High Level Synthesis
k-means clustering

Καρασακαλίδης Αλέξανδρος Ηλίας 57448

Λαζαρίδου Νίνα 57260

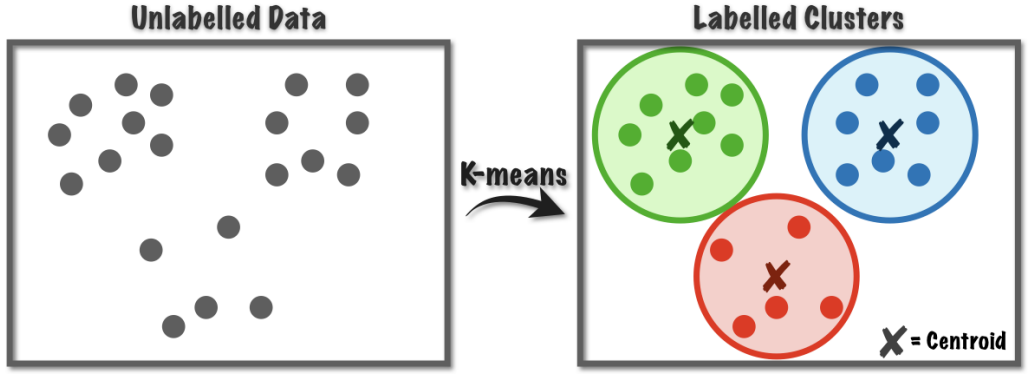
Περιγραφή Προβλήματος

Ο k-means είναι αλγόριθμος ομαδοποίησης (clustering).

Δηλαδή αντιστοιχίζει ένα σύνολο διανυσμάτων n διαστάσεων σε k ομάδες με την εξής επαναληπτική διαδικασία:

Ορίζουμε αρχικά k κέντρα (συνήθως τυχαία) και:

- Υπολογίζουμε για κάθε σημείο την ευκλείδεια απόστασή του από κάθε κέντρο και το αντιστοιχίζουμε στο κοντινότερο.
- Υπολογίζουμε νέα κέντρα παίρνοντας τον μέσο όλων των σημείων που ανήκουν στην εκάστοτε ομάδα.
- Επαναλαμβάνουμε μέχρι να συγκλίνουν τα κέντρα ή μέχρι να φτάσουμε κάποιο όριο επαναλήψεων.
- Εξάγουμε την ομάδα που ανήκει το κάθε σημείο.



Προσέγγιση Προβλήματος

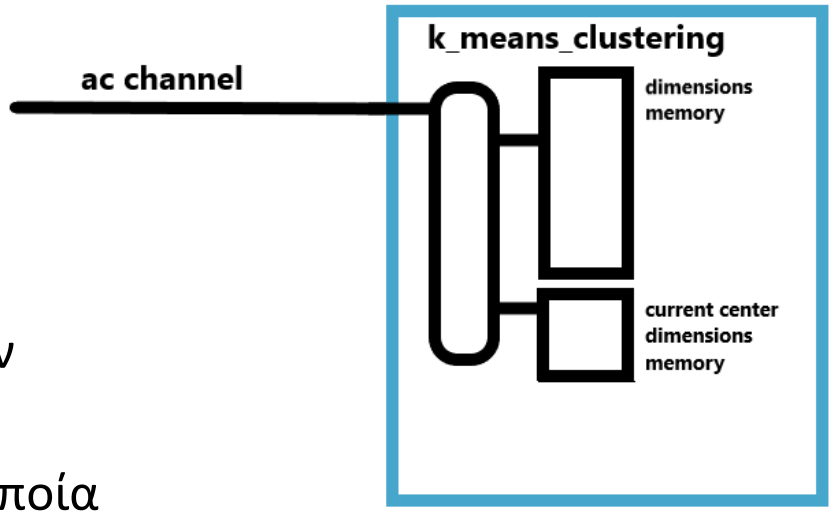
- Φτιάχνουμε ένα ολοκληρωμένο κύκλωμα ειδικού σκοπού το οποίο έχει τη δυνατότητα να παραμετροποιηθεί ως προς:
  - Το πλήθος των δεδομένων (n_points).
  - Τις διαστάσεις τους (n_dimensions).
  - Τον αριθμό των επιθυμητών cluster (k).
  - Τον μέγιστο αριθμό επαναλήψεων (max_iters).

