ΤΕΙ ΜΕΣΟΛΟΓΓΙΟΥ

ΤΜΗΜΑ ΤΗΛΕΠ/ΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ:

«Βελτιστοποίηση της βιβλιοθήκης OpenCV με την χρήση εξειδικευμένου hardware»

Στοιχεία Φοιτητή:

Στέφανος-Μάρκος Ρήγος

ΑΜ: 0296

Επιβλέποντες Καθηγητές:

Νικόλαος Σπ. Βώρος

Ευαγγελινός Μαριάτος

ΝΑΥΠΑΚΤΟΣ 2011

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΜΕΣΟΛΟΓΓΙΟΥ

ΤΜΗΜΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΜΑΗΟΣ 2011

ΕΥΧΑΡΙΣΤΙΕΣ

"ΘΑ ΗΘΕΛΑ ΝΑ ΕΥΧΑΡΙΣΤΗΣΩ ΤΟΝ Κ. ΝΙΚΟ ΒΩΡΟ ΟΝΤΑΣ Ο ΑΝΘΡΩΠΟΣ ΠΟΥ ΠΡΩΤΑ ΜΕ ΕΝΕΠΝΕΥΣΕ ΝΑ ΑΣΧΟΛΗΘΩ ΜΕ ΑΥΤΟ ΤΟ ΑΝΤΙΚΕΙΜΕΝΟ ΚΑΙ ΜΕΤΑ ΝΑ ΑΝΑΠΤΥΞΩ ΤΟ ΤΑΛΕΝΤΟ ΑΥΤΟ, ΤΟΝ Κ. ΕΥΑΓΓΕΛΙΝΟ ΜΑΡΙΑΤΟ ΓΙΑ ΤΗΝ ΠΟΛΥΤΙΜΗ ΒΟΗΘΕΙΑ ΤΟΥ ΚΑΤΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΤΗΣ ΠΑΡΟΥΣΑΣ ΕΡΓΑΣΙΑΣ ΚΑΙ ΤΕΛΟΣ ΤΟΥΣ ΓΟΝΕΙΣ ΜΟΥ ΓΙΑ ΤΗΝ ΥΠΟΜΟΝΗ ΤΟΥΣ ΟΣΑ ΧΡΟΝΙΑ ΦΟΙΤΗΣΑ ΣΤΗΝ ΣΧΟΛΗ."

Περιεχόμενα

Λίστα Εικόνων

Περίληψη

Τα τελευταία χρόνια η χρήση ανίχνευσης προσώπου σε πραγματικό χρόνο για την και η αναγνώριση προσώπου για παρακολούθηση, την διεπαφή ανθρώπου-μηχανής και άλλες εφαρμογές έχει αυξηθεί και, επομένως, η ανάγκη για υψηλής ισχύος, χαμηλού κόστους υλοποιήσεις έχει τεθεί. Σε ενσωματωμένες εφαρμογές, η υπολογιστική ισχύς που απαιτείται για το σύστημα ανίχνευσης προσώπου απαιτεί ειδικά σχεδιασμένο επεξεργαστή. Σε αυτή την εργασία θα συζητήσουμε υλοποιήσεις FPGA εναλλακτικών λύσεων για την επιτάχυνση της λειτουργίες χρησιμοποιούνται σε αλγορίθμων ανίχνευσης πρόσωπο της βιβλιοθήκης OpenCV. Μια σύγκριση μεταξύ του υπολογιστή που βασίζεται Haar αλγόριθμους ανίχνευσης προσώπου του OpenCV και custom hardware όρασης υπολογιστή για το FPGA θα παρουσιαστούν.

Λέξεις-κλειδιά - FPGA? Real-Time? Ανίχνευση Προσώπου? Όραση

Abstract

In recent years the use of real-time face detection and face recognition for surveillance, human-machine interfaces and other applications has increased and thus the need for high power, low cost implementations has been posed. In embedded implementations, the computing power needed for face detection system calls for a custom designed processor. In this paper we will discuss implementations of alternatives for accelerating functions used in face-detection algorithms of the OpenCV library. A comparison between computer based Haar face detection algorithms of OpenCV and custom computer vision hardware will be presented.

Keywords – Hardware acceleration; Real-Time; Face Detection; Computer Vision

1.Εισαγωγή

Οι Real-Time εφαρμογές ανίχνευσης προσώπων είναι πολλές και συνεπώς η ζήτηση για γρήγορες και ακριβείς υλοποιήσεις είναι αυξημένη. Διαφορετικές εφαρμογές επιβάλλουν ποικίλες απαιτήσεις σχετικά με την ακρίβεια ή / και την ταχύτητα. Σε ένα μεγάλο αριθμό των ενσωματωμένων εφαρμογών, η διαθέσιμη ισχύς του κεντρικού επεξεργαστή των υπολογιστών δεν μπορεί να αντιμετωπίσει την πρόκληση της ανίχνευσης προσώπου σε πραγματικό χρόνο, ως εκ τούτου, προκειμένου να επιτευχθεί μια ολοκληρωμένη και χαμηλού κόστους υλοποίηση χρησιμοποιείται μια προσαρμοσμένη εφαρμογή υλικού. Τέτοιες εφαρμογές έχουν δημιουργηθεί με τη χρήση Field Programmable Gate Arrays (FPGA).

Όταν θέλουμε να κάνουμε τη βέλτιστη χρήση των περιορισμένων πόρων που συχνά αντιμετωπίζουμε trade-offs, στην περίπτωσή μας έχουμε επικεντρωθεί γύρω από την ταχύτητα /και την ακρίβεια όπως και οι συντελεστές στάθμισης της εύκολα προσδιορίζονται με την εφαρμογή. Για να επιτευχθεί ταχύτητα πρέπει κανείς να μειώσει το μέγεθος της εικόνας επεξεργασίας και ένα μεγάλο μέρος της υπόλοιπης RAM μπορεί να χρησιμοποιηθεί ως εφεδρεία, ελαχιστοποιώντας έτσι την αργή διεπαφή προς εξωτερικά chip μνήμης RAM. Εάν έχει ανάγκη την ακρίβεια (για παράδειγμα σε ένα αίτημα αναγνώρισης χειλιών) το μέγεθος της εικόνας και την επίλυση πρέπει να αυξηθεί και το buffer θα έπρεπε να μειωθεί.

Στην εργασία αυτή θα επικεντρωθεί στα τμήματα υψηλής πληροφορικής έντασης των αλγορίθμων που είναι στο επίπεδο pixel, προ-επεξεργασίας και τον εντοπισμό των χαρακτηριστικών προσώπου. Αυτό είναι ένα μέρος το οποίο είναι ιδανικό για την χρήση επιτάχυνσης υλικού. Σε αυτό θα συγκρίνουμε εναλλακτικές υλοποιήσεις και θα μελετηθούν ανταλλαγές μεταξύ ταχύτητας και ακρίβειας.

2.OPENCV

2.1. Τι είναι η OpenCV

Η OpenCV [OpenCV] είναι μία open source (βλ. *http://opensource.org*) βιβλιοθήκη υπολογιστικής όρασης (διατίθενται εδώ http://SourceForge.net/projects/opencvlibrary) . Η βιβλιοθήκη αυτή είναι γραμμένη σε C και C + + και τρέχει σε Linux, Windows και Mac OS

X. Υπάρχει επίσης ενεργή ανάπτυξη με τις διεπαφές για Python, Ruby, Matlab, και άλλες γλώσσες.

Η OpenCV σχεδιάστηκε για υπολογιστική αποτελεσματικότητα και με ιδιαίτερη έμφαση στις εφαρμογές πραγματικού χρόνου. Η OpenCV είναι γραμμένη σε βελτιστοποιημένη C και μπορεί να επωφεληθεί τις δυνατότητες των πολυπύρηνων επεξεργαστών. Εάν επιθυμείτε περαιτέρω αυτόματη βελτιστοποίηση σε Intel αρχιτεκτονικές [Intel], μπορείτε να αγοράσετε τις ολοκληρωμένες βιβλιοθήκες [IPP] Intel Performance Primitives (IPP) , οι οποία αποτελούνται από χαμηλού επιπέδου βελτιστοποιημένες ρουτίνες σε πολλούς διαφορετικούς αλγοριθμικούς τομείς. Η OpenCV χρησιμοποιεί αυτόματα την κατάλληλη βιβλιοθήκη IPP κατά το χρόνο εκτέλεσης, εάν αυτή η βιβλιοθήκη είναι εγκατεστημένη.

Ένας από τους στόχους της OpenCV είναι να παρέχει απλές - προς - χρήση υποδομές υπολογιστικής όρασης που βοηθά τους ανθρώπους να χτίσουν αρκετά εξελιγμένες εφαρμογές όρασης γρήγορα. Η βιβλιοθήκη OpenCV περιλαμβάνει πάνω από 500 συναρτήσεις που εκτείνονται σε πολλές περιοχές υπολογιστικής όρασης, συμπεριλαμβανομένης αυτής του έλεγχου των προϊόντων εργοστασίου, της ιατρικής απεικόνισης, ασφάλεια, περιβάλλον χρήστη, ρύθμιση κάμερας, στερεοσκοπική όραση και ρομποτική . Επειδή η υπολογιστική όραση και η μηχανική μάθηση συχνά πάνε χέρι - χέρι, η OpenCV περιέχει επίσης μία πλήρη, γενικής χρήσης Βιβλιοθήκη Μηχανικής Μάθησης (MLL). Αυτή η υπό-βιβλιοθήκη επικεντρώνεται στην στατιστική αναγνώριση προτύπων και την ομαδοποίηση. Η MLL είναι ιδιαίτερα χρήσιμη για τις εργασίες όρασης που αποτελούν τον πυρήνα χρήσης της OpenCV, αλλά είναι αρκετά γενική ώστε να χρησιμοποιηθεί για οποιαδήποτε μαθησιακό πρόβλημα μηχανής.

2.2. Η Καταγωγή της OpenCV

OpenCV αναπτύχθηκε από μια έρευνα της Intel για να προωθήσει CPU-intensive εφαρμογές. Για το σκοπό αυτό, η Intel εγκαινίασε πολλά έργα, συμπεριλαμβανομένων ray tracing πραγματικού χρόνου και τοίχου 3D απεικόνισης. Ένας από τους συντάκτες που εργάζονταν για την Intel εκείνη την περίοδο επισκεπτόταν πανεπιστήμια και παρατήρησε ότι ορισμένες κορυφαίες ομάδες πανεπιστημίων, όπως το MIT Media Lab, είχε καλά σχεδιασμένη και εσωτερικά ανοιχτή υποδομή κώδικα υπολογιστικής όρασης που περνούσε από μαθητή σε μαθητή και έδινε σε κάθε νέο μαθητή ένα πολύτιμο προβάδισμα στην ανάπτυξη δικής του/της

εφαρμογή υπολογιστικής όρασης. Αντί να ανακαλύπτουμε τις βασικές λειτουργίες από την αρχή, ένας νέος φοιτητή θα μπορούσε να ξεκινήσει την κατασκευή πάνω από ότι υπήρχε πριν.

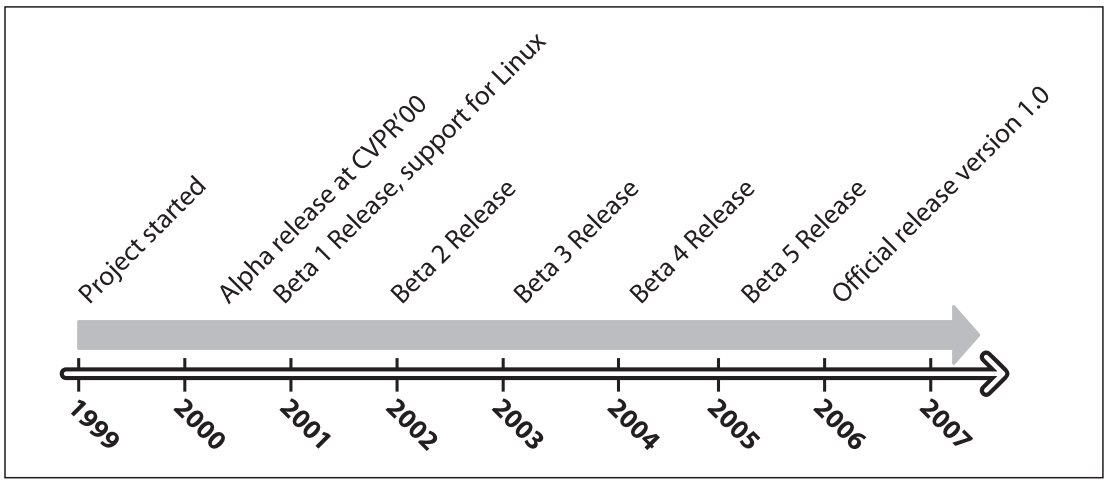
Έτσι, η OpenCV σχεδιάστηκε ως ένας τρόπος για να γίνει η υποδομή της υπολογιστικής όρασης καθολικά διαθέσιμη. Με την βοήθεια της ομάδας Επιδόσεων Βιβλιοθήκης της Intel, η OpenCV ξεκίνησε με έναν πυρήνα εφαρμοσμένου κώδικα και αλγοριθμικών προδιαγραφών που αποστέλλονται στα μέλη της Ρωσικής ομάδας βιβλιοθήκης της Intel. Αυτό είναι το «που» της OpenCV: ξεκίνησε στο εργαστήριο έρευνας της Intel σε συνεργασία με τον όμιλο Software Performance Libraries μαζί με την εφαρμογή και την εμπειρογνωμοσύνη βελτιστοποίησης στη Ρωσία. Ο κορυφαίος μεταξύ των μελών της Ρωσικής ομάδας ήταν ο Vadim Pisarevsky, ο οποίος οργάνωσε, κωδικοποίησε, και να βελτιστοποίησε ένα μεγάλο μέρος της OpenCV και εξακολουθεί να βρίσκεται στο κέντρο του ένα μεγάλο μέρος της προσπάθειας της OpenCV. Μαζί με αυτόν, ο Victor Eruhimov βοήθησε στην ανάπτυξη στις αρχές της δεκαετίας την υποδομή, και ο Valery Kuriakin που ήταν διαχειριστής του Ρωσικού εργαστηρίου και σε μεγάλο βαθμό υποστήριξε την προσπάθεια. Υπήρχαν πολλοί στόχοι για την OpenCV από την αρχή:

- Την προώθηση της έρευνας όρασης, παρέχοντας όχι μόνο ανοιχτό αλλά και βελτιστοποιημένο κώδικα για τις βασικές υποδομές της υπολογιστικής όρασης.
- Διάδοση γνώσεων υπολογιστικής όρασης, παρέχοντας μια κοινή υποδομή την οποία οι προγραμματιστές θα μπορούσαν να αξιοποιήσουν, έτσι ώστε ο κώδικας θα είναι πιο εύκολα αναγνώσιμος και μεταβιβάσιμος.
- Προχωρημένες εμπορικές εφαρμογές με βάση την υπολογιστική όραση, με φορητό, βελτιστοποιημένο κώδικα ο οποίος θα διατίθεται δωρεάν, με μια άδεια που δεν θα απαιτούν απο εμπορικές εφαρμογές να πρέπει να είναι ανοιχτού κώδικα ή απελευθερωμένες

Οι στόχοι αυτοί αποτελούν το «γιατί» της OpenCV. Η ενεργοποίηση των εφαρμογών υπολογιστικής όρασης, θα αυξήσει την ανάγκη για γρήγορους επεξεργαστές. Θα οδήγηση σε αναβαθμίσεις, σε ταχύτερους επεξεργαστές που θα αποφέρουν περισσότερα έσοδα για την Intel από την πώληση ορισμένων περαιτέρω εφαρμογών . Ίσως αυτός είναι ο λόγος για αυτήν τον ανοιχτό και δωρεάν κώδικα που προέκυψε από τον προμηθευτή του υλικού και όχι μία

εταιρεία λογισμικού. Κατά μία άποψη, υπάρχει περισσότερος χώρος για να είναι κάποιος καινοτόμος σε λογισμικό στο εσωτερικό μιας εταιρίας υλικού.

Σε κάθε προσπάθεια ανοικτού κώδικα, είναι σημαντικό να επιτευχθεί ένα κρίσιμο ποσοστό κατά το οποίο το έργο θα καταστεί αυτοσυντηρούμενο. Έχουν υπάρξει τώρα περίπου δύο εκατομμύρια downloads της OpenCV, και αυτός ο αριθμός αυξάνεται κατά μέσο όρο 26.000 downloads το μήνα. Η ομάδα χρηστών προσεγγίζει σήμερα τα 20.000 μέλη. Η OpenCV λαμβάνει πολλές συνεισφορές των χρηστών, και η κεντρική ανάπτυξη έχει σε μεγάλο βαθμό κινηθεί έξω από το παρελθόν χρονοδιάγραμμα της Intel για την OpenCV όπως φαίνεται στο Σχήμα 1. Στην πορεία, η OpenCV επηρεάστηκε από την dot com έκρηξη και επίσης με πολλές αλλαγές διαχείρισης και κατεύθυνσης. Κατά τη διάρκεια αυτών των διακυμάνσεων, υπήρξαν φορές που κανένας στην Intel δεν εργαζόταν καθόλου στην OpenCV . Ωστόσο, με την έλευση των πολυπύρηνων επεξεργαστών και τις πολλές νέες εφαρμογές της υπολογιστικής όρασης, η τιμή της OpenCV άρχισε να αυξάνεται. Σήμερα, η OpenCV είναι ένας ενεργός τομέας της ανάπτυξης σε διάφορα όργανα, έτσι αναμένουμε να δούμε πολλές ενημερώσεις σε περίπτωση πολλαπλής βαθμονόμησης κάμερας, αντίληψη βάθους, οι μέθοδοι για την ανάμειξη υπολογιστικής όρασης με ανιχνευτές ακτίνας λέιζερ, και την καλύτερη αναγνώριση προτύπων, καθώς και μια σημαντική υποστήριξη για την ανάγκη ρομποτικής όρασης.



Σχήμα 1. Χρονοδιάγραμμα OpenCV

2.3. Ποιος χρησιμοποιεί την OpenCV

Οι περισσότεροι επιστήμονες της πληροφορικής και προγραμματιστές γνωρίζουν κάποια πτυχή του ρόλου που παίζει υπολογιστική όραση. Αλλά λίγοι άνθρωποι γνωρίζουν όλους τους τρόπους με τους οποίους χρησιμοποιείται η υπολογιστική όραση. Για παράδειγμα, οι περισσότεροι άνθρωποι κάπως γνωρίζουν την χρήση της στην επιτήρηση, και πολλοί γνωρίζουν επίσης ότι χρησιμοποιείται όλο και περισσότερο για τις εικόνες και βίντεο στο Web. Λίγοι έχουν δει κάποια χρήση της υπολογιστικής όρασης σε διεπαφές παιχνιδιών. Ωστόσο, λίγοι άνθρωποι συνειδητοποιούν ότι τα περισσότερες εναέριες και street-map εικόνες (όπως στο Google Street View) κάνουν χρήση της βαθμονόμησης της κάμερας και τεχνικές ραψίματος εικόνας. Μερικοί γνωρίζουν εξειδικευμένες εφαρμογές για την παρακολουθήσεις ασφάλειας, των μη επανδρωμένων τηλεκατευθυνόμενων οχημάτων, ή βιοϊατρικής ανάλυσης. Αλλά λίγοι γνωρίζουν πόσο διαδεδομένη η μηχανική όρασης έχει γίνει στον τομέα της μεταποίησης, σχεδόν ό, τι είναι μαζικής παραγωγής έχει περάσει αυτόματη επιθεώρηση σε κάποιο σημείο με την χρήση υπολογιστικής όρασης.

Η άδεια ανοικτού κώδικα της OpenCV έχει δομηθεί έτσι ώστε να μπορείτε να χτίσετε ένα εμπορικό προϊόν που χρησιμοποιεί το σύνολο ή μέρος της OpenCV. Δεν υποχρεούστε να δώσετε άδεια ανοικτού κώδικα στο προϊόν σας ή να επιστρέψετε βελτιώσεις στο δημόσιο κομμάτι. Εν μέρει λόγω αυτών των φιλελεύθερων όρων αδειοδότησης, υπάρχει μια μεγάλη κοινότητα χρηστών που περιλαμβάνει ανθρώπους από μεγάλες εταιρείες (IBM, Microsoft, Intel, SONY, η Siemens, και την Google) και τα ερευνητικά κέντρα (όπως το Stanford, MIT , CMU, το Cambridge, και INRIA). Η OpenCV είναι δημοφιλές σε όλο τον κόσμο, με μεγάλες κοινότητες χρηστών στην Κίνα, την Ιαπωνία, τη Ρωσία, την Ευρώπη και το Ισραήλ.

Από την έκδοση alpha της τον Ιανουάριο του 1999, η OpenCV έχει χρησιμοποιηθεί σε πολλές εφαρμογές, προϊόντα, και ερευνητικές προσπάθειες. Οι εφαρμογές αυτές περιλαμβάνουν την μεταξύ τους ραφή εικόνων σε δορυφόρο και web χάρτες, σάρωση εικόνας και ευθυγράμμιση, ιατρική μείωση του θορύβου της εικόνας, ανάλυση αντικειμένου, ασφάλεια και συστήματα ανίχνευσης κινήσεων, αυτόματα συστήματα παρακολούθησης και ασφάλειας, την κατασκευή συστημάτων ελέγχου, βαθμονόμηση κάμερας, στρατιωτικές εφαρμογές, καθώς και μη επανδρωμένα εναέρια, εδάφους, και υποβρύχια οχήματα. Έχει επίσης χρησιμοποιηθεί σε μουσική και αναγνώριση ήχου, όπου οι τεχνικές αναγνώρισης οράματος που εφαρμόζεται

στις εικόνες φασματογραφημάτων ήχου. Η OpenCV ήταν ένα βασικό μέρος του συστήματος όρασης στα ρομπότ από το Στάνφορντ, «Stanley», η οποία κέρδισε το βραβείο $ 2M DARPA Grand Challenge.

