

3^η Εργαστηριακή Άσκηση για το μάθημα

Σχεδιασμός Ενσωματωμένων Συστημάτων

Ομάδα 29

Μέλη : Αριστοτέλης Γρίβας

Σάββας Λεβεντικίδης

Εργαστηριακή Άσκηση :

Ο σκοπός της άσκησης είναι να γίνει εισαγωγή στον προγραμματισμό Field Programmable Gate Arrays (FPGA) με High Level Synthesis (HLS). Η άσκηση αυτή παρουσιάζει ένα Recommendation System (RS) για ταινίες βασισμένο στον αλγόριθμο K Nearest Neighbors (KNN), χρησιμοποιώντας ένα υποσύνολο του MovieLens dataset. Χρησιμοποιώντας το περιβάλλον ανάπτυξης SDSoC 2016.4, θα κάνουμε βελτιστοποιήσεις στον κώδικα που μας δίνεται (και συγκεκριμένα στη συνάρτηση calcDistancesHW), ώστε να πετύχουμε γρηγορότερη εκτέλεση του κώδικα, και στη συνέχεια θα εκτελέσουμε την εφαρμογή στο Zynq-7000 ARM/FPGA SoC (Zybo).

Άσκηση 1 :

A)

Αφού εισάγουμε τα source και header files, που δίνονται από το εργαστηριακό υλικό, στο SDSoC, ορίζουμε τη συνάρτηση calcDistancesHW ως HW function και καταγράφουμε το report με τα estimated resources, μέσω της επιλογής “Estimate Performance”.

Οι απαιτούμενοι κύκλοι για την εκτέλεση της εφαρμογής καθώς και οι πόροι που χρειάστηκαν για την εκτέλεση δίνονται στην ακόλουθη εικόνα :

Details

Performance estimates for 'calcDistancesHW in main.cpp:24 ...

accelerated (Estimated cycles) 2006021

Resource utilization estimates for HW functions

Resource	Used	Total	% Utilization
DSP	2	80	2,5
BRAM	25	60	41,67
LUT	2329	17600	13,23
FF	1230	35200	3,49

Επιπλέον, καταγράφουμε τα αποτελέσματα του HLS report για κάθε loop όπως φαίνεται παρακάτω :

Detail

Instance

Loop

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_DATA_HW_TMP	133120	133120	130	-	-	1024	no
+ LOAD_DATA_HW_TMP.1	128	128	4	-	-	32	no
- LOAD_MOVIE_TMP	64	64	2	-	-	32	no
- COMPUTE_DISTS	149504	149504	146	-	-	1024	no
+ COMPUTE_DISTS.1	128	128	4	-	-	32	no
- WRITE_DISTS	2048	2048	2	-	-	1024	no

Παρατηρούμε πως το Loop το οποίο απαιτεί τους περισσότερους κύκλους και έχει το μεγαλύτερο latency είναι αυτό του “COMPUTE_DISTS” . Αυτό είναι λογικό καθώς πέρα από το μεγάλο αριθμό επαναλήψεων που γίνονται (trip count = 1024), έχουμε και περισσότερες πράξεις σε σύγκριση με τα υπόλοιπα loops (π.χ. αρχικοποίηση των sum,diff , υπολογισμός dists_hw_tmp[]).

B)

Στη συνέχεια, απενεργοποιούμε το “estimate performance” από τα options, και ενεργοποιούμε τις λειτουργίες “generate bitstream” και “generate SD card image”. Έτσι, παράγουμε το bitstream καθώς και τα αρχεία τα οποία θα χρειαστούν για να τρέξει η εφαρμογή στο zybo. Αφού περάσουμε τα αρχεία στην sd card (δεν ξεχνάμε το .csv files που περιέχουν το input dataset), κάνουμε τη σύνδεση με το zybo μέσω του minicom, κάνουμε boot το board και εκτελούμε το αρχείο .elf της κάρτας sd (βρίσκεται στο /mnt). Τα αποτελέσματα που παίρνουμε είναι τα ακόλουθα :

```
BOO1.BIN          dataset.csv          test.elf
System Volume Information image.ub
_sds              nameIdMapping.csv
sh-4.3# ./test.elf
Started reading dataset...
Finished reading dataset...
Started reading name id mapping...
Finished reading name id mapping...
Input movie id = 0
Started distance calculations on software...
Finished distance calculations on software...
Started distance calculations on hardware...
Finished distance calculations on hardware...

Recommendation system start to make inference
...
Recommendations for movie with id 0:
0. Jurassic Park (1993), with distance of 10.7121
1. Fish Called Wanda A (1988), with distance of 11.0793
2. Back to the Future (1985), with distance of 11.6404
3. Star Wars: Episode VI - Return of the Jedi (1983), with distance of 11.8849
4. Lion King The (1994), with distance of 12.0623
5. Raising Arizona (1987), with distance of 12.1552
6. Wizard of Oz The (1939), with distance of 12.1861
7. Batman (1989), with distance of 12.2066
8. Princess Bride The (1987), with distance of 12.3085
9. E.T. the Extra-Terrestrial (1982), with distance of 12.3996

Hardware cycles : 2024815
Software cycles : 1389822
Speedup          : 0.686395
Correct = 1024, Score = 1.000000
sh-4.3#
```

