Tema 2 - Prelucrarea imaginilor

- Responsabili: Bogdan Bădicu, Giorgiana Vlăsceanu, Silvia Cristina Stegaru
- Deadline soft (fără penalizări): **28.11.2015** ora **23:59**; Deadline hard: **5.12.2015** ora **23.59**
- Data publicării: 12.11.2015
- Data ultimei actualizării: 20.11.2015, 17:00
- Istoric modificări:
  - 12.11.2015:
    - Adăugare enunț
  - 14.11.2015:
    - Adăugare modul top/test în schelet
  - 17.11.2015:
    - ~~Adăugare tester Windows~~
    - Adăugare tester Windows, varianta corectă
  - 18.11.2015:
    - Prelungire deadline (soft și hard)
  - 20.11.2015:
    - Corectare bug-uri overflow in tester
    - Corectare bug verificare pixel (0, 0) in tester
  - 24.11.2015:
    - Adăugare precizare.

Obiectiv

Tema are ca scop exersarea lucrului cu noțiunile de Verilog folosite pentru proiectarea circuitelor secvențiale.

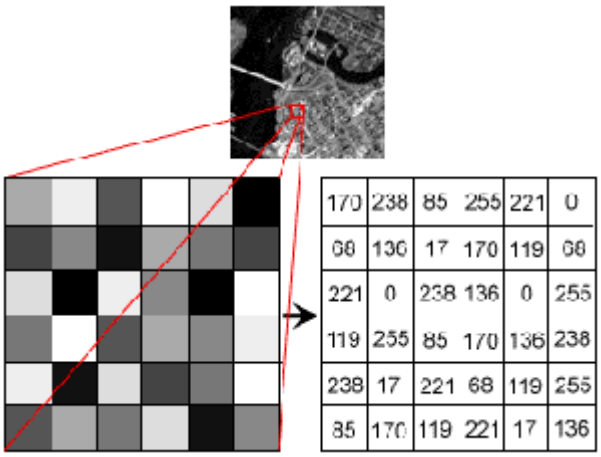
Descriere și cerințe

Implementați în Verilog un circuit secvențial sincron care prelucrează imagini RGB (cu trei canale de culoare, fiecare pe 8 biți). Imaginile au dimensiunea de 64×64 de elemente, în care fiecare element are 24 de biți (8 biți 'R', 8 biți 'G' și 8 biți 'B').

Click pentru informații adiționale despre modul de stocare al imaginilor

Indiferent de formatul în care sunt stocate, imaginile digitale sunt compuse din pixeli (vezi Fig. 1 și 2). Fiecare pixel din imaginea astfel formată este un număr care reprezintă culoarea pe acea unitate. În cazul imaginilor alb-negru un pixel este stocat pe 8 biți, deci poate lua valori între 0 (reprezentând culoarea negru) și 255 (reprezentând alb), toate valorile intermediare reprezentând tonuri de gri.

Fig. 1



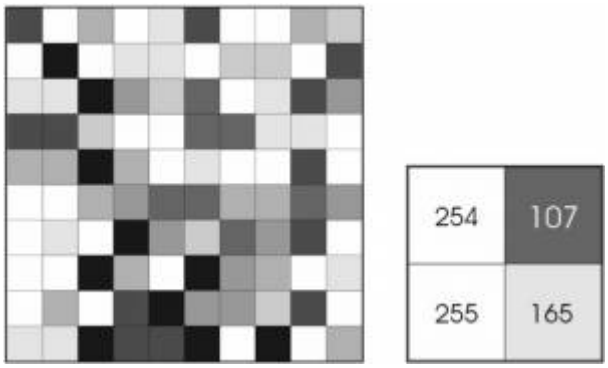


Fig. 2

Figurile 1 și 2 au provenit din urmatoarele surse:

- http://hosting.soonet.ca/eliris/remotesensing/LectureImages/pixel.gif
- https://www.spacetelescope.org/static/projects/img/imgproc_fig2.jpg

Pentru această temă, noi vom folosi imagini RGB în care un pixel este stocat pe 3 canale (Red - Green - Blue), fiecare canal reprezentând saturația culorii respective, stocată pe 8 biți. Imaginea finală este obținută prin compunerea valorilor din cele trei canale. Puteți citi mai multe despre spațiul de culoare RGB aici.