2.4. Τι είναι η υπολογιστική όραση

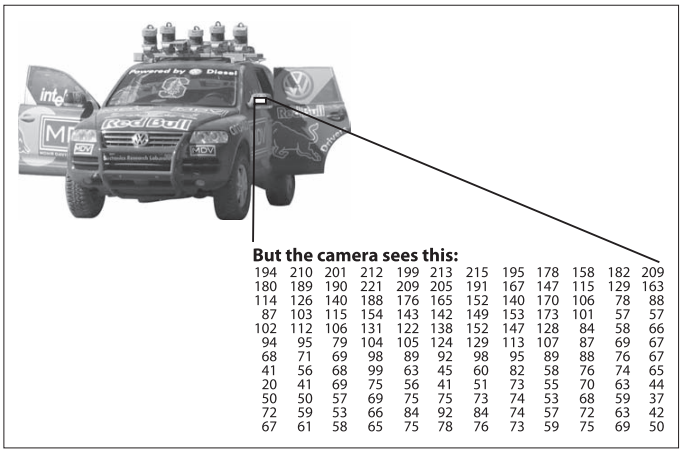
Η υπολογιστική όραση είναι η μετατροπή των δεδομένων από μία φωτογραφία ή βιντεοκάμερα, είτε σε μια απόφαση ή μια νέα αναπαράσταση. Όλοι αυτοί οι μετασχηματισμοί γίνονται για την επίτευξη ορισμένων συγκεκριμένων στόχων. Τα δεδομένα εισόδου μπορεί να περιλαμβάνουν ορισμένες συμφραζόμενες πληροφορίες όπως «η κάμερα είναι τοποθετημένη σε ένα αυτοκίνητο" ή "λέιζερ-αποστασιόμετρο δείχνει ένα αντικείμενο ότι είναι 1 μέτρο μακριά". Η απόφαση θα μπορούσε να είναι : «υπάρχει ένα πρόσωπο σε αυτή τη σκηνή» ή «υπάρχουν 14 καρκινικά κύτταρα σε αυτή τη διαφάνεια". Μια νέα αναπαράσταση μπορεί να σημαίνει μετατροπή μιας έγχρωμης εικόνας σε μια κλίμακα του γκρι ή την αφαίρεση της κίνησης της κάμερας από μια ακολουθία εικόνων.

Επειδή είμαστε τόσο οπτικά πλάσματα, είναι εύκολο να νομίσετε ότι τα καθήκοντα της υπολογιστικής όρασης είναι εύκολα. Πόσο δύσκολο μπορεί να είναι να βρει, ας πούμε, ένα αυτοκίνητο όταν το κοιτάς σε μια εικόνα; Η αρχική διαίσθησή σας αρχικά μπορεί να είναι αρκετά παραπλανητική. Ο ανθρώπινος εγκέφαλος χωρίζει το σήμα όρασης σε πολλά κανάλια, τα οποία μεταφέρουν διαφορετικά είδη πληροφοριών στον εγκέφαλό σας. Ο εγκέφαλός σας έχει ένα σύστημα που εστιάζει την προσοχή στα σημαντικά τμήματα της εικόνας για εξέταση, ενώ μειώνει την εξέταση των άλλων περιοχών. Υπάρχει μαζική ανατροφοδότηση κατά την οπτική ροή που είναι, μέχρι στιγμής, ελάχιστα κατανοητή. Υπάρχουν εκτεταμένες εισροές από τους αισθητήρες ελέγχου των μυών και όλες τις άλλες αισθήσεις που επιτρέπουν στον εγκέφαλο να αξιοποιήσει την πολλαπλές ενώσεις από τα χρόνια της ζωής του στον κόσμο. Τα συστήματα ανατροφοδότησης στον εγκέφαλο πάνε πίσω σε όλα τα στάδια της μεταποίησης, συμπεριλαμβανομένων των ίδιων αισθητήρων υλικού (μάτια), τα οποία μηχανικά ελέγχουν τον φωτισμό, μέσω της ίριδας και συντονίζουν την υποδοχή στην επιφάνεια του αμφιβληστροειδούς.

Σε ένα σύστημα μηχανικής όρασης, ωστόσο, ο υπολογιστής λαμβάνει ένα πλέγμα αριθμών από την κάμερα ή από το δίσκο, και αυτό είναι όλο. Ως επί το πλείστον, δεν υπάρχει

ενσωματωμένη αναγνώριση προτύπων, δεν έχει αυτόματο έλεγχο της εστίασης και διαφράγματος, ούτε να τα ελέγξει με βάση την εμπειρία. Ως επί το πλείστον, τα συστήματα όρασης είναι ακόμα πολύ αφελείς. Το Σχήμα 2 δείχνει την εικόνα ενός αυτοκινήτου. Σε αυτή την εικόνα βλέπουμε έναν καθρέφτη στην πλευρά του οδηγού του αυτοκινήτου. Αυτό που ο υπολογιστής "βλέπει" είναι απλά ένα πλέγμα αριθμών. Κάθε δεδομένος αριθμός σε αυτό το πλέγμα έχει ένα αρκετά μεγάλο ποσοστό θορύβου και έτσι από μόνο του μας δίνει λίγες πληροφορίες, αλλά αυτό το πλέγμα αριθμών "βλέπει" ένας υπολογιστής. Το έργο μας είναι τότε να μετατραπεί αυτό το θορυβώδες πλέγμα αριθμών σε αντίληψη: «καθρέφτης». Το Σχήμα 3 δίνει κάποιες περισσότερες πληροφορίες γιατί η υπολογιστική όραση είναι τόσο δύσκολη.

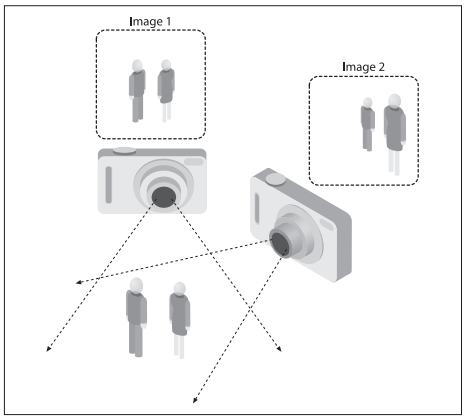
Στην πραγματικότητα, το πρόβλημα, όπως την έχουμε θέσει μέχρι στιγμής, είναι χειρότερο από δύσκολο. Είναι τυπικά αδύνατο να λυθεί. Λαμβάνοντας υπόψη μία δισδιάστατη (2D) προβολή ενός 3D κόσμου, δεν υπάρχει μοναδικός τρόπος για να ανακατασκευαστεί το 3D σήμα. Τυπικά, ένα τέτοιο κακό πρόβλημα δεν έχει μοναδική ή οριστική λύση. Η ίδια 2D εικόνα θα μπορούσε να αντιπροσωπεύει κάθε ενός άπειρου συνδυασμού 3D σκηνή, ακόμα και αν τα δεδομένα ήταν τέλεια. Ωστόσο, όπως ήδη αναφέρθηκε, τα στοιχεία είναι κατεστραμμένα από το θόρυβο και τις παραμορφώσεις. Αυτή η παραμόρφωση πηγάζει από τις διακυμάνσεις στον κόσμο (καιρικές συνθήκες, φωτισμός, αντανακλάσεις, κινήσεις), ατέλειες στις μηχανικές ρύθμισης του φακού, πεπερασμένο χρόνο ένταξης στον αισθητήρα (motion blur), ηλεκτρικό θόρυβο στον αισθητήρα ή άλλα ηλεκτρονικά είδη, και την συμπίεση, μετά την λήψης εικόνας. Με δεδομένες αυτές τις σημαντικές προκλήσεις, πώς μπορούμε να κάνουμε οποιαδήποτε πρόοδο;



Σχήμα 2. Για έναν υπολογιστή, η πλευρά του καθρέφτη ενός αυτοκινήτου είναι απλά ένα πλέγμα αριθμών

Κατά το σχεδιασμό ενός πρακτικού συστήματος, επιπλέον συμφραζόμενη γνώση συχνά μπορεί να χρησιμοποιηθεί για την εργασία γύρω από τους περιορισμούς που μας επιβλήθηκαν από τους οπτικούς αισθητήρες. Εξετάστε το παράδειγμα ενός κινητού ρομπότ που πρέπει να βρει και να πάρει συρραπτικά σε ένα κτίριο. Το ρομπότ θα μπορούσε να χρησιμοποιήσει γεγονότα όπως ένα γραφείο που είναι ένα αντικείμενο που βρίσκετε μέσα σε γραφεία και που τα συρραπτικά βρίσκονται συνήθως σε γραφεία. Αυτό δίνει μια έμμεση αναφορά του μεγέθους. Τα συρραπτικά πρέπει να μπορούν να χωρέσουν σε γραφεία. Βοηθά επίσης στην εξάλειψη των ψευδώς «αναγνωρισμένα» συρραπτικά σε αδύνατους χώρους (π.χ., στην οροφή ή ένα παράθυρο). Το ρομπότ μπορεί να αγνοήσει με ασφάλεια ένα 200-ποδιών μικρό αερόστατο διαφήμιση που διαμορφώνεται όπως ένα συρραπτικό επειδή το μικρό αερόστατο στερείται την προϋπόθεση κόκκων ξύλου στον φόντο ενός γραφείου. Αντίθετα, με καθήκοντα όπως η ανάκτηση εικόνας, όλες οι εικόνες συρραπτικών που βρίσκονται σε μια βάση δεδομένων μπορεί να είναι πραγματικά συρραπτικά σε μεγάλα μεγέθη και άλλες ασυνήθιστες διαμορφώσεις που μπορούν να είναι έμμεσα εμπόδια από τις υποθέσεις των ατόμων που έβγαλαν φωτογραφίες. Δηλαδή, ο φωτογράφος πήρε πιθανότατα εικόνες μόνο του πραγματικού, η κανονικού μεγέθους συρραπτικά. Οι άνθρωποι τείνουν επίσης να τοποθετούν τα αντικείμενα στο κέντρο κατά τη λήψη φωτογραφιών και τείνουν να τους βάζουν σε χαρακτηριστικές κατευθύνσεις. Έτσι, υπάρχει συχνά αρκετά ένα μεγάλο κομμάτι της ακούσιας άρρητης πληροφορίας σε φωτογραφίες που λήφθηκαν από ανθρώπους.

Τα συγκυριακά στοιχεία μπορούν επίσης να διαμορφωθούν ρητά με την εκμάθηση τεχνικών μηχανής. Κρυφές μεταβλητές όπως το μέγεθος, ο προσανατολισμός προς τη βαρύτητα, και ούτω καθεξής μπορεί στη συνέχεια να συσχετιστούν με τις αξίες τους σε ένα Σήμα κατάρτισης. Εναλλακτικά, μπορεί κανείς να επιχειρήσει να μετρήσει κρυμμένες μεταβλητές μεροληψίας με τη χρήση επιπρόσθετων αισθητήρων. Η χρήση ενός αποστασιομετρικού λέιζερ για να μετρηθεί το βάθος μας επιτρέπει να μετρηθεί με ακρίβεια το μέγεθος ενός αντικειμένου.



Σχήμα 3. Η άρρωστη φύση της όρασης: η 2D απεικόνιση των αντικειμένων μπορεί να αλλάξει ριζικά ανάλογα με την γωνία λήψης

Το επόμενο πρόβλημα που αντιμετωπίζει η υπολογιστική όραση είναι ο θόρυβος. Εμείς συνήθως ασχολούμαστε με το θόρυβο με τη χρήση στατιστικών μεθόδων. Για παράδειγμα, μπορεί να είναι αδύνατο να ανιχνευτεί μία ακμή σε μια εικόνα μόνο με τη σύγκριση ενός σημείου με τους άμεσους γείτονές της. Αλλά αν κοιτάξουμε τις στατιστικές για μια τοπική περιοχή, η ανίχνευση ακμών γίνεται πολύ πιο εύκολη. Ένα πραγματικό άκρο θα πρέπει να εμφανίζεται σε μια σειρά από τέτοιες άμεσες απαντήσεις γείτονα για μια τοπική περιοχή, εκ των οποίων ο προσανατολισμός είναι συνεπής με τους γείτονές του. Είναι επίσης δυνατό να αντισταθμίσει το θόρυβο με τη λήψη στατιστικών διαχρονικά. Ακόμα, άλλες τεχνικές λαμβάνουν υπόψη τον θόρυβο ή τις στρεβλώσεις με βάση ρητά μοντέλα που έμαθαν άμεσα από τα διαθέσιμα δεδομένα. Για παράδειγμα, επειδή οι στρεβλώσεις φακού είναι πλήρως κατανοητές, μια ανάγκη είναι να μάθουν μόνο τις παραμέτρους για ένα απλό πολυωνυμικό μοντέλο για να τις περιγράψει και συνεπώς να διορθώσει σχεδόν εντελώς τις εν λόγω στρεβλώσεις.

Οι ενέργειες ή αποφάσεις που η υπολογιστική όραση επιχειρεί και που βασίζονται στην κάμερα δεδομένων και εκτελούνται μέσα στο πλαίσιο ενός συγκεκριμένου σκοπού ή εργασίας. Μπορεί να θέλετε να αφαιρέσετε το θόρυβο ή την ζημία από μια εικόνα, έτσι ώστε το σύστημα ασφαλείας μας να προβεί σε συναγερμό, εάν κάποιος προσπαθεί να αναρριχηθεί σε έναν φράκτη ή γιατί χρειαζόμαστε ένα σύστημα παρακολούθησης που μετράει πόσα άτομα περνάνε μέσα από μια περιοχή σε ένα πάρκο ψυχαγωγίας. Προγράμματα όρασης για ρομπότ που περιπλανώνται μέσα από κτίρια γραφείων θα εφαρμόζουν διαφορετικές στρατηγικές από ό,τι προγράμματα όρασης για σταθερές κάμερες ασφαλείας, καθώς τα δύο συστήματα έχουν πολύ διαφορετικά περιβάλλοντα και στόχους. Κατά γενικό κανόνα: όσο πιο περιορισμένο

πλαίσιο υπολογιστικής όρασης υπάρχει, τόσο περισσότερο μπορούμε να βασιζόμαστε σε αυτά για την απλοποίηση του προβλήματος και την εύρεση πιο αξιόπιστης λύσης.

Η OpenCV στοχεύει στην παροχή των βασικών εργαλείων που απαιτούνται για την επίλυση των προβλημάτων όρασης. Σε ορισμένες περιπτώσεις, υψηλού επιπέδου λειτουργίες στη βιβλιοθήκη θα είναι επαρκής για την επίλυση των πιο περίπλοκων προβλημάτων της υπολογιστικής όρασης. Ακόμα και όταν αυτό δεν συμβαίνει, τα βασικά συστατικά της βιβλιοθήκης είναι επαρκώς πλήρεις για να παράσχουν δυνατότητες για δημιουργία μιας ολοκληρωμένης λύσης της δικής σας σε σχεδόν οποιοδήποτε πρόβλημα όρασης. Στην τελευταία περίπτωση, υπάρχουν αρκετές δοκιμασμένες και αληθινές μέθοδοι που χρησιμοποιούν τη βιβλιοθήκη. Όλοι τους αρχίζουν με την επίλυση του προβλήματος, χρησιμοποιώντας όσο ποιο πολλά διαθέσιμα στοιχεία της βιβλιοθήκης είναι δυνατόν. Τυπικά, αφού έχετε αναπτύξει αυτό το πρώτο - σχέδιο λύσης, μπορείτε να δείτε που η λύση έχει αδυναμίες και στη συνέχεια να διορθώσετε τις εν λόγω αδυναμίες με το δικό σας κώδικα (πιο γνωστό ως "επίλυση προβλήματος που στην πραγματικότητα έχουν, όχι αυτό που φαντάζονται "). Μπορείτε έπειτα να χρησιμοποιήσετε τη λύσης σας ως σημείο αναφοράς για την αξιολόγηση των βελτιώσεων που έχετε κάνει. Από εκείνο το σημείο, ανεξάρτητα από τις αδυναμίες που εξακολουθούν να μπορούν να αντιμετωπισθούν με στο πλαίσιο του ευρύτερου συστήματος στο οποίο η λύση στο πρόβλημά σας είναι ενσωματωμένη.

2.5. Βιβλιογραφία Κεφαλαίου

[1]Learning OpenCV, Gary Bradski, Andrian Kaebler, O’ Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, ISBN: 978-0-596-51613-0, 2008

3.FPGA

3.1. Επισκόπηση Κεφαλαίου

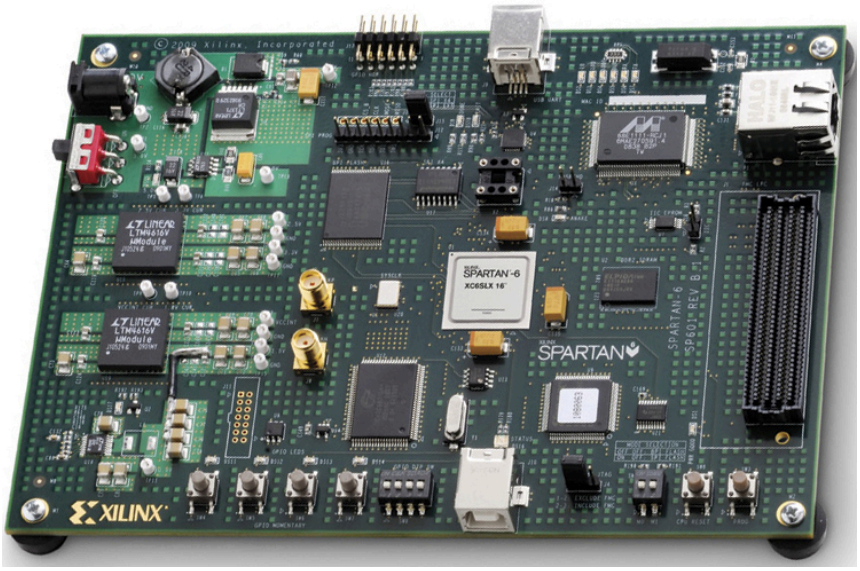
Ένα Πεδίο Προγραμματιζόμενων Συστοιχιών Πυλών (FPGA) είναι ένα ολοκληρωμένο κύκλωμα που έχει σχεδιαστεί για να ρυθμιστεί από τον πελάτη ή τον σχεδιαστή μετά την κατασκευή εξού και το "Πεδίο Προγραμματιζόμενων Πυλών". Η διαμόρφωση του FPGA γίνεται με τη χρήση μιας γλώσσας περιγραφής υλικού (HDL), παρόμοια με εκείνη που χρησιμοποιείται για την εφαρμογή ειδικών ολοκληρωμένων κυκλωμάτων (ASIC) (τα

διαγράμματα κυκλωμάτων προηγουμένως χρησιμοποιούνταν για να προσδιορίσουν τη διαμόρφωση, όπως ήταν για τα ASICs, αλλά αυτό είναι όλο και περισσότερο σπάνιο). Τα FPGAs μπορούν να χρησιμοποιηθούν για την εφαρμογή κάθε λογικής πράξης που ένα ASIC θα μπορούσε να εκτελέσει. Η δυνατότητα να ενημερώσετε τη λειτουργικότητα μετά την διανομή, μερική εκ νέου διαμόρφωση του σχεδιασμού [1] και τα χαμηλά μη επαναλαμβανόμενα έξοδα μηχανικού σε σχέση με ένα ASIC (παρά το γενικά υψηλότερο κόστος ανά μονάδα), είναι πλεονεκτήματα που προσφέρονται για πολλές εφαρμογές. [2]

Τα FPGAs περιέχουν προγραμματιζόμενα λογικά συστατικά που ονομάζεται “λογικά μπλοκ”, καθώς και ιεραρχημένες αναδιαρθρώσιμες διασυνδέσεις που επιτρέπουν στα μπλοκ να είναι "καλοδιομένα μαζί" σαν πολλές (μεταβλητές) λογικές πύλες που μπορούν να είναι σε (πολλές) διαφορετικές διαμορφώσεις. Το Λογικό μπλοκ μπορεί να ρυθμιστεί ώστε να εκτελεί πολύπλοκες συνδυαστικές λειτουργίες, ή απλώς απλές λογικές πύλες, όπως AND και XOR. Στα περισσότερα FPGAs, τα λογικά μπλοκ επίσης περιλαμβάνουν στοιχεία μνήμης, που μπορεί να είναι απλοί μανταλωτές ή πιο ολοκληρωμένα πεδία μνήμης [2].

Εκτός από τις ψηφιακές λειτουργίες, μερικά FPGAs έχουν αναλογικά χαρακτηριστικά. Το πιο συνηθισμένο αναλογικό χαρακτηριστικό είναι προγραμματιζόμενο ρυθμό απόκλισης και τη δύναμη κίνησης σε κάθε pin εξόδου, επιτρέποντας στο μηχανικό να ορίσει αργούς ρυθμούς σε pin με ελαφρύ φορτίο που αλλιώς θα μπορούσαν βουίζουν απαράδεκτα, και να θέσουν πιο ισχυρή, με πιο γρήγορο ρυθμό σε pin με μεγάλο φόρτο σε υψηλής ταχύτητας κανάλια που διαφορετικά θα τρέξουν πολύ αργά. [3] [4] Ένα άλλο σχετικά κοινό αναλογικό χαρακτηριστικό είναι διαφορικοί συγκριτές σε pin εισροής που έχουν σχεδιαστεί για να συνδέονται σε κανάλια διαφορικής σηματοδότησης. Μερικά « FPGAs μικτού σήματος" έχουν ολοκληρωμένους περιφερειακούς μετατροπείς αναλογικού σε ψηφιακό (ADC) και ψηφιακού σε αναλογικό (DAC) με αναλογικά μπλοκ επαναφοράς σήματος που τους επιτρέπει να λειτουργεί σαν ένα system-on-a-chip. [5] Τέτοιες συσκευές κάνουν ασαφή τη διαχωριστική γραμμή μεταξύ ενός FPGA, το οποίο μεταφέρει ψηφιακές μονάδες και μηδενικά μέσο των εσωτερικών προγραμματιζόμενων διασυνδέσεων, και το πεδίο προγραμματιζόμενων αναλογικών πινάκων (FPAA), η οποία μεταφέρει αναλογικές τιμές στις εσωτερικές προγραμματιζόμενες διασυνδέσεις.

Σχήμα 4. Μια Altera Stratix IV FPGA GX



Σχήμα 5. Ένα παράδειγμα ενός Xilinx Spartan 6 FPGA

3.2. Ιστορία

Ο κλάδος των FPGA φύτρωσε από την προγραμματιζόμενη μνήμη μόνο για ανάγνωση (PROM) και τις προγραμματιζόμενες λογικές διατάξεις (PLDs). Τα PROMs και τα PLDs είχαν τη δυνατότητα να είναι προγραμματίζονται σε παρτίδες σε ένα εργοστάσιο ή στο πεδίο (field programmable), ωστόσο η προγραμματιζόμενη λογική ήταν ενσωματωμένη μεταξύ των λογικών πυλών [6].

Στα τέλη της δεκαετίας του 1980 το Naval Surface Warfare Department χρηματοδότησε ένα πείραμα που προτάθηκε από τον Steve Casselman ώστε να αναπτύξει έναν υπολογιστή που επρόκειτο να εφαρμοστεί σε 600.000 επανα-προγραμματιζόμενες πύλες. Ο Casselman ήταν επιτυχής και ένα δίπλωμα ευρεσιτεχνίας που σχετίζονται με το σύστημα εκδόθηκε το 1992 [6].

