Stoica Vlad Ionut - 333AA

Pentru a asigura ca semnalele *_done vor avea valoarea HIGH pentru un ciclu de ceas in care nu se prelucreaza nimic si nu se incepe urmatorul task, am introdus semnalele m_done, g_done, f_done. Cand task-ul e gata, acestea vor primi valoarea HIGH, iar in urmatorul ciclu de ceas semnalul *_done respectiv lor va primi si acela valoarea HIGH.

Am adaugat si un semnal pentru a reprezenta starea procesului (si semnalul cu starea urmatoare, pentru sincronizare):

0: initial (din moment ce blocul initial este nesintetizabil, aceasta stare va functiona drept "bloc initial");

1: mirror;

2: gray;

3: filter.

Dupa ce un proces se termina se duce la urmatorul, cu exceptia procesului filter, care duce inapoi la 0.

S-a implementat un bloc always secvential pentru a sincroniza tot procesul la bataile ceasului si un bloc always combinational unde magia (procesul) ia loc.

Pentru procesul de oglindire a imaginii s-au folosit un semnal pentru reprezentarea starii acestui proces (si semnalul cu starea urmatoare, pentru sincronizare), 2 registre de memorie care sunt folosite drept auxiliare la schimbarea valorii a 2 pixeli (am folosit 2 deoarece cand am folosit doar 1 procesul s-a desincronizat)

La starea

0: se retine valoare lui in_pix intr-un registru de memorie "auxiliar", randul (registrul next_row) devine valoarea vertical simetrica a ei insisi, apoi se trece la starea 1;

1: registrul out_pix primeste valoarea primului registru auxiliar si semnalul out_we devine 1 pentru ca aceasta valoare sa fie scrisa in imagine, al doilea registru auxiliar primeste valoarea pixelului de pe pozitia asta, valoarea randului revine la cea anterioara, iar starea devine 2;

2: registrul out_pix primeste valoarea celui de-al doilea registru auxiliar si semnalul out_we devine 1, valoarea randului este incrementata cu 1, iar starea revine la 0. Daca randul se afla la valoarea 31 (ultimul rand din prima jumatate a matricii), acesta primeste valoarea 0, iar valoarea coloanei este incrementata cu 1 in schimb, daca si aceasta se afla la 63, inseamna ca s-au parcurs toti pixeli, imaginea a fost oglindita in totalitate (sper), astfel coloana devine 0 si semnalul m_done devine 1 pentru a se trece mai departe.

Acest proces ar trebui sa parcurga toti pixeli din prima jumatate de randuri si sa le interschimbe valoarea cu pixelul vertical simetric lor din a doua jumatatea de randuri.

Pentru procesul de ingriire a imagini s-au folosit trei semnale wire de 8 biti, R, G, B, carora le-au fost alocate bitii respectivi din in_pix, 2 registre (gray_max, gray_min) pentru a reprezenta valoarea maxima si minima dintre acelea trei; si un semnal (gray_val) caruia i s-a alocat media aritmetica dintre acele 2 registre.

Acest proces are decat o stare, starea in care se afla valoarea maxima si minima dintre R, G si B intr-un bloc de if-uri, apoi i se ofera lui out_pix valoarea lui gray_val, flancat pe ambele parti de 8 biti de 0, astfel incat doar portiunea "G" sa mai aiba valori, semnalul out_we devine 1, apoi se trece la urmatorul semnal din rand, pana cand ajungem la ultimul din rand, punct in care trecem la urmatorul rand, pana cand se ajunge la ultimul in rand, punct in care semnalul g_done devine 1 si se trece la urmatorul proces.

Pentru procesul de filtrare sau ascutire a imagini s-au folosit o matrice 3x3 de 8 biti (neghbour_matrix) unde sa se cache-uiasca un pixel si "vecinii" sai (doar valoriile din G, aceasta portiune fiind singura care are valori in urma procesului de ingriire), un vector de 8 biti de 64 de registre de memorie (prev_row) care sa cache-usiasca valoriile originale ale unui rand din imagine pentru a fi folosit in construirea matricei de vecini (acesta intotdeauna va reprezenta randul de "deasupra" pixelului curent, astfel toare valorile lui incep cu 0), un registru filter_sum pentru a retine valoarea filtrului aplicat, 2 intergeri pentru a parcurge matricea de vecini, un semnal (shift_done) pentru a sincroniza o portiune a procesului si un semnal pentru a reprezenta starea procesului (si semnalul cu starea urmatoare, pentru sincronizare).

La starea:

0: aceasta este starea "new line" care, dupa cum numele sugereaza, va avea in totdeauna primul pixel dintr-un rand, astfel prima coloana din neghbour_matrix va fi 0 (acei "vecini" ai pixelului fiind "out of bounds"), restul primei linii va fi completata de primele 2 valori din prev_row, valoarea din mijloc va fi G, apoi se incrementeaza randul cu 1 pentru a afla valoarea de "dedesubtul" pixelului curent si se trece la starea 1;insa, daca acesta este ultimul rand, acea valoare va fi zero, precum si valoarea pixelului din coltul dreapta jos, astfel se incrementeaza coloana in schimb si se trece la starea 3.

1: valorii neghbour_matrix[2][1] i se atribuie G, coloana se incrementeaza cu 1 si se trece la starea 2.

2: valorii neghbour_matrix[2][2] i se atribuie G, randul se incrementeaza cu -1 si se trece la starea 3.

3: valorii neghbour_matrix[1][2] i se atribuie G, coloana se incrementeaza cu -1 si se trece la starea 4.

4: se cache-uieste valoarea G in prev_row, apoi se calculeaza valoarea filter_sum, i se atribuie la inceput 9*neghbour_matrix[1][1], apoi i se scad toate celelalte valori din matrice (pentru a simula matricea de convolutie), asigurandu-ma ca nu trece de 255, lui out_pix i se atribuie valoarea finala a lui filter_sum, semnalul out_we devine 1 si apoi se trece la starea 5.

5: pentru a trece la urmatorul pixel, coloanele lui neghbour_matrix se "shifteaza" in stanga, prima coloana devine a doua, si a doua a treia, pentru acest proces a fost folosit shift_done pentru a asigura ca ramane sincron, apoi se trce la a gasi restul valorilor din neghbour_matrix: daca este un caz general in care pixelul curent si cel urmator se afla in mijlocul imaginii, neghbour_matrix[0][2] primiste valoarea respectiva vecinului urmatorului pixel, prev_row[col+2], coloana se incrementeaza cu 2, randul cu 1 si se trece la starea 2, dar, daca este:

- ultimul rand, neghbour_matrix[2][2] devine 0 si doar coloana se incrementeaza cu 2 si se trece la starea 3;

- penultimul pixel de pe rand (urmatorul pixel e ultimul), ultima coloana din neghbour_matrix devine 0 si se trece inapoi la starea 4;

- ultimul pixel din rand, coloana devine 0, randul se incrementeaza cu 1 si satrea devine 0, daca si randul este egal cu 63, atunci s-au parcurs toti pixelii, randul devine 0 o ultima data, iar f_done devine 1 marcand sfarsitul procesului.

Acest proces construieste matricea de vecini al unui pixel, folosind valori cache-uite anterior, in prev_row sau chiar in neghbour_matrix, si o serie de alte stari facute special sa preia valori "din fata" pixelului curent ca apoi sa-i calculeze varianta filtrata a acestuia, parcurgand, ca procesul de ingriire, fiecare pixel in parte.