kruhová topologie vyvinutá společností IBM, která se v dnešní době již v podstatě nepoužívá. Jde o to, že zařízení jsou zapojena do kruhu a vždy je pouze jedno z nich typu master. Rozhodnutí o tom, kdo bude řídit sběrnici, určuje tak zvaný „token“. Token je datový rámec (informace), který pravidelně obchází po kruhu a při dokončení požadavku jednoho prvku přejde na další prvek systému (spravedlivá arbitrace).




Obrázek 1.8 : Arbitrace pomocí tokenu

# Přehled způsobů arbitrace

Arbitrace neboli rozhodování o řízení sběrnice při současné žádosti několika jednotek dělíme do několika skupin.

Tyto skupinou jsou :

- Prioritní rozhodování
    - prioritní dekodér
    - prioritní linka
- Spravedlivé rozhodování
    - udržování požadavků na sběrnici ve frontě (FIFO)
    - funkce cyklického token
- Nahodilé rozhodování
    - náhodný výběr
    - detekce kolize s náhodnou dobou čekání

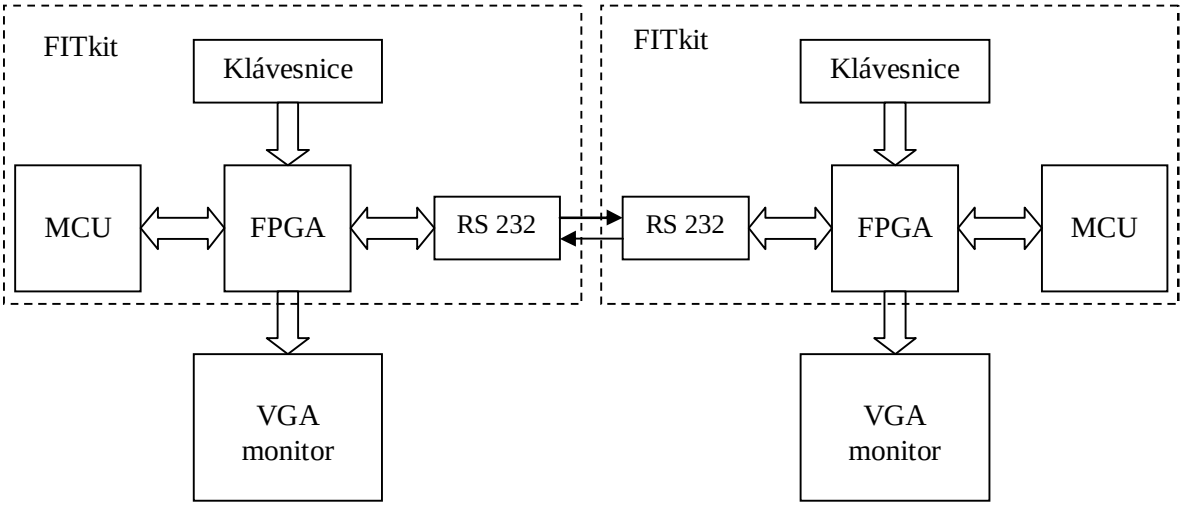
# 2 Návrh

V první kapitole jsou vestavěné systémy popsány převážně obecně, pro přiblížení jejich výhod a nevýhod. Ovšem v kapitolách následujících je pohled směrován spíše na platformu FITkit, o kterou především v této práci jde. V kapitole 2.1 je stručně popsána požadovaná funkčnost aplikace. Kapitola 2.2 popisuje rozvrhnutí výpočtu na jednotlivé výpočetní zdroje popsané obecně a poté rozvrhnutí výpočetních zdrojů speciálně pro platformu FITkit a vyvíjenou aplikaci. Součástí rozvrhnutí výpočtu je i schéma konečného automatu části, která bude řídit chod celého programu. V kapitolách 2.3 a 2.4 je řešeno vykreslování textur a jejich pohyb. Další částí návrhu je zvolení modelu komunikace a způsobu arbitrace. Toto je uvedeno v kapitole 2.5. Výpočet minimální přenosové rychlosti sériové linky je uveden v kapitole 2.6.

## 2.1 Obecný popis aplikace

Zadáním této bakalářské práce je synchronizovat grafické aplikace běžící na dvou FITkitech. Grafická aplikace představující hru „tenis“ je pomocí vykreslování textur zobrazována na VGA monitoru. Hra „tenis“ spočívá v tom, že každý hráč ovládá pálku, která slouží k odražení kuličky směrem na soupeře. Na hrací ploše je možné pohybovat pálkami do všech směrů. Vše se ovládá pomocí klávesnice umístěné na FITkitu. Cílem hry je dostat kuličku za soupeřovu pálku a tímto způsobem získat deset bodů. Aplikace je doplněna o nabídku programu. Nabídka podává informace o autorovi a způsobu používání aplikace. Dále také zajišťuje komunikaci s uživatelem a chod programu. Vzájemná komunikace dvou zařízení probíhá pomocí sériové linky. Na obrázku 2.1 je znázorněno obecné blokové schéma společného zapojení. K propojení dvou FITkitů je zapotřebí použít křížený sériový kabel s konektory RS232.

Při vývoji této aplikace je nutné vyřešit dostupnost zdrojů FITkitu, zprostředkování dat, ovládání hry, zobrazování textur, pohyb kuličky, způsob a rychlost komunikace a mnoho dalšího.




Obrázek 2.1: Obecné blokové schéma společného zapojení

## 2.2 Výpočetní zdroje

V první fázi návrhu aplikace si je zapotřebí uvědomit s jakými výpočetními zdroji je možné počítat při návrhu. Mezi nejpodstatnější výpočetní zdroje lze zařadit mikrokontrolér a programovatelné hradlové pole. Dále pak například velikost dostupné externí paměti a dostupné periférie. Každý výpočetní zdroj má určité výhody a nevýhody.

Mezi hlavní výhody mikrokontroléru patří především jeho snadná použitelnost z pohledu programátora (využití vyšších programovacích jazyků). Oproti tomu je programátor omezen spoustou parametrů výpočetní jednotky, jako jsou například šířky sběrnicí, frekvence procesoru, velikost programové pamětí, dostupné periferie, velikosti registrů a spousta dalších. U reprogramovatelného hardwaru je to obdobné. Výhodou bezesporu je jeho možnost popsání hardwarovým jazykem (HDL) a jeho snadné ladění a testování. Další výhodou je rychlost operací prováděných ve speciálně navrženém obvodu. Je ovšem také omezen různými parametry, jako je například počet logických bloků, počet programovatelných buněk, počet hradel a velikost dostupné paměti.