Μερικές από τις θεμελιώδεις έννοιές της βιομηχανίας και των τεχνολογιών για τις προγραμματιζόμενες συστοιχίες λογικής, πύλες, και τα λογικά μπλοκ βασίζονται σε διπλώματα ευρεσιτεχνίας που χορηγήθηκαν στους David W. Page και Luverne R. Peterson το 1985. [7] [8]

Οι συνιδρυτές της Xilinx, Ross Freeman και Bernard Vonderschmitt, εφειυραν το πρώτο εμπορικά βιώσιμο πεδίο προγραμματιζόμενων συστοιχιών πύλών το 1985 -. το XC2064 [9]. Το XC2064 είχε προγραμματιζόμενες πύλες και προγραμματιζόμενες διασυνδέσεις μεταξύ των πυλών. Ηταν η απαρχή μιας νέας τεχνολογίας και αγοράς. [10] Το XC2064 καυχήθηκε μόλις 64 διαμορφώσιμα λογικά μπλοκ (CLBs), με δύο πίνακες αναζήτησης 3-εισροών (LUTs) [11]. Περισσότερα από 20 χρόνια αργότερα, ο Freeman έχει ενταχθεί στο National Inventors Hall of Fame για την εφεύρεσή του [12].

Η Xilinx συνέχισε χωρίς ανταγωνισμό και αναπτύχθηκε γρήγορα από το 1985 έως τα μέσα της δεκαετίας του 1990, όταν οι ανταγωνιστές πολλαπλασιάστηκαν διαβρώνοντας σημαντικά την μερίδια αγοράς. Μέχρι το 1993, η Actel εξυπηρετούσε περίπου το 18 τοις εκατό της αγοράς [10].

Η δεκαετία του 1990 ήταν ένα εκρηκτικό διάστημα για τα FPGAs, τόσο στην πολυπλοκότητα όσο και στον όγκο της παραγωγής. Στις αρχές της δεκαετίας του 1990, τα FPGAs κυρίως χρησιμοποιούνταν στις τηλεπικοινωνίες και τα δίκτυα. Μέχρι το τέλος της δεκαετίας, τα FPGAs βρήκαν το δρόμο τους σε καταναλωτές, στην αυτοκινητοβιομηχανία και σε βιομηχανικές εφαρμογές [13].

Τα FPGAs πήραν μια γεύση της φήμης τους το 1997, όταν ο Adrian Thompson, ένας ερευνητής που εργαζόταν στο Πανεπιστήμιο του Sussex, συγχώνευσε ένα αλγόριθμο γενετικής τεχνολογίας και FPGAs για να δημιουργήσει μια συσκευή αναγνώρισης ήχου. Ο αλγόριθμος Thomson είχε διαμορφωθεί σε μια σειρά από 10 x 10 κελιά σε ένα FPGA της Xilinx για κάνουν διακρίσεις ανάμεσα σε δύο τόνους, χρησιμοποιώντας τις αναλογικές

δυνατότητες από το ψηφιακό τσιπ. Η εφαρμογή των γενετικών αλγορίθμων για τη διαμόρφωση των συσκευών όπως τα FPGA τώρα αναφέρονται ως εξελίξιμο υλικό [6], [14]

3.3. Σύγχρονες εξελίξεις

Μια πρόσφατη τάση είναι να λαμβάνουν την χονδρόκοκκη αρχιτεκτονική προσέγγιση ένα βήμα παραπέρα, συνδυάζοντας το λογικό μπλοκ και την διασύνδεση των παραδοσιακών FPGAs με τους ενσωματωμένους μικροεπεξεργαστές και των σχετικών περιφερειακών τους για να σχηματίσουν ένα πλήρες «σύστημα σε ένα προγραμματιζόμενο τσιπ (system on a programmable chip)". Το έργο αυτό αντανακλά την αρχιτεκτονική του Ron Perlof και της Hana Potash του Burroughs Advanced Systems Group η οποία συνδύασε μια αναδιαρθρώσιμη αρχιτεκτονική CPU σε ένα μόνο τσιπ που ονομάζεται SB24. Αυτό το έργο έγινε το 1982. Παραδείγματα τέτοιων υβριδικών τεχνολογιών μπορούν να βρεθούν στο Xilinx Virtex-II PRO και Virtex-4, συσκευές οι οποίες περιλαμβάνουν έναν ή περισσότερους επεξεργαστές PowerPC που θα εντάσσονται στο πλαίσιο της λογικής του FPGA. Το Atmel FPSLIC είναι μια άλλη τέτοια συσκευή, που χρησιμοποιεί έναν AVR επεξεργαστή και σε συνδυασμό με τη λογική προγραμματισμού της αρχιτεκτονικής της Atmel. Το Actel SmartFusion ενσωματώσει μια αρχιτεκτονική ARM με επεξεργαστή Cortex-M3 (με έως και 512KB μνήμη flash και 64kB μνήμη RAM) και αναλογικά περιφερειακά όπως ένα πολυκαναλικό ADC και DACs που βασίζονται σε FPGA.

Μια εναλλακτική προσέγγιση για τη χρήση μακρο-επεξεργαστών είναι να κάνουν χρήση των επεξεργαστών που υλοποιούνται στα πλαίσια της λογικής FPGA.

Όπως αναφέρθηκε προηγουμένως, πολλά σύγχρονα FPGAs έχουν τη δυνατότητα να προγραμματισθούν εκ νέου σε "χρόνο εκτέλεσης," και αυτό είναι που οδηγεί στην ιδέα των αναδιαρθρώσιμων υπολογιστών ή αναπροσαρμόσιμων συστημάτων - κεντρικές μονάδες επεξεργασίας που οι ίδιοι θα αναμορφώσουν να ταιριάξει στο έργο. Ο εικονικός επεξεργαστής Mitrion από την Mitrionics είναι ένα παράδειγμα ενώς αναδιαρθρώσιμου επεξεργαστή, υλοποιημένου με FPGAs. Ωστόσο, δεν υποστηρίζει δυναμική αναδιάρθρωση κατά το χρόνο εκτέλεσης, αλλά αντίθετα προσαρμόζεται σε ένα συγκεκριμένο πρόγραμμα.

Επιπρόσθετα, οι νέες, μη FPGA αρχιτεκτονικές έχουν αρχίσει να εμφανίζονται. Με μικροεπεξεργαστές που έχουν δυνατότητα ρύθμισης λογισμικού, όπως η Stretch S5000 που

έχει υιοθετήσει μια υβριδική προσέγγιση, παρέχοντας μια σειρά από πυρήνες επεξεργαστή και FPGA-προγραμματιζόμενους πυρήνες στο ίδιο τσιπ.

Πύλες:

- 1987: 9.000 πύλες, Xilinx [10]
- 1992: 600.000, Naval Surface Warfare Department [6]
- Αρχές 2000: Εκατομμύρια [13]

Το μέγεθος της αγοράς:

- 1985: Πρώτη εμπορική τεχνολογία FPGA που εφευρέθηκε από την Xilinx [10]
- 1987: 14 εκατομμύρια δολάρια [10]
- ~ 1993:> 385 εκατομμύρια δολάρια [10]
- 2005: 1,9 δισεκατομμύρια δολάρια [15]
- 2010 οι εκτιμήσεις: 2,75 δισεκατομμύρια δολάρια [15]

Ο σχεδιασμός FPGA ξεκινά:

- 10.000 [16]
- 2005: 80.000 [17]
- 2008: 90.000 [18]

3.4. Συγκρίσεις FPGA

Ιστορικά, τα FPGAs έχουν πιο αργή, λιγότερο ενεργειακά απόδοση και επιτυγχάνονται γενικά λιγότερες λειτουργίες από ό, τι τα ASIC. Μια μελέτη έδειξε ότι τα σχέδια που υλοποιούνται σε FPGAs χρειάζονται κατά μέσο όρο 18 φορές περισσότερο χώρο, σχεδιάστηκαν με 7 φορές μεγαλύτερη δυναμική ενέργεια, και είναι 3 φορές πιο αργά από τις αντίστοιχες εφαρμογές ASIC. [2] Τα πλεονεκτήματα περιλαμβάνουν τη δυνατότητα να προγραμματιστούν εκ νέου στο πεδίο για να διορθωθούν σφάλματα, και μπορεί να περιλαμβάνει ένα μικρότερο χρόνο προς την αγορά και μείωση των μη επαναλαμβανόμενων εξόδων μηχανικού. Οι πωλητές μπορούν επίσης να πάρουν μια μέση οδό, με την ανάπτυξη του υλικού σε συνηθισμένα FPGAs, αλλά την τελική εκδοχή της κατασκευής τους να μην μπορεί πλέον να τροποποιηθεί αφότου ο σχεδιασμός έχει ολοκληρωθεί.

Η Xilinx ισχυρίζεται ότι πολλές αγορές και η δυναμική τεχνολογία αλλάζουν τα παραδείγματα ASIC / FPGA: [19]

- Οι δαπάνες ολοκληρωμένων κυκλωμάτων αυξάνονται εκθετικά
- Η πολυπλοκότητα των ASIC έχει επιμηκύνει το χρόνο ανάπτυξης
- Οι πόροι έρευνας και ανάπτυξης και ο αριθμός απασχολούμενων μειώνονται
- Η απώλεια φορολογικών εσόδων για τον αργό χρόνο διάθεσης στην αγορά αυξάνονται
- Οι δημοσιονομικοί περιορισμοί σε μια φτωχή οικονομία οδηγούν σε τεχνολογίες χαμηλού κόστους

Οι τάσεις αυτές κάνουν τα FPGAs μια καλύτερη εναλλακτική λύση από ό, τι τα ASICs για μεγαλύτερο αριθμό του μεγαλύτερου όγκου των αιτήσεων από ό, τι έχουν χρησιμοποιηθεί ιστορικά, στις οποίες η εταιρεία αποδίδει την αύξηση του αριθμού του σχεδιασμού των FPGA [19].

Μερικά FPGAs έχουν τη δυνατότητα της μερικής εκ νέου διαμόρφωση που επιτρέπει ένα μέρος της συσκευής να προγραμματίζεται εκ νέου, ενώ τα άλλα τμήματα θα συνεχίσουν να λειτουργούν.

Σχήμα 6. Μια Altera II FPGA, σε Altera teraSIC DE1.

3.4.1. Σύγκριση με πολύπλοκες προγραμματιζόμενες λογικές διατάξεις

Οι βασικές διαφορές μεταξύ CPLDs (Complex Programmable Logic Devices) και FPGAs είναι η αρχιτεκτονική. Ένα CPLD έχει μια κάπως περιοριστική δομή που αποτελείται από ένα ή περισσότερα προγραμματιζόμενα προϊόντα λογικών διατάξεων που τροφοδοτούν ένα

σχετικά μικρό αριθμό καταχωρητών με την βοήθεια ρολογιού. Το αποτέλεσμα αυτού είναι μικρότερη ευελιξία, με το πλεονέκτημα των πιο προβλέψιμων καθυστερήσεων στο χρονοδιάγραμμα και μία υψηλότερη λογική προς την αναλογία διασύνδεσης. Οι αρχιτεκτονικές FPGA, από την άλλη πλευρά, κυριαρχούνται από διασύνδεσης. Αυτό τα καθιστά πολύ πιο ευέλικτα (όσον αφορά τη σειρά των σχεδίων που είναι πρακτικά για την εφαρμογή στο εσωτερικό τους), αλλά και πολύ πιο σύνθετων στο σχεδιασμό για αυτά.

Μια άλλη αξιοσημείωτη διαφορά μεταξύ CPLDs και FPGAs, είναι η παρουσία στα περισσότερα FPGAs του υψηλότερου επιπέδου ενσωματωμένων λειτουργιών (όπως αθροιστές και πολλαπλασιαστές) και ενσωματωμένων μνημών, καθώς και να έχουν λογικά μπλοκ για την εφαρμογή αποκωδικοποιητών ή μαθηματικών λειτουργιών.

3.4.2. Σε θέματα ασφαλείας

Όσον αφορά την ασφάλεια, τα FPGAs έχουν και πλεονεκτήματα και μειονεκτήματα σε σύγκριση με τα ASICs ή ασφαλή μικροεπεξεργαστές. Η ευελιξία των FPGAs έχει χαμηλότερο κίνδυνο ως προς τις κακόβουλες αλλαγές κατά τη διάρκεια της κατασκευής. [20] Για πολλά FPGAs, το σχέδιο είναι εκτεθειμένα όταν φορτώνεται (συνήθως σε κάθε άνοιγμα). Για την αντιμετώπιση αυτού του ζητήματος, κάποια FPGAs υποστηρίζουν κρυπτογράφηση. [21] [22]

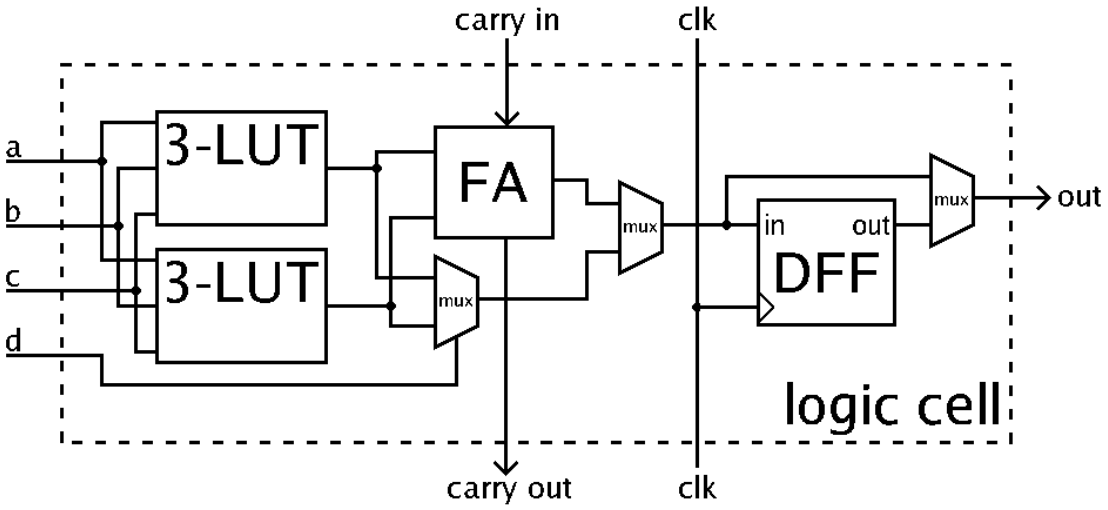
3.5. Αρχιτεκτονική

Η πιο κοινή αρχιτεκτονική των FPGA [25] αποτελείται από μια σειρά από λογικά μπλοκ (που ονομάζεται Configurable Logic Block, CLB, ή Logic Array Block, LAB, ανάλογα με τον προμηθευτή), I/O pads, και τους διαύλους δρομολόγησης. Σε γενικές γραμμές, όλοι οι διαύλοι δρομολόγησης έχουν το ίδιο πλάτος (αριθμός καλωδίων). Πολλαπλα I/O pads μπορούν να εντάσσονται στο ύψος μίας γραμμής ή το πλάτος μιας στήλης του πίνακα.

Ένα κύκλωμα εφαρμογής πρέπει να αντιστοιχίζεται σε ένα FPGA με επαρκείς πόρους. Ενώ ο αριθμός των CLBs/LABs και I / Os που απαιτούνται είναι εύκολο να προσδιοριστούν από το σχεδιασμό, ο αριθμός των κομματιών δρομολόγησης που απαιτούνται μπορεί να ποικίλει σημαντικά, ακόμη και μεταξύ σχεδίων με το ίδιο ποσοστό λογικής. Για παράδειγμα, επειδή ένας διακόπτης απαιτεί πολύ περισσότερες δρομολόγησης από μια συστολική συστοιχία με τον ίδιο αριθμό θυρών. Επειδή αχρησιμοποίητα κομμάτια δρομολόγησης αυξάνουν το κόστος (και μειώνουν την απόδοση) του μέρους που δεν παρέχει κανένα όφελος, οι κατασκευαστές

FPGA προσπαθούν να παρέχουν ακριβός όσα κομμάτια, έτσι ώστε τα περισσότερα σχέδια που θα ταιριάζει από την άποψη των LUTs και IOs που μπορούν να δρομολογηθούν. Αυτό καθορίζεται από τις εκτιμήσεις όπως αυτές που προέρχονται από πειράματα με τα υπάρχοντα σχέδια.

Σε γενικές γραμμές, ένα λογικό μπλοκ (CLB ή LAB) αποτελείται από λίγα λογικά κύτταρα (που λέγονται ALM, LE, φέτα κλπ). Ένα τυπικό κύτταρο αποτελείται από ένα πίνακα αναζήτησης 4-εισόδων (LUT), ένα πλήρη αθροιστή (FA) και ένα D flip-flop, όπως φαίνεται παρακάτω. Τα LUTs σε αυτό το σχήμα χωρίζονται σε δύο σχήματα 3-εισροών LUTs. Σε κανονική λειτουργία αυτά συνδυάζονται σε ένα 4 εισόδων LUT μέσω του αριστερού MUX. Στην αριθμητική λειτουργία, οι εκροές τους τρέφουν το FA. Η επιλογή του τρόπου γίνεται στο μεσαίο MUX. Οι έξοδοι μπορούν να είναι είτε συγχρονισμένες ή ασύγχρονες, ανάλογα με τον προγραμματισμό του mux προς τα δεξιά, στο σχηματικο παράδειγμα. Στην πράξη, το σύνολο ή ένα μέρος των FA τίθενται ως συναρτήσεις στα LUTs, προκειμένου να εξοικονομηθεί χώρος. [26] [27][28]



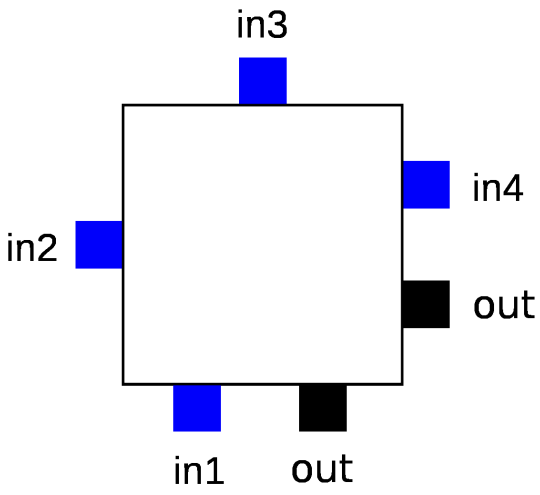
Σχήμα 7. Παράδειγμα απλοποιημένης απεικόνισης ενός λογικού κελιού

Τα ALMS και οι Φέτες περιέχουν συνήθως 2 ή 4 παρόμοιες δομές με το παράδειγμα, με κάποια κοινά σήματα. Τα CLBs / LABS περιέχουν συνήθως μερικά ALMS / LEs / Φέτες.

Τα τελευταία χρόνια, οι κατασκευαστές έχουν αρχίσει να κινούνται προς LUTs 6-εισροών στα πολύ υψηλής ποιότητας εξαρτήματά τους, υποστηρίζοντας αυξημένη απόδοση [29].

Επειδή τα σήματα ρολογιού (και συχνά άλλα υψηλά fanout σήματα) κανονικά δρομολογούνται μέσω αφιερωμένης δρομολόγησης ειδικού σκοπού στα εμπορικά FPGAs, αυτά και τα άλλα σήματα έχουν χωριστή διαχείριση.

Για αυτό το παράδειγμα αρχιτεκτονικής, οι θέσεις των pin του λογικού μπλοκ του FPGA παρουσιάζονται παρακάτω.



Σχήμα 8. Τοποθεσίες Pin Λογικού Μπλοκ

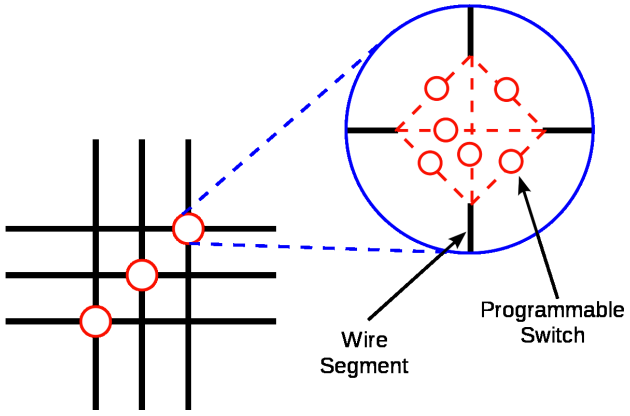
Κάθε είσοδος είναι προσβάσιμη από τη μια πλευρά του λογικού μπλοκ, ενώ το pin έξοδου μπορεί να συνδεθεί με τα καλώδια δρομολόγησης, τόσο το κανάλι στα δεξιά όσο και το κανάλι κάτω από το λογικό μπλοκ.

Κάθε pin έξοδου του λογικού μπλοκ μπορεί να συνδεθεί με οποιοδήποτε από τα τμήματα καλωδίωσης στα κανάλια πλησίον του.

Ομοίως, ένα I / O pad μπορεί να συνδεθεί με οποιοδήποτε από τα τμήματα καλωδίωσης στο κανάλι δίπλα σε αυτό. Για παράδειγμα, ένα I / O pad στην κορυφή του chip μπορεί να συνδεθεί σε κάποιο από τα καλώδια W (όπου W είναι το πλάτος καναλιού) στο οριζόντιο κανάλι αμέσως κάτω από αυτό.

Σε γενικές γραμμές, η δρομολόγηση του FPGA είναι unsegmented. Δηλαδή, κάθε τμήμα καλωδίωσης καλύπτει μόνο ένα τετράγωνο από το λογικό μπλοκ, πριν καταλήξει σε ένα διακόπτη. Με το άνοιγμα μερικών από τους προγραμματιζόμενους διακόπτες μέσα σε ένα κουτί διακόπτη, έτσι μεγαλύτερες διαδρομές μπορούν να κατασκευαστούν. Για υψηλότερες ταχύτητες διασύνδεσης, μερικές αρχιτεκτονικές FPGA χρήση δρομολόγησης με επιπλέον γραμμές που εκτείνονται σε πολλαπλά λογικά μπλοκ.

Κάθε φορά που ένα κάθετο και ένα οριζόντιο κανάλι τέμνονται, υπάρχει ένα Switch box. Σε αυτή την αρχιτεκτονική, όταν ένα σύρμα εισέρχεται σε ένα κουτί διακόπτη, υπάρχουν τρεις προγραμματιζόμενοι διακόπτες που του επιτρέπουν να συνδεθεί με άλλα τρία καλώδια σε παρακείμενα τμήματα του καναλιού. Το μοτίβο, ή η τοπολογία, των διακοπτών που χρησιμοποιούνται σε αυτή την αρχιτεκτονική είναι η επίπεδη ή βασισμένη τοπολογία τομέα με Switch box. Σε αυτή την τοπολογία το κουτί διακόπτη, τα καλώδια με ένα αριθμό του κομματιού συνδέονται μόνο με καλώδια με έναν αριθμό κομματιού σε παρακείμενα τμήματα καναλιού, καλώδια, με αριθμό 2 του κομματιού θα συνδεθεί μόνο με άλλα καλώδια, με αριθμό τροχιάς 2 και ούτω καθεξής. Το σχήμα που ακολουθεί παρουσιάζει τις συνδέσεις σε ένα Switch box.