Όπως βλέπουμε, οι κύκλοι που εκτελούνται στο Hardware είναι αρκετά κοντά στην εκτίμηση που έγινε με το SDSoc, ενώ το speedup σε σχέση με την SW εκτέλεση στον ARM είναι ίση με 0.686395.

Γ)

Στο σημείο αυτό, θα δοκιμάσουμε διάφορα optimizations, με σκοπό να μειώσουμε τους απαιτούμενους κύκλους της εφαρμογής, πετυχαίνοντας γρηγορότερη εκτέλεση. Έτσι δοκιμάζουμε διάφορα HLS pragmas και παρατηρούμε τα αντίστοιχα performance estimations.

- Αρχίζουμε, με το **#pragma HLS pipeline II= 1**, το οποίο επιχειρεί να πετύχει το βέλτιστο δυνατό pipeline, αρχίζοντας τη δρομολόγηση της νέας επανάληψης ενός βρόχου, στον επόμενο κύκλο ρολογιού. Προφανώς για τα μονά loops, η δήλωση του directive γίνεται αμέσως μετά το for statement. Όσον αφορά,ωστόσο, στα διπλά loops, πειραματιστήκαμε τόσο στα εσωτερικά όσο και στα εξωτερικά loops, βάζοντας τα directives σε όλες τις πιθανές θέσεις. Αφού τρέξαμε τα estimations, καταλήξαμε στο συμπέρασμα πως οι βέλτιστες θέσεις για τη δήλωση του directive είναι αμέσως μετά εξωτερικό for statement για τη μέθοδο COMPUTE_DISTS, και αμέσως μετά το εσωτερικό for statement για τη μέθοδο LOAD_DATA_HW_TMP. (Προφανώς δοκιμάσαμε να βάλουμε και στα εσωτερικά και στα εξωτερικά loops ταυτόχρονα, χωρίς να έχουμε κάποια βελτίωση στην απόδοση)

Τα αποτελέσματα που πήραμε είναι τα εξής :

Estimations for pipeline II = 1 :

Details

Performance estimates for 'calcDistancesHW in main.cpp:24 ...

HW accelerated (Estimated cyc	481855
-------------------------------	--------

Resource utilization estimates for HW functions

Resource	Used	Total	% Utilization
DSP	64	80	80
BRAM	25	60	41,67
LUT	7470	17600	42,44
FF	4094	35200	11,63

Loop performance for pipeline II = 1:

Latency (clock cycles)

Summary

Latency		Interval		Type
min	max	min	max	
50253	50253	50254	50254	none

Detail

Instance

Loop

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_DATA_HW_TMP_L	32771	32771	5	1	1	32768	yes
- LOAD_MOVIE_TMP	32	32	2	1	1	32	yes
- COMPUTE_DISTS	16402	16402	35	16	1	1024	yes
- WRITE_DISTS	1024	1024	2	1	1	1024	yes

Παρατηρώντας τους κύκλους που απαιτούνται αυτή τη φορά, καταλαβαίνουμε πως έχουμε μεγάλη βελτίωση σε σύγκριση με την αρχική έκδοση του κώδικα. Αυτό, όπως βλέπουμε και από το loop performance, οφείλεται στο γεγονός πως τα περισσότερα loops είναι πλέον pipelined (II = 1), και έχουν πολύ μικρότερο latency σε σύγκριση με πριν. Ωστόσο, το COMPUTE_DISTS loop, δεν είναι πλήρως pipelined καθώς έχει II = 16. Επομένως, συνεχίζουμε τα optimizations, ώστε να πετύχουμε pipelined. (Σημειώνουμε πως οι απαιτούμενοι πόροι λόγω των pragmas έχουν αυξηθεί)

