ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Σχεδίαση και Υλοποίηση Συστήματος Διεπαφής Βιντεοκάμερας και Επεξεργασίας Βίντεο σε Πλατφόρμα Αναδιατασσόμενης Λογικής

Νικόλαος Μανώλης

Επιβλέπων Καθηγητής:
Καθ. Απόστολος Δόλλας
Επιτροπή:
Καθ. Μιχαήλ Ζερβάκης
Δρ. Ευριπίδης Σωτηριάδης

Διπλωματική εργασία που υποβλήθηκε στα πλαίσια της ολοκλήρωσης του διπλώματος Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών

Πολυτεχνείο Κρήτης

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Χανιά, 2020

Πολυτεχνείο Κρήτης

Περίληψη

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΕΡΙΓΡΑΦΗ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Σχεδίαση και Υλοποίηση Συστήματος Διεπαφής Βιντεοκάμερας και Επεξεργασίας Βίντεο σε Πλατφόρμα Αναδιατασσόμενης Λογικής

Design and Implementation of a System to Interface Video Camera and Process Video on a Reconfigurable Logic-based Platform

Νικόλαος Μανώλης

Η επεξεργασία εικόνας θεωρείται ένας από τους πιο γρήγορα εξελισσόμενους τομείς στους κλάδους της επιστήμης των υπολογιστών και της μηχανικής. Οι εφαρμογές της αυξάνουν συνεχώς τις απαιτήσεις για υπολογιστική ισχύ, ειδικά αν ληφθεί υπόψη ο περιορισμός εξαγωγής αποτελεσμάτων σε πραγματικό χρόνο. Οι μοντέρνες FPGAs χρησιμοποιούνται συχνά ως πλατφόρμες για την υλοποίηση τέτοιου είδους εφαρμογών, επειδή η δομή τους είναι σε θέση να εκμεταλλευτεί τον χωρικό και χρονικό παραλληλισμό. Για τη διεπαφή με συσκευές FPGA, συνήθως γίνεται χρήση αισθητήρων εικόνας υψηλής ανάλυσης, που παρέχονται από εξειδικευμένες εταιρίες και κοστίζουν ακριβά. Το μεγάλο εύρος των ερευνών γύρω από τον τομέα επεξεργασίας εικόνας/βίντεο, που μπορούν να πραγματοποιηθούν σε μια FPGA, σε συνδυασμό με τους υπάρχοντες εξειδικευμένους και ακριβούς αισθητήρες εικόνας (κάμερες), δημιουργεί την ανάγκη για χρήση πολύ φθηνότερων καμερών σε περιπτώσεις όπου η υψηλή ανάλυση δεν είναι το μείζων ζήτημα.

Αυτή η εργασία παρουσιάζει τη σχεδίαση ενός συστήματος που πραγματοποιεί τη διεπαφή φθηνών USB καμερών με μια συσκευή SoC. Τα ιδιαίτερα χαρακτηριστικά που απορρέουν από την παράλληλη φύση της FPGA, οδηγούν σε καλύτερα αποτελέσματα επεξεργασίας εικόνας/βίντεο σε σχέση με έναν τυπικό επεξεργαστή. Παρουσιάζεται μια υλοποίηση αρχιτεκτονικής hardware, ικανή να πραγματοποιήσει επεξεργασία βίντεο και προβολή σε οθόνη HDMI. Μέσω λογισμικού Linux, ελέγχεται το σύστημα hardware και δημιουργούνται δύο εφαρμογές που επιδεικνύουν τις λειτουργίες της υλοποίησης. Για τις επιδόσεις του συστήματος, παρουσιάζονται και συγκρίνονται τα αποτελέσματα πραγματοποίησης επεξεργασίας βίντεο, στην FPGA και στον επεξεργαστή ARM της συσκευής SoC. Επιπλέον διαπιστώνονται οι συνθήκες υπό τις οποίες μπορεί να πραγματοποιηθεί ταυτόχρονη προβολή δύο USB καμερών, χρησιμοποιώντας τον ελεγκτή USB ενός SoC. Για την εφαρμογή του συστήματος σε πραγματικό hardware χρησιμοποιήθηκε το Zynq-7000 SoC ZedBoard.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Απόστολο Δόλλα για τη συνεχή στήριξη και την πολύτιμη καθοδήγησή του καθ' όλη τη διάρκεια της διαδικασίας αυτής της εργασίας, καθώς και για την ευκαιρία που μου έδωσε για περαιτέρω τριβή με τα αναδιατασσόμενα ψηφιακά συστήματα. Επιπλέον, θα ήθελα να ευχαριστήσω τον Δρ. Ευριπίδη Σωτηριάδη που ήταν πάντα δίπλα μου σε όλα τα στάδια της εργασίας, προσφέροντας χρήσιμες συμβουλές, οι οποίες συντέλεσαν στο να ξεπεράσω τα όποια προβλήματα και δυσκολίες προέκυπταν. Επίσης θα ήθελα να ευχαριστήσω τον καθηγητή κ. Μιχάλη Ζερβάκη που δέχτηκε να είναι μέλος της εξεταστικής επιτροπής.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου και τους φίλους μου για την υποστήριξη και τη δύναμη που μου πρόσφεραν όλα αυτά τα χρόνια.

Νικόλαος Μανώλης

Χανιά, 2020

Πίνακας Περιεχομένων

1 Εισαγωγή....9
1.1 Περιγραφή Εργασίας....9
1.2 Συνεισφορά Εργασίας....9
1.3 Διάρθρωση Διπλωματικής Εργασίας....9
2 Σχετική Έρευνα....11
2.1 Field Programmable Gate Array(FPGA)....11
2.2 Περιγραφή Zynq SoC συσκευών....13
2.2.1 Σύστημα Επεξεργασίας του Zynq SoC....15
2.2.2 Προγραμματιζόμενη λογική του Zynq SoC....17
2.3 Διασύνδεση Περιφερειακών Συσκευών με SoC....18
2.3.1 Άμεση Διεπαφή FPGA με κάμερα....18
2.3.2 Συν-επεξεργασία PS-PL για διεπαφή με κάμερα....19
2.4 Πρωτόκολλο USB....19
2.4.1 Σύστημα USB....20
2.4.2 Διεπαφή ULPI....23
2.5 Σχετικές εργασίες διεπαφής καμερών με FPGA....25
3 Μελέτη Συστήματος....31
3.1 Μέθοδοι Υλοποίησης Συστήματος USB....31
3.1.1 Απευθείας σύνδεση σε FPGA....33
3.1.2 USB Soft IP with off-chip PHY....35
3.1.3 Off-chip USB Controller....39
3.1.4 USB Controller on SoC....40
3.1.5 Σύγκριση μεθόδων διεπαφής FPGA με USB περιφερειακά....42
3.2 Μέθοδοι Υλοποίησης Συστήματος HDMI....43
3.2.1 Ελεγκτές HDMI για χρήση σε SoCs....45
3.2.1.1 Xilinx HDMI_TX....47
3.2.1.2 Xylon LogiCVC-ML....48
3.2.1.3 Analog Devices AXI_HDMI_TX....49
3.2.2 Πρόγραμμα Οδήγησης HDMI Συστήματος....51
3.2.3 Τρόποι Προβολής Βίντεο σε Περιβάλλον Linux....52
3.2.3.1 X Server (X11) Direct Connection....53
3.2.3.2 Direct Rendering Manager (DRM) Dumb Buffers....54
3.2.3.3 Linux Framebuffer Device....54
3.3 Αλγόριθμος Αναγνώρισης Ακμών Sobel....54
3.3.1 Αναγνώριση ακμών....55
3.3.2 Τελεστής Ανίχνευσης Ακμών Sobel....55
3.3.3 Διαδικασία Ανίχνευσης Ακμών Sobel....57
3.3.4 Παραδείγματα Εφαρμογής του Αλγορίθμου....57
4 Υλοποίηση Συστήματος....60
4.1 Εργαλεία που χρησιμοποιήθηκαν....60
4.1.1 Vivado Design Suite....60
4.1.2 Vivado HLS....61
4.1.3 Petalinux SDK....63
4.2 Περιγραφή Zynq-7000 SoC Zedboard....67
4.3 Δημιουργία Sobel φίλτρου....68

4.3.1 Διεπαφή....69
4.3.2 Στοιχεία Επεξεργασίας....70
4.3.3 Βελτιστοποιήσεις....70
4.3.4 Ροή Αλγορίθμου....71
4.3.5 Προσομοίωση Κώδικα....72
4.3.6 Αξιοποίηση Πόρων....74
4.4 Αρχιτεκτονική Συστήματος....75
4.4.1 Αρχιτεκτονική Συστήματος Hardware....77
4.4.1.1 Διαμόρφωση Συστήματος Επεξεργασίας (PS)....78
4.4.1.2 Διαμόρφωση Προγραμματιζόμενης Λογικής (PL)....82
4.4.1.3 Σύνθεση Υλικού και εξαγωγή bitstream....85
4.4.1.4 Πόροι Συστήματος....86
4.4.2 Αρχιτεκτονική Συστήματος Software....86
4.4.2.1 Εισαγωγή hardware....88
4.4.2.2 Διαμόρφωση Πυρήνα....89
4.4.2.3 Διαμόρφωση Device Tree....92
4.4.2.4 Δημιουργία Linux Image και File System....94
5 Πειραματική αξιολόγηση συστήματος....96
5.1 Υλοποίηση διεπαφής δύο καμερών....96
5.2 Υλοποίηση φίλτρου Sobel....100
6 Συμπεράσματα και μελλοντική εργασία....104
6.1 Συμπεράσματα....104
6.2 Μελλοντικές Επεκτάσεις....105
7 Βιβλιογραφία....108

Εικόνες

Εικόνα 1: Διάταξη Λογικών *Blocks*. *Πηγή Εικόνας:* [2].....12
Εικόνα 2: Xilinx SoC & MPSoC. *Πηγή Εικόνας:* [6].....14
Εικόνα 3: Zynq Processing System. *Πηγή Εικόνας:* [7].....15
Εικόνα 4: Zynq PS I/O. *Πηγή Εικόνας:* [7].....16
Εικόνα 5: Zynq PL I/O. *Πηγή Εικόνας:* [7].....18
Εικόνα 6: USB3 Vision Camera. *Πηγή Εικόνας:* [13].....19
Εικόνα 7: USB Connector Types. *Πηγή Εικόνας:* [14].....20
Εικόνα 8: USB Pipe Model. *Πηγή Εικόνας:* [16].....21
Εικόνα 9: Καλώδιο USB. *Πηγή Εικόνας:* [17].....23
Εικόνα 10: High-performance FPGAs with their voltage conditions recommended. *Πηγή Εικόνας:* [18].....24
Εικόνα 11: USB ULPI PHY. *Πηγή Εικόνας:* [19].....25
Εικόνα 12: Vita Camera Block Diagram. *Πηγή Εικόνας:* [21].....26
Εικόνα 13: Vita 2000 Image Sensor with FMC *IMAGEON* interfaces Xilinx board. *Πηγή Εικόνας:* [21].....26
Εικόνα 14: AES-FMC-IMAGEON-G. *Πηγή Εικόνας:* [56].....27
Εικόνα 15: Embeded Vision with Pcam 5MP Camera Module. *Πηγή Εικόνας:* [22].....28
Εικόνα 16: OV7670 CMOS Camera Module. *Πηγή Εικόνας:* [24].....28
Εικόνα 17: Zedboard with OV7670 C*MOS* Camera. *Πηγή Εικόνας:* [23].....29
Εικόνα 18: HIKVISION DS-2CD2325FWD-I Camera IP. *Πηγή Εικόνας:* [26].....29
Εικόνα 19: USB Host/Device Detailed View. *Πηγή Εικόνας:* [27].....32
Εικόνα 20: Τυπικό Σύστημα USB.....32
Εικόνα 21: USB wires to XST-3 board. *Πηγή Εικόνας:* [28].....34
Εικόνα 22: FPGA with integrated IP and external USB transceiver.....35
Εικόνα 23: Αναπτυξιακή Πλατφόρμα USB330. *Πηγή Εικόνας:* [29].....35
Εικόνα 24: USB3300 PHY *Block Diagram*. *Πηγή Εικόνας: [29]*.....36
Εικόνα 25: USB Host Software Stack.....37
Εικόνα 26: USB Enumaration Flow. *Πηγή Εικόνας:* [30].....38
Εικόνα 27: External USB Controller.....39
Εικόνα 28: USB Controller Block Diagram. *Πηγή Εικόνας:* [38].....41
Εικόνα 29: Direct Streaming Architecture. *Πηγή Εικόνας:* [39].....43
Εικόνα 30: Framebuffer Streaming Architecture. *Πηγή Εικόνας:* [39].....44
Εικόνα 31: I/O Peripherals System Diagram. *Πηγή Εικόνας:* [7].....45
Εικόνα 32: FMC HDMI 2.0 I/O *M*odule Card. *Πηγή Εικόνας:* [56].....47
Εικόνα 33: Xilinx HDMI_TX System Implementation.....48
Εικόνα 34: logiCVC-ML Architecture. *Πηγή Εικόνας:* [58].....49
Εικόνα 35: Analog AXI_HDMI_TX System Implementation.....50
Εικόνα 36: DRM/KMS Display Pipeline Overview. *Πηγή Εικόνας:* [60].....51
Εικόνα 37: X Window System Architecture.....53
Εικόνα 38: Συνέλιξη πυρήνα για εντοπισμό ακμής στην οριζόντιο *κατεύθυνση*. *Πηγή Εικόνας: [66]* 56
Εικόνα 39: Original and Grayscale *Image*. *Πηγή Εικόνας:* [66].....58
Εικόνα 40: Sobel *on Vertical Changes* (left) and Sobel on Horizontal Changes (right). *Πηγή Εικόνας:*[66].....58

Εικόνα 41: Παρουσίαση διαφοράς στις δύο κατευθύνσεις. *Πηγή Εικόνας:* [66].....59
Εικόνα 42: Result of Sobel Edge Detection. *Πηγή Εικόνας:* [66].....59
Εικόνα 43: Vivado HLS Overview. *Πηγή Εικόνας:* [72].....61
Εικόνα 44: Boot Sequence.....64
Εικόνα 45: Device Tree Example.....65
Εικόνα 46: Zynq-7000 SoC Zedboard.....68
Εικόνα 47: Dataflow Implementation Example. *Πηγή Εικόνας:* [72].....71
Εικόνα 48: Εφαρμογή Φίλτρου.....72
Εικόνα 49: Lena, blur kernel 3x3, no threshold.....73
Εικόνα 50: Original Image Lena.....73
Εικόνα 51: Lena, blur kernel *5*x*5*, no threshold.....73
Εικόνα 52: Lena, blur kernel *5*x*5*, threshold=*70*.....73
Εικόνα 53:Lena, blur kernel *5*x*5*, threshold=25.....73
Εικόνα 54: Lena, blur kernel *5*x*5*, threshold=*40*.....73
Εικόνα 55: Lena Sobel from MATLAB.....74
Εικόνα 56: Εκτιμήσεις Απόδοσης.....75
Εικόνα 57: Εκτιμήσεις Αξιοποίησης Πόρων Υλικού.....75
Εικόνα 58: Dataflow Analysis.....75
Εικόνα 59: Overall Block Design.....76
Εικόνα 60: Hardware Block D*esign*.....77
Εικόνα 61: Hardware Block PS-PL Diagram.....78
Εικόνα 62: Routing Capabilities. *Πηγή Εικόνας:* [38].....80
Εικόνα 63: I/O Peripherals Mapping.....81
Εικόνα 64: HDMI ADV7511 Transmitter Block Design. *Πηγή Εικόνας:* [76].....82
Εικόνα 65: VDMA Block Diagram.....83
Εικόνα 66: AXI_HDMI_TX Block Diagram. *Πηγή Εικόνας:* [78].....84
Εικόνα 67: Resources Utilization.....86
Εικόνα 68: Software Block Design.....87
Εικόνα 69: System Software Architecture.....88
Εικόνα 70: System Config Window.....89
Εικόνα 71: USB Drivers for Host Functionality.....90
Εικόνα 72: USB Video Class Drivers.....91
Εικόνα 73: DRM Support Drivers.....92
Εικόνα 74: USB Device Tree.....93
Εικόνα 75: USB-PHY Device Tree.....93
Εικόνα 76: Ρύθμιση ADV7511 chip μέσω I2C δίαυλου.....94
Εικόνα 77: DRM/KMS Master Node.....94
Εικόνα 78: V4l2 Application Flowchart.....97
Εικόνα 79: Capture State Machine.....98
Εικόνα 80: Colorspace Conversion YUV *and* RGB for HDTV (BT.601).....99
Εικόνα 81: Απεικόνιση εικονοστοιχείων στη μνήμη.....99
Εικόνα 82: Καρέ από την εκτέλεση της εφαρμογής 2 USB webcams: Resolution = 320x240……100
Εικόνα 83: Sobel and VDMA interaction.....101
Εικόνα 84: OpenCV-Sobel Application.....102
Εικόνα 85: Καρέ από την εκτέλεση της εφαρμογής φίλτρου Sobel: Resolution = 640x480.....103
Εικόνα 86: Καρέ από την εφαρμογή φίλτρου Sobel στην FPGA και στον επεξεργαστή ARM.….105

Πίνακες

Table 1: Processing System Inputs/Outputs....16
Table 2: USB Speeds....20
Table 3: USB Device Classes....22
Table 4: Ποιοτική Σύγκριση μεθόδων Υλοποίησης USB....42
Table 5: Διεπαφή HDMI σε διάφορες πλακέτες....46
Table 6: Χαρακτηριστικά HDMI IP cores....50
Table 7: USB webcams....96
Table 8: Αποτελέσματα Φίλτρου Sobel....105

1 Εισαγωγή

1.1 Περιγραφή Εργασίας

Οι συστοιχίες επιτόπια προγραμματιζόμενων πυλών (**F**ield **P**rogrammable **G**ate **A**rrays), γνωστές και ως αναδιατασσόμενη λογική, αποτελούν την ολοένα και περισσότερο προτιμητέα επιλογή, για την εκτέλεση εφαρμογών που αποσκοπούν στην διαχείριση και επεξεργασία εικόνας/βίντεο. Οι τεχνολογίες παράλληλης επεξεργασίας που ενσωματώνει μια τυπική FPGA, δίνει τη δυνατότητα στον σχεδιαστή να δημιουργήσει ένα σύστημα επεξεργασίας βίντεο πραγματικού χρόνου, πετυχαίνοντας πολύ υψηλή ρυθμαπόδοση (throughput) δεδομένων. Αυτό το χαρακτηριστικό επιτρέπει την ταχύτερη απόκτηση και επεξεργασία εικόνας από αισθητήρες υψηλής ανάλυσης και κατ' επέκταση τη πλήρη αξιοποίηση μερικών από των πιο πρόσφατων καμερών που κυκλοφορούν στην αγορά. Η Xilinx, όπως και οι συνεργαζόμενες με αυτήν εταιρίες, προσφέρουν έτοιμες, ολοκληρωμένες λύσεις για τη διεπαφή καμερών με FPGAs. Οι περισσότερες από αυτές τις λύσεις αξιοποιούν κάμερες που υποστηρίζουν πολύ μεγάλη ευκρίνεια αλλά υπερβολικά δαπανηρή αγορά. Για παράδειγμα το Video & Imaging kit που προσφέρει η Xilinx για την οικογένεια Zynq-7000[1] , παρέχει τη δυνατότητα απόκτησης εικόνας από κάμερα πρωτοκόλλου HDMI, προσφέροντας τις κατάλληλες εξωτερικές μονάδες για τη διασύνδεση με μια FPGA. Η πολυδάπανη απόκτηση τέτοιου είδους πακέτων όμως, δημιουργεί την ανάγκη για χρήση φθηνών και αξιόπιστων καμερών. Τέτοιου είδους κάμερες αποτελούν οι USB webcams. Πρόκειται για φορητές κάμερες που ακολουθούν τις προδιαγραφές του πρωτόκολλο USB, προσφέροντας πλέον αναλύσεις έως και 1920x1080, ενώ εκτός από RAW βίντεο, πραγματοποιούν και συμπίεση δεδομένων βίντεο (MJPEG, H.264) κατά τη μετάδοση.

1.2 Συνεισφορά Εργασίας

Στην παρούσα διπλωματική εργασία μελετάται πως USB webcams, με κόστος που δεν υπερβαίνει τα 100€, μπορούν να συνδεθούν με μια FPGA και να προβάλλουν τα βίντεο δεδομένα τους σε μια οθόνη HDMI. Αρχικά αναλύεται το πρωτόκολλο USB ώστε να αναγνωριστούν οι δύο κύριες ενότητες που πρέπει να υλοποιηθούν ώστε να συνδεθεί μια USB webcam σε μια FPGA. Η πρώτη ενότητα αφορά την φυσική σύνδεση μεταξύ της συσκευής USB και της αναπτυξιακής πλακέτας. Η δεύτερη ενότητα, και πιο απαιτητική, αφορά την εφαρμογή του πρωτοκόλλου USB σε ένα System-on-Chip (SoC).

Από άποψη αποτελεσμάτων, το σύστημα έχει τη δυνατότητα να διαχειριστεί σε περιβάλλον Linux έως και δύο USB webcams, ένα φίλτρο υλοποιημένο στην FPGA, και την προβολή βίντεο δεδομένων σε μια οθόνη μέσω HDMI.

1.3 Διάρθρωση Διπλωματικής Εργασίας

Στο **κεφάλαιο 2**, παρουσιάζονται οι τρόποι με τους οποίους μια περιφερειακή συσκευή μπορεί να συνδεθεί σε ένα System-on-Chip (SoC) αλλά και πιο συγκεκριμένα, εργασίες που παρουσιάζουν έτοιμες λύσεις για τη διεπαφή διάφορων τύπων καμερών με μία FPGA. Τέλος γίνεται αναφορά σε

σημεία κλειδιά του πρωτοκόλλου USB, που τονίζουν τα ιδιαίτερα χαρακτηριστικά που έπρεπε να υλοποιηθούν στην παρούσα εργασία.

Στο **κεφάλαιο 3**, περιγράφονται οι πιθανές μέθοδοι με τις οποίες μπορεί να πραγματοποιηθεί η φυσική διασύνδεση USB συσκευών με μια FPGA καθώς και η εφαρμογή του USB πρωτοκόλλου. Επιπλέον περιγράφεται ο τρόπος που θα υλοποιηθεί η προβολή βίντεο μέσω HDMI και ποιες οικογένειες boards αφορά αυτή η λύση. Τέλος περιγράφεται η λειτουργία αναγνώρισης ακμών Sobel που θα υλοποιηθεί στην FPGA.

Στο **κεφάλαιο 4**, περιγράφεται ο σχεδιασμός του συνολικού συστήματος hardware για χρήση USB webcams και μιας οθόνης HDMI, καθώς και η δημιουργία και ένταξη, σε αυτό το σύστημα, ενός φίλτρου Sobel. Τέλος δίδεται περιγραφή της ανάπτυξης ενός συστήματος Linux που δημιουργείται από το προαναφερθέν σύστημα hardware και την κατάλληλη διαμόρφωση που υπέστη ο πυρήνας του.

Στο **κεφάλαιο 5**, περιγράφονται οι 2 εφαρμογές που αναπτύχθηκαν για την επίδειξη λειτουργίας του συστήματος σε περιβάλλον Linux. Η μία εφαρμογή αφορά το φιλτράρισμα (Sobel) του βίντεο μίας USB webcam , υλοποιημένο σε HW και SW μέσω της βιβλιοθήκης OpenCV, ενώ η δεύτερη εφαρμογή πραγματοποιεί παράλληλη προβολή βίντεο από δύο USB webcams κάνοντας χρήση του framework V4L2.

Στο **κεφάλαιο 6**, παρουσιάζονται συμπεράσματα από τα αποτελέσματα των δύο εφαρμογών και σκέψεις για μελλοντικές βελτιώσεις του συστήματος.

2 Σχετική Έρευνα

Σε αυτό το κεφάλαιο γίνεται μια επισκόπηση των κύριων σημείων που μελετήθηκαν για την κατανόηση του θέματος της εργασίας. Στην ενότητα 2.1 δίδεται μια σύντομη περιγραφή της τεχνολογίας FPGA και εξηγείται γιατί η χρήση της μπορεί να επιταχύνει εφαρμογές επεξεργασίας βίντεο αποτελεσματικότερα από πολυπύρηνες CPU. Στην ενότητα 2.2 επιλέγονται οι συσκευές System-on-Chip (SoC) έναντι των απλών FPGA, για την έρευνα της εργασίας και γίνεται ο διαχωρισμός τους σε σύστημα επεξεργασίας (Processing System=PS) και προγραμματιζόμενη λογική (Programmable Logic=PL), ώστε να μελετηθούν τα χαρακτηριστικά τους αποτελεσματικότερα. Στην ενότητα 2.3 παρουσιάζονται οι δύο κύριοι τρόποι κατά τους οποίους μπορεί να πραγματοποιηθεί η διεπαφή μιας συσκευής κάμερας με μία συσκευή SoC και για ποιες περιπτώσεις και πρότυπα διεπαφών είναι κατάλληλος ο κάθε τρόπος. Στην ενότητα 2.4 παρουσιάζεται ο τρόπος λειτουργίας ενός συστήματος USB και γίνεται αναφορά στη διεπαφή ULPI (=UTMI+ low pin interface), μέσω της οποίας πραγματοποιείται η φυσική σύνδεση μιας USB συσκευής. Τέλος στην ενότητα 2.5 παρουσιάζονται εργασίες που υλοποιούν τη διεπαφή μιας συσκευής SoC με τους συχνότερους τύπους καμερών που συναντώνται στην επίσημη βιβλιογραφία. Η απόφαση για ενασχόληση με τις USB webcams παίρνεται λόγω της χαμηλής επιλεξιμότητάς τους ανάμεσα σε αυτούς τους τρόπους.

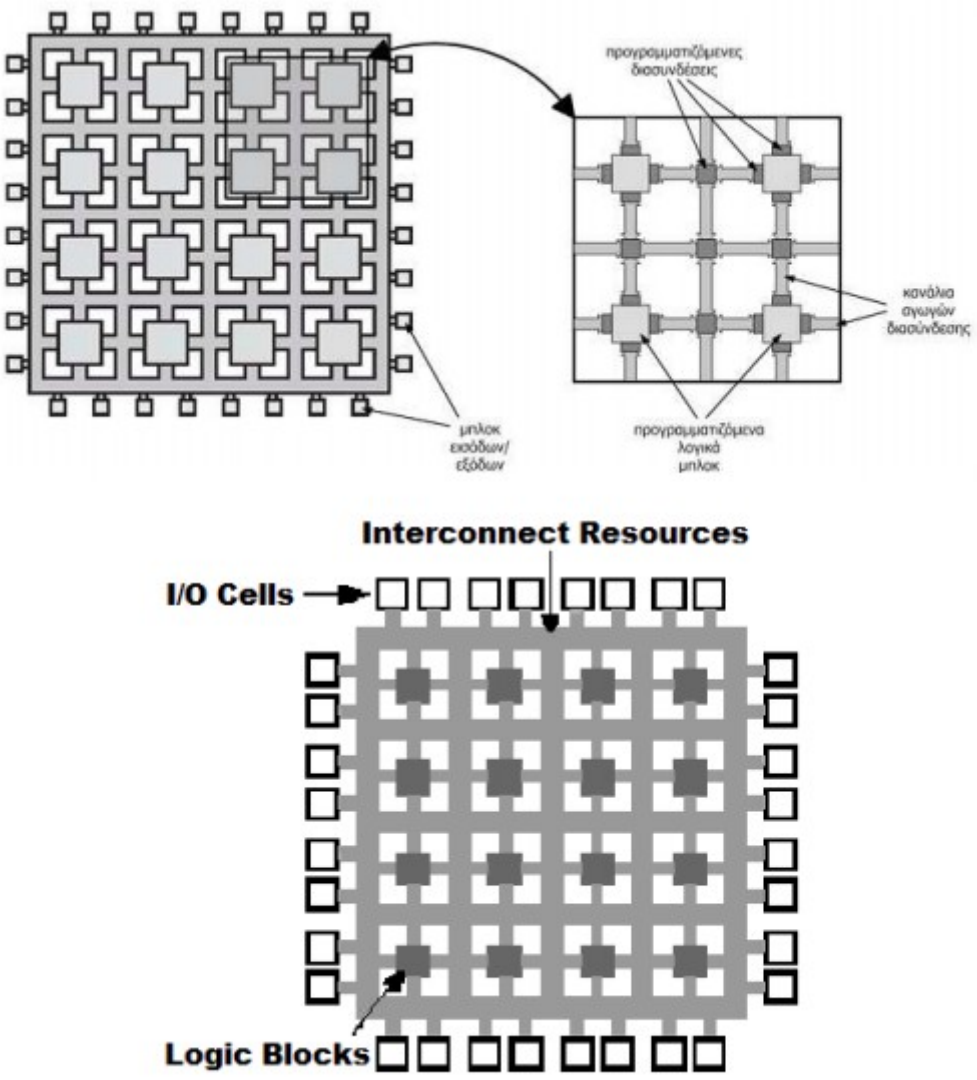
2.1 Field Programmable Gate Array(FPGA)

Στον χώρο των υπολογιστών και ηλεκτρονικής σήμερα, υπάρχουν 2 βασικοί τρόποι για την επεξεργασία πληροφοριών και εκτέλεση υπολογισμών, αυτός του software και αυτός του hardware. Ο υπολογισμός και η επίλυση προβλημάτων σε επίπεδο hardware, όπως τα Application Specific Integrated Circuits(ASICs), προσφέρει εξαιρετική αξιοποίηση πόρων, όπως εξοικονόμηση ενέργειας. Όμως όπως προμηνύει και το ακρωνύμιο, δημιουργούνται με ένα συγκεκριμένο σκοπό και λειτουργούν με τον ίδιο τρόπο καθ' όλη τη διάρκεια ζωής τους. Από την άλλη μεριά, η χρήση software προσφέρει ευελιξία ως προς τον αριθμό των εργασιών και εφαρμογών που μπορούν να εκτελεστούν αλλά η απόδοση είναι κατά πολύ χειρότερη από αυτή των ASICs.

Το 1984 η Xilinx εισήγαγε την ιδέα ενός ολοκληρωμένου κυκλώματος που θα συνδύαζε τα οφέλη των δύο προαναφερθέντων προσεγγίσεων, software και hardware, με αποτέλεσμα τέσσερα χρόνια αργότερα να πάρει την ονομασία με την οποία σήμερα είναι ευρέως γνωστό ως FPGA.

Ο όρος FPGA προέρχεται από τα αρχικά της ονομασίας Field Programmable Gate Array = ”συστοιχίες προγραμματιζόμενων πυλών“ και είναι ένας τύπος ολοκληρωμένου κυκλώματος ημιαγωγών (IC). Αποτελεί μια συστοιχία από προσεκτικά σχεδιασμένα και διασυνδεδεμένα ψηφιακά υποκυκλώματα που υλοποιούν αποτελεσματικά κοινές λειτουργίες, προσφέροντας παράλληλα πολύ υψηλά επίπεδα ευελιξίας. Τα ψηφιακά υποκυκλώματα ονομάζονται προγραμματιζόμενα blocks λογικής (logic blocks) και αποτελούν τον πυρήνα των δυνατοτήτων προγραμματισμού/λογικής του FPGA. Τα blocks λογικής περιλαμβάνουν πίνακες αναζήτησης (LUT), στοιχεία αποθήκευσης (flip-flops ή register) και πολυπλέκτες που επιτρέπουν στο block να εκτελεί λειτουργίες Boolean, αποθήκευσης δεδομένων και αριθμητικής.[2] Επειδή πρέπει να αλληλεπιδρούν μεταξύ τους αλλά και με άλλα

εξωτερικά κυκλώματα, χρησιμοποιεί προγραμματιζόμενες διασυνδέσεις και blocks εισόδου/εξόδου (I/O).



Εικόνα 1: Διάταξη Λογικών Blocks. Πηγή Εικόνας: [2]

Δεδομένου ότι κάθε block λογικής μπορεί να είναι υπολογιστικά ανεξάρτητο από τα υπόλοιπα, τα FPGA είναι εγγενώς παράλληλα. Αυτό έχει ως αποτέλεσμα τη σημαντική επιτάχυνση του χρόνου εκτέλεσης ενός προγράμματος διατηρώντας την επαναπρογραμματιζόμενη ευελιξία του λογισμικού με σχετικά χαμηλό κόστος. Έτσι τα FPGA μπορούν να θεωρηθούν κατάλληλα για την επεξεργασία εικόνας και βίντεο, ειδικά αν ληφθεί υπόψιν η επικείμενη εντατικοποίηση χρήσης αισθητήρων που υποστηρίζουν ultra-high definition formats, καθιστώντας έτσι αναγκαία την επεξεργασία τεράστιων ποσοτήτων δεδομένων σε συγκεκριμένο χρονικό διάστημα επιπρόσθετα με τη ικανότητα πειραματισμού με αλγορίθμους προς εφαρμογή επεξεργασίας. Αυτό μεταφράζεται σε χρήση μεγάλης ποσότητας υπολογιστικών πόρων, ανάγκη για καλό χρόνο επεξεργασίας και σημαντική κατανάλωση επεξεργαστικής ισχύος. Η κατανάλωση ισχύος ενός συστήματος σχετίζεται άμεσα με τη συχνότητα

του ρολογιού, οπότε ένα πιο αργό ρολόι οδηγεί σε σημαντικά χαμηλότερη σχεδίαση ισχύος.[3] Προκύπτει έτσι ότι η επεξεργασία εικόνας σε πραγματικό χρόνο είναι δύσκολο να επιτευχθεί με αποδεκτά αποτελέσματα χρησιμοποιώντας έναν απλό συμβατικό σειριακό επεξεργαστή, όπως μια απλή CPU. Αντιθέτως η παράλληλη φύση των FPGA επιτρέπει χαμηλότερη συχνότητα ρολογιού σε σχέση με τον σειριακό επεξεργαστή.

Table 1: Χαρακτηριστικά FPGA σε σχέση με GPUs/CPUs[4]

Πλεονεκτήματα	**Μειονεκτήματα**
Μπορούν να επαναπρογραμματιστούν στις επιθυμητές απαιτήσεις του σχεδιαστή μετά την κατασκευή τους.	Μεγάλη διάρκεια χρόνου ανάπτυξης
Μπορεί να επαναπρογραμματιστεί ένα μόνο μέρος του chip, ενώ οι υπόλοιπες περιοχές συνεχίζουν να λειτουργούν	Η ανάπτυξη κώδικα απαιτεί βαθιά κατανόηση του hardware και εξειδικευμένες τεχνικές δεξιότητες
Είναι οικονομικές	
Χρησιμοποιούν πραγματικό παραλληλισμό υλικού (hardware parallelism).	
Μεγάλη ταχύτητα επεξεργασίας σε σχέση με άλλους hardware accelerators[5]	
Δυνατότητα παραμετροποίησης της FPGA για βέλτιστη εφαρμογή αλγορίθμων	
Υψηλό throughput δεδομένων	

2.2 Περιγραφή Zynq SoC συσκευών

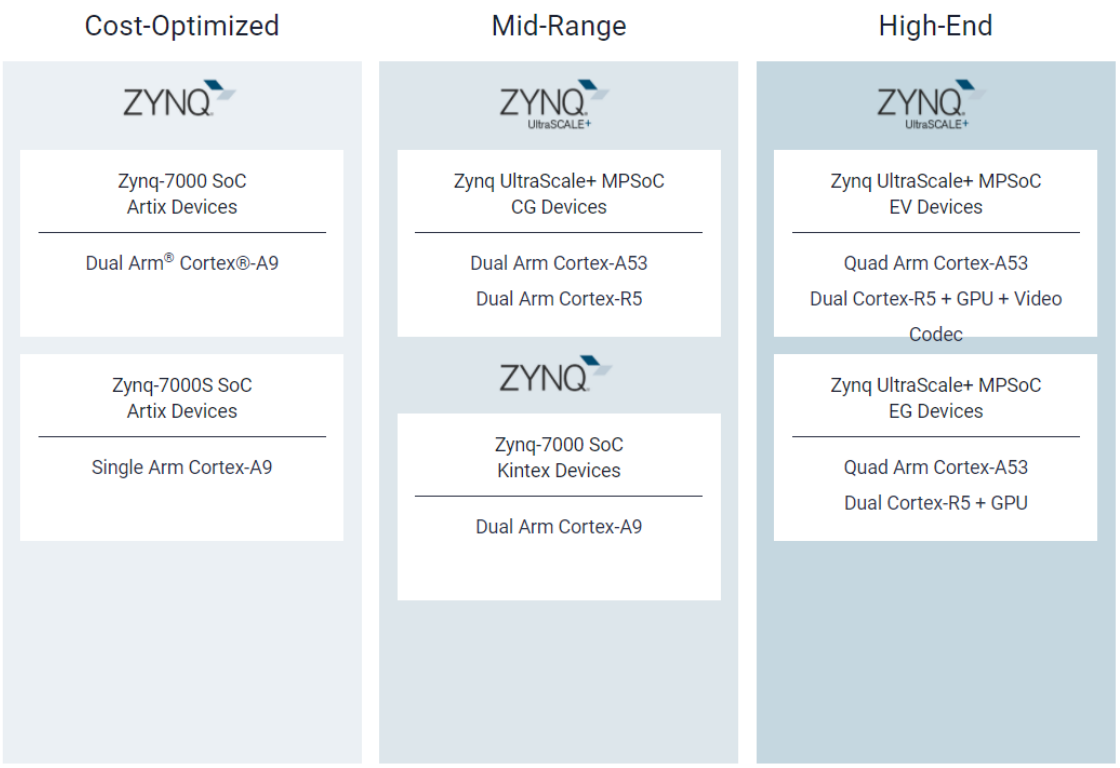
Τα τελευταία χρόνια έχει επικρατήσει η χρήση ενός ολοκληρωμένου κυκλώματος, το οποίο “παντρεύει” επεξεργαστές αρχιτεκτονικής ARM με μια παραδοσιακή FPGA. Αυτές οι συσκευές ονομάζονται System-on-Chip (SoC) ή Multi-Processor System-on-Chip (MPSoC). Αποτελούν την πιο συμφέρουσα επιλογή για υλοποίηση μοντέρνων συστημάτων προσφέροντας ποικίλες λειτουργίες και γιαυτό το λόγο η παρούσα εργασία εστιάζει σε αυτές τις συσκευές αντί των μεμονωμένων FPGAs.

Η Xilinx προσφέρει την οικογένεια Zynq-7000 για συσκευές SoC, κατηγοριοποιώντας τες σε Cost-Optimized και Mid-Range. Στην Cost-Optimized κατηγορία παρέχεται η επιλογή συνδυασμού ενός single ή dual ARM επεξεργαστή (Cortex-A9) με μια συσκευή FPGA της οικογένειας Artix-7. Στη Mid-Range κατηγορία παρέχεται η επιλογή μιας συσκευής που συνδυάζει ένα dual ARM επεξεργαστή με μια ισχυρότερη συσκευή FPGA, αυτή της οικογένειας Kintex-7. Τα συστήματα που διαθέτουν single ARM Cortex-A9 επεξεργαστή μπορούν να φτάσουν ταχύτητες έως και 766 MHz, ενώ αυτά που διαθέτουν dual ARM Cortex-A9 επεξεργαστές μπορούν σε ορισμένα boards (π.χ ZedBoard, Zybo Z7) να φτάσουν ταχύτητες εώς 866 MHz ή ακόμα και 1 GHZ. Ο ARM Cortex-A9

χαρακτηρίζεται ως application processor και βασίζεται στην αρχιτεκτονική ARMv7-A που χρησιμοποιεί εντολές 32-bit.

Από την άλλη η οικογένεια Zynq Ultrascale+ για συσκευές MPSoC, κατηγοριοποιείται σε Mid-Range και High-End. Στη Mid-Range κατηγορία ανήκουν οι συσκευές ***CG*** που ενσωματώνουν dual ARM Cortex-A53 και dual ARM Cortex-R5 επεξεργαστές. Στην κατηγορία High-End ανήκουν οι συσκευές ***EG*** που χρησιμοποιούν επιπλέον quad ARM Cortex-A53 επεξεργαστή αλλά και μια μονάδα επεξεργασίας γραφικών (GPU). Τέλος προσθέτοντας έναν ενσωματωμένο κωδικοποιητή βίντεο H.264/ H.265 στις συσκευές *EG*, προκύπτει η τελευταία και πιο δυνατή οικογένεια συσκευών σε αυτή τη κατηγορία, η **E*V***. Ο επεξεργαστής ARM Cortex-A53 που χρησιμοποιείται σε αυτές τις συσκευές, εντάσσεται στην κατηγορία application processor ενώ ο ARM Cortex-R5 στην κατηγορία real-time processor. Η αρχιτεκτονική στην οποία βασίζεται ο επεξεργαστής ARM Cortex-A53 είναι η ARMv8-A, μεταγενέστερη της αρχιτεκτονικής ARMv7-A των συσκευών SoC. Η ARMv8-A εισάγει την ιδέα της επιλογής μεταξύ 64-bit αρχιτεκτονικής, που ονομάζεται AArch64, και της 32-bit αρχιτεκτονικής, που ονομάζεται AArch32. Η ARMv8-A αποτελεί ουσιαστικά μια προέκταση της ARMv7-A αφού με την χρήση της AArch32 ο Cortex-A53 (MPSoC) έχει τη δυνατότητα να μιμηθεί τον Cortex-A9 (SoC).

Η Xilinx κατηγοριοποιεί τις οικογένειες Zynq Ultrascale+ MPSoC και Zynq-7000 SoC [6] ως Cost-Optimized, Mid-range και High-end.

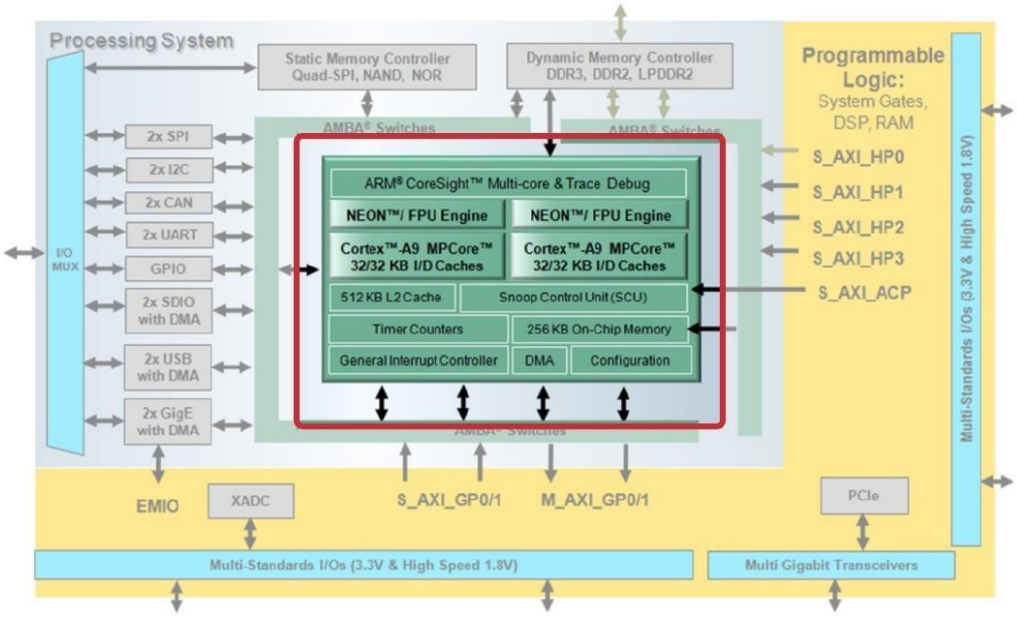


Εικόνα 2: Xilinx SoC & MPSoC. Πηγή Εικόνας: [6]

Η παρούσα διπλωματική δεν εξετάζει την υλοποίηση της διεπαφής USB webcam και HDMI εξόδου για την οικογένεια Zynq Ultrascale+ MPSoC παρά μόνο για την οικογένεια Zynq-7000 SoC. Παρόλα αυτά όπως παρουσιάστηκε και με τους επεξεργαστές, οι δύο οικογένειες παρουσιάζουν ομοιότητες και μέσω αναφορών που θα ακολουθήσουν παρακάτω θα μπορεί κάποιος να μεταφέρει την υλοποίηση από την μια στην άλλη. Η περιγραφή για τα επόμενα υποκεφάλαια αναφέρεται στην οικογένεια Zynq-SoC.[7]

2.2.1 Σύστημα Επεξεργασίας του Zynq SoC

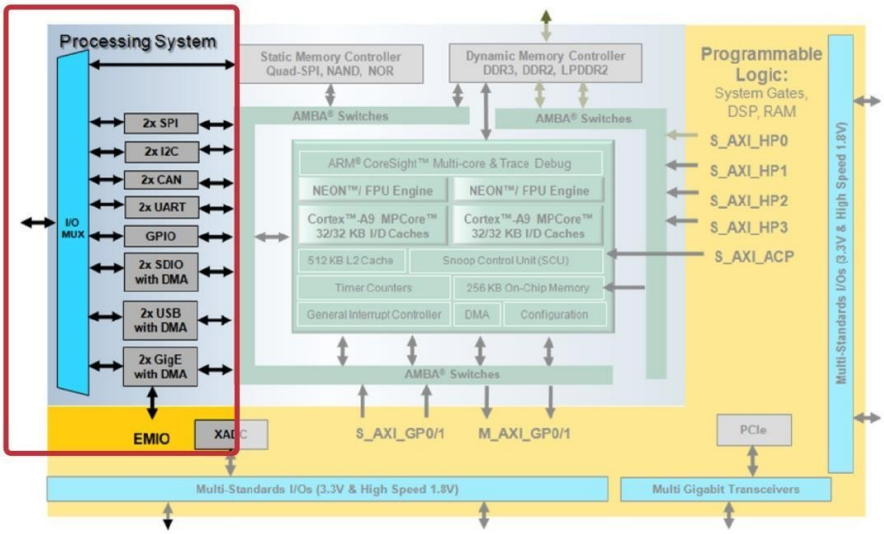
Το σύστημα επεξεργασίας (processing system = **PS**) μιας Zynq-7000 συσκευής περιλαμβάνει single ή dual ARM Cortex-A9 επεξεργαστή. Πρόκειται για ένα "hard" επεξεργαστή υπό την έννοια της φυσικής ύπαρξης ενός ειδικού και βελτιστοποιημένου στοιχείου πυριτίου στη συσκευή. Παρόλα αυτά το Zynq PS δεν περιλαμβάνει μόνο τον επεξεργαστή ARM αλλά και ένα σύνολο από επεξεργαστικούς πόρους που σχηματίζουν το Application Processing Unit (APU), καθώς και περαιτέρω περιφερειακές διεπαφές, μνήμη cache, διεπαφές μνήμης, διασυνδέσεις και κύκλωμα παραγωγής ρολογιού.



Εικόνα 3: Zynq Processing System. Πηγή Εικόνας: [7]

Το σύστημα επεξεργασίας (PS) διαθέτει μια ποικιλία διεπαφών τόσο μεταξύ του PS και του PL αλλά όσο και μεταξύ του PS και εξωτερικών στοιχείων. Η επικοινωνία με τις εξωτερικές διεπαφές πραγματοποιείται μέσω των Multiplexed Input/Output (MIO) τα οποία παρέχουν 54 pins ευέλικτης συνδεσιμότητας, το οποίο σημαίνει ότι η αντιστοίχιση μεταξύ περιφερειακών και των pins μπορεί

να οριστεί όπως απαιτείται από τον σχεδιαστή. Επιπλέον, συγκεκριμένες διασυνδέσεις μπορούν να πραγματοποιηθούν μέσω του Extented Multiplexed I/O (EMIO) το οποίο δεν αντιπροσωπεύει ένα άμεσο μονοπάτι από το PS στις εξωτερικές διεπαφές συσκευής, αλλά περνάει και μοιράζεται τους I/O πόρους της προγραμματιζόμενης λογικής (PL). Τα EMIO μπορούν να χρησιμοποιηθούν ως μέθοδο διεπαφής του PS με IP blocks υλοποιημένα στην PL μεριά, ή όταν τα 54 MIO pins δεν επαρκούν και απαιτείται επέκταση αυτών. Αξίζει να σημειωθεί ότι η διεπαφή μέσω EMIO δεν ισχύει για όλα τα περιφερειακά(π.χ USB). Τα διαθέσιμα I/O περιλαμβάνουν τυπικές διεπαφές επικοινωνίας και I/O γενικής χρήσης (GPIO= General Purpose I/O) τα οποία χρησιμοποιούνται για σκοπούς που περιλαμβάνουν buttons, switches και leds.



Εικόνα 4: Zynq PS I/O. Πηγή Εικόνας: [7]

Ακολουθεί μια σύντομη περιγραφή για τα I/O στην μεριά του PS[8]:

Table 1: Processing System Inputs/Outputs

Διεπαφή I/O	**Περιγραφή**
SPI	Serial Peripheral Interface *Πρότυπο για σειριακές επικοινωνίες που βασίζεται σε διεπαφές με 4-pins. Υποστηρίζει λειτουργίες master και slave.*
I2C	I^2C bus *Συμβατό με την προδιαγραφή I2C bus, version 2. Υποστηρίζει λειτουργίες master και slave.*
CAN	Controller Area Network *Ελεγκτής διεπαφής διαύλου συμβατός με ISO 118980-1, πρότυπα CAN 2.0A και CAN 2.0B.*
UART	Universal Asynchronous Receiver Transmitter *Διεπαφή μόντεμ χαμηλού ρυθμού δεδομένων για σειριακή επικοινωνία. Χρησιμο-*

	ποιείται για συνδέσεις τερματικού σε κεντρικό υπολογιστή.
GPIO	General Purpose Input/Output *Υπάρχουν 4 banks GPIO των 32-bits το καθένα.*
SD	*Χρησιμοποιείται για την διεπαφή με κάρτες μνήμης SD.*
USB	Universal Serial Bus *Συμβατό με την προδιαγραφή USB 2.0 και μπορεί να χρησιμοποιηθεί ως host, device ή otg (η λειτουργία OTG σημαίνει ότι μπορεί να αλλάξει μεταξύ λειτουργιών host και device).*
GigE	Ethernet Περιφερειακό Etherne MAC που υποστηρίζει λειτουργίες 10Mbps, 100 Mbps και 1 Gbps

2.2.2 Προγραμματιζόμενη λογική του Zynq SoC

Η προγραμματιζόμενη λογική (Programmable logic = **PL**) αποτελεί στην ουσία την FPGA για τις συσκευές Zynq-7000 SoC και βασίζεται είτε στην οικογένεια FPGA Artix-7 είτε στην οικογένεια FPGA Kintex-7. Για την επικοινωνία με εξωτερικές συσκευές εδώ, μπορεί να γίνει χρήση των γενικής χρήσης I/O block (IOB) στην PL, τα οποία αναφέρονται ως SelectiveIO Resources και είναι οργανωμένα σε banks των 50 IOBs το καθένα. Κάθε IOB πραγματοποιεί τη φυσική σύνδεση με το έξω κόσμο για ένα σήμα εισόδου ή ένα σήμα εξόδου. Τα I/O banks κατηγοριοποιούνται ως High-Performance (HP) ή High-Range (HR) και περιορίζονται σε τάσεις εξόδου 1.8V και 3.3V αντίστοιχα.