- Ξεκινάμε την περιγραφή από τον τρόπο λήψης των δεδομένων:
  - Δεχόμαστε τα δεδομένα από ένα κανάλι εισόδου.
  - Οι 4 λέξεις που περιγράφουν τις παραπάνω παραμέτρους προηγούνται από τα δεδομένα.
  - Τα δεδομένα αποθηκεύονται έπειτα στη τοπική μνήμη ‘dimensions memory’ ενώ τα πρώτα k·n_dimensions στοιχεία αποθηκεύονται ταυτόχρονα στην ‘current center dimensions memory’ για να οριστούν ως τα πρώτα κέντρα.

- Το μέγεθος των μνημών μας επιλέχθηκε με βάση τις προδιαγραφές που θέσαμε εμείς για το κύκλωμά μας:

  - Η dimension memory θα έχει μέγεθος ίσο με το γινόμενο του πλήθους των σημείων επί τον αριθμό των διαστάσεων που τα περιγράφει. Εμείς επιλέξαμε να έχει 20.000 θέσεις.

  - Η current center dimensions memory θα έχει μέγεθος ίσο με το γινόμενο του πλήθους των κέντρων επί τον αριθμό των διαστάσεων που τα περιγράφει. Εμείς επιλέξαμε να έχει 400 θέσεις.



- Έχουμε μια επιπλέον μνήμη, την new center dimensions memory, η οποία χρησιμοποιείται για να αποθηκεύονται προσωρινά τα νέα κέντρα.

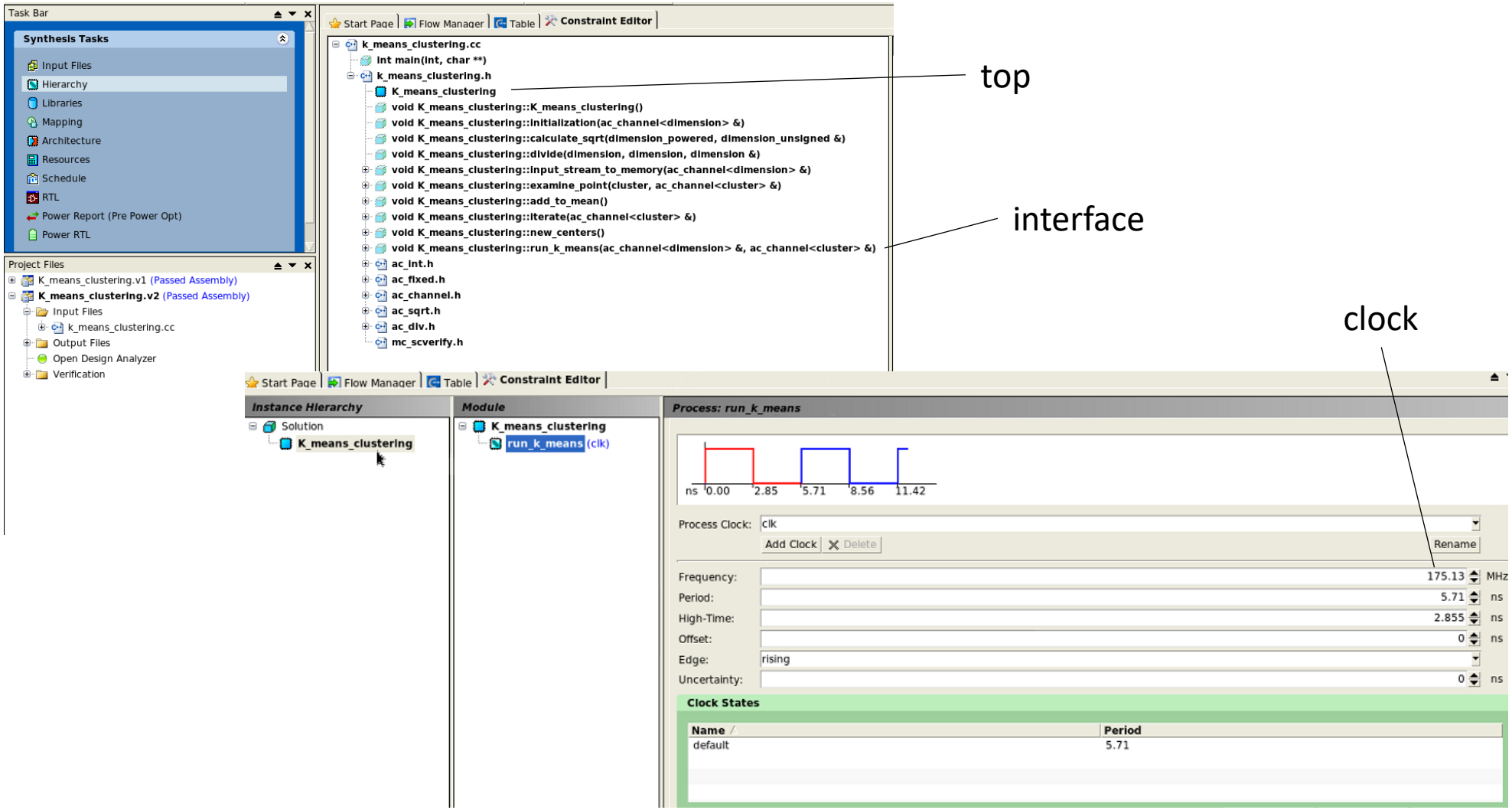
Έχουμε χωρίσει τον αλγόριθμο σε 6 συναρτήσεις τις οποίες συντονίζει η Run k Means συνάρτηση, η οποία είναι και η συνάρτηση “interface” που βλέπει το catapult:

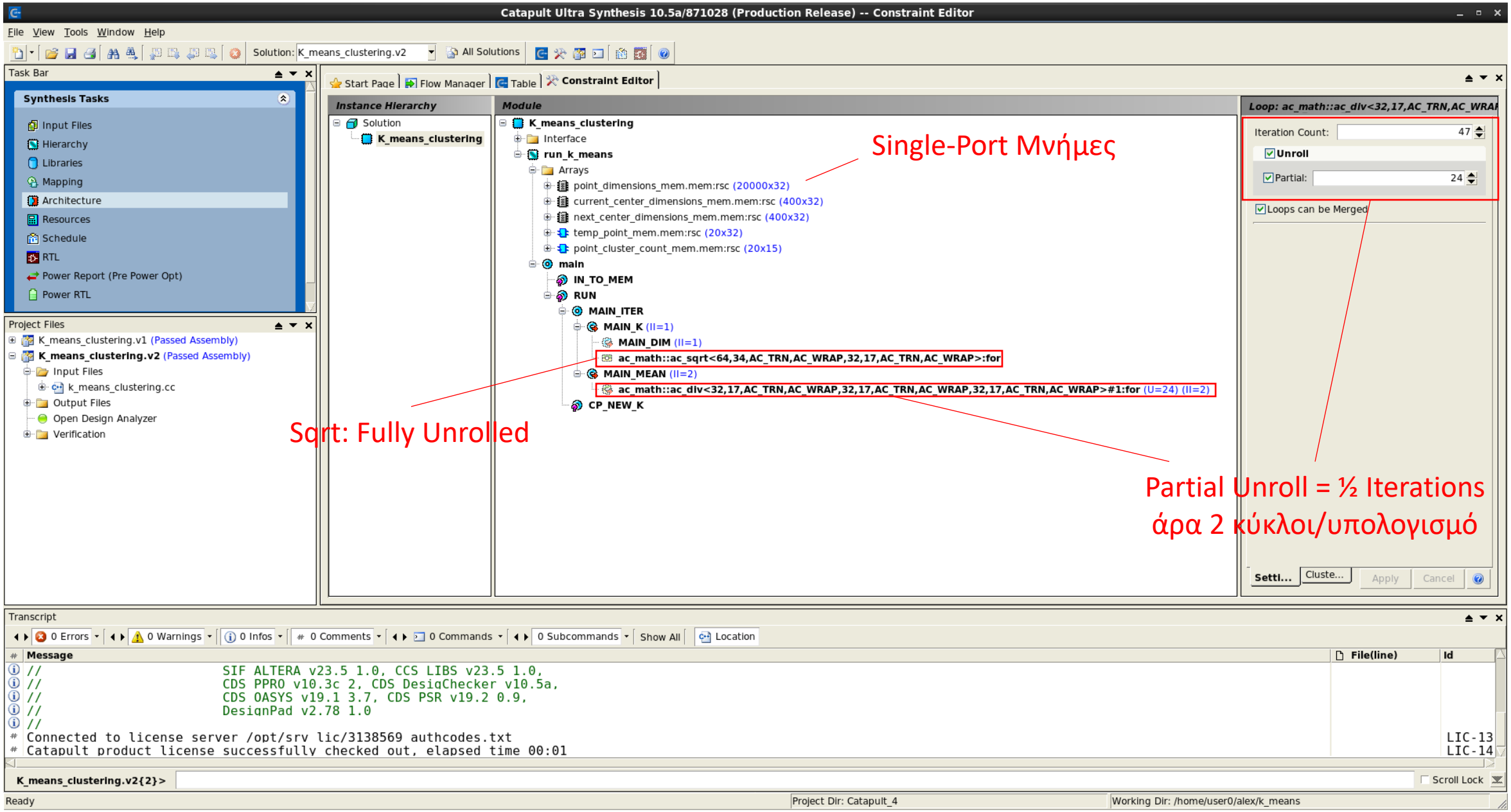
1. Initialization function: Αποθηκεύει σε καταχωρητές τις 4 πρώτες τιμές από το input channel οι οποίες είναι οι παράμετροι των δεδομένων που ακολουθούν.
2. Input Stream to Memory function: Αποθηκεύει όλα τα δεδομένα εισόδου στις τοπικές μνήμες.
3. Examine Point function: Υπολογίζει για ένα σημείο το κοντινότερό του κέντρο.
4. Add to Mean function: Υπολογίζει διαδοχικά (iteratively) τον μέσο για κάθε νέο κέντρο.
5. Iterate function: Επαναλαμβάνει τις 2,3 για κάθε σημείο που έχουμε.
6. New Centers function: Μεταφέρει τα νέα κέντρα στη μνήμη των «παλιών» και ελέγχει αν άλλαξαν (σύγκλιση στα τελικά κέντρα).

