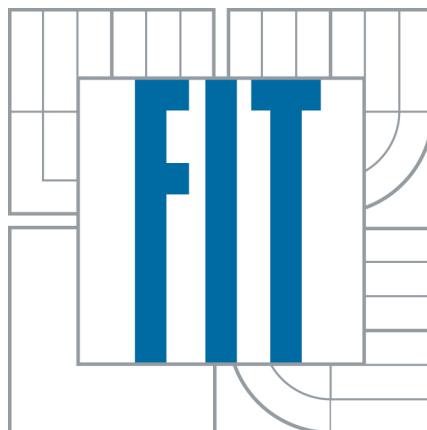


Fakulta informačních technologií

VUT v Brně



Seminář VHDL

Dokumentace k přehrávači SID

2. června 2014

Autor: Jan Pawlus, xpawlu00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1	Popis problému	1
2	Úvod do generování zvuku	1
2.1	Zvuk	1
2.2	Oscilátor	1
2.3	Generátor obálky a amplitudová modulace	1
3	Implementace	2
3.1	Paměť FLASH a MCU	2
3.2	Vstup dat do FPGA a registr SID čipu	2
3.3	Oscilátory	3
3.4	Generátor obálky	3
3.5	Amplitudová modulace	4
4	Blokový diagram	4
5	Závěr	5

1 Popis problému

Mým úkolem je vytvořit přehrávač SID skladeb. Je zde nutné mít znalosti s generováním zvuku za pomoci PWM modulu (ten jsem měl možnost si sám osadit) a poté zkonstruoval automat, jenž bude zpracovávat skladbu ve formě not (výška, délka trvání, typ) uloženou v paměti FLASH. Vzhledem k formě not (.dmp soubory) se můj projekt velmi silně inspiroval čipem SID6581, ale oproti originálu je hlavně jednodušší.

2 Úvod do generování zvuku

2.1 Zvuk

Zvuk je vlnění o určité frekvenci. Lidské ucho rozpozná vlny o frekvenci v intervalu zhruba od 16 Hz do 18 000 Hz (tento interval je však velmi subjektivní). Čím vyšší frekvence, tím vyšší tón. Jako příklad uvedu klasický jednobitový tón - ten může nabývat hodnot pouze 0 nebo 1 (*obdélník*). Čím vícekrát se ale za jednu sekundu změní z 0 na 1 a zpět, tím vyšší nám přijde (například tzv. *komorní a* - tón o frekvenci 440 Hz, podle kterého se mimo jiné ladí nástroje v orchestru). Větším počtem tónů najednou (mixování) dosáhnou tak, že je jednoduše sečtu a vydělím počtem tónů.

2.2 Oscilátor

Zvukový signál generuje oscilátor. K projektu potřebuji 3 oscilátory. Každý z nich může generovat jiný tón - kromě výše popisovaného obdélníkového průběhu vlny například také *pílu*, *trojúhelník* a *šum*. Právě tvar křivek určuje barvu tónu.

2.3 Generátor obálky a amplitudová modulace

Obálka má podíl na parametrech zvuku - určuje jeho průběh hlasitosti v čase. Pomocí obálky lze zvukový signál upravit tak, aby se do značné míry podobal tónu reálného hudebního nástroje. Ke svému projektu stačí ADSR obálka, která se skládá ze čtyř částí: *attack* (udává dobu, za kterou se signál dostane na maximum), *decay* (udává dobu, za kterou se signál dostane na úroveň sustain), *sustain* (konstantní velikost signálu), *release* (postupný útlum signálu).

3 Implementace

Vzhledem k tomu, že některé části přehrávače jsou popsány již v přednáškách nebo ukázkách k předmětu, budu popisovat hlavně věci, které v ukázkách nejsou, a to nahrávání písniček do FPGA přes FLASH, ovládací registr SID čipu (převod zdrojového souboru písničky na relevantní informace pro přehrávání) a ADSR obálku.

3.1 Paměť FLASH a MCU

Pro variabilitu přehrávače je nutné zajistit bezproblémové nahrávání souborů .dmp ("noty") do FLASH paměti. Tuto úlohu řeší mikrokontrolér. Pokud nahraji projekt do FitKitu pomocí QDevKitu a následně zadám do konzole příkaz `HELP`, dozvim se, že pro nahrání souboru do FLASH paměti slouží příkaz `FLASH W X`. Takto se nahrávají soubory za sebou. Pokud by uživatel chtěl celou paměť vymazat, přidal jsem do nápovědy položku `FLASH ERASE ALL`, která celou paměť smaže.