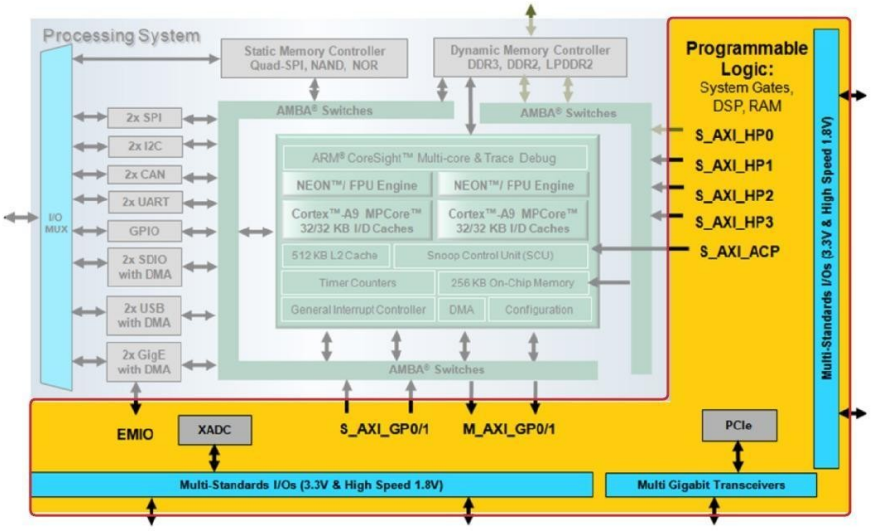
Πιο εξειδικευμένες συσκευές περιλαμβάνουν GTX transceivers, block διεπαφής επικοινωνιών υψηλής ταχύτητας στην προγραμματιζόμενη λογική. Πρόκειται για block πυριτίου (Hard IP block) που είναι ικάνα να υποστηρίξουν μεγάλο αριθμό τυπικών διεπαφών συμπεριλαμβανομένων των PCI Express, Serial RapidIO, SCSI και SATA. Η υλοποίηση των PCIE απαιτεί δέυτερο hard IP block επιπρόσθετα του GTX transceiver.

Επιπλέον εξωτερικές διεπαφές με την PL:

Analogue to Digital Conversion: Η FPGA περιλαμβάνει ένα hard IP block (XADC) που αποτελεί έναν 12-bit αναλογικό-σε-ψηφιακό μετατροπέα. Παρέχει μια αναλογική διεπαφή γενικής χρήσης και υψηλής ακρίβειας για μια σειρά από ποικίλες εφαρμογές.

Clocks: Η FPGA είναι ικανή να παράξει μόνη της δικό της σήμα ρολογιού ή να λάβει 4 ξεχωριστές εισόδους ρολογιού από το PS.

Programming and Debug: Ένα σύνολο JTAG θυρών παρέχονται στην PL μεριά για τη διευκόλυνση της διαμόρφωσης και του εντοπισμού σφαλμάτων.



Εικόνα 5: Zynq PL I/O. Πηγή Εικόνας: [7]

2.3 Διασύνδεση Περιφερειακών Συσκευών με SoC

Όταν αναπτύσσεται ένα ενσωματωμένο σύστημα το οποίο θα πρέπει να συνδέεται με περιφερειακές συσκευές, όπως μια κάμερα, θα πρέπει ο σχεδιαστής να αποφασίσει την αρχιτεκτονική που θα ακολουθήσει στη σχεδίασή του από νωρίς. Αυτό εξαρτάται από τον τύπο κάμερας καθώς και από το ολοκληρωμένο κύκλωμα που διαθέτει. Οι δύο βασικές αρχιτεκτονικές που μπορεί να ακολουθήσει συνοψίζονται στα επόμενα υποκεφάλαια.[9]

2.3.1 Άμεση Διεπαφή FPGA με κάμερα

Σε αυτή την αρχιτεκτονική πραγματοποιείται σύνδεση μιας κάμερας απευθείας στα pins της FPGA και με αυτόν τον τρόπο τα εικονοστοιχεία μεταφέρονται απευθείας στην FPGA όπως αυτά στέλνονται από την κάμερα. Αυτή η αρχιτεκτονική χρησιμοποιείται συνήθως με κάμερες Camera Link επειδή η λογική λήψης από την κάμερα υλοποιείται ευκολότερα χρησιμοποιώντας το ψηφιακό κύκλωμα στην FPGA. Με αυτήν την αρχιτεκτονική παρουσιάζονται δύο κύρια οφέλη. Πρώτον, σε ένα σύστημα που περιλαμβάνει προγραμματιζόμενη λογική μαζί με σύστημα επεξεργασίας δίνεται η δυνατότητα, όπως και με την αρχιτεκτονική συν-επεξεργασίας, για μετακίνηση μέρους του φόρτου εργασίας από την CPU στο FPGA, εκτελώντας λειτουργίες προ-επεξεργασίας στην FPGA. Για παράδειγμα, η FPGA μπορεί να χρησιμοποιηθεί για λειτουργίες προ-επεξεργασίας υψηλής ταχύτητας, όπως φιλτράρισμα ή εφαρμογή κατωφλίου πριν από την αποστολή εικονοστοιχείων στη CPU. Αυτό μειώνει επίσης την ποσότητα δεδομένων που πρέπει να επεξεργαστεί η CPU, επειδή εφαρμόζει λογική μόνο για τη λήψη των pixel από περιοχές ενδιαφέροντος, γεγονός που αυξάνει τη συνολική απόδοση του συστήματος. Το δεύτερο πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι επιτρέπει την πραγματοποίηση λειτουργιών ελέγχου υψηλής ταχύτητας, απευθείας στο FPGA χωρίς να

γίνει χρήση της CPU. Τα FPGA είναι ιδανικά για εφαρμογές ελέγχου [10], διότι μπορούν να πετύχουν εξαιρετικά γρήγορο ρυθμό εκτέλεσης ντετερμινιστικών βρόχων.

2.3.2 Συν-επεξεργασία PS-PL για διεπαφή με κάμερα

Με τον όρο "συν-επεξεργασία" ορίζεται η αρχιτεκτονική σχεδίασης κατά την οποία η προγραμματιζόμενη μεριά (PL) "συνεργάζεται" με τον σύστημα επεξεργασίας (PS) ώστε να μοιραστούν το φορτίο επεξεργασίας. Αυτή η λογική χρησιμοποιείται συχνά με κάμερες που ακολουθούν το πρότυπο διεπαφής GigE Vision ή USB3 Vision. Το GigE Vision[11] είναι ένα βιομηχανικό πρότυπο διεπαφής κάμερας που αναπτύχθηκε για μετάδοση video-stream υψηλής ταχύτητας σε πραγματικό χρόνο μέσω του δικτύου Ethernet. Θεωρείται βέλτιστο για ταχύτητες μετάδοσης από 1 εώς 10 Gigabit Ethernet. Πρόκειται για ένα "κλειστό" πρότυπο καθιστώντας έτσι αναγκαία την απόκτηση ειδικής άδειας για χρήση και ανάπτυξη οδηγών κάμερας. Το USB3 Vision [12] είναι ένα πρότυπο διεπαφής που αφορά βιομηχανικές κάμερες. Πρόκειται για μια προδιαγραφή υλοποιημένη πάνω από το πρότυπο USB με ιδιαίτερη έμφαση να δίνεται στην υποστήριξη καμερών υψηλής απόδοσης βασισμένες στη προδιαγραφή USB 3.0.[13]

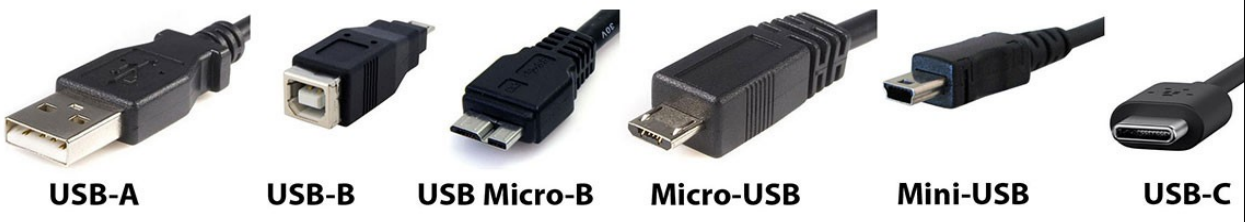
Εικόνα 6: USB3 Vision Camera. Πηγή Εικόνας: [13]

Αυτές οι κάμερες χρησιμοποιούνται σε αυτή τη περίπτωση διότι η λογική λήψης εικόνας από αυτές εφαρμόζεται καλύτερα κάνοντας χρήση του επεξεργαστή. Η λήψη της εικόνας πραγματοποιείται μέσω του επεξεργαστή και στην συνέχεια αποστέλλεται στην FPGA μέσω άμεσης πρόσβασης στη μνήμη (DMA=Direct Memory Access) έτσι ώστε η FPGA να υλοποιήσει τεχνικές επεξεργασίας, όπως φιλτράρισμα. Στην συνέχεια η εικόνα μπορεί να αποσταλεί πίσω στην μεριά του συστήματος επεξεργασίας ώστε να εκτελεστούν πιο προηγμένες λειτουργίες που είναι δύσκολο να εφαρμοστούν μέσω υλικού στην FPGA. Αυτή τη διαδικασία επιτρέπει στο PS να αφιερώσει πόρους σε άλλες διαδικασίες όπως τον έλεγχο κίνησης ή την επικοινωνία δικτύων.

2.4 Πρωτόκολλο USB

Ο Ενιαίος Σειριακός Δίαυλος ή όπως είναι περισσότερο γνωστός Universal Serial Bus (USB), είναι ένα τμηματικό πρότυπο διαύλου για τη διασύνδεση περιφερειακών μονάδων με έναν κεντρικό υπο-

λογιστή. Υπάρχει μια μεγάλη ποικιλία USB υλικού, συμπεριλαμβανομένων πολλών διαφορετικών βυσμάτων [14] εκ των οποίων το USB-C είναι το πιο πρόσφατο, όπως φαίνεται στην Εικόνα 7.



Εικόνα 7: USB Connector Types. Πηγή Εικόνας: [14]

Από το 1996, όταν και δημιουργήθηκε, έχουν υπάρξει συνολικά τέσσερις USB προδιαγραφές. Η αρχική USB 1.0 περιελάμβανε μόνο δύο ταχύτητες: Low-Speed (LS) και Full-Speed (FS). Προορίζονταν για φθηνές συσκευές με χαμηλό ρυθμό δεδομένων όπως ποντίκια, πληκτρολόγια και εκτυπωτές. Στην συνέχεια ακολούθησε ή προδιαγραφή USB 2.0 με την οποία επήλθε μεγάλη αλλαγή αφού πλέον υποστηριζόντουσαν High-Speed (HS) ταχύτητες. Με αυτές τις ταχύτητες δίνεται η δυνατότητα για διεπαφή με συσκευές που απαιτούν υψηλό ρυθμό δεδομένων όπως συσκευές ήχου ή βίντεο. Το USB 2.0 είναι συμβατό με τις εκδόσεις USB 1.0. Ακολούθησαν οι εκδόσεις USB3.0, με Super Speed ταχύτητες, και η πολύ πρόσφατη (2019) USB4, οι οποίες χρησιμοποιούν τον μικρότερο τύπο βύσματος USB-C. Ακολουθεί πίνακας με αναλυτικές ταχύτητες των τύπων που προαναφέρθηκαν.

Table 2: USB Speeds

Type	Data Rate
USB 1.1 (Low Speed)	1.5 Mbits/sec
USB 1.1 (Full Speed)	12 Mbits/sec
USB 2.0 (High Speed)	480 Mbits/sec
USB 3.0 (Super Speed)	5 Gbits/sec
USB 3.1 (Super Speed+)	10 Gbits/sec
USB 3.2 (Super Speed+ dual-lane)	20 Gbits/sec
USB4	40 Gbits/sec

2.4.1 Σύστημα USB

Ένα τυπικό σύστημα USB αποτελείται από 3 κύρια μέρη, την υποδοχή (host) USB που ενεργεί ως master, μία ή περισσότερες συσκευές USB που ενεργούν ως slaves και το φυσικό δίαυλο μέσω του οποίου συνδέεται ο host με τις συσκευές, συνήθως ένα καλώδιο USB. Ο USB host διακρίνεται σε επίπεδο υλικού (hardware) ως USB ελεγκτής υποδοχής και σε επίπεδο λογισμικού (software) ως οδηγός συσκευής USB.[15]

Το επίπεδο υλικού USB είναι υπεύθυνο για:

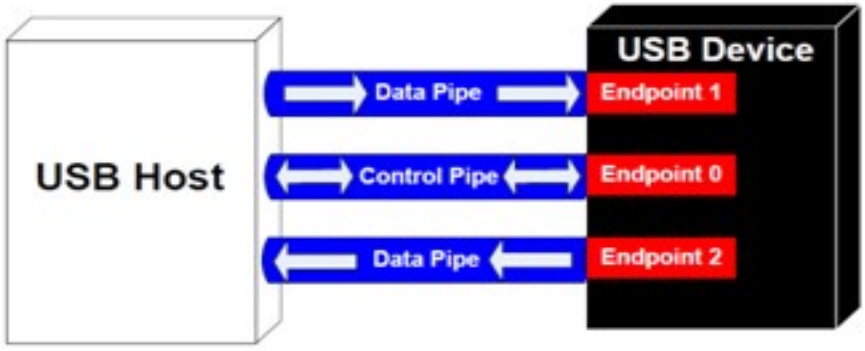
- Ανίχνευση προσάρτησης ή αφαίρεση USB συσκευής από το σύστημα

- Παρακολούθηση της κατάσταστης (status) της USB συσκευής
- Παροχή ισχύος σε συνδεδεμένες συσκευές USB
- Διαχείριση ελέγχου και ροής δεδομένων μεταξύ USB host και USB συσκευής
- Παρακολούθηση δραστηριότητας στο δίαυλο

Το επίπεδο λογισμικού USB είναι υπεύθυνο για:

- Χειρισμός των συσκευών USB και της συνδεσιμότητας τους
- Απαρίθμηση και διαμόρφωση συσκευών USB
- Φόρτωση κατάλληλων προγραμμάτων οδήγησης συσκευών
- Διαχείριση της ισχύος και του εύρους ζώνης του διαύλου επικοινωνίας
- Διαχείριση της μεταφοράς δεδομένων μεταξύ υλικού και λογισμικού

Οι USB συσκευές είναι περιφερειακά που χρησιμοποιούν το πρωτόκολλο USB για αμφίδρομη επικοινωνία με τον host.[16] Η κύρια ευθύνη τους είναι να παρέχουν στον χρήστη συγκεκριμένες λειτουργίες, όπως αυτή του ποντικιού,του πληκτρολογίου ή της βιντεοκάμερας. Η επικοινωνία των περιφερειακών USB είναι βασισμένη πάνω σε κανάλια (pipes). Τα κανάλια αυτά, είναι οι συνδέσεις από την υποδοχή που καταλήγουν στη συσκευή και ονομάζονται άκρα (endpoints).



Εικόνα 8: USB Pipe Model. Πηγή Εικόνας: [16]

Υπάρχουν 2 κατηγορίες καναλιών σε ένα σύστημα USB, κανάλια ελέγχου (Control Pipe) και κανάλια δεδομένων (Data Pipe). Τα κανάλια δεδομένων μπορούν να διαχωριστούν σε 3 διαφορετικούς τύπους μεταφοράς:

- **Isochronous transfers:** Αφορά τις περιπτώσεις όπου απαιτείται εγγυημένος ρυθμός παράδοσης των δεδομένων. Ο εγγυημένος ρυθμός επιτυγχάνεται χάρη στην εγγυημένη καθυστέρηση, εύρος ζώνης του διαύλου και έλλειψη διόρθωσης σφαλμάτων. Με αυτό το τρόπο δεν υπάρχει διακοπή της σύνδεσης ενώ τα πακέτα που περιέχουν σφάλματα αποστέλλονται ξανά. Σε αυτή τη κατηγορία ανήκουν βίντεο ή ήχος πραγματικού χρόνου.

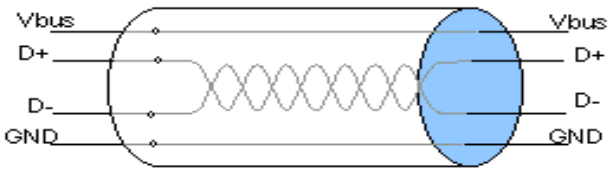
- **Interrupt transfers:** Αφορά τις περιπτώσεις όπου απαιτούνται εγγυημένες γρήγορες αποκρίσεις όπως συσκευές κατάδειξης,ποντίκια και πληκτρολόγια.
- **Bulk transfers:** Χρησιμοποιείται για μεγάλες σποραδικές μεταφορές δεδομένων όπου χρησιμοποιείται όλο το διαθέσιμο εύρος ζώνης USB χωρίς εγγύηση για τη ταχύτητα μεταφοράς ή καθυστέρηση. Σε αυτή τη κατηγορία ανήκουν οι μεταφορές αρχείων.

Όταν μια νέα συσκευή USB συνδέεται με μια υποδοχή (host), αρχίζει η διαδικασία απαρίθμησης USB. Απαρίθμηση ορίζεται ως η διαδικασία κατά την οποία μεταφέρονται πληροφορίες ανάμεσα σε συσκευή και υποδοχή συμπεριλαμβανομένης της ανάθεσης μιας διεύθυνσης στη συσκευή, ανάγνωσης των descriptors (δομές δεδομένων που περιγράφουν τη συσκευή) και φόρτωση των οδηγών της συσκευής. Ένας κωδικός κλάσης στέλνεται από τη συσκευή στην υποδοχή ορίζοντας με αυτόν τον τρόπο την λειτουργικότητα της συσκευής εξαρτώμενη από το είδος του descriptor (device, interface, both) που θα τοποθετηθεί.[Table 3] Όταν ολοκληρωθεί αυτή η διαδικασία η συσκευή είναι έτοιμη να αποστείλει δεδομένα στην υποδοχή. Ο ελεγκτής υποδοχών καθορίζει το δίαυλο για την επικοινωνία και συνήθως έχει κυκλική μορφή (round-robin). Με αυτόν τον τρόπο καμία συσκευή δεν μπορεί να μεταφέρει δεδομένα χωρίς να έχει υπάρξει αίτημα από την υποδοχή πρώτα.

Table 3: USB Device Classes

Class	Usage	Description	Examples
01h	Interface	Audio	Speaker, microphone
02h	Both	Communications and CDC Control	Modem, Ethernet adapter
03h	Interface	Human Interface Device	Keyboard, mouse
07h	Interface	Printer	Laser printer, inkjet printer, CNC printer
08h	Interface	Mass Storage	USB flash drive, memory card reader, digital camera
09h	Device	USB hub	Full bandwidth hub
0Eh	Interface	Video	Webcam

Ένα τυπικό καλώδιο USB το οποίο συνδέει μία υποδοχή με μία συσκευή περιέχει συνολικά 4 καλώδια.[17] Δύο απ' αυτά είναι για την ισχύ (Vbus και γείωση) ενώ υπάρχει και ένα συνεστραμμένο ζεύγος καλωδίων για τη μεταφορά δεδομένων (D+ και D-).Το καλώδιο Vbus φέρει ονομαστική τροφοδοσία 5V, η οποία μπορεί να χρησιμοποιηθεί από μια συσκευή για τροφοδοσία.



Εικόνα 9: Καλώδιο USB. Πηγή Εικόνας: [17]

Διεπαφή USB Ελεγκτή Υποδοχής:

Ένας ελεγκτής υποδοχής παρέχει μια διεπαφή για τη μεταφορά ροών δεδομένων μεταξύ υποδοχέα και συσκευής. Σε ένα σύστημα μπορεί να υπάρχει μόνο ένας υποδοχέας αλλά πολλοί ελεγκτές με διαφορετικούς τύπους διεπαφής για την επικοινωνία του ελεγκτή υποδοχής και του οδηγού συσκευής USB. Οι 4 τύποι διεπαφής ενός USB ελεγκτή υποδοχής είναι οι εξής:

Universal Host Controller Interface (UHCI): Δημιουργήθηκε για τις εκδόσεις USB 1.0 και 1.1 και κατ' επέκταση υποστηρίζει ταχύτητες Low-Speed και Full-Speed. Απαιτεί άδεια χρήσης από την Intel.

Open Host Controller Interface (OHCI): Η διαφορά με τη UHCI είναι ότι πραγματοποιεί περισσότερες λειτουργίες στο υλικό που έχει ως αποτέλεσμα την δημιουργία απλούστερου λογισμικού. Για τη UHCI ισχύει το αντίθετο.

Extented Host Controller Interface (EHCI): Δημιουργήθηκε για το USB 2.0 και High-Speed συναλλαγές ενώ μεταβιβάζει μεταφορές low-speed και full-speed σε συνοδευτικούς ελεγκτές.

eXtensible Host Controller Interface (xHCI): Δημιουργήθηκε για την υποστήριξη Super-Speed (USB 3.0) συσκευών.

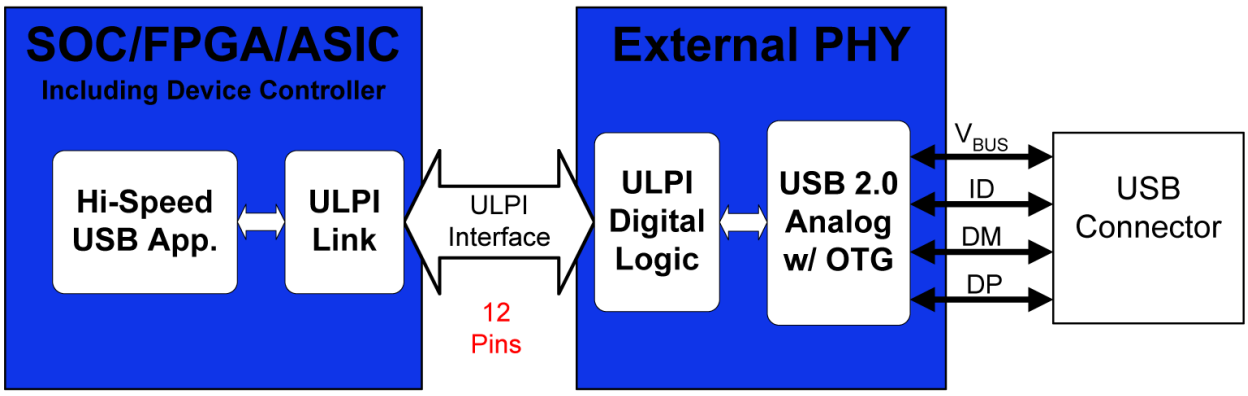
2.4.2 Διεπαφή ULPI

Κάνοντας μια σύνδεση USB απευθείας από ένα μεγάλο και περίπλοκο chip είναι ιδιαίτερα δύσκολο. Το USB χρησιμοποιεί ένα συνεστραμμένο ζεύγος καλωδίων για μεταφορά δεδομένων (D+ και D-), λειτουργεί σε αρκετά υψηλές ταχύτητες (για USB2.0 και άνω) και η τάση I/O ποικίλλει σημαντικά ανάλογα με την έκδοση του USB που χρησιμοποιεί (Low-Speed, Full-Speed, High-Speed). Στην ιδανική περίπτωση, το chip πρέπει να είναι ικανό να δεχτεί τάση 5V, ώστε να μπορεί να ανιχνεύει το σήμα από το καλώδιο Vbus για εφαρμογές USB Host ή OTG ενώ παράλληλα χρειάζεται καλή προστασία από απρόσεκτη κατάχρηση. Αυτές οι απαιτήσεις τείνουν να καθιστούν πολύ δύσκολη την εφαρμογή του USB στο ίδιο κομμάτι πυριτίου με ένα σύγχρονο FPGA ή CPU, τα οποία απαιτούν πολύ μικρότερα τρανζίστορ και πολύ χαμηλότερες τάσεις λειτουργίας για να επιτύχουν την απόδοσή τους. Οι περισσότεροι μικρο-ελεγκτές εφαρμόζουν μόνο την πλήρη ταχύτητα (Full-Speed) USB2.0, η οποία διαχειρίζεται εύκολα από μια τροφοδοσία 3.3V.[18] Τα περισσότερα FPGA παρουσιάζουν αδυναμία εφαρμογής High-Speed USB2.0 ή SuperSpeed USB3.0, αφού δεν μπορούν να παράξουν τάση εξόδου 5V αλλά ούτε και να δεχτούν τάση εισόδου 5V.[Εικόνα 10].

FPGA model	Core voltage range (tolerance), V	Auxiliary voltage range (tolerance), V	I/O voltage range (tolerance), V
Spartan-7	1.0 (50 mv)	1.8 (5%)	1.2 to 3.3 (5%)
Spartan-7 (-1LI)	0.95 (30 mv)	1.8 (5%)	1.2 to 3.3 (5%)
Artix-7	1.0 (50 mv)	1.8 (5%)	1.2 to 3.3 (5%)
Artix-7 (-2LE)	0.9 (30 mv)	1.8 (5%)	1.2 to 3.3 (5%)
Kintex-7 (-2LI)	0.95 (20 mv)	1.8 (5%)	1.2 to 3.3 (5%)
Arria II-GX	0.9 (30 mv)	2.5 (VCCA_PLL) 0.9 (VCCD_PLL)	1.2 to 3.3 (5%)
Stratix IV-GX	0.9 (30 mv)	2.5 (VCCA_PLL) 0.9 (VCCD_PLL)	1.2 to 3.0 (5%)
Cyclone IV	1.2 (50 mv)/ 1.0 (30 mv)	2.5 (VCCA_PLL) 1.2/1.0 (VCCD_PLL)	1.2 to 3.3 (5%)

Εικόνα 10: High-performance FPGAs with their voltage conditions recommended. Πηγή Εικόνας: [18]

Το ULPI (=UTMI+ low pin interface) είναι η λύση σε αυτό το πρόβλημα.[19] Σε αντίθεση με Low-Speed και Full-Speed USB συστήματα, τα οποία χρησιμοποιούν σειριακές διεπαφές, το ULPI παρέχει τη δυνατότητα χρήσης ενός παράλληλου αμφίδρομου δίαυλου για τη διεπαφή, που απαιτούν οι υψηλές ταχύτητες (High-Speed), μεταξύ του USB link στην FPGA και του φυσικού επιπέδου (USB PHY chip). Αυτό οδηγεί σε αντίστοιχη αύξηση της πολυπλοκότητας και του αριθμού των pin. Το ULPI περιορίζει τον αριθμό των σημάτων μόνο σε 8 ή 12, επειδή συνδυάζει μόνο τρία σήματα ελέγχου επιπρόσθετα με ένα ρολόι των 60 MHz. Τα σήματα αυτά αποτελούνται από ένα δίαυλο δεδομένων διπλής κατεύθυνσης των 4-bit ή 8-bit. Το chip PHY χειρίζεται όλες τις δύσκολες συμβάσεις USB και τείνει να έχει αρκετά σταθερή προστασία από ηλεκτρικές βλάβες.

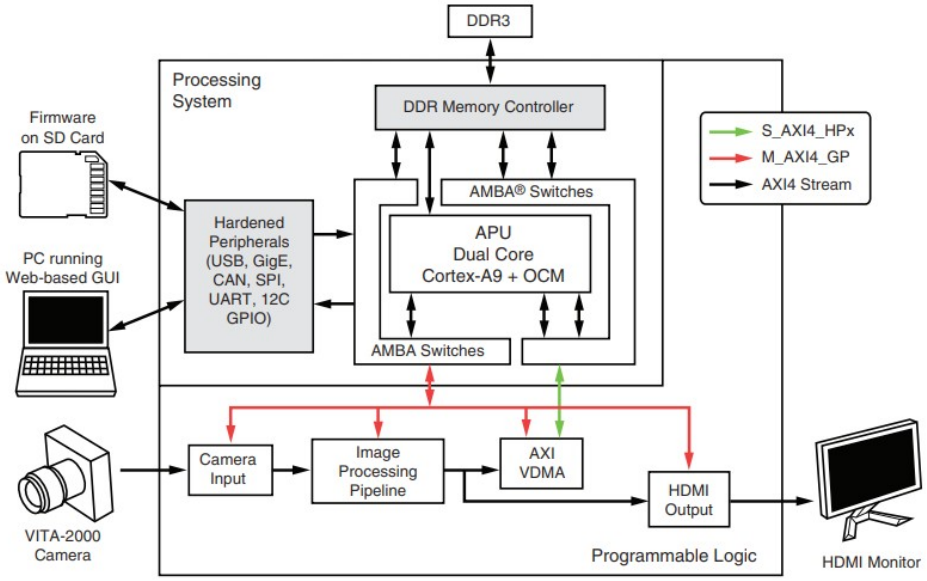


Εικόνα 11: USB ULPI PHY. Πηγή Εικόνας: [19]

2.5 Σχετικές εργασίες διεπαφής καμερών με FPGA

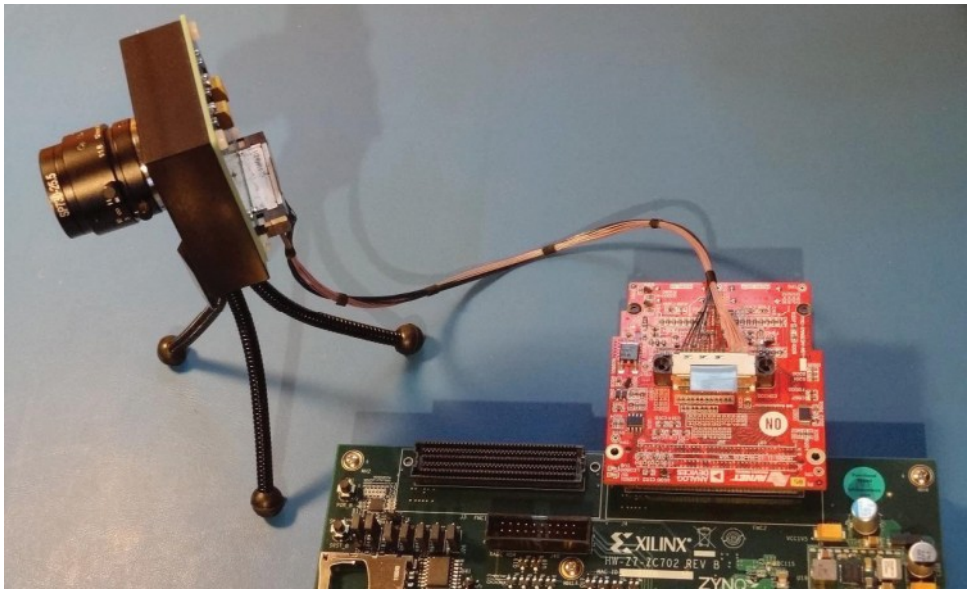
Υπάρχουν αρκετά συστήματα μετάδοσης, επεξεργασίας και εμφάνισης εικόνας που διατίθενται με σκοπό την επίδειξη των δυνατοτήτων μιας αναπτυξιακής πλακέτας. Πολλά απ΄ αυτά παρέχονται ως επίσημες λύσεις από την ίδια την Xilinx, άλλα μπορούν να βρεθούν από άρθρα εργασιών που έχουν εκδοθεί από πανεπιστήμια ανά τον κόσμο καθώς και από τον κόσμο που απαρτίζει την κοινότητα της ενσωματωμένων συστημάτων όπως forums και την Xilinx wiki [20]. Μια ειδοποιός διαφορά μεταξύ αυτών των συστημάτων εντοπίζεται στον τύπο κάμερας που χρησιμοποιείται και κατ' επέκταση το πρότυπο πρωτοκόλλου που θα εφαρμοστεί. Η περιοχή (PS ή PL) που θα γίνει η διεπαφή με το board καθώς και το πρωτόκολλο της κάμερας είναι τα στοιχεία που αλλάζουν την υλοποίηση του συστήματος.

Σε αυτό το application note [21] η Xilinx παρέχει ένα πλήρη οδηγό προς αναφορά για σχεδίαση του συστήματος επεξεργασίας εικόνας μέσω κάμερας. Σκοπός του συστήματος είναι η λήψη βίντεο από τον **αισθητήρα εικόνας Vita-2000** με ανάλυση στα 1080p60. Η ακατέργαστη εικόνα που λαμβάνεται, μετατρέπεται σε RGB μέσω ενός pipeline επεξεργασίας εικόνας υλοποιημένο στην προγραμματιζόμενη λογική (PL). Αφού ολοκληρωθούν οι λειτουργίες επεξεργασίας που απαιτούνται από την σχεδίαση του συστήματος, η εικόνα τελικά προβάλλεται σε μια οθόνη μέσω HDMI.



Εικόνα 12: Vita Camera Block Diagram. Πηγή Εικόνας: [21]

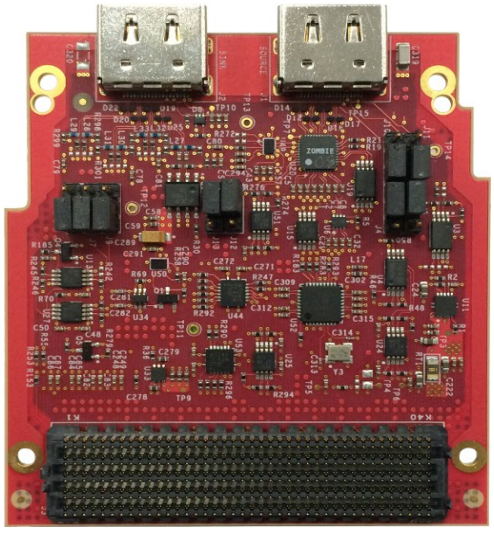
Στην [Εικόνα 12] παρουσιάζεται το μονοπάτι που ακολουθούν τα δεδομένα βίντεο, από την κάμερα στην μονάδα επεξεργασίας και από εκεί κατευθείαν στην έξοδο HDMI. Παρέχεται επίσης η δυνατότητα επικοινωνίας με τη μνήμη DDR3 για λόγους ανάλυσης, χωρίς να είναι υποχρεωτική για την λειτουργία του υπόλοιπου συστήματος. Για να πετύχει τη σύνδεση με το board χρησιμοποιεί την πλακέτα FPGA Mezzanine Card (FMC) IMAGEON.



Εικόνα 13: Vita 2000 Image Sensor with FMC IMAGEON interfaces Xilinx board. Πηγή Εικόνας: [21]

Η μονάδα FMC IMAGEON είναι αρκετά δημοφιλής στα projects της Xilinx. Εκτός από την διεπαφή LCD Coaxial Embedded Display Interface (LCEDI) που χρησιμοποιεί για την σύνδεση με τον

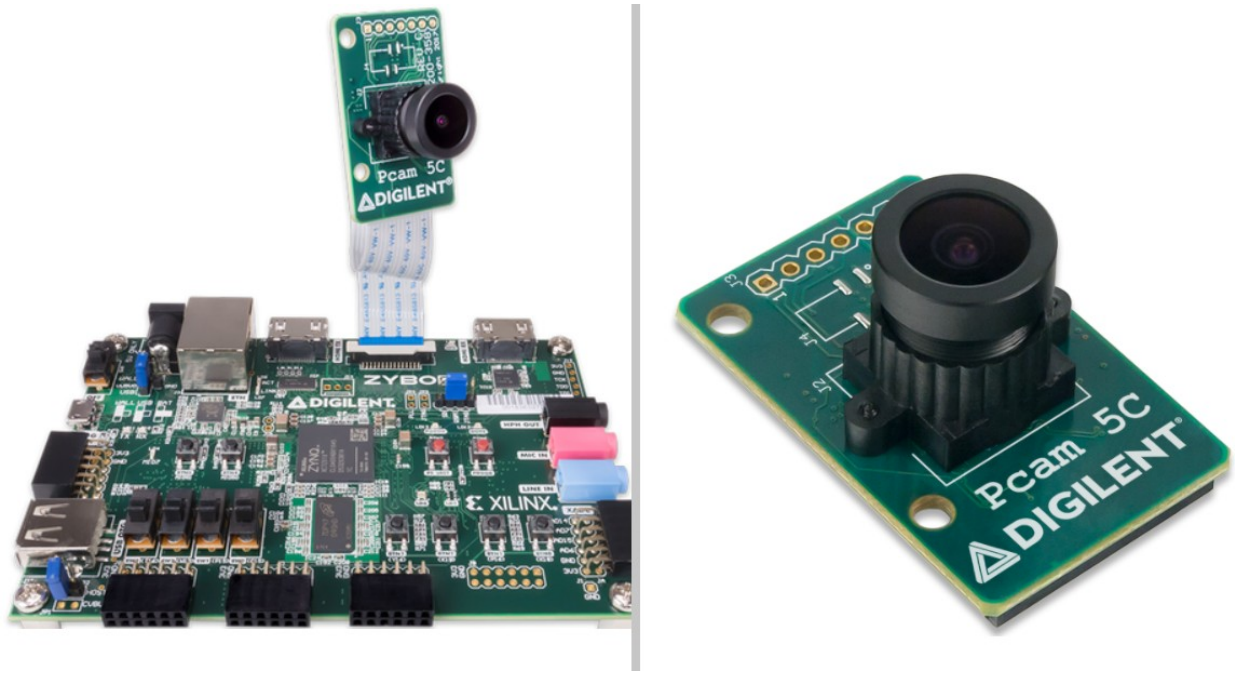
αισθητήρα εικόνας VITA, όπως φαίνεται στην [Εικόνα 13], προσφέρει διεπαφές και για HDMI είσοδο/έξοδο.



Εικόνα 14: AES-FMC-IMAGEON-G.
Πηγή Εικόνας: [56]

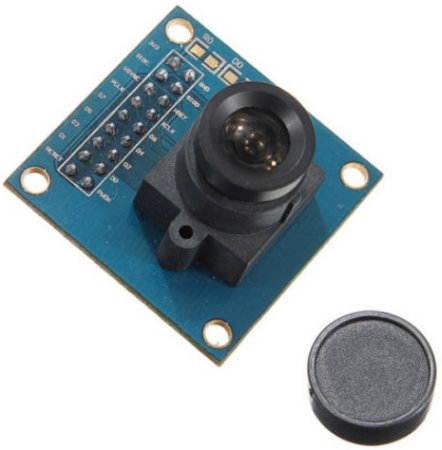
Μέσω αυτού υλοποιεί και το Targeted Reference Design (TRD). Η διαφορά με το προηγούμενο σύστημα είναι καταρχάς η χρήση **κάμερας HDMI**. Αφού πραγματοποιηθεί η λήψη εικόνας από την HDMI κάμερα, μέσω του αντίστοιχου πρωτοκόλλου διασύνδεσης, τα δεδομένα μεταφέρονται από την προγραμματιζόμενη λογική, που είναι συνδεδεμένη η κάμερα, στην εξωτερική μνήμη DDR3 και από εκεί επιστρέφουν στην προγραμματιζόμενη λογική ώστε τελικά να φτάσουν στην έξοδο HDMI. Πραγματοποιείται λογική framebuffer και όχι direct-streaming(όπως στην περίπτωση της Vita). Περισσότερες λεπτομέρειες για τον ορισμό και την διαφορά των δύο θα ακολουθήσουν στο κεφάλαιο 3.2.

Στην ίδια λογική υλοποίησης με αυτή της εισόδου HDMI, προσφέρει η Digilent έτοιμη λύση διεπαφής κάμερας με Zynq board υπό το πρίσμα του “Embedded Vision” πακέτου. Παρέχει μια μονάδα κάμερας που περιλαμβάνει αισθητήρα εικόνας **Omnivision OV5640** και συνδέεται με την αναπτυξιακή πλακέτα μέσω της διεπαφής MIPI CSI-2 [Εικόνα 15], προσφέροντας εξαιρετικά χαμηλό latency.[22]



Εικόνα 15: Embeded Vision with Pcam 5MP Camera Module. Πηγή Εικόνας: [22]

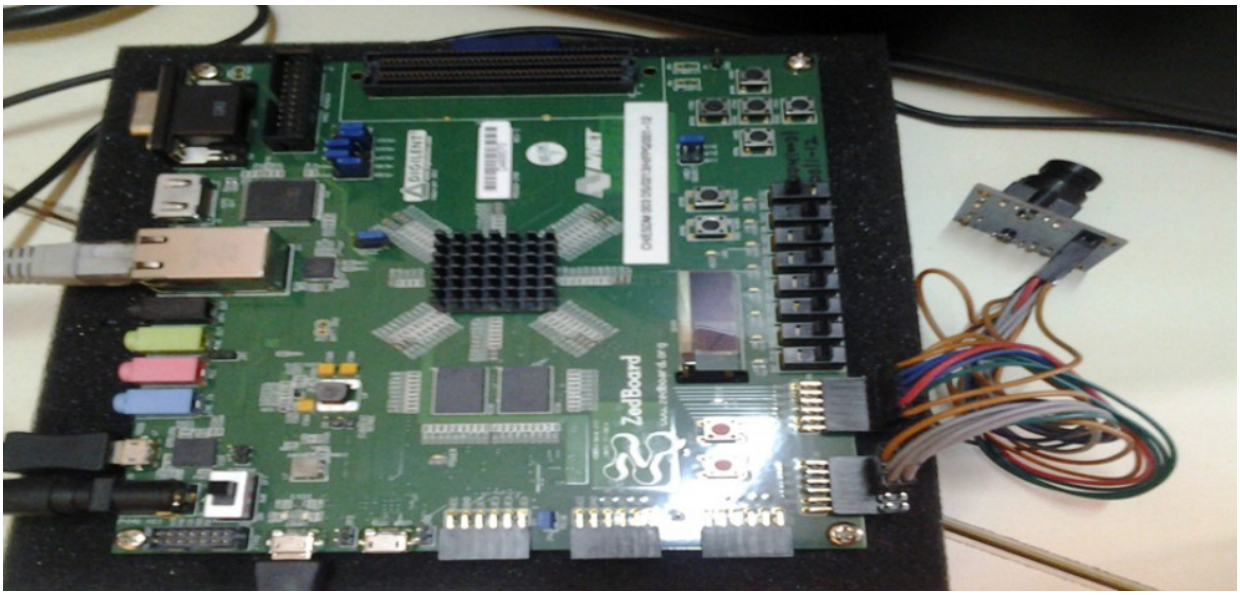
Σε αυτό το άρθρο [23] από το πανεπιστήμιο K L της Ινδίας περιγράφεται η σύνδεση μιας OV7670 κάμερας με ένα Zynq-7000 SoC Zedboard. Η μονάδα **κάμερας OV7670** [24] πρόκειται για αισθητήρα CMOS που υποστηρίζει YUV και RGB formats σε αναλύσεις VGA (640 x 480).



Εικόνα 16: OV7670 CMOS Camera Module. Πηγή Εικόνας: [24]

Σε αυτήν την περίπτωση μονάδες που πραγματοποιούν την λήψη και τον έλεγχο της εικόνας, την μεταφορά στην μνήμη και την έξοδο μέσω VGA είναι όλες υλοποιημένες στην προγραμματιζόμενη λογική. Η διαφορά με τις προηγούμενες περιπτώσεις είναι ότι εδώ γίνεται χρήση της μνήμης Block

Ram (BRAM) που βρίσκεται στην FPGA. Η προσέγγιση αυτή δύναται να συναντήσει προβλήματα, όταν το board που χρησιμοποιείται παρουσιάζει περιορισμένη μνήμη.



Εικόνα 17: Zedboard with OV7670 CMOS Camera. Πηγή Εικόνας: [23]

Τέλος, σε αυτό το άρθρο [25] που προέκυψε από μελέτη που πραγματοποιήθηκε από το στρατιωτικό πανεπιστήμιο της Κίνας, αναλύεται η χρήση μιας **IP Network κάμερας** για την υλοποίηση ενός συστήματος μετάδοσης, επεξεργασίας και εμφάνισης εικόνας. Η κάμερα που χρησιμοποιείται [Εικόνα 18] [26] ακολουθεί το πρωτόκολλο επικοινωνίας GigE και συνδέεται στην θύρα Ethernet που βρίσκεται στην μεριά του συστήματος επεξεργασίας (PS) του Zynq board.

Εικόνα 18: HIKVISION DS-2CD2325FWD-I Camera IP. Πηγή Εικόνας: [26]

Εδώ εντοπίζεται η διαφορά σε σχέση με τις προηγούμενες περιπτώσεις, στις οποίες η διεπαφή της κάμερας γίνεται απευθείας στην μεριά της FPGA, όπου η διεπαφή πραγματοποιείται μέσω των περιφερειακών στη μεριά του επεξεργαστή. Αφού υποστεί, κατ' επιλογήν, επεξεργασία αποστέλλεται στην εξωτερική μνήμη DDR3. Από την μνήμη μεταφέρεται στην προγραμματιζόμενη λογική και τελικά στην έξοδο μέσω HDMI. Το σύστημα λογισμικού αναπτύχθηκε σε περιβάλλον Linux εκμεταλλευόμενο τους αντίστοιχους οδηγούς συσκευών και βιβλιοθήκες που έχει να προσφέρει το συγκεκριμένο λογισμικό.

Από τις παραπάνω περιπτώσεις διαφαίνεται η ποικιλομορφία και το εύρος επιλογών σε κάμερες καθώς και το αντίκτυπο που αυτές έχουν στην λογική ανάπτυξης ενός συστήματος για λήψη και προβολή εικόνας. Παρόλα αυτά το ίδιο εύρος επιλογών δεν φαίνεται να ισχύει και για τις USB webcams. Η κάλυψη USB συστημάτων σε αναπτυξιακές πλακέτες, περιορίζεται ως επί το πλείστον σαν επιμέρους κομμάτια έτοιμων, μεγαλύτερων λύσεων που προσφέρονται από τις ίδιες τις εταιρίες, όπως η Xilinx. Αυτές οι λύσεις αφορούν πολύπλοκα και πολυσύνθετα συστήματα, που σαν κύριο λόγο ύπαρξης έχουν την επίδειξη των χαρακτηριστικών των συσκευών τους. Η ιδιαίτερα περιορισμένη υποστήριξη στα αρχεία και στους κώδικες αυτών των συστημάτων, δημιουργεί επιπλοκές στην εξαγωγή των επιμέρους υποσυστημάτων (π.χ USB) από αυτά. Η τελευταία παραδοχή σε συνδυασμό με το ότι αυτά τα συστήματα προορίζονται κυρίως για συσκευές FPGA/SoC/MPSoC τελευταίας γενιάς, έχει σαν αποτέλεσμα την έλλειψη μιας μεμονωμένης γενικής λύσης για τη δημιουργία και εκμετάλλευση ενός συστήματος USB σε οποιαδήποτε συσκευή.

Επιπρόσθετα στην προηγούμενη περίπτωση, δίνεται και η δυνατότητα δημιουργίας ενός συστήματος USB από το μηδέν. Για την πραγματοποίηση διεπαφής USB συσκευής με μια αναπτυξιακή πλακέτα, χρειάζεται να αναπτυχθεί κατάλληλη λογική στην FPGA και να γίνει χρήση εξωτερικών μονάδων για τη φυσική διεπαφή. Σε αυτές τις περιπτώσεις όμως, η τεκμηρίωση των λύσεων είναι ελλιπής και πολλές φορές "κρυμμένη" πίσω από την αγορά του προϊόντος. Οι μελέτες που αναλύουν διεξοδικά το συνδυασμό όλων των κομματιών που αφορούν τη δημιουργία ενός συστήματος USB, είναι ελάχιστες.

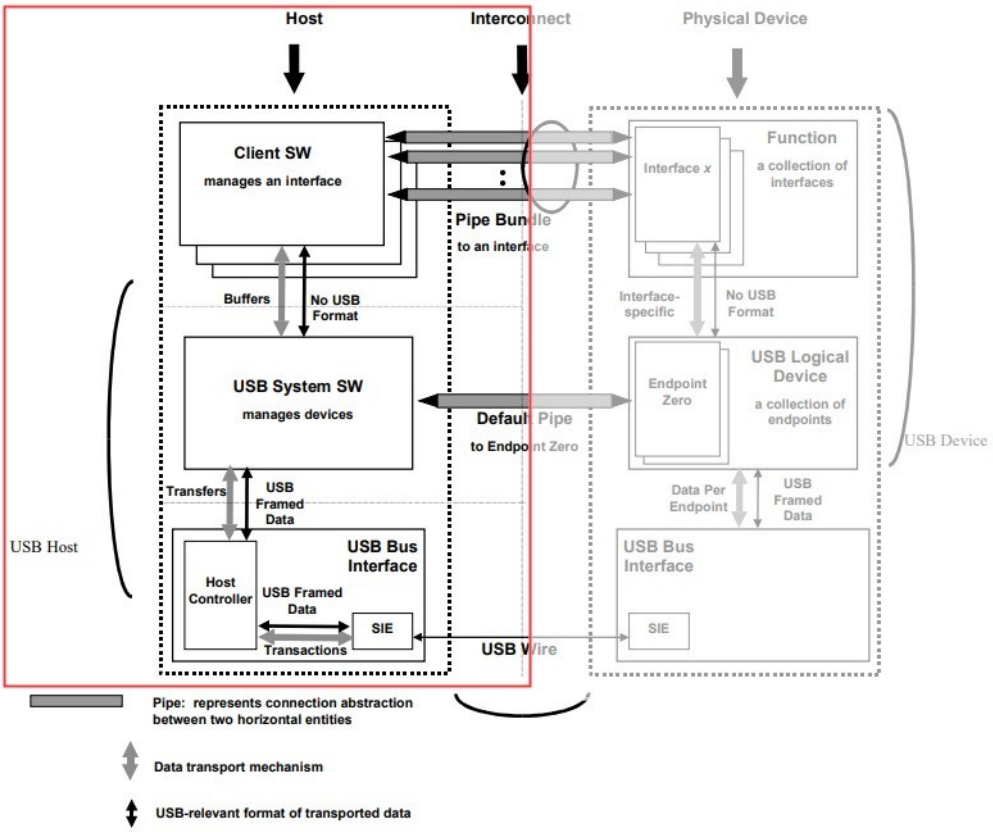
Σύμφωνα με τα παραπάνω, προκύπτει ότι είτε ένα USB σύστημα ως επιμέρους κομμάτι ενός μεγαλύτερου συστήματος, είτε οι περιορισμένες μελέτες για τη δημιουργία ενός συστήματος USB από την αρχή, δεν επαρκούν για να συντελέσουν μια μεθοδική ανάλυση για την αντιμετώπιση του θέματος. Στο επόμενο κεφάλαιο θα αναλυθούν οι τρόποι με τους οποίους μπορούν να συνδυαστούν τα επιμέρους κομμάτια που συντελούν ένα USB σύστημα ώστε να γίνει εξαγωγή μιας γενικής λύσης. Αποτέλεσμα της λύσης θα είναι η πραγματοποίηση διεπαφής μίας ή περισσοτέρων USB καμερών με μια αναπτυξιακή πλακέτα, καθώς και η εφαρμογή επεξεργασίας εικόνας σε πραγματικό χρόνο. Η επιλογή για μελέτη και χρήση USB webcams με μια συσκευή SoC, προκύπτει από το γεγονός των πολύ χαμηλών τιμών τους καθώς και της εύκολης προσβασιμότητας σε αυτές.

