Σχεδιασμός Ενσωματωμένων Συστημάτων

Εργασία 3η

Μαζαράκης Περικλής Α.Μ.: 57595

Παπαδόπουλος Αριστείδης Α.Μ.:57576

Εισαγωγή:

Σκοπός της εργασίας είναι η περαιτέρω βελτιστοποίηση του κώδικα, με επαναχρησιμοποίηση δεδομένων χρησιμοποιώντας buffers. Η συγκεκριμένη τεχνική είναι πολύ προσφιλής σε περιπτώσεις όπου χρησιμοποιούνται συνεχώς τα ίδια δεδομένα και ως στόχο έχει την μείωση των προσπελάσεων στην μνήμη και των wait states,καθώς και την εκμετάλλευση της ταχύτητας που προσφέρει η cache/SRAM. Ωστόσο, υπάρχει πιθανότητα το overhead της δημιουργίας και της προσθήκης στοιχείων σε αυτούς να υπερκερνά τα πιθανά οφέλη της χρήσης τους.

Δομή μνήμης & δημιουργία row buffers:

Η δομή μνήμης που χρησιμοποιείται είναι παραπλήσια σε ταχύτητες όπως αυτή της προηγούμενης άσκησης, με την διαφορά πως η μνήμη SRAM αυξάνεται σε μέγεθος, προκειμένου να χωράει τους buffers, που είναι τρεις στον αριθμό. Έτσι, η χρησιμοποιούμενη δομή είναι η εξής:

Αρχική θέση μνήμης	Μέγεθος Μνήμης	Όνομα μνήμης	Μέγεθος διαύλου	Είδος λειτουργίας	Χρόνοι Ανάγνωσης (N/S)	Χρόνοι Εγγραφής (N/S)
0x0000000	512KB	ROM	4Byte	Read	150/100	150/100
0x0080000	3MB	RAM	4Byte	Read/Write	50/25	50/25
0x0410C3C	4KB	SRAM	4Byte	Read/Write	1/1	1/1

Τέλος, οι buffers ορίζονται να αποθηκεύονται στην μνήμη SRAM, προκειμένου να είναι άμεσα και όσο το δυνατόν πιο γρήγορα προσπελάσιμοι.

```
#pragma arm section zidata="sram"
int i,j,k;
int rowbuffer1[M],rowbuffer2[M],rowbuffer3[M];
#pragma arm section
```

Υλοποίηση 1η (αρχείο rowbuffers1.c)**:**

Βασιζόμενοι στον αλγόριθμο επίδειξης του 3ου εργαστηρίου, επιχειρήθηκε να υλοποιηθεί κάτι παραπλήσιο, διαμορφωμένο φυσικά στο δικό μας πρόβλημα. Έτσι, χρησιμοποιούνται 3 buffers, οι οποίοι αποθηκεύονται στην πιο γρήγορη μνήμη, σε μια προσπάθεια επιτάχυνσης του προγράμματός μας. Στην διαρκή αναζήτηση τρόπων μείωσης των συνολικών κύκλων, μερικές φορές μειώθηκε η έκταση του unroll προκειμένου να αξιοποιηθεί όσο το δυνατόν καλύτερα η χρήση των buffers και το αποτέλεσμα να είναι το επιθυμητό. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα και η σύγκριση με τα βέλτιστα αποτελέσματα της προηγούμενης εργασίας:

Τοποθέτηση row buffers	Τελικοί Κύκλοι
Συνάρτηση υπολογισμού Gauss	2.618.127.255
Όπου υπάρχει συνέλιξη(Gauss + Sobel)	2.636.439.261
Βέλτιστο προηγούμενης άσκησης	2.593.877.674

Στο τελικό παραδοτέο που αντιστοιχεί σε αυτήν την υλοποίηση, έχει χρησιμοποιηθεί σε όλες τις συναρτήσεις. Εν ολίγοις, φαίνεται πως η χρήση της μεθοδολογίας του εργαστηρίου, οδηγεί σε παραπάνω κύκλους ρολογιού. Αυτή η διαπίστωση γίνεται αντιληπτή από το ακόλουθο ενδεικτικό παράδειγμα:

```
void gauss_filter(){
  printf("Create Gaussian Image...\n");
  for (i = 1; i < N-1; i++)
  {
    if (i==1){
        for (k=0;k<M;k++){ //initialize buffers outside of main loop, with first 3 rows
            rowbuffer1[k] = current_y[0][k];
            rowbuffer2[k] = current_y[1][k];
            rowbuffer3[k] = current_y[2][k];
            }
        }
    else {
    for (k=0;k<M;k++){ // fill buffers according to lab exercise
        rowbuffer1[k] = rowbuffer2[k];
        rowbuffer2[k] = rowbuffer3[k];
        rowbuffer3[k] = current_y[i][k];
    }}
      for (j = 1; j < M-1; j=j+2){
          //convolve gauss filter in grayscale image,grayscale is the Y Channel
            gauss_img[i][j] = rowbuffer1[j-1] * gauss[0][0] + rowbuffer1[j] * gauss[0][1] + rowbuffer1[j+1] * gauss[0][2]
            + rowbuffer2[j-1] * gauss[1][0] + rowbuffer2[j] * gauss[1][1] + rowbuffer2[j+1] * gauss[1][2]
            + rowbuffer3[j-1] * gauss[2][0] + rowbuffer3[j] * gauss[2][1]+ rowbuffer3[j+1] * gauss[2][2];
            gauss_img[i][j+1] = rowbuffer1[j] * gauss[0][0] + rowbuffer1[j+1] * gauss[0][1] + rowbuffer1[j+2] * gauss[0][2]
            + rowbuffer2[j] * gauss[1][0] + rowbuffer2[j+1] * gauss[1][1] + rowbuffer2[j+2] * gauss[1][2]
            + rowbuffer3[j] * gauss[2][0] + rowbuffer3[j+1] * gauss[2][1]+ rowbuffer3[j+2] * gauss[2][2];
        }
    }
}
```