- Δοκιμάζουμε, λοιπόν, τα loop unroll directives **#pragma HLS unroll factor=X**, τα οποία στοχεύουν στη μείωση των επαναλήψεων στο εκάστοτε loop, και

στην παραλληλοποίηση της εκτέλεσης. Τονίζουμε σε αυτό το σημείο, πως δοκιμάσαμε το συγκεκριμένο directive, τόσο στα εσωτερικά, όσο στα εξωτερικά loops, και για διάφορες τιμές του unroll factor (από 2 έως 32). Ωστόσο, το μόνο σημείο στο οποίο το directive είχε θετική επίδραση ήταν στο loop **LOAD_MOVIE_TMP** και για τιμή unroll factor ίση με 2. Οποιαδήποτε άλλη τιμή του directive σε οποιοδήποτε άλλο loop, δεν είχε καμία επίδραση πέρα από τη χρήση επιπλέον πόρων, που προφανώς δεν είναι επιθυμητό. Τρέχοντας και πάλι τα estimations έχουμε τα εξής αποτελέσματα :

Estimations for pipeline 1 and loop unroll 2 :

Details

Performance estimates for 'calcDistancesHW in main.cpp:24 ...

HW accelerated (Estimated cyc	481751
-------------------------------	--------

Resource utilization estimates for HW functions

Resource	Used	Total	% Utilization
DSP	64	80	80
BRAM	26	60	43,33
LUT	7464	17600	42,41
FF	4050	35200	11,51

Loop performance for pipeline II = 1 and loop unroll = 2:

Latency (clock cycles)

Summary

Latency		Interval		Type
min	max	min	max	
50237	50237	50238	50238	none

Detail

Instance

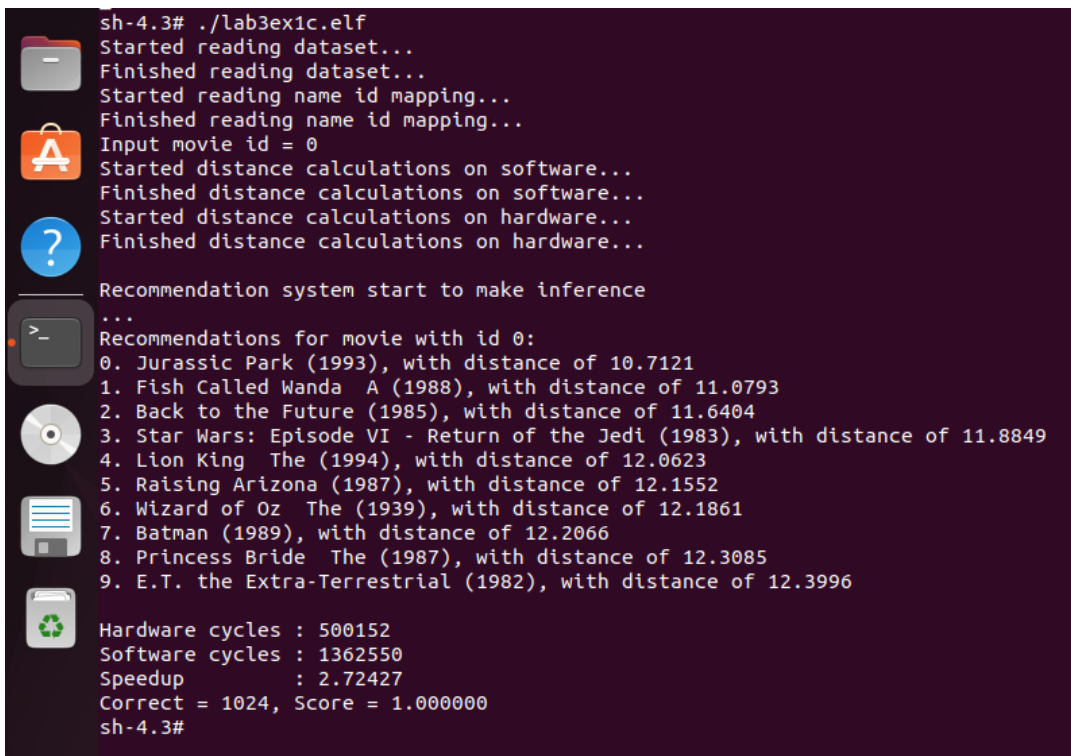
Loop

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_DATA_HW_TMP_L	32771	32771	5	1	1	32768	yes
- LOAD_MOVIE_TMP	16	16	2	1	1	16	yes
- COMPUTE_DIST	16402	16402	35	16	1	1024	yes
- WRITE_DIST	1024	1024	2	1	1	1024	yes