Σχήμα 9. Τοπολογία Switch box

Σύγχρονες οικογένειες FPGA διερευνούν τις παραπάνω δυνατότητες να περιλαμβάνουν υψηλότερα επίπεδα λειτουργικότητας που καθορίζονται στο πυρίτιο. Έχοντας αυτές τις κοινές λειτουργίες ενσωματωμένες σε πυρίτιο μειώνετε η απαιτούμενη περιοχή και δίνει σε αυτές τις λειτουργίες αυξημένη ταχύτητα. Παραδείγματα αυτών των πολλαπλασιαστών είναι τα DSP, ενσωματωμένοι επεξεργαστές, IO υψηλής ταχύτητας και ενσωματωμένες μνήμες.

Τα FPGAs επίσης χρησιμοποιούνται ευρέως στα συστήματα επικύρωσης, προ-πυριτίου επικύρωσης πριν, μετά-πυριτίου επικύρωσης, καθώς και στο firmware ανάπτυξης. Αυτό επιτρέπει σε εταιρείες υψηλής κεφαλαιοποίησης την επικύρωση του σχεδιασμού τους πριν το τσιπ παραχθεί στο εργοστάσιο, μειώνοντας έτσι το χρόνο διάθεσης στην αγορά.

3.5.1. Σχεδιασμός και προγραμματισμός FPGA

Για να ορίσει τη συμπεριφορά του FPGA, ο χρήστης χρησιμοποιεί μια γλώσσα περιγραφής υλικού (HDL) ή ένα σχηματικό σχεδιασμό. Το έντυπο της HDL είναι πιο κατάλληλο για την

εργασία με μεγάλες δομές, επειδή είναι δυνατόν να προσδιοριστούν μόνο αριθμητικά, αντί να χρειάζεται να καταρτιστεί κάθε κομμάτι με το χέρι. Ωστόσο, η σχηματική είσοδος μπορεί να επιτρέψει την ευκολότερη απεικόνιση ενός σχεδίου ή υποδείγματος.

Στη συνέχεια, χρησιμοποιώντας ένα ηλεκτρονικό εργαλείο γίνετε αυτοματισμός του σχεδιασμού, και δημιουργείται μια τεχνολογία αντίστοιχη της netlist. Η netlist μπορεί στη συνέχεια να τοποθετηθεί στην πραγματική αρχιτεκτονική FPGA χρησιμοποιώντας μια διαδικασία που ονομάζεται τοποθέτησε και δρομολόγησε, και συνήθως εκτελείται από τις διαδρομές και το λογισμικό της εταιρείας του FPGA . Ο χρήστης θα επικυρώσει τον χάρτη, τον τόπο και τα αποτελέσματα διαδρομής μέσω ανάλυσης χρονισμού, προσομοίωση, και άλλων μεθόδων επαλήθευσης. Αφότου η επικύρωση και η διαδικασία σχεδιασμού είναι πλήρης, το δυαδικό αρχείο που παράγεται (και μέσω της χρήσης του ιδιόκτητου λογισμικού της εταιρείας του FPGA) χρησιμοποιείται για να (επαν) ρυθμίσετε το FPGA.

Πηγαίνοντας από το σχηματικό / HDL αρχείο προέλευσης στην πραγματική διαμόρφωση: Τα αρχεία πηγής τροφοδοτούνται σε μια σουίτα λογισμικού του FPGA / CPLD του προμηθευτή που μέσα από διάφορα στάδια, θα παράγει ένα αρχείο. Αυτό το αρχείο στη συνέχεια μεταφέρεται στο FPGA / CPLD μέσω μιας σειριακής θύρας (JTAG) ή σε μια εξωτερική συσκευή μνήμης σαν την EEPROM.

Οι πιο συνηθισμένες HDLs είναι η VHDL και η Verilog, αν και σε μια προσπάθεια να μειωθεί η πολυπλοκότητα του σχεδιασμού σε HDLs, οι οποίες έχουν σχέση με το αντίστοιχο των γλωσσών συναρμολόγησης, υπάρχουν κινήσεις για αύξηση το επίπεδο αφαίρεσης, μέσω της εισαγωγής εναλλακτικών γλωσσών. Το LabVIEW της National Instrument's είναι μία γλώσσα γραφικόού προγραμματισμού (μερικές φορές αναφέρεται ως "G"), έχει μια πρόσθετη μονάδα στο FPGA που αναγνωρίζει και προγραμματίζει το FPGA.

Για να απλοποιηθεί ο σχεδιασμός των πολύπλοκων συστημάτων σε FPGAs, υπάρχουν βιβλιοθήκες προκαθορισμένων πολύπλοκων λειτουργιών και κυκλώματα που έχουν δοκιμαστεί και βελτιστοποιηθεί για να επιταχυνθεί η διαδικασία σχεδιασμού. Αυτά τα προκαθορισμένα κυκλώματα είναι κοινώς ονομαζόμενα ως πυρήνες IP και είναι διαθέσιμα από προμηθευτές FPGA και τρίτους προμηθευτές IP (σπάνια ελεύθερα, και τυπικά, διατίθενται βάσει ιδιόκτητων αδειών). Άλλα προκαθορισμένα κυκλώματα είναι διαθέσιμα από τις κοινότητες προγραμματιστών, όπως OpenCores (κατά κανόνα, που διατίθεται με

βάσει άδειες ελεύθερου και ανοιχτού κώδικα, όπως η GPL, BSD ή παρόμοιες άδειες), και άλλες πηγές.

Σε μια τυπική ροή σχεδιασμού, η ανάπτυξη εφαρμογών για FPGA θα προσομοιώνουν το σχεδιασμό σε πολλαπλά στάδια σε όλη τη διαδικασία του σχεδιασμού. Αρχικά η RTL περιγραφή σε VHDL ή Verilog προσομοιώνεται με τη δημιουργία δοκιμών για την προσομοίωση του συστήματος και για να παρατηρήσετε τα αποτελέσματα. Στη συνέχεια, αφού η σύνθεση έχει χαρτογραφήσει το σχεδιασμό σε ένα netlist, το netlist, μεταφράζεται σε μία περιγραφή επιπέδου πύλης όπου προσομοίωση επαναλαμβάνεται για να επιβεβαιώσετε ότι η σύνθεση ολοκληρώθηκε χωρίς λάθη. Τέλος, η σχεδίαση τοποθετήτε στο FPGA και σε αυτό το σημείο καθυστερήσεις διάδοσης μπορούν να προστεθούν και να τρέξει ξανά η προσομοίωση και πάλι με τις τιμές αυτές επάνω στο netlist.

3.5.2. Βασικά είδη τεχνολογίας επεξεργασίας

- SRAM - με βάση τις στατικές τεχνολογίες μνήμης. Εσωτερικός προγραμματιζόμενο σύστημα και εκ νέου προγραμματιζόμενα συστήματα. Απαιτεί εξωτερικές συσκευές εκκίνησης. CMOS.
- Antifuse - Μία φορά προγραμματιζόμενο. CMOS.
- PROM - Προγραμματιζόμενη Μόνο για ανάγνωση τεχνολογία μνήμης. Μία φορά προγραμματιζόμενο λόγω των πλαστικών συσκευασιών.
- EPROM - αποσβέσιμες και προγραμματιζόμενες μόνο για ανάγνωση τεχνολογίες μνήμης. Μία φορά προγραμματιζόμενη αλλά με παράθυρο, που μπορεί να διαγραφεί με υπεριώδη ακτινοβολία(UV). CMOS
- EEPROM - Ηλεκτρικά αποσβέσιμες και προγραμματιζόμενες μνήμες μόνο για ανάγνωση. Μπορούν να διαγραφούν, ακόμα και σε πλαστικές συσκευασίες. Μερικές EEPROM όμως μπορούν να προγραμματιστουν σε συσκευές εσωτερικά. CMOS.
- Flash - Flash-erase τεχνολογία EPROM. Μπορούν να διαγραφούν, ακόμα και σε πλαστικές συσκευασίες. Μερικές όμως όχι όλες οι συσκευές flash μπορούν να προγραμματιστεούν σε σύστημα. Συνήθως, ένα κελί φλας είναι μικρότερο από το αντίστοιχο κελί EEPROM και είναι συνεπώς λιγότερο ακριβό να κατασκευαστούν. CMOS.
- Fuse - Μία φορά προγραμματιζόμενο. Διπολική.

3.6. Εφαρμογές

Οι εφαρμογές των FPGAs περιλαμβάνουν ψηφιακή επεξεργασία σήματος, που στηρίζονται σε λογισμικό ραδιοφώνου, της αεροδιαστημικής και άμυνας, προτυποποίηση ASIC, την ιατρική απεικόνιση, τη μηχανική όραση, την αναγνώριση φωνής, κρυπτογραφία, βιοπληροφορική, ο προσομοίωσης υλικού υπολογιστή, τη ραδιοαστρονομία, την ανίχνευση μετάλλων και άλλους τομείς .

Τα FPGAs αρχικά ξεκίνησαν ως ανταγωνιστές των CPLDs και διαγωνίστηκαν σε ένα παρόμοιο χώρο. Καθώς το μέγεθός τους, οι δυνατότητες, και η ταχύτητα αυξάνεται, άρχισαν να αναλαμβάνουν μεγαλύτερες και μεγαλύτερες λειτουργίες και μερικα είναι τώρα στην αγορά ως πλήρη συστήματα σε τσιπ (SoC). Ιδιαίτερα με την εισαγωγή ειδικών πολλαπλασιαστών σε αρχιτεκτονικές FPGA στα τέλη του 1990, οι εφαρμογές που είχαν παραδοσιακά αποτελούσαν το μοναδικό αποθεματικό και τα DSPs άρχισαν να ενσωματώνονται στα FPGAs αντ 'αυτού. [23] [24]

Τα FPGAs βρίσκουν εφαρμογές σε οποιαδήποτε περιοχή ή αλγόριθμο που μπορεί να κάνει χρήση του μαζικού παραλληλισμού που προσφέρει η αρχιτεκτονική τους. Ένας τέτοιος τομέας είναι το σπάσιμο κωδικών, ιδιαίτερα σε δυναμικές επιθέσεις, των κρυπτογραφικών αλγορίθμων.

Τα FPGAs χρησιμοποιούνται όλο και περισσότερο στης συμβατικές εφαρμογές υπολογιστικής υψηλών επιδόσεων, όπου πυρήνες όπως FFT ή συνέλιξη είναι υπολογιστικά εκτελούνται στο FPGA αντί σε ένα μικροεπεξεργαστή.

Ο εγγενής παραλληλισμός των λογικών πόρων σε ένα FPGA επιτρέπει σημαντική υπολογιστική απόδοση ακόμα και σε χαμηλές ταχύτητες ρολογιού. Η ευελιξία του FPGA επιτρέπει ακόμα υψηλότερες επιδόσεις ανταλλάσσοντας ένα κομμάτι από την ακρίβεια και το εύρος στη μορφή αριθμού για την αύξηση του αριθμού των παράλληλων αριθμητικών μονάδων. Αυτό είχε ως αποτέλεσμα ένα νέο είδος της επεξεργασίας που ονομάζεται επανα-προγραμματιζόμενη πληροφορική, όπου χρονοβόρα tasks εξέρχονται από το λογισμικό σε FPGAs.

Η έγκριση των FPGAs σε υπολογιστές υψηλών επιδόσεων περιορίζεται σήμερα με την πολυπλοκότητα του σχεδιασμού των FPGA σε σύγκριση με τα συμβατικά λογισμικά και την αλλαγή του χρόνου απόπλου των σημερινών εργαλείων σχεδιασμού.

Παραδοσιακά, τα FPGAs έχουν κρατηθεί για συγκεκριμένες κάθετες εφαρμογές όπου ο όγκος της παραγωγής είναι μικρό. Γι 'αυτές της χαμηλής μάζας εφαρμογές, το τίμημα που πληρώνουν οι εταιρείες για το κόστος του υλικού ανά μονάδα για ένα προγραμματιζόμενο τσιπ είναι πιο προσιτό από το αναπτυξιακό κομμάτι που δαπανάτε για τη δημιουργία ενός ASIC για μια χαμηλού όγκου εφαρμογή. Σήμερα, τα νέα κόστη και η δυναμική των επιδόσεων έχουν διευρύνει το φάσμα των βιώσιμων εφαρμογών.

3.7. Μεγαλύτεροοι κατασκευαστές

Η Xilinx και η Altera είναι οι τρέχουσες ηγέτες της αγοράς των FPGA και πολύ καιρό ανταγωνιστές της βιομηχανίας. Μαζί, ελέγχουν άνω του 80 τοις εκατό της αγοράς [30], με την Xilinx από μόνη της αντιπροσωπεύει πάνω από το 50 τοις εκατό.

Τόσο η Xilinx και η Altera παρέχουν δωρεάν λογισμικό σχεδιασμού για Windows και Linux. [31] [32]

Άλλοι ανταγωνιστές περιλαμβάνουν Lattice Semiconductor (SRAM βάση με ενσωματωμένη φλας διαμόρφωσης, instant-on, χαμηλής ισχύος, ζωντανά αναδιάρθρωσης), Actel (antifuse, flash-based, μικτού σήματος), SiliconBlue Technologies (εξαιρετικά χαμηλή ισχύς SRAM FPGAs με δυνατότητα ολοκληρωμένης αμετάβλητης μνήμης διαμόρφωσης), Achronix [33] (RAM, 1,5 GHz ταχύτητα) που θα διευρύνουν τις μάρκες τους στο διαγωνισμό Intels' state-of-the art 22nm [34], και QuickLogic .

Τον Μάρτιο του 2010, η Tabula [35] ανακοίνωσε το FPGA νέας τεχνολογίας, που χρησιμοποιεί το χρόνο λογικής πολυπλεξίας και διασύνδεσης για μεγαλύτερες δυνατότητες μείωσης του κόστους για υψηλής πυκνότητας εφαρμογές.

3.8. Βιβλιογραφία Κεφαλαίου

[1] Wiśniewski, Remigiusz (2009). Synthesis of compositional microprogram control units for programmable devices. Zielona GoÅLra: University of Zielona GoÅLra. pp. 153. ISBN 978-83-7481-293-1.

[2]edu/~vaughn/challenge/fpga_arch.html FPGA Architecture for the Challenge]

[3] FPGA Signal Integrity tutorial (http:/ / wiki. altium. com/ display/ ADOH/ FPGA+ SI+ Tutorial+ -+ Simulating+ the+ Reflection+Characteristics)

[4]NASA: FPGA drive strength (http:/ / klabs. org/ richcontent/ fpga_content/ DesignNotes/ signal_quality/ actel_drive_strength/ index. htm)

[5]Mike Thompson. "Mixed-signal FPGAs provide GREEN POWER" (http:/ / www. eetimes. com/ showArticle. jhtml?articleID=200000777). EE Times, 2007-07-02.

[6] History of FPGAs (http:/ / filebox. vt. edu/ users/ tmagin/ history. htm)

[7] Google Patent Search, " Re-programmable PLA (http:/ / www. google. com/ patents?id=BB4vAAAAEBAJ& dq=4508977)". Retrieved February 5, 2009.

[8]Google Patent Search, " Dynamic data re-programmable PLA (http:/ / www. google. com/ patents?id=1-gzAAAAEBAJ& dq=4524430)". Retrieved February 5, 2009.

[9]Peter Clarke, EE Times, " Xilinx, ASIC Vendors Talk Licensing (http:/ / www. eetimes. com/ story/ OEG20010622S0091)." June 22, 2001. Retrieved February 10, 2009.

[10] Funding Universe. “ Xilinx, Inc. (http:/ / www. fundinguniverse. com/ company-histories/ Xilinx-Inc-Company-History. html)” Retrieved January 15, 2009.

[11] Clive Maxfield, Programmable Logic DesignLine, " Xilinx unveil revolutionary 65nm FPGA architecture: the Virtex-5 family (http://www. pldesignline. com/ products/ 187203173). May 15, 2006. Retrieved February 5, 2009.

[12] Press Release, " Xilinx Co-Founder Ross Freeman Honored as 2009 National Inventors Hall of Fame Inductee for Invention of FPGA (http:// press. xilinx. com/ phoenix. zhtml?c=212763& p=irol-newsArticle& ID=1255523& highlight)"

[13]Clive Maxfield, book, " The Design Warrior's Guide to FPGAs (http:/ / books. google. com/ books?id=dnuwr2xOFpUC& pg=PA4&lpg=PA4& dq=FPGA+ Market+ growth+ 1990s& source=web& ots=YjFedB35Vp& sig=EH8y56Cih9iNLEqYXkZ63iO46K4& hl=en&sa=X& oi=book_result& resnum=4& ct=result)".Published by Elsevier, 2004. ISBN 0750676043, 9780750676045. Retrieved February 5, 2009

[14] "Original Paper on Work, Thompson" (http:/ / www. informatics. sussex. ac. uk/ users/ adrianth/ ices96/ paper. ps). .

[15]Dylan McGrath, EE Times, " FPGA Market to Pass $2.7 Billion by '10, In-Stat Says (http:/ / www. eetimes. com/ news/ design/ business/showArticle. jhtml?articleID=188102617)". May 24, 2006. Retrieved February 5, 2009.

[16]Narinder Lall eASIC Corporation, " FPGA Judgment Day:Rise of Second Generation Structured ASICs (http:/ / www.opensystems-publishing. com/ e-letter/ dsp/ 2008/ 03/ easic. pdf). March, 2008. Retrieved February 5, 2009.

[17]Dylan McGrath, EE Times, " Gartner Dataquest Analyst Gives ASIC, FPGA Markets Clean Bill of Health (http:/ / www. eetimes. com/conf/ dac/ showArticle. jhtml?articleID=164302400)". June 13, 2005. Retrieved February 5, 2009.

[18]Virtex-4 Family Overview (http:/ / www. xilinx. com/ support/ documentation/ data_sheets/ ds112. pdf)

[19]Tim Erjavec, White Paper, " Introducing the Xilinx Targeted Design Platform: Fulfilling the Programmable Imperative (http:/ / www. xilinx.com/ publications/ prod_mktg/ Targeted_Design_Platforms. pdf)." February 2, 2009. Retrieved February 2, 2009

[20] Huffmire Paper " Managing Security in FPGA-Based Embedded Systems (http:/ / www2. computer. org/ portal/ web/ csdl/ doi/ 10. 1109/MDT. 2008. 166)." Nov-Dec 2008. Retrieved Sept 22, 2009

[21] "Virtex-5 FPGA Configuration User Guide" (http:/ / www. xilinx. com/ support/ documentation/ user_guides/ ug191. pdf). Xilinx Inc.. August 2010. pp. 33–35. . Retrieved 2010-10-31.

[22]"Protecting Intellectual Property Through FPGA Design Security" (http:/ / www. altera. com/ literature/ ads/ fpgadesignsecurity. pdf). Altera Corporation. . Retrieved 2010-11-01.

[23]FPGA/DSP Blend Tackles Telecom Apps (http:/ / www. bdti. com/ articles/ info_eet0207fpga. htm)

[24]Xilinx aims 65-nm FPGAs at DSP applications (http:/ / www. eetimes. com/ showArticle. jhtml?articleID=197001881)

[25] http:/ / www. eecg. toronto. edu/ ~vaughn/ challenge/ fpga_arch. html

[26]http:/ / www. altera. com/ literature/ hb/ cyc2/ cyc2_cii51002. pdf

[27] http:/ / www. altera. com/ literature/ hb/ stratix-iv/ stx4_5v1_01. pdf

[28] http:/ / www. xilinx. com/ support/ documentation/ user_guides/ ug070. pdf

[29] http:/ / www. xilinx. com/ bvdocs/ whitepapers/ wp245. pdf

[30]Seeking Alpha, " Altera and Xilinx Report: The Battle Continues (http:/ / seekingalpha. com/ article/85478-altera-and-xilinx-report-the-battle-continues)". July 17, 2008. Retrieved February 5, 2009.

[31] "Xilinx ISE WebPACK" (http:/ / www. xilinx. com/ ise/ logic_design_prod/ webpack. htm). .

[32]"Quartus II Web edition software" (https:/ / www. altera. com/ support/ software/ download/ altera_design/ quartus_we/ dnl-quartus_we.jsp). .

[33] http:/ / www. achronix. com/

[34] "Achronix to use Intel's 22 nm manufacturing" (http:/ / newsroom. intel. com/ community/ intel_newsroom/ blog/ 2010/ 11/ 01/chip-shot-achronix-to-use-intel-s-22nm-manufacturing). .

[35]http:/ / www. tabula. com/

4.Face detection

4.1. Επισκόπηση Κεφαλαίου

Σήμερα υπάρχουν πολλοί ακριβείς και αξιόπιστοι αλγόριθμοι υπολογιστικής όρασης. Όταν τα πρόσωπα αυτά σε συνδυασμό με την τεχνολογία των ενσωματωμένων συστημάτων, μπορούν να επιτευχθούν εύρωστες πραγματικού χρόνου και χαμηλού κόστους υλοποιήσεις.

Σε εφαρμογές λογισμικού, η βιβλιοθήκη η οποία έχει το μεγαλύτερο αριθμό χρηστών είναι η OpenCV. Είναι μία ανοιχτού κώδικα βιβλιοθήκη υπολογιστικής όρασης οι λειτουργίες της οποίας στοχεύουν σε Real-Time επεξεργασία εικόνας / βίντεο, που εγκαινιάστηκε το 1999 από την Intel. Οι συναρτήσεις ανίχνευσης προσώπου χρήση ανίχνευση χαρακτηριστικών της OpenCV με τη μεθοδολογία Haar, βασίζονται σε δημοσιευμένες εργασίες που χρησιμοποίησαν στο [1] οι Viola Jones αντιμετωπίζουν τους αλγόριθμους ανίχνευσης που χρησιμοποιούν ταξινομητές, μία ολόκληρη εικόνα για την αποφυγή υπολογισμών

πλεονασματικών ποσών και καταρράκτη σταδίων προκειμένου να παρακάμψει περιττές αναλύσεις.

