Projekt

Zadání projektu IVH 2018

Vojtěch Mrázek, 2018
imrazek@fit.vutbr.cz

Projekt
zadání projektu IVH

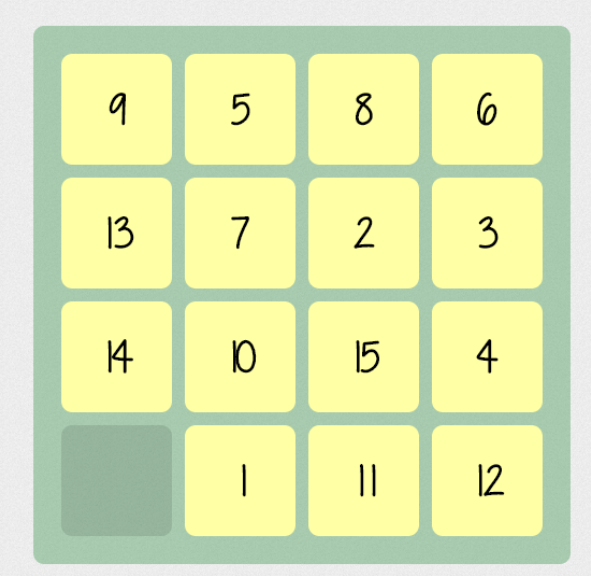
- **Tři varianty zadání**
  - Implementace hry Fifteen na FPGA
    Řešení této hry na FITkitu s výstupem přes VGA rozhraní a ovládáním přes klávesnici. Kompletně implementováno v FPGA pomocí VHDL.
  - Solver hry Lights Out
    Najít řešení této hry pomocí brute-force přístupu je relativně časově náročné (na Intel Xeon řádově sekundy, na MSP430 řádově hodiny). Proto je cílem projektu implementovat akcelerátor tohoto výpočtu v FPGA s tím, že součástí bude jednoduchá komunikace s MCU (kód v C pro mikrokotroler).
  - Vlastní zadání
    Je nutné schválení předem a konci semestru řešení osobně prezentovat
- **Přihlašování**

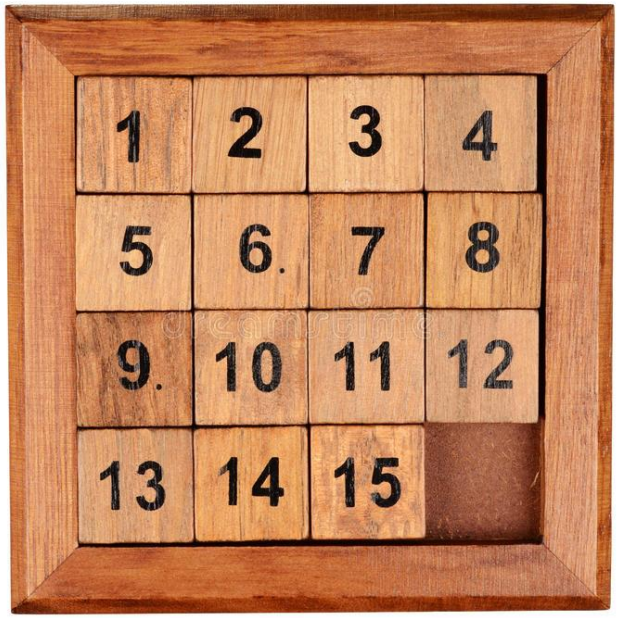
Středa 28. února 2018, od 20 hodin

Hra Fifteen

pravidla hry

- Logická hra, kdy se posouvají čtverce tak, aby vznikla posloupnost 1 – 15. Místo jednoho čtverce je prázdné políčko.
- Ne všechny počáteční stavy mají řešení