Παρατηρούμε πως οι κύκλοι έχουν βελτιωθεί αν και σε μικρό βαθμό, και το loop performance για το loop **LOAD_MOVIE_TMP** έχει αρκετά μικρότερο latency. Ωστόσο, η υλοποίηση μας εξακολουθεί να μην είναι πλήρως pipelined. Συνεπώς, χρειαζόμαστε κι' άλλα optimizations.

- Στο σημείο αυτό, επιχειρούμε να χρησιμοποιήσουμε το directive **#pragma HLS partition variable=<variable> <block, cyclic, complete> factor=<int> dim=<int>** στους πίνακες που χρησιμοποιούμε, σπάζοντάς τους σε μικρότερους πίνακες, ώστε να αυξήσουμε τα input/output ports και να βελτιώσουμε το memory bandwidth. Ωστόσο, τρέχοντας το performance estimation, παρατηρούμε πως αντιμετωπίζουμε ένα μεγάλο πρόβλημα, και αυτό είναι πως υπερβαίνουμε τους διαθέσιμους πόρους. Συνεπώς, δεν μπορεί να τρέξει η εφαρμογή μας στο board. Καταλαβαίνουμε, λοιπόν, πως δεν μπορούμε να συνεχίσουμε τα optimizations. Λόγω αυτού, επιστρέφουμε στον κώδικα του προηγούμενου βήματος και τρέχουμε την εφαρμογή στο zybo. (pipeline ll=1 , unroll factor = 2)

Zybo Run for pipeline 1 and unroll 2 :



```
sh-4.3# ./lab3ex1c.elf
Started reading dataset...
Finished reading dataset...
Started reading name id mapping...
Finished reading name id mapping...
Input movie id = 0
Started distance calculations on software...
Finished distance calculations on software...
Started distance calculations on hardware...
Finished distance calculations on hardware...

Recommendation system start to make inference
...
Recommendations for movie with id 0:
0. Jurassic Park (1993), with distance of 10.7121
1. Fish Called Wanda A (1988), with distance of 11.0793
2. Back to the Future (1985), with distance of 11.6404
3. Star Wars: Episode VI - Return of the Jedi (1983), with distance of 11.8849
4. Lion King The (1994), with distance of 12.0623
5. Raising Arizona (1987), with distance of 12.1552
6. Wizard of Oz The (1939), with distance of 12.1861
7. Batman (1989), with distance of 12.2066
8. Princess Bride The (1987), with distance of 12.3085
9. E.T. the Extra-Terrestrial (1982), with distance of 12.3996

Hardware cycles : 500152
Software cycles : 1362550
Speedup          : 2.72427
Correct = 1024, Score = 1.000000
sh-4.3#
```

Παρατηρούμε πως παρόλο που η σχεδίαση μας δεν είναι pipelined, τα optimizations που κάναμε έχουν οδηγήσει σε πολύ λιγότερους Hardware κύκλους, όπως έδειξε και το SDSoc, και το speedup έχει αυξηθεί σημαντικά από 0.6 στο 2.7. Έχουμε, επομένως, μία πολύ πιο γρήγορη σχεδίαση για την εφαρμογή μας σε σύγκριση με την unoptimized.

Άσκηση 2 :

Στην άσκηση αυτή, καλούμαστε να βρούμε τις βέλτιστες τιμές για το ακέραιο και το δεκαδικό τμήμα του τύπου DTYPE1 που δημιουργούμε με τη βιβλιοθήκη `ar_fixed`, αλλάζοντας τις τιμές των μεταβλητών `INT_BITS` και `DEC_BITS`. Προφανώς, οι βέλτιστες τιμές είναι οι ελάχιστες τιμές των bits που χρειάζονται για τους υπολογισμούς μας, καθώς κάθε έξτρα αχρείαστο bit, δεσμεύει χώρο στη μνήμη, με αποτέλεσμα να χρειαζόμαστε περισσότερους πόρους.