Οι Vanderlei Bonato et al. [2], χρησιμοποίησαν μία μετατροπή RGB σε HSI καθώς και από το φιλτράρισμα της εικόνας και τη μείωση του μεγέθους της, πέτυχαν ποσοστό επιτυχίας 99,57%. Οι Hong Neoh Sham et al. [3] πρότειναν ένα έξυπνο αλγόριθμο ανίχνευσης ακμών και τον συνέκριναν με τον λιγότερο ακριβή Prewitt. Οι Chun He et al. [4], χρησιμοποίησαν Haar-like αλγόριθμο με AdaBoost που είχε τρία στάδια. Το στάδιο προ-επεξεργασίας όπου η εικόνα γίνετε downscaled, το στάδιο της ανίχνευσης, όπου βρίσκονται τα χαρακτηριστικά και γίνονται οι κλιμακώσεις, και το τελικό στάδιο επεξεργασίας όπου αφαιρούνται ο θόρυβος και οι επικαλυπτόμενες ανιχνεύσεις. Οι Hossein Sahoolizadeh et al. [5], στην έρευνά τους χρησιμοποίησαν ένα Haar -based αλγόριθμο που προέκυπτε από 3-Dimensional κυματίδια μεταμόρφωσης.

Στην δική τους μελέτη οι Stavros Paschalakis et al. [6] πρότειναν ένα αλγόριθμο τριών σταδίων στον οποίο πρώτα υπό-δειγματιζόταν η εικόνα, τότε χρησιμοποιούταν ένα φιλτράρισμα δέρματος και, τέλος, γινόταν ανίχνευση προσώπων. Οι Mohamed S. Sadri et al. [7] περιγράφουν στην εργασία τους ένα σύστημα πέντε σταδίων στο οποίο πρώτα χρησιμοποιείται ένα φίλτρο για το χρώμα δέρματος που παραλείπονται όλα τα μέρη μιας εικόνας εκτός από εκείνα με το χρώμα του δέρματος. Μετά από αυτό ένα φίλτρο ανίχνευσης ακμών εφαρμόστηκε, έτσι ώστε κάθε pixel είχε μετατραπεί σε ένα με τιμή 1 ή 0. Ύστερα γίνετε περιστροφή και υπό-διαμόρφωση και ακολούθως MLP κατά την οποία ανιχνεύεται αν τυχόν σε ένα υποπλαίσιο περιέχονται οποιαδήποτε πρόσωπα. Τέλος, ένα σύστημα απόφασης συγχωνεύετε και αναλύει τα αποτελέσματα της MLP. Οι Maissa Ali et al. [8] κατά την εφαρμογή τους χρησιμοποίησαν παρεμβολή Bayer με ξεχωριστά ιστογράμματα RGB και ένα ιστόγραμμα 3-διαστάσεων.

Ο αλγόριθμος της OpevCV βασίζεται στο πρωτότυπο συστήματος για τον εντοπισμό προσώπων των Viola και Jones [9]. Μια εικόνα σε επεξεργασία προκειμένου να υπολογιστούν γρήγορα τα χαρακτηριστικά Haar που έχουν επιλεγεί έτσι ώστε μια γρήγορη αναγνώριση προσώπου να επιτευχθεί. Σύμφωνα με τους Jungok et al. [10], τα Haar-like χαρακτηριστικά είναι όλα βασισμένα σε δύο ή τρία τετράγωνα.Ένας αλγόριθμος AdaBoost τότε μαθαίνει, στη συνέχεια, το βάρος και το μέγεθος κάθε χαρακτηριστικού. Κάθε ένα χαρακτηριστικό Haar έχει μια αξία που υπολογίζεται. Η περιοχή που καλύπτεται από καθένα

από τα ορθογώνια μπορεί εύκολα να υπολογιστεί χρησιμοποιώντας την εσωτερική εικόνα που είναι το άθροισμα των pixel από την αρχική εικόνα. Τέλος χρησιμοποιείται cascading για να απορρίψει γρήγορα υποψήφια πρόσωπα. Σε κάθε στάδιο cascading, ο αλγόριθμος χρησιμοποιεί αυστηρότερες απαιτήσεις. Σε περίπτωση που το υποψήφιο πρόσωπο αποτύχει να περάσει ένα στάδιο απορρίπτεται. Μόνο εάν το υποψήφιο πρόσωπο περάσει όλα τα στάδια ανιχνεύεται.

4.2. Software based

Σε υψηλό επίπεδο, ο αλγόριθμος των Viola-Jones ανιχνεύει σε ένα παράθυρο με μια εικόνα τα χαρακτηριστικά ενός ανθρώπινου προσώπου. Αν αρκετά από αυτά τα χαρακτηριστικά βρεθούν, τότε η εικόνα αυτή λέγεται ότι είναι ένα πρόσωπο. Για να μπορέσουν να αντιπροσωπευτούν τα διαφορετικά μεγέθη προσώπων, το παράθυρο είναι κλιμακωτό και η διαδικασία επαναλαμβάνεται. Κάθε κλιμακωτό παράθυρο προχωράει μέσα από τον αλγόριθμο, ανεξάρτητα από τις άλλες κλίμακες. Για να μειωθεί ο αριθμός των χαρακτηριστικών κάθε παραθύρου πρέπει να ελεγχθούν, κάθε παράθυρο περνάει από στάδια. Στα πρώτα στάδια έχουν λιγότερα χαρακτηριστικά να ελέγχουν και να είναι πιο εύκολο να περάσουν ενώ στα προχωρημένα στάδια έχουν αυστηρότερα χαρακτηριστικά και είναι πιο αυστηρά. Σε κάθε στάδιο, οι υπολογισμοί των χαρακτηριστικών γνωρισμάτων για το εκάστοτε στάδιο συσσωρεύονται και, αν αυτή η συσσωρευμένη αξία δεν περνά το όριο, το στάδιο αυτό είναι αποτυχημένο και η εικόνα δεν θεωρείται ένα πρόσωπο. Αυτό επιτρέπει σε εικόνες, που δεν μοιάζουν με ένα πρόσωπο να μην περνάνε διεξοδικό έλεγχο.

Για να σε κατανοήσουμε σε βάθος τον αλγόριθμο, ορισμένες λεπτομέρειες πρέπει να καθοριστούν συμπεριλαμβανομένων των χαρακτηριστικών γνωρισμάτων, μίας ειδικής αναπαράστασης της εικόνας που είναι γνωστή ως εσωτερική εικόνα, καθώς και ένα στάδιο cascading.

4.2.1. Χαρακτηριστικά

Οι ταξινομητές χαρακτηριστικών χρησιμοποιούνται για την ανίχνευση ιδιαίτερων χαρακτηριστικών του προσώπου. Οι εικόνες σαρώνονται συνεχώς για να βρεθούν τα χαρακτηριστικά, με τον αριθμό των χαρακτηριστικών να αυξάνεται ανάλογα με τη φάση που είναι. Τα χαρακτηριστικά παρουσιάζονται ως ορθογώνια και οι ιδιαίτεροι ταξινομητές που

χρησιμοποιούμε αποτελούνται. Στο Σχήμα 10 φαίνεται ένα παράδειγμα μιας τέτοιας ταξινόμησης χαρακτηριστικών.

Για να υπολογίσουμε την αξία ενός χαρακτηριστικού, υπολογίζουμε πρώτα το άθροισμα όλων των εικονοστοιχείων που περιέχονται σε καθένα από τα ορθογώνια που αποτελούν το χαρακτηριστικό. Μόλις υπολογιστεί, κάθε ποσό πολλαπλασιάζεται με το αντίστοιχο παραλληλόγραμμο του βάρους του και το αποτέλεσμα είναι συσσωρευμένο για όλα τα ορθογώνια του χαρακτηριστικού. Εάν η συσσωρευμένη αξία είναι ίση η μεγαλύτερη ένα όριο, τότε το χαρακτηριστικό έχει βρεθεί στην εικόνα υπό εξέταση. Τα βάροι και τις διαστάσεις των ορθογωνίων για κάθε χαρακτηριστικό λαμβάνονται από τα δεδομένα εκπαίδευσης της OpenCV.

Σχήμα 10. Τα μάτια. Αυτό είναι ένα παράδειγμα ενός ορθογωνίου 2:. Παράδειγμα ένα απλού χαρακτηριστικού του ατόμου. Το μέτωπο είναι ελαφρύτερο από τα μάτια τους. Αυτό είναι ένα παράδειγμα ενός ορθογωνίου 2 χαρακτηριστικών.

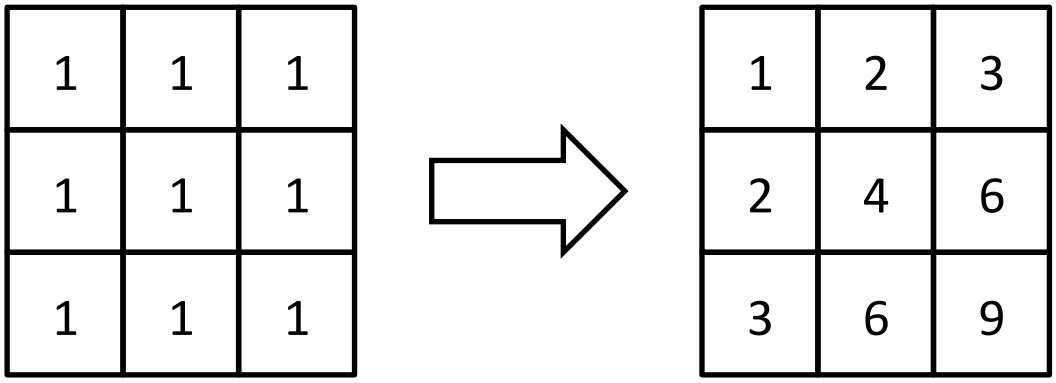
4.2.2. Εσωτερική Εικόνα

Για να αποφύγετε τον υπολογισμό των ποσών των ορθογωνίων, υπολογίζουμε την εσωτερική Εικόνα (II) ως προ-στάδιο της επεξεργασίας. Η εσωτερική εικόνα στη θέση (X, Y) περιλαμβάνει το άθροισμα των pixels πάνω και στα αριστερά του (x, y). Πιο

τυπικά, Ες. 1 δείχνει πώς η εσωτερική εικόνα ορίζεται, όπου II αντιπροσωπεύει την εσωτερική εικόνα και η εικόνα είναι στην αρχική διάσταση.

II (x, y) = Image (x, y) + II (x - 1, y) + II (x, y - 1) - II (x - 1, y - 1)

Το II (x - 1, y - 1) αφαιρείται αφού συμπεριλαμβάνεται άσκοπα στο σύνολο II (x-1, y) και II (X, Y-1). Το Σχήμα 11 δείχνει αυτό εικονογραφικά.



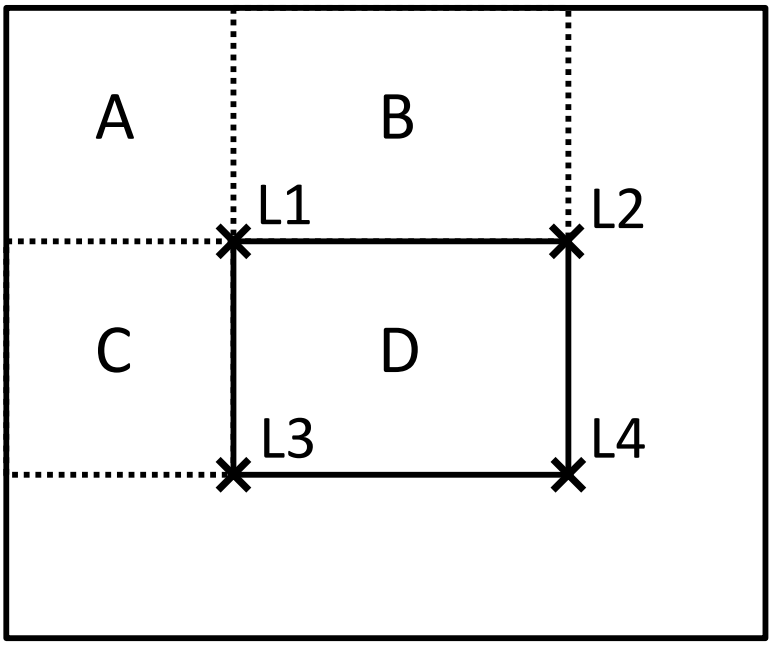
Σχήμα 11. Ένας 3 × 3 πίνακας και ο πίνακας της αντίστοιχης εσωτερικής εικόνας.

Χρησιμοποιώντας την εσωτερική εικόνα, τα χαρακτηριστικά μπορούν να υπολογιστούν με συνεχή φορά, μετά μπορούμε να υπολογίσουμε το άθροισμα των pixels με βάση τα ορθογώνια σε σχέση με τον χρόνο. Το σχήμα 12 δείχνει πώς η διαδικασία αυτή λαμβάνει χώρα.

Συγκεκριμένα, αν απαιτείται το άθροισμα των pixels στο ορθογώνιο D, εκτελούμε τον εξής υπολογισμό χρησιμοποιώντας τις Εσωτερικές Εικόνες στις τέσσερις γωνίες:

Sum = II (L4) - II (L3) - II (L2) + II (L1)

Προσθέτουμε πίσω το L1 επειδή το II (L1), έχει αφαιρεθεί από δύο φορές? L3 και L2 δύο περιέχουν την περιοχή που καλύπτεται από L1. Αν και τα χαρακτηριστικά μπορούν να υπολογιστούν σε συνεχή χρόνο, η υπερβολική εργασία θα γίνει αν σε μια συγκεκριμένη περιοχή εικόνας δεν φαίνεται τίποτα όπως ένα πρόσωπο. Ο αλγόριθμος χρησιμοποιεί πάνω από 2000 χαρακτηριστικά γνωρίσματα και θα ήταν αναποτελεσματικό να υπολογίσει όλα αυτά τα χαρακτηριστικά χωρίς λόγο. Για να αποφευχθεί αυτό το πρόβλημα ο αλγόριθμος χρησιμοποιεί ένα stage cascade για να μοιράσει τον αριθμό των χαρακτηριστικών και να εξαλείψει τα παράθυρα γρήγορα, όταν έχει διαπιστωθεί ότι δεν περιέχουν ένα πρόσωπο.

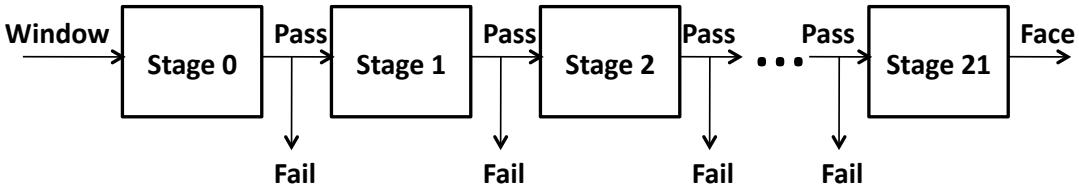


Σχήμα 12. Μέθοδος υπολογισμού της αξίας ενός χαρακτηριστικού χρησιμοποιώντας την εσωτερική εικόνα.

4.2.3. Στάδιο Cascade

Το στάδιο Cascade κρατά εικόνες, που δεν μοιάζουν με ένα πρόσωπο από το να αναλυθούν χωρίς λόγο. Αμέσως ετικετοποιεί ένα παράθυρο ως "μη πρόσωπο", όταν το παράθυρο αποτυγχάνει να περάσει το συγκεκριμένο στάδιο. Η εφαρμογή που χρησιμοποιούμε περιέχει 22 φάσεις, με τα πρώτα στάδια να περιέχουν λιγότερο λεπτομερή χαρακτηριστικά και τα μεταγενέστερα στάδια να περιέχουν περισσότερο λεπτομερή χαρακτηριστικά. Σε γενικές γραμμές, στα νωρίτερα στάδια περνιούνται πιο συχνά σε σχέση με τα προχωρημένα στάδια που είναι πιο αυστηρά. Έτσι, το ποσό της εργασίας σε κάθε συγκεκριμένο στάδιο ποικίλλει σε μεγάλο βαθμό. Η διαδικασία αυτή μπορεί να είναι πιο εύκολα κατανοητή από το σχήμα 13.

Στο Σχήμα 13, μπορεί να θεωρηθεί ότι ένα παράθυρο εισέρχεται στο στάδιο cascade στο 0. Εάν τα χαρακτηριστικά του συγκεκριμένου σταδίου όλα βρίσκονται στην εικόνα, το στάδιο μπορεί να περάσει και το παράθυρο διαδίδεται στο επόμενο στάδιο και το παράθυρο μπορεί και πάλι να σαρωθεί για τα χαρακτηριστικά του επομένου σταδίου. Αν το παράθυρο περάσει όλα τα στάδια, τότε λέγεται ότι είναι ένα πρόσωπο και η επόμενη εικόνα βρίσκεται προς επεξεργασία με τον ίδιο τρόπο.



Σχήμα 13. Εικόνες εισάγονται στον αλγόριθμο στο στάδιο 0 και διαδίδονται μέσα από τα στάδια. Στα πρώιμα στάδια περνούν πιο εύκολα και στα μεταγενέστερα στάδια πιο επιλεκτικά.

4.3. Hardware Based

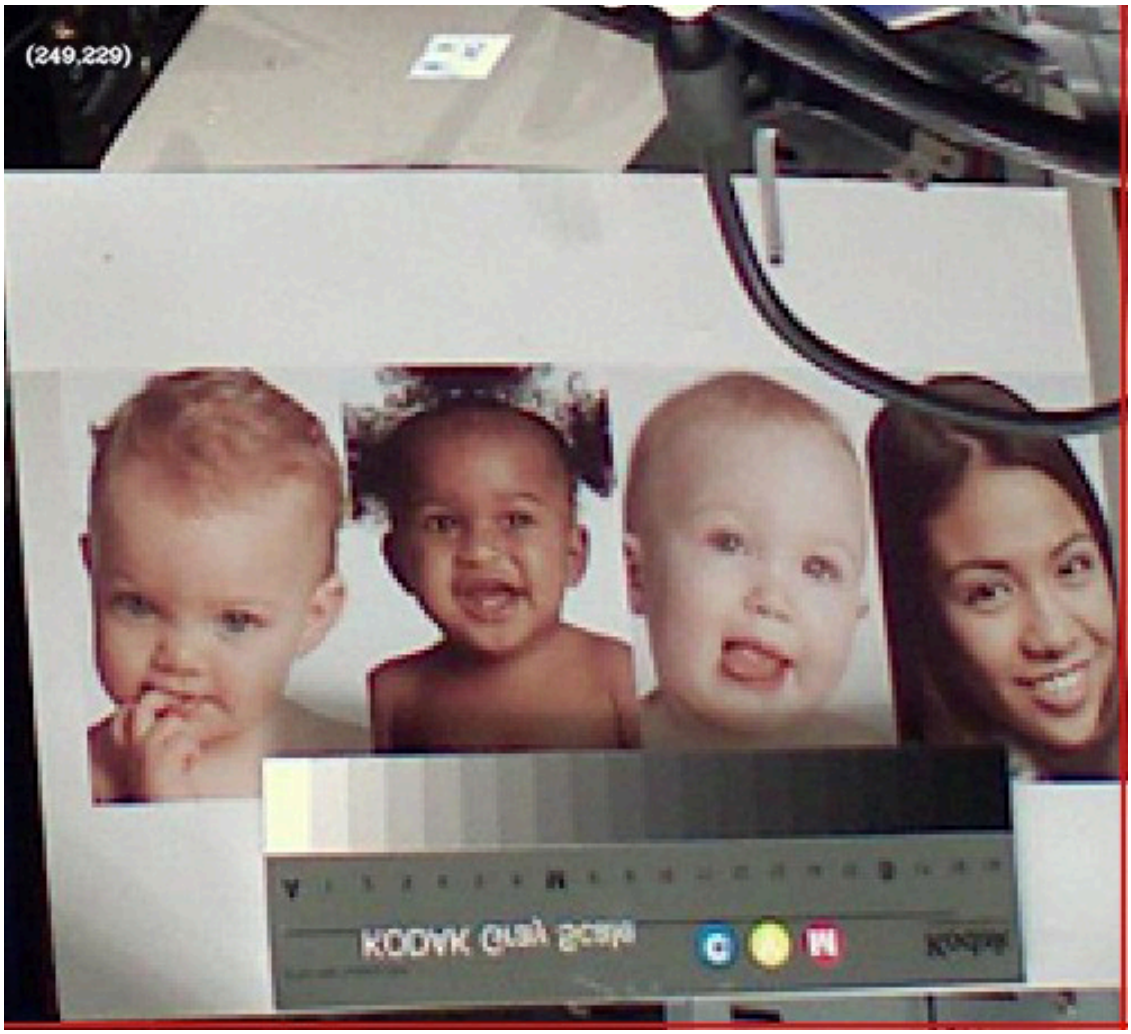
4.3.1. Ο αλγόριθμος ανίχνευσης προσώπου Εφαρμοσμένος

Οι εφαρμογές που αναφέρονται στο προηγούμενο κεφάλαιο έχουν μια κοινή λεπτομέρια: απαιτούν ένα μάλλον μεγάλο αριθμό πολύπλοκων μεταβιβάσεων δεδομένων pixel προς και από τη μνήμη του συστήματος. Αν μπορούσαμε να ελαχιστοποιήσουμε τον αριθμό των μεταφορών αυτών μέσω ειδικών επιταχυντών hardware στην μεταποίηση pixel και στα βήματα χαρακτηρισμού χαρακτηριστικών, τότε πολύ πιο γρήγορα και χαμηλότερης κατανάλωσης συστήματα μπορούν να δημιουργηθούν. Έτσι χωρίς να χάσει τη συμβατότητα με τη δομή της βιβλιοθήκης OpenCV δημιουργήθηκε ένας τροποποιημένος αλγόριθμος ανίχνευσης προσώπων που ταιριάζει καλύτερα στην επιτάχυνση υλικού.

Κατά την εφαρμογή μας επιλέξαμε να εφαρμοστεί ένας αλγόριθμος πέντε βημάτων που μπορεί άμεσα να επιταχυνθεί με δομές υλικού.