V souboru `main.c` jsem si vytvořil několik proměnných datového typu `unsigned int - buf`, `reg`, `offset` a `page` (všechny proměnné kromě `page` jsou nastavené na začátku na 0 - proměnná `page` je nastavená na počátek uživatelské FLASH paměti pomocí hodnoty `FLASH_USER_DATA_PAGE`). Následuje nekonečný cyklus, ve kterém je třeba postupně načítat z FLASH paměti jedno osmi-bitové číslo do proměnné `buff` pomocí funkce `FLASH_ReadData`.










FLASH paměť je rozdělená na stránky a pro pohybování po stránce slouží proměnná `offset`. Kdykoliv je tato proměnná větší než velikost stránky, `page` se inkrementuje o 1 a `offset` se dekrementuje o velikost stránky. Pokud nastane tato situace (přechod na další stránku), je nezbytné ověřit, zda nenastal konec písničky, a to funkcí `FLASH_isPageBlank`. Pokud funkce vrátí 1, je aktuální stránka prázdná a je potřeba vrátit se na začátek - všechny proměnné tedy nastavím na počáteční hodnoty.

Pokud stránka není prázdná, může se načtená hodnota v proměnné `buff` poslat přes rozhraní SPI do FPGA pomocí funkce `FPGA_SPI_RW_AN_DN`. Toto se opakuje 25x, poté se čeká 20 ms a cyklus jede dál (vysvětlení v další kapitole).

3.2 Vstup dat do FPGA a registr SID čipu

Přenos dat z FLASH do FPGA řeší rozhraní SPI, které jsem ale nijak detailněji nezkoumal. Důležité jsou tyto signály, přicházející do FPGA: `di` - 8bitový signál - samotná data, `we`, `cs` a `addr` - první dva jsou ovládací, třetí udává pořadí přichozích dat.

V kapitole uvedené výše zmiňuji, že cyklus posílání dat se opakuje 25x, poté se počká 20 ms a následují další iterace. 25 osmi-bitových dat za sebou totiž tvoří jednu "sadu" informací pro všechny tři oscilátory generující 3 zvuky současně. MCU počká 20 ms (noty jsou uzpůsobeny právě tomuto časovému prodlení) a pošle se další sada. První dvě informace tvoří 16bitovou frekvenci, další dvě šířku pulsu (to ve mém projektu ignoruju), pátá je signál `control` (bit 0 udává hodnotu `GATE` důležitou pro ADSR obálku, bit 4 zda je tón typu trojúhelník, 5 pila, 6 obdélník a 7 šum. Bity 1 - 3 jsou pro můj projekt nepodstatné), šestá a sedmá jsou informace pro ADSR obálku. SID generuje a následně mixuje 3 tóny, těchto 7 osmi-bitových informací musí být v jedné sadě zároveň pro 3 oscilátory, je tedy vysvětleno 21 z 25 informací. Další tři jsou pro filtr, který ale ve mém projektu nepoužívám, a poslední signál `filter_vol` je pro amplitudovou modulaci, která na rozdíl od obálky zvyšuje hlasitost celého tónu.

Reg # (Hex)	Data								Reg Name
	D7	D6	D5	D4	D3	D2	D1	D0	
VOICE 1									
00	F7	F6	F5	F4	F3	F2	F1	F0	Freq Lo
01	F15	F14	F13	F12	F11	F10	F9	F8	Freq Hi
02	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW LO
03	—	—	—	—	PW11	PW10	PW9	PW8	PW HI
04	NOISE				TEST	RING MOD	SYNC	GATE	Control Reg
05	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attack/Decay
06	STN3	STN2	STN1	STN0	RIS3	RIS2	RIS1	RIS0	Sustain/Release
VOICE 2									
07	F7	F6	F5	F4	F3	F2	F1	F0	Freq LO
08	F15	F14	F13	F12	F11	F10	F9	F8	Freq Hi
09	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW LO
0A	—	—	—	—	PW11	PW10	PW9	PW8	PW HI
0B	NOISE				TEST	RING MOD	SYNC	GATE	Control Reg
0C	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attack/Decay
0D	STN3	STN2	STN1	STN0	RIS3	RIS2	RIS1	RIS0	Sustain/Release
VOICE 3									
0E	F7	F6	F5	F4	F3	F2	F1	F0	Freq Lo
0F	F15	F14	F13	F12	F11	F10	F9	F8	Freq Hi
10	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	PW LO
11	—	—	—	—	PW11	PW10	PW9	PW8	PW HI
12	NOISE				TEST	RING MOD	SYNC	GATE	Control Reg
13	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attack/Decay
14	STN3	STN2	STN1	STN0	RIS3	RIS2	RIS1	RIS0	Sustain/Release