Το σημαντικό που πρέπει να προσέξουμε σε αυτό το σημείο, είναι το γεγονός πως, ως DTYPE1 ορίζουμε τις μεταβλητές που περιέχουν τη βαθμολογία ενός χρήστη για μία ταινία καθώς και τις μεταβλητές οι οποίες είναι παράγωγα πράξεων των βαθμολογιών αυτών. Συνεπώς, σκοπός μας είναι να βρούμε τον αριθμό ακέραιων bits που χρειάζονται για τη μεγαλύτερη ακέραια τιμή, και τον αριθμό δεκαδικών bits που χρειάζονται για τη πιο σύνθετη δεκαδική τιμή, για οποιαδήποτε από τις μεταβλητές που περιγράψαμε παραπάνω.

Τονίζουμε το γεγονός πως οι βαθμολογίες έχουν ακέραιο μέρος στο σύνολο {0,1,2,3,4,5} και δεκαδικό μέρος στο σύνολο { .0 , .5 }.

Με μια ματιά στον κώδικα `calcDist.cpp`, παρατηρούμε ότι το `loop` στο οποίο γίνονται πράξεις σε μεταβλητές DTYPE1, είναι το **COMPUTE_DISTS**.

Ειδικότερα σε αυτό το `loop` , παρατηρούμε πως η πιο σύνθετη πράξη που γίνεται σε DTYPE1 μεταβλητή είναι η :

`sum += diff * diff;` , η οποία επαναλαμβάνεται 32 φορές.

Γνωρίζοντας πως η μεταβλητή `diff` παίρνει την τιμή μιας βαθμολογίας, μπορούμε εύκολα να οδηγηθούμε στα εξής συμπεράσματα :

- Η μέγιστη ακέραια τιμή που μπορεί να πάρει η μεταβλητή `sum` είναι η $32 * (\max\{\text{diff} * \text{diff}\}) = 32 * (5 * 5) = 800$, συνεπώς χρειαζόμαστε το πολύ 10 bits για το ακέραιο μέρος της μεταβλητής, ώστε να μπορούμε να αναπαραστήσουμε τον αριθμό 800. (έχουμε `ufixed` -> `unsigned` αριθμό)
- Η μεταβλητή `sum` είναι άθροισμα αριθμών με δεκαδικό μέρος στο σύνολο { .0 , .25 , .5 } (αφού έχουμε πολλαπλασιασμό τιμών με δεκαδικό μέρος στο σύνολο { .0 , .5 }).
Συνεπώς, η μεταβλητή `sum` έχει δεκαδικό μέρος στο σύνολο { .0 , .25 , .5 , 0.75 } , άρα χρειαζόμαστε το πολύ 2 δεκαδικά bits για την αναπαράσταση των τιμών αυτών.

Αλλάζουμε, επομένως, τις τιμές INT_BITS, DEC_BITS του αρχείου calcDist.h, από 16,8 σε 10,2 αντίστοιχα.

Τρέχουμε, και πάλι performance estimation και έχουμε τα παρακάτω αποτελέσματα :

Estimations for pipeline 1, unroll 2 and optimized Bits:

Details

Performance estimates for 'calcDistancesHW in main.cpp:24 ...

HW accelerated (Estimated cyc	481751
-------------------------------	--------

Resource utilization estimates for HW functions

Resource	Used	Total	% Utilization
DSP	32	80	40
BRAM	13	60	21,67
LUT	5146	17600	29,24
FF	2285	35200	6,49

Loop performance for pipeline 1, unroll 2 and optimized Bits:

▣ **Latency (clock cycles)**

▣ **Summary**

Latency		Interval		Type
min	max	min	max	
50237	50237	50238	50238	none

▣ **Detail**

⊞ **Instance**

▣ **Loop**

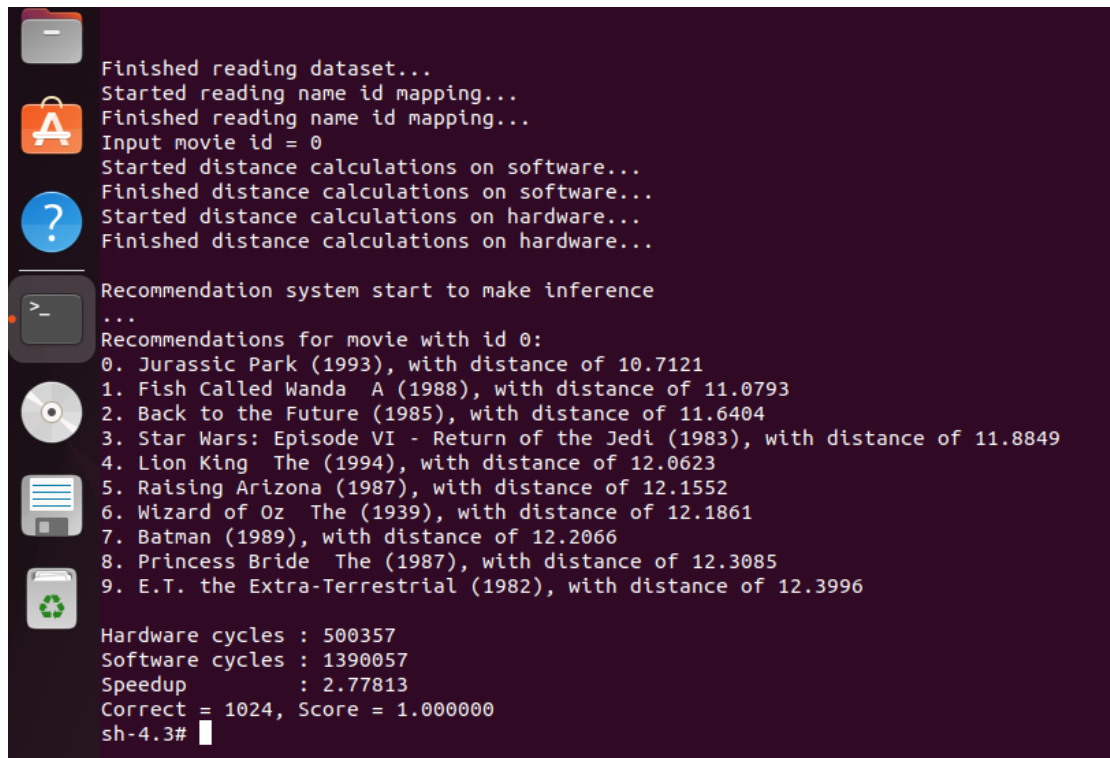
Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_DATA_HW_TMP_L	32771	32771	5	1	1	32768	yes
- LOAD_MOVIE_TMP	16	16	2	1	1	16	yes
- COMPUTE_DISTS	16402	16402	35	16	1	1024	yes
- WRITE_DISTS	1024	1024	2	1	1	1024	yes

Παρατηρούμε, πως παρόλο που οι απαιτούμενοι κύκλοι παρέμνηναν αμετάβλητοι (λογικό αφού δεν κάναμε κάποιο optimization στην υλοποίηση), οι πόροι οι οποίοι χρησιμοποιήθηκαν αυτή τη φορά είναι πολύ λιγότεροι από αυτούς με τον αρχικό αριθμό δεκαδικών και ακέραιων bits. Αυτό είναι λογικό, καθώς όπως εξηγήσαμε και

παραπάνω, περισσότερα bits για τις μεταβλητές, έχουν ως αποτέλεσμα δέσμευση περισσότερου χώρου στη μνήμη και συνεπώς την ανάγκη για επιπλέον πόρους.

Στη συνέχεια, κάνουμε generate bitstream και περνάμε όπως και στα προηγούμενα ερωτήματα ,τα αρχεία στην sd για να τα τρέξουμε στο zybo. Ακολουθούν τα αποτελέσματα :

Zybo run for pipeline 1, unroll 2 and optimized Bits:



```
Finished reading dataset...
Started reading name id mapping...
Finished reading name id mapping...
Input movie id = 0
Started distance calculations on software...
Finished distance calculations on software...
Started distance calculations on hardware...
Finished distance calculations on hardware...

Recommendation system start to make inference
...
Recommendations for movie with id 0:
0. Jurassic Park (1993), with distance of 10.7121
1. Fish Called Wanda A (1988), with distance of 11.0793
2. Back to the Future (1985), with distance of 11.6404
3. Star Wars: Episode VI - Return of the Jedi (1983), with distance of 11.8849
4. Lion King The (1994), with distance of 12.0623
5. Raising Arizona (1987), with distance of 12.1552
6. Wizard of Oz The (1939), with distance of 12.1861
7. Batman (1989), with distance of 12.2066
8. Princess Bride The (1987), with distance of 12.3085
9. E.T. the Extra-Terrestrial (1982), with distance of 12.3996

Hardware cycles : 500357
Software cycles : 1390057
Speedup          : 2.77813
Correct = 1024, Score = 1.000000
sh-4.3#
```