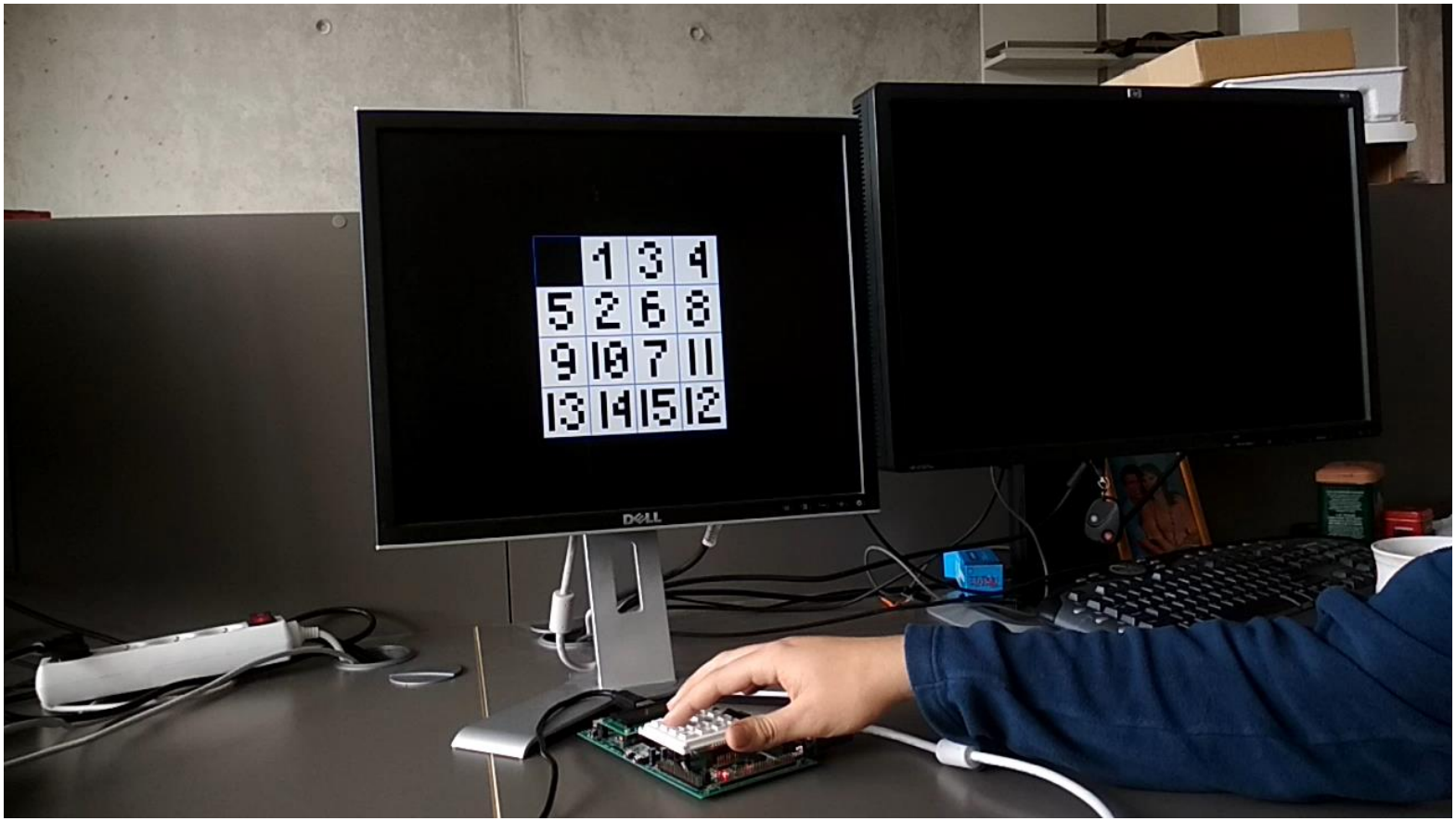




- http://lorecioni.github.io/fifteen-puzzle-game/

Hra Fifteen

ukázka implementace na FITkitu



Příklad implementace naleznete zde:
https://vimeo.com/256930698

Varianta 1: Implementace hry na VGA displeji

zadání

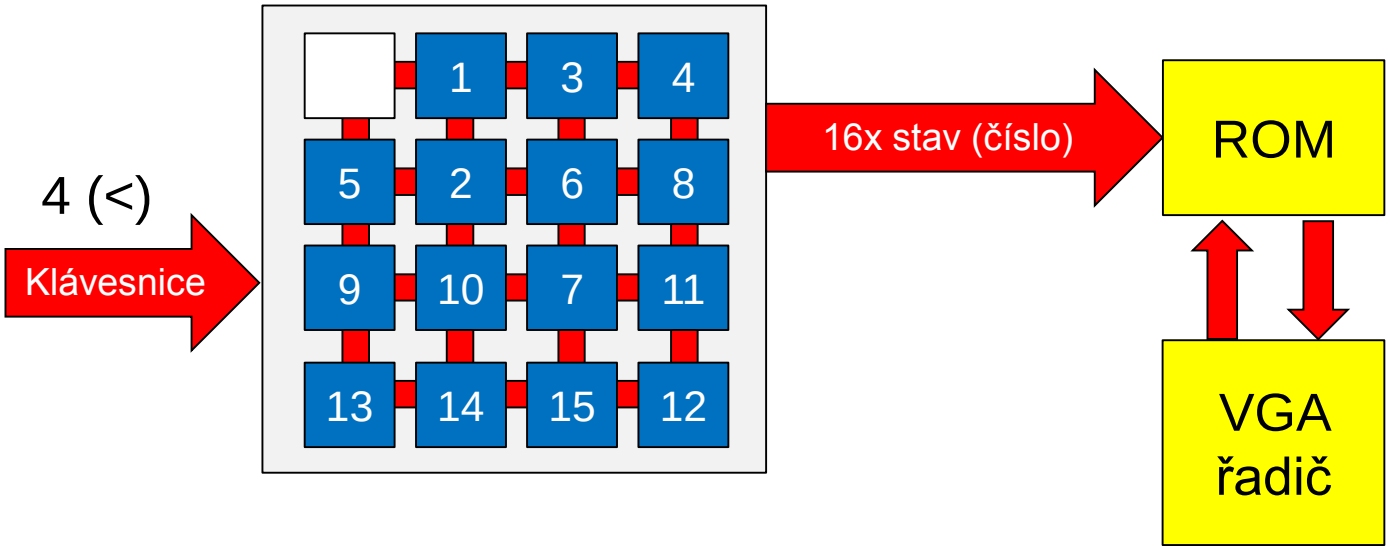
- Implementujte hru Patnáctka na FITkitu pouze s využitím FPGA. Cílem hry je přesouvat kameny s čísly tak, aby vznikla posloupnost 1 – 15 a „díra“.
- Stav hry bude zobrazován na monitoru, který bude připojen přes rozhraní VGA.
- Ovládání je realizováno pomocí klávesnici FITkitu. Při zmáčknutí klávesy dolů (klávesa č. 8) se kamen, 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 - viz video) z paměti.

Varianta 1: Implementace hry na VGA displeji

specifikace

- K řešení potřebujete monitor s VGA rozhraním (příp. DVI a redukci)
- Aby fungovalo VGA rozhraní na FITkitu, musí být spojena propojka J6
- Můžete se inspirovat libovolným projektem z repozitáře FITkitu (doporučuji se podívat na implementaci Pexesa)

- Hra neobsahuje žádný centrální řídicí prvek, jedná se mřížku uzlů, které navzájem komunikují