Mnoho operací lze popsat jak na programové úrovni ,tak i na úrovni obvodové. Návrh funkčnosti na obvodové úrovni je prováděn především z důvodu rychlosti odezvy nebo z důvodu specifických požadavků, které mikrokontrolér nenabízí. Návrh na programové úrovni je oproti tomu snazší, protože využívá již vytvořených obvodů. Například pokud aplikace vyžaduje sčítání desetibitových čísel a mikrokontrolér obsahuje pouze sčítačku osmibitovou, nezbývá nic jiného, než sestavit desetibitovou sčítačku v hardwaru. Často realizovaným obvodem je funkce konečného automatu, kterou sice lze vytvořit programově, ale nedosahuje takových rychlostí odezvy.

Dalším možným rozvržením zdrojů je v systému obsahujícím více jednotek se stejnou funkčností použití paralelního zpracovaní. Důležitým aspektem je u takovýchto systémů vzájemná komunikace.

# Návrh použití zdrojů platformy FITkit

Vzhledem k tomu, že vyvíjená aplikace není aplikací první a je zde možné čerpat z již vytvořených projektů nebo z připravených součástí, tak dalším hlediskem při návrhu jsou parametry některých komponent, které budou používány. Tímto je myšleno například použití již vytvořeného řadiče VGA monitoru, sériového rozhraní RS232, klávesnice a tak podobně.

Parametry použitých komponent:

- **VGA řadič** – nabízí tři režimy zobrazování. První číslice udává šířku zobrazení v obrazových bodech, druhá výšku zobrazení v obrazových bodech a třetí hodnotu frekvence zobrazení v jednotkách Hz.
    - 640x480x60
    - 640x480x66
    - 800x600x72

  Jednotlivé režimy zobrazení potřebují pro zobrazení během jedné vteřiny různý počet obrazových bodů. Například u režimu 640x480x60 je to 18432000 bodů. U režimu 800x600x72 je to už téměř dvojnásobek a to 34560000 bodů. Aby mohly být všechny tyto body včas zobrazeny, je u každého režimu potřeba jiná hodnota vstupních hodin. Například u prvního uvedeného režimu je to 25MHz a u posledního už 50MHz. Dalším důležitým parametrem VGA řadiče je počet bitů jednotlivých barevných složek. Tato hodnota je omezena možnostmi použitých digitálně analogových převodníků a paměťových možností na FITkitu. Používaný řadič používá devíti bitové zakódování obrazového bodu. Ten je tvořen složkou červené, zelené a modré barvy (RGB). Každá složka je uložena na třech bitech. Z toho vyplývá, že jsme schopni zobrazit až 512 různých barev.

- **Řadič sériového rozhraní RS232 –** je schopen pracovat na několika rychlostech přenosu udávaných v jednotkách Bd (Baud). Tyto hodnoty jsou dány řadou v rozmezí od 1200Bd, která pokračuje vždy dvojnásobkem předchozí hodnoty až do hodnoty 921600Bd. Vzhledem k tomu, že v jedné signálové změně je přenášen pouze jeden bit, je možné použít místo jednotek Baud (Bd) jednotku bit za sekundu (b/s). Dalším parametrem je skladba přenosového rámce. Tím je myšleno, jaká je použita parita, kolik je datových bitů a kolik je stopbitů.

Požadavky na vyvíjenou aplikaci nejsou nějak náročné, a proto je zvoleno zobrazení s nejmenšími nároky na výpočetní výkon. Tím zobrazením je režim s šířkou 640 obrazových bodů, výškou 480 obrazových bodů a frekvencí 60 Hz.

Hlavním problémem při zobrazování je uložení textur v paměti. Texturou se rozumí grafika vytvořená pomocí kódování používaného na FITkitu (3 červené bity, 3 zelené bity a 3 modré bity). Pro využití paměti se nabízí několik možností.

Možnosti jsou následující:

- **Uložení v paměti SDRAM** – SDRAM je synchronní dynamickou pamětí o velikosti 8MB. Je tvořena čtyřmi nezávislými bloky (banky). Pro ovládání této paměti je zapotřebí do aplikace zakomponovat řadič. Komunikace s výpočetním jádrem přes systémovou sběrnici je poměrně značná a tím celý systém brzdí. Pomocí třiadvaceti bitové adresy je možné zapsat, do kteréhokoli místa v paměti osmi bitová data.
- **Uložení ve FLASH paměti** – Tato paměť slouží především jako úložiště programu mikrokontroléru a konfigurace FPGA, ale lze ji použít i jako běžně používanou paměť, jakou je třeba SDRAM. Nastává ovšem obdobný problém s přenášením dat po systémové sběrnici.
- **Uložení v blokové paměti FPGA** – Bloková paměť označovaná jako BRAM, jak již bylo zmíněno v kapitole o obvodu FPGA, je také rozdělena na čtyři bloky. Její velikost je ovšem pouze 72 kb. Dále také bylo řečeno, že se jedná o paměť typu RAM, která po ukončení přísunu energie uloženou informaci ztrácí. To znamená, že je potřeba zajistit nahrání textury při každém startu aplikace. Toho lze docílit nahráním textury například z FLASH paměti. Oproti tomu je hlavní výhodou rychlost paměti a možnost použití této paměti bez řadiče. Také je možné pracovat s celým rozsahem paměti pomocí šestnácti bitové adresy a ukládat devíti bitová data. Netradiční devátý bit datové složky lze dostat využitím paritního bitu.