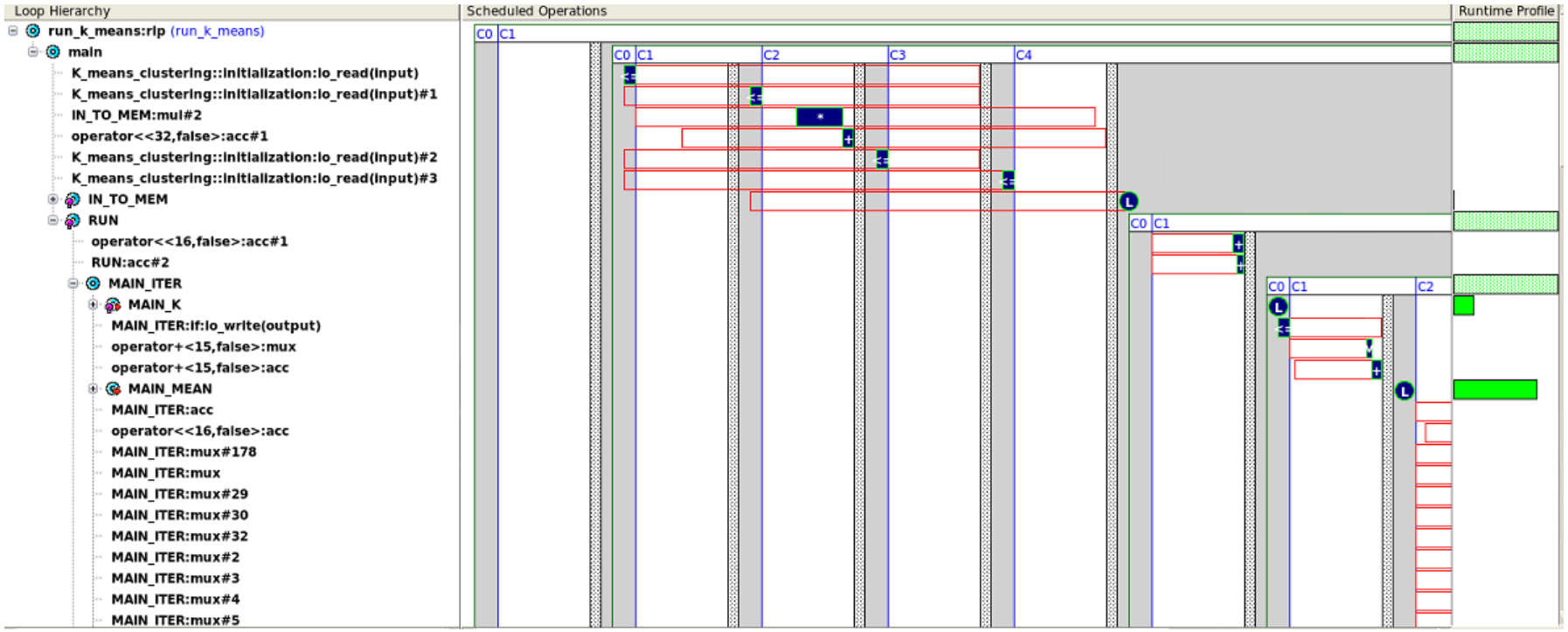
Διευκρινήσεις αναφορικά με τον κώδικα

- Examine Point
  - Για να αποφύγουμε το overflow που μπορεί να προκύψει κατά τον υπολογισμό του αθροίσματος τετραγώνων στον υπολογισμό της ευκλείδειας απόστασης του σημείου με ένα κέντρο, αναπαριστούμε το αποτέλεσμα των τετραγώνων και του αθροίσματός τους σε 64-bit unsigned fixed μεταβλητές.
  - Για να υπολογίσουμε την ρίζα, εισάγουμε την αντίστοιχη συνάρτηση από την βιβλιοθήκη ac_math, την οποία καλούμε με την “calculate_sqrt”. Η είσοδός της έχει τους 64-bit αριθμούς που περιγράψαμε παραπάνω και η έξοδός της επανέρχεται στα 32-bit.
- Add to Mean
  - Για να αποφύγουμε το overflow που μπορεί να προκύψει κατά τον υπολογισμό του μέσου για κάθε νέο κέντρο, τόσο λόγω του αθροίσματος πολλών - μεγάλων - διαστάσεων όσο και από την διαίρεση πολύ μικρών αθροισμάτων, ακολουθήσαμε μια επαναληπτική μέθοδο για τον υπολογισμό του μέσου.
  - Επειδή πρέπει να υπολογίζουμε σε κάθε iteration για κάθε σημείο που περνάει από την examine point τον μέσο για το κέντρο στο οποίο ανήκει και επειδή η πράξη της διαίρεσης είναι πολύ απαιτητική σε χρόνο, εισάγουμε την αντίστοιχη συνάρτηση από την βιβλιοθήκη ac_math, ώστε να μπορέσουμε να «σπάσουμε» την πράξη αυτή σε δύο κύκλους και να καταφέρουμε να μεγαλώσουμε το ρολόι μας αλλά και να κάνουμε καλύτερο pipeline.
- Έξοδος
  - Μετά το τελευταίο iteration, η examine point εκτελείται μια τελευταία φορά και το cluster στο οποίο βρίσκεται το κάθε σημείο με βάση τα τελευταία κέντρα που υπολογίστηκαν, εξάγονται απευθείας στο output channel.

Παραμετροποίηση στο Catapult




Catapult Ultra Synthesis 10.5a/871028 (Production Release) -- Constraint Editor
File View Tools Window Help