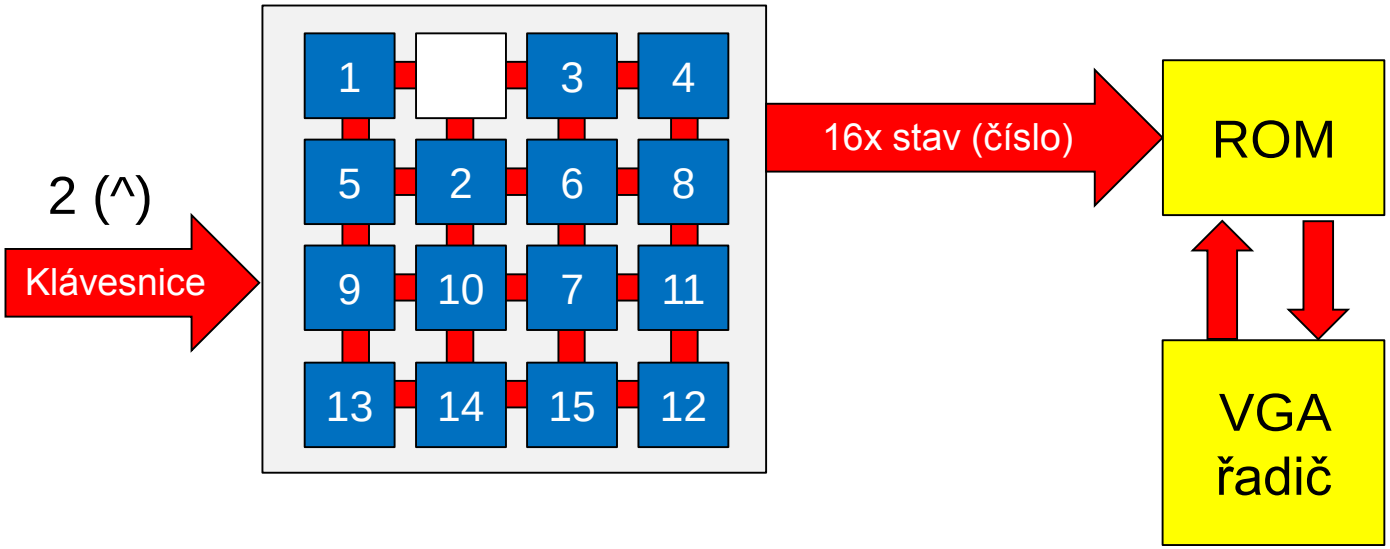


Varianta 1: Implementace hry na VGA displeji

specifikace

- K řešení potřebujete monitor s VGA rozhraním (příp. DVI a redukci)
- Aby fungovalo VGA rozhraní na FITkitu, musí být spojena propojka J6
- Můžete se inspirovat libovolným projektem z repozitáře FITkitu (doporučuji se podívat na implementaci Pexesa)

- Hra neobsahuje žádný centrální řídicí prvek, jedná se mřížku uzlů, které navzájem komunikují

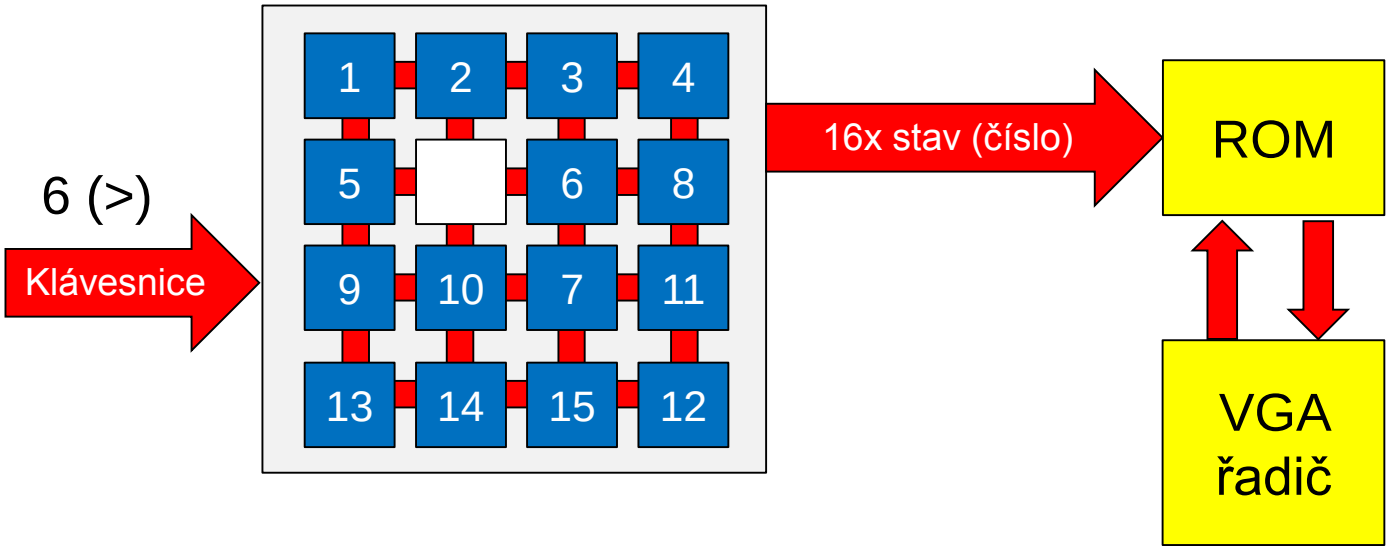


Varianta 1: Implementace hry na VGA displeji

specifikace

- K řešení potřebujete monitor s VGA rozhraním (příp. DVI a redukci)
- Aby fungovalo VGA rozhraní na FITkitu, musí být spojena propojka J6
- Můžete se inspirovat libovolným projektem z repozitáře FITkitu (doporučuji se podívat na implementaci Pexesa)

- Hra neobsahuje žádný centrální řídicí prvek, jedná se mřížku uzlů, které navzájem komunikují