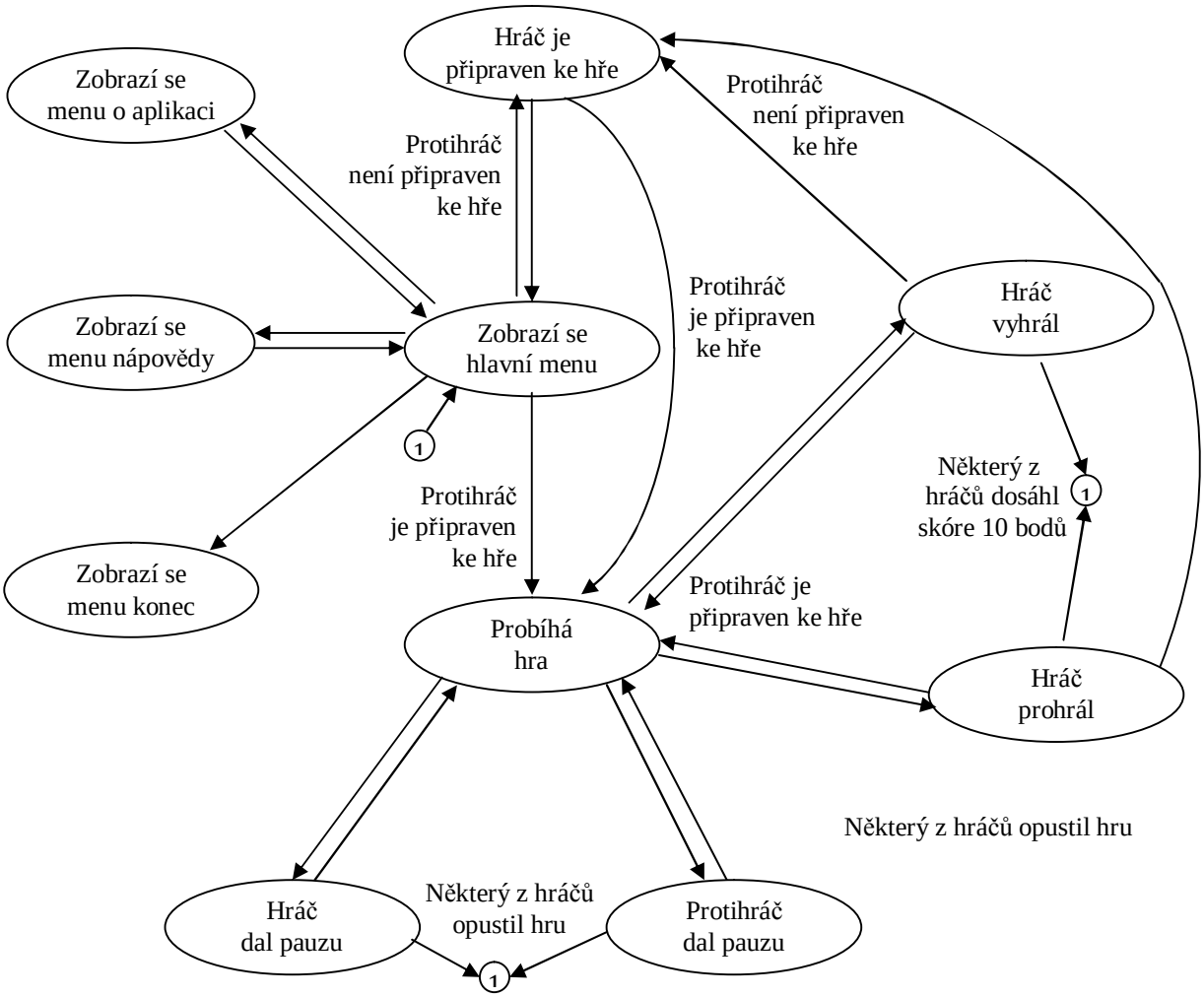
V zadání funkčnosti aplikace je požadována grafická aplikace představující hru „tenis“ a textová nabídka zobrazená pomocí textur. Protože se bude jednat o aplikaci využívající malý počet opakujících se textur, je zvoleno ukládání v blokové paměti BRAM, která je znatelně rychlejší a jednodušší. Do jednoho bloku paměti (18kb) je možné uložit texturu o velikost až 32x64 obrazových bodů.

Pro vytvoření programové nabídky (zobrazení ASCII znaků) je zapotřebí tak zvaná mapa znaků a video paměť. Pro jednoduchost vytvoření nabídky je použita již vytvořená mapa znaků, ve které je uloženo 128 znaků. Každý znak je definován 8x16 bity, což také udává velikost znaku v obrazových bodech. To znamená že pro uložení celé mapy znaků postačí jeden blok BRAM (16kb). Aby bylo možné zaplnit celou obrazovku znaky bylo by potřeba 80 znaků na 30 řádcích. Video paměť, která slouží pro zapamatování určitého znaku na určitém místě, tedy bude uchovávat 7 bitové informace udávající hodnotu příslušného znaku. Sedmi bitové proto, že se jedná o mapu tvořenou 128 znaky. Rozšíření abecedy by vyžadovalo další paměťové prostory. Velikost video paměti pro zobrazení jedné obrazovky je potom 2400 slov. Slovo zde představuje 7bitů, ale stejně jako ostatní informace je uloženo minimálně na osmi bitech. Tudíž je zapotřebí dvou pamětí BRAM. Konečný automat pro vykreslování textur na určité pozici obrazovky bude implementován z důvodu rychlosti přímo v FPGA.

Tímto je vyřešeno zobrazování. Vzhledem k tomu, že není požadována nějaká přesně stanovená garantovaná rychlost výpočtu, může výpočet souřadnic jednotlivých textur probíhat

v aritmeticko logické jednotce mikrokontroléru. Řízení celé aplikace, které bude představovat konečný automat, bude také řešeno na programové úrovni. Na obrázku 2.2 je nakresleno schéma takového konečného automatu.

V rychlosti komunikace probíhající mezi FITkity, která bude realizována po již zmíněném rozhraní RS232, nejsou stanovena žádná omezení a tak lze použít nejvyšší realizovatelnou rychlost. Rychlost komunikace by mohla být omezena například z důvodu komunikace rozdílných zařízení nebo sdílením sběrnice více zařízeními. Zvolená rychlost tedy činí 921600Bd, i když by stačila rychlost mnohem menší. Výpočet minimální přenosové rychlosti je uveden v kapitole 2.6.

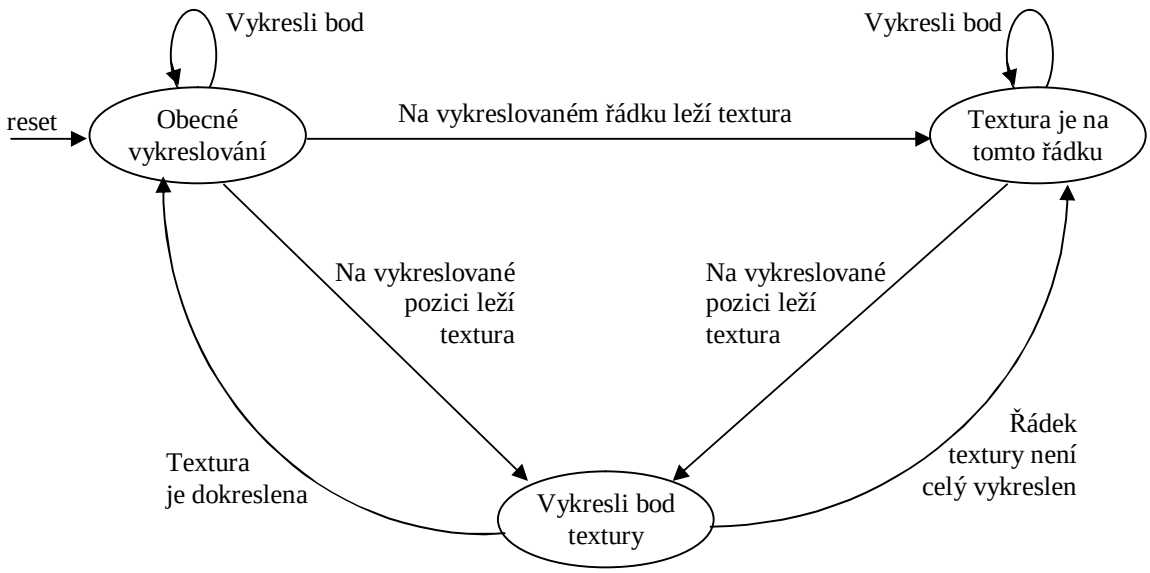



Ostatní nepopsané přechody mezi stavy, představují pohyb po struktuře nabídky aplikace