Solution: K_means_clustering.v2
All Solutions
Task Bar
Synthesis Tasks
Input Files
Hierarchy
Libraries
Mapping
Architecture
Resources
Schedule
RTL
Power Report (Pre Power Opt)
Power RTL
Project Files
K_means_clustering.v1 (Passed Assembly)
K_means_clustering.v2 (Passed Assembly)
Input Files
k_means_clustering.cc
Output Files
Open Design Analyzer
Verification
Start Page
Flow Manager
Table
Constraint Editor
Instance Hierarchy
Solution
K_means_clustering
Module
K_means_clustering
Interface
run_k_means
Arrays
point_dimensions_mem.mem:rsc (20000x32)
current_center_dimensions_mem.mem:rsc (400x32)
next_center_dimensions_mem.mem:rsc (400x32)
temp_point_mem.mem:rsc (20x32)
point_cluster_count_mem.mem:rsc (20x15)
main
IN_TO_MEM
RUN
MAIN_ITER
MAIN_K (II=1)
MAIN_DIM (II=1)
ac_math::ac_sqrt<64,34,AC_TRN,AC_WRAP,32,17,AC_TRN,AC_WRAP>:for
MAIN_MEAN (II=2)
ac_math::ac_div<32,17,AC_TRN,AC_WRAP,32,17,AC_TRN,AC_WRAP,32,17,AC_TRN,AC_WRAP>#1:for (U=24) (II=2)
CP_NEW_K
Loop: ac_math::ac_div<32,17,AC_TRN,AC_WRAP
Iteration Count: 47
Unroll
Partial: 24
Loops can be Merged
Setti...
Cluste...
Apply
Cancel
Single-Port Μνήμες
Sqrt: Fully Unrolled
Partial Unroll = ½ Iterations
άρα 2 κύκλοι/υπολογισμό
Transcript
0 Errors
0 Warnings
0 Infos
0 Comments
0 Commands
0 Subcommands
Show All
Location
Message
File(line)
Id
// SIF ALTERA v23.5 1.0, CCS LIBS v23.5 1.0,
// CDS PPRO v10.3c 2, CDS DesignChecker v10.5a,
// CDS OASYS v19.1 3.7, CDS PSR v19.2 0.9,
// DesignPad v2.78 1.0
//
Connected to license server /opt/srv_lic/3138569_authcodes.txt
Catapult product license successfully checked out, elapsed time 00:01
LIC-13
LIC-14
K_means_clustering.v2{2}>
Scroll Lock
Ready
Project Dir: Catapult_4
Working Dir: /home/user0/alex/k_means