3 Μελέτη Συστήματος

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός συστήματος που θα λαμβάνει δεδομένα βίντεο από μια USB webcam και θα τα προβάλει σε μια οθόνη μέσω HDMI. Σε αυτό το κεφάλαιο παρουσιάζονται οι διαφορετικές προσεγγίσεις που μελετήθηκαν για την υλοποίηση ενός τέτοιου συστήματος. Στην πρώτη ενότητα (3.1) αναλύονται μέθοδοι ανάπτυξης ενός συστήματος USB σε μια συσκευή SoC, τόσο από την οπτική του hardware και την φυσική διασύνδεση αλλά τόσο και από την οπτική του software και τη χρήση προγραμμάτων οδήγησης USB συσκευών. Επιπλέον στη δεύτερη ενότητα (3.2) αναλύεται ο τρόπος με τον οποίο τα δεδομένα βίντεο θα προβληθούν σε μια οθόνη HDMI, ποιες αναπτυξιακές πλακέτες αφορά η συγκεκριμένη υλοποίηση και με ποια μέθοδο αυτές οι μονάδες hardware θα “μεταφραστούν” σε software ώστε να είναι προσβάσιμες από τον χρήστη. Στη τρίτη ενότητα (3.3) περιγράφονται τα χαρακτηριστικά ενός φίλτρου Sobel που θα αναπτυχθεί στην FPGA και θα εφαρμοστεί στα δεδομένα βίντεο που καταφθάνουν κατά τη λήψη της USB webcam, πριν αυτά προβληθούν.

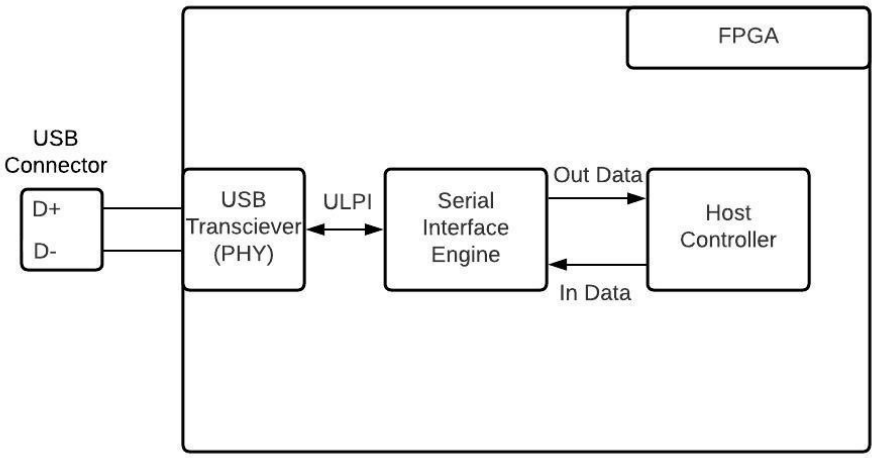
3.1 Μέθοδοι Υλοποίησης Συστήματος USB

Όπως προαναφέρθηκε στο κεφάλαιο 2.2 οι περισσότερες αναπτυξιακές πλακέτες FPGA που συναντώνται σήμερα δεν περιλαμβάνουν USB θύρα υποδοχής (host) ικανή να αναγνωρίσει USB συσκευές και να τις διαχειριστεί. Παρακάτω μελετάται πως μπορεί να υλοποιηθεί USB host λειτουργία σε μια FPGA. Όπως φαίνεται και στην Εικόνα 19 [27] η λύση αυτού του προβλήματος ανάγεται σε δύο επίπεδα. Το πρώτο αφορά την φυσική σύνδεση της USB συσκευής με την FPGA σε επίπεδο hardware (USB Bus Interface), ενώ το δεύτερο την διαχείριση αυτής σε επίπεδο software (USB System Software).



Εικόνα 19: USB Host/Device Detailed View. Πηγή Εικόνας: [27]

Ένα τυπικό σύστημα USB host αποτελείται από έναν πομποδέκτη (transceiver), ένα serial interface engine (SIE) και έναν ελεγκτή υποδοχής (host controller) όπως φαίνεται στην παρακάτω εικόνα. [Εικόνα 20]



Εικόνα 20: Τυπικό Σύστημα USB

Ο πομποδέκτης είναι υπεύθυνος για την υλοποίηση του φυσικού επιπέδου (physical layer) του USB πρωτοκόλλου, καθώς και για την διαμόρφωση των δύο άκρων της σύνδεσης δεδομένων μεταξύ του host και της συσκευής. Το interface engine έχει ως βασική αρμοδιότητα το serialization/deserialization των μεταδόσεων USB.

Ένα σύστημα USB μπορεί να υλοποιηθεί σε μια FPGA με τις εξής μεθόδους:

I. **Απευθείας σύνδεση σε FPGA**

II. **USB Soft IP with off-chip PHY**

III. **Off-chip USB Controller**

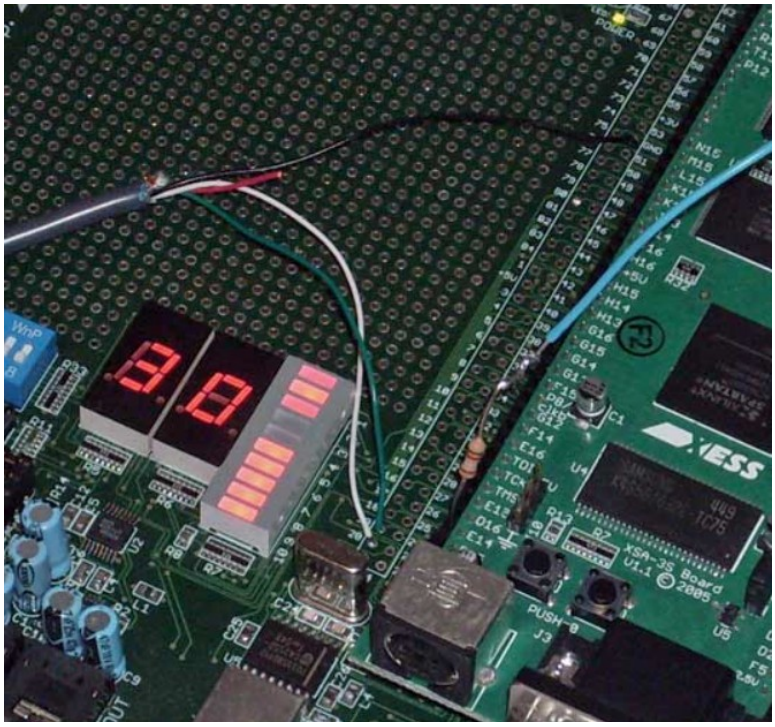
IV. **USB Controller on SoC**

Στα επόμενα υποκεφάλαια (3.1.1 - 3.1.4), ακολουθεί αναλυτική περιγραφή των παραπάνω μεθόδων αξιολογώντας τες με τα εξής κριτήρια:

- Πολυπλοκότητα Υλοποίησης: Αναφέρεται στην τεχνογνωσία που απαιτείται για την υλοποίηση του συστήματος καθώς επίσης και τον απαιτούμενο προσεγγιστικό χρόνο ανάπτυξης.
- Κατανάλωση Πόρων: Αναφέρεται στους πόρους που χρειάζεται να καταναλωθούν από την FPGA προκειμένου να λειτουργεί το σύστημα.
- Χρήση εξωτερικού Hardware: Αναφέρεται στην χρήση υλικού εξωτερικά της FPGA.
- Κόστος Υλοποίησης: Το συνολικό κόστος που χρειάζεται να δαπανηθεί προκειμένου να υλοποιηθεί το σύστημα (αγορά hardware, soft-cores ips)
- Χρήση CPU/Drivers: Το είδος επεξεργαστή και οδηγών που μπορούν να χρησιμοποιηθούν
- Ευελιξία Επαναχρησιμοποίησης: Αναφέρεται στην ευελιξία επαναχρησιμοποίησης του συστήματος από κάποιον χρήστη και την ποσότητα επιπλέον δουλειάς που απαιτείται από αυτόν.
- Αποτελεσματικότητα: Αναφέρεται σε μια ποιοτική εκτίμηση σε σχέση με τα παρεμφερή άρθρα που έχουν αναφερθεί.

3.1.1 Απευθείας σύνδεση σε FPGA

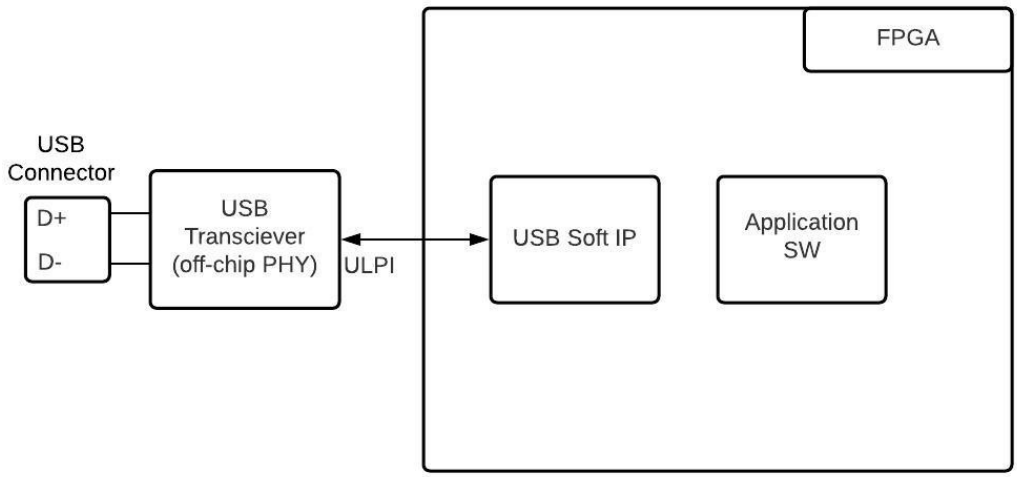
Αυτή η μέθοδος δεν περιλαμβάνει την χρήση κάποιου εξωτερικού πομποδέκτη, ο οποίος θα αναλαμβάνει τον ρόλο του physical layer (PHY). Το module που θα σχεδιαστεί θα καταναλώνει πόρους της FPGA και θα δρα ως ένας ελεγκτής με ενσωματωμένο πομποδέκτη (PHY). Η σύνδεση της συσκευής USB θα γίνει μέσω δύο I/O pin που βρίσκονται στη FPGA και στα σήματα D+, D- του καλωδίου USB[Εικόνα 21][28]. Επιπλέον χρειάζεται η υλοποίηση USB πρωτοκόλλου, το οποίο είναι αρκετά περίπλοκο. Η επικρατέστερη προσέγγιση θα ήταν να υπάρξει ένας soft-core CPU (Microblaze) για την υλοποίηση του Software Stack κάνοντας χρήση οδηγών. Αυτός ο τρόπος προσφέρει μια λύση στην φυσική διεπαφή μεταξύ συσκευής και FPGA ή οποία δεν χρειάζεται εξωτερικά chip. Παρ' όλα αυτά με αυτή τη προσέγγιση συγκεντρώνονται αρκετά προβλήματα. Καταρχάς δεν θα υπάρχει συμβατότητα με τις επίσημες προδιαγραφές USB 2.0 ή ακόμα και USB 1.0 αφού



Εικόνα 21: USB wires to XST-3 board. Πηγή Εικόνας: [28]

πλέον η διεπαφή θα γίνεται με εξατομικευμένη (custom) λογική. Επιπλέον για λόγους που αναφέρθηκαν στο κεφάλαιο 2.4 αυτή η μέθοδος δεν θα είναι λειτουργική αν η FPGA χρειάζεται να έχει ρόλο Host λόγω της τάσης εξόδου που μπορεί να παρέχει μια FPGA στα περιφερειακά της. Με αυτόν τον τρόπο δεν υποστηρίζονται isochronous μεταφορές (βίντεο πραγματικού χρόνου) αλλά και ούτε υψηλές ταχύτητες (480Mbps), παρά μόνο περιπτώσεις μεταφορών control πλήρης ταχύτητας μετάδοσης στα 12 Mbps. Τέλος, και πιο σημαντικό, η πλήρης ανάπτυξη τόσο για τον πομποδέκτη (transceiver PHY) αλλά τόσο και για τον ελεγκτή, χρησιμοποιώντας αποκλειστικά πόρους της FPGA, θα δεσμεύσει σημαντικό μέρος των λογικών block αυτής.

3.1.2 USB Soft IP with off-chip PHY



Εικόνα 22: FPGA with integrated IP and external USB transceiver

Αυτή η περίπτωση αφορά την χρήση ενός υψηλής ταχύτητας φυσικού επιπέδου (PHY) πομποδέκτη, που θα συνδέεται εξωτερικά με την FPGA μέσω I/O γενικής χρήσης, συνδυασμένο με ένα USB soft core IP υλοποιημένο στην FPGA. Μια αναπτυξιακή πλατφόρμα χρησιμοποιείται ως USB 2.0 High-Speed PHY βασισμένη στην προδιαγραφή ULPI. Παράδειγμα τέτοιας συσκευής αποτελεί η αναπτυξιακή πλακέτα USB3300 [29]. Το USB PHY περιέχει έναν αναλογικό πομποδέκτη (transciever) και κάποια λογική ψηφιακής επεξεργασίας, ενώ για τη λειτουργία του συστήματος απαιτείται η σύνδεση του USB PHY με το υλοποιημένο soft core IP στην FPGA.

Εικόνα 23: Αναπτυξιακή Πλατφόρμα USB330. Πηγή Εικόνας: [29]

Όπως φαίνεται και στην Εικόνα 24 το USB PHY από τη μία μεριά δέχεται τα δεδομένα υπό μορφή USB πακέτων, από τη συνδεδεμένη USB συσκευή, ενώ από την άλλη μεριά χρησιμοποιείται η διεπαφή ULPI για επικοινωνία με FPGA. Ένα USB 2.0 PHY εκτελεί χαμηλού επιπέδου (low-level)

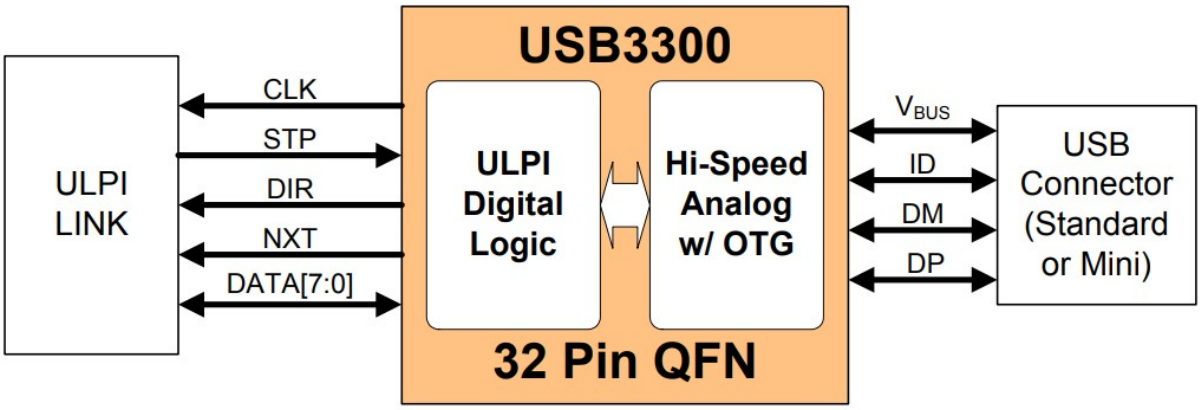
λειτουργίες πρωτοκόλλου ελεγχόμενο πλήρως από τον USB IP στην FPGA. Μερικές από αυτές τις λειτουργίες είναι οι εξής:

Κατά τη μετάδοση:

- Τα δεδομένα που έρχονται παράλληλα από το SIE του USB Host Bus Interface[Εικόνα 19] μετατρέπονται σε σειριακά για να μεταδοθούν στο καλώδιο USB.
- Δημιουργεί δείκτες για τον χρονισμό ώστε να σηματοδοτείται η αρχή και το τέλος του κάθε USB πακέτου.
- Εκτελεί bit stuffing και χρησιμοποιεί κωδικοποίηση Non-Return-to-Zero (NRZI) για τα δεδομένα μετάδοσης.

Κατά τη λήψη δεδομένων:

- Σειριακά δεδομένα από το καλώδιο USB μετατρέπονται σε παράλληλα
- Αφαιρεί stuff bits και εκτελεί αποκωδικοποίηση NRZI για τα δεδομένα που έχουν ληφθεί
- Ανακτά εισερχόμενα δεδομένα σύμφωνα με τις απαιτήσεις USB για συχνότητα και duty cycle



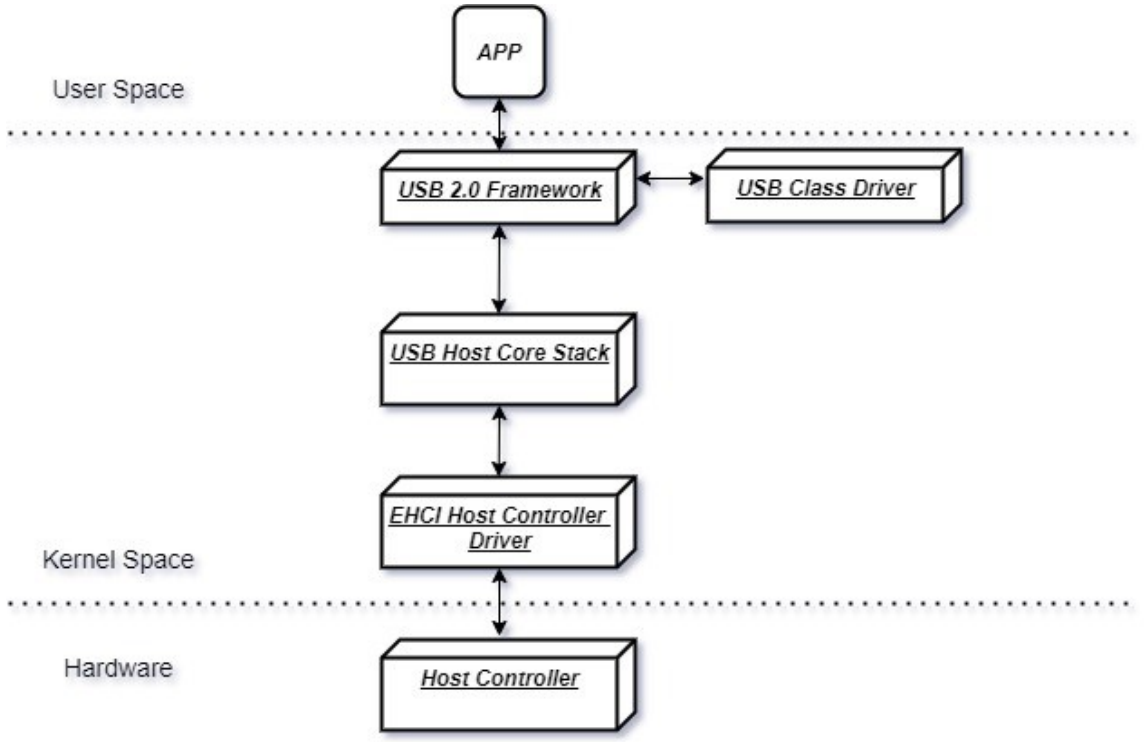
Εικόνα 24: USB3300 PHY Block Diagram. Πηγή Εικόνας: [29]

Το block ULPI LINK αποτελεί μέρος της FPGA. Πρόκειται στην ουσία για το hardware κομμάτι στην ανάπτυξη ενός USB host. Υλοποιημένο σε γλώσσα περιγραφής υλικού (HDL) το USB Host IP core αναλαμβάνει την αρμοδιότητα του ελεγκτή ως προς το USB PHY και θα πρέπει να μπορεί να υποστηρίξει λειτουργία host κατά τις προδιαγραφές του πρωτοκόλλου USB 2.0. Ο USB host controller:

- Ελέγχει τις ρυθμίσεις λειτουργίας του PHY.
- Επικοινωνεί μέσω standard buses, όπως ARM Advanced Microcontroller Bus Architecture (AMBA) interconnect, με το υπόλοιπο σύστημα (μνήμη, επεξεργαστή).

- Αποθηκεύει τα δεδομένα σε buffer κατά τη λήψη και μετάδοσης
- Ελέγχει το PHY ώστε να εκτελείται το High-Speed Handshake πρωτόκολλο
- Εκτελεί κωδικοποίηση και ελέγχους (CRC) για εντοπισμό και διόρθωση λαθών προκειμένου τα δεδομένα να μεταδίδονται χωρίς σφάλματα.

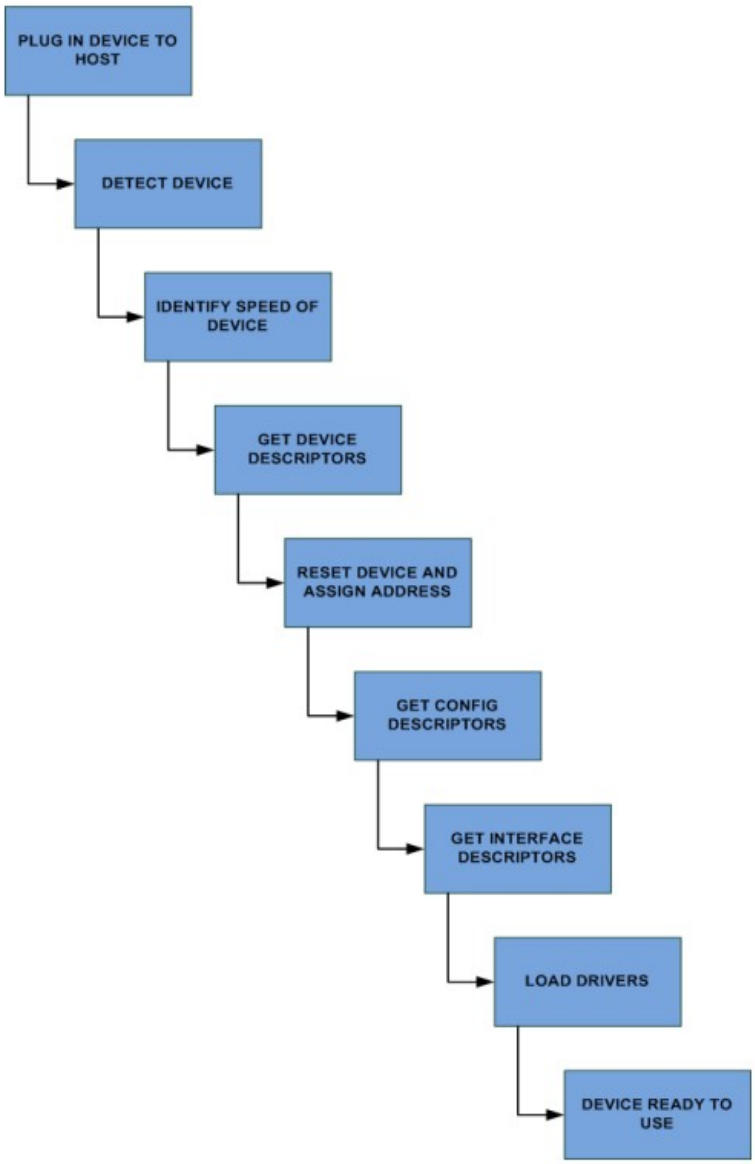
Παράλληλα με το USB host controller θα πρέπει να αναπτυχθούν και οι κατάλληλοι οδηγοί, οι οποίοι μαζί με έναν soft-core επεξεργαστή (Microblaze) συντελούν το USB 2.0 software stack του συστήματος όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 25: USB Host Software Stack

EHCI Host Controller Driver: Αποτελεί έναν οδηγό που εξαρτάται από το υλικό (hardware specific) και υλοποιεί τη διεπαφή μεταξύ του host controller, που βρίσκεται στη μεριά του hardware, και του USB συστήματος λογισμικού. Ο οδηγός αυτός παρέχει αφαιρετικότητα (abstraction) για τη μεταφορά δεδομένων και κατανομή πόρων από το host controller.

USB Host Core Stack Driver: Πρόκειται για ένα ανεξάρτητο του υλικού (hardware independent) οδηγό που υλοποιεί το USB 2.0 πρωτόκολλο. Πραγματοποιεί την πιο επίπονη διεργασία που χρειάζεται για την έγκυρη επικοινωνία μεταξύ υποδοχής και μιας συσκευής, την απαρίθμησή (enumaration) καθώς και το χειρισμό της μεταφοράς δεδομένων στο δίαυλο USB.[Εικόνα 26][30]



Εικόνα 26: USB Enumaration Flow. Πηγή Εικόνας: [30]

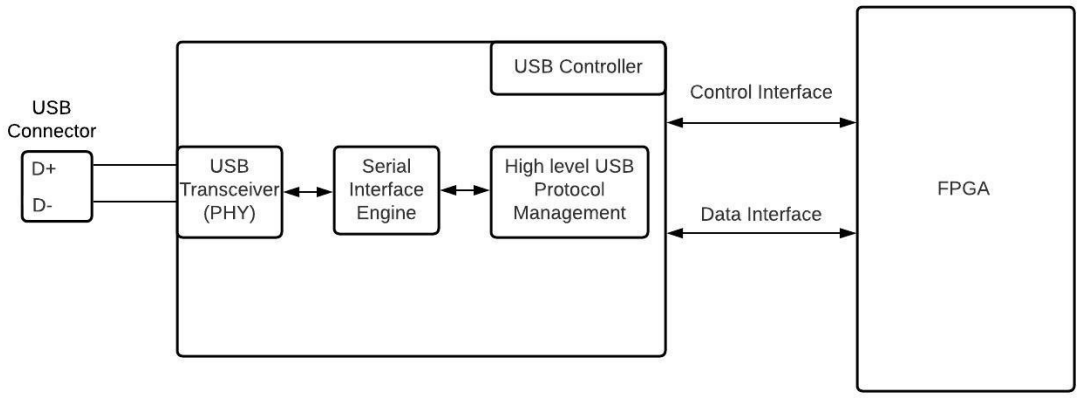
USB Class Driver: Πρόκειται για τον κύριο οδηγό της συσκευής (function driver), ο οποίος πρέπει να φροντίσει μόνο τις λεπτομέρειες πρωτοκόλλου συγκεκριμένης κλάσης [Table 3] και δεν χρειάζεται να χειρίζεται λειτουργίες του host controller. Για παράδειγμα ο USB video class driver χειρίζεται συσκευές ικανές για βίντεο streaming, όπως webcams.

USB 2.0 Host Framework: Πρόκειται για οδηγούς οι οποίοι δεν ελέγχουν άμεσα τις συσκευές, αντ' αυτού παρέχουν ένα αφαιρετικό μοντέλο κλάσεων USB. Φροντίζει για χειρισμούς σφαλμάτων, μεταβιβάζει αιτήσεις στον USB Host Core Stack Driver και ενημερώνει τους οδηγούς κλάσης για συμβάντα/αλλαγές στο υλικό.

Το πλεονέκτημα αυτής της προσέγγισης είναι η ελάχιστη χρήση εξωτερικού υλικού. Το μεγάλο μειονέκτημα όμως έγκειται στην ανάπτυξη του USB IP soft core (μαζί με του οδηγούς του) και

στην σημαντική κατανάλωση πολύτιμων πόρων της FPGA. Επιπλέον αξίζει να σημειωθεί ότι η εν αντιθέσει με classes όπως Human Interface Device(HID), δεν υπάρχουν οδηγοί για USB video class σε baremetal (no-OS) περιβάλλον. Η ανάπτυξή τους από το μηδέν είναι κάτι που πρέπει να αποφευχθεί λόγω πολυπλοκότητας και χρόνου που απαιτείται. Γι' αυτόν το λόγο προτείνεται η χρήση linux drivers που υπάρχουν σε πληθώρα και ικανοποιούν πολλές συσκευές [31]. Επιπρόσθετα τέτοιου είδους IP cores απαιτούν βαθιά κατανόηση του υλικού που θα χρησιμοποιηθεί και γνώση του USB πρωτοκόλλου. Λαμβάνοντας υπόψιν το μεγάλο χρόνο ανάπτυξης (τουλάχιστον 1 χρόνος) ενός USB host core υψηλής ταχύτητας, αφήνει σαν επιλογή την αγορά ενός τέτοιου IP από 3rd Party εταιρίες. Έπειτα από σχετική έρευνα όμως, διαπιστώθηκε ότι αυτές οι εταιρίες κοστολογούν πολύ ακριβά τέτοια προϊόντα. Τέλος παρατηρήθηκε μερική διαθεσιμότητα και από open source κοινότητες, όμως τα περισσότερα από αυτά αποδεικνύονται ως επί το πλείστον δυσλειτουργικά και μη αξιόπιστα.

3.1.3 Off-chip USB Controller



Εικόνα 27: External USB Controller

Με αυτή τη προσέγγιση εξετάζεται η διεπαφή ενός πλήρους ελεγκτή USB με την FPGA ώστε να χρησιμοποιούνται από αυτή όσο το δυνατό λιγότεροι πόροι. Οι περισσότερες λειτουργίες που αναφέρθηκαν στην προηγούμενη περίπτωση (USB controller + PHY) πραγματοποιούνται πλέον σε ένα εξωτερικό USB chip. Η FPGA απαλλάσσεται από τη διαχείριση του πρωτοκόλλου USB, αφού ο ελεγκτής εκτελεί τις λειτουργίες της σύνδεσης σε φυσικό επίπεδο (PHY), του SIE καθώς και τη λογική πρωτοκόλλου. Η χρήση ενός τέτοιου ελεγκτή προσφέρει ευελιξία στον αριθμό endpoints, στο μέγεθος της ουράς FIFO και σε άλλους περιγραφείς (descriptors) μεταφοράς. Η λειτουργία που πρέπει να εκτελέσει η FPGA εδώ, είναι η λογική για την διεπαφή με αυτό το ολοκληρωμένο κύκλωμα. Η διεπαφή μεταξύ των δύο μπορεί να είναι οποιαδήποτε από τις πρότυπες διεπαφές που υπάρχουν και υποστηρίζονται από τις 2 μεριές, όπως Serial Peripheral Interface (SPI) ή Inter-Integrated Circuit (I2C) ή ακόμα και εξατομικευμένη (custom) διεπαφή που βρίσκεται στο συγκεκριμένο USB chip. Στην FPGA θα πρέπει να υλοποιηθεί ένας soft-core μικροεπεξεργαστής (π.χ Microblaze) ο οποίος αναλαμβάνει τη διαμόρφωση λογισμικού (USB stack) για τις λειτουργίες της USB συσκευής που είναι συνδεδεμένη στον εξωτερικό ελεγκτή USB. Αυτό συνεπάγεται με ανάπτυξη/χρήση USB linux οδηγών όπως αυτοί περιγράφηκαν στην προηγούμενη προσέγγιση. Συχνά οι απαραίτητοι οδηγοί προσφέρονται μαζί με την αγορά του USB ελεγκτή. Ένα τέτοιο παράδειγμα

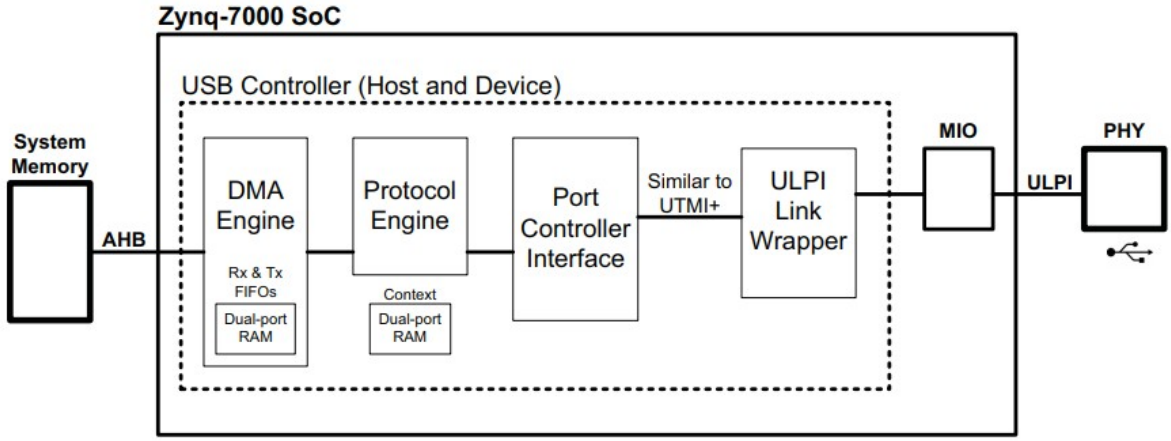
ελεγκτή αποτελεί ο CY7C67300 USB EZ-Host controller από την Cypress ο οποίος περιέχει ενσωματωμένη μνήμη και μικροεπεξεργαστή [32]. Σύμφωνα με το άρθρο των Ansiya Eshack και Jagadeesh Kumar [33], η διαδικασία απαρίθμησής (enumeration) ξεκινάει από τον Microblaze στην FPGA, όταν προκύπτει κάποια διακοπή (interrupt) από τον EZ-Host που να σηματοδοτεί ότι συνδέθηκε κάποια συσκευή USB. Μετά τη διακοπή ακολουθούν τα εξής βήματα:

- Διαμόρφωση EZ-Host controller
- USB Reset
- USB Set Address
- USB Get Device Description

Από το παραπάνω paper αλλά και από άλλα, όπως το [34] που κάνει χρήση του ISP1761 Hi-Speed USB controller [35], προκύπτει ότι η διεπαφή μιας συσκευής USB με έναν μικροεπεξεργαστή που βρίσκεται σε μια FPGA, χρησιμοποιώντας εξωτερικό USB controller, αποτελεί εύκολη διαδικασία, καλά τεκμηριωμένη και δεν "βαραίνει" ιδιαίτερα την FPGA. Με αυτόν τον τρόπο ωστόσο, η υλοποίηση του συστήματος απαιτεί την αγορά εξωτερικού κυκλώματος, των οποίων τα κόστη χρήσης υπερβαίνουν κατά πολύ τις ανάγκες της παρούσας εργασίας.

3.1.4 USB Controller on SoC

Σε αντίθεση με τις μεμονωμένες FPGA, οι αναπτυξιακές πλακέτες Zynq-7000 SoCs παρέχουν μια πληθώρα από διεπαφές για περιφερειακές συσκευές (π.χ USB) στην μεριά του επεξεργαστή. Σε ένα τυπικό σύστημα Zynq SoC υπάρχουν ενσωματωμένοι δύο πανομοιότυποι USB ελεγκτές. Ο κάθε ελεγκτής είναι ανεξάρτητος, μπορεί να πραγματοποιήσει ένα μεγάλο εύρος από εφαρμογές και μπορεί να διαμορφωθεί ως host, ως device αλλά και ως On-The-Go (OTG). Εδώ εισάγεται μια καινούρια έννοια, αυτή του USB On-The-Go (OTG). Αποτελεί μια προδιαγραφή που επιτρέπει σε συσκευές USB να εναλλάσσονται μεταξύ των ρόλων host και device. Όταν δύο συσκευές συνδεθούν και μια από αυτές είναι συσκευή USB OTG τότε δημιουργούν έναν σύνδεσμο επικοινωνίας. Η συσκευή που ελέγχει τον σύνδεσμο γίνεται ο master (host) ενώ η άλλη γίνεται ο slave (device). Πρόκειται δηλαδή για ένα υπερσύνολο των άλλων δύο προδιαγραφών (host, device) αφού μια συσκευή μπορεί να αλλάξει ρόλους δυναμικά, χωρίς να χρειάζεται επαναπρογραμματισμός. Ο αρχικός ρόλος κάθε συσκευής ορίζεται από το βύσμα καλωδίου που θα συνδεθεί.



Εικόνα 28: USB Controller Block Diagram. Πηγή Εικόνας: [38]

Από το παραπάνω σχήμα[Εικόνα 28] φαίνονται τόσο οι εξωτερικές συνδέσεις όσο και τα εσωτερικά modules του USB ελεγκτή. Τα Zynq-7000 SoC χρησιμοποιούν το ULPI πρωτόκολλο για να συνδεθούν με τον εξωτερικό ULPI PHY (που περιλαμβάνεται στο board) μέσω των MIO pins. Η διεπαφή ULPI παρέχει μια 8-bit παράλληλη "single data rate (SDR)" διαδρομή δεδομένων από τον εσωτερικό δίαυλο του ελεγκτή στο PHY. Διαφορετικές συσκευές περιλαμβάνουν και διαφορετικά PHYs, όπως για παράδειγμα το ZYBO Z7 χρησιμοποιεί ως PHY το USB3320 USB 2.0 Transceiver Chip [36] ενώ το ZedBoard χρησιμοποιεί το TI TUSB1210 USB Transceiver Chip [37]. Από την άλλη μεριά το DMA engine του ελεγκτή πρέπει να έχει πρόσβαση στη μνήμη του συστήματος (DDR) μέσω του AHB δίαυλου. Το DMA engine διατηρεί τους περιγραφείς (descriptors) μεταφοράς και buffers δεδομένων.

Οι λειτουργίες του USB controller και του PHY chip είναι παρεμφερείς με αυτές που περιγράφηκαν στο "*USB IP soft core with off-chip PHY* " και γι' αυτό δεν θα αναφερθούν ξανά εδώ. Περαιτέρω πληροφορίες και ανάλυση σχετικά με τα επιμέρους modules του USB controller δίνονται στο *"Chapter 15"* του [38]. Επιπρόσθετα και σε αυτή τη περίπτωση είναι αναγκαία η ανάπτυξη, σε περιβάλλον linux, των οδηγών:

- EHCI Host Controller Driver
- USB Host Core Stack
- USB Class Driver
- USB Host Framework

Η διαφορά στις δύο μεθόδους ωστόσο προκύπτει από την χρήση ενός hard-core processor, του ARM, για επικοινωνία με τον ελεγκτή και υλοποίηση του USB software stack. Σε σχέση με το soft-core processor (Microblaze) της προηγούμενης περίπτωσης, ο ARM είναι κατά πολύ ταχύτερος και δεν καταλαμβάνει καθόλου πόρους από την FPGA, αφήνοντας ελεύθερο χώρο για πιο ουσιαστικές εργασίες όπως επεξεργασία εικόνας, σήματος κλπ.

3.1.5 Σύγκριση μεθόδων διεπαφής FPGA με USB περιφερειακά

Σύμφωνα με την ανάλυση που έγιναν στα προηγούμενα υποκεφάλαια προκύπτουν τα παρακάτω συμπεράσματα. Για τις περιπτώσεις "Απευθείας σύνδεση με FPGA " και "USB Soft IP with off-chip PHY" η πολυπλοκότητα υλοποίησης είναι μεγάλη. Απαιτείται βαθιά κατανόηση και τεχνογνωσία πάνω στο πρωτόκολλο USB καθώς και το εκάστοτε hardware στο οποίο θα εφαρμοστεί. Η υλοποίηση γίνεται μέσα στην FPGA, καταλαμβάνοντας σημαντικό μέρος των διαθέσιμων πόρων της, ενώ η επαναχρησιμοποίησή της σε άλλες συσκευές απαιτεί εκ νέου μελέτη της FPGA. Για την περίπτωση του "Off-chip USB Controller" δεν θα επιβληθεί μεγάλη κατανάλωση πόρων στην FPGA και η σύνδεση με οποιαδήποτε συσκευή θεωρείται τετριμμένη, εξαρτάται όμως από την υποστήριξη της εταιρίας που το παρέχει. Το κόστος για την υλοποίηση αυτής της επιλογής όμως, αποδεικνύεται συχνά αρκετά μεγάλο.

Για τους παραπάνω λόγους η περίπτωση που επιλέχθηκε για την υλοποίηση του συστήματος είναι αυτή του "USB Controller on SoC". Στο κεφάλαιο 2.2 έγινε η παραδοχή χρήσης συσκευών SoC αντί των απλών FPGA, λόγω των πολλαπλών πλεονεκτημάτων που απορρέουν από την προσθήκη επεξεργαστή ARM. Επομένως η χρήση του USB controller που βρίσκεται στο σύστημα επεξεργασίας (PS) μιας συσκευής SoC, αποτελεί την πιο λογική και πρακτική επιλογή. Αυτό το γεγονός όμως, δεν αποτελεί τον μοναδικό λόγο αφού με αυτόν τον τρόπο, η πολυπλοκότητα υλοποίησης περιορίζεται σε αρκετά χαμηλά επίπεδα ενώ η επανεφαρμογή της υλοποίησης σε οποιοδήποτε SoC χαρακτηρίζεται ως τετριμμένη διαδικασία. Δεν απαιτείται η χρήση κάποιας εξωτερικής μονάδας αφού το απαραίτητο υλικό (USB PHY Chip) και λογισμικό (USB drivers) υπάρχει και ο χρήστης πρέπει απλά να γνωρίζει πως να πραγματοποιήσει την απαραίτητη διαμόρφωση (όπως αυτή θα παρουσιαστεί στο κεφάλαιο 4). Επιπλέον αυτή η μέθοδος δεν υλοποιείται κατευθείαν στην μεριά της FPGA, αφήνοντας αδέσμευτους τους πόρους της για εφαρμογές που εκμεταλλεύονται αρτιότερα τα ιδιαίτερα χαρακτηριστικά της, όπως επεξεργασία εικόνας/βίντεο. Τέλος, η ύπαρξη Linux οδηγών για USB βίντεο συσκευές, οδηγεί στη χρήση περιβάλλοντος Linux για το χειρισμό και την επικοινωνία USB καμερών, στην μεριά του PS, με τα λογικά block που έχουν υλοποιηθεί στη μεριά του PL.

Ακολουθεί μια συγκεντρωτική σύγκριση των μεθόδων που παρουσιάστηκαν στα κεφάλαια 3.1.1 - 3.1.4:

Table 4: Ποιοτική Σύγκριση μεθόδων Υλοποίησης USB

	Πολυπλοκότητα Υλοποίησης	***Κατανάλωση Πόρων***	***Χρήση Εξωτερικού HW***	***Κόστος Υλοποίησης***	***Ευελιξία Επαναχρησιμοποίησης***	***Αποτελεσματικότητα***
Απευθείας Σύνδεση σε FPGA	*Μεγάλη*	*Μεγάλη*	*Όχι*	*Μικρό*	*Μεσαία*	*Μικρή*
USB Soft IP with off-chip PHY *(1)*	*Μεγάλη*	*Μεσαία*	*Ναι*	*Μικρό*	*Μεσαία*	*Μεγάλη*

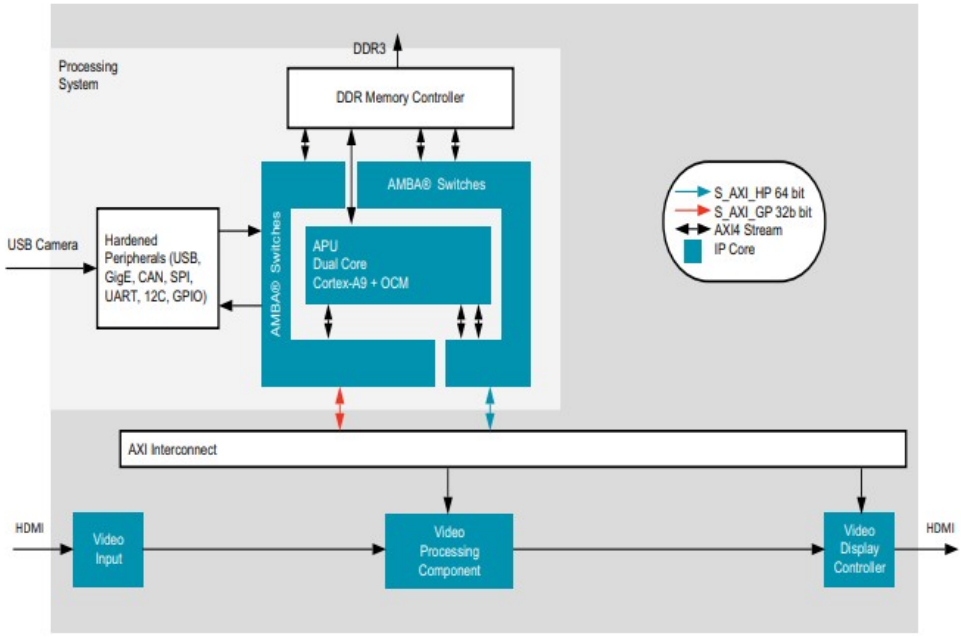
USB Soft IP with off-chip PHY *(2)*	*Μεσαία*	*Μεσαία*	*Ναι*	*Μεγάλο*	*Μεσαία*	*Μεγάλη*
Off-chip USB Controller	*Μικρή*	*Μικρή*	*Ναι*	*Μεγάλο*	*Μεγάλη*	*Μεγάλη*
USB Controller on SoC	*Μικρή*	*Μικρή*	*Όχι*	*Μικρό*	*Μεγάλη*	*Μεσαία*

(1): Με ανάπτυξη του Soft-Core IP και drivers από τον χρήστη

(2): Με αγορά του Soft-Core IP(+ drivers) από 3rd Party εταιρίες

3.2 Μέθοδοι Υλοποίησης Συστήματος HDMI

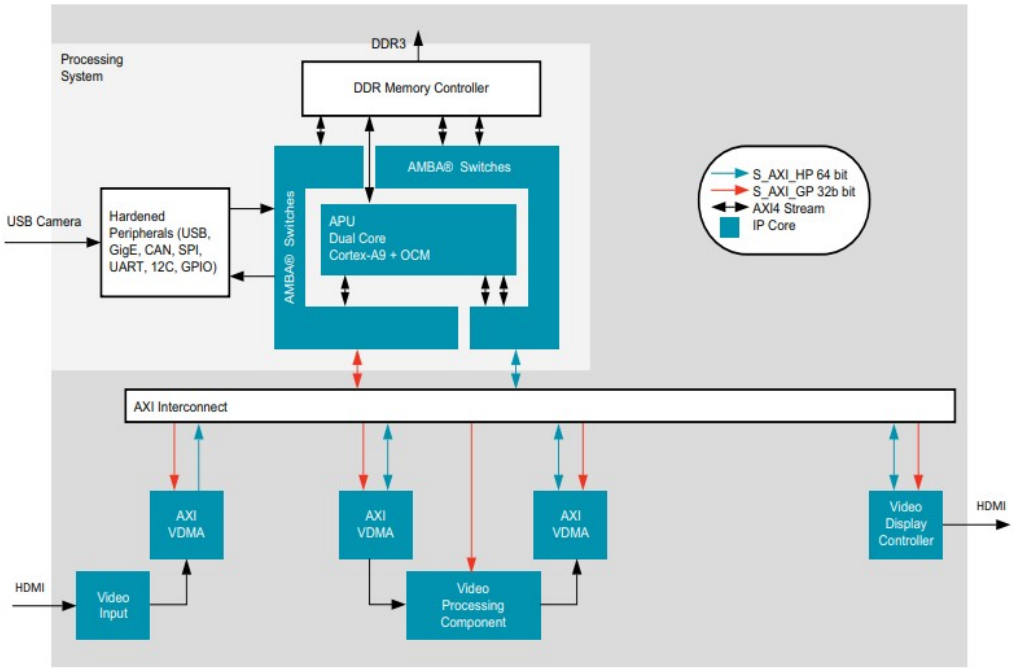
Σε πλατφόρμες Zynq SoCs υπάρχουν δύο βασικές ιδέες για την επεξεργασία και προβολή βίντεο σε κάποια οθόνη μέσω HDMI.[39] Η πρώτη αναφέρεται ως "άμεση ροή" (direct streaming) κατά την οποία δεδομένα εικονοστοιχείων καταφθάνουν στα pins εισόδου της προγραμματιζόμενης λογικής (FPGA) και μεταδίδονται απευθείας σε μια μονάδα επεξεργασίας βίντεο και στη συνέχεια σε μια έξοδο βίντεο. Μια τέτοια προσέγγιση είναι τυπικά πιο απλή και πιο αποδοτική αλλά απαιτεί από την μονάδα επεξεργασίας βίντεο να είναι σε θέση να επεξεργάζεται frames αυστηρά σε πραγματικό χρόνο.



Εικόνα 29: Direct Streaming Architecture. Πηγή Εικόνας: [39]

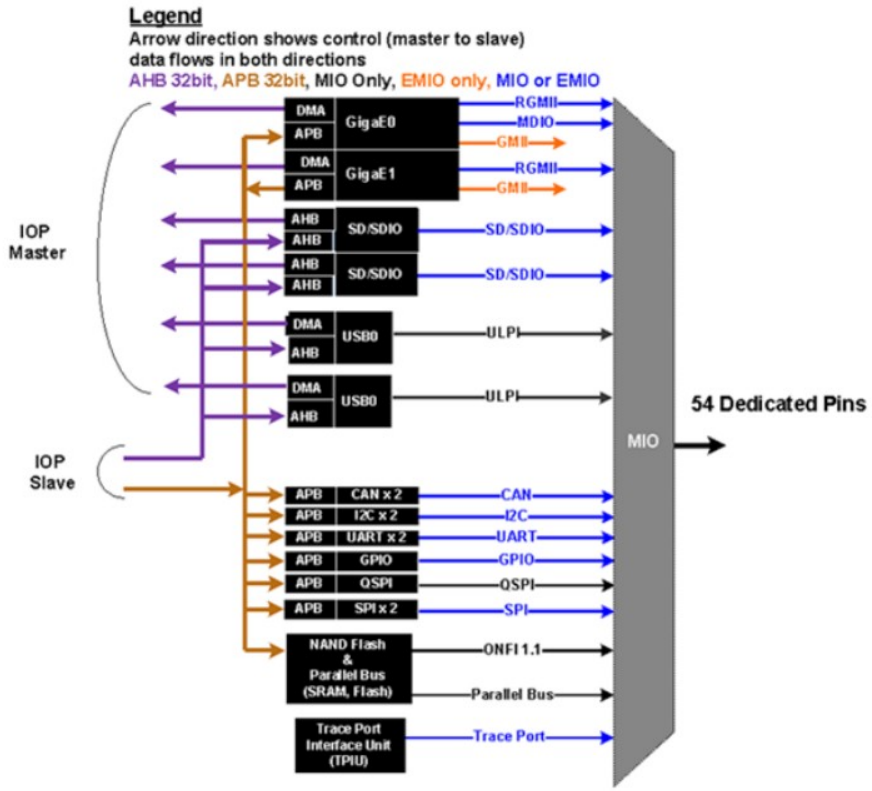
Η δεύτερη ιδέα για επεξεργασία και προβολή βίντεο ορίζεται ως "framebuffer streaming", κατά την οποία δεδομένα εικονοστοιχείων αποθηκεύονται πρώτα στην εξωτερική μνήμη DDR3, έπειτα μεταφέρονται στην προγραμματιζόμενη λογική (FPGA) ώστε να επεξεργαστούν και στην συνέχεια

αποθηκεύονται πάλι στην μνήμη. Τέλος μέσω κάποιου display controller τα επεξεργασμένα δεδομένα προβάλλονται στην οθόνη. Αυτή η μέθοδος επιτρέπει τον διαχωρισμό μεταξύ του ρυθμού ανανέωσης καρέ και της ταχύτητας επεξεργασίας της μονάδας επεξεργασίας αλλά απαιτεί αρκετό εύρος ζώνης (bandwidth) ώστε να γραφούν και διαβαστούν καρέ βίντεο στην εξωτερική μνήμη.