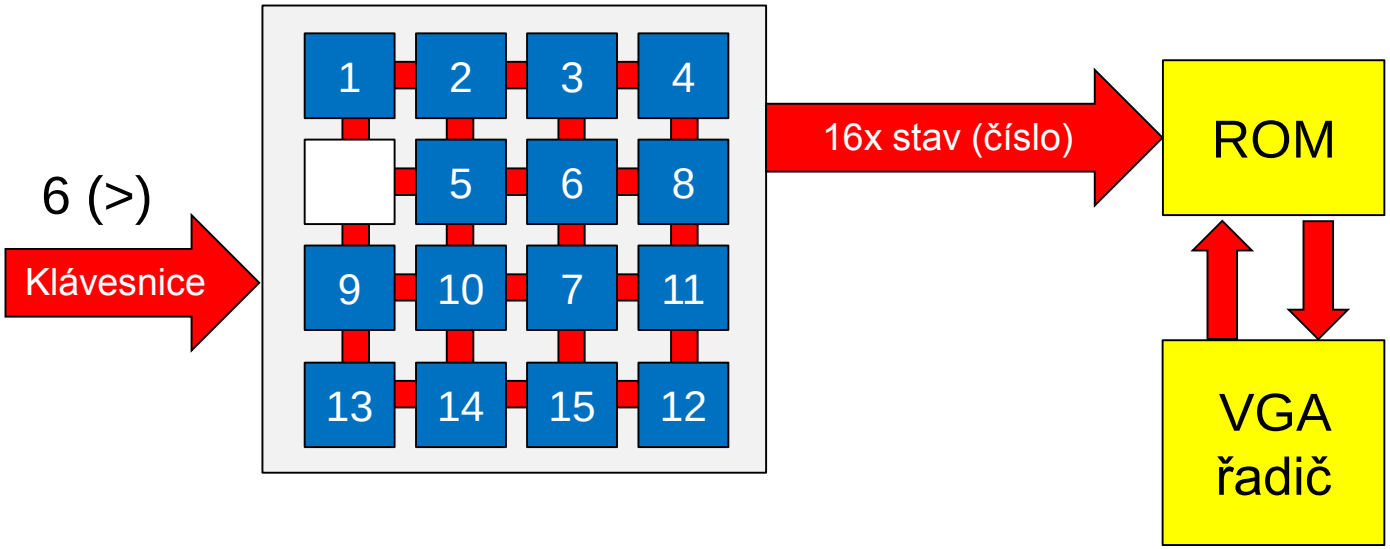


Varianta 1: Implementace hry na VGA displeji

specifikace

- K řešení potřebujete monitor s VGA připojením
- Aby fungovalo VGA rozhraní na FITkitu, musí být spojena propojka J6
- Můžete se inspirovat libovolným projektem z repozitáře FITkitu (doporučuji se podívat na implementaci Pexesa)

- Hra neobsahuje žádný centrální řídicí prvek, jedná se mřížku uzlů, které navzájem komunikují

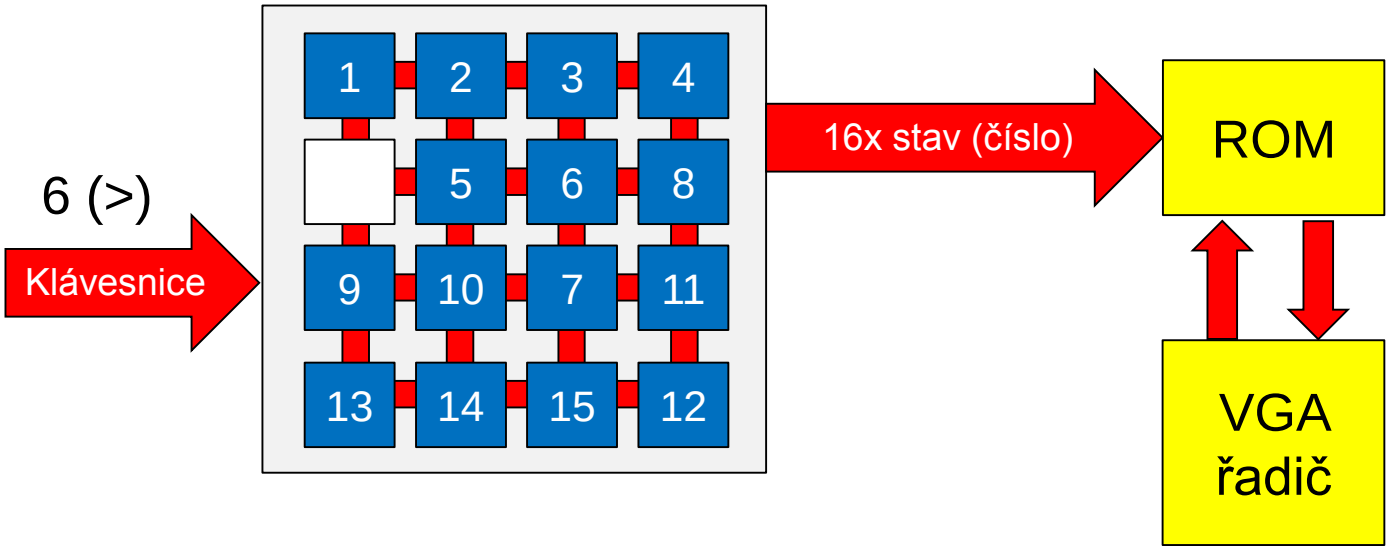


Varianta 1: Implementace hry na VGA displeji

specifikace

- K řešení potřebujete monitor s VGA rozhraním (příp. DVI a redukci)
- Aby fungovalo VGA rozhraní na FITkitu, musí být spojena propojka J6
- Můžete se inspirovat libovolným projektem z repozitáře FITkitu (doporučuji se podívat na implementaci Pexesa)

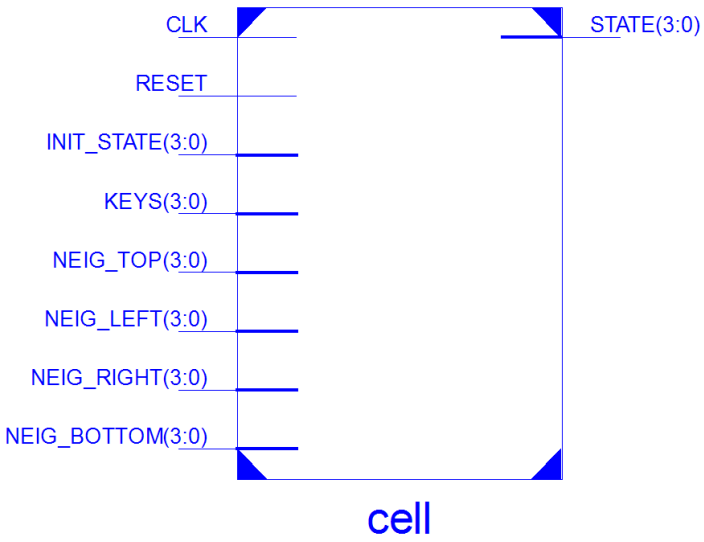
- Hra neobsahuje žádný centrální řídicí prvek, jedná se mřížku uzlů, které navzájem komunikují