Obrázek 2.2: Schéma konečného automatu pro řízení chodu programu

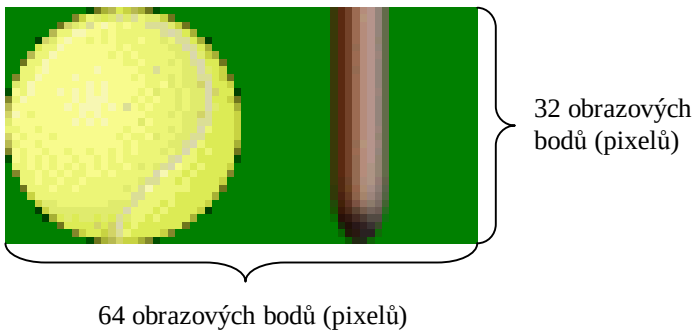
## 2.3 Vykreslování textur

Důležitou částí u grafické aplikace je způsob vykreslování na obrazovku. Pro správné vykreslení na obrazovku je nutné připravit řadiči grafického adaptéru ve správný okamžik správná data. Jelikož se jedná o operace probíhající nepřetržitě, je v podstatě nemožná softwarová realizace a nabízí se vhodnější možnost v podobě konečného automatu realizovaného v hradlovém poli. Konečný automat na základě informací o poloze textur a vykreslovaném bodu rozhodne, který bod se bude vykreslovat. Zda-li to bude bod obsažený v jedné z textur nebo vykreslí bod pozadí, který má neustále stejnou hodnotu. Na obrázku 2.3 je znázorněn popisovaný konečný automat.




Obrázek 2.3 : Schéma konečného automatu pro vykreslování textury

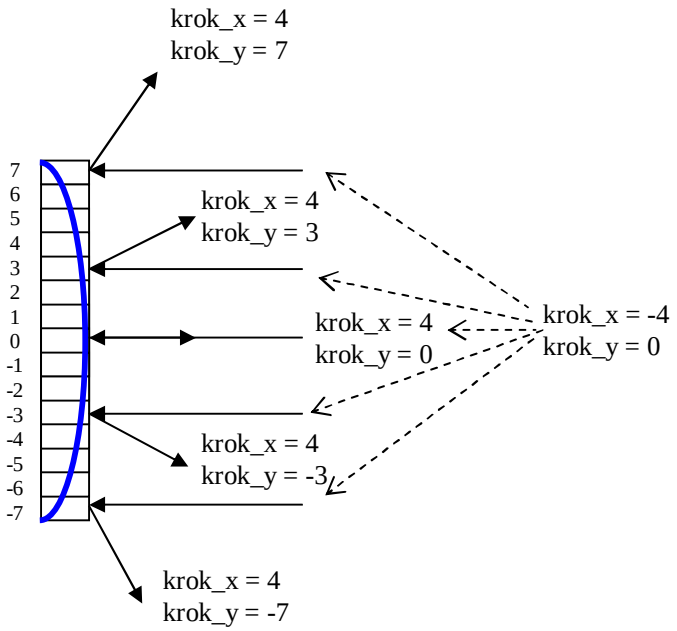
Protože je během hry nutné vykreslovat dvě pálky, které jsou symetrické, a jeden míček, je naprosto dostačující textura uvedená na obrázku 2.4. Další výhodou takto navržené textury je fakt, že pro její uložení postačuje jedna bloková paměť BRAM. Textura byla vytvořena jako 24 bitový rastr (BMP) a následně převedena pomocným programem napsaným v jazyce (C++) na devíti bitovou reprezentaci.




Obrázek 2.4 : Použitá textura

## 2.4 Výpočet souřadnic kuličky

Pro upřesnění pravidel hry jsou níže uvedena určitá pravidla pro výpočet souřadnic kuličky a skóre. Cílem hry je jako první získat deset bodů. Bod získáte tím, že dostanete kuličku za soupeřovu pálku. Pálkou je možné pomocí klávesnice pohybovat ve všech směrech. Popis ovládání aplikace je uveden v následují kapitole. Pohyb kuličky lze samozřejmě rozdělit na horizontální a vertikální. Pohyb po horizontální ose (osa X) je konstantní a mění pouze svůj směr, a to tehdy, pokud kulička narazí na některou z pálek. Pohyb po vertikální ose (osa Y) je mnohem složitější. Přírůstky na ose Y nejsou konstantní, jak je tomu u osy X, ale jsou závislé na předchozím odrazu. Odraz kuličky je možný od horizontální okrajové hrany hrací plochy nebo od pálek. Odraz od hrany je podstatně jednodušší a mění pouze směr, nikoli velikost kroku. Velikost kroku na ose Y se mění pouze po nárazu na pálku a její změna je závislá na místě dopadu. Rozdělení pálky a způsob změny kroku simuluje odraz od kulové plochy a je uveden na obrázku 2.5.




Obrázek 2.5: Rozdělení pálky a způsob změny kroku

## 2.5 Návrh komunikace

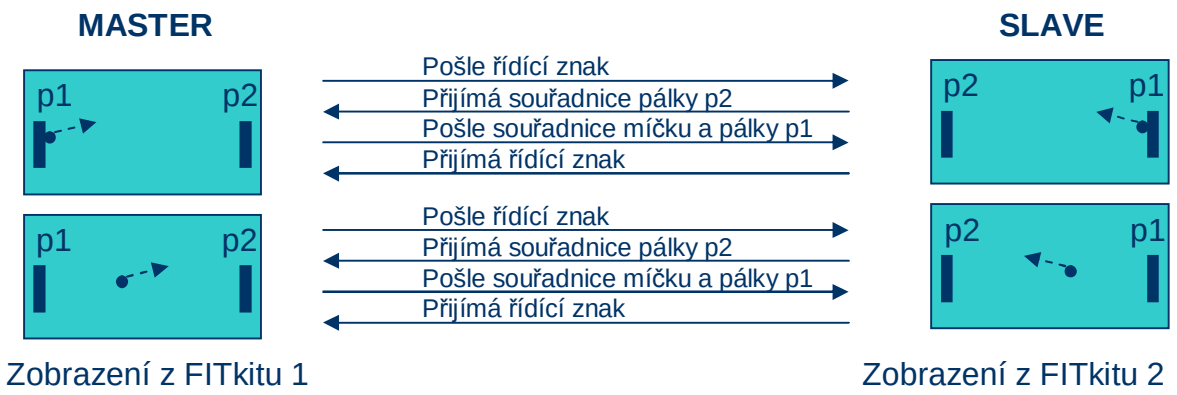
V kapitole 1.3 jsou představeny některé možné druhy komunikace. Z důvodů, které uvedu v následujících odstavcích, je zvolena komunikace master-slave.