Εικόνα 30: Framebuffer Streaming Architecture. Πηγή Εικόνας: [39]

Επιλέχθηκε η δεύτερη μέθοδος λόγω της ευελιξίας που παρέχει αλλά κυρίως διότι σύμφωνα με το [38] και όπως φαίνεται και στην [Εικόνα 31] το USB είναι από τα λίγα περιφερειακά σε ένα Zynq-7000 SoC που δεν υπάρχει τρόπος να δρομολογηθεί στην προγραμματιζόμενη λογική (FPGA) μέσω της διεπαφής Extended multiplexing I/O (EMIO). Αντ' αυτού σήματα ULPI "ρέουν" μέσω των muliplexing I/O (MIO) ώστε να περάσουν στην μνήμη.



Εικόνα 31: I/O Peripherals System Diagram. Πηγή Εικόνας: [7]

Σε προηγούμενο υποκεφάλαιο (3.1.2) έγινε αναφορά στην υλοποίηση του USB Software Stack κάνοντας χρήση linux οδηγών για USB συσκευές. Γνωρίζοντας τον περιορισμό για την ανάπτυξή σε περιβάλλον linux, η διαδικασία υλοποίησης ενός συστήματος που θα υποστηρίζει προβολή βίντεο/εικόνας μέσω HDMI περιπλέκεται. Παρακάτω παρουσιάζεται η λογική που πρέπει να ακολουθηθεί στην FPGA, από άποψη hardware, καθώς και η λογική που πρέπει να ακολουθηθεί στην εφαρμογή του κατάλληλου οδηγού συσκευής, από άποψη software. Τέλος θα αναλυθεί ο τρόπος που το βίντεο θα προβάλλεται σε μια οθόνη, με ποιον τρόπο δηλαδή θα γίνει η ουσιαστική αξιοποίηση του hardware μέσω του περιβάλλοντος linux.

3.2.1 Ελεγκτές HDMI για χρήση σε SoCs

Εδώ αξίζει να σημειωθεί πως ενώ τα περισσότερα boards παρέχουν στην μεριά της προγραμματιζόμενης λογικής έξοδο βίντεο High-Definition Multimedia Interface (HDMI), δεν χρησιμοποιούν όλα τον ίδιο τρόπο για την διεπαφή βίντεο. Συγκεκριμένα υπάρχουν boards που χρησιμοποιούν εξωτερικές συσκευές ως HDMI πομπούς (transmitter), όπως είναι ο ADV7511 HDMI transmitter [40]. Σε άλλες περιπτώσεις board δεν χρησιμοποιείται κάποιο εξωτερικό chip και ο HDMI connector συνδέεται κατευθείαν στα I/O pins της FPGA. Τέλος υπάρχουν και οι περισσότερο high-end κα-

τηγορίες board που χρησιμοποιούν multi-gigabit serial transceivers GTH και GTX [41]. Μερικά από τα board που εμπεριέχονται στις προαναφερθείσες κατηγορίες φαίνονται στον παρακάτω πίνακα:

Table 5: Διεπαφή HDMI σε διάφορες πλακέτες

ADV7511 HDMI Transmitter[1]	HDMI Connector Directly to I/O Pins[2]	Multi-Gigabit Transceivers[3]
Zynq-7000 SoC ZedBoard	Zynq-7000 SoC Arty Z7	Zynq UltraScale+ MPSoC ZCU102
Zynq-7000 SoC ZC702	Artix-7 FPGA Arty A7	Zynq UltraScale+ MPSoC ZCU104
Zynq-7000 SoC ZC706	Zynq-7000 SoC Zybo Z7	Zynq UltraScale+ MPSoC ZCU106
Artix-7 FPGA AC701	Zynq-7000 SoC Pynq-Z1	
Kintex-7 FPGA KC705	Zynq-7000 SoC Pynq-Z2	
Virtex-7 FPGA VC707		

[1]:Παραθέτονται σύνδεσμοι για κάθε board σε αυτήν τη κατηγορία [42], [43], [44], [45], [46], [47]

[2]:Παραθέτονται σύνδεσμοι για κάθε board σε αυτήν τη κατηγορία [48], [49], [50], [51]

[3]:Παραθέτονται σύνδεσμοι για κάθε board σε αυτήν τη κατηγορία [52], [53], [54]

Η παρούσα διπλωματική εργασία υλοποιεί την μέθοδο που χρησιμοποιεί τον εξωτερικό transmitter ADV7511 κάνοντας χρήση του *Zynq-7000 SoC ZedBoard*, παρ' όλα αυτά θα γίνουν κάποιες γενικές αναφορές και στον τρόπο υλοποίησης που αφορά τις άλλες 2 περιπτώσεις.

Λαμβάνοντας υπόψιν την επιλογή της μεθόδου framebuffer streaming, επιβάλλεται να υλοποιηθεί κάποιο module το οποίο θα διαβάζει και θα μεταφέρει δεδομένα από συγκεκριμένο τμήμα της εξωτερικής μνήμης DDR3 στην FPGA. Ευτυχώς η Xilinx προσφέρει τέτοιου είδους modules υπό την μορφή IP blocks ώστε να ενσωματωθούν εύκολα σε οποιοδήποτε σύστημα. Επιπλέον ένας ελεγκτής χρονισμού βίντεο θα χρειαστεί ώστε να ανιχνεύει τον χρονισμό βίντεο στην είσοδο και να παράγει το ίδιο χρονισμό στην έξοδο. Από εκεί ένας ελεγκτής προβολής βίντεο (display controller) δέχεται τα δεδομένα από την μνήμη και τα σήματα από τον ελεγκτή χρονισμού και χρησιμοποιείται σαν "γέφυρα" μεταξύ του Zynq SoC και του εξωτερικού HDMI transmitter ADV7511. Ο ελεγκτής προβολής βίντεο παρέχει πληροφορίες ελέγχου στο chip για την επεξεργασία του video stream που έχει καταφθάσει από την μνήμη. Για παράδειγμα το εξωτερικό chip ADV7511 στο Virtex-7 FPGA VC707 έχει την δυνατότητα υποστήριξης δεδομένων εισόδου 36-bits ενώ στο ZedBoard υποστηρί-

ζει δεδομένα εισόδου 16-bits. Ο ελεγκτής θα πρέπει να είναι σε θέση να σηματοδοτήσει σωστά το μήκος bit που αρμόζει σε κάθε περίσταση, τα οριζόντια και κάθετα σήματα συγχρονισμού (VSYNC, HSYNC) καθώς και να παράγει παράλληλη διεπαφή βίντεο (DVI/HDMI) από το AXI4-Stream πρωτόκολλο.

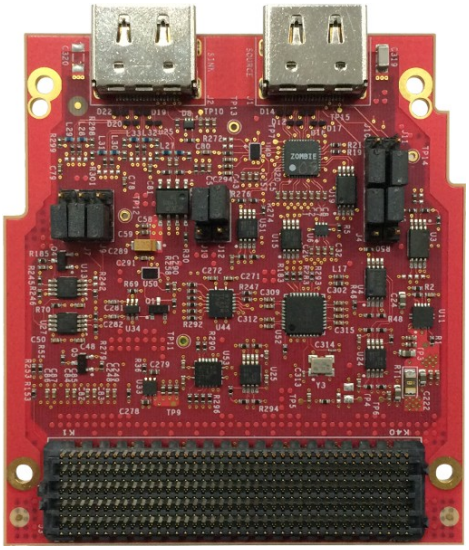
Μελετήθηκαν τρεις τέτοιοι ελεγκτές οι οποίοι διατίθενται ως soft-core IP και ακολουθούν διαφορετική λογική υλοποίησης

I. **Xilinx HDMI_TX**
II. **Xylon LogiCVC-ML**
III. **Analog Devices AXI_HDMI_TX**

Ακολουθεί αναλυτική περιγραφή των παραπάνω:

3.2.1.1 Xilinx HDMI_TX

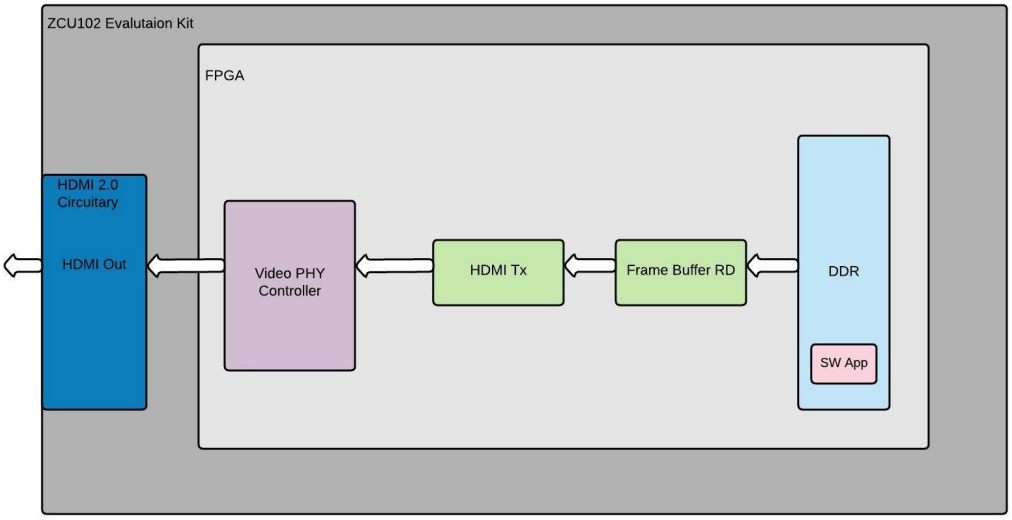
Αυτό το IP core προσφέρεται από την Xilinx ως μέρος του Vivado Design Suite και για την πλήρη πρόσβαση σε λειτουργίες προσομοίωσης και υλοποίησης σε hardware χρειάζεται η αγορά άδειας. Σύμφωνα με το [55], το IP υποστηρίζει όλες τις οικογένειες συσκευών της Xilinx (Ultrascale, Zynq-7000, FPGA 7 series) αρκεί το board να διαθέτει gigabit transceivers καλωδιωμένους στην διεπαφή HDMI(GTX, GTH, GTP,GTY). Τέτοιου είδους πομποδέκτες αφορούν κυρίως την Zynq UltraScale+ MPSoC οικογένεια και για την υλοποίηση σε άλλες συσκευές είναι απαραίτητη η χρήση HDMI κάρτας όπως φαίνεται στην [Εικόνα 32][56].



Εικόνα 32: FMC HDMI 2.0 I/O Module Card. Πηγή Εικόνας: [56]

Για να υλοποιηθεί στο hardware της FPGA αυτό το IP χρειάζονται άλλα δύο cores τα οποία προσφέρονται από την Xilinx. Το Frame Buffer Read IP αναλαμβάνει τον ρόλο του μεταφορέα δεδομένων από την μνήμη DDR στο HDMI_TX core. Εκεί το πρωτόκολλο AXI4-Stream μετατρέπεται σε HDMI stream και μεταφέρεται στο τρίτο IP core της συνδεσμολογίας, το Video Phy Controller,

και μεταδίδεται πλέον ως high-speed serial data stream. Το επόμενο σχήμα αποτελεί παραλλαγή της εικόνας από την wiki σελίδα της Xilinx.[57]

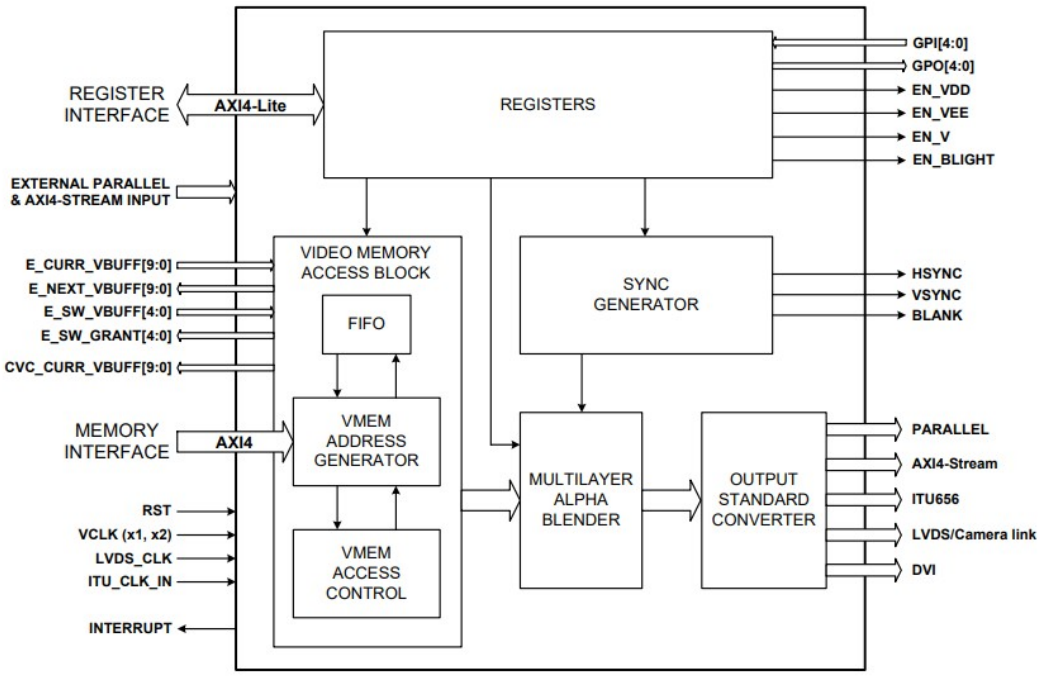


Εικόνα 33: Xilinx HDMI_TX System Implementation

3.2.1.2 Xylon LogiCVC-ML

Αυτό το IP core προσφέρεται από συνεργάτιδα εταιρία της Xilinx ή οποία ονομάζεται Xylon. Η Xylon προσφέρει άδεια αξιολόγησης γι' αυτό το προϊόν, ορίζοντας όμως χρονικό όριο 1 ώρας με την οθόνη να παγώνει μετά την λήξη του περιθωρίου. Για πλήρη πρόσβαση στις λειτουργίες απαιτείται η αγορά μιας αρκετά δαπανηρής άδειας. Ωστόσο το παρών IP core συγκεντρώνει αρκετά χρήσιμα χαρακτηριστικά. Πρόκειται για έναν ελεγκτή βίντεο βελτιστοποιημένο για χρήση με συσκευές Xilinx 7 series FPGA, Ultrascale+ MPSoC και Zynq-7000 SoC. Όλα τα boards που ανήκουν στην κατηγορία ADV7511 Transmitter μπορούν να κάνουν χρήση του συγκεκριμένου IP υποστηρίζοντας πολλαπλές μορφές εξόδου (DVI, AXI4-Stream, Parallel RGB / YUV). Ίσως το κυριότερο χαρακτηριστικό που το κάνει να διαφέρει από τους άλλους δύο ελεγκτές είναι η υποστήριξη απεικόνισης 5 διαφορετικών layer ταυτόχρονα, με δυνατότητα διαμόρφωσης διαφορετικού μεγέθους και θέσης για κάθε επίπεδο (layer). Επιπλέον περιέχει block λογικής το οποίο είναι υπεύθυνο για πρόσβαση στην μνήμη, ανάκτηση δεδομένων και μεταφορά στην προγραμματιζόμενη λογική. Όπως φαίνεται και από το διάγραμμα αρχιτεκτονικής του συγκεκριμένου IP core [Εικόνα 34], η υλοποίηση του σε hardware είναι διαφορετική από αυτή του Xilinx HDMI_TX. Το συγκεκριμένο block έχει δικό του μηχανισμό για την μεταφορά δεδομένων από το τμήμα μνήμης όπως επίσης και μετατροπέα των δεδομένων ανάγνωσης σε ροή δεδομένων (data stream) αποδεκτή από την διεπαφή οθόνης (display interface). Παράγει επίσης τα κατάλληλα οριζόντια και κάθετα σήματα χρονισμού (VSYNC, HSYNC) προς τον εξωτερικό transmitter ADV7511. Στην συνέχεια ο ADV7511 transmitter είναι υπεύθυνος για την σειριοποίηση της εισερχόμενης ροής βίντεο δεδομένων ώστε να τα στείλει στην οθόνη μέσω HDMI. Συνοψίζοντας από τα παραπάνω, η υλοποίηση αυτής της μεθόδου σε hardware απαιτεί μόνο την χρήση ενός block λογικής (logiCVC-ML)[58]. Σύμφωνα με την Xy-

lon απαραίτητη είναι όμως και η χρήση μιας γεννήτριας ρολογιού (clock generator) σε περιπτώσεις όπου το board δεν διαθέτει κάτι αντίστοιχο σε hardware. Για παράδειγμα το Zynq-7000 SoC ZC702 διαθέτει σε hardware το SI570 clock synthesizer, μέσω του οποίου παράγεται ένα ακριβές ρολόι για το βίντεο. Στις περισσότερες περιπτώσεις όπου δεν υπάρχει αντίστοιχο clock synthesizer, όπως στο ZedBoard, η Xylon προτείνει την χρήση ενός δικού της Programmable Clock Generator IP core [59], χωρίς αυτό να διατίθενται δωρεάν και με την άδεια αξιολόγησης να διαρκεί μόλις μερικές εβδομάδες.

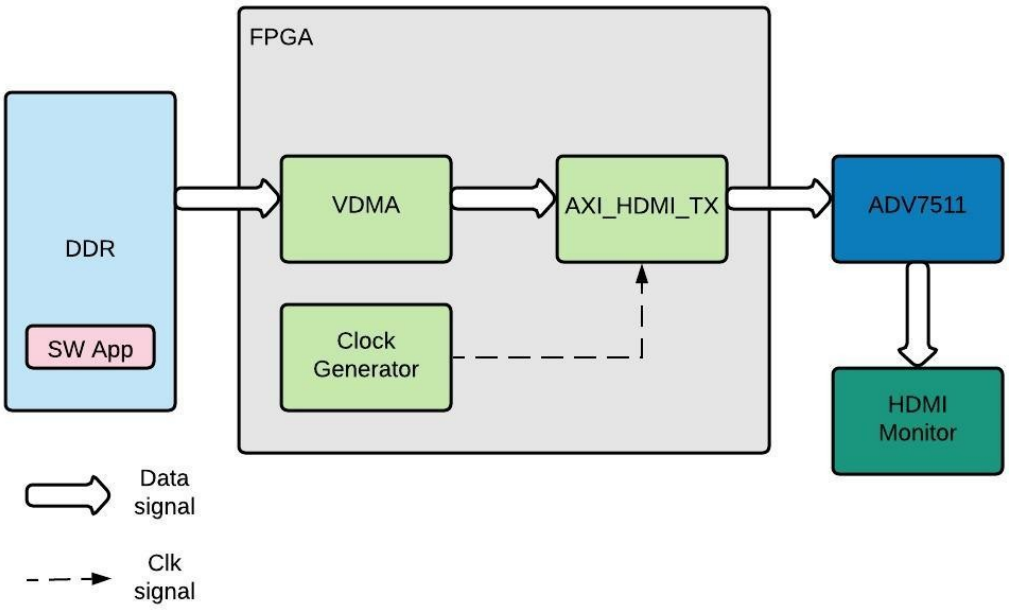


Εικόνα 34: logiCVC-ML Architecture. Πηγή Εικόνας: [58]

3.2.1.3 Analog Devices AXI_HDMI_TX

Αυτό το IP core παρέχεται από μια άλλη συνεργάτιδα εταιρία της Xilinx, την Analog Devices. Η συγκεκριμένη εταιρία είναι και αυτή που παράγει τα ADV7511 chip που βρίσκονται στα boards της Xilinx. Όπως είναι φυσικό αυτή η περίπτωση αφορά κυρίως την κατηγορία ADV7511 HDMI transmitter. Η Analog Devices παρέχει άδεια αξιολόγησης χωρίς περιορισμούς καθώς επίσης και έτοιμα σχέδια αναφοράς όπου επιδεικνύεται η video/audio λειτουργία του εν λόγω transmitter. Παραλλαγή αυτών των σχεδίων αποτελεί και η υλοποίηση του hardware όπως αυτό θα περιγραφεί περισσότερο αναλυτικά στο κεφάλαιο 4. Η παρούσα προσέγγιση χρησιμοποιεί ένα Video Direct Memory Access (VDMA) IP core, το οποίο παρέχεται από την Xilinx και χρησιμοποιεί την διεπαφή AXI4-Master για να διαβάζει καρέ από την εξωτερική μνήμη και να τα μεταφέρει σε μια διεπαφή AXI4-Stream Master (AXI_HDMI_TX IP) στην προγραμματιζόμενη λογική. Με αυτόν τον τρόπο ο επεξεργαστής Zynq εκφορτώνεται από τις λεπτομέρειες για διατήρηση ροής βίντεο εξόδου. Στην συνέχεια αφού τα δεδομένα έχουν μεταφερθεί από το VDMA στο AXI_HDMI_TX IP core, το AXI_HDMI_TX IP core δέχεται σήμα ρολογιού από ένα clock generator IP core , που παρέχεται

από την Analog Devices, ώστε να παράξει τα κατάλληλα οριζόντια και κάθετα σήματα χρονισμού VSYNC και HSYNC. Τέλος το AXI_HDMI_TX IP core είναι υπεύθυνο και για την παραγωγή άλλων σημάτων ελέγχου ως προς τον εξωτερικό ADV7511 transmitter, όπως το μήκος bit που αντιστοιχεί στην εκάστοτε συσκευή.



Εικόνα 35: Analog AXI_HDMI_TX System Implementation

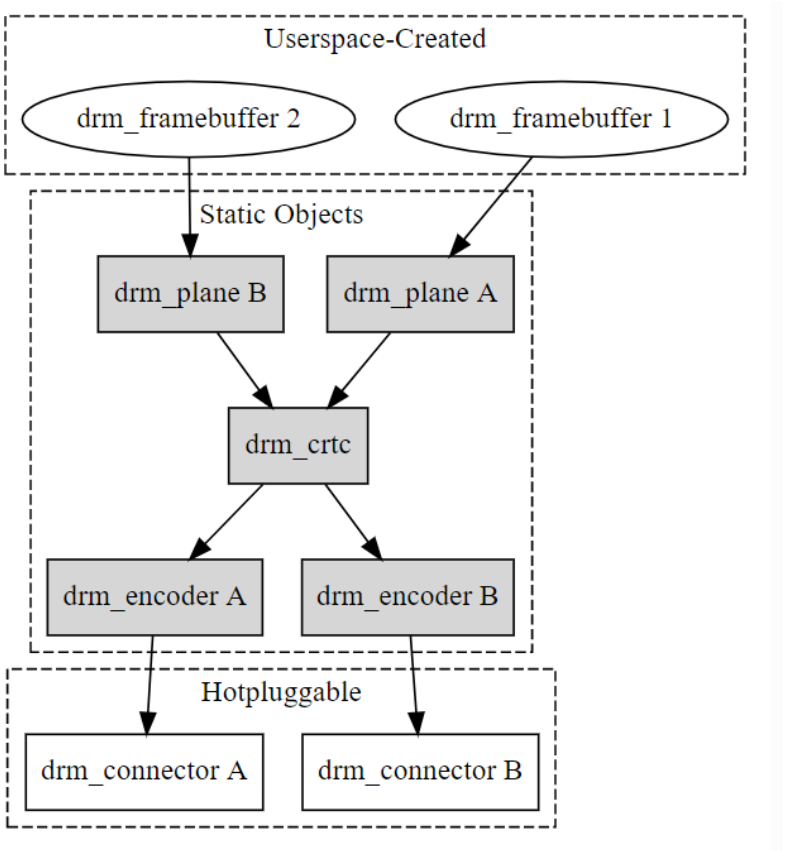
Από τις επιλογές που παρουσιάστηκαν παραπάνω επιλέχθηκε αυτής της Analog Devices AXI_HDMI_TX λόγω της έλλειψης περιορισμών που θέτονται από την εταιρία στην υλοποίηση των IP cores και στη παραχώρηση αδειών.

Table 6: Χαρακτηριστικά HDMI IP cores

XILINX HDMI_TX	*Αφορά κυρίως οικογένειες Zynq Ultrascale+ MPSoC και απαιτεί χρήση άλλων IP cores για μεταφορά δεδομένων από την μνήμη και για την σειριοποίηση δεδομένων προς έξοδο.*
XYLON logiCVC-ML	*Αφορά boards με εξωτερικό ADV7511 transmitter. Είναι ικανό για τις απαραίτητες λειτουργίες μεταφοράς/μετατροπής format και προβολής video από μόνο του. Απαιτεί μόνο αγορά clock generator IP core. Ο driver του, χρειάζεται να κατανέμει κομμάτι μνήμης, για τον framebuffer, στατικά. Η άδεια του εισάγει χρονικούς περιορισμούς χρήσης.*
ANALOG AXI_HDMI_TX	*Αφορά boards με εξωτερικό ADV7511 transmitter. Απαιτούνται IP cores για μεταφορά δεδομένων από την μνήμη και clock generator. Ο driver του, εκχωρεί δυναμικά τη μνήμη, για τον framebuffer. Παρέχεται άδεια χρήσης χωρίς περιορισμούς (ισχύει και για το clock generator).*

3.2.2 Πρόγραμμα Οδήγησης HDMI Συστήματος

Για την μεταφορά του παραπάνω hardware σε περιβάλλον linux και δέσμευση ενός μέρους της μνήμης για αποθήκευση δεδομένων βίντεο (framebuffer) χρειάζονται τα κατάλληλα προγράμματα οδήγησης υλικού. Ο μοντέρνος τρόπος κατά τον οποίο δημιουργείται ένα πλήρες video pipeline, χρησιμοποιώντας μια σειρά από hardware blocks στην FPGA, είναι αυτός που κάνει χρήση ενός συστήματος Direct Rendering Manager/ Kernel Mode Setting (DRM/KMS). Τα κύρια μέρη μιας DRM/KMS συσκευής φαίνονται στο παρακάτω σχήμα.[Εικόνα 36][60] Οι Framebuffers τροφοδοτούν τα planes. Ένα ή περισσότερα planes τροφοδοτούν τα δεδομένα των εικονοστοιχείων (pixel) τους σε ένα CRT controller (CRTC) για ανάμειξη. Ένας CRTC μπορεί να είναι συνδεδεμένος με πολλαπλούς encoders και για έναν ενεργό CRTC πρέπει να υπάρχει τουλάχιστον ένας encoder στη συνδεσμολογία. Τέλος για κάθε έναν ενεργό encoder πρέπει να υπάρχει και ένας connector.



Εικόνα 36: DRM/KMS Display Pipeline Overview. Πηγή Εικόνας: [60]

Framebuffer: Πρόκειται για ένα αντικείμενο το οποίο αποθηκεύει πληροφορίες στην μνήμη αναφορικά με το περιεχόμενο που επρόκειτο να προβληθεί. Οι πληροφορίες που αποθηκεύονται αφορούν:

- την αναφορά στις περιοχές μνήμης που χρησιμοποιούνται
- τη μορφή πλαισίου (frame format) που είναι αποθηκευμένο στην μνήμη. Τα format που χρησιμοποιούνται από το DRM/KMS κυρίως είναι τα RGB,YUV και C8

- την ενεργή περιοχή εντός της περιοχής μνήμης (περιεχόμενο που επρόκειτο να προβληθεί)

Planes: Δεν πρόκειται για ένα εγγενώς block υλικού αλλά για ένα αντικείμενο μνήμης που περιέχει έναν buffer από τον οποίο τροφοδοτείται ο CRTC. Το plane που συνδέεται με τον framebuffer ονομάζεται πρωτεύον plane και κάθε CRTC πρέπει να είναι συσχετισμένο τουλάχιστον με ένα. Δεδομένου ότι ένα plane αποτελεί πηγή του CRTC είναι σε θέση να καθορίσει λειτουργίες βίντεο όπως την ανάλυση οθόνης (πλάτος και ύψος), μέγεθος pixel, μορφή pixel, ρυθμός ανανέωσης, κ.λπ.

CRTC: Κάθε CRTC αντιπροσωπεύει ένα μηχάνημα το οποίο σαρώνει τις περιοχές μνήμης που έχουν δεσμευτεί για τον framebuffer, διαβάζει τα δεδομένα εικονοστοιχείων από εκεί και παράγει σήματα χρονισμού και ανάλυσης (resolution) για το βίντεο. Άμα 2 ή περισσότερα planes είναι συνδεδεμένα με ένα CRTC, τότε η έξοδος του CRTC προς τον encoder έχει προκύψει από την σύνθεση των δεδομένων των εικονοστοιχείων αυτών των planes.

Encoder: Ένας encoder είναι υπεύθυνος για να λαμβάνει δεδομένα εικονοστοιχείων από τον CRTC και να τα μετατρέπει στην κατάλληλη μορφή (format) που αρμόζει στον συνδεδεμένο connector. Ένας connector μπορεί να δεχτεί σήματα από έναν encoder τη φορά και κάθε τύπος connector υποστηρίζει συγκεκριμένες κωδικοποιήσεις. Για παράδειγμα ένας HDMI connector μεταδίδει Transition-minimized differential signaling (TMDS) κωδικοποιημένα δεδομένα και γι' αυτόν τον λόγο χρειάζεται έναν TMDS encoder.[61]

Connector: Ένας connector μεταδίδει τα σήματα που έχουν κωδικοποιηθεί από τον encoder στην εξωτερική οθόνη που είναι συνδεδεμένη στο σύστημα. Η έννοια του connector σε ένα KMS σύστημα αντιστοιχεί σε έναν φυσικό σύνδεσμο (VGA, DVI, HDMI) στο υλικό. Ανίχνευση της σύνδεσης και αφαίρεσης της οθόνης όπως επίσης και η έκθεση των λειτουργιών που υποστηρίζονται από την συνδεδεμένη οθόνη είναι μερικές από τις πληροφορίες που χειρίζεται ο connector.

Η Xilinx όπως και η Analog Devices, προσφέρουν τη δικιά τους προσαρμοσμένη εκδοχή για την δημιουργία μιας DRM/KMS συσκευής παρέχοντας στον χρήστη και τα κατάλληλα προγράμματα οδήγησης υλικού. Σε κάθε περίπτωση τα block υλικού που περιγράφηκαν στο[3.2.1] θα πρέπει να αντιστοιχηθούν στα components της αρχιτεκτονικής DRM/KMS όπως περιγράφηκαν παραπάνω [Εικόνα 36]. Για παράδειγμα ο ADV7511 transmitter αντιμετωπίζεται ως encoder και connector ταυτόχρονα ενώ ο χειρισμός του μεταφορέα (VDMA, logiCVC) από την μνήμη στην προγραμματιζόμενη λογική γίνεται μέσω του CRTC.

3.2.3 Τρόποι Προβολής Βίντεο σε Περιβάλλον Linux

Αφού ολοκληρωθούν τα απαραίτητα βήματα για την υλοποίηση του software και του hardware που περιγράφηκαν στις 2 προηγούμενες κατηγορίες πρέπει να εξεταστούν οι πιθανοί τρόποι πρόσβασης στο υλικό και η προβολή του βίντεο από την οπτική του χρήστη πλέον. Ο επικρατέστερος τρόπος προβολής γραφικών σε περιβάλλον Linux είναι αυτός που ακολουθεί το πρωτόκολλο απεικόνισης X Window System (X11) μαζί με τις βοηθητικές εργαλειοθήκες που το πλαισιώνουν όπως Xlib, GTK+, QT5 και XCB. Πρόκειται στην ουσία για υλοποίηση ένος πλήρες γραφικού περιβάλλοντος Linux (desktop). Παρ' όλα αυτά το επίπεδο αφαιρετικότητας που προσφέρουν αυτές οι επιλογές εί-

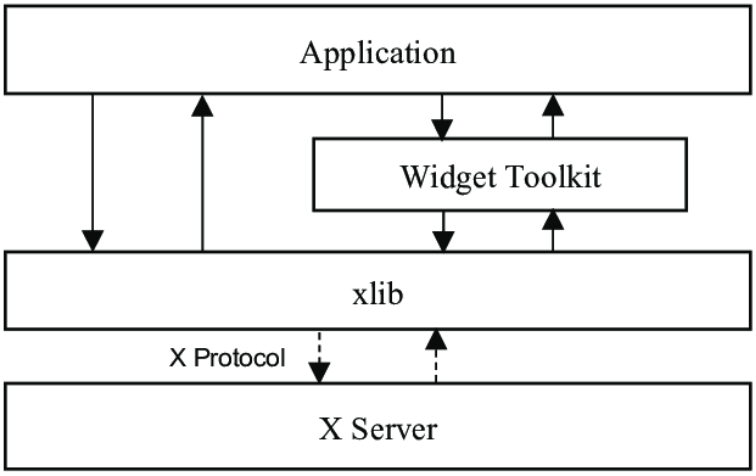
ναι πολύ υψηλό και γιαυτό το λόγο μελετήθηκαν οι τρεις παρακάτω, low-level abstraction, τρόποι. [62]

I. **X Server (X11) Direct Connection**
II. **Direct Rendering Manager (DRM) Dumb Buffers**
III. **Linux Framebuffer Device**

Ακολουθεί αναλυτική περιγραφή των παραπάνω:

3.2.3.1 X Server (X11) Direct Connection

Η σύνδεση με τον X Server (X11) είναι η πιο συνηθισμένη μέθοδος απεικόνισης γραφικών σε σχέση με τις άλλες 2 που θα συζητηθούν. Ωστόσο, το X11 δεν αποτελεί το πιο απλό και εύχρηστο πρωτόκολλο ως προς την υλοποίηση του. Συνήθως, μια εφαρμογή που χρησιμοποιεί το X11 για γραφικά θα χρησιμοποιεί και μια εργαλειοθήκη widget υψηλού επιπέδου, όπως το GTK+ ή το Qt5, το οποίο με τη σειρά του χρησιμοποιεί βιβλιοθήκες Xlib ή XCB για να δημιουργήσει τη σύνδεση και να πραγματοποιήσει τις απαραίτητες ενέργειες για επικοινωνία με τον X Server. Η χρήση των GTK+ ή Qt5 μπορεί να παρακαμφθεί ώστε να γίνει αποκλειστική χρήση των Xlib ή XCB, σε μια προσπάθεια για απλούστευση της εφαρμογής. Κάτι τέτοιο όμως, προϋποθέτει βαθιά κατανόηση του πρωτοκόλλου και τεχνογνωσία από τον προγραμματιστή.



Εικόνα 37: X Window System Architecture

Το πρωτόκολλο του X Server χρησιμοποιεί το μοντέλο πελάτη-διακομιστή (client-server) για επικοινωνία. Αυτό σημαίνει ότι αν ο χρήστης χρησιμοποιήσει sockets, μπορεί να πραγματοποιήσει σύνδεση με τον X Server χωρίς να χρειάζεται να βασιστεί στα Xlib ή XCB για τη διευκόλυνση της επικοινωνίας. Με αυτόν τον τρόπο ο χειρισμός του πρωτοκόλλου X11 γίνεται μέσω του λογισμικού της εφαρμογής. Κατά τη σύνταξη του διακομιστή (X Server), αυτό θα αποτελούσε μια αρκετά επίπονη διαδικασία με το επιτυχές αποτέλεσμα να φαντάζει σχεδόν αδύνατο, δεδομένου του εύρους των λειτουργειών που αναμένεται να χειριστεί ο X Server. Ωστόσο, η σύνταξη ενός πελάτη (client) είναι μια πολύ απλούστερη διεργασία, καθώς μόνο ορισμένα μέρη του πρωτοκόλλου θα πρέπει να

εφαρμοστούν. Δεδομένης της πολυπλοκότητας και της απαιτούμενης, υψηλού επιπέδου, αφαιρετικότητας αυτή η προσέγγιση απορρίφθηκε.

3.2.3.2 Direct Rendering Manager (DRM) Dumb Buffers

Το DRM που περιγράφηκε προηγουμένως [σελ. 51] σε επίπεδο user-space προσφέρει ένα χαρακτηριστικό που ονομάζεται "dumb buffers", το οποίο είναι ουσιαστικά ένας framebuffer. Υποτίθεται ότι είναι εύκολος στη ρύθμιση του, ωστόσο μετά από σχετική ενασχόληση μαζί του έγινε αντιληπτή η πολυπλοκότητα του. Οι περισσότερες εφαρμογές στο user-space χρησιμοποιούν τη βιβλιοθήκη libdrm προκειμένου να διευκολύνουν τη διεπαφή τους με το υποσύστημα DRM. Αυτή η βιβλιοθήκη πρόκειται απλώς για ένα wrapper που παρέχει μια συνάρτηση, γραμμένη σε γλώσσα C, για κάθε system call (ioctl) του DRM API, καθώς και σταθερές, δομές και άλλα βοηθητικά στοιχεία. Η χρήση του libdrm όχι μόνο αποτρέπει την έκθεση του πυρήνα απευθείας σε εφαρμογές, αλλά υιοθετεί τα συνήθη πλεονεκτήματα της επαναχρησιμοποίησης και της κοινής χρήσης κώδικα μεταξύ προγραμμάτων. Όμως με αυτόν τον τρόπο εισάγει και ένα υψηλό επίπεδο αφαιρετικότητας που μπορεί να επηρεάσει την απόδοση. Σε μια προσπάθεια για διατήρηση όσο το δυνατόν χαμηλό επίπεδο αφαιρετικότητας απορρίφθηκε και αυτή η προσέγγιση.

3.2.3.3 Linux Framebuffer Device

Ο οδηγός DRM/KMS που αναφέρεται στη σελίδα 51 καταχωρεί ένα Linux framebuffer (fbdev). Πρόκειται για έναν ανεξάρτητο-υλικού επίπεδο αφαιρετικότητας γραφικών, για εμφάνιση γραφικών σε κάποια οθόνη, συνήθως μέσω της κονσόλα του συστήματος. Ο χρήστης αποκτά πρόσβαση σε αυτό κάνοντας χρήση του device file “dev/fb0”, το οποίο στην πραγματικότητα αντιπροσωπεύει την διεπαφή για τον πρόγραμμα οδήγησης της συσκευής (device driver). Τα προγράμματα οδήγησης framebuffer είναι ιδιαίτερα χρηστικά για τα ενσωματωμένα συστήματα, όπου η ποσότητα κύριας μνήμης που χρησιμοποιείται (memory footprint) είναι απαραίτητο να είναι η ελάχιστη δυνατή. Στον πυρήνα του, ένα πρόγραμμα οδήγησης framebuffer εφαρμόζει τις λειτουργίες mode setting και την προαιρετική επιτάχυνση 2D. Το mode setting συνίσταται στη διαμόρφωση του framebuffer για λήψη εικόνας στην οθόνη. Αυτό περιλαμβάνει την επιλογή της ανάλυσης βίντεο και των ρυθμών ανανέωσης. Τα προγράμματα οδήγησης framebuffer μπορούν να παρέχουν βασική επιτάχυνση 2D που χρησιμοποιείται για την επιτάχυνση της κονσόλας Linux. Το Linux framebuffer διασυνδέεται με το υλικό γραφικών και εμφανίζει απευθείας το βίντεο σε επίπεδο Linux kernel, γεγονός που θα πρέπει να ελαχιστοποιεί την καθυστέρηση (latency) και να έχει καλή απόδοση στα frames per second (fps). Το framebuffer αποδεικνύεται η απλούστερη και ευκολότερη στη χρήση προσέγγιση, κατά την εξέταση των τριών εναλλακτικών λύσεων. Δεδομένων της προσπάθειας για καλή απόδοση στο τελικό σύστημα, όσον αφορά τα fps και latency, καθώς και την διατήρηση της αφαιρετικότητας σε χαμηλό επίπεδο, επιλέχθηκε η παρούσα προσέγγιση για την υλοποίηση του συστήματος.

3.3 Αλγόριθμος Αναγνώρισης Ακμών Sobel

Η κατάτμηση εικόνων είναι μια ορολογία που χρησιμοποιείται όταν γίνεται αναφορά στην διαδικασία διαίρεσης μιας ψηφιακής εικόνας σε πολλαπλά τμήματα, με σκοπό την απλοποίηση ή την αλλα-

γή της αναπαράστασης της εικόνας σε μια πιο εύκολη και ουσιαστική εκδοχή για ανάλυση. Αποτέλεσμα της εφαρμογής της είναι κυρίως ο εντοπισμός αντικειμένων και ορίων, όπως γραμμές ή ακμές. Πρόκειται στην ουσία για την διαδικασία εκχώρησης μιας ετικέτας σε κάθε εικονοστοιχείο, έτσι ώστε εικονοστοιχεία με την ίδια ετικέτα να μοιράζονται ορισμένα χαρακτηριστικά.[63]

3.3.1 Αναγνώριση ακμών

Στην κατάτμηση εικόνων πολύ σημαντικό ρόλο παίζει η λειτουργία αναγνώρισης ακμών. Η ανίχνευση ακμών περιλαμβάνει μια ποικιλία από μαθηματικές μεθόδους που στοχεύουν στην αναγνώριση σημείων μιας ψηφιακή εικόνα όπου η φωτεινότητά της αλλάζει απότομα ή εμφανίζει ασυνέχειες. Τα σημεία στα οποία αλλάζει έντονα η φωτεινότητα της εικόνας, τα σημεία δηλαδή που εντοπίζεται ακραία μεταβολή στις εντάσεις των εικονοστοιχείων, ονομάζονται άκρα. Η ανίχνευση ακμών είναι ένα θεμελιώδες εργαλείο στην επεξεργασία εικόνας, τη μηχανική όραση και την υπολογιστική όραση, ιδιαίτερα στους τομείς της ανίχνευσης χαρακτηριστικών και της εξαγωγής χαρακτηριστικών. Ο σκοπός της ανίχνευσης αιχμηρών αλλαγών στη φωτεινότητα μιας εικόνας είναι η καταγραφή σημαντικών γεγονότων και αλλαγών. Διαπιστώνεται ότι κάτω από γενικές υποθέσεις, οι ασυνέχειες της φωτεινότητας σε μια εικόνα αντιστοιχούν σε: [64]

- μεταβολές στο φωτισμό της σκηνής
- ασυνέχειες βάθους
- αλλαγές στις ιδιότητες των υλικών
- ασυνέχειες στον προσανατολισμό της επιφάνειας

Στην ιδεατή περίπτωση, η εφαρμογή ενός ανιχνευτή ακμών μπορεί να οδηγήσει σε ένα σύνολο συνδεδεμένων καμπυλών που υποδεικνύουν τα όρια των αντικειμένων και των επιφανειών μίας εικόνας. Η εφαρμογή ενός τέτοιου αλγορίθμου μπορεί να μειώσει σημαντικά την ποσότητα των δεδομένων προς επεξεργασία με αποτέλεσμα την δυνατότητα φιλτραρίσματος πληροφοριών που μπορούν να θεωρηθούν λιγότερο σημαντικές, διατηρώντας παράλληλα τις σημαντικές δομικές ιδιότητες της εικόνας. Εάν η διαδικασία της ανίχνευσης ακμών παρουσιάσει καλό ποσοστό επιτυχίας, η ερμηνεία των πληροφοριών της εικόνας απλοποιείται. Στην πραγματικότητα όμως, είναι αρκετά δύσκολο να αποκτηθούν ιδανικές ακμές σε εικόνες της καθημερινής ζωής διότι σε πολλές περιπτώσεις οι ακμές που εντοπίζονται δεν συνδέονται ώστε να σχηματίσουν τμήματα καμπυλών, όπως επίσης και σύνηθες φαινόμενο είναι ο εντοπισμός ψευδών ακμών που δεν αντιστοιχούν σε σημεία ενδιαφέροντος.[65] Η ανίχνευση ακμών είναι ένα από τα θεμελιώδη βήματα στην επεξεργασία, την ανάλυση και την αναγνώριση προτύπων της εικόνας.

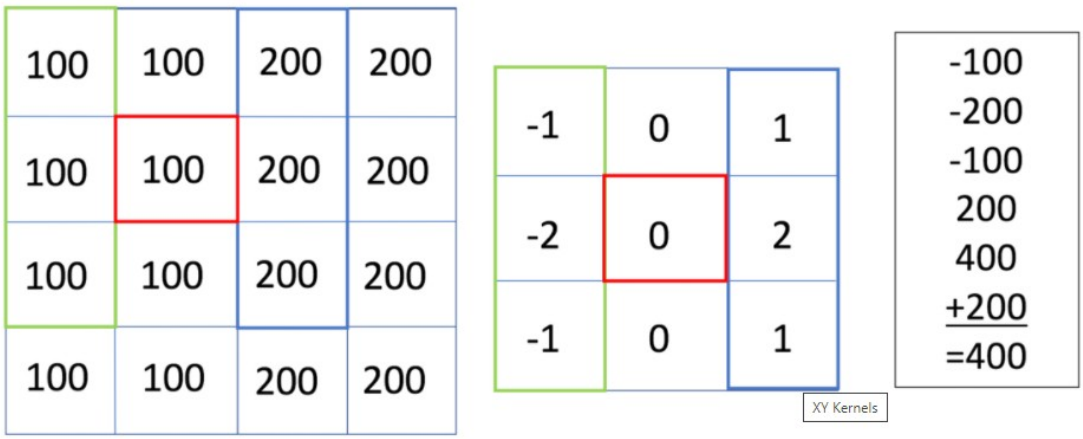
3.3.2 Τελεστής Ανίχνευσης Ακμών Sobel

Ο τελεστής Sobel υπολογίζει προσεγγιστικά, στο δισδιάστατο χώρο, την κλίση (gradient) της έντασης σε κάθε σημείο για μια εικόνα σε αποχρώσεις του γκρι που έχει δοθεί ως είσοδος. Το αποτέλεσμα εφαρμογής το sobel δείχνει πόσο "απότομα" ή "ομαλά" η εικόνα αλλάζει σε αυτό το σημείο και συνεπώς πόσο πιθανό είναι το τμήμα της εικόνας που εφαρμόζεται σε κάθε στιγμή να αντιπροσωπεύει ένα άκρο. Ο τελεστής άκρων Sobel χρησιμοποιεί ένα ζεύγος μασκών 3x3 που συνελίσο-

νται με την αρχική εικόνα εισόδου και μέσω της μίας μάσκας (G_x) εκτιμάται η κλίση στην οριζόντια κατεύθυνση ενώ μέσω της άλλης μάσκας (G_y) εκτιμάται η κλίση στην κάθετη κατεύθυνση.

$$G_x=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix}, \quad G_y=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ +1 & +2 & +1\end{bmatrix}$$

Μια συνέλιξη είναι συνήθως πολύ μικρότερη από την πραγματική εικόνα. Ως αποτέλεσμα, η μάσκα ολισθαίνει πάνω από την εικόνα μεταβάλλοντας ένα εικονοστοιχείο την φορά. Η μάσκα μετακινείται σε μια περιοχή όπου η εικόνα εισόδου αλλάζει με την τιμή του μεταβαλλόμενου εικονοστοιχείου και στη συνέχεια μετατοπίζεται κατά ένα εικονοστοιχείο προς τα δεξιά και συνεχίζει προς τα δεξιά μέχρι να φτάσει στο τέλος της σειράς. Τότε ξεκινά ξανά στην αρχή της επόμενης σειράς.[66]



Εικόνα 38: Συνέλιξη πυρήνα για εντοπισμό ακμής στην οριζόντιο κατεύθυνση. Πηγή Εικόνας: [66]

Μερικά από τα πλεονεκτήματα που καθιστούν ευνοϊκότερη τη χρήση Sobel για τον εντοπισμό ακμών σε σχέση με άλλους ανιχνευτές ακμής είναι τα εξής:[67]

Προσανατολισμός ακμής: Η γεωμετρία του τελεστή καθορίζει μια χαρακτηριστική κατεύθυνση η οποία είναι πιο ευαίσθητη στα άκρα. Οι τελεστές μπορούν να βελτιστοποιηθούν για να αναζητήσουν οριζόντια, κάθετα ή διαγώνια άκρα.

Περιβάλλον θορύβου: Η ανίχνευση ακμών είναι δύσκολη σε θορυβώδεις εικόνες, καθώς τόσο ο θόρυβος όσο και οι ακμές περιέχουν περιεχόμενο υψηλής συχνότητας. Η προσπάθεια μείωσης του θορύβου έχει ως αποτέλεσμα θολές και παραμορφωμένες ακμές. Οι τελεστές που χρησιμοποιούνται σε θορυβώδεις εικόνες είναι τυπικά μεγάλοι, έτσι ώστε κατά την εφαρμογή να μπορούν να λάβουν αρκετά δεδομένα για να μειώσουν τα τοπικά θορυβώδη εικονοστοιχεία. Αυτό έχει ως αποτέλεσμα λιγότερο ακριβή εντοπισμό των ανιχνευμένων άκρων.