Παρατηρούμε πως τα αποτελέσματα μας είναι αρκετά κοντινά με αυτά που είχαμε στο προηγούμενο ερώτημα, πράγμα λογικό καθώς όπως δείξαμε η εφαρμογή μας δεν έγινε πιο γρήγορη, αλλά έχει επωφεληθεί κυρίως σε κατανάλωση πόρων. Επίσης, είναι σημαντικό το γεγονός πως έχουμε ως αποτέλεσμα τις ίδες ταινίες και αποστάσεις με τον αρχικό κώδικα, γεγονός που δείχνει ότι η μείωση των bits δεν επέφερε οποιαδήποτε αλλοίωση των αποτελεσμάτων.

Άσκηση 3 :

Τέλος, στο στάδιο αυτό, δοκιμάζουμε επιπλέον optimizations, με σκοπό να πετύχουμε pipelined υλοποίηση με τις νέες τιμές INT/DEC_Bits. Όπως προσπαθήσαμε και στην πρώτη άσκηση, θα δούμε ξανά την επίδραση του directive **#pragma HLS partition variable=<variable> <block, cyclic, complete> factor=<int> dim=<int>** . Δοκιμάζουμε να κάνουμε partition στην δεύτερη διασταση των 2D arrays, και στην πρώτη (και μοναδική) των 1D arrays. Αυτό που παρατηρούμε αυτή τη φορά, είναι πως τα resources επαρκούν για να τρέξει η εφαρμογή μας με τα

partition pragmas. Προφανώς, αυτό συμβαίνει καθώς ένας μεγάλος αριθμός από resources απελευθερώθηκαν λόγω του μικρότερου αριθμού INT/DEC_Bits, όπως δείξαμε προηγουμένως.

Δοκιμάζουμε διάφορες τιμές για το block factor, παρατηρώντας πως όσο μεγαλύτερη η τιμή του, τόσο πιο pipelined είναι η σχεδίαση μας. Εν τέλη, μετά από πειραματισμό για διάφορες τιμές, καταλήγουμε σε partition και των τριών πινάκων **data_hw_tmp**, **movie_tmp**, **dists_hw_tmp** , με block factor ίσο με 16 (για μεγαλύτερη τιμή δεν παρατηρήθηκε σημαντική βελτίωση , ενώ παράλληλα απαιτούνταν πολλά resources). Η σχεδίαση μας, είναι πλέον pipelined.

Παραθέτουμε τα αποτελέσματα τόσο από τα estimations του SDSoC ,όσο και του zybo:

Estimations for pipeline 1, unroll 2, partition and optimized Bits:

Details

Performance estimates for 'calcDistancesHW in main.cpp:24 ...

HW accelerated (Estimated cyc	392601
-------------------------------	--------

Resource utilization estimates for HW functions

Resource	Used	Total	% Utilization
DSP	32	80	40
BRAM	24	60	40
LUT	5358	17600	30,44
FF	2494	35200	7,09

Loop performance for pipeline 1, unroll 2, partition and optimized Bits:

Latency (clock cycles)