Τα πέντε βήματα είναι τα εξής:

- Προ-Επεξεργασία
- Ανίχνευση Πιθανών Χαρακτηριστικά Προσώπου (Candidate Face Features -- CFF)
- Δημιουργία μίας «φαίνεται-σαν-ένα πρόσωπο-(appears-like-a-face -- ALF)" μετρικής για κάθε CFF
- Άμεση αντιστοίχιση των υποψηφίων με ένα "πρωτότυπο πρόσωπο"
- Κατάργηση χαμηλών ALF υποψήφιων (οι όψεις δεν μπορούν να επικαλύπτονται)



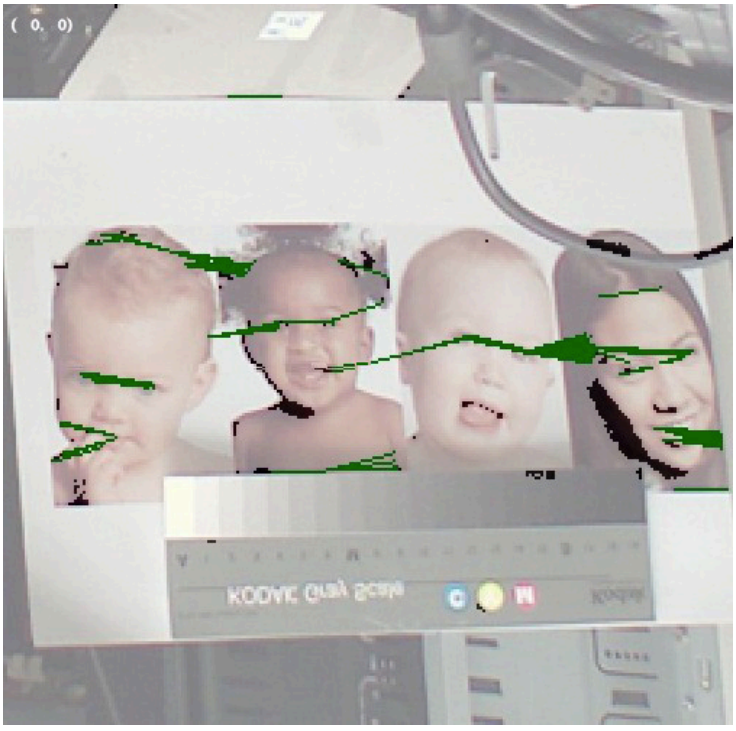
Σχήμα 14. Αρχική Εικόνα

Μερικές λεπτομέρειες για κάθε βήμα, ιδιαίτερα σε μέρη που μπορεί να επιταχυνθεί με υλικό, αναφέρονται παρακάτω:

Προ-επεξεργασία: Πρόκειται για επεξεργασία στο επίπεδο του κάθε pixel, δημιουργώντας δύο συστοιχίες pixel, μία για τα pixel με τα χαρακτηριστικά χρώματα που ταιριάζουν στο ανθρώπινο δέρμα και το άλλο με τα τοπικά ανώτατα όρια της έντασης. Και οι δύο λειτουργίες μπορούν να εφαρμόζονται άμεσα από τη ροή εικόνων χωρίς λογισμικό. Τα σχήματα (15,16) δείχνουν την αρχική εικόνα, και την έξοδο επικάλυψης δέρματος.

Σχήμα 15. Επιλογή των pixels "δέρματος"

Ανίχνευση CFF: Έχουν δοκιμαστεί πολλές προτάσεις για τα υποψήφια χαρακτηριστικά προσώπου. Επιλέξαμε να χρησιμοποιήσουμε μια μέθοδο απλή και γρήγορη, επιλέγοντας όλα εκείνα τα σκοτεινά σημεία σε μια περιοχή του δέρματος και να βρεθεί το αντίστοιχο σημείο σε μια συγκεκριμένη περιοχή στα δεξιά του. Αν ο αλγόριθμος βρίσκει ένα δεύτερο σκοτεινό σημείο, στη συνέχεια, ένα νέο CFF δημιουργείται, που αντιστοιχεί σε ένα ζευγάρι μάτια σε ένα πραγματικό πρόσωπο.



Σχήμα 16. Υποψήφια Χαρακτηριστικά Προσώπου (ζεύγος ματιών)

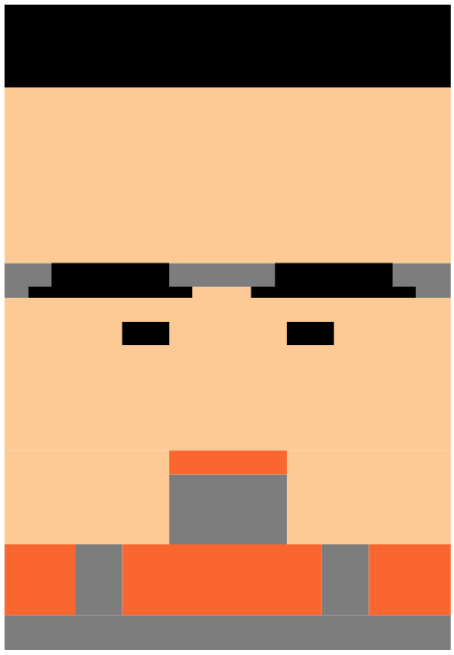
Το μετρικό ALF: για κάθε υποψήφιο πρόσωπο θα προσπαθήσει να εξακριβώσει εάν είναι ένα πραγματικό πρόσωπο εξετάζοντας το χρώμα και το σχήμα του. Σε αυτό το βήμα, κάτω και πάνω από τα μάτια, καθώς και στο μέτωπο και το πηγούνι αξιολογούνται συγκεκριμένες περιοχές. Κάθε αξιολόγηση γίνεται με τη δημιουργία ενός μετρητή των σκοτεινών pixels, τα φωτεινά pixel, pixels δέρματος και μη-δέρματος της αντίστοιχης περιοχής. Το άθροισμα όλων αυτών των μετρήσεων είναι η μετρική ALF. Αν και απλή, η προσέγγιση αυτή παρουσιάζει δύο σημαντικά πλεονεκτήματα: (α) η υλοποίηση σε υλικό είναι πραγματικά γρήγορη και απλή και (β) η συμπεριφορά του αλγορίθμου είναι πολύ εύκολα κατανοητή από τον χρήστη και βασίζεται στην ανθρώπινη αντίληψη των προσώπων. Το τελευταίο είναι πολύ χρήσιμο όταν ο χρήστης μπορεί να προσαρμόσει την έκφραση του / στάση του σώματος και να βελτιστοποιηθεί η ανίχνευση όπως συμβαίνει με πολλά σε συστήματα πραγματικού χρόνου ανίχνευσης.

Σχήμα 17. Ορισμένες περιοχές που χρησιμοποιούνται σε μετρικές ALF

Ταιριάζοντας το "πρωτότυπο πρόσωπο» είναι μια άμεση σύγκριση του κάθε προσώπου υποψηφίου με το πρότυπο. Το πρότυπο καθορίζει μια γεωμετρική περιγραφή ενός προσώπου με τις περιοχές που καλύπτουν ένα ολόκληρο τετράγωνο (σχήμα 18). Κάθε περιοχή είναι ορθογώνια και έχει δύο χαρακτηριστικά:

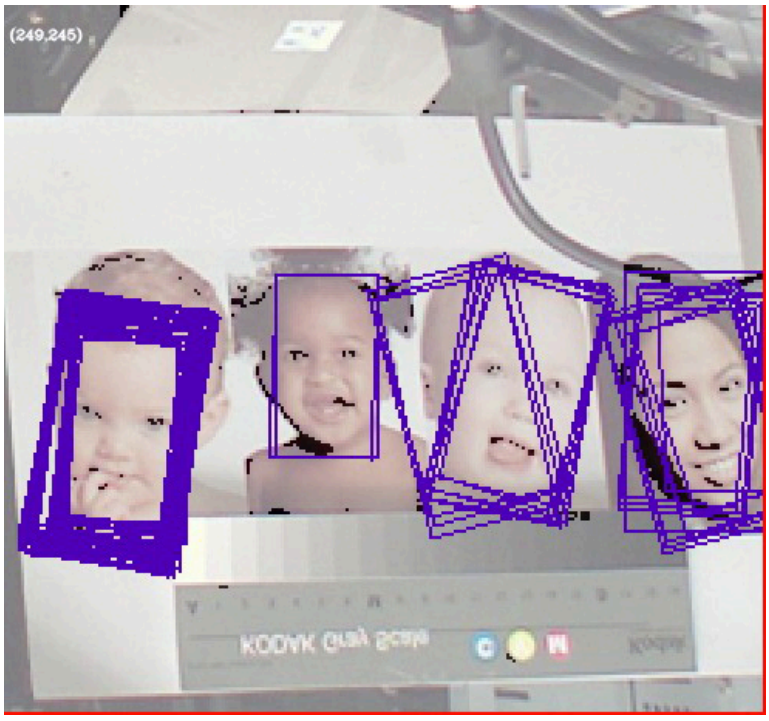
-Ποσοστό pixel Δέρματος

-Τοπικό Ποσοστό Σκοταδιού (ανά pixel)



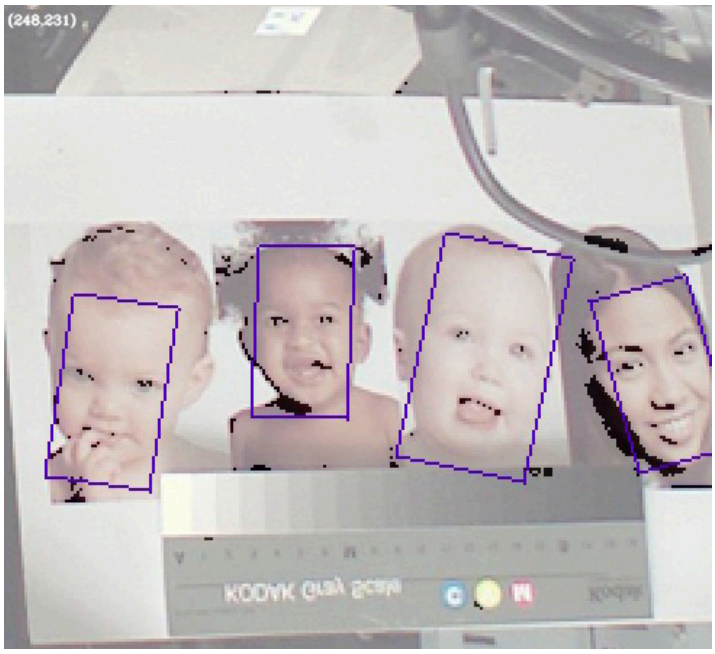
Σχήμα 18. Πρότυπο Προσώπου

Δεδομένου ότι πρόκειται για μια διαδικασία που θα χρησιμοποιήσει την επεξεργασία πόρων, ακόμη και αν εφαρμοστούν σε εξειδικευμένο υλικό, αυτό θα ισχύει μόνο για τους υποψηφίους που έχουν ήδη μια σημαντική ALF μετρική μορφή από το προηγούμενο βήμα. Το αποτέλεσμα είναι μια ενημερωμένη μετρική ALF η οποία θα χρησιμοποιείται κατά το τελευταίο βήμα για την αξιολόγηση των υποψηφίων.

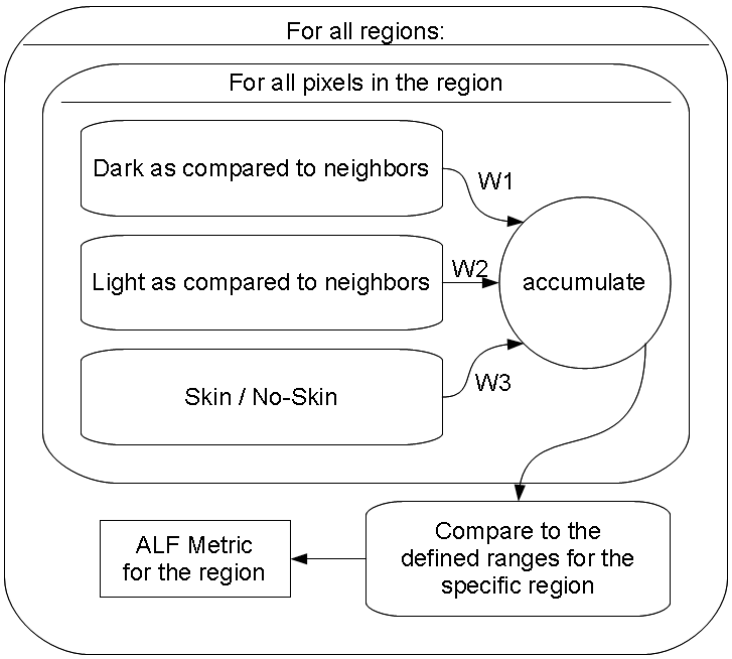


Σχήμα 19: Τα πρόσωπα που έχουν εντοπιστεί.

Η κατάργηση των «ψευδών» γίνεται σε επίπεδο λογισμικού με μια γρήγορη επαναληπτική έρευνα γύρω από κάθε υποψήφιο που αφήνει μόνο μία όψη ανάλογα με την περιοχή. Η σύγκριση γίνεται βάση των ALF μετρικών, αλλά και με άλλα υψηλότερου επιπέδου κριτήρια μπορούν επίσης να προστεθούν στο σημείο αυτό (π.χ. όταν η κάμερα δείχνει σε ένα μεγάλο δωμάτιο, πρόσωπα με μικρές διαστάσεις, που δεν αναμένεται να είναι κοντά στην κάμερα). Το παρακάτω σχήμα δείχνει τα πραγματικά πρόσωπα που έχουν ανιχνευθεί.



Σχήμα 20: Τελικό αποτέλεσμα



Σχήμα 21: Η αρχιτεκτονική ροής δεδομένων του συστήματος ALF

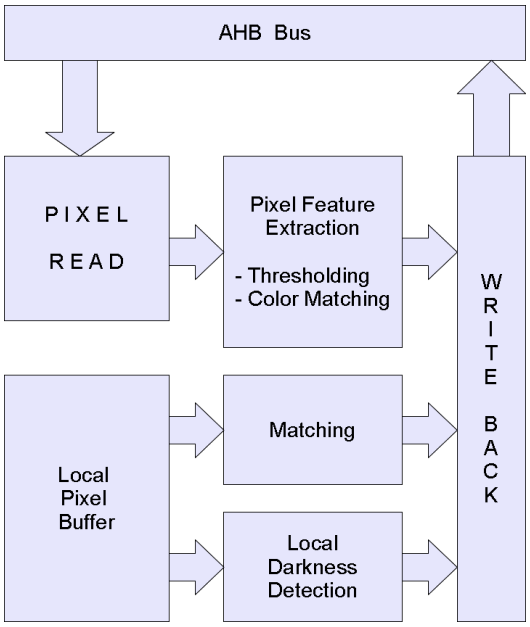
4.3.2. Η Υλοποιημένη Εναλλακτική Εφαρμογή

Προκειμένου να αξιολογηθούν οι συμβιβασμοί που η προγραμματισθείσα θα έχει να αντιμετωπίσει, έχουμε καθορίσει τέσσερις εναλλακτικές υλοποιήσεις, δύο που έχουν σχεδιαστεί ως ενσωματωμένες εφαρμογές λογισμικού της OpenCV και δύο που χρησιμοποιούν το τροποποιημένο αλγόριθμο που παρουσιάστηκε στην προηγούμενη ενότητα. Για κάθε ζεύγος είχαμε μια υπόθεση με εικόνες χαμηλής ανάλυσης (π.χ. χαμηλή ακρίβεια), καθώς και μια περίπτωση με εικόνες υψηλής ανάλυσης. Οι χαμηλής ανάλυσης εικόνες θεωρούμε ότι θα χωρέσουν σε on-chip μνήμες ενώ η υψηλής ευκρίνειας εικόνες θα έχουν πρόσβαση σε εξωτερική SDRAM στην πλακέτα του συστήματος.

- SW-based 256x256
- SW-based 1024x1024
- HW-accelerated 256x256
- HW-accelerated 1024x1024

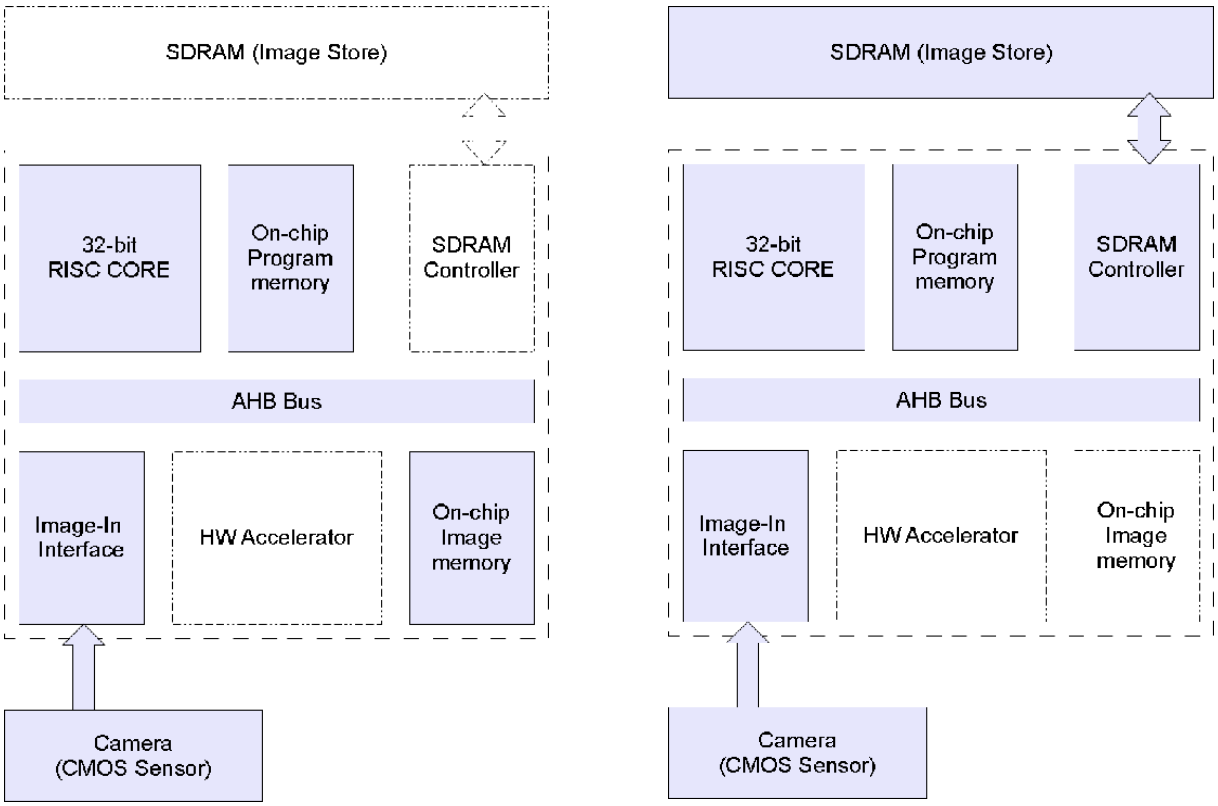
Η μεγαλύτερη διαφορά μεταξύ των δύο σειρών είναι η μονάδα επιτάχυνσης HW η οποία είναι ένας κινητήρας με ενσωματωμένο δίαυλο για την εύρεση χαρακτηριστικών επιπέδου pixel και τη σύγκριση των περιφερειών. Η μονάδα αυτή περιγράφεται στο σχήμα 22 και αποτελείται από:

- Διάβασε Pixel (Pixel Read): ένας ενσωματωμένος AHB οδηγός που μεταφέρει pixel από την εικόνα στη μνήμη.
- Γράψε (Write Back): ένας ενσωματωμένος AHB οδηγός γράφει επιπλέον bits για κάθε pixel στην μνήμη της εικόνας (Όταν η μονάδα λειτουργεί σε κατάσταση Συγκριτή, το μπλοκ που κάνει καταγραφή αποθηκεύει ένα μετρικό ομοιότητας μεταξύ των δύο περιοχών που συγκρίνονται).
- Τοπικό Pixel Buffer: χρησιμοποιείται για την αποθήκευση της γειτονιάς του κάθε pixel, εξοικονομώντας κύκλους μεταφοράς στον δίαυλο AHB.
- Τα τρία μπλοκ του φορέα (που ταιριάζουν, ελέγχουν τη φωτεινότητα και την εξαγωγή χαρακτηριστικών κάνουν την πραγματική επιτάχυνση των λειτουργιών).



Σχήμα 22: Ο επιταχυντής υλικού

Τα σχήματα των εικόνων 23α και 23β δείχνουν τις τέσσερις συνθέσεις όπως προέβλεπε για το FPGA.



Σχήμα 23α: διαμορφώσεις με βάση το λογισμικό

Σε περιπτώσεις λογισμικού, δεν υπάρχει επιτάχυνση υλικού. Χρησιμοποιείται κάποια λογική για τη διασύνδεση με τον αισθητήρα CMOS για λήψη εικόνας, καθώς και για τον δίαυλο του συστήματος. Σε μια διαμόρφωση μίας μεγάλης on-chip μνήμης χρησιμοποιείται, ενώ το άλλο περιλαμβάνει ένα on-chip ελεγκτή μνήμης SDRAM για τη σύνδεση με το εξωτερικό τσιπ μνήμης.

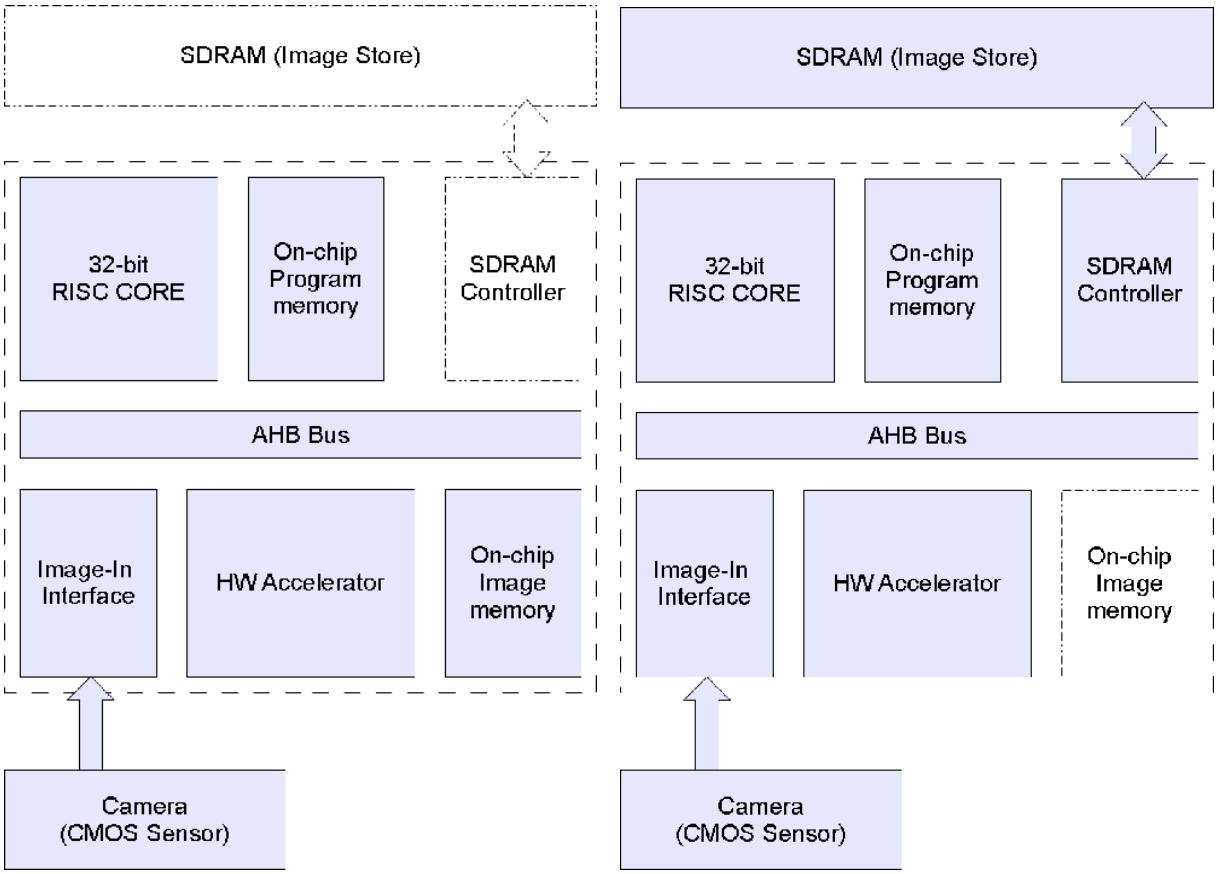
Οι HW-επιταχυνόμενες συνθέσεις διατηρούν όλες τις λειτουργικές μονάδες της SW-μόνο (συμπεριλαμβανομένου του κεντρικού επεξεργαστή). Ωστόσο, ο επεξεργαστής σε αυτές τις συνθέσεις δεν έχει να εκτελέσει όλες τις λειτουργίες και μπορούν ως εκ τούτου να λειτουργούν σε σημαντικά χαμηλότερη ταχύτητα.