Cerințe

1. Efectuați transformarea imaginii prin oglindire. Oglindirea va fi realizată pe verticală, relativ la rândurile imaginii.
2. Realizați echivalentul imaginii în grayscale. Imaginea rezultantă va fi stocată pe 8 biți în canalul 'G'. Valoarea din canalul 'G' va fi calculată drept media dintre maximul si minimul valorilor din cele trei canale. După această operație, canalele 'R' și 'B' vor fi setate pe valoarea '0'. Filtrul grayscale va fi realizat pe imaginea oglindită.
3. Transformați imaginea grayscale obținută la punctul anterior prin aplicarea unui filtru de sharpness, folosind matricea de convoluție: $SharpMatrix = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$
4. Cele 3 cerințe de mai sus trebuie să se comporte ca un proces continuu pe aceeași imagine. Astfel, task-ul 2 va începe automat după task-ul 1, iar task-ul 3 va începe automat după task-ul 2. Pentru a putea fi luate în considerare de tester, semnalele *_done trebuie păstrate pe 1 cel puțin un ciclu de ceas.

Click aici pentru informații despre cum se folosește matricea de convoluție

Fiecare pixel din imaginea nouă reprezintă rezultatul aplicării matricei de convoluție pe pixelii corespunzători din imaginea care se dorește a fi modificată. Astfel, pentru a calcula valoarea pixelului de pe poziția [i, j], vom lua în considerare matricea 3×3 din jurul acestei poziții și vom înmulți element cu element matricea obținută cu cea de convoluție (în același mod în care ați efectua înmulțirea element cu element în Matlab). Pixelul de pe poziția [i, j] din imaginea nouă va fi dat de suma acestor 9 valori.

Pentru mai multe detalii, citiți informațiile de aici.

Implementare

Automatele cu stări finite care vor modela comportamentul general trebuie implementate în modulul process, având interfața:

```
module process(
    input clk,                  // clock
    input [23:0] in_pix,            // valoarea pixelului de pe pozitia [in_row, in_col] din imaginea de intrare (R 23:16; G 15:8; B 7:0)
    output [5:0] row, col,              // selecteaza un rand si o coloana din imagine
    output out_we,                  // activeaza scrierea pentru imaginea de iesire (write enable)
    output [23:0] out_pix,            // valoarea pixelului care va fi scrisa in imaginea de iesire pe pozitia [out_row, out_col] (R 23:16; G 15:8; B 7:0)
    output mirror_done,             // semnaleaza terminarea actiunii de oglindire (activ pe 1)
    output gray_done,             // semnaleaza terminarea actiunii de transformare in grayscale (activ pe 1)
    output filter_done             // semnaleaza terminarea actiunii de aplicare a filtrului de sharpness (activ pe 1)
    );
```

Funcționalitatea modulului `process` este următoarea:

- Semnalele `row` și `col` selectează un pixel din imagine din poziția (row, col), atât pentru input, cât și pentru output.
- Pentru a citi un pixel din imaginea care urmează a fi prelucrată (`in_pix`), trebuie setate semnalele `row` și `col`.
- Pentru a scrie un pixel în imaginea prelucrată (`out_pix`), trebuie setate semnalele `row` și `col`, precum și semnalul `out_we`.
- Semnalele `*_done` semnalează terminarea prelucrării imaginii pentru fiecare acțiune cerută.

Declararea ieșirilor de tip reg este permisă pentru modulul process. În rest, intrările și ieșirile pentru modulele process și image nu trebuie modificate.