Struktura registru SID čipu

3.3 Oscilátory

Jelikož je třeba instancovat 3 oscilátory a 3 generátory obálky, nejlepším řešením je vytvoření si komponenty v souboru `sid6581.vhd` (pojmenoval jsem ji `voice_gen`) a pro lepší přehlednost popis této komponenty v novém souboru `voice_gen.vhd`. Oscilátory jsem použil stejné, jako v jedné z ukázek ke generování zvuku, nebudu tedy zbytečně rozebírat implementaci. Lehce jsem je upravil, aby je je ovládal signál `control`.

3.4 Generátor obálky

Vzhledem k formátu registru SID čipu a `.dmp` souborů nelze udělat obálku jiným způsobem, než jak byla vytvořena v originálním SID6581. Mám k dispozici osmibitové signály `env_acc_dec` (horní čtyři bity reprezentují délku náběhu *attack*, dolní *decay*) a `env_sus_rel` (horní *sustain*, dolní *release*) a nultý bit ze signálu `control` `GATE`. Čtyřbitová hodnota parametru *attack* určuje dobu náběhu v rozsahu 2 ms až 8 sekund (za předpokladu, že čip řídí hodinový signál 1 MHz - v opačném případě by bylo rozdělení času samozřejmě jiné), hodnoty parametrů *decay* a *release* určují dobu trvání sestupných hran v rozsahu 6 ms až 24 sekund (sice se u nich též používá pouze čtyřbitová hodnota, ale všechny časy jsou třikrát delší). Hodnota parametru *sustain* představuje zlomek amplitudy v rozsahu 0/15 (tj. ticho – na výstupu z bloku *Amplitude modulator* je stále signál s nulovou amplitudou) až 15/15 (odpovídá 100 %).