Prvním důvodem je možnost provádět pomocí zařízení typu master synchronizaci činnosti celého systému. Synchronizace je zaručena tím, že například zařízení přijímající požadavky (slave) nezačne s vykreslováním na obrazovku dříve, než to povolí řídicí prvek systému (master). Aby byly obě strany informovány o tom, ve kterém z možných stavů hry se soupeř právě nachází, je důležité

zasílat řídicí informace (znaky). Tyto znaky druhé zařízení dekóduje a podle nich případně změní svůj stav hry.

Dalším důvodem je fakt, že se jedná o komunikaci dvou zařízení, které provádějí některé společné výpočty (výpočet souřadnic). Díky výpočtu na jednom zařízení získáme kontrolu nad shodou zobrazovaných informací. Komunikace potom může vypadat tak, že jedno zařízení přijme data potřebná pro výpočet, vypočítá a pošle výsledky druhému zařízení. Tato možnost komunikace je znázorněna na obrázku 2.6.

Použití komunikace každý s každým je sice možné řešení, ale v tomto případě hůře použitelné. Protože by výpočet probíhal na obou mikrokontrolérech, považuji toto řešení za zbytečné. Hůře použitelné proto, že by bylo nutné zajistit kontrolu vypočítané hodnoty a synchronizaci jinými technikami. Použití modelu komunikace token ring je pro dvě zařízení nevhodné a zbytečné.




Obrázek 2.6: Návrh komunikace

Model komunikace už je vybrán, a tak je teď nutné nějak rozlišit jednotlivé FITkity. Přesně řečeno označit jedno zařízení jako master a jedno jako slave. Jelikož se jedná o systém dvou zařízení, není zapotřebí složitého hardwarového arbitra, ale je možné použít arbitraci softwarovou. Pro použití programové arbitrace dvou zařízení je dostačující a vhodná skupina arbitrací s nahodilým rozhodováním. Použití nahodilého rozhodování spočívá v tom, že například každé zařízení vygeneruje náhodné číslo. Číslo slouží jako časový interval, po který zařízení čeká, než pošle určitý znak druhému zařízení. Pokud provede odeslání znaku dříve než soupeř, stává se po přijetí znaku soupeře masterem. Tímto postupem je také zajištěna kontrola, zda-li je druhé zařízení vůbec připojeno. Dalším možným způsobem náhodného rozhodování je vygenerování náhodného čísla a jeho následné odeslání soupeři. Každý FITkit si porovná své vygenerované číslo s číslem přijatým a podle jistého pravidla (menší nebo větší) se rozhodne, jakou funkci v systému zastává.

## 2.6 Výpočet minimální rychlosti RS232

Při výpočtu je vycházeno z předpokladu, že vzdálenost horizontálních okrajů hracího plánu je 640 obrazových bodů. Pohyb textury probíhá konstantním krokem (2, 4 nebo 8 obrazových bodů), který je možný v programu změnit. Implicitní nastavení je hodnota 4. Pro tento krok je níže uveden vypočet minimální potřebné rychlosti přenosové linky.

Přepokládejme tyto hodnoty parametrů:

- Velikost kroku na horizontální ose jsou 4 obrazové body
- Šířka hrací plochy je 640 obrazových bodů

Z těchto hodnot lze vypočítat počet kroku nutných k překonání celé hrací plochy.

```
Počet kroků = šířka hrací plochy / velikost kroku = 640px / 4px = 160
```

Nyní je možné vypočítat rychlost přenosové linky sériového rozhraní RS232 (baud rate) pro určitou rychlost kuličky. Místo rychlosti, která by byla v obrazových bodech za sekundu, je použit pouze čas potřebný k překonání dráhy kuličky z jedné strany na stranu druhou. Hodnota času má mnohem větší vypovídací hodnotu a lze si ji snadněji představit.

Přepokládejme tyto hodnoty parametrů:

- Počet kroků je 160
- Velikost přenosového rámce je 11 bitů (1xSTART bit, 8xDATA, 1xPARITA a 1xSTOP bit)
- Počet rámců je 14
    - 2x řídicí znak = 2 rámce (řídicí znak má 8 bitů - char)
    - 6x souřadnice = 6*2rámce = 12 rámců (souřadnice x i y má 16 bitů - int)
- čas potřebný pro přeběhnutí kuličky u jedné strany na druhou je zvolen na 5s

```
baud rate = počet kroků * velikost rámce * počet rámců / doba k přeběhnutí
baud rate = 160 * 11 * 14 / 5 = 4928 Bd  => nejbližší vyšší je 9600 Bd
```

Z výsledku tohoto výpočtu lze říci, že nastavení rychlosti přenosové linky záleží na zvoleném kroku a požadované době k překonání hrací plochy. Pro hodnoty zvolené ve výše uvedeném příkladě je naprosto dostačující hodnota 9600 Bd.

# 3 Implementace

Ve vývojovém cyklu po návrhu následuje implementace. Systém byl vyvíjen souběžně jak na obvodové, tak i na programové úrovni. V následující kapitole jsou stručně popsány přípravné práce před započetím samotné implementace. Dále je přiblížen návrh hardwaru a jsou popsány použité komponenty a způsob jejich propojování. Kapitola 3.3 obsahuje informace o návrhu programového vybavení a přehled funkcí a popis výpočtu souřadnic kuličky.Následující kapitola obsahuje popis, jak uvést aplikaci do chodu a jak jí ovládat.

## 3.1 Instalace programového vybavení

Aby bylo možné s implementací aplikace začít, bylo zapotřebí připravit počítač, na kterém bude vývoj probíhat. Na notebooku, který byl k tomuto účelu použit, byl nainstalován operační systém Windows XP Professional od společnosti Microsoft. Po prvním připojení FITkitu k počítači bylo zapotřebí nainstalovat ovladač universální sériové sběrnice pro čip FT2232C. Výsledkem instalace jsou dva uměle vytvořené porty sériové komunikace (COM). Jeden z portů slouží pro komunikaci s mikrokontrolérem. Druhý slouží pro komunikaci s programovatelným hradlovým polem. Komunikace na obou portech probíhá pomocí terminálu, který je možné spustit například programem „Hyperterminal“. Tento program je součástí operačního systémů, ovšem pro jeho požadovanou funkčnost je nutné správně nastavit hodnoty jednotlivých sériových portů. Instalaci lze provést i na operačním systému Linux. Podrobnější informace jsou uvedeny na internetových stránkách [4].

Dalším krokem je instalace nástrojů pro vytváření a překlad programů napsaných ve VHDL a C. Pro návrh konfigurace FPGA slouží software, který je dostupný na webových stránkách společnosti Xilinx [2]. Tímto programovým balíkem je „ISE WebPACK“ a lze ho získat bezplatně po potvrzení licenční smlouvy. Jedná se o rozsáhlý balík obsahující programy pro návrh, ladění a syntézu konfigurace přímo určené pro konkrétní integrovaný obvod. Pro překlad programu mikrokontroléru napsaných v jazyce C je zapotřebí nainstalovat překladač tohoto jazyka speciálně vytvořený pro daný typ procesoru (pro danou instrukční sadu). Program „mspgcc-win32“ je možné také získat zadarmo. Obsahuje program pro překlad, ladění a některé jiné činnosti s vývojem spojené.