Summary

Latency		Interval		Type
min	max	min	max	
34874	34874	34875	34875	none

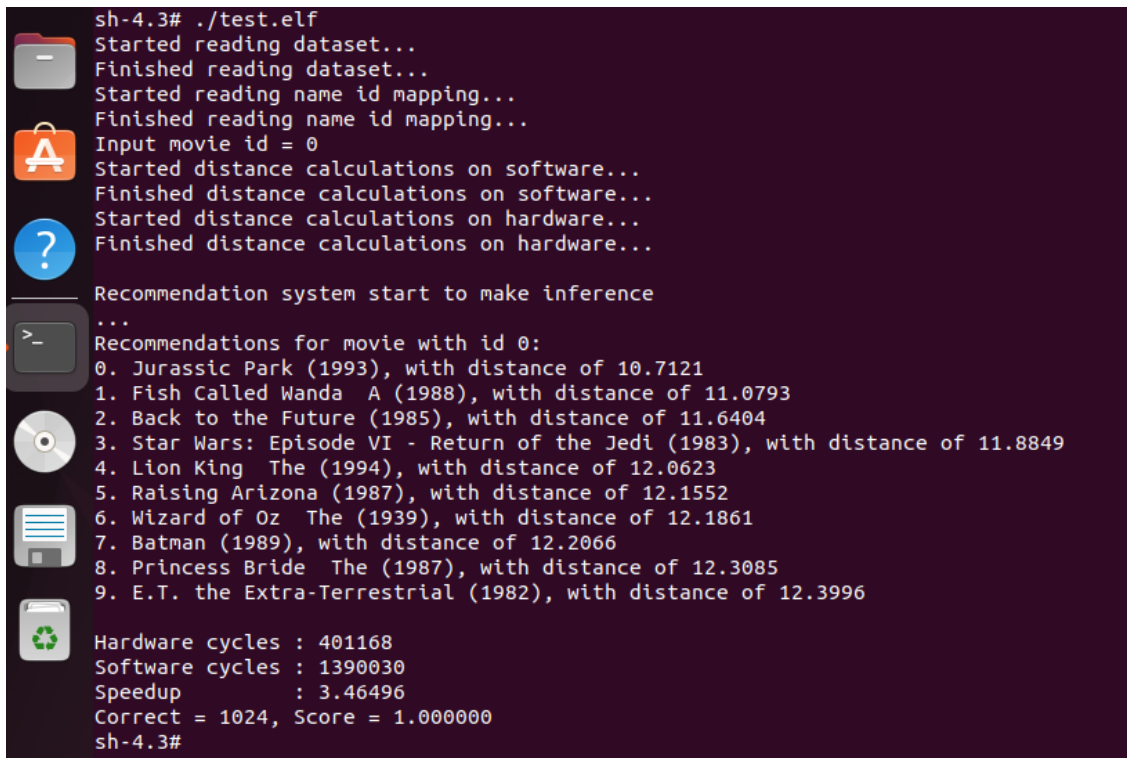
Detail

Instance

Loop

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- LOAD_DATA_HW_TMP_L	32771	32771	5	1	1	32768	yes
- LOAD_MOVIE_TMP	16	16	1	1	1	16	yes
- COMPUTE_DISTS	1054	1054	32	1	1	1024	yes
- WRITE_DISTS	1024	1024	2	1	1	1024	yes

Zybo Run for pipeline 1, unroll 2, partition and optimized Bits :



```
sh-4.3# ./test.elf
Started reading dataset...
Finished reading dataset...
Started reading name id mapping...
Finished reading name id mapping...
Input movie id = 0
Started distance calculations on software...
Finished distance calculations on software...
Started distance calculations on hardware...
Finished distance calculations on hardware...

Recommendation system start to make inference
...
Recommendations for movie with id 0:
0. Jurassic Park (1993), with distance of 10.7121
1. Fish Called Wanda A (1988), with distance of 11.0793
2. Back to the Future (1985), with distance of 11.6404
3. Star Wars: Episode VI - Return of the Jedi (1983), with distance of 11.8849
4. Lion King The (1994), with distance of 12.0623
5. Raising Arizona (1987), with distance of 12.1552
6. Wizard of Oz The (1939), with distance of 12.1861
7. Batman (1989), with distance of 12.2066
8. Princess Bride The (1987), with distance of 12.3085
9. E.T. the Extra-Terrestrial (1982), with distance of 12.3996

Hardware cycles : 401168
Software cycles : 1390030
Speedup : 3.46496
Correct = 1024, Score = 1.000000
sh-4.3#
```

Όπως βλέπουμε τόσο από το SDSoC και το HLS report, όσο και από τα αποτελέσματα από το zybo, η υλοποίηση μας είναι fully pipelined, χρειάζεται πολύ λιγότερους Hardware κύκλους από τις υπόλοιπες σχεδιάσεις, και έχει πολύ μεγαλύτερο speedup (3.46) από οποιαδήποτε από αυτές.

Όλα τα παραπάνω αποτελέσματα μας δείχνουν πως, ένας καλός συνδιασμός HLS pragmas και μιας καλής σχεδίασης του κώδικα μας, είναι κρίσιμα σημεία για τη υλοποίηση μιας τέτοιας εφαρμογής, και μπορούν να οδηγήσουν τόσο σε καλύτερη διαχείριση πόρων όσο και σε μια πολύ πιο γρήγορη και αποδοτική εφαρμογή.