Δομή ακμών: Δεν περιλαμβάνουν όλες οι ακμές μια βαθμιαία αλλαγή στην ένταση. Εφέ όπως η διάθλαση ή η κακή εστίαση μπορεί να οδηγήσει σε αντικείμενα με όρια που ορίζο-

νται από μια σταδιακή αλλαγή στην ένταση. Ο χειριστής επιλέγεται να ανταποκρίνεται σε μια τέτοια σταδιακή αλλαγή σε αυτές τις περιπτώσεις. Οι νεότερες τεχνικές που βασίζονται σε κύματα χαρακτηρίζουν πραγματικά τη φύση της μετάβασης για κάθε άκρο προκειμένου να διακρίνουν, για παράδειγμα, άκρα που σχετίζονται με τα μαλλιά από άκρα που σχετίζονται με ένα πρόσωπο

3.3.3 Διαδικασία Ανίχνευσης Ακμών Sobel

Η διαδικασία της ανίχνευσης ακμών μπορεί να χαρακτηριστεί από τα εξής βήματα:

1. Αποδοχή εικόνας εισόδου I
2. Μεταφορά της RGB εικόνας σε grayscale
3. Εφαρμογή Gaussian φίλτρου για τη μείωση θορύβου και την εξομάλυνση της εικόνας
4. Εφαρμογής συνέλιξης της μάσκας Sobel με την εικόνα για τον υπολογισμό των αλλαγών στον οριζόντιο άξονα:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

5. Εφαρμογής συνέλιξης της μάσκας Sobel με την εικόνα για τον υπολογισμό των αλλαγών στον κάθετο άξονα:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

6. Για κάθε εικονοστοιχείο της εικόνας υπολογίζεται η απόλυτη τιμή της κλίσης (gradient) συνδυάζοντας τα αποτελέσματα των δύο προηγούμενων βημάτων:

$$G = \sqrt{G_x^2 + G_y^2}$$

ή

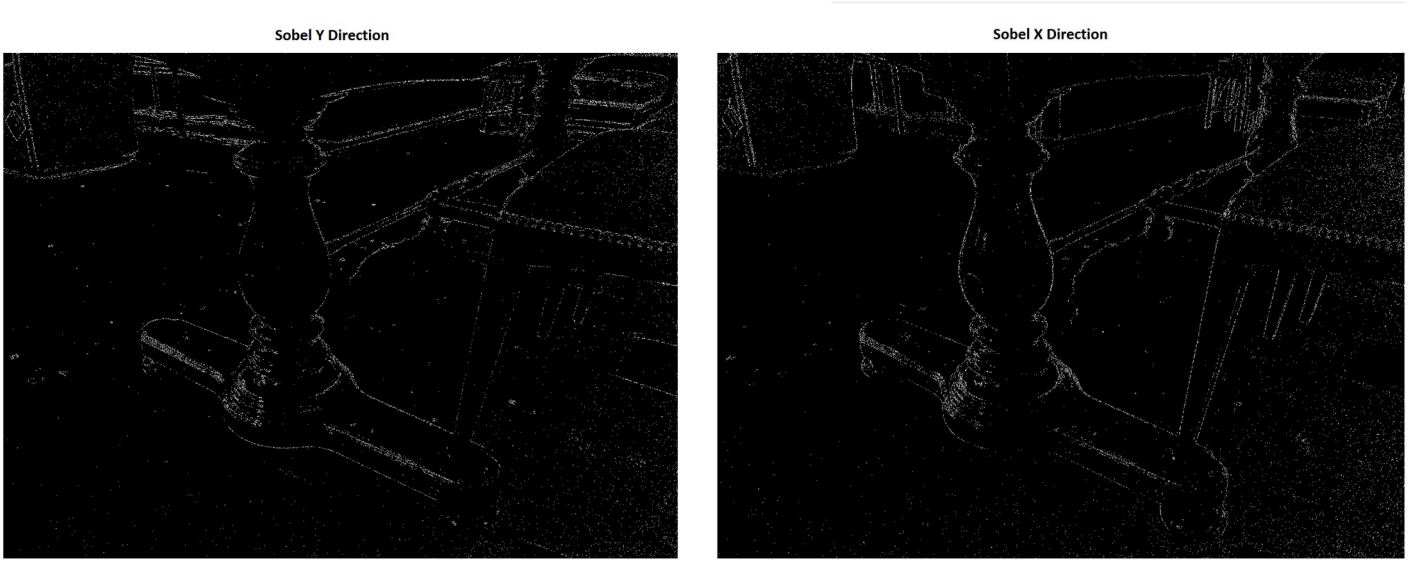
$$G = |G_x| + |G_y|$$

3.3.4 Παραδείγματα Εφαρμογής του Αλγορίθμου

Παρουσιάζεται ένα παράδειγμα επίδειξης της εύρεσης ακμών Sobel.[68] Μια εικόνα λαμβάνεται και μετατρέπεται σε grayscale.

Εικόνα 39: Original and Grayscale Image. Πηγή Εικόνας: [66]

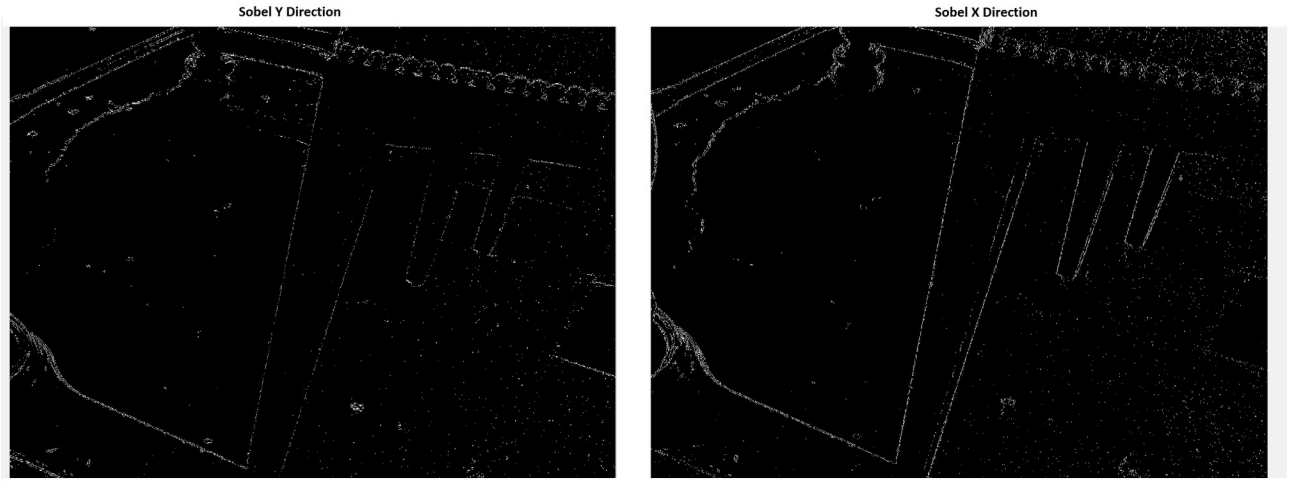
Αφού εφαρμοστεί το φίλτρο Sobel για την οριζόντια και κάθετη κατεύθυνση προκύπτουν οι παρακάτω εικόνες:



Εικόνα 40: Sobel on Vertical Changes (left) and Sobel on Horizontal Changes (right). Πηγή Εικόνας:[66]

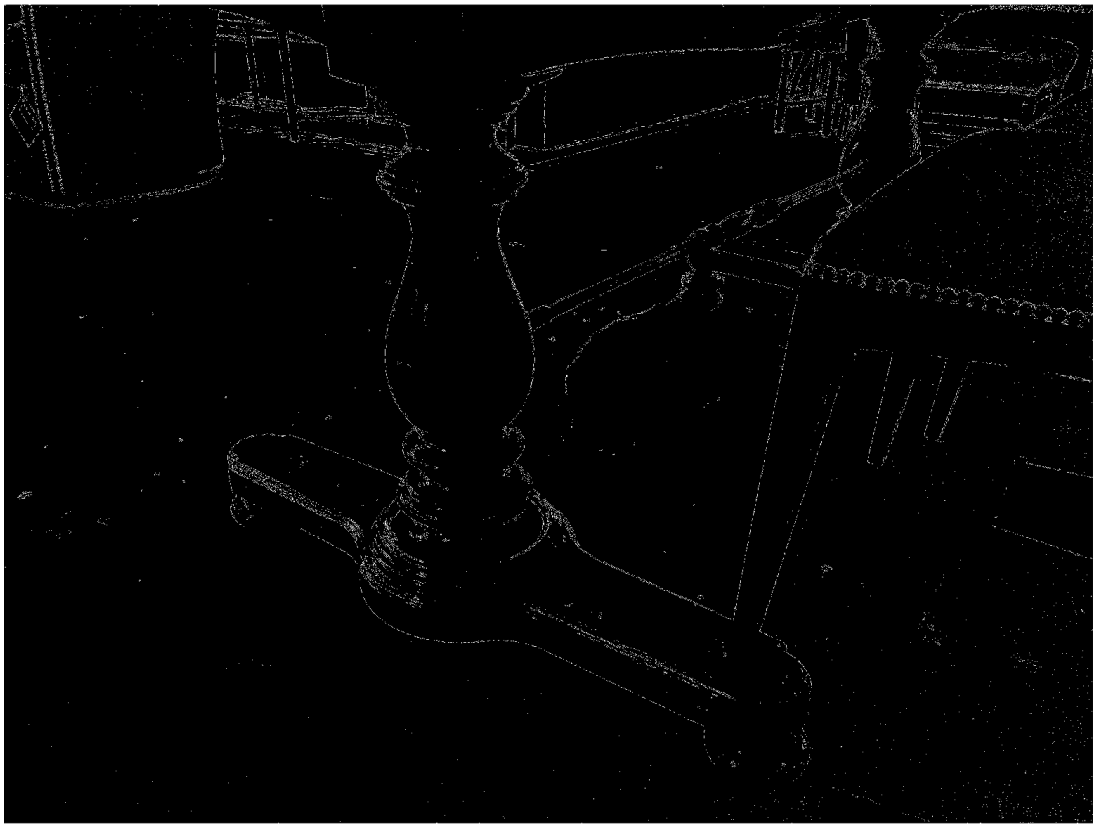
Οι εικόνες που παρουσιάζονται παραπάνω είναι σε μεγάλο βαθμό παρόμοιες και αυτό οφείλεται στο γεγονός ότι πολλές ακμές στην εικόνα υπάρχουν υπό γωνία στον οριζόντιο και κάθετο άξονα. Ωστόσο, στην εικόνα Sobel Y Direction, το πόδι της καρέκλας στα δεξιά δεν φαίνεται όσο καθαρά φαίνεται στην εικόνα Sobel X Direction. Αυτό συμβαίνει επειδή όταν ως κατεύθυνση αναφοράς χρησιμοποιείται η κάθετη, η σάρωση γίνεται από πάνω προς τα κάτω με αποτέλεσμα να εντοπίζονται μόνο ακμές που είναι οριζόντια στην εικόνα. Από την άλλη πλευρά, η εικόνα Sobel X Direction θα ανιχνεύσει τα άκρα του ποδιού της καρέκλας επειδή η εικόνα θα υποβληθεί σε επεξεργασία από αριστερά προς τα δεξιά χρησιμοποιώντας ένα διαφορετικό φίλτρο. Αυτό θα πιάσει το αριστερό και το δεξί άκρο του ποδιού της καρέκλας, επειδή αυτό θα δει τη διαφορά στις εντάσεις

των αντικειμένων που είναι κάθετα ευθυγραμμισμένα στην εικόνα. Οι παρακάτω εικόνες δείχνουν αυτή τη διάκριση.



Εικόνα 41: Παρουσίαση διαφοράς στις δύο κατευθύνσεις. Πηγή Εικόνας: [66]

Τελικά τα αποτελέσματα της εφαρμογής των δύο φίλτρων συνδυάζονται και προκύπτει η παρακάτω ακριβέστερη αναπαράσταση όλων των ακμών (κάθετη και οριζόντια κατεύθυνση) στην εικόνα:



Εικόνα 42: Result of Sobel Edge Detection. Πηγή Εικόνας: [66]

4 Υλοποίηση Συστήματος

Σε αυτό το κεφάλαιο, παρουσιάζεται με πλήρη λεπτομέρεια η αρχιτεκτονική που έχει σχεδιαστεί για αυτήν την εργασία. Αρχικά δίνεται μια συνοπτική περιγραφή της αναπτυξιακής πλακέτας Zedboard και των εργαλείων της Xilinx που χρησιμοποιήθηκαν. Επιπλέον παρουσιάζεται το μοντέλο διαχείρισης συσκευών ενός συστήματος Linux, με σκοπό να κατανοήσει ο αναγνώστης τους διαφορετικούς τύπους driver που θα χρησιμοποιηθούν. Στη συνέχεια, στην ενότητα 4.4 παρουσιάζεται η γενική αρχιτεκτονική του συστήματος, η οποία περιλαμβάνει τη διαμόρφωση όλων των μονάδων του hardware (PS & PL) και τη μεταφορά τους σε περιβάλλον Linux. Σε αυτό το σύστημα προστίθεται και ένα block λογικής Sobel, υλοποιημένο σε περιβάλλον HLS, ώστε να επιδειχθεί η δυνατότητα επαναχρησιμοποίησης του συστήματος αλλάζοντας απλά τη λογική αυτού του block.

4.1 Εργαλεία που χρησιμοποιήθηκαν

Η Xilinx παραδίδει κάθε χρόνο 3 με 4 διαφορετικές εκδόσεις για τα εργαλεία της, οι οποίες μπορεί να παρουσιάζουν, σε ορισμένες περιπτώσεις, προβλήματα συμβατότητας μεταξύ τους. Γι' αυτόν τον λόγο προτείνεται η επιλογή μιας έκδοσης μεταξύ των εργαλείων της. Για την παρούσα υλοποίηση χρησιμοποιήθηκε έκδοση 2017.4 για όλες τις πλατφόρμες.

4.1.1 Vivado Design Suite

Το Vivado αποτελεί εργαλείο της Xilinx το οποίο, σε σχέση με τον προκάτοχό του "Xilinx ISE", χρησιμοποιεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης(intergrated design environment = "IDE") προσφέροντας έτσι επιπρόσθετα χαρακτηριστικά για ανάπτυξη με FPGA και system on a chip(SoC) boards. Το ολοκληρωμένο περιβάλλον ανάπτυξης του Vivado παρέχει μια γραφική διεπαφή (GUI) στον χρήστη ώστε να μπορεί να αλληλεπιδρά εύκολα και άμεσα με το σχέδιο προς υλοποίηση. Περιέχει ένα κατάλογο από προδιαμορφωμένες λογικές συναρτήσεις οι οποίες ονομάζονται Intellectual Property(IP) cores και μπορούν να κυμανθούν σε διάφορα επίπεδα πολυπλοκότητας, όπως από απλούς αριθμητικούς τελεστές έως και πολυπλέκτες, μνήμη, φίλτρα. Τα IP cores χωρίζονται σε 2 κύριες κατηγορίες: "hard cores" και "soft cores". Τα hard cores δεν δύναται να τροποποιηθούν αφού αναπαριστούν "υλικά (hard)" χαρακτηριστικά και λειτουργικές μονάδες της FPGA. Τα soft cores από την άλλη είναι παραμετροποιήσημα και μπορούν να υπάρξουν είτε ως netlist, δηλαδή μια λίστα από λογικές πύλες μαζί με τις σχετικές διασυνδέσεις, είτε ως κώδικας γλώσσας περιγραφής υλικού (hardware description language). Όταν τα επιθυμητά IP cores, παραμετροποιηθούν και διασυνδεθούν, πραγματοποιούνται τα Synthesis, Implementation και Design Rule Check τεστ. Όταν τα τεστ ολοκληρωθούν με επιτυχία παράγεται ένα δυαδικό αρχείο (bitstream) το οποίο περιέχει όλη την λογική του hardware συστήματος.

Κατά την διαδικασία της **Synthesis** στο Vivado πραγματοποιείται μετατροπή του Register-transfer level (RTL) σχεδίου σε αναπαράσταση επιπέδου λογικών πυλών. Η διαδικασία είναι "timing-driven" και συμβάλει στην βελτιστοποίηση της χρήσης της μνήμης και της επίδοσης του συστήματος. Με το πέρας του Synthesis τεστ, δημιουργείται μια αναφορά που παρέχει στον χρήστη χρήσι-

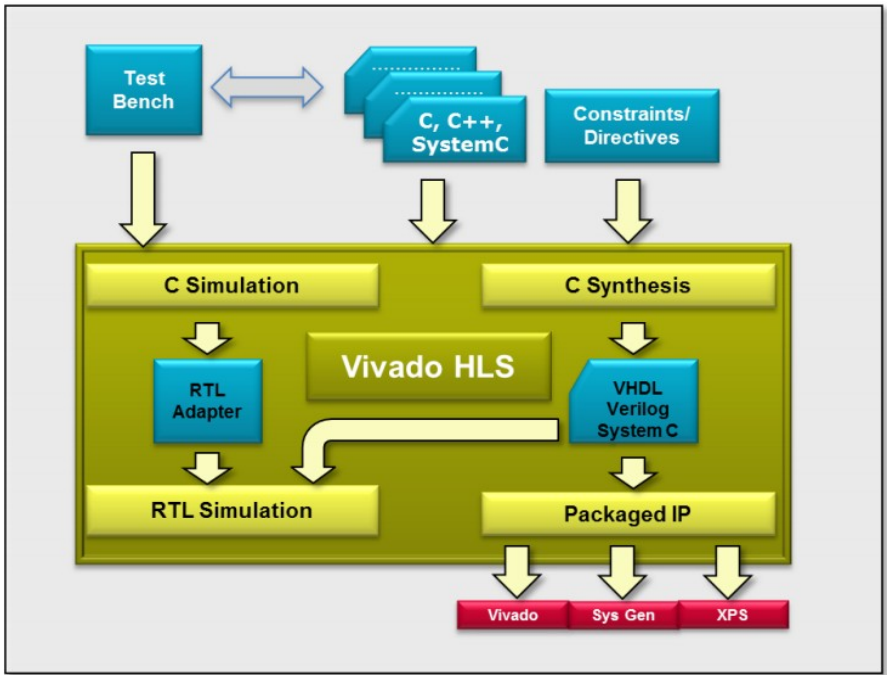
μες πληροφορίες σχετικά με την αξιοποίηση της μνήμης, τον χρονισμό, εισόδους/εξόδους και άλλα χαρακτηριστικά που απορρέουν από το συγκεκριμένο σχέδιο.[69]

Κατά την διαδικασία του **Implementation** στο Vivado εκτελούνται όλα τα απαραίτητα βήματα για την τοποθέτηση και δρομολόγηση της netlist στους πόρους της FPGA, λαμβάνοντας υπόψη τους λογικούς,φυσικούς και χρονικούς περιορισμούς που έχουν τεθεί από τον σχεδιασμό του συστήματος. Με αυτόν τον τρόπο, εξασφαλίζεται ότι το κάθε IP core έχει ανατεθεί στην κατάλληλη περιοχή της FPGA, σύμφωνα πάντα με τους προαναφερθέντες περιορισμούς.[70]

Αφού ολοκληρωθούν τα 2 προηγούμενα τεστ ακολουθεί αυτό του **Design Rule Check**, με το οποίο διασφαλίζεται η τήρηση μια σειράς από κανόνες με βάση τις προτεινόμενες παραμέτρους. Με αυτό το τρόπο εντοπίζονται κρίσιμες προειδοποιήσεις και σφάλματα που μπορεί να προκύψουν σε φυσικό επίπεδο.[71]

4.1.2 Vivado HLS

Το εργαλείο Vivado HLS της Xilinx παρέχει ένα υψηλό επίπεδο αφαιρετικότητας, μεταμορφώνοντας (**Synthesis**) συναρτήσεις γραμμένες σε γλώσσες υψηλού επιπέδου(C/C++) σε κώδικα χαμηλού επιπέδου (VHDL,Verilog) και εν συνεχεία σε IP cores. Με αυτόν τον τρόπο μπορούν να ενσωματωθούν σε οποιοδήποτε hardware σύστημα μέσω του Vivado IDE. Ανάμεσα στις επιλογές γλώσσας C, C++, SystemC και Open Computing Language(OpenCL) API C kernel που παρέχει το εργαλείο, επιλέχθηκε αυτή της C++, αφού φαίνεται να είναι καλύτερα υποστηριζόμενη, παρέχοντας αρκετά παραδείγματα και πολύ χρήσιμες βιβλιοθήκες όπως αυτή της OpenCV που θα χρησιμοποιηθεί εκτεταμένα. Πριν την μετατροπή σε RTL(γλώσσα HDL) και εξαγωγή σε IP block, δίνεται η δυνατότητα για προσομοίωση του κώδικα C μέσω ενός test bench. Με αυτόν τον τρόπο μπορεί να εξακριβωθεί η επιθυμητή και αναμενόμενη λειτουργία του κώδικα.



Εικόνα 43: Vivado HLS Overview. Πηγή Εικόνας: [72]

Μετά την εκτέλεση της λειτουργίας **Synthesis**, το Vivado hls παράγει μία αναφορά η οποία περιέχει μετρήσεις για την επίδοση του συστήματος ώστε ο σχεδιαστής να μπορεί να κρίνει αν το σύστημα πληροί της απαιτήσεις. Ακολουθεί παρουσίαση και περιγραφή αυτών των μετρήσεων:[72]

- **Area:** Η ποσότητα πόρων υλικού που απαιτείται για την υλοποίηση του συστήματος βάση των διαθέσιμων πόρων στην επιλεγμένη FPGA, συμπεριλαμβανομένων των look-up tables(LUT), καταχωρητών, block ram και DSP48.
- **Latency:** Ο αριθμός κύκλων ρολογιού που απαιτείται από την συνάρτηση για τον υπολογισμό όλων των τιμών εξόδου.
- **Initiation interval(II):** Ο αριθμός των κύκλων ρολογιού προτού η συνάρτηση μπορεί να δεχτεί νέα δεδομένα.
- **Loop iteration latency:** Ο αριθμός των κύκλων ρολογιού που απαιτείται για να ολοκληρωθεί μια επανάληψη βρόχου.
- **Loop initiation interval:** Ο αριθμός των κύκλων ρολογιού πριν από την επόμενη επανάληψη του βρόχου.
- **Loop latency:** Ο αριθμός των κύκλων που χρειάζεται για να εκτελεστούν όλες οι επαναλήψεις του βρόχου.

Ο σχεδιασμός με το συγκεκριμένο εργαλείο συχνά καταλήγει ως μια διαμάχη για βελτιστοποίηση, ανάμεσα στη χρήση μεγάλης ποσότητας πόρων και υψηλή απόδοση του υλικού ή στην εξοικονόμηση πόρων θυσιάζοντας μέρος της απόδοσης του υλικού. Το Vivado hls κατά την διαδικασία της Synthesis προσπαθεί να πετύχει τον καλύτερο δυνατό σχεδιασμό RTL τείνοντας συχνά προς την 2η προσέγγιση που προαναφέρθηκε. Η δυνατότητα για παρέμβαση του χρήστη κατά την διαδικασία σχεδιασμού και γραψίματος του κώδικα C++ δίνεται με έναν πολύ βολικό τρόπο, τα directives. Με τις τεχνικές που θα περιγραφούν παρακάτω ο χρήστης μπορεί να μειώσει το latency, να βελτιώσει το throughput και να περιορίσει την περιοχή δέσμευσης της FPGA από τον κώδικα.[73] [74]Οι βασικότερες μέθοδοι βελτιστοποίησης είναι οι εξής:

- **Pipelining:** Αυτή η τεχνική βελτίωσης διαιρεί τα στάδια επεξεργασίας σε πολλά μικρότερα τα οποία μπορούν να επεξεργαστούν διαφορετικά δεδομένα εισόδου ταυτόχρονα, πετυχαίνοντας έτσι την παραλληλοποίηση των διεργασιών. Μειώνεται έτσι το αριθμός των κύκλων για την επόμενη λήψη δεδομένων (initiation interval) ενώ το thoughput αυξάνεται σημαντικά. Η αλόγιστη χρήση αυτής της τεχνική όμως, οδηγεί σε αυξημένη κατανάλωση πόρων.
- **Βελτιστοποιήσεις Πινάκων:** Στο περιβάλλον του Vivado hls ένας πίνακας αντιπροσωπεύει αποθηκευτικό χώρο και γι' αυτό το λόγο συντίθεται σε μνήμες. Η μνήμη που ορίζεται στο Vivado hls αντιστοιχίζεται σε φυσικούς πόρους της FPGA, όπως BRAM, και έτσι είναι σημαντικό να είναι γνωστό το μέγεθός της και ο τρόπος με τον οποίο αντιστοιχίζεται σε σχέσης με τους πόρους που είναι διαθέσιμοι στην FPGA κατά την διάρκεια της σχεδίασης και συγγραφής του κώδικα. Παρακάτω περιγράφονται οι βελτιστοποιήσεις που μπορούν να εφαρμοστούν:

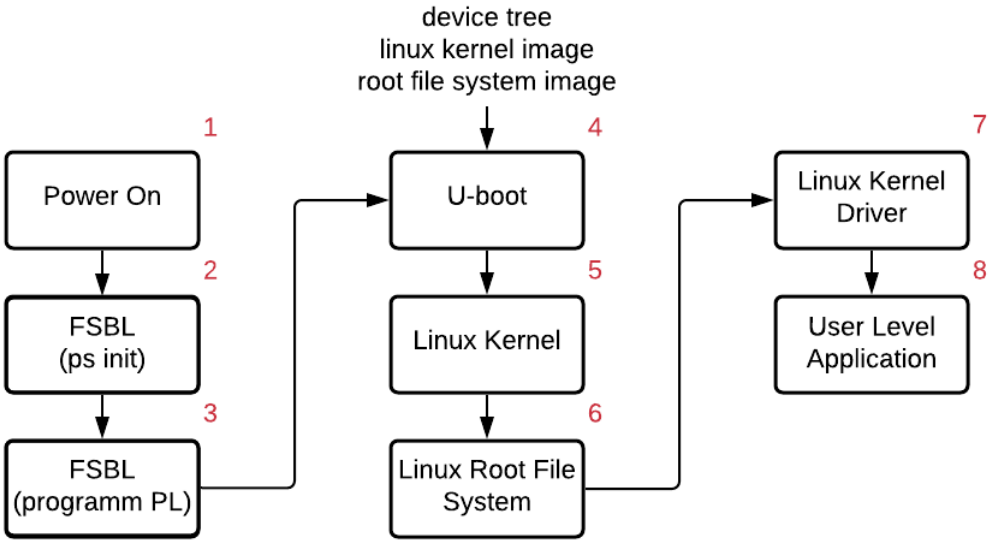
- *Resource:* Ο σχεδιαστής μπορεί να προσδιορίσει με ποιον ακριβώς πόρο θα υλοποιήσει μια μεταβλητή (πίνακα, ως όρισμα συνάρτησης ή αριθμητική λειτουργία) σε επίπεδο HDL (RTL).
- *Array Map:* Με αυτή τη τεχνική πολλοί μικροί πίνακες συνδυάζονται σε έναν μεγαλύτερο ώστε να να μειωθούν οι πόροι Block Ram που απαιτούνται.
- *Array Partition:* Με αυτή τη τεχνική ένας πίνακας διαιρείται σε μικρότερους υποπίνακες ή μεμονωμένα στοιχεία, αυξάνοντας έτσι τον ρυθμό με τον οποίο πραγματοποιούνται μεταφορές δεδομένων από και προς την μνήμη(2 port Block Ram).
- *Stream:* Με την εντολή Stream, ο χρήστης έχει την ικανότητα να αντιστοιχίσει έναν πίνακα σε δομές FIFO αντί RAM. Ιδιαίτερα χρήσιμο σε περιπτώσεις όπου τα αποθηκευμένα δεδομένα παράγονται ή καταναλώνονται διαδοχικά.

■ **Loop Unrolling**: Με αυτή τη τεχνική ο χρήστης έχει την ικανότητα να μετασχηματίσει ένα βρόχο με διαφορετικούς τρόπους. Ξετυλίγοντας ένα βρόχο δημιουργούνται πολλαπλά αντίγραφα του σώματος του βρόχου στο RTL σχέδιο, κάνοντας εφικτή έτσι, την παράλληλη επανάληψη κάποιων ή όλων των βρόχων.

■ **Dataflow:** Αυτή η τεχνική βελτιστοποίησης θα μπορούσε να χαρακτηριστεί ως η τεχνική pipeline αλλά εφαρμοσμένη σε ένα υψηλότερο επίπεδο στη σχεδιαστική ιεραρχία. Πιο συγκεκριμένα, επιτρέπει στις συναρτήσεις και στους βρόχους να επικαλύπτονται κατά τη λειτουργία, αυξάνοντας με αυτόν τον τρόπο τον συγχρονισμό της RTL υλοποίησης και του συνολικού throughput. Αν για παράδειγμα σε μια συνάρτηση πρέπει να γίνει η προσπέλαση ενός πίνακα, τότε η λειτουργία της επόμενης συνάρτησης δεν μπορεί να ξεκινήσει πριν εκτελεστούν όλες οι εντολές ανάγνωσης και εγγραφής του πίνακα. Η ντιρεκτίβα Dataflow επιτρέπει την εκκίνηση της επόμενης συνάρτησης πριν την ολοκλήρωση όλων των εντολών του πίνακα.

4.1.3 Petalinux SDK

Το Petalinux είναι ένα εργαλείο που προσφέρει η Xilinx με σκοπό την ανάπτυξη ενός embedded Linux συστήματος. Αποτελεί ένα εργαλείο το οποίο συγκεντρώνει πολλά χαρακτηριστικά και δυνατότητες ώστε η ανάπτυξη του συστήματος να γίνει γρηγορότερη και περισσότερο φιλική προς τον χρήστη κάνοντας χρήση απλών εντολών μέσω του terminal. Το Petalinux παράγει τα κατάλληλα αρχεία που χρειάζονται ώστε να εκκινήσει ένα embedded Linux σύστημα. Ακολουθεί μια περιγραφή της ακολουθίας εκκίνησης ενός τέτοιου συστήματος:



Εικόνα 44: Boot Sequence

First Stage Boot Loader(FSBL): Όταν το σύστημα εκκινήσει για πρώτη φορά το FSBL αποτελεί το πρώτο βήμα. Μέσω αυτού αρχικοποιείται και διαμορφώνεται, σύμφωνα με τις προδιαγραφές του χρήστη, ο επεξεργαστής του Zynq (PS). Επίσης αυτό το στάδιο είναι υπεύθυνο και για την φόρτωση του δυαδικού αρχείου (bitstream) που έχει παραχθεί από το Vivado IDE, ώστε προγραμματιστεί και η FPGA (PL).

U-BOOT: Αφού ολοκληρωθεί η διαδικασία του FSBL, ο program counter της CPU θα περάσει στο u-boot και θα ξεκινήσει την λειτουργία του. Η δουλεία του u-boot είναι να ρυθμίσει το περιβάλλον του συστήματος για την φόρτωση του linux kernel, του linux filesystem και του device tree που έχουν οριστεί από τον χρήστη.

- Linux Kernel: Ο πυρήνας είναι το κύριο κομμάτι του λογισμικού και αποτελεί τη βασική διεπαφή μεταξύ του hardware και των διαδικασιών που εκτελούνται. Το Petalinux χρησιμοποιεί λογισμικό Linux που διανέμεται από το Yocto project το οποίο είναι ένα toolchain. Μέσω αυτού δίνεται η δυνατότητα στο χρήστη να παραμετροποιήσει τον πυρήνα, προσφέροντας στήριξη για εισαγωγή νέων συσκευών και περιφερειακών,διαχείριση μνήμης κ.α .
- Linux Filesystem: Το Petalinux SDK προσφέρει την δυνατότητα παραγωγής και παραμετροποίησης ενός δικού του filesystem. Ο χρήστης έχει την επιλογή ενσωμάτωσης εφαρμογών, βιβλιοθηκών και εργαλείων στο σύστημα.
- Device Tree: Το device tree, στην πιο απλή περιγραφή του, είναι μια δομή δεδομένων η οποία περιέχει τις προδιαγραφές και τα χαρακτηριστικά του hardware ώστε ο πυρήνας του λειτουργικού συστήματος να μπορέσει την συσκευή και να τα τη διαχειριστεί. Μπορεί να περιέχει τις διευθύνσεις για τα διαφορετικά ip cores που έχουν

χρησιμοποιηθεί στο σύστημα, δείχνει ποιες συσκευές υπάρχουν και ποιες λείπουν από το σύστημα.

```
ps7_ttc_1: ps7-ttc@0xf8002000 {
  clocks = <&clkc 6>;
  compatible = "xlnx,ps7-ttc-1.00.a";
  interrupt-parent = <&ps7_scugic_0>;
  interrupts = <0 37 4>,<0 38 4>,<0 39 4>;
  reg = <0xF8002000 0x1000>;
  status = "disabled";
} ;
```

Εικόνα 45: Device Tree Example

Στον παραπάνω κώδικα η πρώτη γραμμή καθορίζει το όνομα (ps7-ttc) και την διεύθυνση (0xf8002000).Ο όρος “compatible” επιτρέπει στο λειτουργικό σύστημα να αναγνωρίσει τον device driver που αντιστοιχεί σε αυτό το όνομα. Με τον όρο “reg” σηματοδοτείται η φυσική διεύθυνση και το μέγεθος που καταλαμβάνει στη μνήμη. Ο όρος “interrupt-parent” δείχνει απλά την διεύθυνση του interrupt controller που χρησιμοποιεί και αντίστοιχα ο όρος “clocks” αναφέρει το ρολόι από το οποίο εξαρτάται αυτός ο κόμβος. Τέλος με την ιδιότητα “status” παρουσιάζεται αν η συσκευή είναι ενεργοποιημένη ή απενεργοποιημένη.

Linux Kernel: Αφού ολοκληρωθεί η φόρτωση των στοιχείων που αναφέρθηκαν στο βήμα του U-boot, ο πυρήνας του Linux ξεκινάει την εκτέλεση των αρχικών του εργασιών. Με το πέρας αυτών των εργασιών, πραγματοποιείται η προσάρτηση (mount) του filesystem.

Linux Root File System: Αποτελεί το περιβάλλον το οποίο περιέχει τις εφαρμογές,τα αρχεία και τα directories που θα χρειαστούν για την υλοποίηση του συστήματος.

Μετά την ολοκλήρωση της εκκίνησης του συστήματος linux, έρχεται η ώρα για την φόρτωση των linux kernel drivers στον πυρήνα. Αυτό σηματοδοτεί και το τελευταίο βήμα της ακολουθίας εκκίνησης.

Linux Device Model

Το μοντέλο με το οποίο το linux διαχειρίζεται τις συσκευές, έχει χτιστεί γύρω από τις έννοιες των δίαυλων επικοινωνίας (busses), των συσκευών (devices) και των οδηγών (drivers) τους. Όλες οι συσκευές στο σύστημα συνδέονται μεταξύ τους μέσω κάποιου είδους δίαυλου που είναι υπεύθυνος για να συγκεντρώσει μαζί παρόμοιες συσκευές,να συντονίσει την αρχικοποίηση τους, το τερματισμό λειτουργίας και τη διαχείριση ισχύος. Όταν μια συσκευή στο σύστημα ταιριάξει με κάποιον οδηγό (driver), τότε συνδέονται μεταξύ τους. Ο τρόπος με τον οποίο γίνεται το “ταίριασμα” μεταξύ

συσκευής και οδηγού εξαρτάται από το είδος του δίαυλο που τους ενώνει. Δίαυλοι επικοινωνίας, όπως αυτός του τύπου Peripheral Component Interconnect (PCI) ή του USB, έχουν ενσωματωμένη την δυνατότητα του εντοπισμού. Μια συσκευή που βρίσκεται σε δίαυλο PCI μπορεί να πει στο σύστημα τι είδους συσκευή είναι και που βρίσκονται οι πόροι του. Με αυτόν το τρόπο ο πυρήνας μπορεί κατά την εκκίνηση να απαριθμήσει τις διαθέσιμες συσκευές . Στα ενσωματωμένα συστήματα όμως, πολλές συσκευές ενώ παρευρίσκονται στο σύστημα δεν είναι εγγενώς ανιχνεύσιμες από αυτό. Αντί, η απαρίθμησή τους να γίνει σε επίπεδο hardware κατά τον χρόνο εκτέλεσης, χρειάζεται, το λογισμικό να γνωρίζει κατά την στιγμή της μεταγλώττισης ποιες συσκευές είναι συνδεδεμένες σε κάθε τμήμα του δίαυλου. Τέτοιου είδους συσκευές, όπως για παράδειγμα οι συσκευές I2C, συναντώνται συχνά στα ενσωματωμένα συστήματα.

Σύμφωνα με τα παραπάνω προκύπτει ότι η "ταυτότητα" μιας συσκευής εξαρτάται άμεσα από τον τύπο δίαυλου που χρησιμοποιεί. Στην $2_η$ περίπτωση όπου η συσκευή δεν συνδέεται με κάποιον δίαυλο και δεν ανιχνεύεται δυναμικά από το υπόλοιπο σύστημα, δημιουργείται η ανάγκη περιγραφή τους με στατικό τρόπο χρησιμοποιώντας το device tree. Δημιουργείται έτσι ένας δίαυλος πλατφόρμας (platform bus) για να χειρίζεται τέτοιου είδους συσκευές. Αυτός ο δίαυλος μπορεί να μην έχει φυσική υπόσταση αλλά να αποτελεί κομμάτι αφαιρετικότητας σε επίπεδο software. Υποστηρίζει τους οδηγούς πλατφόρμας (platform drivers) οι οποίοι είναι υπεύθυνοι για να χειριστούν τις **platform devices**.

Στην ουσία οι 2 κυριότεροι τύποι συσκευών σε περιβάλλον linux είναι:

Block devices: Αυτές οι συσκευές χρησιμοποιούνται από διεργασίες που τρέχουν σε userspace (δηλαδή σε εικονικές διευθύνσεις και όχι τις πραγματικές/φυσικές) ώστε να αποκτήσουν πρόσβαση σε αποθηκευτικές συσκευές όπως σκληρούς δίσκους.

Character Devices: Αυτές οι συσκευές χρησιμοποιούνται από διεργασίες που τρέχουν σε userspace ώστε να αποκτήσουν πρόσβαση σε άλλους τύπους συσκευών όπως ήχου, γραφικών, σειριακές κ.α. Από την οπτική γωνία της εφαρμογής μια συσκευή χαρακτήρα είναι ένα αρχείο. Ο οδηγός ενός character device υλοποιεί λειτουργίες που επιτρέπουν τις εφαρμογές να διαχειρίζονται τη συσκευή ως αρχείο. Ένα platform device μπορεί να θεωρηθεί character device.

Πολλοί οδηγοί συσκευών δεν υλοποιούνται κατευθείαν ως συσκευές block ή συσκευές χαρακτήρα, αλλά υλοποιούνται στα πλαίσια ενός framework που είναι συγκεκριμένο για τον εκάστοτε τύπο συσκευής (framebuffer, V4L2, serial). Με αυτόν το τρόπο προσδιορίζονται τα κοινά κομμάτια του κώδικα των οδηγών για ίδιους τύπους συσκευών και έτσι μειώνεται η επανάληψη ίδιου κώδικα.

Ένας οδηγός συσκευής μπορεί να ενσωματωθεί στον πυρήνα του linux με δύο διαφορετικούς τρόπους.

- Ο πρώτος είναι να ενσωματωθεί κατευθείαν στον πυρήνα κατά την μεταγλώττιση του. Με αυτόν τον τρόπο ο driver θα παραμείνει στον πυρήνα και κατ' επέκταση στην μνήμη RAM, καθ' όλη την διάρκεια λειτουργίας του συστήματος ακόμα και αν η αντίστοιχη συσκευή αφαιρεθεί ή δεν χρησιμοποιείται. Σε αυτήν τη περίπτωση χώρος στην μνήμη μένει δεσμευμένος χωρίς να προσφέρει κάποια λειτουργία.

- Ο δεύτερος τρόπος αφορά την εισαγωγή, χρήση και αφαίρεση οδηγών δυναμικά. Οι οδηγοί αυτοί ονομάζονται **kernel modules**. Αυτός ο τρόπος δίνει την επιλογή στον χρήση να επεκτείνει την λειτουργικότητα του πυρήνα χωρίς να χρειαστεί να γίνει επανεκκίνηση ή αλλαγή στο κώδικά του όσο αυτός "τρέχει". Ένα kernel module βοηθάει στο να κρατηθεί το μέγεθος του πυρήνα στο ελάχιστο δυνατό παρέχοντας παράλληλα ευελιξία στην επιλογή της ποιας συσκευής και πότε θα χρησιμοποιηθεί.

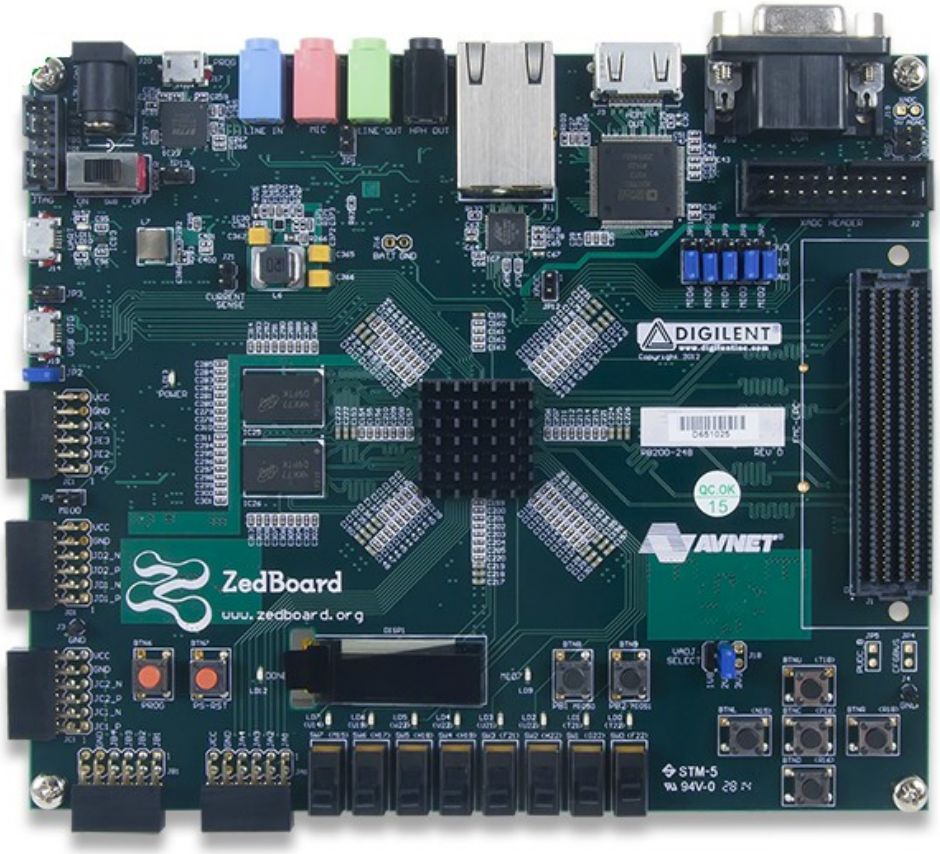
4.2 Περιγραφή Zynq-7000 SoC Zedboard

Για την υλοποίηση της αρχιτεκτονική που περιγράφεται στα παρακάτω κεφάλαια χρησιμοποιήθηκε ένα Zedboard. Πρόκειται για μια αναπτυξιακή πλακέτα χαμηλού κόστους για το Xilinx Zynq-7000 All Programmable SoC XC7Z020-CLG484. Περιέχει όλα τα απαραίτητα στοιχεία για τη δημιουργία σχεδιασμού που βασίζεται σε OS / RTOS (real time operating system). Επιπλέον, αρκετοί σύνδεσμοι επέκτασης εκθέτουν το σύστημα επεξεργασίας (PS) και τα I/Os της προγραμματιζόμενης λογικής (PL) για εύκολη πρόσβαση από το χρήστη. Το σύστημα επεξεργασίας παρέχει απαραίτητα περιφερειακά για το σύστημα όπως θύρες USB 2.0, Ethernet αλλά και υποδοχή για SD κάρτα. Η προγραμματιζόμενη λογική βασίζεται στη σειρά FPGA Kintex-7 παρέχοντας υποστήριξη για πολλαπλούς τύπους οθονών (1080p HDMI, 8-bit VGA, 128 x 32 OLED) και κωδικοποιητή ήχου. Σύμφωνα με τα παραπάνω το Zedboard είναι κατάλληλο για εφαρμογές που στοχεύουν σε :

- Επεξεργασία βίντεο
- Επιτάχυνση λογισμικού
- Ανάπτυξη Linux / Android / RTOS λογισμικού
- Ενσωματωμένη επεξεργασία ARM

Παρακάτω παρουσιάζονται τα κύρια χαρακτηριστικά της FPGA:

Logic Cells (k)	B-RAM (kB)	DSP Slices	Look-Up Tables	Flip-Flops	DDR3 RAM (MB)
85	560	220	53200	106400	512



Εικόνα 46: Zynq-7000 SoC Zedboard

4.3 Δημιουργία Sobel φίλτρου

Για λόγους επίδειξης της λειτουργίας του συστήματος, δημιουργήθηκε σε hardware ένα φίλτρο Sobel το οποίο θα λαμβάνει τα δεδομένα της USB κάμερας από την μνήμη, θα πραγματοποιεί ανίχνευση ακμών και θα εξάγει τα φιλτραρισμένα πλέον δεδομένα, ώστε να αποσταλούν στη μνήμη.

Το εργαλείο της Xilinx High Level Synthesis (HLS) δίνει την δυνατότητα ανάπτυξης εφαρμογών FPGA δουλεύοντας σε υψηλότερα επίπεδα αφαιρετικότητας, γεγονός που διευκολύνει την ανάλυση του συστήματος και καθιστά ευκολότερη τη βελτιστοποίηση της αντιστάθμισης μεταξύ λογικών πόρων και ταχύτητας επεξεργασίας. Το φιλτράρισμα μιας εικόνας αποτελεί μια από τις πιο συχνά υλοποιημένες εφαρμογές σε περιβάλλον HLS όπου ο σχεδιαστής έχει την επιλογή να δημιουργήσει ένα μοντέλο υψηλού επιπέδου σε C, C++ ή να κάνει χρήση πρότυπων βιομηχανικών framework ανοιχτού κώδικα, όπως το OpenCV.[75]

Όπως περιγράφηκε στο κεφάλαιο 3.3 ο αλγόριθμος Sobel λειτουργεί ανιχνεύοντας τις ακμές σε μια εικόνα και τονίζοντας τες έτσι ώστε να είναι εύκολα αναγνωρίσιμες. Αν αυτή η διαδικασία υλοποιούταν σε FPGA χρησιμοποιώντας μια προσέγγιση VHDL/Verilog RTL ο χρόνος ανάπτυξης εν

δυνάμει, να αποτελούσε πρόβλημα. Θα χρειαζόταν η δημιουργία buffers γραμμής για τη συνέλιξη και στη συνέχεια ο υπολογισμός του απόλυτου μεγέθους. Επίσης επιτακτική θα ήταν και η ανάπτυξη ενός test bench, για την διασφάλιση της ομαλής λειτουργίας του κώδικα όπως προβλεπόταν από τον σχεδιαστή πριν την εφαρμογή στο υλικό.

Με την χρήση του Vivado HLS, δίνεται η δυνατότητα στον σχεδιαστή να παρακάμψει αρκετές από τις δύσκολες και χρονοβόρες διαδικασίες και να αφήσει το Vivado HLS να εφαρμόσει τις απαιτούμενες υλοποιήσεις χαμηλού επίπεδου Verilog/VHDL RTL. Ο σχεδιαστής σε αυτή τη περίπτωση εκμεταλλεύεται τις βιβλιοθήκες υψηλού επίπεδου Vivado HLS_OpenCV και HLS_Video.

Η βιβλιοθήκη HLS_OpenCV επιτρέπει την εργασία με το πολύ δημοφιλές framework OpenCV. Η χρήση αυτής της βιβλιοθήκης κρύβει έναν βασικό περιορισμό όμως, το γεγονός ότι δεν μπορεί εφαρμοστεί απευθείας η λειτουργία Synthesis για τις συναρτήσεις της αλλά πρέπει να αντικατασταθούν με τις αντίστοιχες συναρτήσεις από την synthesizable βιβλιοθήκη. Αυτό ευθύνεται κυρίως στο γεγονός ότι οι λειτουργίες OpenCV περιλαμβάνουν συνήθως δυναμική κατανομή μνήμης, όπως συμβαίνει κατά τη διαδικασία κατασκευής του αντικειμένου, αυθαίρετου μεγέθους, cv :: Mat, το οποίο δεν είναι synthesizable. Αυτή η βιβλιοθήκη θα χρησιμοποιηθεί για την δημιουργία του test bench.

Από την άλλη η βιβλιοθήκη HLS_Video παρέχει μια σειρά από synthesizable συναρτήσεις διεπαφής και επεξεργασίας εικόνας που μπορούν να επιταχυνθούν σε προγραμματιζόμενη λογική.

4.3.1 Διεπαφή

Τα κομάτια επεξεργασίας βίντεο που παρέχει η Xilinx χρησιμοποιούν ένα κοινό πρωτόκολλο ροής AXI4 (AXI4 Stream) για την επικοινωνία δεδομένων εικονοστοιχείων. Αυτό το πρωτόκολλο περιγράφει κάθε γραμμή εικονοστοιχείων βίντεο ως πακέτο AXI4, με το τελευταίο εικονοστοιχείο κάθε γραμμής να επισημαίνεται με το σήμα TLAST. Επιπλέον, η αρχή του καρέ βίντεο υποδεικνύεται επισημαίνοντας το πρώτο εικονοστοιχείο της πρώτης γραμμής με το bit του TUSER[0] σήματος.

Αν και το πρωτόκολλο AXI4 Stream δεν ενέχει περιορισμούς όσον αφορά το μέγεθος των γραμμών σε μια εικόνα, οι πιο περίπλοκοι υπολογισμοί επεξεργασίας βίντεο απλοποιούνται πολύ όταν όλες οι γραμμές βίντεο έχουν το ίδιο μήκος. Αυτός ο περιορισμός ικανοποιείται σχεδόν πάντα από οποιαδήποτε μορφή ψηφιακού βίντεο, εκτός από περιπτώσεις που εμφανίζεται παρατυπία στην αρχή μιας ακολουθίας καρέ βίντεο. Η αντιμετώπιση τέτοιων περιπτώσεων είναι συνήθως πρόβλημα για τη διεπαφή εισόδου ενός μπλοκ επεξεργασίας, το οποίο πρέπει να χειριστεί σωστά αυτή τη κατάσταση πριν από τη μετάβαση στην επεξεργασία συνεχών frames. Η διεπαφή εισόδου που λαμβάνει ένα πρωτόκολλο AXI4 Stream μπορεί να διασφαλίσει ότι κάθε πλαίσιο βίντεο αποτελείται από έναν αριθμό (γραμμές x στήλες) εικονοστοιχείων.