Pro dobrou ovladatelnost a nenáročnost na výkon počítače byly všechny zdrojové kódy psány v editoru „PSPad“ a překládány pomocí skriptů napsaných pro program „make“. Použité skripty spolu s knihovnou platformy FITkit a se zdrojovými kódy již vytvořených aplikací, jsou uloženy v úschovně. O úschovnu se stará verzovací systém SVN dostupný z internetových stránek projektu .

## 3.2 Realizace obvodové části

Popis konfigurace FPGA psané v jazyce VHDL je rozdělena do dvou základních částí. První částí je popis entity. Entita obvodu definuje rozhraní. To znamená, že popisuje vstupy a výstupy obvodu a také obsahuje parametry pro generický (obecný) návrh. Parametry mohou například udávat hodnotu určující šířku sběrnice. Druhou částí je popis architektury. Architektura obsahuje VHDL kód, který popisuje funkčnost celého obvodu. Architektura se dále dělí na část deklarační a část příkazovou. Pro jednu entitu může existovat více architektur.

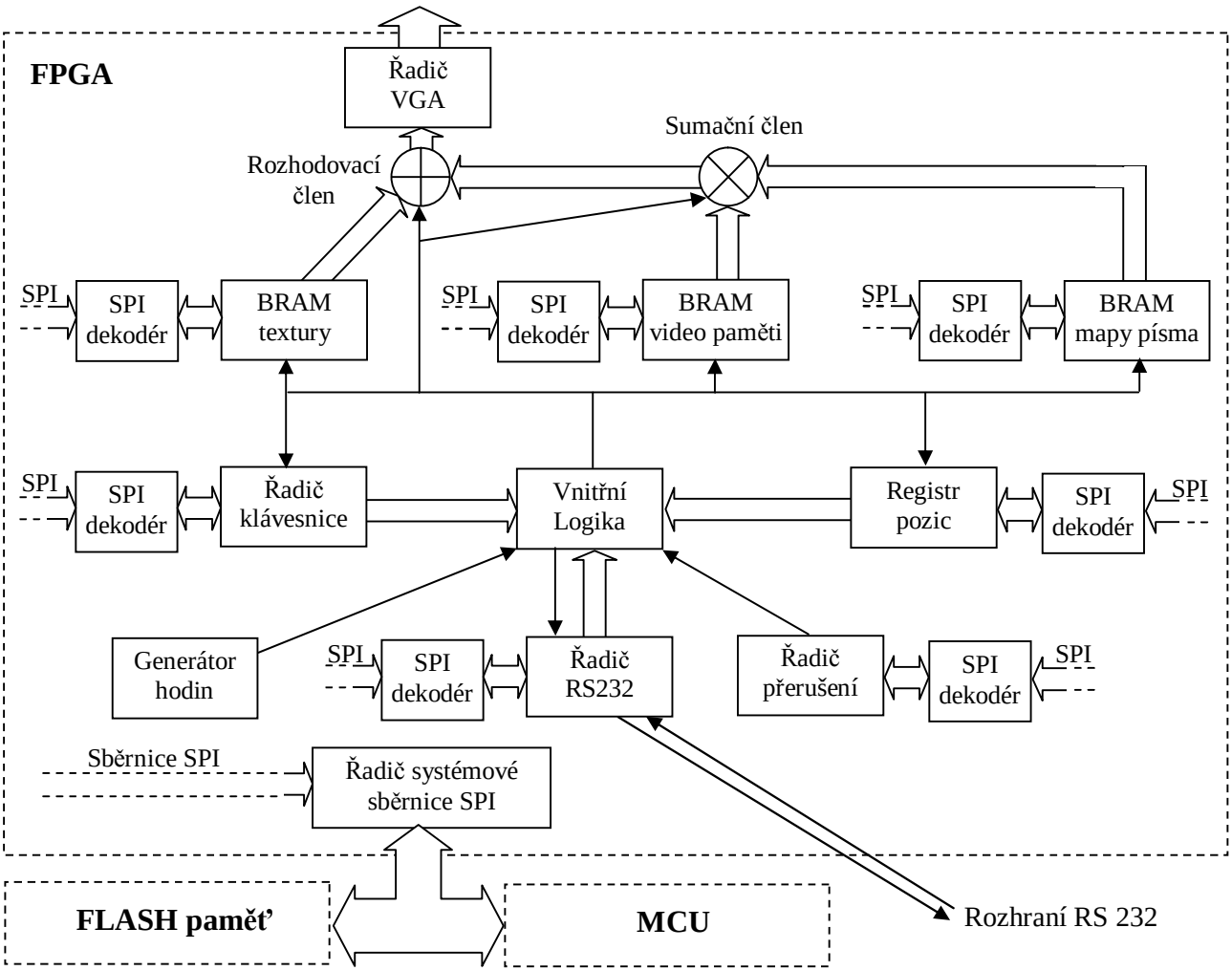
Vývoj na FITkitu se odvíjí od entity nejvyšší úrovně (top level entity). V úschovně verzovacího systému je možné najít tyto připravené entity.

Entity nejvyšší úrovně:

- **Entita s označením tlv_bare_ifc** – je určena pro jednoduché aplikace, které nepotřebují přistupovat ke vstup-výstupní sběrnici X
- **Entita s označením tlv_pc_ifc** – je určena pro aplikace využívající rozhraní VGA, PS2 nebo RS232
- **Entita s označením tlv_ide_ifc** – je určena pro aplikace, které potřebují přistupovat ke sběrnici X například jako k portu IDE

Pro realizaci projektu je použita entitu s označením „tlv_pc_ifc“, protože nabízí propojení rozhraní, která jsou zapotřebí. Asi nejdůležitější částí vnitřní struktury obvodu je sběrnice SPI. Ta slouží k propojení MCU, FLASH paměti a komponent vytvořených v FPGA. Na sběrnici SPI probíhá sériový synchronní přenos typu master-slave. Role mastera obstarává mikrokontrolér, který pomocí určitého protokolu přistupuje k ostatním zařízením. Každá komponenta vytvořená v hardwaru, a která má být připojena ke sběrnici SPI, musí navíc obsahovat dekodér. Dekodér je synchronní komponenta, která převádí vnitřní sériovou komunikaci na paralelní rozhraní, jež umožňuje komunikaci se zařízením.

Výše zmíněný protokol je rozdělen na tři části. První částí je odeslání operačního kódu. Operační kód je tvořen osmi bity, předává informace SPI řadiči a určuje, které zařízení bude komunikovat (FLASH nebo FPGA). Po operačním kódu následuje adresace cílového zařízení. Velikost adresy může být libovolně nastavená podle požadavků dané aplikace. Je nutné si však uvědomit, že ze strany procesoru dochází k přenosu osmi bitových dat. Poslední částí protokolu je přenos dat. Velikost dat je také libovolná a záleží přímo na dané aplikaci. SPI umožňuje současné čtení i zápis dat. Nastavení bázové adresy zařízení je provedeno při vytváření SPI dekodéru daného zařízení nastavením parametru „BASE_ADDR“.