Varianta 1: Implementace hry na VGA displeji

entita buňky

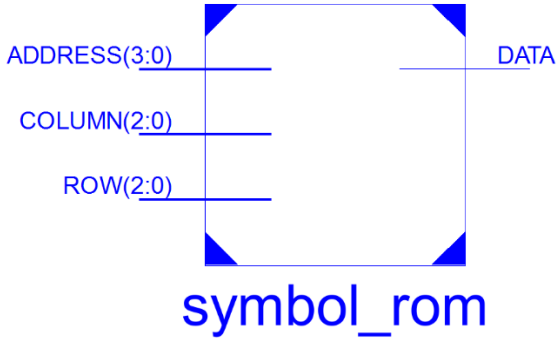
- Vstupní signály a parametry
  - CLK, RESET
  - Klávesnice (indikace stisku 4,8,6,2)
  - Počáteční stav (číslo 0-15)
  - Stavy sousedů (čísla 0-15)
  - MASKA – které sousedy brát v potaz, zjednodušuje generický popis ve VDHL
- Výstupní signály
  - Stav (číslo)
- Propojení
  - `for-generate` konstrukce
  - krajní buňky mohou být propojené s opačnou stranou (operace modulo)
  - výpočet masky, kteří sousedi opravdu sousedí, pomocí funkce `maska(x, y)`



Varianta 1: Implementace hry na VGA displeji

mapa znaků (entita SYMBOL_ROM)

- Vstupní signály
  - Adresa znaku (číslo 0 – 15, 0 = mezera)
  - Adresa pixelu - sloupec (0 – 7)
  - Adresa pixelu - řádek (0 – 7)
- Výstupní signály
  - Jednobitový signál Data



- Propojení
  - Budete muset vygenerovat ROM paměť s 16ti položkami (každá položka má 64b).
  - Pamět vrací konkrétní bit znaku ADDRESS podle COLUMN a ROW.
  - Abyste nemuseli vytvářet složitě tyto 64 bitové řetězce, připravili jsme pro vás skript

    http://www.fit.vutbr.cz/study/courses/IVH/public/gen_mem.php
  - Pro získání správné hodnoty dat pak použijte následující pseudo-konstrukci

```
DATA = MEMORY(ADDRESS)(ROW & COLUMN);
```

Varianta 1: Implementace hry na VGA displeji

další entity

- **Ovladač klávesnice**
  - Využijte v SVN dostupný řadič work.keyboard_controller.
- **BCD čítač**
  - Je napojený na klávesu kliknutí a počítá je. Využívá se k počítání score.
- **VGA řadič**
  - Inspirujte se projektem pexeso
  - Na základě řádku a sloupce vrací generuje signál RGB (3x3b)
  - Čtverce mějte o velikosti 64 x 64
  - Je vhodné si udělat signály row_adjusted a col_adjusted, kde pomocí odčítačky odečtete hodnotu posunu, aby byly buňky zobrazeny na středu monitoru
  - Pro zobrazení score můžete použít přístup z pexesa.
  - Při generování řádků není nutné neustále testovat

```
if col > LIM and col < LIM + WIDTH then …
```

  - Vhodnější je využít znalosti činnosti řadiče a vytvořit pomocný signál is_inside

```
signal is_inside : std_logic := '0';
if col == LIM then
   is_inside <= '1';
elsif col == LIM + WIDTH then
   is_inside <= '0';
end if;
```

Varianta 1: Implementace hry na VGA displeji

termíny

Do 18.3.2018 odevzdejte do WISu implementaci balíčku game_pack (game_pack.vhd), který obsahuje deklaraci a implementaci funkce pro výpočet masky — 10 bodů

Do 1.4.2018 odevzdejte do WISu implementaci paměti symbolů (symbol_rom.vhd) a jeho testbench (symol_rom_tb.vhd), který ověří — 10 bodů

- načítání všech 64 bitů jednoho symbolu (použijte assert)
- načítání jednoho konkrétního bitu všech 16 symbolů (použijte assert)

Do 15.4.2018 odevzdejte do WISu implementaci buňky (cell.vhd) a její testbench (cell_tb.vhd), který zkontroluje, že: — 15 bodů

- přijme stav od souseda, pokud je nulová a zmáčkla se příslušná klávesa
- zruší (vynuluje) svůj stav, pokud se zmáčkla příslušní klávesa
- reaguje správně na masku
- správně funguje inicializace (RESET)

Do 4.5.2018 odevzdejte do WISu kompletní projekt pro FITkit. Kromě zdrojových souborů odevzdejte stručnou dokumentaci ve formátu PDF obsahující informaci o množství zabraných zdrojů a výkonnosti (max. pracovní frekvence) vašeho řešení.