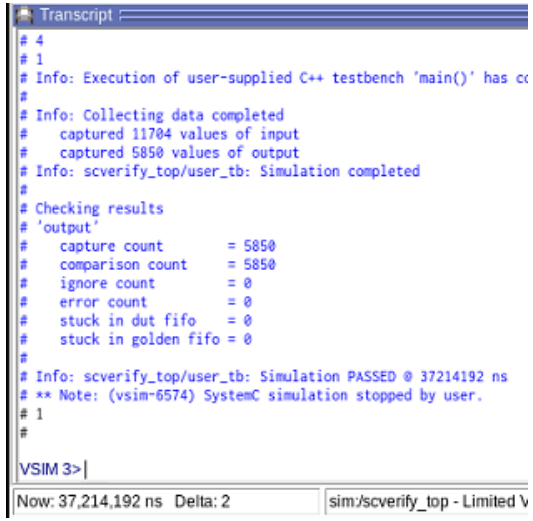
Ο μεγαλύτερος χρονικός περιορισμός είναι η διαίρεση. Εάν την σπάσουμε σε περισσότερα κομμάτια, θα μπορέσουμε να έχουμε μεγαλύτερη συχνότητα ρολογιού. Όμως θα σπάσει το pipeline μας.

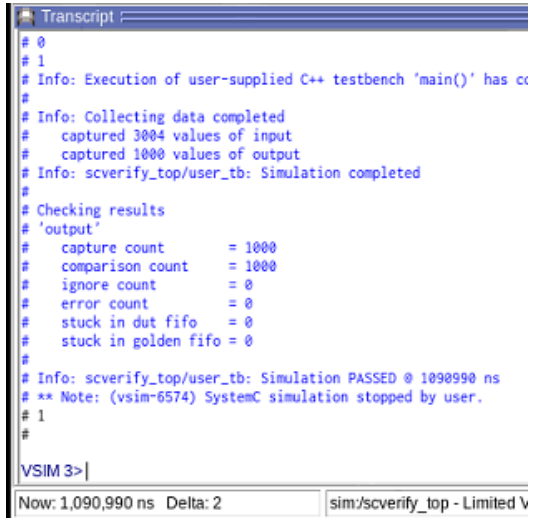
Testbenches και Αποτελέσματα

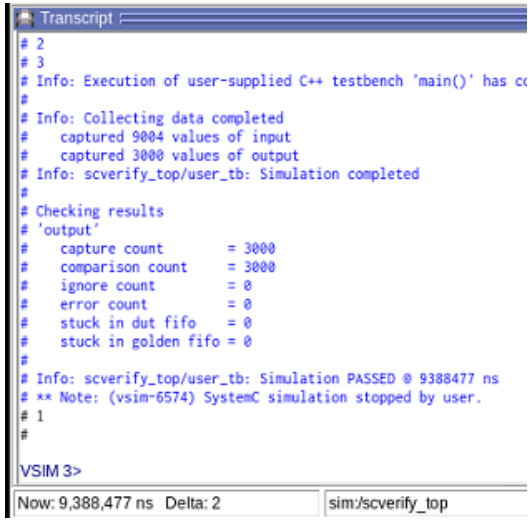
Χρησιμοποιήσαμε ένα python script (δίνεται σε pdf το notebook του) για να παράξουμε σετ δεδομένων με συγκεκριμένες κατανομές, τα οποία χρησιμοποιήσαμε ως testbenches για το κύκλωμά μας.