Obrázek 3.1: Schéma propojení použitých komponent

Na obrázku 3.1 je uvedeno schéma propojení komponent použitých ve vytvářené aplikaci. Jednoduché šipky představují hlavní řídicí signály. Plné šipky představují sběrnice a paralelní přenos dat. Uvedené schéma je zjednodušené a orientační. Komponenta nazvaná vnitřní logika představuje prvek, který řídí způsob použití všech komponent. Tento prvek je ovšem značně abstraktní a ve skutečnosti je funkčnost rozmístěna po celém FPGA a představuje několik konečných automatů a další důležité prvky. Detail konečného automatu, který zajišťuje zobrazení textury, je uveden na obrázku 2.3. v kapitole 2.3. Dále na schématu vidíte sumační a rozhodovací člen. Sumační člen znázorňuje obvod, který předá do řadiče data představující určitý znak. Znak je nutné vybrat takový, jaký se má podle video paměti na vykreslovaném místě nacházet. Prvek nazvaný rozhodovací člen rozhoduje o způsobu vykreslování. Rozhoduje, má-li probíhat vykreslování v grafickém nebo v textovém režimu. Grafický režim je použit na vykreslování textur. Textový režim slouží na vykreslování nabídky programu a používá výstup z již zmíněného sumačního členu. Níže jsou představeny jednotlivé komponenty a způsob jejich použití.

Přehled komponent:

- **Generátor hodinových impulzů** – je obvod rozšiřující možnosti standardního hodinového signálu o kmitočtu 7.3728MHz. Na základním kmitočtu pracuje například sběrnice SPI. Určitým nastavením obvodu lze dosáhnout až frekvence 100MHz. Hodnota generovaného kmitočtu je ve vytvářené aplikaci 25MHz a slouží pro vykreslování grafického adaptéru. Pod tímto taktem je také nutné přistupovat do pamětí, které uchovávají vykreslované informace.
- **Bloková RAM paměť pro uložení textury** – obsahuje texturu uloženou v devíti bitové reprezentaci. Vzhledem k tomu, že jsou BRAM paměti dvoukanálové, je možné s pamětí pracovat na dvou různých kmitočtech. První, již zmíněný kmitočet 25MHz, slouží k přenosu dat mezi pamětí a VGA řadičem. Druhý kanál pracuje na základním kmitočtu a slouží k propojení s SPI, které zajišťuje dekodér SPI sběrnice. Přes tento dekodér jsou do paměti při každém restartování zařízení nahrána data textury, která jsou uložena ve FLASH paměti.
- **Bloková RAM paměť pro uložení masky písma** – je velmi obdobná s předchozí pamětí. Jediný rozdíl je v tom, že uchovává informace o masce písma v osmi bitové reprezentaci.
- **Bloková RAM paměť pro uložení video paměti** – pracuje na stejných kmitočtech jako předchozí paměti. Do této paměti jsou ale data nahrávána průběžně za chodu programu. Šířky adres všech blokových pamětí jsou jedenácti bitové.
- **Řadič klávesnice** – zpracovává stisknutá tlačítka na klávesnici a pomocí SPI dekodéru spolupracuje s mikrokontrolérem.
- **Registr pozic** – slouží jako úložiště informací o aktuálním umístění textur. Jedná se o 64 bitový registr a přistupuje se k němu prostřednictvím SPI dekodéru.
- **Řadič přerušení** – představuje komponentu, která na základě žádosti řadiče sériového rozhraní generuje globální žádost o přerušení. Globální žádost předává přímo do MCU. Přerušení je možné maskovat.
- **Řadič sériového rozhraní RS232** – je komponenta složená ze dvou základních řadičů. Jedná se o řadič pro odesílání dat a řadič pro příjem dat. Pomocí generických parametrů jsou nastaveny vlastnosti. Vlastnosti jsou nastaveny tak, že při přenosové rychlosti 921600Bd je přenášen jeden start bit, osm datových bitů, jeden bit liché parity a jeden stop bit. Řadič generuje žádosti o přerušení, které následně zpracovává řadič přerušení.
- **Řadič grafického adaptéru VGA** – generuje signály pro VGA port. To znamená, že převádí informace o obrazovém bodu, které jsou uloženy na devíti bitech, na analogový signál. Řadič poskytuje dvojici signálů, která informuje o číslech právě vykreslovaného sloupce a řádku.
- **Řadič sériové sběrnice SPI** – je prvek, který zajišťuje adresaci vnitřní sběrnice SPI a tím umožňuje komunikaci mikrokontroléru a libovolné komponenty implementované v reprogramovatelném obvodu FPGA.

## 3.3 Realizace programové části

Hardware popsaný v předchozí kapitole je pro obě zařízení totožný. Jelikož je hlavním úkolem programu provést synchronizaci obou zařízení, je nutné rozlišit zařízení typu master a zařízení typu slave. K tomuto účelu slouží jedna z funkcí obsažená ve zdrojových souborech. Jednotlivé funkce budou popsány později. Dále je také nutné řídit chod celé aplikace. Toho je docíleno realizací konečného automatu, který je uveden na obrázku 2.2 v kapitole 2.2.

Vývoj programu na platformě FITkit je velice obdobný jako u běžného programu. Ovšem spolu se standardními knihovnami jazyka C vývojář používá knihovnu „libfitkit“. Tato knihovna obsahuje základní funkce nezbytné pro práci s mikrokontrolérem. Zajišťuje komunikaci s terminálem a obsahuje funkce pro nahrání dat do FLASH a FPGA.

Každý program psaný pro platformu FITkit musí obsahovat hlavní funkci (main), ve které je potřeba zavolat funkci na inicializaci hardwaru a definovat hlavní programovou smyčku. V hlavní programové smyčce je nutné volat funkci, která zajišťuje obsluhu terminálu. Další funkce, bez které se program neobejde, je funkce pro dekódování uživatelských příkazu terminálu. Aby nedošlo k zaseknutí celé aplikace, je chod programu hlídán jednoduchým mechanismem zvaným „programový watchdog“. Princip spočívá v tom, že je sledován programový čítač instrukcí a pokud čítač dosáhne určité hodnoty, je proveden restart programu. Restart není nutné provést, pokud program prochází přes kontrolní body, které zajistí nulování čítače. Kontrolní body se píší do nekonečných smyček, kterou může být například hlavní programová smyčka.

Díky knihovně jsme schopni pracovat s mikrokontrolérem, pamětí FLASH, sběrnicí SPI a dalšími prvky, ale neumíme pracovat z řadiči. Proto je nutné vždy připojit k programu ovladač daného řadiče, který poskytuje funkce pro jeho ovládání.