Σύγκριση σε συχνότητα ρολογιού δεν είναι δυνατή. Για να έχουμε μια δίκαιη σύγκριση, θα ερευνηθούν για κάθε σύνθεση, δύο διαφορετικές απαιτήσεις ταχυτήτων:

- Υψηλής Ταχύτητας (60fps)
- Χαμηλής ταχύτητας (5fps)

Όπως διαφορετικές εφαρμογές απαιτούν διαφορετικές ταχύτητες ανίχνευσης, η ετυμηγορία μπορεί να διαφέρει για κάθε εφαρμογή, επισημαίνοντας τι είναι καλύτερο για κάποιο συγκεκριμένο σκοπό. Η απαίτηση του καρέ ανά δευτερόλεπτο (fps) έχει άμεσες επιπτώσεις για την απαιτούμενη ταχύτητα ρολογιού του εκάστοτε τσιπ. Εξαιτίας της ανάγκης να τρέξουν τα πάντα σχετικά με το λογισμικό σημαίνει ότι πρέπει να χρησιμοποιήσουμε ένα υψηλής ταχύτητας επεξεργαστή. Ευτυχώς, οι επεξεργαστές είναι κυκλώματα τα οποία έχουν σχεδιαστεί για ταχύτητα, ώστε να μπορούν σχετικά εύκολα κλιμακώνονται σε συχνότητα αν η τεχνολογία της εφαρμογής το υποστηρίζει.

Από την άλλη πλευρά, αυτή του σχεδιασμού του δικού μας επιταχυντή υλικού έχει επίσης αναβαθμιστεί αρκετά καλά (με την προσθήκη σταδίων αγωγού), δεδομένου ότι είναι από πολλές απόψεις μία αρχιτεκτονική ροής δεδομένων.



Σχήμα 23β: διαμορφώσεις με βάση τον επιτανχυτή υλικού

Για να μπορέσουμε να συγκρίνουμε σε ταχύτητα σε ίδια βάση, έπρεπε να βρούμε που είναι η ταχύτητα ρολογιού για κάθε διαμόρφωση που μπορεί να υποστηρίζει την απαιτούμενη ποσοτητά πλαισίων. Αυτό που είμαστε σε θέση να μετρήσουμε δίνεται στα ακόλουθα σημεία.

- Για τη περίπτωση Λογισμικού / Χαμηλή Ανάλυση, ο RISC πυρήνας μας πρέπει να τρέχει σε 200MHz για 5fps και 600MHz για 60fps
- Για τη περίπτωση Λογισμικού / Υψηλή Ανάλυση , ο RISC πυρήνας πρέπει να τρέχει σε 300MHz για 5fps και 800MHz για 60fps
- Για την περίπτωση HW-επιταχυντή / Χαμηλή Ανάλυση, το chip πρέπει να λειτουργεί σε 40MHz για 5fps και 150MHz για 60fps
- Για την περίπτωση HW-επιταχυντή / Υψηλή Ανάλυση, το chip πρέπει να λειτουργεί σε 50MHz για 5fps και 200MHz για 60fps

Τα αποτελέσματα αυτά έχουν συγκεντρωθεί από τον πειραματισμό στον πίνακα FPGA δεδομένο ότι ο πυρήνας RISC δεν μπορεί να πάει πάνω από 100MHz, έπρεπε να εξομαλύνουμε τις βασικές δοκιμές. Π.χ. για την υψηλής ευκρίνειας SW-only περίπτωση που

δεν θα μπορούσε να πάρει πάνω από 8 fps, ακόμη και όταν εφαρμόζονται όλες οι υποθέσεις για το frame-rate βίντεο συνεχούς ροής σε υψηλά επίπεδα. Αυτό σημαίνει ότι για να πάρει 60fps θα πρέπει να έχει τουλάχιστον 100 * 60 / 8 = 750MHz, εξ ου και η εκτίμηση 800MHz.

4.3.3. Χώρος και τα ενεργειακά αποτελέσματα

Για να αποκτηθούν τα αποτελέσματα που-εκτός από το σχεδιασμό του συστήματος και FPGA βασίζονται σε μετρήσεις κατά τη χρήση, χρησιμοποιήθηκε ο εκτιμητής τσιπ InCyte για την αξιολόγηση του χώρου του τσιπ και την κατανάλωση ενέργειας σε υλοποιήσεις IC.

Ο εκτιμητής InCyte είναι μια εφαρμογή που χρησιμοποιείτε για την εκτίμηση μεγέθους IC και την κατανάλωση ενέργειας για τις διάφορες διαμορφώσεις συστημάτων και τις διάφορες τεχνολογίες εφαρμογής. Μια βασική έκδοση είναι διαθέσιμη στο " http://www.chipestimate.com/incyteStarterEdition.html ". Σε εκτιμήσεις μας χρησιμοποιήσαμε ένα πρότυπο 130nm CMOS Standard Cell Technology και μπήκαν οι ίδιες διαμορφώσεις συστήματος που έχουμε για το FPGA.

Με τη λειτουργία του εκτιμητή σε κάθε μία διάταξη και σε διαφορετικές ταχύτητες ρολογιού, πήραμε τα αποτελέσματα που δίνονται στο σχήμα 24.

Algorithm	Image Resolution	Low Speed Clock (Mhz)	High Speed Clock (Mhz)	Low Speed Power (W)	High Speed Power (W)	Chip Area (sqmm)
SW Based	256x256 (on chip)	200	600	0,72	2,18	3,5
SW Based	1024x1024 (off chip)	300	800	1,02	2,8	1,7
HW Accelerated	256x256 (on chip)	40	150	0,18	0,65	4,1
HW Accelerated	1024x1024 (off chip)	50	200	0,21	0,85	2,1

Σχήμα 24. Πίνακας σύγκρισης Μεγέθους / Εκτιμήσεις ισχύος

4.4. Βιβλιογραφία Κεφαλαίου

[1]Raul Marin, German Leon, Raul Wirz, Jorge Sales, Jose M. Claver and Pedro Sanz, "Remote Control within the UJI Robotics Manufacturing Cell using FPGA-based vision", European Control Conference, 2007Ivan Aleksi, Zeljko Hocenski and Ivica Lukic "CLPD - and FPGA- Based Technology Applications in Embedded Systems Used in Transport and Industrial Control", unpublished

[2]Vanderlei Bonato, Adriano K. Sanches, M.M. Fernandes, Joao M. P. Cardoso, E. D. V. Simoes and Eduardo Marques, "A Real Time Gesture Recognition System for Mobile Robots", unpublished

[3]Daniel Hefenbrock, Jason Oberg, Nhat Tan Nguyen Thanh, Ryan Kastner and Scott B. Baden, "Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs", IEEE, IEEE International Symposium on Field-Programmable Custom Computing Machines, May 2010

[4]Hong Shan Neoh and Asher Hazanchuk, "Adaptive Edge Detection for Real Time Video Processing using FPGAs", unpublished

[5]Hossein Sahoolizadeh and Ahmad Keshavarz, "A FPGA Implementation of Neural/ Wavelet Face Detection System", Australian Journal of Basic and Applied Sciences, 4(3): 379-388, 2010

[6]Chun He, Aleksandros Papakonstantinou, and Deming Chen, "A Novel SoC Architecture on FPGA for Ultra Fast Face Detection", IEEE, IEEE International Conference on Computer Design, 2009

[7]Stavros Paschalakis and Miroslaw Bober, "Real-time face detection and tracking for mobile videoconferencing", Real-Time Imaging 10, 2004, pp. 81-94

[8]Sparsh Mittal, Saket Gupta and S. Dasupta, "FPGA: An Efficient And Promising Platform For Real-Time Image Processing Aplications", National Conference On Research & Development In Hardware & Systems (CSI-RDHS 2008), 2008, Kolkata

[9]Duy Nguyen, David Halupka, Parham Aarabi and Ali Sheikholeslami, "Real-Time Face Detection and Lip Feature Extraction Using Field-Programmable Gate Arrays", IEEE,

IEEE Transactions on Systems, Man., And Cybernetics - Part B: Cybernetics, Vol. 36, No4, August 2006

[10]Junguk Cho, Shahnam Mirzaei, Jason Oberg and Ryan Kastner, “FPGA-Based Face Detection Using Haar Classifiers”, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 103-112, 2009

[11]W. James MacLean, “An Evaluation of the Suitability of FPGAs for Embedded Vision Systems”, IEEE, IEEE Workshop on Embedded Computer Vision Systems, San Diego, June 2005

[12]Abbas Bigdeli, Colin Sim, Morteza Biglari-Abhari and Brian C. Lovell, “Face Detection on Embedded Systems”, ICESS, 2007, pp. 295-308

[13]Clement Farabet, Cyril Poulet and Yann LeCun, “An FPGA-Based Steam Processor for Embedded Real-Time Vision With Convolutional Networks”, IEEE, IEEE Workshop on Embedded Computer Vision, Kyoto, 2009

[14]Mohammad S. Sadri, Nasim Shams, Masih Rahmaty, Reihane Changiz, Shahed Mortazavian, Iraj Hosseini, Shima Kheradmand and Roozbeh Jafari, “An FPGA Based Fast Face Detector”, Global Signal Processing Expo & Conference (GSPx), July 2004, Santa Clara, CA

[15]Fethi Smach, Johel Miteran, Mohamed Atri, Julien Dubois, Mohamed Abid and Jean-Paul Gauthier, “An FPGA-based accelarator for Fourier Descriptors computing for color object recognition using SVM”, Journal of Real-Time Image Processing, 2007, pp. 249-258

[16]Maissa Ali, Joaquin Sitte and Ulf Witkowski, “Parallel Early Vision for Mobile Robots”, Proceedings of the 4th International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2007), U. Rueckert et al. (Eds.), HNI-Verlagsschiftenreihe, Vol. 216, Paderborn, Germany, October 2007, pp.133-140

[17]Debyo Saptono and Eri Prasetyo, “Development of Uclinux Platform for Computer Vision Algorithm in FPGA Devices”, unpublished

[18]Rob McCready, “Real-Time Face Detection on a Configurable Hardware Platform”, 2000, unpublished

[19]Christos Kyrkou, “Neural Network-Based Face Detector Implementation on a Virtex2 Pro FPGA Platform”, 2008, Unpublished

[20] Changjian Gao and Shih-Lien Lu, “Novel FPGA Based Haar Classifier Face Detection Algorithm Acceleration”, IEEE, IEEE International Conference Field Programmable Logic and Applications, 2008

[21]Stavros Paschalakis and Miroslaw Bober, “A Low Cost FPGA System for High Speed Face Detection and Tracking”, IEEE, IEEE International Conference on Field Programmable Technology (FTP ’03), Tokyo, Japan, Dec. 15-17, pp. 214-221, 2003

[22]Dimitris Maroulis, Dimitris K. Iakovidis and Dimitris Bariamis, “FPGA-based System for Real-time Video Texture Analysis”, Journal Of Signal Processing Systems, Volume 53, Number 3, pp. 419-433, SpringerLink

[23]Clement Farabet, Cyril Poulet, Jefferson Y. Han and Yann LeCun, “CNP: An FPGA-based Processor for Convolutional Networks”, IEEE, IEEE International Conference on Field Programmable Logic and Applications (FPL ’09), Prague, 2009

[24]K. Benkrid and D. Crookes, “Design and FPGA Implementation of a Perimeter Estimator”, unpublished

[25]Yuichi Hori, Kenji Shimizu, Yukata Nakamura and Tadahiro Kuroda, “A Real-Time Multi Face Detection Technique Using Positive-Negative Lines-of-Face Template”, IEEE, IEEE Conference on Pattern Recognition ’04, 2004

[26]Seunghun Jin, Dongkyin Kim Nguyen, T. T. Bongjin, Jun Daijin Kim and Jae Wook Jeon, “An FPGA-based Parallel Hardware Architecture for Real-Time Face Detection Using a Face Certainty Map”, IEEE, IEEE International Conference on Application-specific Systems, Architectures and Processors , 2009

5.Συμπεράσματα

Η παρούσα εργασία παρουσιάζει τις προσπάθειές μας στην διερεύνηση για το ποιοι είναι οι συμβιβασμοί που θα ασχοληθεί ένας σχεδιαστής κατά την προσθήκη λειτουργίας ανίχνευσης προσώπου σε ένα ενσωματωμένο σύστημα. Εμείς πειραματίστηκε με δύο αρχιτεκτονικές, ένα που χρησιμοποιεί μόνο το λογισμικό και ένα άλλο που εμπεριέχει ένα επιταχυντή υλικού.

Επιπλέον, εξετάσαμε δύο εφαρμογές που εξαρτώνται από την : Ανάλυσης εικόνας (ακρίβεια) και το Frame Rate (ταχύτητας). Οι διαφορετικές διαμορφώσεις εφαρμόζονται σε FPGA γενικότερης ανάπτυξης. Τα αποτελέσματα έχουν εξομαλυνθεί για να αναλάβει την υλοποίηση ASIC με τη χρήση άμεσων αναλογιών και περιοχής / Λογισμικού εκτίμησης της δύναμης.

Το αποτέλεσμα αυτής της δουλειάς είναι υπέρ των hardware-accelerated διαμορφώσεων εάν η κατανάλωση ενέργειας είναι η ανάγκη. Αυτό μπορεί να αναμένεται καθώς από την επιτάχυνση υλικού επιτρέπετε σε ολόκληρο το τσιπ που πρέπει να χρονομετρηθεί να είναι σε πολύ χαμηλότερες συχνότητες για ισοδύναμα ποσοστά πλαισίων.

Αν η κατανάλωση ενέργειας δεν είναι πρόβλημα, το αρχιτεκτονικές λογισμικού θα μπορούσαν να επιλεχθούν δεδομένου ότι απαιτούν μια περιοχή τσιπ μικρότερη κατά περίπου 15-20%.

Οι σχεδιαστές θα πρέπει φυσικά να λαμβάνουν άλλες παραμέτρους υπόψη, για παράδειγμα την απεριόριστη ευελιξία της λύσης λογισμικού σε σχέση με την περιορισμένη προσαρμοστικότητα μίας προσέγγισης υλικού. Ωστόσο, η μέθοδος που χρησιμοποιήθηκε (κινητήρας επιτάχυνσης) δεν επιβάλλει πολύ σημαντικούς περιορισμούς, δεδομένου ότι επιταχύνει λειτουργίες όπως εύρεση χαρακτηριστικών pixel και συγκρίσεις μπλοκ που αποτελούν μέρος όλων σχεδόν των αλγορίθμων υπολογιστικής όρασης.

6.ΠΑΡΑΡΤΗΜΑ Α

http://opencv.willowgarage.com/wiki/

http://www.2dix.com/pdf-2010/classification-face-detection-fpga-pdf.php

http://www.sciweavers.org/publications/fpga-based-face-detection-system-using-haar-classifiers

http://cseweb.ucsd.edu/~kastner/papers/fpga09-face_detection.pdfv

http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=4629966

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1334309&isnumber=29385

http://ebookbrowse.com/cvpr05-ecv-maclean-eval-pdf-d80635089

http://ebookbrowse.com/2007-remote-control-within-the-uji-robotics-fpga-based-vision-ecc2007-pdf-d46699409

http://www.altera.com/literature/cp/gspx/edge-detection.pdf

http://www.cs.qub.ac.uk/~D.Crookes/webpubs/papers/perimeter.doc

http://bib.irb.hr/datoteka/382068.SiP2008.pdf

http://www.nicta.com.au/__data/assets/pdf_file/0004/14944/Parallel_Early_Vision_Algorithms_for_Mobile_Robots.pdf

http://www.google.gr/url?sa=t&source=web&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.109.2042%26rep%3Drep1%26type%3Dpdf&ei=SJnTTcyDMoO28QOI7rHvCg&usg=AFQjCNHZovn1avFeLhMt8HhIThZmV05jFw&sig2=jbIlyCxqimlEZyEgjSW1CQ

http://www.google.gr/url?sa=t&source=web&cd=1&ved=0CB0QFjAA&url=http%3A%2F%2Fpusatstudi.gunadarma.ac.id%2Fpscitra%2Falldocpn%2Fpub-(wosoc08debyo_pdf)-6a5aa.pdf&ei=fpnTTeLrEsKp8QPdsPThCg&usg=AFQjCNGATpuXjyx-wuVXZwbFrI0HTe_6-w&sig2=hZ3GvDW2zNGPy6n2WqR8QA

http://www.insipub.com/ajbas/2010/379-388.pdf

http://class.ece.iastate.edu/cpre583/papers/Shih-Lien_Lu_FPL2008.pdf

http://cseweb.ucsd.edu/~kastner/papers/fpga09-face_detection.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.148.5713&rep=rep1&type=pdf

http://www.eecg.toronto.edu/~jayar/pubs/theses/Mccready/RobertMccready.pdf

http://icims.csl.illinois.edu/~dchen/research/Face-iccd09.pdf

http://nnguyenthanh.ucsd.edu/tan/publish/fccm_10.pdf

http://sokryk.tripod.com/Senior_Design_Project.pdf

http://www.uk.mitsubishielectric-rce.eu/pubdocs/VIL04-D123.pdf

http://www.cs.cmu.edu/~dst/NIPS/nips08-workshop/Yann_LeCun.pdf

http://yann.lecun.com/exdb/publis/pdf/farabet-ecv-09.pdf

http://www.nicta.com.au/__data/assets/pdf_file/0020/14924/Face_Detection_on_Embedded_Systems.pdf

http://www.apl.utoronto.ca/publication/i/nguyen-j.pdf

http://johel.m.free.fr/user_data/publications/JRTIPversion_18draft.pdf

http://scholar.lib.vt.edu/theses/available/etd-05242007-052642/unrestricted/INVESTIGATION_OF_NEW_TECHNIQUES_FOR_FACE_DETECTION_Abdallah_Abdallah.pdf

http://rtsimage.di.uoa.gr/publications/VLSI.pdf

http://www.clapedar.com/Documents/PerezDArpinoClaudia_2009_CLAWAR_FPGAvisionLuminosityVariability.pdf

http://en.wikipedia.org/wiki/Field-programmable_gate_array

7.ΠΑΡΑΡΤΗΜΑ Β

Εγκατάσταση OpenCV σε OSX

Πρώτος τρόπος μέσο MacPorts (Θεωρούμε οτι τα MacPorts είναι εγκατεστημένα).

sudo port selfupdate

sudo port install opencv

/*(64 bit)*/

sudo port -v install opencv +python26

ή

sudo port -v install opencv +python27

Δεύτερος τρόπος μέσο HomeBrew

sudo brew update

sudo brew install opencv (για 64 bit)

ή

sudo brew install opencv --build32 (για 32 bit)

sudo brew update

sudo brew install lame (για libmp3lame το οποίο χρειάζεται η opencv)

sudo brew install libpng (το οποίο χρειάζεται η theora - κομμάτι του πακέτου ffmpeg)

sudo brew link libpng (χρειάζεται εάν η theora/ffmpeg δεν μπορούν να το βρουν)

sudo brew install ffmpeg (για libavcodec και λοιπά stuff τα οποία χρειάζεται η opencv)

sudo brew install python (το σύστημα της python δεν είναι αρκετό)

sudo brew install doxygen (εάν κάνεις από μόνος σου compile opencv, χρειάζεται)

sudo chmod 000 /System/Library/Frameworks/Python.framework/

sudo chmod 000 /Library/Frameworks/Python.framework/

(απενεργοποιεί το σύστημα της python/s προσωρινά για το επόμενο βήμα)

είτε:

sudo brew install opencv (επιτέλους! - αυτό μάλλον θα δουλέψει?)

ή:

svn co https://code.ros.org/svn/opencv/trunk/opencv cd opencv (ή βάλε τους πηγαίους κώδικες με το χέρι)

cd opencv

ccmake . (απενεργοποίησε ότι δεν χρειάζεται, ίσως να χρειαστεί να καταλάβεις πώς γίνεται το build? μην ξεχάσεις να βάλεις 't' για advanced mode. Τροποποίησε τις παραμέτρους όπως σε βολεύουν , μετά 'c' και 'g')

cmake .

make -j8

make install (προαιρετικό)

sudo chmod 755 /System/Library/Frameworks/Python.framework/

sudo chmod 755 /Library/Frameworks/Python.framework/

(τα βάζει όπως ήταν)

Τρίτος τρόπος μέσω CMake build system (recomended)

sudo port install subversion

εάν βγάλει λάθος

sudo fink install svn-ssl

sudo port install cmake

svn co https://code.ros.org/svn/opencv/trunk/opencv cd opencv

svn co https://code.ros.org/svn/opencv/trunk/opencv -r 3059 cd opencv

Χρήση CMake για εκτέλεση

sudo cmake -G "Unix Makefiles" .

Μπορεί κανείς να προσδιορίσει πολλά configuration options στην εντολή. Παραδείγματος χάριν :

•-D BUILD_TESTS=OFF -> εάν δεν θέλεις να εκτελέσεις tests.

•-D BUILD_NEW_PYTHON_SUPPORT=OFF -> εάν δεν θές υποστήριξη Python.