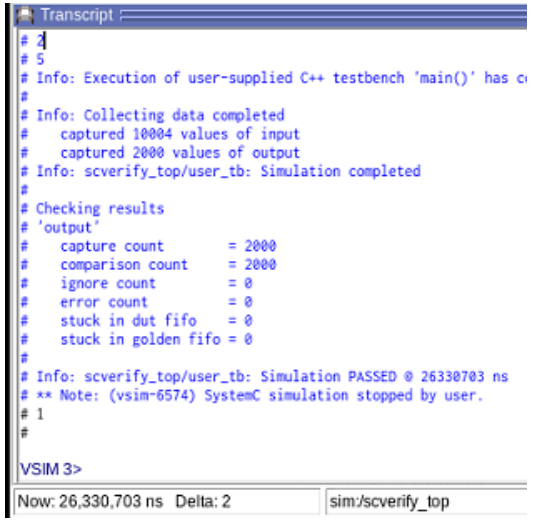
Συγκρίναμε τα αποτελέσματά μας με τα αποτελέσματα της υλοποίησης της k-means από την γνωστή βιβλιοθήκη scikit-learn, αλλά και της δικιάς μας υλοποίησης του αλγορίθμου – όπως αυτός περιγράφεται - σε python.

Σετ Α	Σετ Β	Σετ Γ	Σετ Δ
5850 σημεία 2 διαστάσεις 6 κέντρα 30 επαναλήψεις	1000 σημεία 3 διαστάσεις 3 κέντρα 50 επαναλήψεις	3000 σημεία 3 διαστάσεις 4 κέντρα 150 επαναλήψεις	2000 σημεία 5 διαστάσεις 10 κέντρα 100 επαναλήψεις

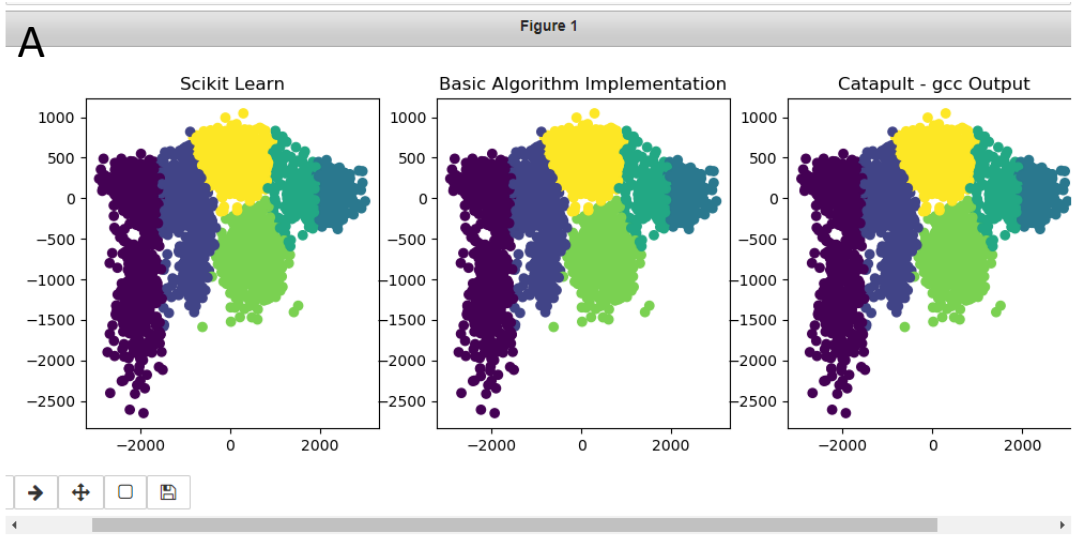








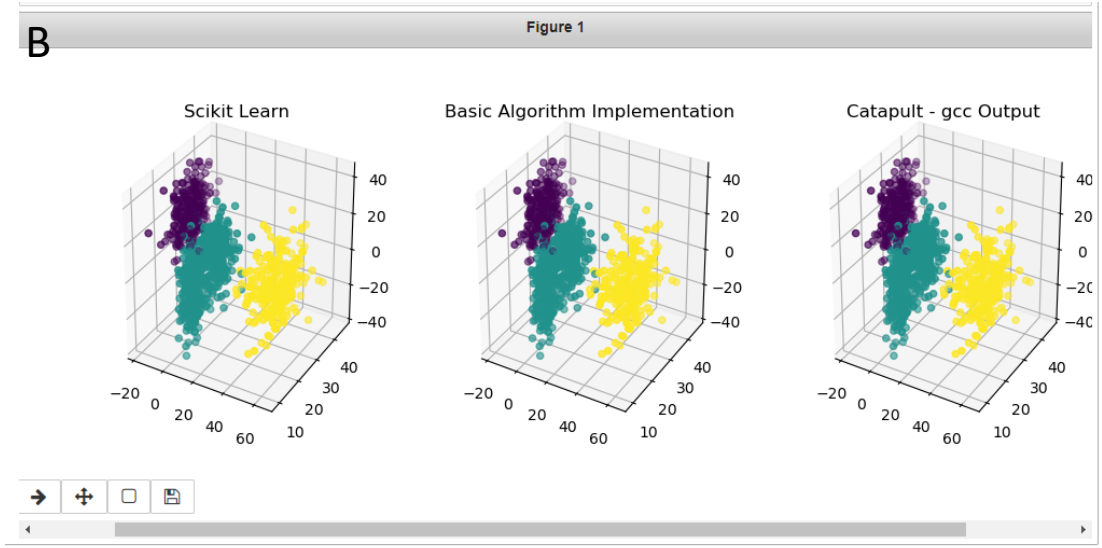
A




```
98]: print("gcc Output vs Scikit Learn: " + str(sum(labels == np.loadtxt('2d_labels.txt', dtype=int))) + '/' + str(len(labels)))
     print("gcc Output vs B.A.I.: " + str(sum(our_clusters == np.loadtxt('2d_labels.txt', dtype=int))) + '/' + str(len(labels)))
```

```
gcc Output vs Scikit Learn: 5850/5850
gcc Output vs B.A.I.: 5850/5850
```

