Vysoké učení technické v Brně

Fakulta informačních technologií

IVH

Úvodní část dokumentace k projektu

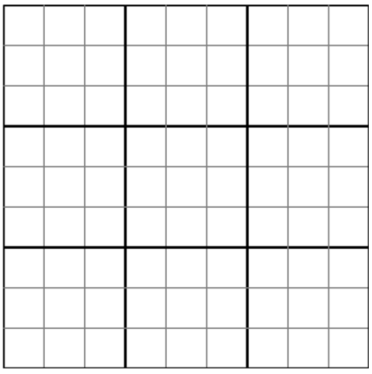
Návrh a popis řešení

Petr Stehlík
xstehl14

V Brně 1.6. 2015

Zadání

Projektem je tvorba logické zábavné hry Sudoku. Sudoku je logická hra s jedním velmi jednoduchým pravidlem. Číslo 1 - 9 se nesmí opakovat v řádku, sloupci a čtverci 3x3. Hrací je pole je rozděleno dle následujícího schématu:



Pole je rozděleno na 9x9 políček a v každém políčku se může nacházet číslo od 1 do 9. Hrací pole je následně rozděleno na dílčí čtverce o rozměru 3x3 políčka.

Úkolem je vytvořit grafickou aplikaci pomocí FPGA a MCU, která vytvoří dané hrací pole, vygeneruje korektní řešení pomocí permutace existující vyřešené matice a uživateli zobrazí pouze určitou část řešení, kterou musí následně uživatel sám doplnit.

Paradigmatem v našem případě je zejména tvorba unikátní hrací matice a kontrola správného uživatelského řešení.

Ovládání hry

Hra se ovládá pomocí alfanumerického bloku na FitKitu, kde písmena A, B, C, D umožňují pohyb po poli a numerickými klávesami uživatel vkládá čísla do pole. Klávesa “*” (hvězdička) zobrazí uživateli nápovědu v podobě korektního čísla v daném políčku a dalším stisknutím dané číslo opět schová.

Kontrola správnosti probíhá za běhu hry, tudíž hráč ihned ví, zda je jeho řešení správné.

Implementace

Generování matice čísel v MCU

Matice čísel se generuje z již vytvořených korektních řešení, které se následně permutují a tím dokáží vytvořit vždy unikátní řešení. Lze implemetovat i backtracking algoritmus na generování matice, ale pro účely projektu je toto řešení více než vyhovující.

Algoritmus nejprve vybere náhodně ze 4 poskytnutých hotových řešení. Poté až 25x prohodí řádky a sloupce podle těchto pravidel:

1. Prohazování sloupců nebo řádků je moužné pouze v triádách, např. se nesmí prohodit první a čtvrtý řádek.
2. Nesmí se prohodit řádek se sloupcem.

Generování semínka pro funkci srand() nelze v MCU provádět pomocí času, který v MCU chybí. Knihovna time.h není přítomna a proto jsem zvolil jiný přístup. Generování prvotní pseudonáhodné hodnoty (semínka) je pomocí teploměru obsaženého na FitKitu. Teplotní čidlo je velmi citlivé a proto při každém měření dává mírně odlišnou hodnotu. Toho jsem využil a teplotu nejprve desekrát změřím a součet těchto hodnot je použit jako semínko pro funkci srand(). To nám zaručí pseudonáhodná čísla s velmi nízkým opakováním.

Výsledná matice se nahraje do BRAM na FPGA, které ovládá herní logiku.

Logika hry

FPGA načte z BRAM vytvořenou matici čísel a náhodně vybere políčka, která zobrazí a která vynechá. Poté vykreslí na obrazovku danou sudoku hru a čeká na vstup od uživatele v podobě pohybu po desce, příp. zadání čísla.

Výběr políček, které se zobrazí je realizován pomocí 81-bitového vektoru, maskou, který překrývá matici čísel. Pokud je bit na dané pozici 0, číslo se nezobrazí.

Při zadání čísla FPGA zkontroluje z vygenerované matice, zda je to korektní volba.

Nad maticí čísel je umístěna také druhá maska, která zobrazuje/skrývá uživatelem zadané hodnoty.

Matice pro hodnoty hráče je implementována do ROM paměti, abychom s ní mohli efektivně pracovat. Základní maska je statická a není nijak upravována.

Pokud obě masky po logickém OR vrací vektor s maximální hodnotou (samé jedničky), je jasné, že uživatel vyplnil všechna prázdná pole a hra je tím pádem dokončena. Obrazkovka zmodrá a čeká se na reset.

Vykreslování na displej

Displej o rozměrech 640x480px je rozdělen na bloky o velikosti 32x32px, abychom dokázali lépe pracovat s texturami, adresováním pozice v poli a obecné orientace.

Každých 32 pixelů (v prvních 9 sloupcích a řádcích) je vykreslena 1px linka. To reprezentuje hrací desku, po které se může hráč pohybovat a kde jsou umístěna čísla. Pro usnadnění orientace je každý třetí sloupec a řádek linie jiné barvy pro reprezentaci triád.

Znázornění pozice hráče, kam může vkládat číslo je realizován pomocí kurzoru, kterým může pohybovat po hrací desce, ale ne po celé ploše.

Pohybování kurzorem je zpřístupněno pomocí kláves A (doprava), B (doleva), C (dolů) a D (nahoru).

Vykreslování jednotlivých čísel je vytvořeno pomocí bitových textur, které jsou uloženy v ROM paměti. Každý pixel je tvořen dvěma bity, které dekódujeme na 9-bitový vektor vyžadovaný VGA portem.

Komplikace při návrhu

Během implementace řešení jsem několikrát narazil na problém s obsazeností FPGA. Po důkladné a náročné revizi celého kódu se mi podařilo řešení optimalizovat na cca 50% využití slice. Největší místo zabíraly matematické operace násobení, dělení a modulo spolu s několika poli integerů. Po eliminaci těchto operací a polí se FPGA uvolnilo a doba syntézi byla cca desetinová (Core 2 Duo asi 120 s ve FitKit VM).

Současné využití zdrojů je znázorněno v příloze **zdroje.pdf**.

Vyhodnocení

Implementace komunikace BRAM s FPGA bylo velmi komplikované a nezvládl jsem jej implementovat. Logika hry tedy pracuje se statickou maticí čísel a jde tedy hrát pouze jedno sudoku. V MCU je také generováno náhodné číslo pro masku matice a tím pádem je také statická.

Vše ostatní je implementováno. Pro demonstraci komunikace BRAM přes SPI s MCU je v terminálu fitkitu demonstrováno její využití (generování desky, její uložení na BRAM a následné načtení).

Částečně implementované části jsou v archivu také přiložené.

Závěr

Celkové řešení návrhu je zpracováno pomocí jednotlivých komponent, které řeší dílčí celky daného problému. Při návrhu bylo doporučeno generovat matici v MCU z hlediska náročnosti operace na velikost FPGA čipu.

Výsledek projektu je možné shlédnout na YouTube: https://youtu.be/w_kRwBpi_ZI

RTL schéma návrhu je přiloženo v souboru **rtl.pdf**.

Blokový diagram