•-D WITH_CARBON=ON -> (κυρίως για Leopard και κάτω) εάν θες να χρησιμοποίησης Carbon για UI αντί για Cocoa

•-D WITH_QUICKTIME=ON -> (κυρίως για Leopard και κάτω) εάν θες να χρησιμοποίησης QuickTime για Video I/O αντί του QTKit . Εάν χρησιμοποιείς Snow Leopard και χριάζεσαι Carbon και QuickTime, πρέπει να προσδιορίσεις επίσης

- •-D CMAKE_OSX_ARCHITECTURES=i386, -D CMAKE_C_FLAGS=-m32 και
- • -D CMAKE_CXX_FLAGS=-m32.

Στην περίπτωση που χρειάζεσαι την CMake GUI για άλλες επιλογές, αντικατέστησε την cmake με ccmake. Και εάν δεν δουλεύουν τα βελάκια, μπορείς να χρησιμοποιήσεις τις ακόλουθες εντολές:

•ctrl + N : επόμενο

•ctrl + P :προηγούμενο

•ctrl + D :κάτω μία σελίδα

•ctrl + U : πάνω μία σελίδα

Τέλος:

sudo make -j8

sudo make install

Χρήση των βιβλιοθηκών OpenCV σε ένα Xcode OS X project (opencv 2.1 since SVN r3060) (για Xcode 3.1.x και C++)

•Δημιούργησε ένα νέο Xcode project χρησιμοποιώντας το Command Line Utility/ Standard Tool template

•Επέλεξε Project -> Edit Project Settings

•Επέλεξε το Build tab

•Όρισε Configuration to All Configurations

•Στο Architectures, double-click Valid Architectures και αφαίρεσε όλες τις PPC αρχιτεκτονικές (Εάν χρησιμοποιείς τις βιβλιοθήκες μόνο σε 64-bit, αφαίρεσε και τις i386 αρχιτεκτονικές)

•Στο Search Paths όρισε Header Search Paths σε /usr/local/include (/opt/local/include εάν χρησιμοποιείς MacPorts) και τσέκαρε το Recursive (11/5/2011: σε εμένα δούλεψε ορίζοντας το Header search paths σε /usr/local/include AND /usr/local/include/ opencv , χωρίς recursive.)

•Κλείσε το παράθυρο Project Info

•Επέλεξε Project -> New Group και δημιούργησε ένα group που θα λέγεται OpenCV Frameworks

•Με το νέο group επιλεγμένο, επέλεξε Project -> Add to Project…

•Πάτα το "/" για να πας στο folder prompt

•Βάλε /usr/local/lib (/opt/local/lib εάν χρησημοποιείς MacPorts)

•Επέλεξε libopencv_core.dylib, libopencv_highgui.dylib, και libopencv_imgproc.dylib .

•Πάτα Add

•Ξετσέκαρε το Copy Items… και πάτα Add

Λογικά τώρα θα μπορείς να προσθέσεις τις βιβλιοθήκες OpenCV, να κάνεις compile, και run τα project. Το μόνο #include directive που θα χρειαστείς είναι το #include <opencv.hpp>. Με, την σειρά του, περιλαμβάνει τα υπόλοιπα opencv header files.

Χρήση των βιβλιοθηκών OpenCV σε ένα Xcode OS X project (opencv 2.0, and 2.1 μέσο MacPorts) (για Xcode 3.1.x)

•Δημιούργησε ένα νέο Xcode project χρησιμοποιώντας το Command Line Utility/ Standard Tool template

•Επέλεξε Project -> Edit Project Settings

•Επέλεξε το Build tab

•Όρισε Configuration to All Configurations

•Στο Architectures, double-click Valid Architectures και αφαίρεσε όλες τις PPC αρχιτεκτονικές

•Στο Search Paths όρισε Header Search Paths σε /usr/local/include (/opt/local/include εάν χρησιμοποιείς MacPorts)

•Κλείσε το παράθυρο Project Info

•Επέλεξε Project -> New Group και δημιούργησε ένα group που θα λέγεται OpenCV Frameworks

•Με το νέο group επιλεγμένο, επέλεξε Project -> Add to Project…

•Πάτα το "/" για να πας στο folder prompt

•Βάλε /usr/local/lib (/opt/local/lib εάν χρησημοποιείς MacPorts)

•Επέλεξε libopencv_core.dylib, libopencv_highgui.dylib, και libopencv_imgproc.dylib .

•Πάτα Add

•Ξετσέκαρε το Copy Items… και πάτα Add

Λογικά τώρα θα μπορείς να προσθέσεις τις βιβλιοθήκες OpenCV, να κάνεις compile, και run τα project.

Κώδικας C++

```
#include <OpenCV/OpenCV.h>

#include <cassert>

#include <iostream>

#include <time.h>


const char  * WINDOW_NAME  = "Face Tracker";

const CFIndex CASCADE_NAME_LEN = 2048;

      char      CASCADE_NAME[CASCADE_NAME_LEN] = "~/opencv/data/haarcascades/
haarcascade_frontalface_alt2.xml";


using namespace std;


// simeio orismou tou metriti

float timer; /* orismos tou timer*/

void tic() /* timer input */
```

```
{
	timer=clock();
}
int toc() /* timer outcome */
{
	float tdif;
	tdif=clock()-timer;
	printf("%f\t\t",tdif/1000);
	return tdif;
}



#define SOFTWARE
//#define HARDWARE

void tesyd_portable_prepare_frame(IplImage *current_frame, IplImage *gray_image, IplImage *small_image)
{
	//int time_in_function;

#ifdef SOFTWARE
	// convert to gray and downsize
	//printf("\nconvert to gray->\t");
	//tic();
```

```
        cvCvtColor (current_frame, gray_image, CV_BGR2GRAY);
        //toc();
        //printf("downsize->\t");
        //tic();
        cvResize (gray_image, small_image, CV_INTER_LINEAR);
        //toc();
#else
        HW_cvCvtColor (current_frame, gray_image, CV_BGR2GRAY);
        HW_cvResize (gray_image, small_image, CV_INTER_LINEAR);
#endif




}



CvSeq * tesyd_portable_detect_faces(IplImage *small_image, CvHaarClassifierCascade*
cascade, CvMemStorage* storage)
{
#ifdef SOFTWARE
   // detect faces
        printf("\ndetect faces->\t");
        tic();
   CvSeq* faces = cvHaarDetectObjects (small_image, cascade, storage,
```

```
                                                        1.1,        2,
CV_HAAR_DO_CANNY_PRUNING,
                                                        cvSize (30, 30));
        toc();
        return faces;
#else
        // detect faces
   CvSeq* faces = HW_cvHaarDetectObjects (small_image, cascade, storage,
                                                        1.1,        2,
CV_HAAR_DO_CANNY_PRUNING,
                                                        cvSize (30, 30));
        return faces;
#endif
}

int main (int argc, char * const argv[])
{
        //tic();
   const int scale = 2;

   // locate haar cascade from inside application bundle
   // (this is the mac way to package application resources)
   CFBundleRef mainBundle  = CFBundleGetMainBundle ();
   assert (mainBundle);
       CFURLRef        cascade_url  =  CFBundleCopyResourceURL (mainBundle,  CFSTR
("haarcascade_frontalface_alt2"), CFSTR("xml"), NULL);
```

```
   assert (cascade_url);
   Boolean     got_it      = CFURLGetFileSystemRepresentation (cascade_url, true,
                                                              reinterpret_cast<UInt8 *>(CASCADE_NAME),
CASCADE_NAME_LEN);
   if (! got_it)
      abort ();

   // create all necessary instances
         printf("\ncreate all necessary instances->\t");
         tic();
   cvNamedWindow (WINDOW_NAME, CV_WINDOW_AUTOSIZE);
   CvCapture * camera = cvCreateCameraCapture (CV_CAP_ANY);
            CvHaarClassifierCascade*  cascade  =  (CvHaarClassifierCascade*)  cvLoad
(CASCADE_NAME, 0, 0, 0);
   CvMemStorage* storage = cvCreateMemStorage(0);
   assert (storage);
         toc();

   // you do own an iSight, don't you ?!?
   if (! camera)
      abort ();

   // did we load the cascade?!?
   if (! cascade)
      abort ();
```

```
// get an initial frame and duplicate it for later work
        printf("\nget an initial frame and duplicate it for later work->\t");
        tic();
IplImage *  current_frame = cvQueryFrame (camera);
IplImage *  draw_image    = cvCreateImage(cvSize (current_frame->width, current_frame->height), IPL_DEPTH_8U, 3);
IplImage *  gray_image    = cvCreateImage(cvSize (current_frame->width, current_frame->height), IPL_DEPTH_8U, 1);
      IplImage *    small_image     = cvCreateImage(cvSize (current_frame->width / scale, current_frame->height / scale), IPL_DEPTH_8U, 1);
assert (current_frame && gray_image && draw_image);
        toc();

// as long as there are images ...
while (current_frame = cvQueryFrame (camera))
{
                printf("\ntesyd_portable_prepare_frame->\t");
                tic();
    tesyd_portable_prepare_frame(current_frame, gray_image, small_image);
    toc();

    CvSeq* faces = tesyd_portable_detect_faces(small_image, cascade, storage);

    // draw faces
                printf("\ndraw faces->\t");
```

```
                tic();
    cvFlip (current_frame, draw_image, 1);
    for (int i = 0; i < (faces ? faces->total : 0); i++)
    {
       CvRect* r = (CvRect*) cvGetSeqElem (faces, i);
       CvPoint center;
       int radius;
       center.x = cvRound((small_image->width - r->width*0.5 - r->x) *scale);
       center.y = cvRound((r->y + r->height*0.5)*scale);
       radius = cvRound((r->width + r->height)*0.25*scale);
       cvCircle (draw_image, center, radius, CV_RGB(0,255,0), 3, 8, 0 );
    }
    toc();
    // just show the image
                printf("\njust show the image->\t");
                tic();
    cvShowImage (WINDOW_NAME, draw_image);

    // wait a tenth of a second for keypress and window drawing
    int key = cvWaitKey (100);
    if (key == 'q' || key == 'Q')
       break;
                toc();
}
```

```
    // be nice and return no error
    return 0;
}
```

8.Βιβλιογραφία

[1]Learning OpenCV, Gary Bradski, Andrian Kaebler, O' Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, ISBN: 978-0-596-51613-0, 2008

[2] Wiśniewski, Remigiusz (2009). Synthesis of compositional microprogram control units for programmable devices. Zielona GoÅLra: University of Zielona GoÅLra. pp. 153. ISBN 978-83-7481-293-1.

[3]edu/~vaughn/challenge/fpga_arch.html FPGA Architecture for the Challenge]

[4] FPGA Signal Integrity tutorial (http:/ / wiki. altium. com/ display/ ADOH/ FPGA+ SI+ Tutorial+ -+ Simulating+ the+ Reflection+Characteristics)

[5]NASA: FPGA drive strength (http:/ / klabs. org/ richcontent/ fpga_content/ DesignNotes/ signal_quality/ actel_drive_strength/ index. htm)

[6]Mike Thompson. "Mixed-signal FPGAs provide GREEN POWER" (http:/ / www. eetimes. com/ showArticle. jhtml?articleID=200000777). EE Times, 2007-07-02.

[7] History of FPGAs (http:/ / filebox. vt. edu/ users/ tmagin/ history. htm)

[8] Google Patent Search, " Re-programmable PLA (http:/ / www. google. com/ patents? id=BB4vAAAAEBAJ& dq=4508977)". Retrieved February 5, 2009.

[9]Google Patent Search, " Dynamic data re-programmable PLA (http:/ / www. google. com/ patents?id=1-gzAAAAEBAJ& dq=4524430)". Retrieved February 5, 2009.

[10]Peter Clarke, EE Times, " Xilinx, ASIC Vendors Talk Licensing (http:/ / www. eetimes. com/ story/ OEG20010622S0091)." June 22, 2001. Retrieved February 10, 2009.

[11] Funding Universe. " Xilinx, Inc. (http:/ / www. fundinguniverse. com/ company-histories/ Xilinx-Inc-Company-History. html)" Retrieved January 15, 2009.

[12] Clive Maxfield, Programmable Logic DesignLine, " Xilinx unveil revolutionary 65nm FPGA architecture: the Virtex-5 family (http://www. pldesignline. com/ products/ 187203173). May 15, 2006. Retrieved February 5, 2009.

[13] Press Release, " Xilinx Co-Founder Ross Freeman Honored as 2009 National Inventors Hall of Fame Inductee for Invention of FPGA (http:// press. xilinx. com/ phoenix. zhtml?c=212763& p=irol-newsArticle& ID=1255523& highlight)"

[14]Clive Maxfield, book, " The Design Warrior's Guide to FPGAs (http:/ / books. google. com/ books?id=dnuwr2xOFpUC& pg=PA4&lpg=PA4& dq=FPGA+ Market+ growth+ 1990s& source=web& ots=YjFedB35Vp& sig=EH8y56Cih9iNLEqYXkZ63iO46K4& hl=en&sa=X& oi=book_result& resnum=4& ct=result)".Published by Elsevier, 2004. ISBN 0750676043, 9780750676045. Retrieved February 5, 2009

[15] "Original Paper on Work, Thompson" (http:/ / www. informatics. sussex. ac. uk/ users/ adrianth/ ices96/ paper. ps). .

[16]Dylan McGrath, EE Times, " FPGA Market to Pass $2.7 Billion by '10, In-Stat Says (http:/ / www. eetimes. com/ news/ design/ business/showArticle. jhtml? articleID=188102617)". May 24, 2006. Retrieved February 5, 2009.

[17]Narinder Lall eASIC Corporation, " FPGA Judgment Day:Rise of Second Generation Structured ASICs (http:/ / www.opensystems-publishing. com/ e-letter/ dsp/ 2008/ 03/ easic. pdf). March, 2008. Retrieved February 5, 2009.

[18]Dylan McGrath, EE Times, " Gartner Dataquest Analyst Gives ASIC, FPGA Markets Clean Bill of Health (http:/ / www. eetimes. com/conf/ dac/ showArticle. jhtml? articleID=164302400)". June 13, 2005. Retrieved February 5, 2009.

[19]Virtex-4 Family Overview (http:/ / www. xilinx. com/ support/ documentation/ data_sheets/ ds112. pdf)

[20]Tim Erjavec, White Paper, " Introducing the Xilinx Targeted Design Platform: Fulfilling the Programmable Imperative (http:/ / www. xilinx.com/ publications/ prod_mktg/ Targeted_Design_Platforms. pdf)." February 2, 2009. Retrieved February 2, 2009

[21] Huffmire Paper " Managing Security in FPGA-Based Embedded Systems (http:/ / www2. computer. org/ portal/ web/ csdl/ doi/ 10. 1109/MDT. 2008. 166)." Nov-Dec 2008. Retrieved Sept 22, 2009

[22] "Virtex-5 FPGA Configuration User Guide" (http:/ / www. xilinx. com/ support/ documentation/ user_guides/ ug191. pdf). Xilinx Inc.. August 2010. pp. 33–35. . Retrieved 2010-10-31.

[23]"Protecting Intellectual Property Through FPGA Design Security" (http:/ / www. altera. com/ literature/ ads/ fpgadesignsecurity. pdf). Altera Corporation. . Retrieved 2010-11-01.

[24]FPGA/DSP Blend Tackles Telecom Apps (http:/ / www. bdti. com/ articles/ info_eet0207fpga. htm)

[25]Xilinx aims 65-nm FPGAs at DSP applications (http:/ / www. eetimes. com/ showArticle. jhtml?articleID=197001881)

[26] http:/ / www. eecg. toronto. edu/ ~vaughn/ challenge/ fpga_arch. html

[27]http:/ / www. altera. com/ literature/ hb/ cyc2/ cyc2_cii51002. pdf

[28] http:/ / www. altera. com/ literature/ hb/ stratix-iv/ stx4_5v1_01. pdf

[29] http:/ / www. xilinx. com/ support/ documentation/ user_guides/ ug070. pdf

[30] http:/ / www. xilinx. com/ bvdocs/ whitepapers/ wp245. pdf

[31]Seeking Alpha, " Altera and Xilinx Report: The Battle Continues (http:/ / seekingalpha. com/ article/85478-altera-and-xilinx-report-the-battle-continues)". July 17, 2008. Retrieved February 5, 2009.

[32] "Xilinx ISE WebPACK" (http:/ / www. xilinx. com/ ise/ logic_design_prod/ webpack. htm). .

[33]"Quartus II Web edition software" (https:/ / www. altera. com/ support/ software/ download/ altera_design/ quartus_we/ dnl-quartus_we.jsp). .

[34] http:/ / www. achronix. com/

[35] "Achronix to use Intel's 22 nm manufacturing" (http:/ / newsroom. intel. com/ community/ intel_newsroom/ blog/ 2010/ 11/ 01/chip-shot-achronix-to-use-intel-s-22nm-manufacturing). .

[36]http:/ / www. tabula. com/

[37]Raul Marin, German Leon, Raul Wirz, Jorge Sales, Jose M. Claver and Pedro Sanz, “Remote Control within the UJI Robotics Manufacturing Cell using FPGA-based vision”, European Control Conference, 2007Ivan Aleksi, Zeljko Hocenski and Ivica Lukic “CLPD - and FPGA- Based Technology Applications in Embedded Systems Used in Transport and Industrial Control”, unpublished

[38]Vanderlei Bonato, Adriano K. Sanches, M.M. Fernandes, Joao M. P. Cardoso, E. D. V. Simoes and Eduardo Marques, “A Real Time Gesture Recognition System for Mobile Robots”, unpublished

[39]Daniel Hefenbrock, Jason Oberg, Nhat Tan Nguyen Thanh, Ryan Kastner and Scott B. Baden, “Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs”, IEEE, IEEE International Symposium on Field-Programmable Custom Computing Machines, May 2010

[40]Hong Shan Neoh and Asher Hazanchuk, “Adaptive Edge Detection for Real Time Video Processing using FPGAs”, unpublished

[41]Hossein Sahoolizadeh and Ahmad Keshavarz, “A FPGA Implementation of Neural/ Wavelet Face Detection System”, Australian Journal of Basic and Applied Sciences, 4(3): 379-388, 2010

[42]Chun He, Aleksandros Papakonstantinou, and Deming Chen, “A Novel SoC Architecture on FPGA for Ultra Fast Face Detection”, IEEE, IEEE International Conference on Computer Design, 2009

[43]Stavros Paschalakis and Miroslaw Bober, “Real-time face detection and tracking for mobile videoconferencing”, Real-Time Imaging 10, 2004, pp. 81-94

[44]Sparsh Mittal, Saket Gupta and S. Dasupta, “FPGA: An Efficient And Promising Platform For Real-Time Image Processing Aplications”, National Conference On Research & Development In Hardware & Systems (CSI-RDHS 2008), 2008, Kolkata

[45]Duy Nguyen, David Halupka, Parham Aarabi and Ali Sheikholeslami, “Real-Time Face Detection and Lip Feature Extraction Using Field-Programmable Gate Arrays”, IEEE, IEEE Transactions on Systems, Man., And Cybernetics - Part B: Cybernetics, Vol. 36, No4, August 2006

[46]Junguk Cho, Shahnam Mirzaei, Jason Oberg and Ryan Kastner, “FPGA-Based Face Detection Using Haar Classifiers”, ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 103-112, 2009

[47]W. James MacLean, “An Evaluation of the Suitability of FPGAs for Embedded Vision Systems”, IEEE, IEEE Workshop on Embedded Computer Vision Systems, San Diego, June 2005

[48]Abbas Bigdeli, Colin Sim, Morteza Biglari-Abhari and Brian C. Lovell, “Face Detection on Embedded Systems”, ICESS, 2007, pp. 295-308

[49]Clement Farabet, Cyril Poulet and Yann LeCun, “An FPGA-Based Steam Processor for Embedded Real-Time Vision With Convolutional Networks”, IEEE, IEEE Workshop on Embedded Computer Vision, Kyoto, 2009

[50]Mohammad S. Sadri, Nasim Shams, Masih Rahmaty, Reihane Changiz, Shahed Mortazavian, Iraj Hosseini, Shima Kheradmand and Roozbeh Jafari, “An FPGA Based Fast Face Detector”, Global Signal Processing Expo & Conference (GSPx), July 2004, Santa Clara, CA

[51]Fethi Smach, Johel Miteran, Mohamed Atri, Julien Dubois, Mohamed Abid and Jean-Paul Gauthier, “An FPGA-based accelarator for Fourier Descriptors computing for color object recognition using SVM”, Journal of Real-Time Image Processing, 2007, pp. 249-258

[52]Maissa Ali, Joaquin Sitte and Ulf Witkowski, “Parallel Early Vision for Mobile Robots”, Proceedings of the 4th International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2007), U. Rueckert et al. (Eds.), HNI-Verlagsschiftenreihe, Vol. 216, Paderborn, Germany, October 2007, pp.133-140

[53]Debyo Saptono and Eri Prasetyo, “Development of Uclinux Platform for Computer Vision Algorithm in FPGA Devices”, unpublished

[54]Rob McCready, “Real-Time Face Detection on a Configurable Hardware Platform”, 2000, unpublished

[55]Christos Kyrkou, “Neural Network-Based Face Detector Implementation on a Virtex2 Pro FPGA Platform”, 2008, Unpublished

[56] Changjian Gao and Shih-Lien Lu, “Novel FPGA Based Haar Classifier Face Detection Algorithm Acceleration”, IEEE, IEEE International Conference Field Programmable Logic and Applications, 2008

[57]Stavros Paschalakis and Miroslaw Bober, “A Low Cost FPGA System for High Speed Face Detection and Tracking”, IEEE, IEEE International Conference on Field Programmable Technology (FTP ’03), Tokyo, Japan, Dec. 15-17, pp. 214-221, 2003

[58]Dimitris Maroulis, Dimitris K. Iakovidis and Dimitris Bariamis, “FPGA-based System for Real-time Video Texture Analysis”, Journal Of Signal Processing Systems, Volume 53, Number 3, pp. 419-433, SpringerLink

[59]Clement Farabet, Cyril Poulet, Jefferson Y. Han and Yann LeCun, “CNP: An FPGA-based Processor for Convolutional Networks”, IEEE, IEEE International Conference on Field Programmable Logic and Applications (FPL ’09), Prague, 2009

[60]K. Benkrid and D. Crookes, “Design and FPGA Implementation of a Perimeter Estimator”, unpublished

[61]Yuichi Hori, Kenji Shimizu, Yukata Nakamura and Tadahiro Kuroda, “A Real-Time Multi Face Detection Technique Using Positive-Negative Lines-of-Face Template”, IEEE, IEEE Conference on Pattern Recognition ’04, 2004

[62]Seunghun Jin, Dongkyin Kim Nguyen, T. T. Bongjin, Jun Daijin Kim and Jae Wook Jeon, “An FPGA-based Parallel Hardware Architecture for Real-Time Face Detection Using a Face Certainty Map”, IEEE, IEEE International Conference on Application-specific Systems, Architectures and Processors , 2009