Varianta 2: Hledání řešení hrubou silou

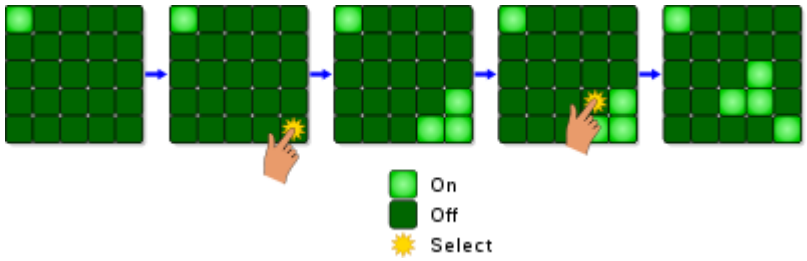
zadání

- Cílem hry Lights Out je zhasnout všechna rozsvícená políčka. Hra spočívá v tom, že máme danou mřížku buněk o velikosti 5x5. Každá buňka buď svítí, nebo je zhasnutá. Při kliknutí na jednu buňku invertuje svůj stav nejen tato buňka, ale také její čtyřokolí (t.j. buňka nalevo, napravo, nad a pod - ne však zešikma).

- Cílem tohoto projektu bude vytvořit akcelerátor pro hledání řešení pro zadanou vstupní kombinaci. Pomocí terminálu QDevKitu se zadá počáteční stav (25 bitů, kde 0. bit značí stav buňky [0,0], 1. bit stav buňky [0,1] atd.). Tato konfigurace se odešle do FPGA, které projde všechny možné kombinace kliknutí 0 až (225 - 1) a řešení, které vedlo k přechodu do stavu samých 0 (všechna světla zhaslá), se odešle do MCU a vypíše na terminál. Pokud takové neexistuje, na terminál se vypíše informace.

Lights Out
pravidla hry

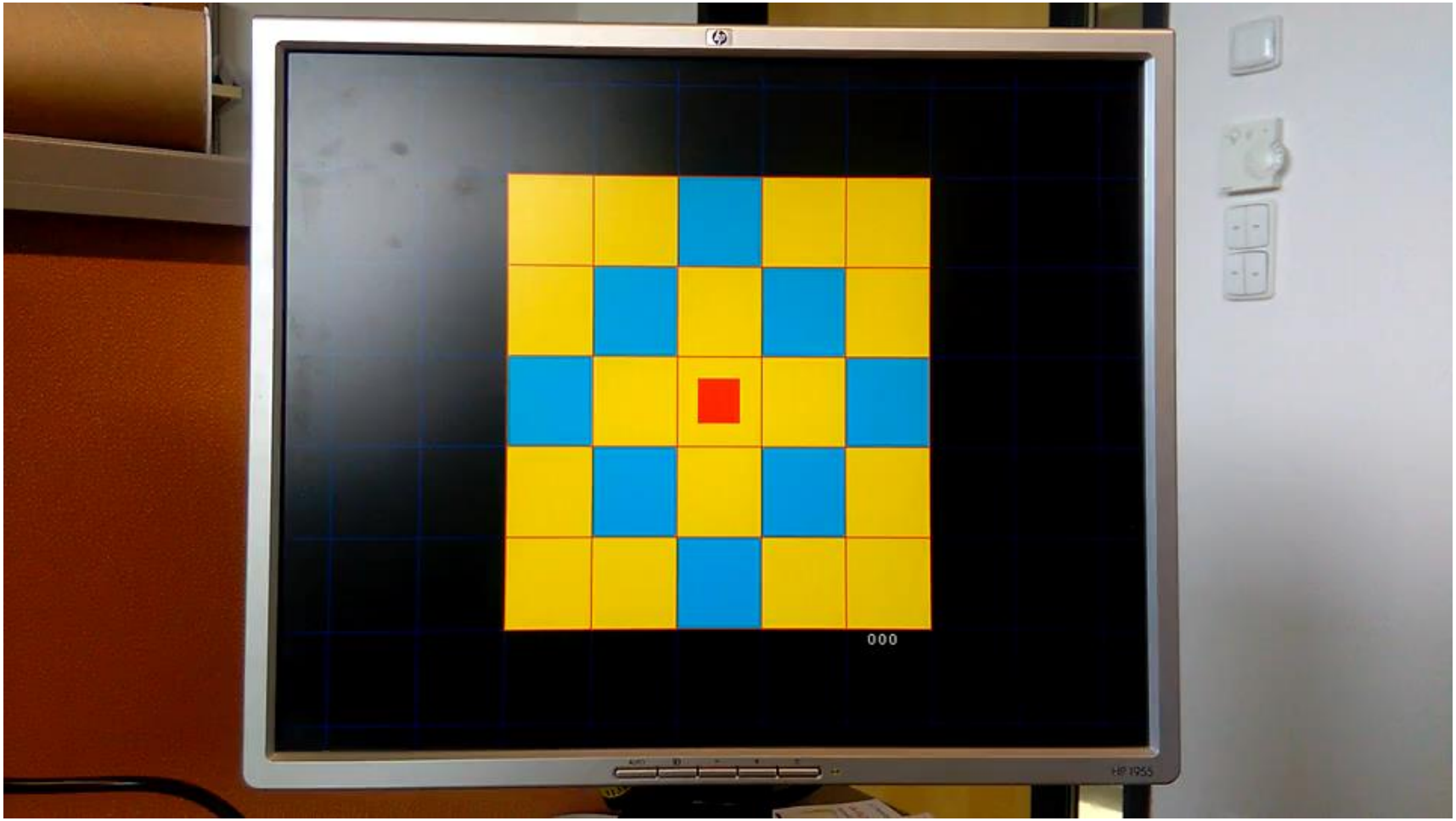
- Populární elektronická hra (verze 3x3 v 1970, verze 5x5 v roce 1995)
- Pravidla jsou taková, že při stisku jedné buňky invertuje svůj stav nejen tato buňka, ale také její 4-okolí



- Ne všechny poč. stavy mají řešení, řešení se dá získat matematicky
  - Solver: http://www.ueda.info.waseda.ac.jp/~n-kato/lightsout/
  - Příklad: http://www.logicgamesonline.com/lightsout/