Η βιβλιοθήκη HLS_Video παρέχει δύο πολύ σημαντικές συναρτήσεις που αποδεσμεύουν τον προγραμματιστή από την ενασχόληση με τα προβλήματα διεπαφής που περιγράφηκαν παραπάνω. Οι συναρτήσεις που παρουσιάζονται παρακάτω είναι αναγκαίες για την διεπαφή του IP core φίλτρου που θα δημιουργηθεί, με τις υπόλοιπες μονάδες του hardware σύστηματος (όπως θα περιγραφεί στο κεφάλαιο 4.4), καθώς το φίλτρο θα πρέπει να δέχεται είσοδο AXI Stream και να παράγει ίδια AXI Stream format εξόδου :

hls::AXIvideo2Mat : Αυτή η συνάρτηση μετατρέπει τα δεδομένα εισόδου από τη AXI4 video stream μορφή (format) σε αναπαράσταση μορφής hls::MAT.

hls::Mat2AXIvideo : Αυτή η συνάρτηση μετατρέπει τα δεδομένα που έχουν αποθηκευτεί ως hls::MAT format σε AXI4 video stream format για έξοδο από το φίλτρο IP core.

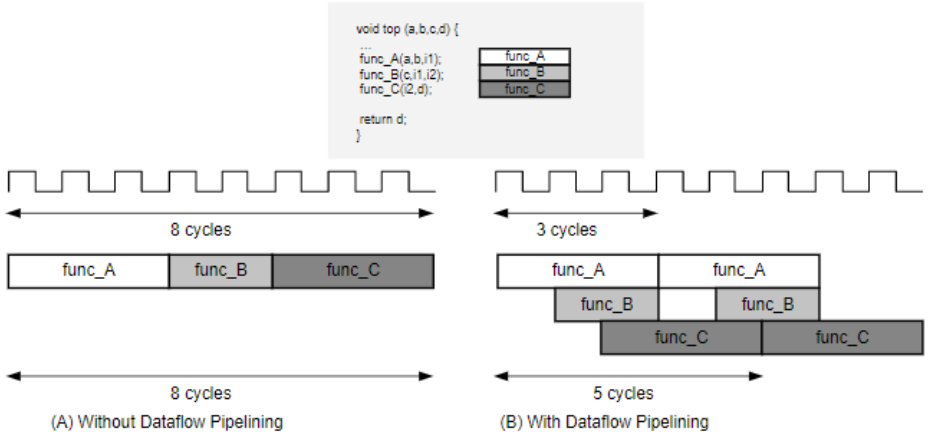
4.3.2 Στοιχεία Επεξεργασίας

Η βιβλιοθήκη HLS_Video παρέχει επίσης συναρτήσεις που διευκολύνουν πολύ την εφαρμογή του φίλτρου και συντελούν τα διάφορα βήματα που υφίσταται μια εικόνα κατά την διαδικασία ανίχνευσης ακμών Sobel:

- ***hls::CvtColor*** **:** Αυτή η συνάρτηση χρησιμοποιείται για την μετατροπή μιας εικόνας από έναν χρωματικό χώρο σε έναν άλλο. Για ευκολότερη υλοποίηση και καλύτερα αποτελέσματα η BGR εικόνα στην είσοδο μετατρέπεται σε αποχρώσεις του γκρι.
- ***hls::GaussianBlur*** **:** Αυτή η συνάρτηση εφαρμόζει Gaussian θόλωμα (φίλτρο) με σκοπό τη μείωση των υψηλών συχνοτήτων των στοιχείων στην εικόνα και κατ' επέκταση την μείσωση θορύβου. Δίνεται η δυνατότητα επιλογής για τις διαστάσεις του φίλτρου που θα εφαρμοστεί.
- ***hls::Sobel :*** Αυτή η συνάρτηση υλοποιεί την συνέλιξη Sobel είτε στην οριζόντια είτε στην κάθετη κατεύθυνση, ανάλογα με τις ρυθμίσεις που θέτει ο χρήστης. Δύο τέτοιες συναρτήσεις θα εφαρμοστούν παράλληλα.
- ***hls::AddWeighted :*** Αυτή η συνάρτηση εκτελεί τον υπολογισμό του απόλυτου μεγέθους των αποτελεσμάτων από τις δύο συναρτήσεις Sobel (οριζόντια και κάθετη).

4.3.3 Βελτιστοποιήσεις

Η χρήση του εργαλείου HLS προσφέρει τη δυνατότητα επιλογής μεταξύ δέσμευσης λογικών πόρων και συνολικής απόδοσης, μέσα από διάφορους κανόνες που επιβάλει κατά την εκτέλεση της λειτουργίας Synthesis. Για παράδειγμα η προκαθορισμένη λειτουργία του HLS για ένα βρόγχο είναι να το διατηρήσει rolled χωρίς να λαμβάνει υπόψιν την αρνητική επίδραση στην απόδοση του συστήματος. Ο προγραμματιστής χρησιμοποιώντας τα #pragmas έχει την δυνατότητα να προσαρμόσει αυτούς τους κανόνες σύμφωνα με τις ανάγκες του συστήματος. Στην προκειμένη υλοποίηση χρησιμοποιήθηκε το *#pragma DATAFLOW* με το οποίο το Vivado HLS αναλύει τη ροή δεδομένων μεταξύ διαδοχικών συναρτήσεων ή βρόγχων και δημιουργεί κανάλια που επιτρέπουν στις συναρτήσεις να ξεκινήσουν τη λειτουργία τους πριν ολοκληρωθούν προηγούμενες λειτουργίες. Αυτό επιτρέπει στις συναρτήσεις να λειτουργούν παράλληλα, γεγονός που μειώνει το latency και βελτιώνει την απόδοση του RTL.



Εικόνα 47: Dataflow Implementation Example. Πηγή Εικόνας: [72]

Το συγκεκριμένο #pragma επιλέχθηκε λόγω της απόφασης για σχεδίαση της ανίχνευσης ακμών Sobel μέσα από διαδοχικές συναρτήσεις του framework OpenCV και αποσκοπεί στην επίτευξη όσο το δυνατόν υψηλότερου frame rate. Το HLS θα χρησιμοποιήσει το DATAFLOW κατά την λειτουργία της Synthesis ώστε οι δύο εφαρμογές της λειτουργίας Sobel (κάθετη και οριζόντια κατεύθυνση) να εκτελεστούν παράλληλα. Σε αυτό το σημείο για να αποφευχθεί η σειριακή υλοποίηση του Sobel θα πρέπει η έξοδος του Gaussian θολώματος να διασπαστεί σε δύο πανομοιότυπα μονοπάτια όπου το ένα προορίζεται για το Sobel που ανιχνεύει τις αλλαγές στην οριζόντια και το άλλο στην κάθετη κατεύθυνση. Θα γίνει χρήση μιας συνάρτησης από την βιβλιοθήκη HLS_Video:

hls::Duplicate : Αυτή η συνάρτηση αντιγράφει μια εικόνα εισόδου σε δύο ξεχωριστές εξόδους ώστε να μπορούν να επεξεργαστούν παράλληλα.

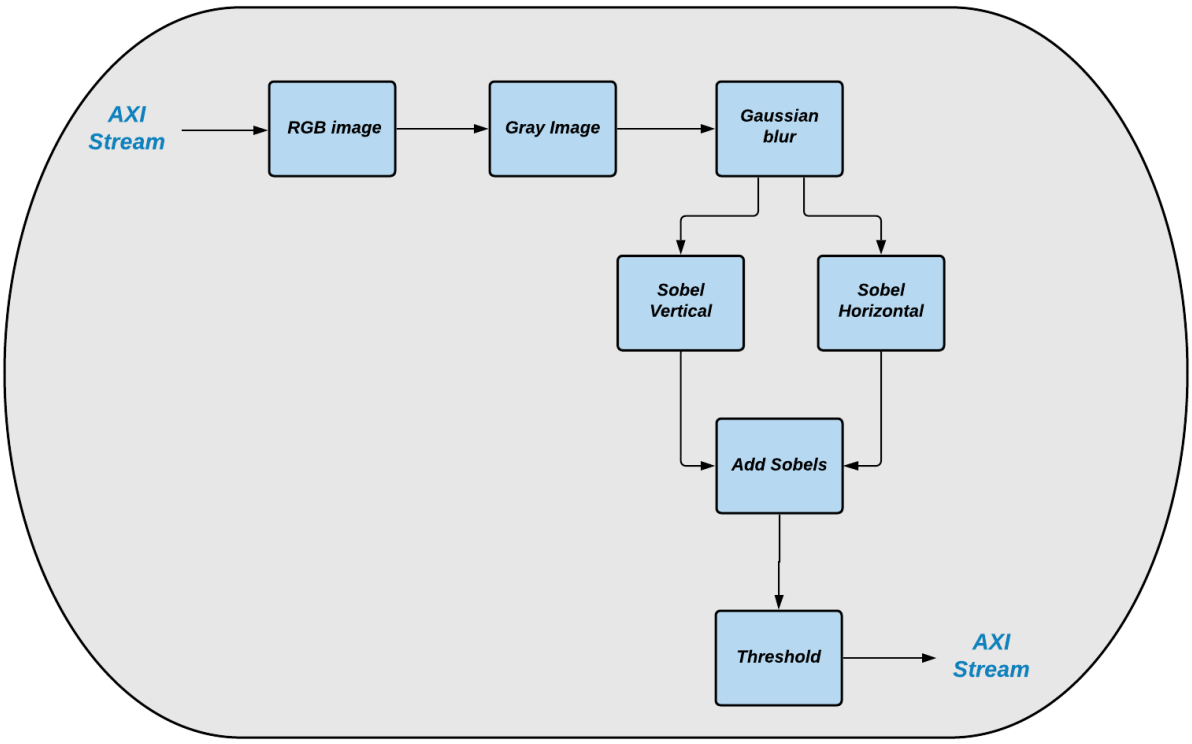
Τέλος για καλύτερη απεικόνιση της grayscale εικόνας υπάρχει και η δυνατότητα εφαρμογής μιας τιμής κατωφλίου που καθορίζεται εμπειρικά λαμβάνοντας υπόψιν το περιεχόμενο της εικόνας εισόδου ύστερα από δοκιμές. Η εφαρμογή τιμής κατωφλίου αποτελεί έναν απλό αλλά αποτελεσματικό τρόπο για τον διαχωρισμό των αντικειμένων από το φόντο (background). Για αυτή τη λειτουργία χρησιμοποιείται η:

hls::Threshold : Αυτή η συνάρτηση δέχεται μια εικόνα ενός καναλιού, αναλύει κάθε εικονοστοιχείο της και άμα αυτό ξεπερνάει την τιμή κατωφλίου που έχει οριστεί, λαμβάνει την μέγιστη τιμή 255 (άσπρο), διαφορετικά λαμβάνει την τιμή 0 (μαύρο). Η εικόνα που προκύπτει είναι πλέον μια binary εικόνα ενός καναλιού.

4.3.4 Ροή Αλγορίθμου

Ένας δίαυλος δεδομένων, που αντιπροσωπεύει το AXI Stream format, πλάτους 24-bit και σήματα TUSER, TID, TDEST και TLAST πλάτους 1 bit, δημιουργείται με σκοπό την εισαγωγή δεδομένων στο IP core. Το format μετατρέπεται σε HLS::MAT RGB εικόνα τριών καναλιών από 8-bit το κάθε κανάλι. Στη συνέχεια η RGB εικόνα μεταφέρεται στη κλίμακα του γκρι και της εφαρμόζεται Γκαουσιανό θόλωμα. Η μειωμένη από θόρυβο πλέον εικόνα, αντιγράφεται μια φορά, με το ένα αντί-

γραφο να οδηγείται σαν είσοδος στο Sobel οριζόντιας κατεύθυνσης και το άλλο αντίγραφο στο Sobel κάθετης κατεύθυνσης. Οι έξοδοι των δύο φίλτρων προστίθενται, εφαρμόζεται τιμή κατωφλίου και τελικά η 8-bit, ενός καναλιού γκρι εικόνα μετατρέπεται πάλι σε AXI Stream μορφή ώστε να είναι δυνατή η επικοινωνία με το υπόλοιπο σύστημα hardware.



Εικόνα 48: Εφαρμογή Φίλτρου

4.3.5 Προσομοίωση Κώδικα

Για την προσομοίωση του κώδικα δημιουργήθηκαν δύο test benches, τα οποία βασίζονται σε συναρτήσεις της βιβλιοθήκη HLS_OpenCV. Το πρώτο αρχείο test bench δημιουργήθηκε για την προσομοίωση εύρεσης ακμών για αρχείο βίντεο, ωστόσο η προσομοίωση μέσα από το εργαλείο Vivado HLS απαιτεί αρκετή ώρα για να ολοκληρωθεί και γι' αυτό το λόγο περισσότερη προσοχή για επίτευξη των επιθυμητών αποτελεσμάτων δόθηκε στο δεύτερο test bench το οποίο αφορά εικόνες. Ένα αρχείο εικόνας 24-bit διαβάζεται και μετατρέπεται σε AXI Stream μορφή ώστε να δοθεί ως είσοδος στο φίλτρο Sobel. Αφού εκτελεστεί το φίλτρο Sobel εξάγεται μια grayscale εικόνα 8-bit η οποία αποθηκεύεται σε επιλεγμένο φάκελο. Παρακάτω παρουσιάζεται η εφαρμογή του Sobel στην εικόνα εισόδου (Lena) για διαφορετικές ρυθμίσεις, όπως μεγαλύτερο Γκαουσιανό θόλωμα (kernel 3x3 και 5x5) για μείωση του θορύβου ή χρήση διαφορετικών τιμών threshold:

Εικόνα 50: Original Image Lena

Εικόνα 49: Lena, blur kernel 3x3, no threshold

Εικόνα 51: Lena, blur kernel 5x5, no threshold

Εικόνα 53:Lena, blur kernel 5x5, threshold=25

Εικόνα 54: Lena, blur kernel 5x5, threshold=40

Εικόνα 52: Lena, blur kernel 5x5, threshold=70

Για σύγκριση των αποτελεσμάτων εφαρμόστηκε ανίχνευση ακμών Sobel σε μια grayscale εικόνα μέσω MATLAB κάνοντας χρήση της έτοιμης συνάρτησης “edge()”. Η συνάρτηση αυτή ανιχνεύει τις ακμές και στις δύο κατευθύνσεις ενώ παράλληλα εφαρμόζει threshold και αραίωση ακμών (edge-thinning).

Εικόνα 55: Lena Sobel from MATLAB

Το αποτέλεσμα του Εικόνα 52 φαίνεται να είναι αρκετά κοντά με αυτό της MATLAB αλλά όχι το ίδιο λεπτομερές. Το παρών κομμάτι της διπλωματικής μελετάει απλά την επίδειξη δημιουργίας και ενσωμάτωσης κάποιου φίλτρου στο σύστημα, για μελλοντική χρήση και εκμετάλλευση, και γιαυτό το λόγο δεν διερευνήθηκε περαιτέρω βελτιστοποίηση.

Στη συνέχεια πραγματοποιείται η λειτουργία Synethesis ώστε να μετατραπεί ο πηγαίος κώδικας C σε γλώσσα περιγραφής υλικού (VHDL, Verilog) και να μπορεί να εκτελεστεί το C/RTL Co-Simulation όπου θα επιβεβαιωθεί ότι το RTL είναι λειτουργικά πανομοιότυπο με τον πηγαίο κώδικα C. Αφού επιβεβαιωθεί το τελευταίο, γίνεται η εξαγωγή της λογικής ως IP block για να ενσωματωθεί στο υπόλοιπο σύστημα.

4.3.6 Αξιοποίηση Πόρων

Όταν ολοκληρωθεί η σύνθεση, παράγεται μια αναφορά για τις συναρτήσεις του top-level. Η αναφορά παρέχει λεπτομέρειες τόσο για την απόδοση(performance) όσο και για τον τομέα του σχεδιασμού RTL(utilization). Στην Εικόνα 56 παρουσιάζονται λεπτομέρειες για την απόδοση που αφορούν κυρίως τον χρονισμό και το latency ενώ στην Εικόνα 57 παρουσιάζονται λεπτομέρειες για την αξιοποίηση του υλικού, όπως οι πόροι (LUTs, Flip-Flops, DSP48s) που θα χρησιμοποιούν για την υλοποίηση του συστήματος.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.40	2.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
271	492055	267	492051	dataflow

Εικόνα 56: Εκτιμήσεις Απόδοσης

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	16
FIFO	0	-	55	220
Instance	11	26	4056	9482
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	-	-
Total	11	26	4111	9718
Available	280	220	106400	53200
Utilization (%)	3	11	3	18

Εικόνα 57: Εκτιμήσεις Αξιοποίησης Πόρων Υλικού

Τέλος μέσω της προοπτικής Analysis παρέχεται μια γενική εικόνα της ροής των συναρτήσεων που χρησιμοποιούνται. Παρατηρώντας στους κύκλους 9 και 10 διαπιστώνεται ότι οι δύο εφαρμογές του Sobel για αλλαγές στην οριζόντια και κάθετη κατεύθυνση πραγματοποιούνται παράλληλα.

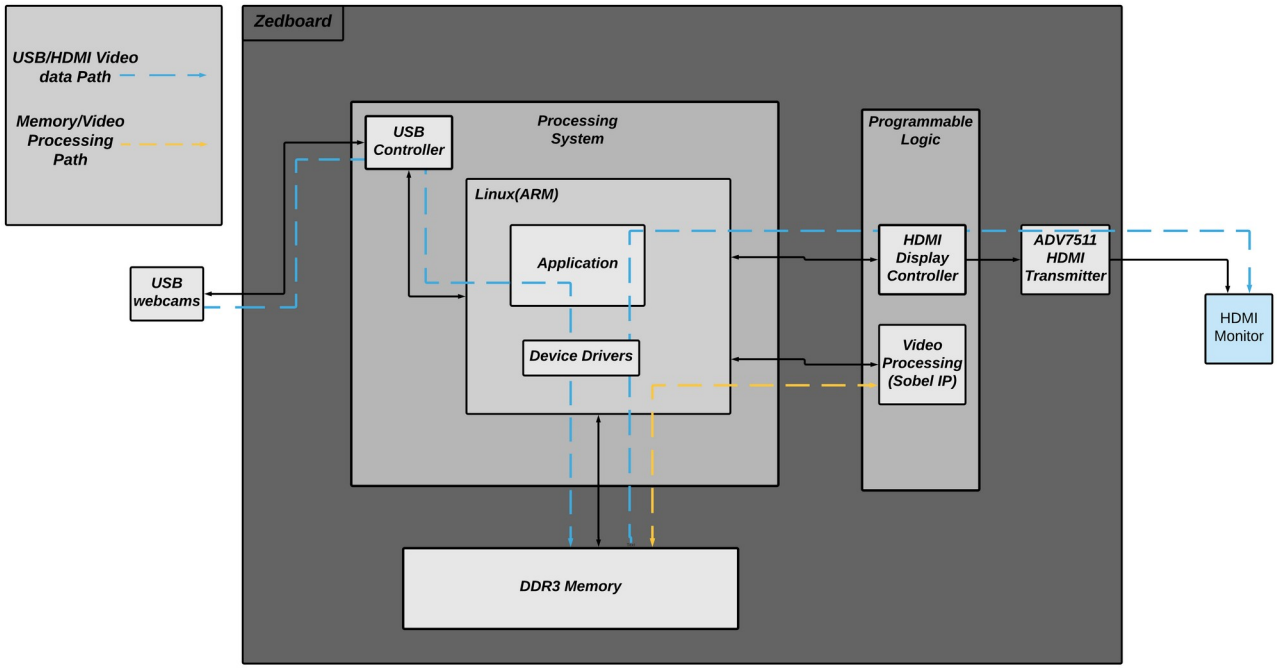
	Operation\Control Step	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
1	cols_read(read)																
2	rows_read(read)																
3	AXIvideo2Mat(function)																
4	CvtColor(function)																
5	GaussianBlur(function)																
6	Duplicate(function)																
7	Sobel_1(function)																
8	Sobel_2(function)																
9	AddWeighted(function)																
10	Threshold(function)																
11	Mat2AXIvideo(function)																

Εικόνα 58: Dataflow Analysis

4.4 Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του συστήματος που θα υλοποιηθεί φαίνεται στην Εικόνα 59. Μία ή δύο κάμερες διασυνδέονται στο σύστημα χρησιμοποιώντας τον USB ελεγκτή που υπάρχει στο σύστημα επεξεργασίας (PS) του SoC. Στην συνέχεια με την προσθήκη των κατάλληλων Linux drivers, η USB συσκευή αναγνωρίζεται ως συσκευή βίντεο και τα δεδομένα της αποθηκεύονται στην εξωτερική, του PS, μνήμη DDR3. Έπειτα δίνεται η επιλογή μεταφοράς αυτών των δεδομένων στην προγραμματιζόμενη λογική, για εφαρμογή του φίλτρου Sobel μέσα από το block επεξεργασίας βίντεο. Τα επεξεργασμένα πλέον δεδομένα επιστρέφουν στην μνήμη DDR3. Το HDMI display controller block στην PL αποκτά πρόσβαση στη περιοχή της μνήμης που αποθηκεύονται τα δεδομένα βίντεο, είτε

αυτά έρχονται κατευθείαν από την USB webcam είτε έχουν υποστεί πρώτα επεξεργασία, και τελικά τροφοδοτεί το εξωτερικό chip ADV7511 HDMI transmitter με δεδομένα βίντεο και τα κατάλληλα σήματα διαμόρφωσης ώστε να προβληθεί το περιεχόμενο που έχει ληφθεί από την κάμερα σε HDMI οθόνη.



Εικόνα 59: Overall Block Design

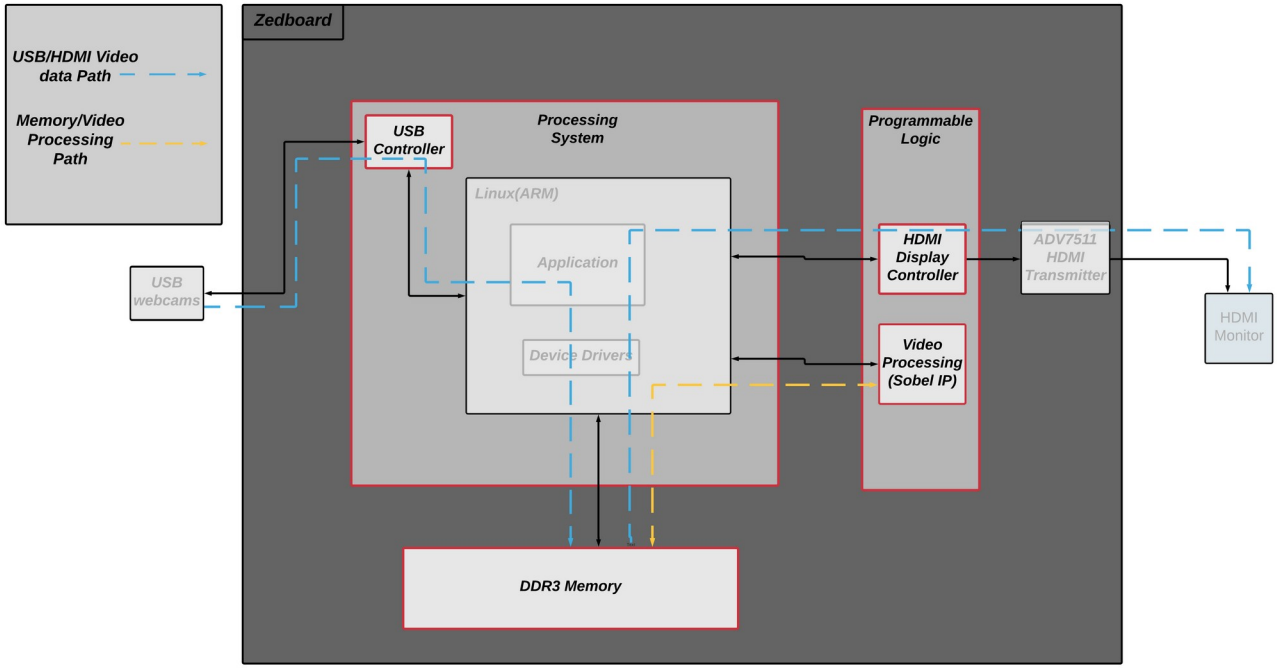
Η υλοποίηση της παραπάνω αρχιτεκτονικής χωρίζεται σε 2 ενότητες. Η πρώτη ενότητα αφορά την δημιουργία του συστήματος hardware. Μέσα από το Vivado, το σύστημα επεξεργασίας (PS) θα υποστεί κατάλληλη διαμόρφωση ώστε να συμπεριλάβει τα απαιτούμε περιφερειακά (USB, microSD card) στο σχέδιο και να καθιερώσει τους κατάλληλους μηχανισμούς επικοινωνία με την PL. Επιπλέον μέσα από το Vivado θα υλοποιηθεί η κατάλληλη λογική στη PL για επεξεργασία και έξοδο βίντεο HDMI καθώς και η επικοινωνία με το σύστημα επεξεργασίας και τη μνήμη DDR3. Με την ολοκλήρωση των διαδικασιών Synthesis και Implementation, το Vivado θα εξάγει ένα αρχείο bitstream το οποίο περιέχει τα απαραίτητα δεδομένα για τον προγραμματισμό της πλακέτας Zedboard.

Η δεύτερη ενότητα αφορά τη δημιουργία του συστήματος software. Θα αναπτυχθεί μια εικόνα Linux που θα εμπεριέχει τα κατάλληλα προγράμματα οδήγησης για τη συσκευή USB webcam αλλά και των block λογικής για έξοδο HDMI στη PL. Με άλλα λόγια, κάθε μονάδα του συστήματος hardware θα μεταφερθεί σε περιβάλλον Linux ώστε να είναι προσβάσιμη από το χρήστη. Για την εφαρμογή του λειτουργικού συστήματος Linux στο Zedboard, το Petalinux θα παράξει ένα αρχείο που περιέχει την εικόνα του διαμορφωμένου πυρήνα (**image.ub**), ένα αρχείο που είναι υπεύθυνο για την αντιστοίχιση του hardware με τους drivers (**devicetree.dtb**) καθώς και ένα αρχείο που είναι υπεύθυνο για τον προγραμματισμό της FPGA, μέσω του bitstream που δημιουργήθηκε από το Vivado, και την εκκίνηση του συστήματος Linux (**BOOT.BIN**). Τέλος θα συμπεριληφθεί ένα σύστημα αρχείων (**File System**) για τη διαχείριση και οργάνωση των δεδομένων αποθήκευσης στον αποθηκευτικό χώρο. Η φόρτωση του λογισμικού Linux θα γίνει μέσα από μια κάρτα microSD η

οποία είναι χωρισμένη σε δύο διαμερίσματα (partitions). Το πρώτο partition περιέχει τα αρχεία BOOT.BIN, image.ub, devicetree.dtb και το δεύτερο το Linux FileSystem.

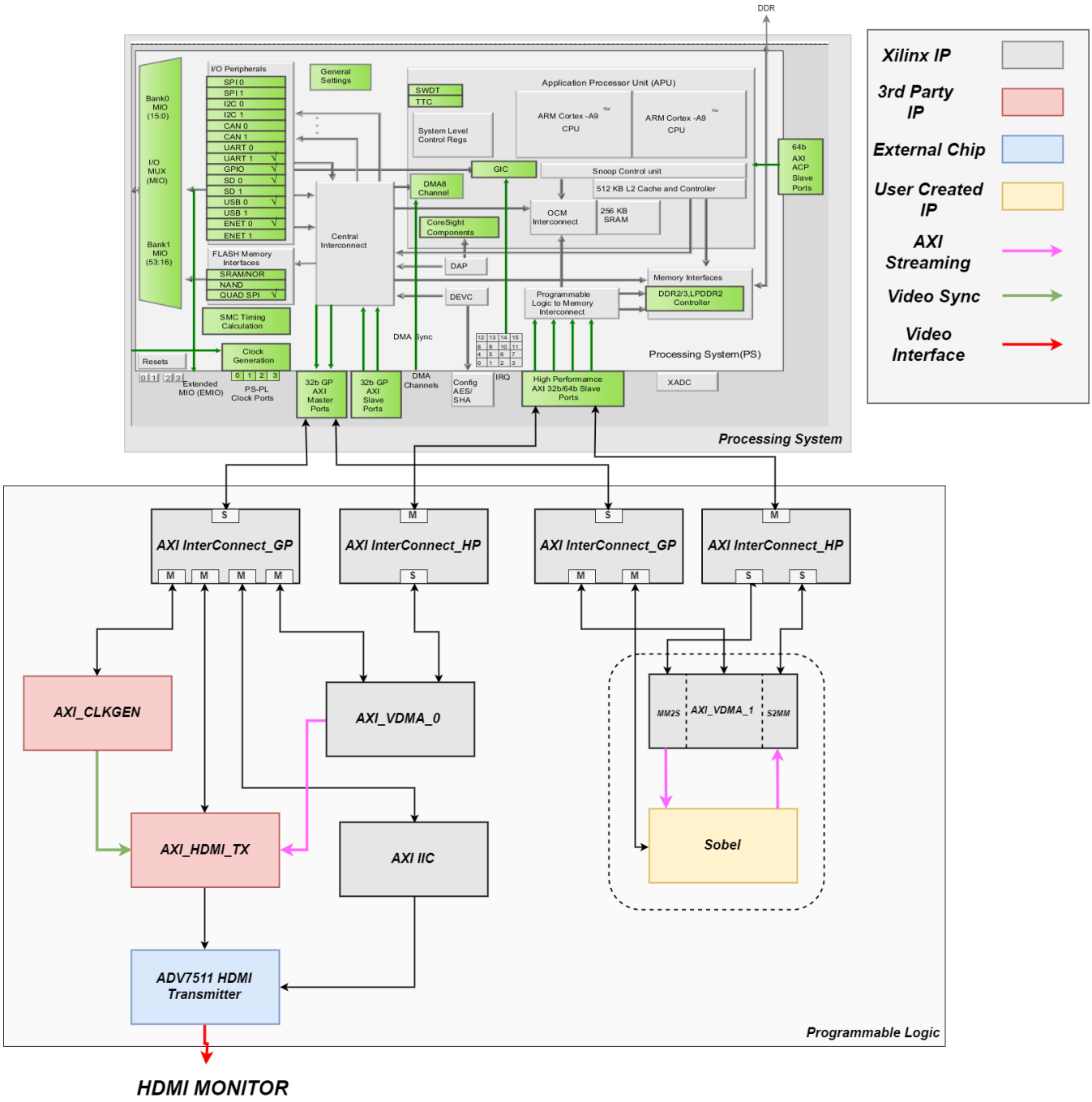
4.4.1 Αρχιτεκτονική Συστήματος Hardware

Η αρχιτεκτονική του συστήματος hardware αφορά κυρίως τη δημιουργία των ελεγκτών USB και HDMI, καθώς και την προσθήκη του Sobel IP block.



Εικόνα 60: Hardware Block Design

Όπως φαίνεται στην Εικόνα 60 η σχεδίαση χωρίζεται κυρίως σε δύο μέρη. Το πρώτο μέρος αφορά το σύστημα επεξεργασίας και την επικοινωνία του με την μνήμη DDR3, ενώ το δεύτερο την προγραμματιζόμενη λογική. Στο παρακάτω σχήμα παρουσιάζεται μια πιο λεπτομερής απεικόνιση των επιμέρους μονάδων του συστήματος, που θα περιγραφούν στις επόμενες ενότητες:



Εικόνα 61: Hardware Block PS-PL Diagram

4.4.1.1 Διαμόρφωση Συστήματος Επεξεργασίας (PS)

Τα κύρια μέρη του συστήματος επεξεργασίας (PS) διακρίνονται σε::

1. **Application Processor Unit (APU)**
2. **Interconnect**
3. **Input/Output peripherals (IOP)**

4. **Memory Interfaces**

Ακολουθεί αναλυτική περιγραφή των παραπάνω:

1. **Application Processor Unit (APU)**

Το APU περιλαμβάνει dual-core ARM Cortex-A9 επεξεργαστή, snoop control unit (SCU), ελεγκτή προσωρινής μνήμης L2, μνήμη on-chip (OCM), DMA 8 καναλιών, watchdog χρονοδιακόπτη συστήματος (SWDT) και ελεγκτή τριπλού χρονοδιακόπτη (TTC).

Cortex-A9 Core: Οι επεξεργαστές ARM® Cortex -A9 με NEON συν-επεξεργαστές συνδεδεμένοι σε διαμόρφωση MP που μοιράζεται προσωρινή μνήμη 512 KB L2. Κάθε επεξεργαστής είναι ένας πυρήνας υψηλής απόδοσης και χαμηλής ισχύος που εφαρμόζει δύο ξεχωριστές προσωρινές μνήμες 32 KB L1 για οδηγίες και δεδομένα. Ο επεξεργαστής Cortex-A9 εφαρμόζει την αρχιτεκτονική ARM v7-A με πλήρη υποστήριξη εικονικής μνήμης και μπορεί να εκτελέσει οδηγίες ARM 32-bit. Ο συν-επεξεργαστής NEON για πολυμέσα και σήματα επεξεργασίας, εισάγει μια αρχιτεκτονική η οποία προσθέτει οδηγίες που στοχεύουν σε επεξεργασία ήχου, βίντεο, εικόνας και τρισδιάστατα γραφικά. Σε αυτό το σχέδιο και οι δύο πυρήνες ARM λειτουργούν στα 667 MHz.

General Intertupt Controller: Αυτός ο ελεγκτής συλλέγει τα interrupts από διάφορες πηγές και τα διανέμει σε κάθε έναν από τους πυρήνες ARM. Ο διανομέας διακοπής (interrupt) διατηρεί τη λίστα των εκκρεμών διακοπών για κάθε επεξεργαστή πυρήνα ARM Cortex-A9 και στη συνέχεια επιλέγει τη διακοπή με την υψηλότερη προτεραιότητα πριν την εκδώσει στη διεπαφή επεξεργαστή Cortex-A9.

2. **Interconnect**

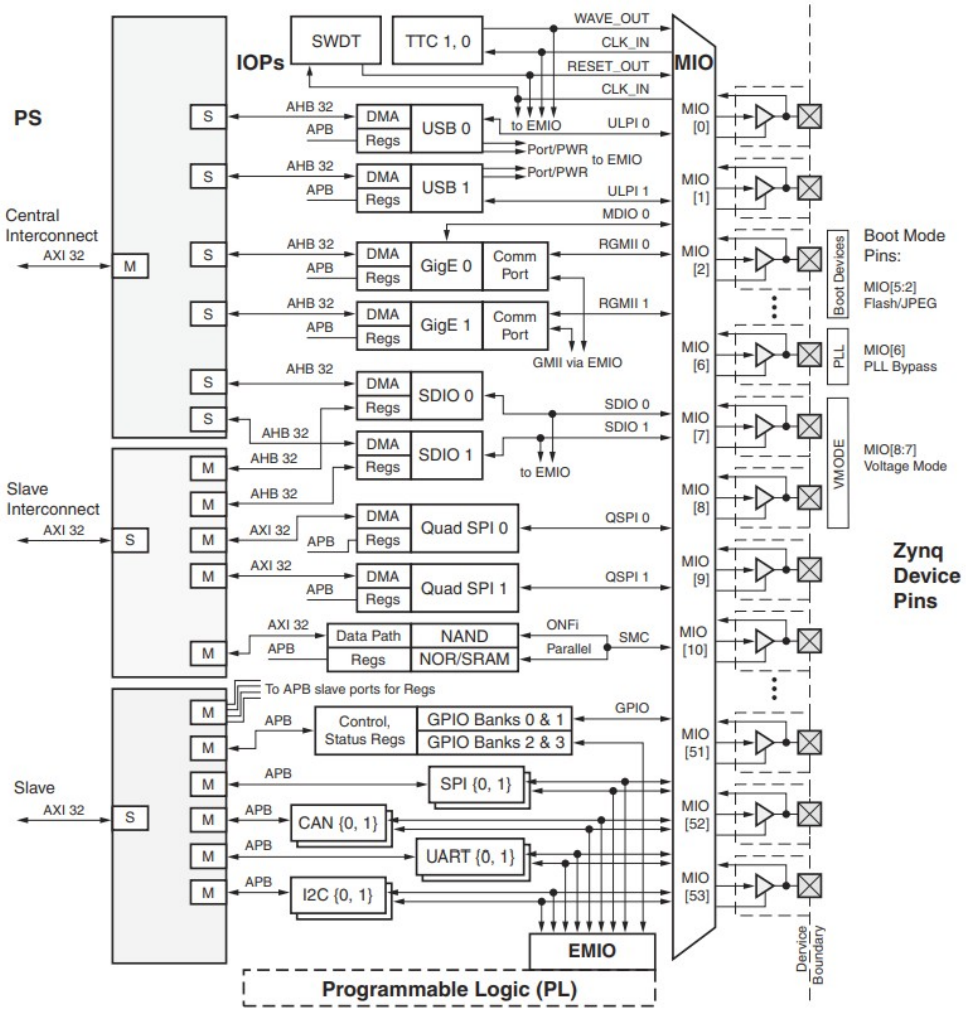
Η μονάδα διασύνδεσης (interconnect) συνδέει όλες τις master και slave συσκευές μεταξύ PS και PL. Υπάρχει δυνατότητα για συνολικά τέσσερις Advanced eXtensible Interface (AXI) slave θύρες που προορίζονται για διασύνδεση με AXI masters που υπάρχουν στη PL και τέσσερις από αυτές τις θύρες περιέχουν deep FIFOs (S_AXI_HP ports) για τη βελτίωση της απόδοσης δεδομένων. Επιπρόσθετα δύο AXI master θύρες (M_AXI_GP) παρέχουν πρόσβαση σε AXI slaves IPs στη PL. Σε αυτόν τον σχεδιασμό, οι masters στο PL συνδέονται μέσω δύο AXI slave θυρών με deep FIFO. Δύο AXI master θύρες χρησιμοποιούνται για πρόσβαση σε registers στα AXI slave IPs στη PL μεριά. Οι διεπαφές AXI στην πλευρά του PS είναι συμβατές με τη προδιαγραφή διασύνδεσης AXI3, τα soft IP που παρέχονται από τη Xilinx στη PL είναι συμβατές με τη προδιαγραφή διασύνδεσης AXI4 και τα soft IPs AXI interconnect λειτουργούν σαν γέφυρα πρωτοκόλλου μεταξύ των δύο ανάλογα με τις ανάγκες.

S_AXI_HP: Οι διεπαφές AXI slave υψηλής απόδοσης (S_AXI_HP) συνδέουν τα μπλοκ PL σε AXI FIFO interface (AFI) στο PS. Το PL διαθέτει τρεις AXI master από τους οποίους ένας συνδέεται στη θύρα S_AXI_HP0 και δύο συνδέονται στη θύρα S_AXI_HP1. Η θύρα HP επιτρέπει μια υψηλή ταχύτητα δεδομένων μεταξύ των AXI masters στην προγραμματιζόμενη λογική και της μνήμης DDR3 του συστήματος επεξεργασίας. Ο κύριος στόχος των μονάδων διασύνδεσης AXI FIFO (AFI) είναι η εξομάλυνση του latency, επιτρέποντας τη δυνατότητα συνεχούς ροής δεδομένων από DDR σε PL masters και από PL masters σε DDR.

M_AXI_GP: Αυτή η θύρα AXI master διασυνδέεται με AXI slave IPs στη PL μέσω μιας μονάδας AXI Lite interconnect. Η CPU διαχειρίζεται την αρχικοποίηση και τον έλεγχο των μονάδων βίντεο μέσω αυτής της θύρας.

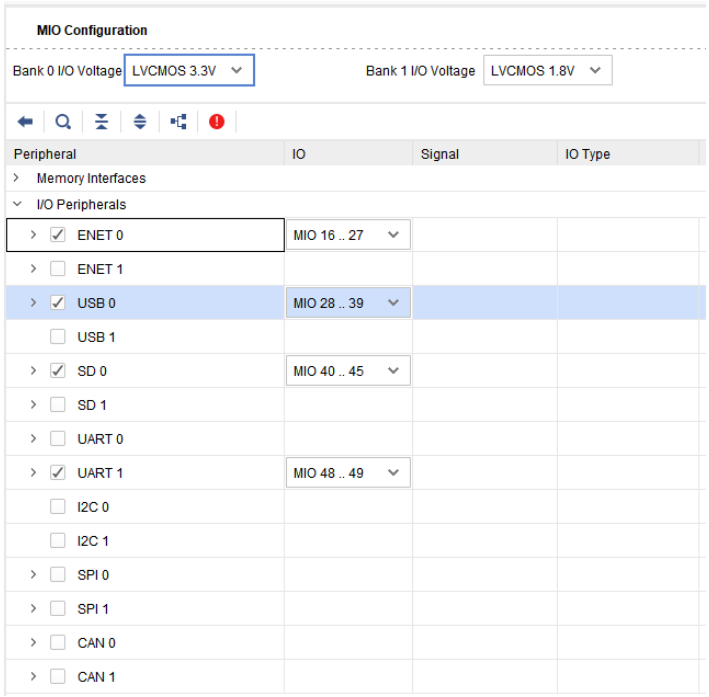
3. Input/Output peripherals (IOP)

Η μονάδα IOP περιλαμβάνει περιφερειακά επικοινωνίας του συστήματος επεξεργασίας. Για τα περισσότερα περιφερειακά, υπάρχει ευελιξία για τον τρόπο που μπορούν τα σήματα εισόδου / εξόδου να χαρτογραφηθούν(mapping). Οι δυνατότητες δρομολόγησης (routing) παρουσιάζονται στην Εικόνα 62, όπου φαίνεται ότι η πλειονότητα των σημάτων I / O για περιφερειακά PS, εκτός από USB, μπορεί να δρομολογηθεί είτε στους ακροδέκτες PS μέσω του MIO είτε στους ακροδέκτες PL μέσω του EMIO. Τα περισσότερα περιφερειακά διατηρούν επίσης το ίδιο πρωτόκολλο μεταξύ MIO και EMIO, εκτός από το Gigabit Ethernet.



Εικόνα 62: Routing Capabilities. Πηγή Εικόνας: [38]

Στην Εικόνα 63 παρουσιάζονται τα επιλεγμένα περιφερειακά (checkmarked) που είναι ενεργά και συμπεριλαμβάνονται στο σχέδιο, αφού γίνουν map στα αντίστοιχα επιλεγμένα MIO. Τα περιφερειακά που είναι επιλεγμένα χρησιμοποιούνται για πρόσβαση στο internet (ENET), τη σύνδεση με συσκευές USB (keyboard, mouse, webcam), τη σύνδεση με κάρτα microSD (SD0) και τη σειριακή επικοινωνία με έναν εξωτερικό υπολογιστή (UART). Τα μη ενεργά περιφερειακά είναι απενεργοποιημένα, μειώνοντας τη συνολική ισχύ που καταναλώνει το σύστημα κατά τη διάρκεια λειτουργίας του.



Εικόνα 63: I/O Peripherals Mapping

4. Memory Interfaces

Η μονάδα διασυνδέσεων μνήμης περιλαμβάνει τους ελεγκτές μνήμης DDR και τους ελεγκτές μη πτητικής (non-volatile) μνήμης. Ο ελεγκτής μνήμης DDR περιλαμβάνει έναν arbiter 4 θυρών. Μία θύρα AXI είναι αφιερωμένη στην πρόσβαση του επεξεργαστή ARM και δύο θύρες προορίζονται για συσκευές διασύνδεσης AXI master υψηλής απόδοσης, που βρίσκονται στην προγραμματιζόμενη λογική. Η υπολειπόμενη θύρα μοιράζεται από όλους τους άλλους AXI master. Η DDR3 έχει ρυθμιστεί για λειτουργία στα 533 MHz.

PL Reset: Το PS παρέχει τέσσερα FCLK_RESET[3:0] _N πλήρως προγραμματιζόμενα σήματα επαναφοράς στο PL. Αυτά τα σήματα σχετίζεται με το FCLK του ίδιου αριθμού, ωστόσο, ο χρονισμός είναι τέτοιος που πρέπει να θεωρούνται ασύγχρονα. Το block processor system reset σε αυτό το σχέδιο λαμβάνει είσοδο από το FCLK_RESET0_N και δημιουργεί τα απαραίτητα σήματα επαναφοράς για το σχεδιασμό που υλοποιείται στο PL.

PL Clocks: Το σύστημα επεξεργασίας (PS) παρέχει τέσσερα πλήρως προγραμματιζόμενα ρολόγια (FCLK_CLK) για χρήση στη προγραμματιζόμενη λογική (PL). Στο παρών σχέδιο χρησιμοποιούνται δύο τέτοια ρολόγια, FCLK_CLK0 και FCLK_CLK1, με 100MHz και 200MHz αντίστοιχα. Το FCLK_CLK1 μεταφέρεται σε μονάδα γεννήτριας ρολογιού στη PL όπου και μετατρέπεται σε 148.5 MHz για να σταλεί στο HDMI ελεγκτή. Οι υπόλοιπες μονάδες του συστήματος τροφοδοτούνται από το FCLK_CLK0.

4.4.1.2 Διαμόρφωση Προγραμματιζόμενης Λογικής (PL)

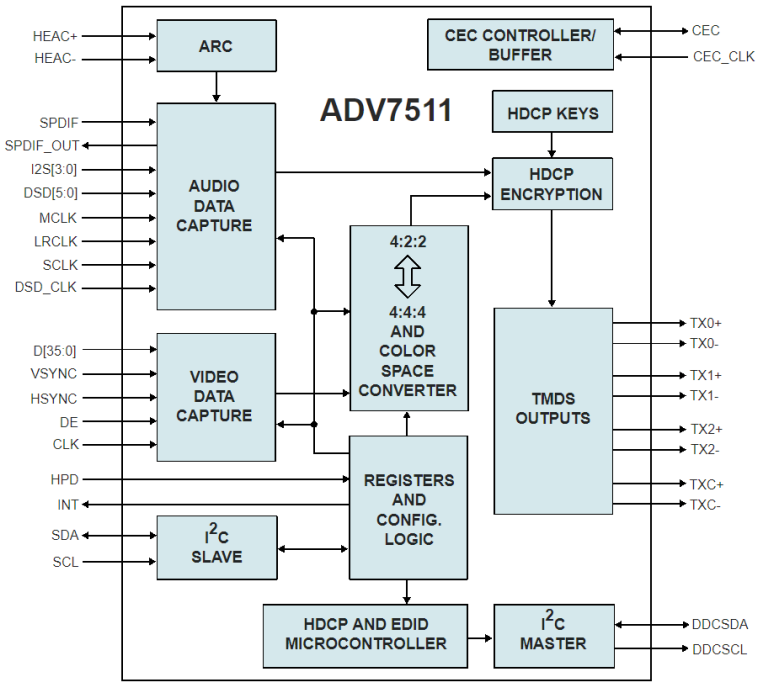
Τα κύρια μέρη της προγραμματιζόμενης λογικής (PL) διακρίνονται στα εξής IP blocks:

1. **HDMI**
2. **SOBEL FILTER**
3. **INTERRUPTS**
4. **INTERCONNECTS**

Ακολουθεί αναλυτική περιγραφή των παραπάνω:

1. **HDMI**

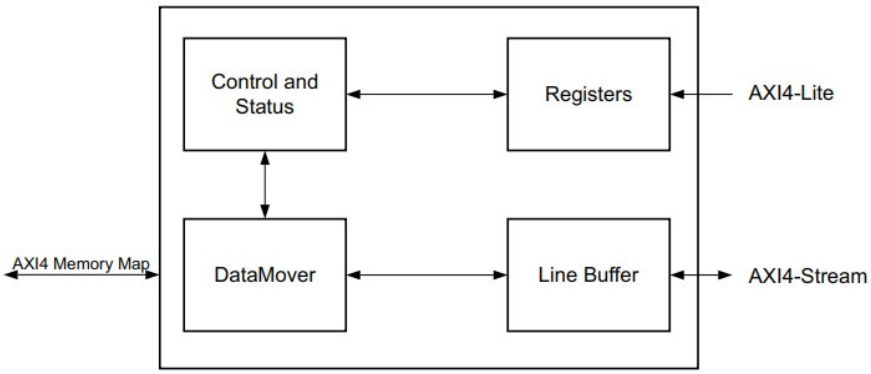
Όπως προαναφέρθηκε το συγκεκριμένο board διαθέτει έναν HDMI ADV7511 transmitter [Εικόνα 64]. Λογική στη PL μεριά πρέπει να υλοποιηθεί με σκοπό την επικοινωνία,την μεταφορά δεδομένων βίντεο καθώς και τη διαμόρφωση της λειτουργίας. [76]



Εικόνα 64: HDMI ADV7511 Transmitter Block Design.
Πηγή Εικόνας: [76]

axi_vdma_0: Πρόκειται για ένα DMA engine, προσαρμοσμένο για λειτουργίες video (Video DMA), το οποίο δίνεται από την Xilinx ως μέρος της δωρεάν άδειας αξιολόγησης. Το VDMA IP core έχει

σχεδιαστεί για να παρέχει δυνατότητες μεταφοράς read/write βίντεο από τον τομέα AXI4 memory-mapped στον τομέα AXI4-Stream και αντίστροφα. Το AXI VDMA παρέχει μεταφορά δεδομένων υψηλής ταχύτητας μεταξύ μνήμης του συστήματος και βίντεο IP με AXI4-Stream format. Ένα ρυθμιζόμενο ασύγχρονο line buffer χρησιμοποιείται για την προσωρινή αποθήκευση των δεδομένων εικονοστοιχείων πριν από την εγγραφή τους στη διεπαφή AXI4-Memory Map ή στη διεπαφή AXI4-Stream. Πιο συγκεκριμένα σε αυτή τη περίπτωση, μέσω του read channel παρέχεται ένα "μονοπάτι" για πρόσβαση στη μνήμη DDR και μετατροπή των δεδομένων βίντεο HDMI από AXI-Memory Map σε AXI-Stream μορφή, ώστε τελικά να μεταφερθούν στο HDMI ADV7511transmitter chip. Το DMA engine αποφορτίζει από τους πυρήνες του επεξεργαστή Zynq, τις λεπτομέρειες για τη διατήρηση ροής εξόδου βίντεο. Δέχεται ρολόι στα 100MHz και έχει οριστεί 64-bit data width ώστε να καλύπτεται ρυθμό ροής στα 800Mbytes/s, που είναι αρκετός ώστε να υποστηρίξει τις απαιτήσεις των 1080p. Η πρόσβαση στο σύστημα επεξεργαστή, και κατ' επέκταση στην μνήμη DDR, παρέχεται μέσω της διεπαφής S_AXI_HP0 ενώ οι registers προγραμματίζονται μέσω της διασύνδεσης AXI4-Lite ώστε να εκκινήσουν οι εντολών εγγραφής και ανάγνωσης στη διεπαφή AXI4 Master. [77]



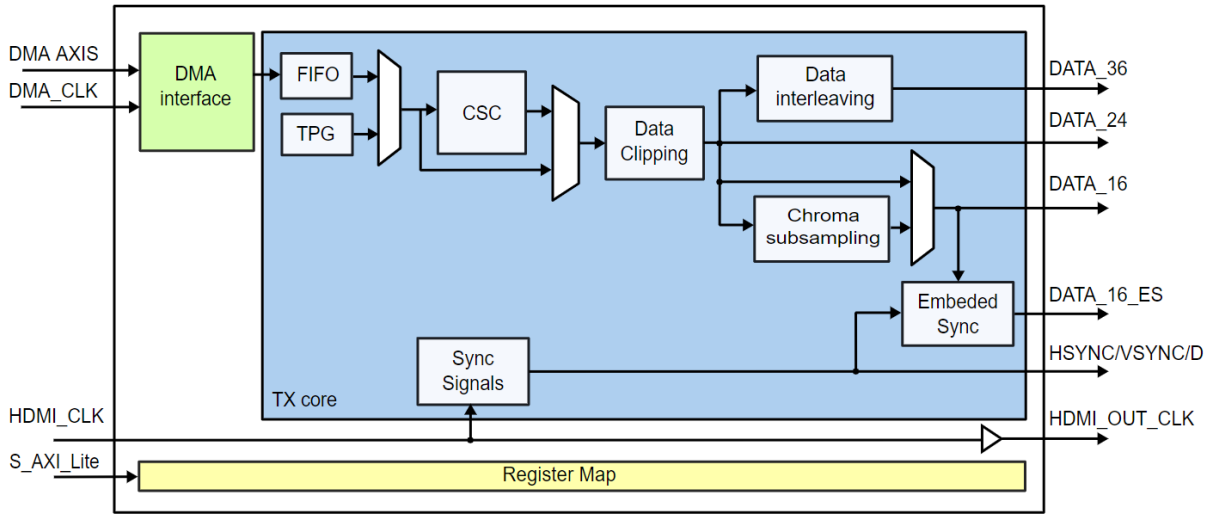
Εικόνα 65: VDMA Block Diagram

axi_iic_0: Αυτό το block λογικής παρέχει μια σύνδεση στον επεξεργαστή με τον πομπό HDMI, μέσα από το δίαυλο επικοινωνίας I2C, ώστε να διαμορφώσει τη λειτουργία του. Μέσω software ελέγχεται ο δίαυλος I2C ώστε να οριστούν οι απαραίτητοι καταχωρητές σύμφωνα με τις ανάγκες του συστήματος. Δύο σήματα, Serial Clock (SCL) και Serial Data (SDA), συντελούν το ρολόι που συγχρονίζει τη μεταφορά δεδομένων στο δίαυλο I2C και τη γραμμή που μεταφέρει τα δεδομένα, αντίστοιχα.

axi_clkgen: Αυτό το IP core παρέχεται από την Analog Devices και η χρήση του είναι να παίρνει τα 200MHz του FCLK_CLK1 από το PS ως είσοδο και δημιουργεί έξοδο ρολογιού 148,6MHz που χρησιμοποιείται ως είσοδος στο axi_hdmi_core.

axi_hdmi_tx: Αυτό το IP core παρέχεται από την Analog Devices και λειτουργεί σαν μια γέφυρα μεταξύ του Zynq - 7000 All Programmable SoC και του ADV7511 HDMI transmitter chip για δεδομένα βίντεο. Παρέχει πληροφορίες ελέγχου στο chip για να επεξεργαστεί τη ροή βίντεο που φθάνει μέσω του VDMA. Σύμφωνα με το documentation από την ιστοσελίδα της Analog Devices

[78], ο πομπός ADV7511 μπορεί να δεχτεί από 8-bit εώς και 36-bit δεδομένα εισόδου ανάλογα με το board που χρησιμοποιείται. Το axi_hdmi_tx IP core για το Zedboard, εξάγει δεδομένα βίντεο 16-bits με σήματα HSYNC, VSYNC και Data Enable(DE) για την είσοδο στο ADV7511 chip. Επιπρόσθετα δέχεται ρολόι στα 148.5MHz ώστε να καλύπτει τις απαιτήσεις των 1080p.



