VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Seminář VHDL

2018/2019

Patnáctka

Michal Cibák (xcibak00) Brno, 19. května 2019

Obsah

Zadanie

Základné požiadavky

Implementujte hru Fifteen na FITkitu pouze s využitím FPGA. Cílem hry je přesouvat kameny s čísly tak, aby vznikla posloupnost 1 – 15 a „díra“. Stav hry bude zobrazován na monitoru, který bude připojen přes rozhraní VGA. Ovládání bude realizováno pomocí klávesnici FITkitu. Při zmáčknutí klávesy dolů (klávesa č. 8) se kámen, který má pod sebou „díru“ posune směrem dolů. Obdobně se bude logika chovat i pro ostatní směry. Kameny se nemohou přesouvat přes okraj mřížky. Při stisku kláves A – D se provede reset a nahraje uložená hra (výchozí stav) z paměti.

Rozširujúce požiadavky

Využijte v SVN dostupný řadič work.keyboard_controller.

Čtverce mějte o velikosti 64 x 64.

Buňky zobrazujte na středu monitoru.

Balíčky

game_pack

Obsahuje funkciu na výpočet masky bunky.

Pre sprehľadnenie kódu som do neho pridal tiež jednoduchú funkciu na výpočet pozície vo vektore stavu hry pre jednotlivé bunky.

vga_controller_cfg

Balíček zo SVN repozitára FITkitu používaný pri práci s VGA radičom.

Entity / komponenty

symbol_rom

Pamäť ROM, v ktorej sú uložené jednotlivé znaky kameňov v hre.

cell

Jedna bunka/kameň v hre, každý v sebe uchováva svoj stav, hodnotu 0 – 15. Hra sa potom skladá z matice 4x4 týchto kameňov, ktoré spolu komunikujú – vymieňajú si svoj stav.

bcd_counter

Počítadlo 0-9, ktoré pri prechode z 9 na 0 na jeden takt nastaví hodnotu pretečenia na 1.

keyboard_controller

Radič klávesnice zo SVN repozitára FITkitu nutný k práci s klávesnicou. Využívam iba jeden jeho výstup – vektor stlačených kláves, ktorý je aktualizovaný s nábežnou hranou hodinového signálu.

vga_controller

Radič VGA zo SVN repozitára FITkitu nutný k práci s VGA rozhraním.

top

Top level entity. Sú v nej vzájomne prepojené inštancie jednotlivých komponent a riešená zvyšná logika hry.

Základné prepojenie

V tomto odstavci je opísané prepojenie tak, ako je implementované v základnej verzii projektu. Odlišnosti, ktoré vznikli pridaním rozšírení, sú uvedené pri popise jednotlivých rozšírení.

Výstup radiča klávesnice je pripojený na stavový automat, ktorý čaká na stlačenie nejakej klávesy, túto zmenu ihneď spropaguje na svoj výstup a s nábežnou hranou hodinového signálu prechádza do druhého stavu, v ktorom už bez výstupu čaká, kým na klávesnici nebudú stlačené žiadne klávesy, následne prechádza späť do prvého stavu. Informácia o stlačenej klávese je teda na výstup propagovaná práve jeden hodinový takt. Ďalej je používaný už len tento výstup.

Bunky sú navzájom prepojené do matice 4x4, kde pre každý vstup stavu suseda je pripojený výstup daného suseda, na pozícii, kde sused nie je, sa uvažuje sused z opačnej strany daného riadku či stĺpca (vo výsledku po kompilácii však tieto porty ostanú nepripojené). Reset do tejto matice je privedený ako logický súčet resetu systému a resetu spôsobenému stlačením niektorej klávesy načítavajúcej stav hry (A, B, C, D). Podľa toho, ktorá z týchto kláves sa stlačí, sa na inicializačný vstup privedie príslušný počiatočný stav hry. Všetky stavy sú riešiteľné, jeden slúži ako referenčný – ako by malo vyzerať riešenie. Na klávesový vstup pre ovládanie sú privedené príslušné klávesy (2, 4, 6, 8).

Pamäť symbolov pracuje s relatívnymi súradnicami, ktoré sú voči tým poskytovaným VGA radičom posunuté (oneskorené) tak, aby bola hra vycentrovaná. Nulové súradnice teda pre ňu začínajú v ľavom hornom rohu hernej plochy. Z týchto súradnicových vektorov sú vybrané určité bity pre zostavenie select signálu pre multiplexor, ktorý vyberá príslušnú adresu bunky na adresový vstup pamäte, a iné byty pre výber príslušného stĺpca a riadka tejto bunky.

Na VGA radič sú dáta posielané okamžite po požiadavke, keďže pamäť ROM je asynchrónna a teda nevyžaduje oneskorenie, preto je možné pre generický parameter *REQ_DELAY* použiť predvolenú hodnotu 1 – dáta sú pri najbližšej nábežnej hrane hodín pripravené. Na základe *is_inside* signálu, ktorý určuje, či sa požadovaný pixel nachádza vo vnútri hracej plochy, je zvolená jednotná farba pre celé okolie, pre vnútro je na základe výstupu pamäte symbolov volená jedna farba pre pixely, ktoré sú označené ako ON a druhá pre pixely označené ako OFF.