Modulele voastre vor mai interacționa cu modulul `image` (pe care îl găsiți deja implementat în scheletul temei), reprezentând imaginea pe care trebuie să aplicați transformările. Modulul `image` are interfața următoare:

```
module image(
    input clk,         // clock
    input[5:0] row,         // selectează un rând din imagine
    input[5:0] col,         // selectează o coloană din imagine
    input we,         // write enable (activează scrierea în imagine la
rândul și coloana date)
    input[23:0] in,         // valoarea pixelului care va fi scris pe poziția
dată
    output[23:0] out    // valoarea pixelului care va fi citit de pe poziția
dată
    );
```

Observații

- Fiecare „pixel” din imagine este de dimensiunea 3 bytes (1 byte pentru fiecare canal).
- Imaginile sunt mărginite. Pentru aplicărea filtrului pe pixelii aflați în marginea imaginii se vor lua în considerare doar pixelii din imediata vecinătate. Tot ce iese în afara imaginii va fi egal cu 0.
- Semnalele mirror_done, gray_done și filter_done trebuie să mențină valoarea HIGH timp de un ciclu de ceas pentru a putea fi luate în considerare de tester. În acest ciclu de ceas nu veți face alte procesări și nu veți începe rezolvarea următorului task.
- Nu este permisă cache-uirea matricei întregi (citerea matricei și salvarea acesteia în cadrul modulului pentru procesare ulterioară). Puteți, în schimb, sa cache-uiți până la 6 rânduri dacă aveți nevoie. Această cache-uire va fi explicată apoi în cadrul readme-ului (motivul folosirii).

Precizări

- Arhiva temei (de tip zip) trebuie să cuprindă în rădăcina sa (fără alte directoare) doar:
  - fișierele sursă (extensia .v)
  - fișierul README
- Arhiva nu trebuie să conțină fișiere de test, fișiere specifice proiectelelor etc.
- Conținutul fișierului README:
  - numele, grupa și denumirea temei
  - prezentarea generală a soluției alese
  - explicarea porțiunilor complexe ale implementării (poate fi făcută și în comentarii)
  - alte detalii relevante
- Tema trebuie realizată individual; folosirea de porțiuni de cod de la alți colegi sau de pe Internet (cu excepția site-ului de curs) poate fi considerată copiere și va fi penalizată conform regulamentului.

Notare

- 10 pct: corectitudine.
  - 3 pct: oglindirea
  - 3 pct: grayscale

    - 3 pct: sharpness
    - 1 pct: dacă cele 3 acțiuni (oglindire, grayscale, sharpness) sunt executate în ordine (una dupa alta)
- **-1.5 pct pentru fiecare subpunct în care e folosită matricea cache-uită (total -4.5 puncte dacă e folosită în toate subpunctele)**
- -10 pct: folosirea construcțiilor nesintetizabile din Verilog (while, repeat, for cu număr variabil de iterații etc.)
- -1 pct: lipsa fișierului README.
- -0.5 pct: pentru fiecare zi de întârziere; tema poate fi trimisă cu maxim 7 zile întârziere față de termenul specificat în enunț (față de deadline-ul soft).
- -0.2 pct: folosirea incorectă a atribuirilor continue (`assign`), blocante (=) și non-blocante (<=).
- -0.2 pct: indentare haotică
- -0.2 pct: lipsa comentariilor **utile**
- -0.1 pct: comentarii inutile (ex. `wire x; // semnalul x`)
- -0.2 pct: diverse alte probleme constatate în implementare (per problemă)

Dacă tema primește 0 pe *vmchecker*, se pot acorda maxim 2 puncte pe ideea implementării, la latitudinea asistentului. Ideea și motivele pentru care nu funcționează trebuie documentate temeinic în README și/sau comentarii. Temele care au erori de compilare vor fi notate cu 0 puncte.

Resurse

- Schelet
- Tester Windows
- PDF temă