Varianta 2

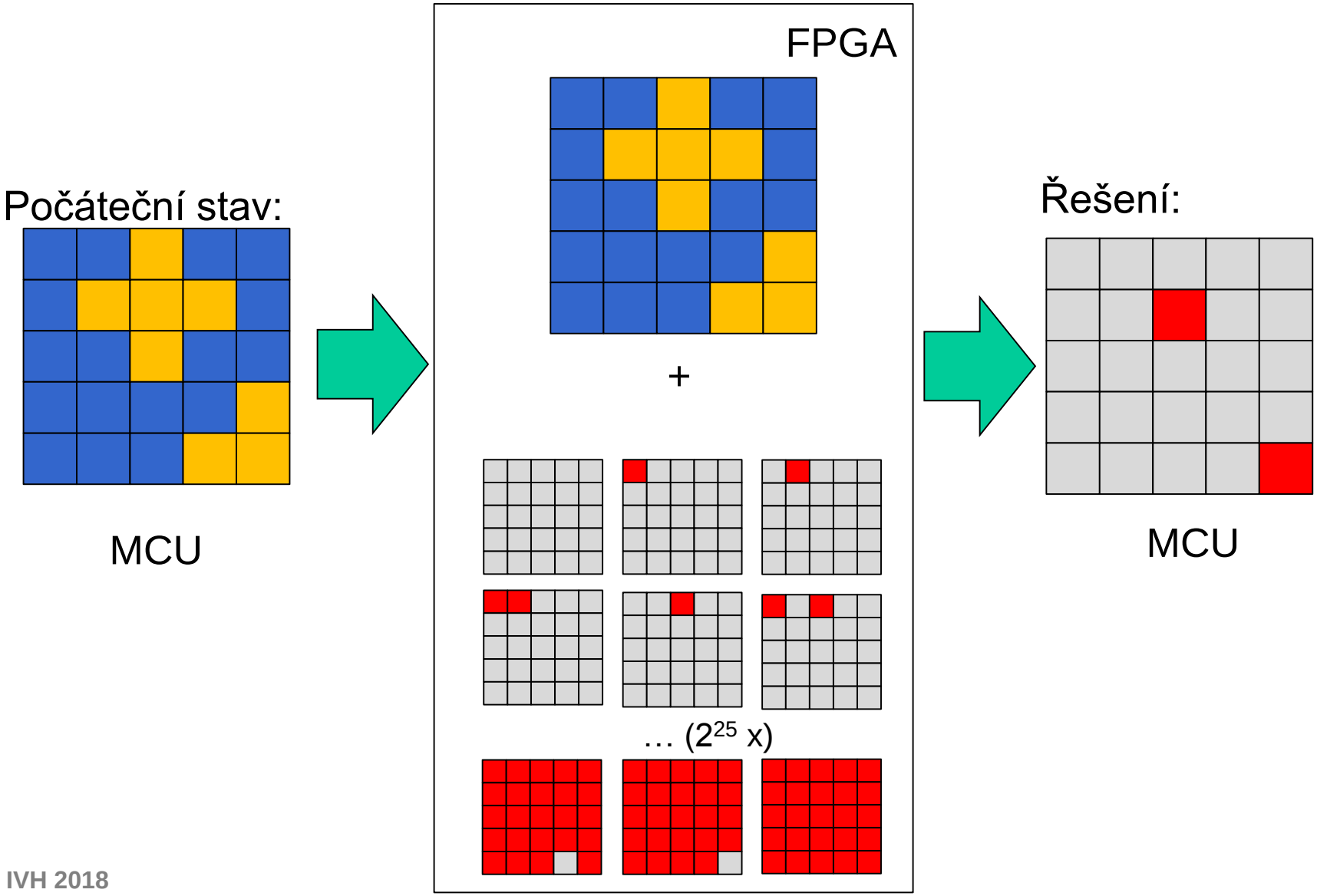
ukázka



Příklad implementace hry v FPGA je
https://vimeo.com/154698976

Varianta 2: Hledání řešení hrubou silou

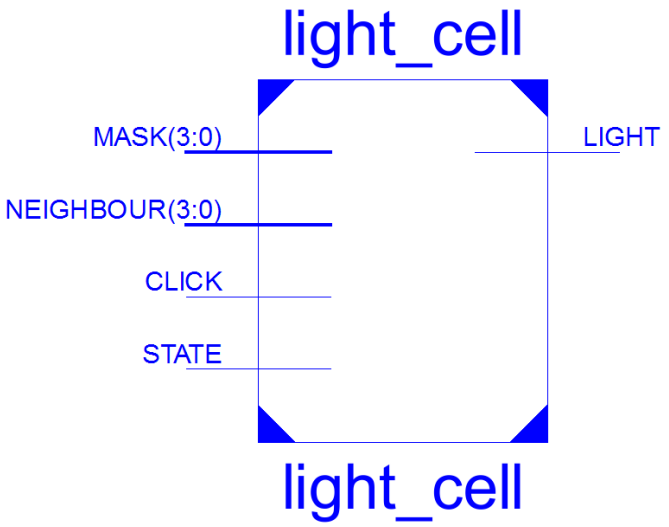
specifikace



Varianta 2: Hledání řešení hrubou silou

entita buňky

- Vstupní signály a parametry
  - CLK, RESET nejsou (kombinační)
  - Počáteční stav buňky
  - Indikace kliknutí na buňku
  - Indikace kliknutí na souseda
  - MASKA – které sousedy brát v potaz, zjednodušuje generický popis ve VDHL
- Propojení
  - `for-generate` konstrukce
  - krajní buňky mohou být propojené s opačnou stranou (operace modulo)
  - výpočet masky, kteří sousedi opravdu sousedí, pomocí funkce `maska(x, y)`
- Výstupní signály
  - Stav buňky