Στην παραπάνω συνάρτηση, η αρχικοποίηση γίνεται 1 φορά, ενώ το γέμισμα των buffer γίνεται κάθε φορά που τρέχει η εξωτερική loop, δηλαδή για 276 φορές. Υπάρχει όμως τρόπος, να μην γίνεται αυτό το γέμισμα κάθε φορά, αλλά να μετακυλείονται τα περιεχόμενα των buffer, στην ίδια λογική μετακίνησης του παραθύρου της συνέλιξης, αντί να ξαναγεμίζονται σε κάθε επανάληψη.

Υλοποίηση 2η (αρχείο rowbuffers2.c)**:**

Επιδιώκοντας μια πιο αποτελεσματική υλοποίηση, η οποία περιέχει μεν περισσότερες αναθέσεις, αλλά έχει ως απόρροια την μη χρησιμοποίηση βρόχου επανάληψης, μειώνοντας έτσι τους κύκλους ρολογιού. Ειδικότερα, αφού αρχικοποιηθούν οι 3 buffers εκτός του «main» βρόχου, στην συνέχεια οι τιμές τους μετακυλείονται με τις επόμενες με «χειροκίνητο» τρόπο.

```
void gauss_filter(){
  for(k = 0;k<M;k++){
    rowbuffer1[k] = current_y[0][k];
    rowbuffer2[k] = current_y[1][k];
    rowbuffer3[k] = current_y[2][k];
  }
  printf("Create Gaussian Image...\n");
  for (i = 1; i < N-1; i++)
  {
      for (j = 1; j < M-1; j++){
            //convolve gauss filter in grayscale image,grayscale is the Y Channel
              gauss_img[i][j] = rowbuffer1[j-1] * gauss[0][0] + rowbuffer1[j] * gauss[0][1] + rowbuffer1[j+1] * gauss[0][2]
              + rowbuffer2[j-1] * gauss[1][0] + rowbuffer2[j] * gauss[1][1] + rowbuffer2[j+1] * gauss[1][2]
              + rowbuffer3[j-1] * gauss[2][0] + rowbuffer3[j] * gauss[2][1]+ rowbuffer3[j+1] * gauss[2][2];
            // metakulisi twn stoixeiwn twn rowbuffers
            rowbuffer1[j-1] = rowbuffer2[j-1];
            rowbuffer2[j-1] = rowbuffer3[j-1];
            rowbuffer3[j-1] = current_y[i+1][j-1];
        }
        rowbuffer1[j] = rowbuffer2[j];
        rowbuffer1[j+1] = rowbuffer2[j+1];
        rowbuffer2[j] = rowbuffer3[j];
        rowbuffer2[j+1] = rowbuffer3[j+1];
        rowbuffer3[j] = current_y[i+1][j];
        rowbuffer3[j+1] = current_y[i+1][j+1];
    }
}
```

Έτσι, παρατηρείται πως τα στοιχεία αλλάζουν τιμή σε κάθε επανάληψη, όμως με αυτόν τον τρόπο δεν υλοποιείται 3ος βρόχος επανάληψης και αλλάζουν οι τιμές με έναν πιο ορθολογικό τρόπο. Τα αποτελέσματα που προκύπτουν φαίνονται στον ακόλουθο πίνακα:

Τοποθέτηση row buffers	Τελικοί Κύκλοι
Συνάρτηση υπολογισμού Gauss	2.618.711.872
Όπου υπάρχει συνέλιξη(Gauss + Sobel)	2.641.844.966
Βέλτιστο προηγούμενης άσκησης	2.593.877.674

Παρόλα αυτά, φαίνεται πως ούτε τώρα επιτυγχάνεται ο αρχικός στόχος, δηλαδή η επιτάχυνση του αρχικού προγράμματος. Έτσι, η διαπίστωση της προηγούμενης ενότητας εφαρμόζεται και εδώ, αφού συνάγεται το συμπέρασμα πως το κόστος των αναθέσεων των ανανεώσεων των buffers υπερνικά τα οφέλη της χρήσης τους.

Συμπεράσματα - Σχόλια :

Εν κατακλείδι, εκεί που αναμέναμε βελτιστοποίηση των συνολικών κύκλων ρολογιού με την χρήση των buffers, ειδικά από την στιγμή που υπάρχει η πράξη της συνέλιξης, το αποτέλεσμα ήταν εκ διαμέτρου αντίθετο. Έτσι, σε κάθε είδος υλοποίησης των buffers, οι κύκλοι αυξήθηκαν σε κάθε περίπτωση. Η πιθανότερη αιτία του φαινομένου αυτού,

εντοπίζεται στο γεγονός ότι οι πράξεις που τρέχουν για περισσότερο χρόνο, είναι αυτές της τετραγωνικής ρίζας και της atan2, όποτε γι’ αυτό και δεν μειώθηκαν τελικά οι κύκλοι ρολογιού. Επιπλέον, δοκιμάζοντας την χρήση buffer στην συνάρτηση που υπολογίζει τις παραπάνω πράξεις, το αποτέλεσμα ήταν ελάχιστη αύξηση των clock cycles.

Σημείωση:

Υπάρχει περίπτωση κατά την εκτέλεση των παραδοτέων αρχείων να υπάρχουν ελάχιστες διαφορές στους κύκλους ρολογιών, όμως το τελικό αποτέλεσμα είναι ίδιο, δηλαδή περισσότεροι κύκλοι ρολογιού.