Εικόνα 66: AXI_HDMI_TX Block Diagram. Πηγή Εικόνας: [78]

2. SOBEL FILTER

axi_vdma_1: Για την υλοποίηση του video pipeline που αφορά το φίλτρο Sobel είναι αναγκαία η χρήση ενός διαφορετικού VDMA engine από αυτό που χρησιμοποιείται για την έξοδο video. Πιο συγκεκριμένα εδώ γίνεται χρήση και του καναλιού γραψίματος αλλά και του καναλιού διαβάσματος. Αφού προγραμματιστούν οι καταχωρητές μέσω της διεπαφής AXI4-Lite, το AXI VDMA χρησιμοποιεί τη διεπαφή AXI4 Master για ανάγνωση frames από τη μνήμη του συστήματος και το εξάγει στη διεπαφή AXI4-Stream Master. Από εκεί δεδομένα AXI4-Stream με πλάτος 24-bit μεταφέρονται στο Sobel IP core και αφού εφαρμοστεί το φίλτρο επιστρέφουν δεδομένα AXI4-Stream με πλάτος 8-bit. Το AXI VDMA δέχεται αυτά τα δεδομένα (frames) στη διεπαφή AXI4-Stream Slave και τα γράφει στη μνήμη του συστήματος χρησιμοποιώντας τη διεπαφή AXI4-Master. Εδώ αξίζει να υπενθυμιστεί ότι στο κεφάλαιο 4.3, το φίλτρο δέχεται δεδομένα πλάτους 24-bit για εικόνα RGB τριών καναλιών και επιστρέφει grayscale εικόνα 8-bit ενός καναλιού.

Sobel_0: Πρόκειται για το IP core που δημιουργήθηκε στο 4.3. Διαθέτει μια διεπαφή AXI4-Slave μέσω της οποίας διαμορφώνεται και ελέγχεται, καθώς και δύο διεπαφές AXI4-Stream για έισοδο και έξοδο δεδομένων από και προς το axi_vdma_1. Χρονίζεται όπως όλο το υπόλοιπο σύστημα, εκτός του axi_hdmi, στα 100MHz από το FCLK_CLK1 του PS.

3. INTERRUPTS

xlconcat_0: Αυτό το block συνδυάζει πολλές διακοπές από τα VDMA και IIC IP blocks και στη συνέχεια συνδέεται με την είσοδο διακοπής στο Zynq PS.

4. INTERCONNECTS

AXI_INTERCONNECT_GP: Το σχέδιο περιλαμβάνει 2 interconnects για διεπαφές AXI register. Οι θύρες του PS, M_AXI_GP0 και M_AXI_GP1, λειτουργούν ως master σε αυτή τη διασύνδεση και τα συνδεδεμένα IP cores ως slaves. Στο M_AXI_GP0 συνδέονται όλα τα IP cores που αφορούν τη προβολή βίντεο (vdma_0, axi_hdmi, axi_iic, axi_clkgen) ενώ στο M_AXI_GP1 τα IP cores που αφορούν την εφαρμογή φίλτρου (vdma_1, Sobel_0). Πρόκειται για κανάλια διεπαφής γενικού σκοπού (GP=General Purpose) και αφορούν κυρίως σήματα ελέγχου από τον επεξεργαστή, γεγονός που οδηγεί σε σχετικά χαμηλό bandwidth.

AXI_INTERCONNECT_HP: Υπάρχουν δύο καταναλωτές υψηλού εύρους ζώνης στο σύστημα. Ο ένας αφορά την μεταφορά δεδομένων βίντεο από το vdma_0 στην έξοδο HDMI και ο άλλος την μεταφορά δεδομένων βίντεο από το vdma_1 για την εφαρμογή του φίλτρου. Γιαυτό το λόγο χρησιμοποιούνται δύο διασυνδέσεις υψηλής απόδοσης μέσω των θυρών S_AXI_HP0 και S_AXI_HP1 του PS. Για να αυξηθεί το μέγιστο bandwidth σε ένα σύστημα πρέπει είτε να αυξηθεί ο ρυθμός ρολογιού είτε να αυξηθεί το πλάτος της διαδρομής δεδομένων. Όμως το πλάτος της διαδρομής δεδομένων για τη θύρα HP είναι σταθερό και δεν μπορεί να υπερβεί τα 64-bit. Ο λόγος που χρησιμοποιούνται 2 AXI_INTERCONNECT_HP είναι για να συνδεθούν τα δύο DMAs σε διαφορετικά interconnects και να επιτευχθεί το καλύτερο δυνατό throughput.

4.4.1.3 Σύνθεση Υλικού και εξαγωγή bitstream

Αφού γίνει η εισαγωγή, παραμετροποίηση και διασύνδεση όλων των IP blocks που συζητήθηκαν στα προηγούμενα υποκεφάλαια, αναγκαία είναι και η δημιουργία ενός αρχείου το οποίο θα αντιστοιχεί τα τα σήματα εξόδου HDMI (data, hsync, vsync, clk, DE) με τα πραγματικά pins του Zedboard. Στην συνέχεια για την εξαγωγή του hardware και προγραμματισμό της PL σε πραγματικό υλικό (Zedboard) ακολουθούν τα εξής βήματα:

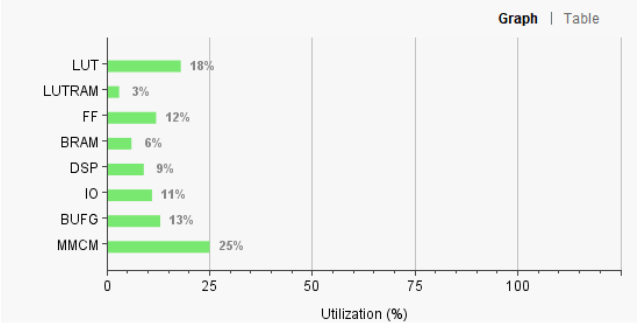
Output Products Generation: Με αυτή την εντολή παράγονται τα πηγαία αρχεία για τα IP cores που χρησιμοποιούνται στη σχεδίαση του υποσυστήματος, το σχετικό αρχείο constraints που αντιστοιχίζει τα σήματα με τα pins, αρχεία προσομοίωσης κα.

HDL Wrapper Creation: Αυτή η εντολή είναι αναγκαία διότι το block design δεν μπορεί να συντεθεί άμεσα. Η δημιουργία ενός HDL wrapper για τον τρέχοντα σχεδιασμό, το καθιστά ως το top-level με αποτέλεσμα να επιτρέπεται πλέον η σύνθεση, υλοποίηση και παραγωγή του bitstream

Generate bitstream: Σε αυτό το σημείο εκτελείται η εντολή για τη δημιουργία του αρχείου bitstream. Αυτό το αρχείο περιέχει τις απαραίτητες πληροφορίες του σχεδίου που υλοποιήθηκε και ο προγραμματισμός της FPGA αποτελεί ουσιαστικά διαδικασία φόρτωσης αυτού του αρχείου στη μνήμη. Για την παραγωγή του bitstream πρέπει πρώτα να πραγματοποιηθούν οι λειτουργίες Synthesis και Implementation, όπως αυτές περιγράφηκαν στο κεφάλαιο 4.1.1.

4.4.1.4 Πόροι Συστήματος

Παρακάτω παρουσιάζεται η χρήση των πόρων που θα δεσμεύσει το Zedboard κατά τον προγραμματισμό του:

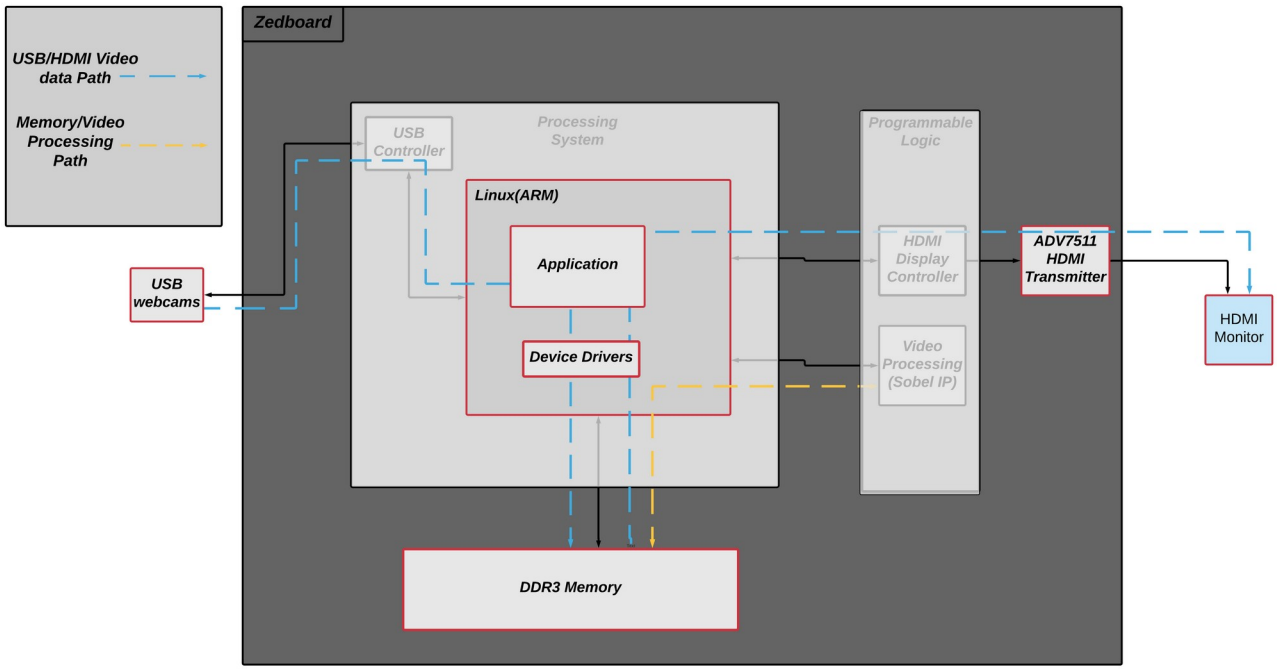


Resource	Utilization	Available	Utilization %
LUT	9628	53200	18.10
LUTRAM	517	17400	2.97
FF	12442	106400	11.69
BRAM	9	140	6.43
DSP	20	220	9.09
IO	22	200	11.00
BUFG	4	32	12.50
MMCM	1	4	25.00

Εικόνα 67: Resources Utilization

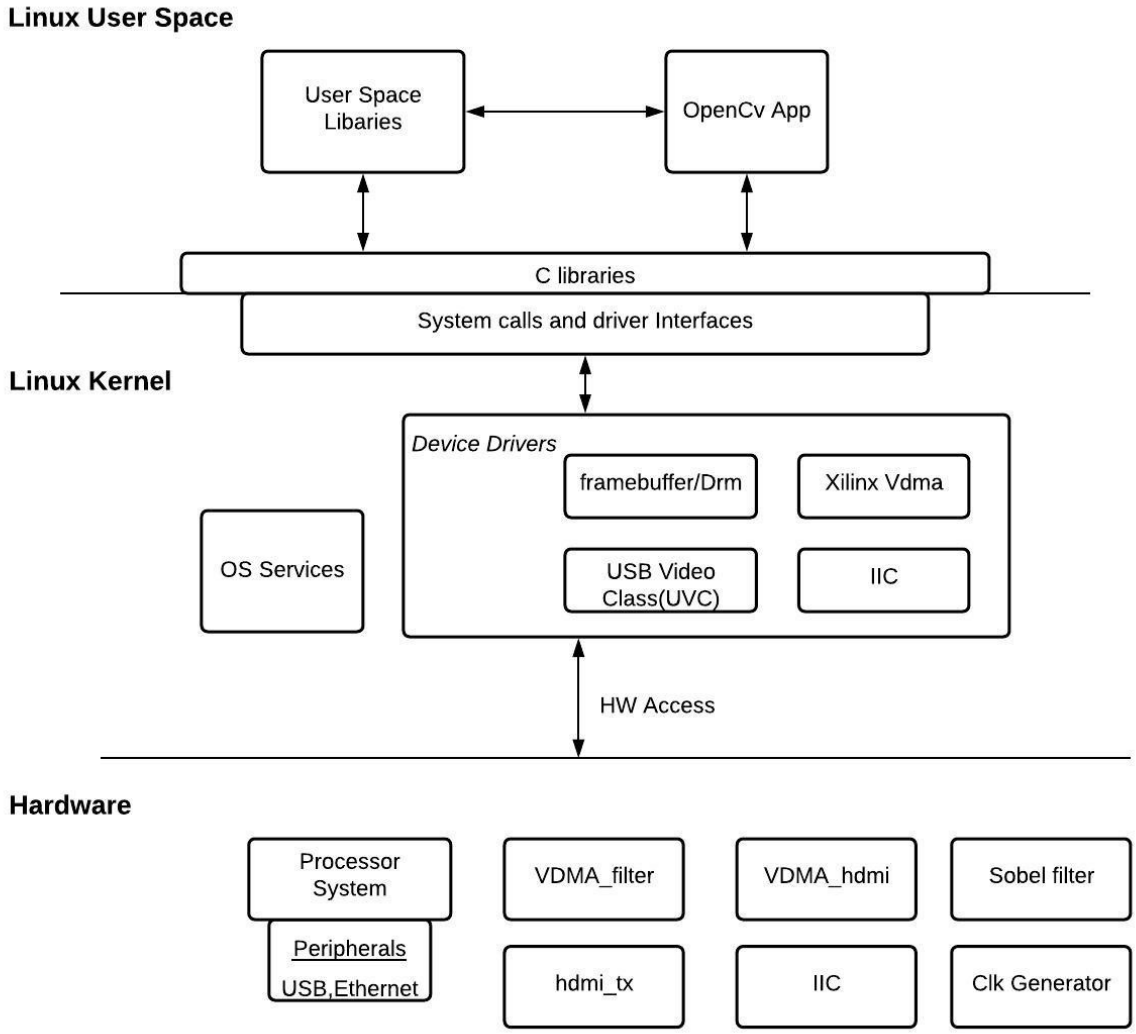
4.4.2 Αρχιτεκτονική Συστήματος Software

Σε αυτό το κεφάλαιο θα παρουσιαστεί η δημιουργία ενός ενσωματωμένου συστήματος Linux, ώστε να γίνει χρήση της θύρας HDMI και USB, για σύνδεση συσκευών όπως USB webcam και ποντίκι/πληκτρολόγιο [Εικόνα 68]. Ο κύριος λόγος που γίνεται χρήση περιβάλλοντος Linux και όχι Standalone (no-OS) είναι η εκμετάλλευση του ήδη υπάρχοντος USB Software Stack για συστήματα Linux. Διατίθεται μια πληθώρα από επιλογές για προγράμματα οδήγησης USB που πραγματοποιούν την ενσωμάτωση των USB κλάσεων, όπως αυτή της κλάσης βίντεο που αφορά τη USB κάμερα, αλλά και οδηγοί για τη διεπαφή ULPI. Η δημιουργία αυτών των οδηγών από το μηδέν για Standalone (no-OS) περιβάλλον, είναι μια διαδικασία που θα απαιτούσε μακρά διάρκεια ενασχόλησης και εξειδικευμένη τεχνογνωσία για την έρευνα, ανάπτυξη και υλοποίηση τους, αφού το USB αποτελεί ένα αρκετά περίπλοκο πρωτόκολλο.



Εικόνα 68: Software Block Design

Σκοπός αυτής της ενότητας είναι η μεταφορά όλης της λογικής που διαμορφώθηκε στη προηγούμενη ενότητα, σε περιβάλλον Linux. Αυτό θα συμβεί κάνοντας χρήση των κατάλληλων προγραμμάτων οδήγησης συσκευών για όλες τις μονάδες υλικού που φαίνονται στο top-level σχήμα [Εικόνα 69], ώστε να είναι προσβάσιμες από τον χρήστη. Η αρχιτεκτονική του συστήματος λογισμικού φαίνεται στο παρακάτω σχήμα:



Εικόνα 69: System Software Architecture

4.4.2.1 Εισαγωγή hardware

Η διαδικασία υλοποίησης του ενσωματωμένου συστήματος Linux ξεκινάει με την εντολή δημιουργίας Petalinux project, μέσω της οποίας ρυθμίζεται κατάλληλα το όνομα του project καθώς και ο τύπος του επεξεργαστή που θα χρησιμοποιηθεί (Ultrasclae+ MPSoC, Zynq, Microblaze).

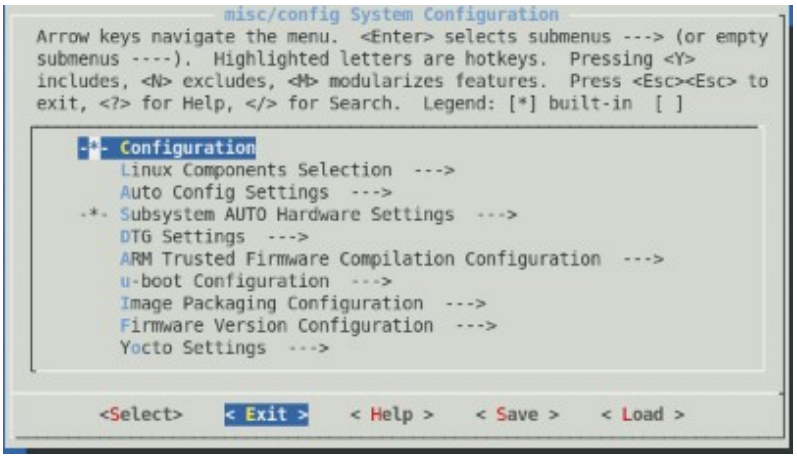
petalinux-create --type project –template zynq --name USBHDMI

Έπειτα το PetaLinux πρέπει να γνωρίζει το hardware του συστήματος. Αυτό πραγματοποιείται εισάγοντας το αρχείο bitstream που δημιουργήθηκε στο Vivado. Πρόκειται για ένα αρχείο που κατά την εξαγωγή του από το Vivado συντάσσεται ως αρχείο περιγραφής υλικού (*.hdf) και περιλαμβάνει όλες τις ρυθμίσεις που αφορούν τόσο το σύστημα επεξεργασίας (PS) όσο και τα block λογικής που υπάρχουν στη προγραμματιζόμενη λογική (PL). Με βάση αυτές τις πληροφορίες που περιέχονται στο αρχείο περιγραφής υλικού, το PetaLinux δημιουργεί αρχεία κεφαλίδας (header files)

u-boot, το αρχείο προέλευσης device tree και επιτρέπει την εφαρμογή των κατάλληλων προγραμμάτων οδήγησης για τον πυρήνα Linux.

petalinux-config --get-hw-description=<path_to_hardware_description_file >

Η παραπάνω εντολή ενημερώνει το Petalinux για το hardware που θα χρησιμοποιηθεί. Με την εκτέλεση της θα εμφανίσει ένα menu διαμόρφωσης συστήματος με τη μορφή ενός ASCII GUI [Εικόνα 70]. Μέσα από αυτό το menu πραγματοποιούνται ρυθμίσεις που διαμορφώνουν το Boot loader, το οποίο είναι υπεύθυνο για τη διαδικασία εκκίνησης του συστήματος. Κατά την ενεργοποίηση του συστήματος διαβάζεται ο καταχωρητής λειτουργίας εκκίνησης για να καθοριστεί ο τρόπος με τον οποίο θα εκκινήσει. Για τους σκοπούς αυτού του σχεδιασμού, η λειτουργία εκκίνησης έχει ρυθμιστεί για εκκίνηση μέσω κάρτας SD. Επιπλέον σε αυτό το σημείο εισάγεται ο περιορισμός στον πυρήνα του Linux για χρήση των χαμηλότερων 384MB από τα 512MB που υπάρχουν συνολικά στη μνήμη του συστήματος. Αυτός ο περιορισμός εφαρμόζεται χάρη στο VDMA που χρησιμοποιείται για το φίλτρο Sobel. Το VDMA engine έχει πρόσβαση και στα 512MB της φυσικής μνήμης της αναπτυξιακής πλακέτας Zedboard μέσω της AXI High Perfomance θύρας, έτσι ελοχεύει ο κίνδυνος για τον πυρήνα του Linux να εκχωρήσει την περιοχή μνήμη του VDMA σε άλλες εφαρμογές και κατ' επέκταση σε γενική διαφθορά μνήμης. Με αυτή τη μέθοδο ο πυρήνας δεν μπορεί να χρησιμοποιήσει την περιοχή μνήμης μεταξύ 512MB και 384MB για άλλες διαδικασίες και threads. Το ίδιο πρόβλημα δεν ισχύει για το VDMA που αφορά το HDMI video output, διότι εισάγεται ως kernel module με τον πυρήνα να αναλαμβάνει όλες τις απαραίτητες διαδικασίες για την αντιστοίχιση μεταξύ εικονικών διευθύνσεων μνήμης και φυσικών διευθύνσεων.



Εικόνα 70: System Config Window

4.4.2.2 Διαμόρφωση Πυρήνα

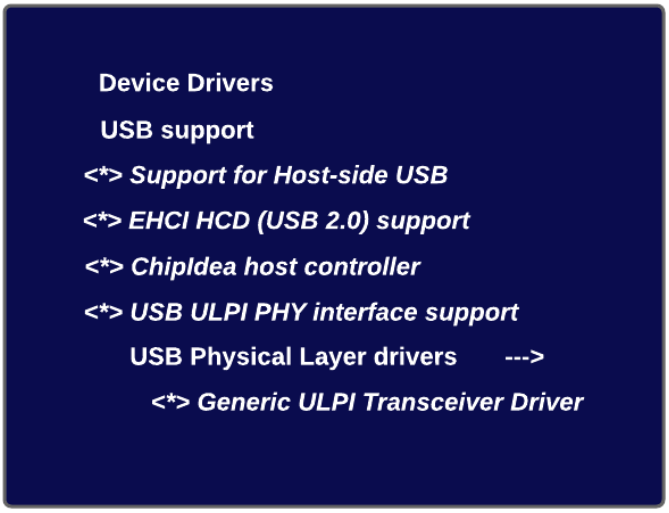
Ο πυρήνας Linux είναι υπεύθυνος για την εκκίνηση και τη διαχείριση των διαδικασιών, των πόρων και των περιφερειακών εφαρμογών που χρησιμοποιούν το λειτουργικό σύστημα. Στην ουσία ο πυρήνας αποτελεί τον "συνδετικό κρίκο" μεταξύ του λειτουργικού συστήματος και του υλικού. Με την ολοκλήρωση του προηγούμενου βήματος, ο πυρήνας Linux διαμορφώνεται κατάλληλα ώστε να

ενεργοποιηθούν τα περισσότερα από τα απαραίτητα προγράμματα οδήγησης. Τα πιο εξειδικευμένα προγράμματα οδήγησης συσκευών όμως, πρέπει να ενεργοποιηθούν χειροκίνητα. Εδώ πρέπει να γίνει ενεργοποίηση των οδηγών που αφορούν το USB και το HDMI output. Με την εκτέλεση της παρακάτω εντολής ένα νέο menu μορφής ASCII GUI ανοίγει και ο σχεδιαστής έχει τη δυνατότητα ενεργοποίησης των επιθυμητών προγραμμάτων οδηγών συσκευής.

petalinux-config -c kernel

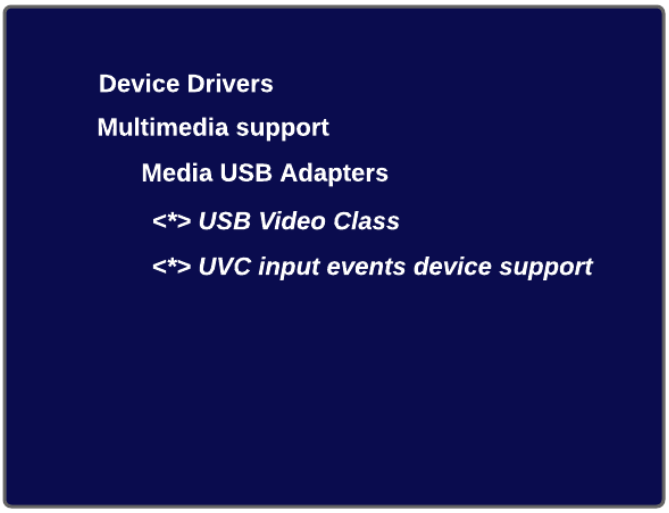
USB Drivers

Για την χρήση μιας USB κάμερας χρειάζεται να ενεργοποιηθούν αρκετοί οδηγοί που συντελούν το USB software stack. Σύμφωνα με το USB Host Software Stack πρώτα πρέπει να ενεργοποιηθεί ο οδηγός που αφορά τη ***διεπαφή ULPI***. Το UTMI + Low Pin Interface (ULPI) είναι η προδιαγραφή που χρησιμοποιείται για τη διεπαφή USB 2.0 PHY. Μέσω αυτής καθορίζεται ένα σύνολο καταχωρητών που μπορούν να χρησιμοποιηθούν για την ανίχνευση του host και της device στην USB επικοινωνία, επιτρέποντας έτσι τον χειρισμό του ULPI ως δίαυλο. Αυτό το δίαυλο χειρίζεται το πρόγραμμα οδήγησης USB ULPI PHY. Οι διεπαφές ULPI καταχωρούνται από τα προγράμματα οδήγησης των ελεγκτών USB που υποστηρίζουν πρόσβαση στους καταχωρητές ULPI και έχουν προσαρτημένο ένα ULPI PHY chip. Μια USB webcam συγκαταλέγεται στις συσκευές που ακολουθούν τη προδιαγραφή USB 2.0 High Speed (480Mbit/sec, 60 Mbyte/sec). Γιαυτό το λόγο το πρόγραμμα οδήγησης του ελεγκτή USB (***ChipIdea Host Controller Driver***) διαμορφώνει τον USB ελεγκτή υλικού με την υποστήριξη της πρότυπης ***διεπαφής EHCI***, ώστε να υποστηρίζεται το πρότυπο USB 2.0 αλλά και να διαμορφωθεί η USB host λειτουργία του. Οι ελεγκτές EHCI είναι συσκευασμένοι με "συνοδευτικούς" host ελεγκτές (OHCI ή UHCI) για τον χειρισμό συσκευών USB 1.1 συνδεδεμένων σε θύρες root hub. Οι θύρες θα συνδεθούν στο EHCI εάν η συσκευή είναι υψηλής ταχύτητας (High Speed), διαφορετικά συνδέονται με έναν συνοδευτικό ελεγκτή.



Εικόνα 71: USB Drivers for Host Functionality

Με την παραπάνω διαδικασία το λογισμικό Linux θα είναι σε θέση να δέχεται και να προχωρά στη διαδικασία του enumeration μιας USB συσκευής. Όμως για να είναι πλήρως λειτουργική μια USB συσκευή, πρέπει πρώτα να οριστεί η κλάση της. Χρησιμοποιώντας έναν πρόγραμμα οδήγησης για τον καθορισμό της κλάσης προσδιορίζεται η λειτουργία της συσκευής. Η USB Video Class (UVC) είναι μια προδιαγραφή USB που τυποποιεί τη λειτουργία αποστολής βίντεο μέσω της θύρας USB. Το λειτουργικό σύστημα Linux παρέχει υποστήριξη για συσκευές που βασίζονται σε UVC, όπως κάμερες web, λειτουργώντας ως host για τη συσκευή. Ενεργοποιώντας το USB Video Class driver, το Linux θα είναι σε θέση κατά την εισαγωγή μιας USB webcam στο σύστημα να δημιουργήσει ένα κόμβο για τη συσκευή και να του αναθέσει την ονομασία "/dev/video0". Αυτός ο κόμβος επιτρέπει σε user space προγράμματα να έχουν πρόσβαση στη USB webcam, χρησιμοποιώντας το πρόγραμμα οδήγησης της συσκευής μέσω τυπικών κλήσεων συστήματος (system calls) εισόδου / εξόδου.



Εικόνα 72: USB Video Class Drivers

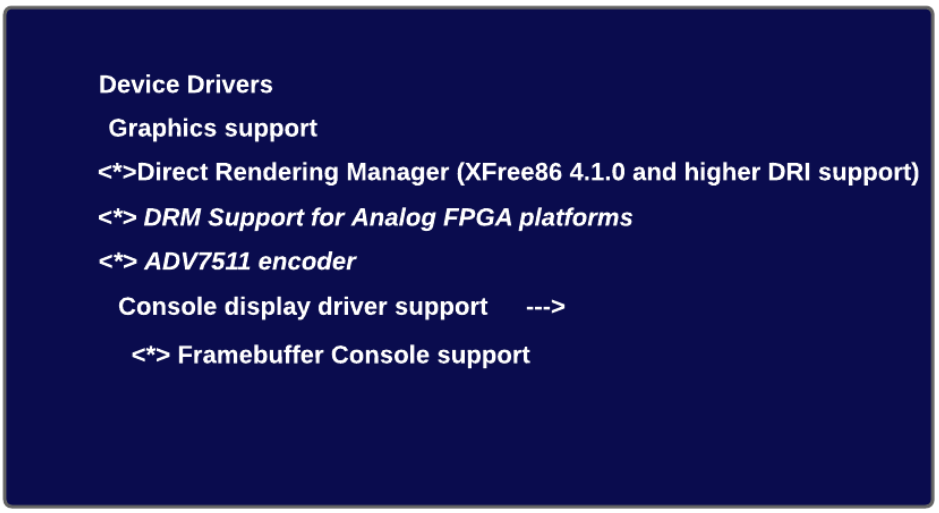
HDMI Drivers

Στο κεφάλαιο [3.3Software Implementation] έγινε αναφορά στο DRM/KMS driver που θα χρησιμοποιηθεί για την υλοποίηση της εξόδου HDMI. Επιπλέον στο κεφάλαιο Αρχιτεκτονική Συστήματος έγινε αναφορά για δύο IP cores που ανήκουν στην Analog devices, το axi_hdmi_tx και το axi_clkgen. Αυτά τα Third-Party IPs δεν ανήκουν στο κύριο πυρήνα που διατίθεται από τη Xilinx και προκειμένου να χρησιμοποιηθούν στο λογισμικό Linux θα πρέπει να προστεθούν. Η Analog devices παρέχει δωρεάν τα kernel modules που αντιστοιχούν σε αυτά τα IPs οπότε ο σχεδιαστής χρειάζεται μόνο να τα προσθέσει στο σύστημα. Με την εκτέλεση της εντολής:

petalinux-create -t modules -n name

και για τα δύο IPs, προστίθενται στο σύστημα δύο kernel modules. Αυτό πρακτικά σημαίνει ότι δεν θα ενεργοποιηθούν δυναμικά κατά την εκκίνηση (boot) του συστήματος Linux, αλλά θα πρέπει ο

χρήστης να τα εισάγει και ενεργοποιήσει με το πέρας της διαδικασίας εκκίνησης και σύνδεσης του στο σύστημα. Επιπλέον χρειάζεται να ενεργοποιηθούν κάποια προγράμματα οδήγησης για υποστήριξη της διαμόρφωσης του ADV7511 transmitter ως encoder στη DRM/KMS υποδομή και γενικότερα των FPGA πλατφορμών. Μέσα από αυτά παρέχεται υποστήριξη για συγχρονισμό, ασφάλεια και μεταφορές DMA.



Εικόνα 73: DRM Support Drivers

Με την ενεργοποίηση όλων των παραπάνω καταχωρείται στο σύστημα ένα κόμβος που αντιστοιχεί σε συσκευή framebuffer και να της αναθέτει την ονομασία *"/dev/fb0"*. Μέσω αυτού του κόμβου δημιουργείται μια διεπαφή όπου user-space προγράμματα έχουν πρόσβαση στην περιοχή της μνήμης που αντιστοιχεί στο video hardware.

4.4.2.3 Διαμόρφωση Device Tree

Το device tree στο Linux αποτελεί μια βάση δεδομένων που χρησιμοποιείται ως ένα μέσο από τον πυρήνα για αναγνώριση του υλικού που υπάρχει στο σύστημα και την αντιστοίχιση του με τα κατάλληλα προγράμματα οδήγησης κατά την εκκίνηση του συστήματος. Επιπλέον μέσω αυτού θέτονται οι απαραίτητες ιδιότητες όλων των συσκευών υλικού όπως interrupts, σχετικοί καταχωρητές και ρολόγια. Αρκετές από αυτές τις ιδιότητες αναθέτονται αυτόματα κατά τη διαμόρφωση του πυρήνα. Και εδώ οι ρυθμίσεις που πρέπει να γίνουν αφορούν τη διαμόρφωση του USB και του HDMI.

USB Device Tree

Στην περίπτωση του USB αρκετές ιδιότητες έχουν ρυθμιστεί σωστά από την προεπιλογή του πυρήνα. Ο USB controller αντιστοιχίζεται σωστά στη base address και το μήκος των καταχωρητών που έχουν τεθεί από το Vivado καθώς καλείται και το κατάλληλο πρόγραμμα οδήγησης, "chipIdea,usb2". Επιπλέον σωστές είναι και οι ρυθμίσεις που έχουν γίνει για το ρολόι και τις διακοπές που δέχεται . Ο χρήστης όμως, χρειάζεται να δημιουργήσει ένα καινούριο node [Εικόνα 75]

στο device tree το οποίο είναι υπεύθυνο για την αντιστοίχιση του USB PHY με τον αντίστοιχο ULPI driver που συζητήθηκε παραπάνω. Στην συνέχεια το node που αντιστοιχεί στον ελεγκτή USB [Εικόνα 74] θα πρέπει να διαμορφωθεί κατάλληλα ώστε να έχει πρόσβαση στο PHY node μέσω της ιδιότητας "*phandle*". Τέλος η ιδιότητα "*dr_mode*" πρέπει να οριστεί ως "*host*" για να περάσει σαν όρισμα στο πρόγραμμα οδήγησης του USB ελεγκτή και να του επιτρέπει να λειτουργεί ως host.

```
phy0@e0002000 {
    compatible = "ulpi-phy";
    #phy-cells = <0x0>;
    reg = <0xe0002000 0x1000>;
    view-port = <0x170>;
    drv-vbus;
    linux,phandle = <0x7>;
    phandle = <0x7>;
};
```

Εικόνα 75: USB-PHY Device Tree

```
usb@e0002000 {
    compatible = "xlnx,zynq-usb-2.20a", "chipidea,usb2";
    status = "okay";
    clocks = <0x1 0x1c>;
    interrupt-parent = <0x4>;
    interrupts = <0x0 0x15 0x4>;
    reg = <0xe0002000 0x1000>;
    phy_type = "ulpi";
    dr_mode = "host";
    usb-phy = <0x7>;
};
```

Εικόνα 74: USB Device Tree

HDMI Device Tree

Το πρόγραμμα οδήγησης για το HDMI αποτελεί έναν πρόγραμμα οδήγησης πλατφόρμας [σελ. 66] και μπορεί να δημιουργηθεί μόνο μέσω του device tree. Αυτή η ενότητα, που θα δημιουργηθεί στο device tree, αφορά τα IP cores που προστέθηκαν στην προγραμματιζόμενη λογική κατά τη δημιουργία του σχεδίου στο Vivado. Σύμφωνα με την Analog Devices το πρόγραμμα οδήγησης για το HDMI υλοποιείται ως οδηγός συσκευής DRM/KMS. Ο οδηγός του axi_hdmi_tx λειτουργεί ως master αφού δέχεται ως ορίσματα τους οδηγούς συσκευών των άλλων IPs, που συντελούν κομμάτια του συστήματος DRM/KMS. Πιο συγκεκριμένα ο κόμβος που αντιστοιχεί στο axi_hdmi_tx IP [Εικόνα 77], πρέπει να αναφέρεται σε 3 άλλους κόμβους του device tree, που αφορούν τον HDMI ADV7511 transmitter, το ρολόι με το οποίο χρονίζεται καθώς και το video DMA που χρησιμοποιείται για πρόσβαση στη μνήμη. Ο οδηγός συσκευής ADV7511 αντιμετωπίζεται ως encoder και connector στο σύστημα του DRM/KMS [Εικόνα 36]. Η αλληλεπίδραση με το ADV7511 chip γίνεται μέσα από τον δίαυλο I2C. Εκεί ο χρήστης έχει τη δυνατότητα να ρυθμίσει το chip ορίζοντας ιδιότητες όπως το χρωματικό βάθος ("*adi,input-depth*"), τη μορφή διαρρύθμισης των χρωμάτων στους καταχωρητές ("*adi,input-style*"), ρολόγια εισόδου ("*adi,input-clock*") κ.α [Εικόνα 76]. Ο οδηγός συσκευής για το VDMA αποτελεί slave ως προς τον οδηγό axi_hdmi_tx, όμως η πρόσβαση σε αυτόν δεν γίνεται άμεσα. Ο χειρισμός του ανατίθεται στο CRTC κομμάτι του DRM/KMS μέσω του οποίου διαμορφώνονται οι απαραίτητες παράμετροι ώστε να μπορέσει να ξεκινήσει τη λειτουργία του. Έτσι, το πρόγραμμα οδήγησης δίνει εντολή στο VDMA IP να ξεκινήσει τη μεταφορά DMA από τη μνήμη στο axi_hdmi_tx IP. Τέλος δημιουργείται και ένας κόμβος για την αντιστοίχιση του axi_clkgen IP με το πρόγραμμα οδήγησης του ώστε να δοθεί ως ρολόι στον master του συστήματος DRM/KMS, axi_hdmi_tx.

```
&axi_iic_0 {
    adv7511: adv7511@39 {
        compatible = "adi,adv7511";
        reg = <0x39>, <0x3f>;
        reg-names = "primary", "edid";

        adi,input-depth = <8>;
        adi,input-colorspace = "yuv422";
        adi,input-clock = "1x";
        adi,input-style = <1>;
        adi,input-justification = "right";
        adi,clock-delay = <0>;

        ports {
            #address-cells = <1>;
            #size-cells = <0>;

            port@0 {
                reg = <0>;
                adv7511_in: endpoint {
                    remote-endpoint = <&axi_hdmi_out>;
                };
            };
        };
    };
};
```

Εικόνα 76: Ρύθμιση ADV7511 chip μέσω I2C δίαυλου

```
&axi_hdmi_tx_0 {
    compatible = "adi,axi-hdmi-tx-1.00.a";
    dmas = <&axi_vdma_0 0>;
    dma-names = "video";
    clocks = <&axi_clkgen_0>;

    port {
        axi_hdmi_out: endpoint {
            remote-endpoint = <&adv7511_in>;
        };
    };
};
```

Εικόνα 77: DRM/KMS Master Node

Τα δύο IP blocks που αντιστοιχούν στο δεύτερο video DMA και το φίλτρο Sobel δεν διαμορφώνονται μέσω του device tree, αλλά η χαρτογράφηση της φυσικής τους μνήμης σε εικονική καθώς και η διαρρύθμιση των καταχωρητών τους, γίνεται χειροκίνητα μέσα από την εφαρμογή λογισμικού που θα περιγραφεί στο κεφάλαιο 5.

4.4.2.4 Δημιουργία Linux Image και File System

Με την ολοκλήρωση των παραπάνω διαδικασιών εκτελείται η εντολή:

petalinux-build

μέσω της οποίας παράγονται τα απαραίτητα αρχεία για τη λειτουργία του συστήματος. Πιο συγκεκριμένα παράγονται τα αρχεία που είναι υπεύθυνα για το First Stage BootLoader (FSBL) και το Second Stage BootLoder (u-boot) ώστε να φορτωθεί η εικόνα και το File System κατά την εκκίνηση του συστήματος. Αυτά τα δύο αρχεία, σε συνδυασμό με το αρχείο που περιγράφει την υλοποίησης του υλικού στο Vivado (bitstream) συντελούν το αρχείο **BOOT.BIN**, το οποίο απλά συγκεντρώνει όλες τις λειτουργίες των προαναφερθέντων. Επιπλέον ένα αρχείο, **image.ub,** περιέχει την εικόνα του πυρήνα καθώς και ένα αρχείο **devicetree.dtb** το οποίο προσδιορίζει τη διαμόρφωση του υλικού στον πυρήνα. Τέλος πρέπει να επιλεχθεί ένα σύστημα αρχείων (**File System**) για τη διαχείριση και οργάνωση των δεδομένων αποθήκευσης στον αποθηκευτικό χώρο. Το σύστημα αρχείων που παράγεται αυτόματα από το Petalinux είναι ικανό να συμπεριλάβει τις απαραίτητες βιβλιοθήκες για υποστήριξη στην εκτέλεση κάποιας εφαρμογής, παρουσιάζει ωστόσο αρκετούς περιορισμούς. Ο χρήστης δεν έχει τη δυνατότητα για ανάπτυξη προγράμματος αφού δεν περιέχεται μεταγλωττιστής στο σύστημα. Αντ' αυτού πρέπει να πραγματοποιηθεί η διαδικασία του cross-compile από κάποιον host computer και η μεταφορά του μετά στο σύστημα για εκτέλεση. Η διαδικασία του cross-compile μπορεί να αποδειχθεί μακρά και επίπονη ειδικά αν συμπεριληφθεί το γεγονός ότι οι εκδόσεις για τις βιβλιοθήκες που προσφέρει το Petalinux Tools μπορεί να έχουν καταργηθεί ή να εί-

ναι δύσκολο να βρεθούν για την πραγματοποίηση του cross-compile σε εξωτερικό host. Για τους παραπάνω λόγους και για εξοικονόμηση χρόνου αποφασίστηκε η χρήση μιας minimal έκδοσης των Ubuntu 16.04 για ενσωματωμένα συστήματα που χρησιμοποιούν επεξεργαστή ARM. Η minimal έκδοση πρόκειται για εικόνες Linux με ένα πολύ μειωμένο προεπιλεγμένο σύνολο πακέτων. Είναι πολύ μικρότερη από μια κανονική έκδοση Ubuntu, εκκινεί γρηγορότερα και απαιτεί λιγότερες ενημερώσεις ασφαλείας με την πάροδο του χρόνου, καθώς έχουν αρκετά λιγότερα εγκατεστημένα πακέτα.[79]

5 Πειραματική αξιολόγηση συστήματος

Για την αξιολόγηση λειτουργίας του συστήματος δημιουργήθηκαν 2 εφαρμογές σε περιβάλλον Linux. Η πρώτη εφαρμογή σε γλώσσα C++, χρησιμοποιεί τη βιβλιοθήκη λειτουργιών προγραμματισμού OpenCV και παρουσιάζει την εφαρμογή του φίλτρου Sobel σε hardware, πάνω σε καρέ (frames) που λαμβάνονται από μια USB webcam. Η δεύτερη εφαρμογή σε γλώσσα C, αποτελεί επίδειξη της σύλληψης και εμφάνισης καρέ (frames) από 2 USB κάμερες ταυτόχρονα, κάνοντας χρήση του framework Video4Linux2 (V4L2). Στο [31] αναφέρονται αρκετές από τις κάμερες που είναι συμβατές με τη προδιαγραφή UVC. Οι 2 κάμερες που χρησιμοποιήθηκαν για της υλοποίηση των παρακάτω εφαρμογών είναι οι Logitech HD Webcam C270 και Microsoft LifeCam VX-5000. Στον πίνακα 7 παρουσιάζονται τα χαρακτηριστικά των δύο UVC συμβατών webcams.

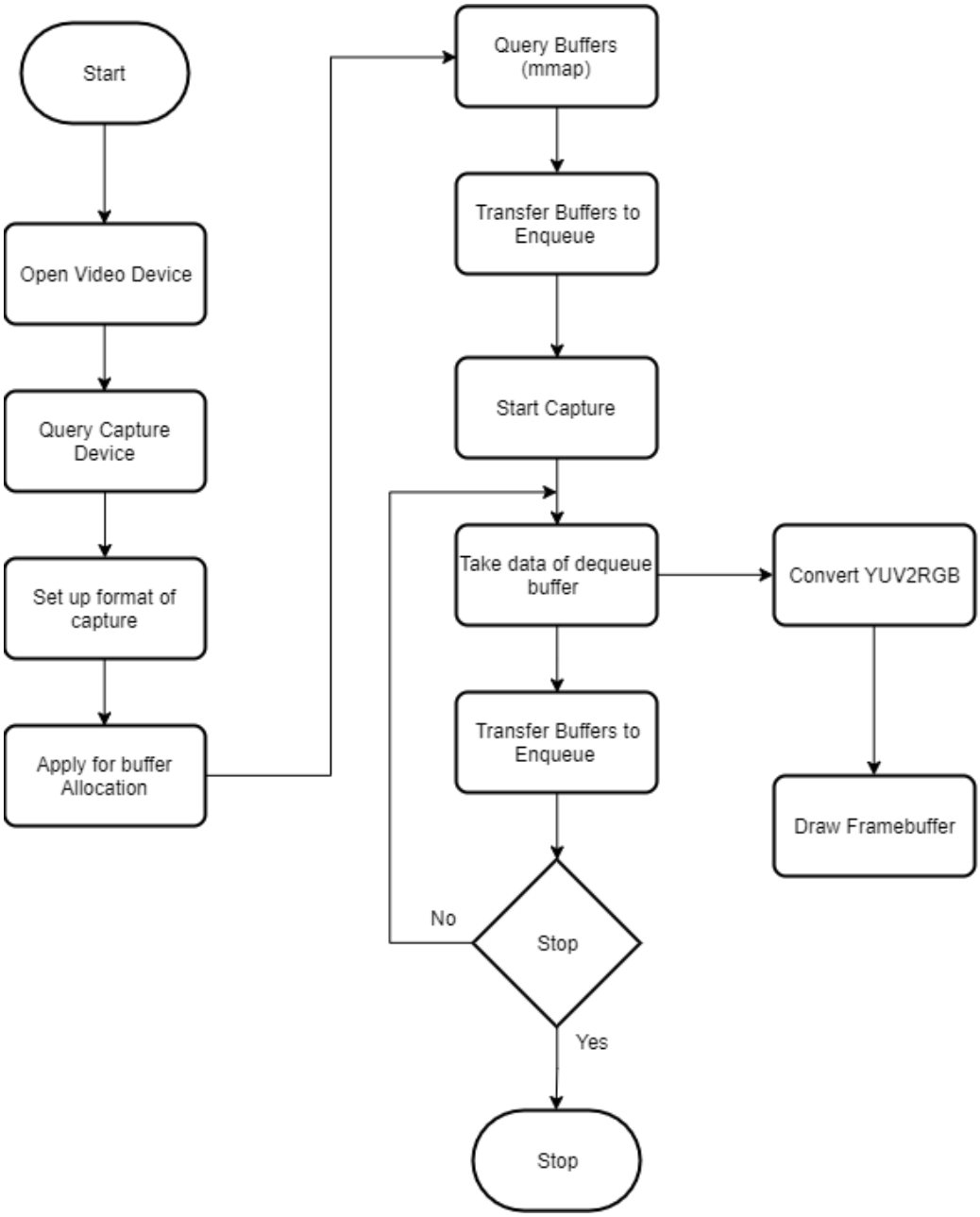
Table 7: USB webcams

	Pixel Format	*Max Resolution*	*Max Frame Per Second*
Logitech C270	YUYV(Raw), MJPG(compressed)	1280X960	30
Microsoft VX-5000	YUYV(Raw), MJPG(compressed)	640X480	30

5.1 Υλοποίηση διεπαφής δύο καμερών

Στο κεφάλαιο 4.4.2.2 έγινε ανάλυση της προσθήκης της κλάσης βίντεο USB (UVC) στο σύστημα, προκειμένου να αναγνωρίζονται USB κάμερες από αυτό. Με αυτόν τον τρόπο, οι κάμερες USB μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σύστημα που εφαρμόζει υποστήριξη για συσκευές συμβατές με τη προδιαγραφή UVC. Η υποστήριξη του UVC απαιτεί το V4L2, το οποίο προστίθενται αυτόματα με την ενεργοποίηση του UVC. Πρόκειται για ένα framework που δεν ελέγχει τις συσκευές απευθείας, αντ' αυτού, παρέχει ένα αφηρημένο μοντέλο της κλάσης συσκευών βίντεο για χρήση σε εφαρμογές. Έτσι, το V4L2 είναι ένα πρόγραμμα οδήγησης υψηλού επιπέδου που οδηγεί το πρόγραμμα οδήγησης UVC, το οποίο οδηγεί το πρόγραμμα οδήγησης USB. Οι δύο USB webcams συνδέονται στη θύρα USB OTG του Zedboard μέσω ενός USB hub. Η χρήση δύο καμερών σε έναν ελεγκτή προκαλεί προβλήματα bandwidth όταν χρησιμοποιούνται αναλύσεις άνω των 352 x 288.