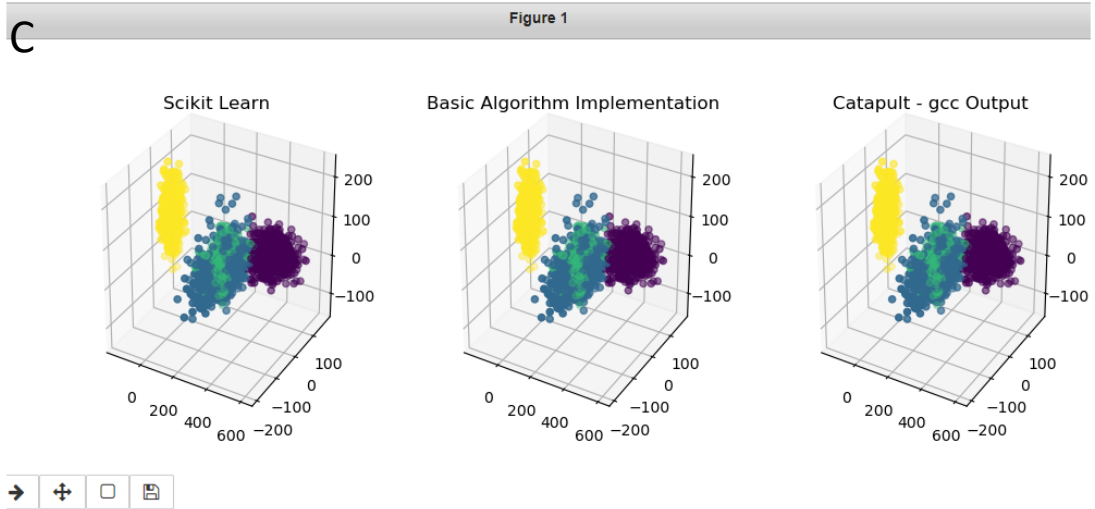
B




```
: print("gcc Output vs Scikit Learn: " + str(sum(labels == np.loadtxt('3d_a_labels.txt', dtype=int))) + '/' + str(len(labels)))
  print("gcc Output vs B.A.I.: " + str(sum(our_clusters == np.loadtxt('3d_a_labels.txt', dtype=int))) + '/' + str(len(labels)))
```

```
gcc Output vs Scikit Learn: 1000/1000
gcc Output vs B.A.I.: 1000/1000
```

C




```
|: print("gcc Output vs Scikit Learn: " + str(sum(labels == np.loadtxt('3d_b_labels.txt', dtype=int))) + '/' + str(len(labels)))
   print("gcc Output vs B.A.I.: " + str(sum(our_clusters == np.loadtxt('3d_b_labels.txt', dtype=int))) + '/' + str(len(labels)))
```

```
gcc Output vs Scikit Learn: 3000/3000
gcc Output vs B.A.I.: 3000/3000
```

D

```
print("gcc Output vs Scikit Learn: " + str(sum(labels == np.loadtxt('5d_labels.txt', dtype=int))) + '/' + str(len(labels)))
print("gcc Output vs B.A.I.: " + str(sum(our_clusters == np.loadtxt('5d_labels.txt', dtype=int))) + '/' + str(len(labels)))
```

```
gcc Output vs Scikit Learn: 2000/2000
gcc Output vs B.A.I.: 2000/2000
```