Rozšírenia

Implementované

Detekcia výhry:

Pokiaľ sa užívateľ nachádza vo výhernom stave, nie je mu umožnené sa ďalej pohnúť a musí si zvoliť novú hru. POZOR – východiskový stav je výherný!

Riešenie:

Do matice buniek je aktuálny stav stlačených kláves propagovaný len vtedy, keď stav hry nie je výherný, v opačnom prípade je tam nastavená 0.

Rozpracované

Počítadlo ťahov:

Pod hernou plochou sa zobrazuje počet ťahov, ktoré užívateľ vykonal v danej hre.

Riešenie:

Využíva 3 počítadlá 0-9 (možno teda napočítať až 999 ťahov), pričom prvý pripočíta 1, pokiaľ je stlačená nejaká klávesa pohybujúca kameňmi (počíta teda aj potenciálny neplatný ťah), ďalšie pripočítavajú 1 vždy, keď dôjde k pretečeniu predošlého. Aby nebolo potrebné tvoriť ďalšiu pamäť symbolov, pri výpise počtu ťahov sa využíva už implementovaná pamäť ROM pre herné kamene. Keďže táto neobsahuje znak nuly, bolo nutné ju doplniť a teda aj zmeniť logiku kreslenia diery. Taktiež bolo nutné zmeniť zapojenie tejto pamäte, keďže treba vyberať, či sa má vybrať znak podľa kameňa alebo počítadla. Znaky tohto počítadla majú veľkosť 32x32, pričom po pretečení posledného počítadla (1000 ťahov) toto prestane vykresľovať.

Aktuálny stav:

Po zapojení počítadla funguje zvyšok hry správne, počítadlo však správne nepracuje. Vykresľuje sa, kým sa nestlačí nejaká klávesa, potom prestane – musí teda ihneď dôjsť k pretečeniu počítadla. Po nahraní novej hry sa však počítadlo správne znovu objaví. Logika zobrazovania znakov počítadla je asi chybná (pri manuálnom nastavení počítadla na 111 sa znaky nezobrazovali, pre stav 000 by sa však taktiež nezobrazilo nič, pretože nula v pamäti je diera). Nula v pamäti nebola nahradená za znak 0, spôsob jej vykresľovania v hernej ploche teda ostal rovnaký.

Plánované

Zmena farby znakov buniek na základe ich Manhattanskej vzdialenosti od správnej pozície.

Generovanie náhodných levelov alebo na základe seed-u – použitie displeja na FITkite.

Animácia ťahov.

Algoritmus prehľadávajúci stavový priestor, aby vypočítal minimálny počet ťahov potrebných pre víťazstvo – tento počet by bol použitý pre výpočet bodov, ktoré je možné získať. Každý ťah navyše by potom znamenal stratu nejakého počtu získateľných bodov.

Zdroje a výkonnosť

Množstvo zabraných zdrojov

Základná verzia projektu bez rozšírení:

- Adders/Subtractors
  - 2: 8-bit subtractor
- Comparators
  - 2: 12-bit comparator equal
  - 2: 4-bit comparator equal
  - 2: 6-bit comparator equal
  - 3: 7-bit comparator equal
- Counters
  - 2: 12-bit up counter
  - 1: 15-bit up counter
- FSMs
  - 4
- Multiplexers
  - 1: 1-bit 64-to-1 multiplexer
  - 1: 4-bit 16-to-1 multiplexer
- ROMs
  - 1: 16x64-bit ROM
- Registers
  - 90: Flip-Flops

Čo viedlo na:

- Slices: 313 out of 768 - 40%
- Slice Flip Flops: 150 out of 1536 - 9%
- 4 input LUTs: 606 out of 1536 - 39%
- IOs: 124
- Bonded IOBs: 118 out of 124 - 95%
- GCLKs: 1 out of 8 - 12%
- DCMs: 1 out of 2 - 50%

Maximálna pracovná frekvencia

Synthesis estimate: 14.351 MHz

Trace report after PAR: Neznáme.

Záver

Pri vypracovaní projektu som sa zlepšil v práci s jazykom VHDL a naučil sa vnímať a pracovať s paralelizmom v obvodoch. Pri opravách chýb, ktoré som pri písaní kódu často robil, som si začal uvedomovať mnohé kľúčové vlastnosti návrhu a hlavne fungovania hardvéru. Taktiež mi veľmi pomohlo, keď som sa snažil rozlúštiť spôsob fungovania už napísaných kódov (najmä radiča klávesnice).

Pri spätnom pohľade na prácu na projekte môžem potvrdiť, že projekt nie je náročný, ale je potrebné rozumieť tomu, o čo sa pokúšame. Časová náročnosť dosť závisí od predošlých znalostí, od šikovnosti študenta, teda ako rýchlo dokáže niečo pochopiť a tieto znalosti aplikovať, ako aj od toho, ako hlboko chce do problematiky zájsť.