Το παρακάτω διάγραμμα αντιπροσωπεύει την ακολουθία για τη λήψη frame από μια κάμερα και τη προβολή του:



Εικόνα 78: V4l2 Application Flowchart

Open Device: Ένας περιγραφέας αρχείων (file descriptor) χρησιμοποιείται για να ανοίξει η USB συσκευή βίντεο. Σε αυτήν την περίπτωση, χρησιμοποιείται μια συμβολοσειρά που αντιπροσωπεύει τη συσκευή. Η συμβολοσειρά περιέχει το κατάλληλο όνομα συσκευής στο /dev directory του Linux FileSystem. Για παράδειγμα, η πρώτη κάμερα που συνδέθηκε αναγνωρίζεται "*/dev/video0*", η δεύτερη ως "*/dev/video1*" κοκ.

Query Capture: Το V4L2 επιτρέπει την ανάκτηση και ανάλυση των δυνατοτήτων της συσκευής, δηλαδή το σύνολο των λειτουργιών που υποστηρίζει. Κάμερες που δεν υποστηρίζουν το V4L2 πρέπει να ελεγχθούν για την αποφυγή σφαλμάτων.

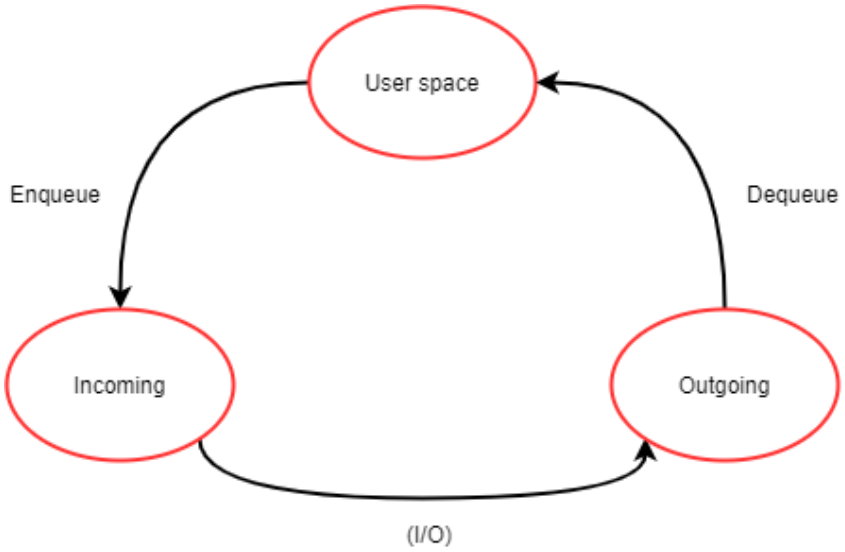
Format of Capture: Το V4L2 παρέχει μια εύκολη διεπαφή για τον έλεγχο των μορφών εικόνας και του χώρου χρωμάτων που υποστηρίζει και παρέχει η κάμερα web. Εδώ επιλέγεται το μέγεθος του καρέ,η μορφή (MJPEG, RGB, YUV) και ούτω καθεξής. Η συσκευή πρέπει να μπορεί να χειριστεί τη μορφή που θα επιλεχθεί. Στη συγκεκριμένη εφαρμογή η μορφή εικονοστοιχείων που επιλέχθηκε ως είσοδος είναι η YUV, την οποία υποστηρίζουν και οι δύο κάμερες, ενώ το μέγεθος καρέ ορίστηκε στα 320x240, για τη σταθερή λειτουργία της ταυτόχρονης χρήσης των δύο καμερών.

Buffer Allocation: Η συσκευή βίντεο ενημερώνεται για το αριθμό των buffers που θα έχει διαθέσιμους για την αποθήκευση των frame. Τέθηκαν 4 buffers για τη κάθε συσκευή αφού λιγότεροι από δύο δεν αρκούν ενώ πάνω από 5 εξαντλούν τη μνήμη. Επιπλέον η συσκευή ενημερώνεται για τη χρήση της μεθόδου memory map για ροή I/O.

Query the buffers: Για κάθε buffer που έχει εκχωρηθεί, πρέπει να οριστούν τα κατάλληλα χαρακτηριστικά και να πραγματοποιηθεί μια νέα αντιστοίχιση μνήμης (memory map). Η συνάρτηση mmap() καλείται ώστε να χαρτογραφήσει την εικονική μνήμη που θα μοιραστεί μεταξύ user-space και της συσκευής.

Enqueue Buffers: Οι buffers τοποθετούνται στην εισερχόμενη ουρά του προγράμματος οδήγησης από την εφαρμογή. Για τη συσκευή USB webcam, οι buffers στην εισερχόμενη ουρά είναι κενοί, περιμένοντας από το πρόγραμμα οδήγησης να τους γεμίσει με δεδομένα βίντεο.

Start Capture: Η συσκευή τίθεται σε λειτουργία streaming I/O, όπου αφού εξέλθουν όλοι οι buffers εισέρχονται καινούριοι στις ουρές του οδηγού επανειλημμένα. Κάθε φορά που εξέρχονται οι buffers διαβάζεται και καρέ. Πρόκειται για την διαδικασία μεταφοράς των frames στο user-space για αξιοποίηση.



Εικόνα 79: Capture State Machine

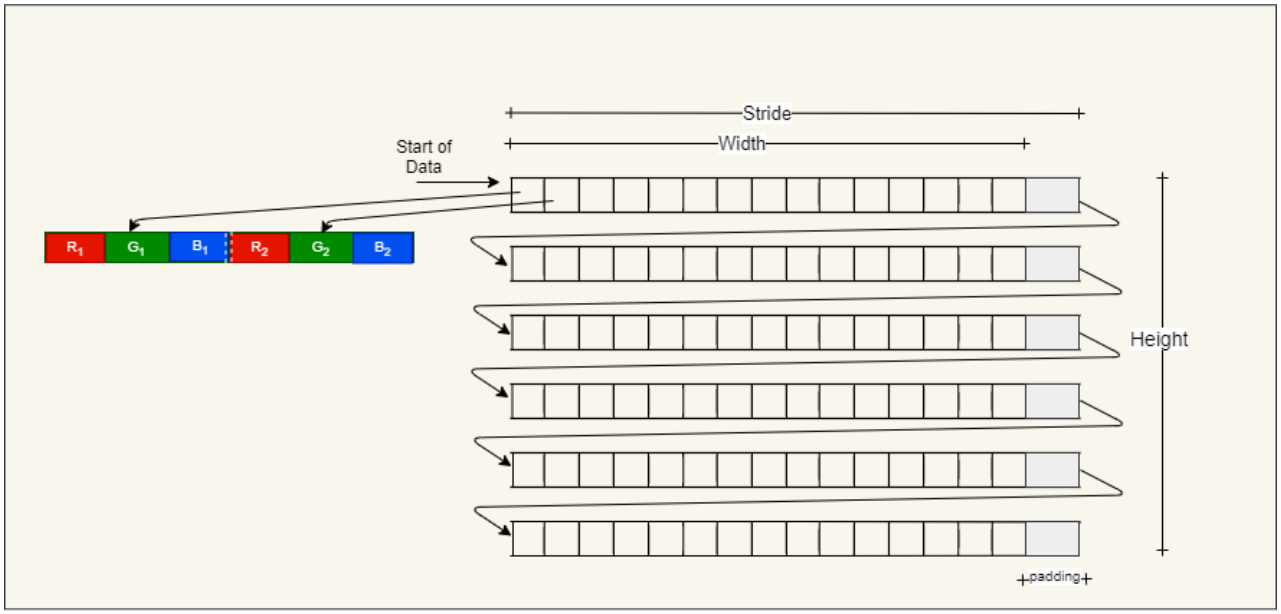
Dequeue Buffers: Στην εξερχόμενη ουρά του οδηγού, οι buffer έχουν υποβληθεί σε επεξεργασία από το πρόγραμμα οδήγησης και περιμένουν την εφαρμογή για να τους διεκδικήσει. Οι εξερχόμενοι buffers περιέχουν νέα δεδομένα καρέ (frames) για διάβασμα.

Convert YUV to BGR: Τα δεδομένα κάθε frame μεταφέρονται στο χρωματικό χώρο RGB, για να μπορούν να απεικονιστούν από τη περιοχή της μνήμης της συσκευής εξόδου (framebuffer). Με την παρακάτω formula μετατρέπεται η μορφή των εικονοστοιχείων των καμερών από YUV σε RGB:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}; \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

Εικόνα 80: Colorspace Conversion YUV and RGB for HDTV (BT.601)

Draw Framebuffer: Τα βήματα που περιγράφονται παραπάνω εφαρμόζονται δύο φορές, από μια για κάθε κάμερα. Δύο μονοδιάστατοι πίνακες μεγέθους "*width x height x 3(κανάλια RGB)*" περιέχουν τα frames για κάθε κάμερα. Στην Εικόνα 82 φαίνεται η διάταξη των εικονοστοιχείων στην μνήμη με κάθε εικονοστοιχείο να περιέχει τρεις διαφορετικές τιμές, μία για κάθε κανάλι. Ιδιαίτερο ενδιαφέρον παρουσιάζει ο όρος line stride που φαίνεται στο σχήμα. Πρόκειται για τον αριθμό των byte που πρέπει να προστεθούν στη διεύθυνση του πρώτου pixel μιας γραμμής για να μεταβεί στη διεύθυνση του πρώτου pixel της επόμενης γραμμής. Είναι σημαντικό να σημειωθεί ότι το πλάτος (width) μιας εικόνας μετράται σε εικονοστοιχεία και περιγράφει την ίδια την εικόνα (και δεν εξαρτάται από τον τρόπο αποθήκευσης μιας εικόνας στη μνήμη). Αντίθετα, το line stride εξαρτάται από το πώς μια εικόνα αντιπροσωπεύεται στη μνήμη και μετριέται σε byte . Στη συνέχεια **σ**ύμφωνα με τα χαρακτηριστικά της συσκευής εξόδου (resolution, bits per pixel), επιλέγεται η τοποθεσία όπου θα σχεδιαστούν τα 3 κανάλια (RGB) του κάθε frame στην οθόνη.



Εικόνα 81: Απεικόνιση εικονοστοιχείων στη μνήμη

Εκτέλεση Εφαρμογής

Στην παρακάτω εικόνα παρουσιάζεται η εκτέλεση της εφαρμογής για διεπαφή δύο USB webcams και η απεικόνιση τους μέσω HDMI. Η αριστερή κάμερα (Logitech c270) και η δεξιά κάμερα (Microsoft VX-5000), συνδέονται μέσω ενός USB hub στο Zedboard. Η αριστερή οθόνη είναι συνδεδεμένη σε ηλεκτρονικό υπολογιστή αναπαράγοντας ένα βίντεο ενώ η δεξιά είναι συνδεδεμένη μέσω HDMI με το Zedboard και απεικονίζει τα καρέ που λαμβάνονται από τις webcams.

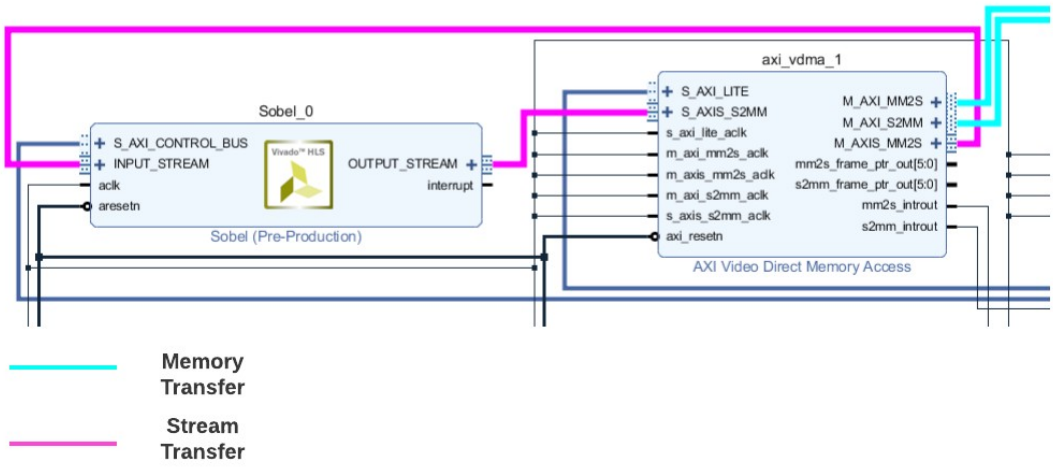
Εικόνα 82: Καρέ από την εκτέλεση της εφαρμογής 2 USB webcams: Resolution = 320x240

5.2 Υλοποίηση φίλτρου Sobel

Η χρήση της βιβλιοθήκης OpenCV χαρίζει ένα υψηλότερο επίπεδο αφαιρετικότητας στην υλοποίηση αυτής της εφαρμογής. Χρησιμοποιεί κατά κόρον την κλάση Mat η οποία χαρακτηρίζεται από μια κεφαλίδα μήτρας (που περιέχει πληροφορίες όπως το μέγεθος της μήτρας, η μέθοδος που χρησιμοποιείται για την αποθήκευση, σε ποια διεύθυνση αποθηκεύεται η μήτρα και ούτω καθεξής) και ένα δείκτη στον πίνακα που περιέχει τις τιμές των εικονοστοιχείων. Μέσω του Mat ο προγραμματιστής αποκτά το πλεονέκτημα της εκχώρησης και απελευθέρωσης μνήμης με αυτόματο τρόπο. Κάθε στιγμή δηλαδή, χρησιμοποιείται μόνο η μνήμη που χρειάζεται για την εκτέλεσης μιας εργασίας. Μέσω της κλάσης VideoCapture() δημιουργείται ένα αντικείμενο λήψης βίντεο, από τη USB webcam, ώστε οι επιθυμητές λειτουργίες να μπορούν να εκτελεστούν σε αυτό το βίντεο. Τα δεδομένα που λαμβάνονται μετατρέπονται αυτόματα από το API λήψης βίντεο σε BGR (Blue-Green-Red) μορφή και καταχωρούνται σε ένα Mat αντικείμενο ως 8-bit unsigned integer των τριών καναλιών.

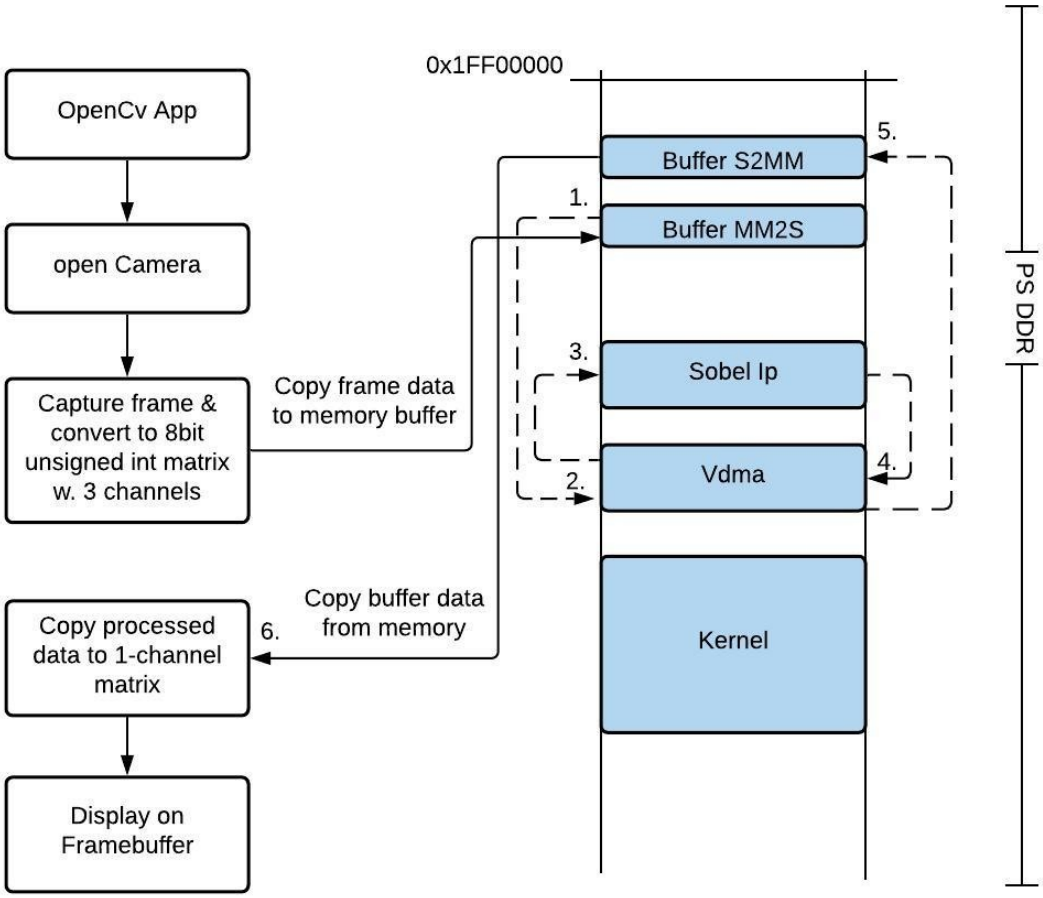
Τα δεδομένα του αντικειμένου Mat αποτελούν στην ουσία το frame που λαμβάνεται από την web-cam.

Σε αυτό το σημείο αξίζει να σημειωθεί τρόπος που θα αντιμετωπιστούν τα λογικά blocks που αφορούν το φίλτρο Sobel (VDMA και Sobel IPs). Σε αντίθεση με την λογική που αφορά την μεταφορά των frames από την μνήμη και την HDMI έξοδο της, για το φίλτρο Sobel δεν χρησιμοποιήθηκαν Linux drivers. Αντ' αυτού η εκχώρηση των κατάλληλων διευθύνσεων στη μνήμη και η διαμόρφωση των καταχωρητών γίνεται χειροκίνητα μέσα από αυτό το πρόγραμμα. Ο τρόπος αυτός παρομοιάζει τον τρόπο που θα αναπτυσσόταν η εφαρμογή αν αφορούσε no-OS περιβάλλον. Στο κεφάλαιο 4.4.2.2 αναλύεται πως αποφεύγεται η χρήση συγκεκριμένης περιοχής μνήμης από τον πυρήνα Linux. Σε αυτή τη περιοχή εκχωρούνται διευθύνσεις για 2 frame buffers. Στην περιοχή μνήμης του πρώτου frame buffer, αντιγράφονται τα δεδομένα του Mat αντικειμένου (ένα frame). Το VDMA engine διαβάζει τα δεδομένα από την μνήμη που βρίσκεται ο πρώτος frame buffer και μέσα από το κανάλι M_AXIS_MM2S τα μεταφέρει σε μορφή AXI-Stream στο Sobel IP block. Τα δεδομένα επιστρέφονται στο VDMA engine και μέσω του καναλιού M_AXI_S2MM γράφονται στην περιοχή μνήμης που έχει δεσμευτεί για τον δεύτερο frame buffer.



Εικόνα 83: Sobel and VDMA interaction

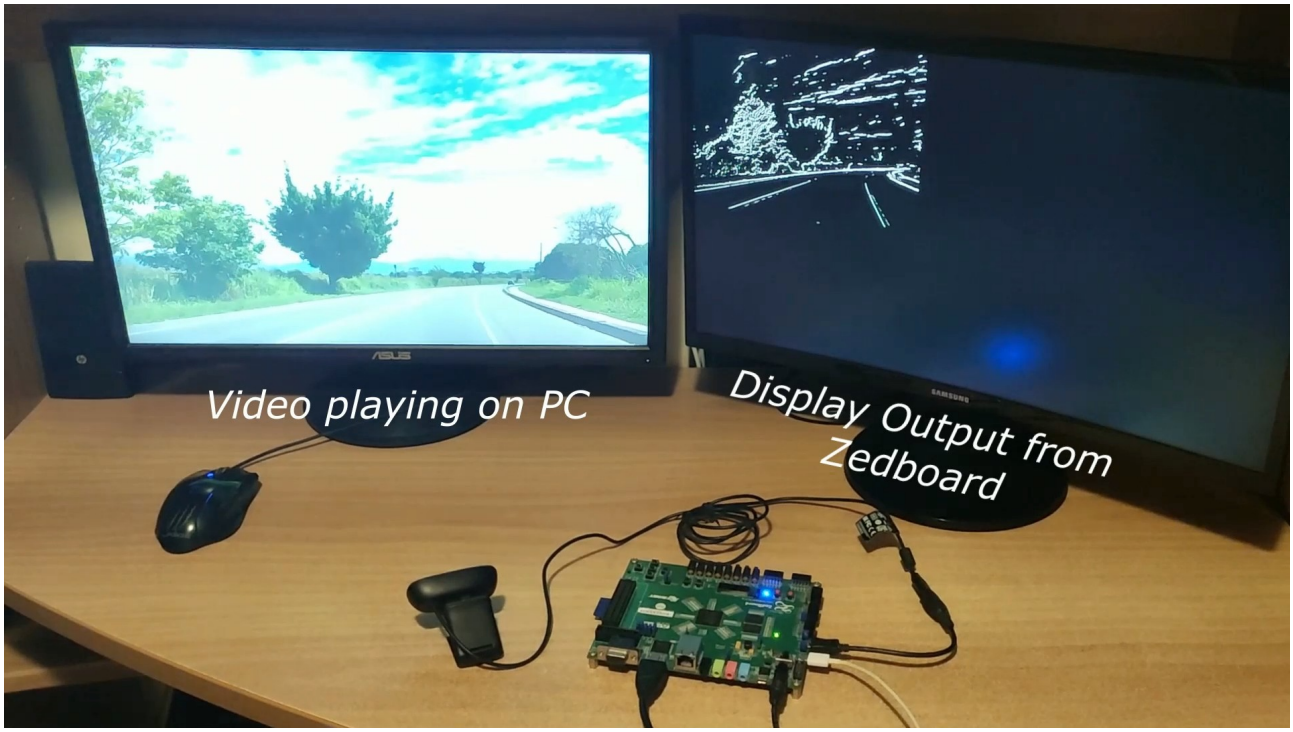
Από εκεί τα δεδομένα αντιγράφονται σε ένα νέο αντικείμενο Mat και τελικά σχεδιάζονται στον framebuffer που αντιστοιχεί στην έξοδο HDMI, όπως στο βήμα *Draw Framebuffer* της προηγούμενης εφαρμογής.



Εικόνα 84: OpenCV-Sobel Application

Εκτέλεση Εφαρμογής

Στην παρακάτω εικόνα παρουσιάζεται η εκτέλεση της εφαρμογής φίλτρου Sobel στα καρέ μιας USB webcam και η απεικόνιση τους μέσω HDMI. Η αριστερή οθόνη είναι συνδεδεμένη σε ηλεκτρονικό υπολογιστή αναπαράγοντας ένα βίντεο ενώ η δεξιά είναι συνδεδεμένη μέσω HDMI με το Zedboard και απεικονίζει τα καρέ που λαμβάνονται από τις webcams αφού πρώτα έχουν υποστεί φιλτράρισμα Sobel.



Εικόνα 85: Καρέ από την εκτέλεση της εφαρμογής φίλτρου Sobel: Resolution = 640x480

6 Συμπεράσματα και μελλοντική εργασία

Το τελευταίο κεφάλαιο αυτής της διπλωματικής συνοψίζει τα αποτελέσματα που εξήχθησαν από τη πειραματική αξιολόγηση του συστήματος και τις μελλοντικές κατευθύνσεις και βελτιστοποιήσεις που μπορούν να εφαρμοστούν.

6.1 Συμπεράσματα

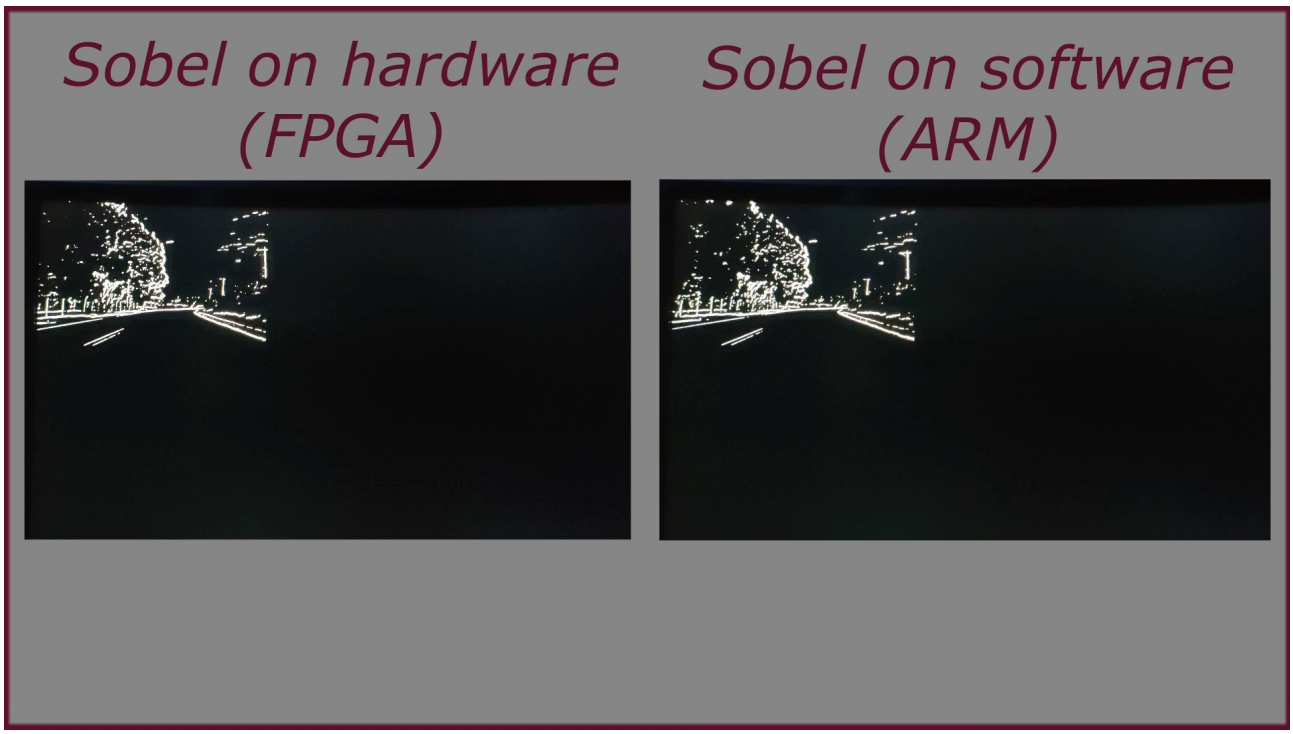
Με την ολοένα και αυξανόμενη κυκλοφορία USB webcams που ενσωματώνουν αισθητήρες εικόνας που φτάνουν τα 15 Megapixels/60 fps, είναι θέμα μερικών χρόνων τέτοιου είδους χαρακτηριστικά να αποτελέσουν την νόρμα όσον αφορά αυτές τις USB συσκευές. Η μελλοντική "υπόσχεση" για την βελτιστοποίηση αυτών των καμερών σε συνδυασμό με το πολύ χαμηλό κόστος, τις καθιστούν μια πολύ ελκυστική επιλογή για εκμετάλλευσή τους από συστήματα FPGA. Αυτό αποδεικνύεται ακόμα και από την αξιοποίηση μιας κάμερα 10ετίας (Microsoft VX-5000) που όμως αποδίδει αξιοπρεπή αποτελέσματα.

Τα αποτελέσματα που εξάγονται από την εφαρμογή της παράλληλης χρήσης 2 USB webcams στο κεφ[5.1], αφορούν τον περιορισμό που τίθεται από τη προδιαγραφή πρωτοκόλλου USB 2.0. Πιο συγκεκριμένα, το θεωρητικό bandwidth:

$$Bandwidth = Resolution \times Fps \times bit\ depth \quad ,$$

που υποστηρίζει το USB 2.0, είναι τα 480 Mbps ή ~60 MBps. Ωστόσο η μεταφορά δεδομένων από USB webcam περιορίζεται, βάσει πρωτοκόλλου, στο πρακτικό bandwidth που αναλογεί κοντά στο 80% του θεωρητικού (48 MBps). Επιπλέον, αν ληφθεί υπόψη η χρήση του USB hub, το bandwidth που απομένει θα πρέπει να διαμοιραστεί στις δύο κάμερες παραχωρώντας περίπου, 24 MBps στη καθεμία. Ο περιορισμός του θεωρητικού bandwidth επιβεβαιώνεται στην πράξη αφού έπειτα από δοκιμές παρατηρήθηκε ότι για αναλύσεις μεγαλύτερες των 640x480 ο USB ελεγκτής εξαντλείται και η εφαρμογή δεν εκτελείται. Από την άλλη για αναλύσεις μικρότερες των 320x240 η εφαρμογή εκτελείται κανονικά προβάλλοντας βίντεο από δύο κάμερες κοντά ή ίσα (ανάλογα με το φωτισμό στο χώρο) με το μέγιστο fps που υποστηρίζουν, δηλαδή ~30 fps.

Τα αποτελέσματα που εξάγονται από την εφαρμογή του φίλτρου Sobel σε hardware, μέσω της FPGA, και σε software, μέσω του ARM, είναι τα αναμενόμενα. Η παράλληλη φύση της FPGA προσφέρει πλεονεκτήματα και καλύτερους χρόνους επεξεργασίας βίντεο, όπως φαίνεται και στον πίνακα 8. Σε μικρές αναλύσεις τα αποτελέσματα είναι πανομοιότυπα και για τις δύο υλοποιήσεις (sw και hw), φτάνοντας το μέγιστο ρυθμό καρέ ανά δευτερόλεπτο που υποστηρίζει η κάθε κάμερα (30 fps) . Ωστόσο όσο η ανάλυση αυξάνεται, τόσο πιο αισθητή γίνεται η ανωτερότητα της FPGA στη συγκεκριμένη εργασία. Αν η επεξεργασία που υφίσταται το βίντεο ήταν περισσότερο απαιτητική, τότε η διαφορά ανάμεσα στις δύο προσεγγίσεις θα ήταν ακόμα μεγαλύτερη. Στην Εικόνα 86, έχει παρθεί ένα καρέ κατά την εκτέλεση φιλτραρίσματος στην FPGA και ένα κατά την εκτέλεση φιλτραρίσματος στον ARM και τοποθετήθηκαν το ένα δίπλα στο άλλο για σύγκριση των δύο υλοποιήσεων.



Εικόνα 86: Καρέ από την εφαρμογή φίλτρου Sobel στην FPGA και στον επεξεργαστή ARM

Στον πίνακα που ακολουθεί παρουσιάζεται ο χρόνος που απαιτείται για την απόκτηση ενός καρέ βίντεο από την USB webcam, η εφαρμογή του φίλτρου sobel σε αυτό και τελικά η προβολή του στην οθόνη μέσω HDMI.

Table 8: Αποτελέσματα Φίλτρου Sobel

	Sobel στην FPGA		*Sobel στον ARM*	
Resolution (width x height)	Καρέ	Χρόνος (s)	Καρέ	Χρόνος (s)
320 x 240	150	5	150	5,3
432 x 240	150	5	150	6,8
544 x 288	150	5,7	150	10
640 x 360	150	8,3	150	13,6
640 x 480	150	10,7	150	18,75
800 x 600	150	16,6	150	30
1280 x 960	150	37,5	150	75

6.2 Μελλοντικές Επεκτάσεις

Το σύστημα που υλοποιήθηκε προσφέρει αρκετές δυνατότητες για μελλοντική εργασία, καθώς σκοπός ήταν η λήψη της τεχνογνωσίας που θα οδηγήσει στην ανάπτυξη μιας πλατφόρμας ικανή να

λειτουργήσει ως σκελετός για μελλοντική εκμετάλλευση των USB webcams σε συσκευές FPGA. Οι εφαρμογές και κατ' επέκταση οι σχεδιαστικές επιλογές που έγιναν λειτουργούν με ικανοποιητικά αποτελέσματα όταν το μέγεθος καρέ που χρησιμοποιείται είναι μικρό. Πιο συγκεκριμένα:

Για την περίπτωση της διεπαφής 2 καμερών η λειτουργία διατηρείται ομαλή όταν ορίζεται ανάλυση 320x240 και για τις δύο κάμερες. Ο περιορισμός εδώ εισάγεται λόγω του bandwidth του πρωτοκόλλου και μπορεί να ξεπεραστεί:

- Κάνοντας χρήση δύο usb host controllers, έναν για κάθε webcam.
- Αναπτύσσοντας/προσαρμόζοντας τους USB Linux drivers, ώστε να παρακάμπτεται ο περιορισμός του πρακτικού bandwidth ή οι drivers να προσαρμόζονται καλύτερα στην εκάστοτε USB webcam

Για την περίπτωση της διεπαφής μιας USB webcam, η εφαρμογή[5.2] προσφέρει ικανοποιητικά αποτελέσματα αλλά προτείνεται να λειτουργεί σε αναλύσεις μικρότερες των 800x600. Ωστόσο υπάρχει δυνατότητα να αξιοποιηθεί το σύστημα και για μεγαλύτερες αναλύσεις. Όπως για παράδειγμα κάνοντας χρήση του framework GStreamer, το οποίο δέχεται, σε ένα υψηλότερο επίπεδο αφαιρετικότητας, ως source(πηγή) την κάμερα και ως sink το HDMI και επιβάλλοντας τις κατάλληλες μετατροπές στο βίντεο, πραγματοποιεί απεικόνιση βίντεο με HD αναλύσεις σε πολύ υψηλά frames (1280x960 σε ~30 fps για την c270 κάμερα). Επομένως, σε επίπεδο Software Linux, δύναται να επιτευχθεί πολύ υψηλός ρυθμός καρέ στις μέγιστες αναλύσεις αν αναπτυχθεί κατάλληλη λογική.

Από την άλλη οι απαιτήσεις για το εύρος ζώνης της μνήμης στο σύστημα HW (υλοποίηση HDMI και Sobel) εξαρτώνται από το χρονισμό των εικονοστοιχείων και από τον αριθμό των bytes ανά pixel. Στο σύστημα χρησιμοποιούνται 4 bytes ανά pixel, οπότε για ανάλυση 1080p με ρολόι στα 148MHz απαιτείται ρυθμός στα ~600MB/s. Το σύστημα HW διαθέτει δύο "διαδρομές" διασύνδεσης (High Performance Interconnects) για την επικοινωνία μεταξύ μνήμης και PL. Η πρώτη "διαδρομή" γίνεται μεταξύ του υποσυστήματος Sobel και της μνήμης ενώ η δεύτερη μεταξύ του υποσυστήματος HDMI και της μνήμης. Τα video DMA που μεταφέρουν δεδομένα μεταξύ των διασυνδέσεων χρησιμοποιούν ρολόι 100 MHz και πλάτος δεδομένων 64-bit, ώστε τελικά να επιτυγχάνεται μέγιστος ρυθμός δεδομένων τα 800 MB/s, που είναι αρκετός για να υποστηρίξει τις απαιτήσεις για 1080p αναλύσεις.

Επομένως ενώ το σύστημα μπορεί να υποστηρίξει HD αναλύσεις σε υψηλά fps, οι εφαρμογές που δημιουργήθηκαν δεν είναι σε θέση να τις αξιοποιήσουν. Προτάσεις για βελτιστοποιήσεις και προσθήκες ώστε να επιτευχθούν καλύτερα αποτελέσματα είναι οι εξής:

- Διατηρώντας την αφαιρετικότητα σε χαμηλά επίπεδα, η συσκευή framebuffer (fbdev) που χρησιμοποιείται για την απεικόνιση του βίντεο, θα μπορούσε να αντικατασταθεί με τους “dumb buffers” του DRM/KMS [3.2.3.2]. Πρόκειται στην ουσία για framebuffer που παρέχει αρκετές παραπάνω δυνατότητες εκμετάλλευσης υλικού από τον χρονολογικά παλαιότερο fbdev, που χρησιμοποιείται σε αυτή την εργασία.
- Στην εφαρμογή που πραγματοποιείται το φίλτρο sobel, χρησιμοποιείται η κλάση της βιβλιοθήκης OpenCV, VideoCapture(). Δεν χαρακτηρίζεται για το πόσο γρήγορη είναι, αλλά για

την αξιοπιστία της και γιαυτό το λόγο επιλέχθηκε. Για τη βελτίωση των fps μπορεί να αντικατασταθεί με έναν πιο αποδοτικό τρόπο λήψης και μορφοποίησης καρέ από μια webcam. Τέτοια παραδείγματα είναι το framework GStreamer.

- Ανάπτυξη κάποιου αλγορίθμου για την επεξεργασία του δεδομένων βίντεο εισόδου, χωρίς τη χρήση έτοιμων συναρτήσεων OpenCV. Η δημιουργία εξειδικευμένων/προσαρμοσμένων συναρτήσεων από τον χρήστη, αφήνει ανοιχτό το ενδεχόμενο για επίτευξη καλύτερης εκμετάλλευσης των πόρων της FPGA, μεγαλύτερη ακρίβεια των αποτελεσμάτων και ευελιξία στον τρόπο χρήσης.
- Εκμετάλλευση character device Linux drivers για χρήση του βίντεο pipeline που αφορά το φίλτρο, από το userspace . Η χειροκίνητη δέσμευση / από-δέσμευση μνήμης από τον χρήστη, γι' αυτά τα block λογικής, είναι πιθανό να προκαλέσει προβλήματα ασφαλείας που είναι δύσκολο να επιλυθούν και συχνά οδηγούν τον πυρήνα στο να κολλήσει.
- Ανάπτυξη του συστήματος υλικού ώστε να μπορεί να υποστηρίξει προβολή βίντεο σε αναπτυξιακές πλακέτες που δεν διαθέτουν εξωτερικό HDMI transmitter, διευρύνοντας περισσότερο τις δυνατότητες του συστήματος. Επιπλέον, έρευνα για την ενσωμάτωση προγραμμάτων οδήγησης για τη προδιαγραφή πρωτοκόλλου USB 3.0 που διαθέτουν αρκετές από τις MPSoC συσκευές. Η εκμετάλλευση του USB 3.0 θα οδηγήσει σε καλύτερη απόδοση κατά τη μεταφορά δεδομένων βίντεο.
- Περαιτέρω μελέτη όσον αφορά τη μετατροπή των RAW YUYV δεδομένων σε RGB δεδομένα 5.1, διότι σε ορισμένες περιπτώσεις παρατηρήθηκε λανθασμένη απεικόνιση μπλε εικονοστοιχείων, σε αντικείμενα που βρίσκονται στο βάθος του βίντεο. Η αποδέσμευση του προγραμματιστή από αυτήν την υλοποίηση, μπορεί να πραγματοποιηθεί σε υψηλότερο επίπεδο αφαιρετικότητας χρησιμοποιώντας το λογισμικό FFmpeg για αυτοματοποιημένο και βέλτιστο χειρισμό βίντεο.
- Χρήση της μνήμης BRAM που βρίσκεται στην FPGA για την προσωρινή αποθήκευση δεδομένων ώστε να γίνει εκμετάλλευση του παραλληλισμού που προσφέρει μια τέτοια συσκευή.
- Χρήση των ACP ports αντί των HP ports, επειδή είναι πιο γρήγορες και μπορούν να εκμεταλλευτούν τις cache που υπάρχουν στο APU. Με αυτόν τον τρόπο θα βελτιωθεί, εν δυνάμει, η επικοινωνία μεταξύ του επεξεργαστή ARM, της μνήμης DDR3 και της FPGA.

7 Βιβλιογραφία

1: Xilinx, Getting Started Guide: ZC702 Evaluation Kit and Video and Imaging Kit, UG926 (v2012).

2: Γ. Βάσιλας, "Επισκόπηση αλγορίθμων τοποθέτησης σε Field Programmable Arrays (FPGAs)." Master's thesis, Πανεπιστήμιο Πειραιώς, (2015).

3: B. Ramesh, A. D. George and H. Lam, "Real-time, low-latency image processing with high throughput on a multi-core SoC," In 2016 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-7, (2016).

4: H.R. Amir, A. J. Taberner, M. P. Nash and P. MF. Nielsen. "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms." Signal Processing: Image Communication 68, pp. 101-119, (2018).

5: J. Diaz, E. Ros, A. Prieto and F. J. Pelayo. "Fine grain pipeline systems for real-time motion and stereo-vision computation." International Journal of High Performance Systems Architecture 1, no. 1, pp. 60-68, (2007).

6: Xilinx, SoC Portofolio, URL: https://www.xilinx.com/products/silicon-devices/soc.html.

7: Xilinx, Zynq Architecture, URL: http://www.ioe.nchu.edu.tw/Pic/CourseItem/4468_20_Zynq_Architecture.pdf.

8: L. H. Crockett, R. A. Elliot, M. A. Enderwitz and R. W. Stewart. "The zynq book tutorials." Department of Electronic and Electrical Engineering University of Strathclyde Glasgow, Scotland, UK (2014).

9: Inline-CoProcessing FPGA Interface, URL: https://www.vision-systems.com/embedded/article/16737656/cpu-or-fpga-for-image-processing-which-is-best.

10: E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan and M. W. Naouar, "FPGAs in Industrial Control Applications," in IEEE Transactions on Industrial Informatics, vol. 7, no. 2, pp. 224-243, (2011).

11: GigE Vision, Camera Interface, URL: https://www.visiononline.org/vision-standards-details.cfm?type=5.

12: USB3 Vision, Camera Interface, URL: https://www.visiononline.org/vision-standards-details.cfm?id=167&type=11.

13: USB3 Camera, URL: https://automation.omron.com/en/us/products/family/stu3v.

14: USB Connector Types, URL: https://getprostorage.com/blog/usb-c-thunderbolt-3-rundown/.

15: USB System Architecture, URL: http://www.usblyzer.com/usb-internals.htm .

16: Linux USB Subsystem, URL: https://sumeetjainengineer.wordpress.com/2014/05/04/insight-into-the-linux-usb-subsystem-architecture-and-device/.

17: Makeup of USB Cable, URL: http://www.usbmadesimple.co.uk/ums_2.htm.

18: X. Li, B. Yang, "Powering your FPGA Applications." Industrial Analog & Power Group, Renesas Electronics Corp, (2018).

19: USB, ULPI PHY, URL: https://www.eetimes.com/partitioning-hi-speed-usb-systems/.

20: Xilinx, Wiki Page, URL: https://xilinx-wiki.atlassian.net/wiki/spaces/A/overview.

21: M. Bergeron, S. Elzinga, G. Szedo, G. Jewett and T. Hill, “1080p60 Camera Image Processing.” Xilinx Reference Design, XAPP794 (2013)

56: FMC Imageon, URL: https://www.sundance.technology/wp-content/uploads/2016/08/GS-AES-FMC-IMAGEON-G-14.2-v1.1c.pdf.

22: Digilent, Embedded Vision Bundle, URL: https://store.digilentinc.com/embedded-vision-bundle/.

23: J. SriVarshini and V. NarasimhaNayak, "Implementation of video-processing and control on a zynq soc platform." Int. J. Eng. Technol. 8.1, pp. 274-279 (2016): .

24: CMOS Camera OV7670: https://www.hellasdigital.gr/electronics/sensors/cameras/vga-ov7670-cmos-camera-module-lens-640x480-sccb-w-i2c-interface-for-arduino/.

25: R. He and C. Zili, "Design of Image transmission and Display System Based on ZYNQ." 2018 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018). Atlantis Press, (2018).

26: IP Camera, URL: https://www.hikvision.com/en/products/IP-Products/Network-Cameras/Pro-Series-EasyIP-/ds-2cd2325fwd-i/.

27: USB Host Detailed View, URL: http://csiewiki.crboy.net/embedded/USB.

28: USB Direct to FPGA, URL: https://xess.com/projects/fpga-usb-v2-project/index.html.

29: Microchip, USB3300 PHY, URL: http://ww1.microchip.com/downloads/en/DeviceDoc/00001783C.pdf.

30: USB Enumeration, URL: https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_113_Simplified%20Description%20of%20USB%20Device%20Enumeration.pdf.

31: UVC Compatible Devices, URL: http://www.ideasonboard.org/uvc/#devices.

32: Cypress, Product Guide, "USB Controller", URL: https://www.cypress.com/products/ez-host.

33: A. Eshack and J. Kumar, "Design of a Data Acquisition System for USB Devices over Gigabit Ethernet." In International Conference on Emerging Technology Trends on Advanced Engineering Research. Proceedings published by International Journal of Computer Applications, (2012).

34: M. Cao and P. Yin. "Research of USB OTG Technology in Image High-Speed Data Transfer." In Advanced Materials Research, vol. 951, pp. 261-264, (2014).

35: Mouser, "ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller", v.05 (2008) .

38: Xilinx, Technical Reference Manual: Zynq-7000 All Programmable SoC, UG585 (v2018).

36: Digilent, Refference Manual: Zybo Z7 Board, Rev. B (2018).

37: Avnet, Hardware User's Guide: Zedboard, (v2014.2).

39: N. Stephen, L. Thomas and W. Devin. "Accelerating OpenCV Applications with Zynq-7000 All Programmable SoC using Vivado HLS Video Libraries." Xilinx Application Note, XAPP1167 (v2015).

40: Analog Devices, Product Info: ADV7511 225 MHz, High Performance HDMITransmitter with ARC, (2010).

41: Xilinx, User Guide: 7 Series FPGAs GTX/GTH Transceivers, UG476 (v2017.4).

42:Avnet, Board: "Zedboard SoC", URL: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf.

43: Xilinx, Board: ZC702 SoC, URL: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html.

44: Xilinx, Board: ZC706 SoC, URL: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html.

45: Xilinx, Board: AC701 FPGA-7, URL: https://www.xilinx.com/products/boards-and-kits/ek-a7-ac701-g.html.

46: Xilinx, Board: KC705 FPGA-7, URL: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html.

47: Xilinx, Board: VC707 FPGA-7, URL: https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html.

48: Xilinx, Board: Arty Z7 SoC, URL: https://www.xilinx.com/products/boards-and-kits/1-pdb0q2.html.

49: Xilinx, Board: Arty A7 FPGA, URL: https://www.xilinx.com/products/boards-and-kits/1-w51quh.html.

50: Xilinx, Board: Zybo Z7 SoC, URL: https://www.xilinx.com/products/boards-and-kits/1-pukio3.html.

51: Digilent, Board: Pynq SoC, URL: https://www.xilinx.com/products/boards-and-kits/1-hydd4z.html.

52: Xilinx, Board: ZCU102 MPSoC, URL: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html.

53: Xilinx, Board: ZCU104 MPSoC, URL: https://www.xilinx.com/products/boards-and-kits/zcu104.html.

54: Xilinx, Board: ZCU106 MPSoC, URL: https://www.xilinx.com/products/boards-and-kits/zcu106.html.

55: Xilinx, Product Guide: HDMI 1.4/2.0 Transmitter Subsystem, PG235 (v2017.4).

57: Xilinx Wiki, HDMI Framebuffer Example, URL: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/33128528/HDMI+FrameBuffer+Example+Design+2018.3.

58: LogicBricks, Data Sheet: logiCVC-ML Compact Multilayer Video Controller, v5.5 (2020)

59: LogicBricks, Data Sheet: logiCLK Programmable Clock Generator, v1.5 (2018).

60: Linux Kernel, Kernel Mode Setting(KMS), URL: https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html.

61: B. Feng, "Implementing a tmds video interface in the spartan-6 fpga.", Xilinx, (2010).

62: R. L. Gregg, "Real-Time Streaming Video and Image Processingon Inexpensive Hardware with Low Latency.", (2018)

63: L.G. Shapiro and G. C. Stockman, "Computer Vision.", New Jersey, Prentice-Hall, pp. 279–325, (2001).

64: H. G. Barrow and J. M. Tenenbaum. "Interpreting line drawings as three-dimensional surfaces." Artificial intelligence 17, no. 1-3, pp. 75-116, (1981).

65: T. Lindeberg, "Edge detection and ridge detection with automatic scale selection." International journal of computer vision 30, no. 2, pp. 117-156, (1998).

66: Sobel Edge Detection Implementation, URL: https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection.

67: O. R. Vincent and O. Folorunso. "A descriptive algorithm for sobel image edge detection." In Proceedings of Informing Science & IT Education Conference (InSITE), California: Informing Science Institute, vol. 40, pp. 97-107, (2009).

68: Sobel Edge Detection Example, URL: https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection.

69: Xilinx, Vivado Design Suite User Guide: Synthesis, UG901 (v2017.4).

70: Xilinx, Vivado Design Suite User Guide: Implementation, UG904 (v2017.4).

71: M. T. Amoruso, "Implementation of Video Processing Techniques on a Field Programmable Gate Array Development Platform.", diss. PhD , Drexel University, (2015).

72: Xilinx, Vivado Design Suite User Guide: High-Level Synthesis, UG902 (v2017.4).

73: A. G. Pitsis, "Design and implementation of an FPGA-based convolutional neural network accelerator", Diploma Work, School of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece, (2018)

74: Χ. Α. Νικολακάκης, "Ανάπτυξη αρχιτεκτονικών υλικού για αναγνώριση ακμών σε εικόνες με τεχνικές σύνθεσης υψηλού επιπέδου (High level synthesis).", diss. PhD, (2017).

75: OpenCV Library, URL: https://opencv.org/.

76: Analog Devices, ADV7511 Diagram, URL: https://www.analog.com/en/products/adv7511.html.

77: Xilinx, LogiCore IP Product Guide: AXI Video Direct Memory Access, PG020 (2017.4).

78: Analog Devices, HDMI IP Core, URL: https://wiki.analog.com/resources/fpga/docs/axi_hdmi_tx.

79: Minimal Ubuntu Image, URL: https://wiki.ubuntu.com/Minimal.