Program vytvořený pro požadovanou aplikaci je rozdělen do dvou zdrojových souborů. Níže je uveden přehled funkcí, které se v těchto souborech nacházejí. U každé funkce je jednou větou popsána její činnost. Obsáhlejší popis funkcí je na tomto místě zbytečný, protože lze nahlédnout do zdrojových kódů.

Přehled funkcí ze souboru „main.c“:

- **user_help** – vypisuje uživatelskou nápovědu na terminál
- **USER_CMD_Decode** – dekóduje uživatelské příkazy
- **multiplayer_init** – provede arbitraci
- **USER_FPGA_interrupt** – provede obsluhu přerušení z FPGA
- **user_INIT_FPGA_after_prog** – provede inicializaci hardwarových komponent po naprogramování FPGA
- **main** – hlavní funkce

Přehled funkcí ze souboru „vga.c“:

- **get_state** – získá stav soupeře
- **Keyboard_FPGA** – provede obsluhu klávesnice
- **game_FSM** – obsahuje konečný automat, který řídí celý program
- **check** – zkontroluje vzájemné pozice kuličky a pálek a podle potřeby změní směr pohybu kuličky
- **VGA_Clear** – provede vyčištění video paměti
- **VGA_TextOut** – provede zapsání textu do video paměti
- **TEX_Flash_FPGA** – provede zapsání textury z FLASH paměti do blokové paměti BRAM
- **FONT_Flash_FPGA** - provede zapsání mapy znaků z FLASH paměti do blokové paměti BRAM
- **print_menu** – voláním funkce VGA_TextOut zapíše určité menu do video paměti
- **print_main_menu_sel** – zajistí označení určité položky hlavní nabídky
- **print_int_menu_sel** – zajistí označení určité položky v menu přerušení

## 3.4 Zprovoznění a ovládání hry

Pro spuštění samotné hry je zapotřebí toto:

- Dva funkční FITkity
- Dva VGA monitory
- Dva napájecí zdroje nebo kabely USB
- Propojovací křížený sériový kabel
- U starších typů FITkitů dvě drátové propojky, které slouží k propojení dvou pinů na desce plošných spojů. Tato propojka zajistí správnou funkci systému přerušení.

Pokud je vše správně zapojené, lze přistoupit k programovaní zařízení.

Postup zprovoznění:

- Přeložení a nahrání programu do MCU
  - zadejte příkaz `make load` ve složce `sw`
- Vytvoření konfigurace pro FPGA
  - zadejte příkaz `make` ve složce `top`
- Propojení propojky J6 v případě, že není propojena
  - propojte propojky podle postupu uvedeného na webových stránkách projektu
- Nahrání konfigurace do FPGA
  - spusťte terminálový program a navažte spojení

- zadejte příkaz **`flash w fpga`** a pomocí "File transfer" a protokolu "xmodem" nahrajte soubor **`output.bin`** z adresare **`top`**
- zadejte příkaz **`flash w tex`** a pomocí "File transfer" a protokolu "xmodem" nahrajte soubor textury **`tenis.bin`** ze složky **`comp`**
- zadejte příkaz **`flash w font`** a pomocí "File transfer" a protokolu "xmodem" nahrajte soubor masky písma **`font.bin`** ze složky **`comp`**
- zapsáním příkazu restart proveďte **`restart`** FITkitu

Tento postup je nutné provést pro oba FITkity. Jakmile jsou obě zařízení připraveny, je možné spustit hru. Toho docílíte tak, že v časovém intervalu maximálně 5 sekund provedete jejich restart. Restart je možný provést zapsáním příkazu do terminálu nebo resetovacím tlačítkem. Pokud vše probíhalo v pořádku a zobrazila se obrazovka s úvodním menu, můžeme přistoupit ke hře. Pro pohyb po nabídce slouží tlačítka **`1`** a **`4`**. Pro potvrzení tlačítko **`#`**. Ovládání pálky během hry je nastaveno na tlačítka **`2`**, **`4`**, **`5`** a **`6`**. Pro vyvolání pauzy slouží tlačítko **`9`**.

# 4 Závěr

Během této práce jsem se blíže seznámil s velmi zajímavým odvětvím informačních technologií. Vestavěné systémy jsou v dnešní době řídícím prvkem téměř každého elektronického zařízení, a tak získané znalosti a zkušenosti se mohou v budoucnu hodit při hledání práce.

Výsledkem této bakalářské práce je aplikace, kterou je možné použít při prezentaci platformy nebo jako prototypové řešení komunikace při řešení obdobných projektů. Vzhledem k tomu, že se jedná o zařízení na kterém neustále probíhá vývoj, narazil jsem na určité problémy. Problém týkající se paměti FLASH se nepodařilo vyřešit ani za pomoci mého vedoucího ,a proto jsme na to upozornili osoby zabývající se vývojem platformy. Náhradním řešením problému bylo nahrávání konfigurace přímo do FPGA namísto nahrávání z FLASH paměti. Řešení sice bylo časově náročnější a obtížněji realizovatelné, ale bylo možné. Naopak značnou úsporu času při vývoji poskytuje opětovné použití některých komponent z předchozích projektů. Aplikace je z části postavena na spojení dvou projektů. Tyto projekty se zabývají vykreslováním textur a zobrazováním v textovém režimu.

Pro vývoj aplikace bylo zapotřebí prostudovat především materiály týkajících se vestavěných systémů a vzájemné komunikace a také se seznámit s platformou FITkit a již vytvořenými projekty. Po seznámení s jistými fakty bylo možné zahájit vlastní vývoj, který spočíval v návrhu a následné implementaci řízení aplikace, komunikace zařízení a způsobu vykreslování. Konfigurace nahraná v FPGA nepokrývá celý čip, ale využívá přibližně tři čtvrtiny logických buněk (slice) a 3 blokové RAM paměti. Těchto volných prostředků lze využít při tvorbě nějakého rozšíření.

Zadáním aplikace nebyla přesně určeny pravidla hry, proto jedno z možných rozšíření by se mohlo zabývat například vytvořením jiných pravidel (proměnná rychlost kuličky, způsob odrážení kuličky,…). Vzhledem k tomu, že ovládání hry je realizováno na klávesnici FITkitu, nabízí se další možné rozšíření. Například by bylo možné implementovat řadič rozhraní PS2 a ovládat hru pomocí klasické počítačové klávesnice nebo myši.

# Zdroje


[1] FITkit, internetové stránky, http://merlin.fit.vutbr.cz/FITkit/

[2] Texas Instruments, internetové stránky, http://www.ti.com

[3] Xilinx, internetové stránky, http://www.xilinx.com

[4] návody FITkit, internetové stránky, http://merlin.fit.vutbr.cz/FITkit/navody.html

# Seznam příloh

Ukázka úvodní obrazovky


Ukázka obrazovky při hře