Varianta 2: Hledání řešení hrubou silou

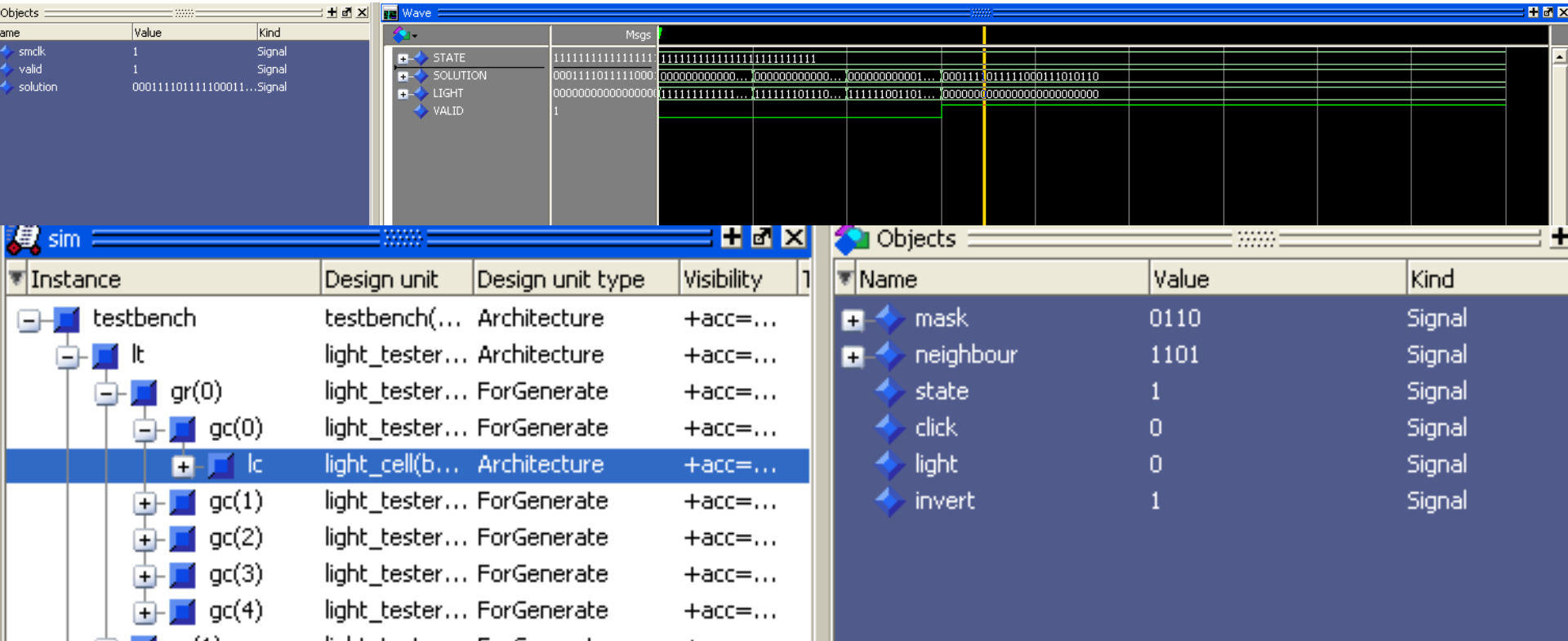
řízení

- Stavový automat
  - Komunikuje přes SPI s MCU (bude ukázáno na přednášce)
  - v FPGA se aktivuje čítač, který generuje všechna kandidátní řešení.
  - Pokud jsou všechna světla zhaslá, čítač se pozastaví a řešení se zašle do MCU.
  - Pokud se nepodařilo řešení najít, informuje se o tom MCU taktéž.
  - Komunikační protokol je na vás.

Varianta 2: Hledání řešení hrubou silou

ladění

- Testbench
  - počáteční stav 111111111111111111111111
  - řešení 000111101111100011101011O
- Po napsání testbenche uvidíte, který prvek se vám nezhasl
- Tam hledejte chybu



Varianta 2: Hledání řešení hrubou silou

termíny

Do 18.3.2018 odevzdejte do WISu implementaci balíčku game_pack (game_pack.vhd), který obsahuje deklaraci a implementaci funkce pro výpočet masky 10 bodů

Do 1.4.2018 odevzdejte do WISu implementaci 25-bitového čítače (cntr.vhd) a jeho testbench (cntr_tb.vhd), který ověří 10 bodů

- korektní funkci inicializace (RESET)
- korektní funkci čítače při povoleném i zakázaném čítání
- korektní funkci hlásící dosažení konečné hodnoty (nesimulujte všech 2^25 kombinací, ale implementujte podporu pro vložení počáteční a koncové hodnoty pomocí dvou generických parametrů)

Do 17.4.2018 odevzdejte do WISu implementaci buňky (cell.vhd) a její testbench (cell_tb.vhd), který zkontroluje, že: 15 bodů

- pro všech 9 variant buněk podle masky (horní roh, pravý kraj, střed …) buňka korektně vyhodnotí svůj stav po kliknutí
- pro všech 9 variant buněk podle masky (horní roh, pravý kraj, střed …) buňka korektně vyhodnotí svůj stav po kliknutí na některou sousední buňku

Do 4.5.2018 odevzdejte do WISu kompletní projekt pro FITkit. Kromě zdrojových souborů odevzdejte stručnou dokumentaci ve formátu PDF obsahující informaci o množství zabraných zdrojů a výkonnosti (max. pracovní frekvence) vašeho řešení.

Projekty
závěr

- Projekty jsou obdobně náročné
- Všechny projekty jsou individuální (pokud není domluveno předem s vyučujícím jinak). Jakákoliv spolupráce (i na průběžných úkolech) bude řešena jako disciplinární přestupek.
- Nezapomeňte na průběžné úkoly (až 35 bodů!)
- Pro získání více než 75 bodů bude nutné implementaci nějak rozšířit (počítadlo tahů, animace při skončení, testbench pro matici, paralelní zpracování pro solver, apod.). Rozšíření zmiňte v dokumentaci
- Potřebné znalosti týkající se implementace VGA, obsluhy klávesnice, komunikace s MCU získáte na přednáškách.
- V případě vlastního zadání je nutné zadání konzultovat a domluvit se na specifikací úkolu a průběžných úkolech
- Dotazy na projekt řešte v průběhu/po přednášce nebo s Vojtěchem Mrázkem - imrazek@fit.vutbr.cz, L307