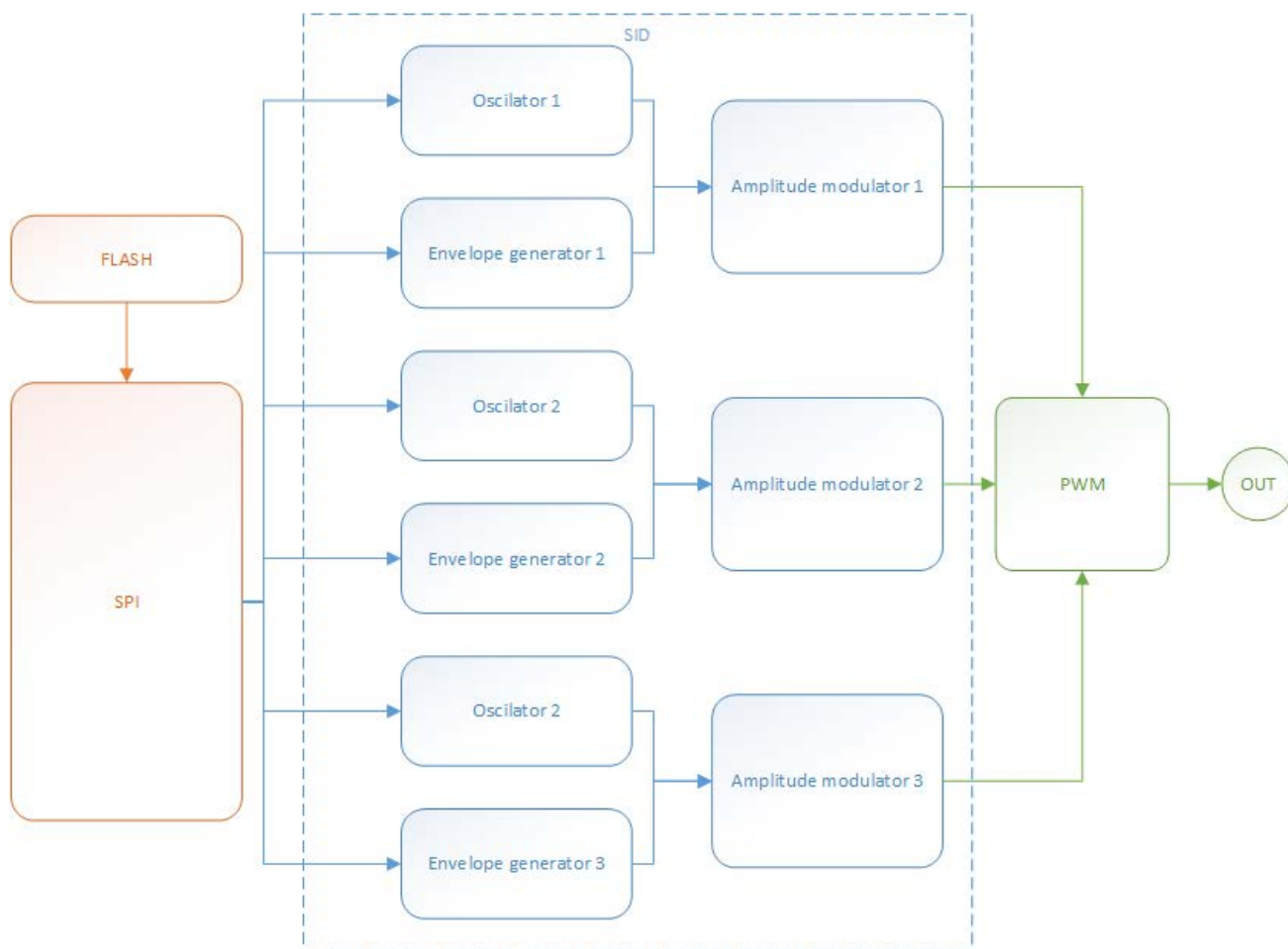
Doba trvání úrovně *sustain* je určena bitem `GATE`. Ve chvíli, kdy je tento bit nastaven na logickou jedničku, spustí se cyklus *attack* (obálka se začne měnit dle náběžné hrany), za nímž automaticky následuje cyklus *release* (první sestupná hrana) až do chvíle, kdy se úroveň obálky ustálí na hodnotě *sustain*. Tato hodnota se udržuje tak dlouho, dokud má bit `GATE` hodnotu logické jedničky. Ve chvíli, kdy se programově tento bit vynuluje, začne probíhat cyklus *release*, tj. úroveň obálky se snižuje až k nule.

3.5 Amplitudová modulace

Výsledný tón vznikne vynásobením signálu vycházejícího z oscilátoru a signálu z generátoru obálky (výsledek je rozšířený na 20 bitů). Výstup z generátoru obálky je hodnota čítače inkrementujícího se s nástupnou hranou hodin, jehož maximální úroveň je určená tabulkou (tabulky jsou ovládány signály, které jsou zmíněny v kapitole výše). Výsledek (3 zmixované tóny) jdou přes PWM na výstup. Tento princip je ukázán v přednáškách/ukázkách, nebudu jej tedy rozepisovat. Nakonec se ještě celý tón vynásobí signálem `filter_vol`.

4 Blokový diagram

Blokový diagram nepotřebuje další zvláštní popis, jelikož jej v podstatě popisuje kapitola Implementace.



5 Závěr

Tento projekt jsem si vybral, protože mi přišel problém generování zvuku docela zajímavý. Před implementací jsem musel čerpat hodně teorie z internetu, hlavně strukturu registru SID čipu. Když jsem ale kompletně pochopil, jak SID čip vlastně funguje, už jsem neměl žádný větší problém s napsáním projektu. Malý problém mi dělala ADSR obálka, kterou jsem chtěl původně implementovat jinak - k tomu bych ale potřeboval informaci *velocity*, která se ve struktuře registru SID čipu nenachází (obálka je tedy generována na základě časových intervalů šitých na míru frekvenci 1 MHz). Nahrávání do FLASH a následný přenos do FPGA jsem zkonstruoval na základě ukázek k přednášce o zvuku.

Video ukázka písničky Ghostbusters zde: <http://www.youtube.com/watch?v=epVjXYjaZ9o>