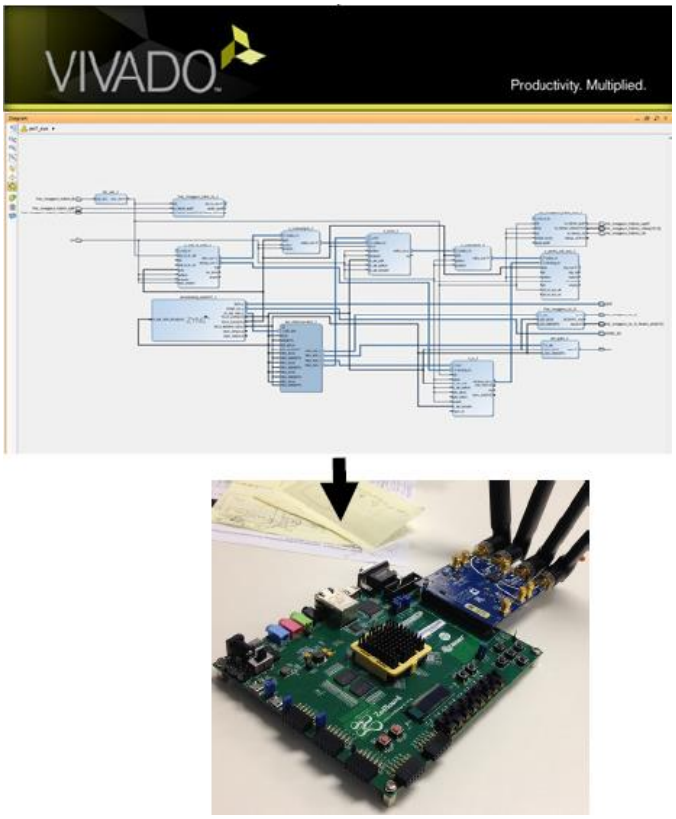
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Ανάπτυξη τηλεπικοινωνιακών υποσυστημάτων σε FPGA με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC και την RF αναπτυξιακή πλακέτα AD-FMCOMMS2/3/4/5 FMC.



Φοιτητής: Ανδρέας Πατεράκης
ΑΜ: 50106895

Επιβλέπων Καθηγητής

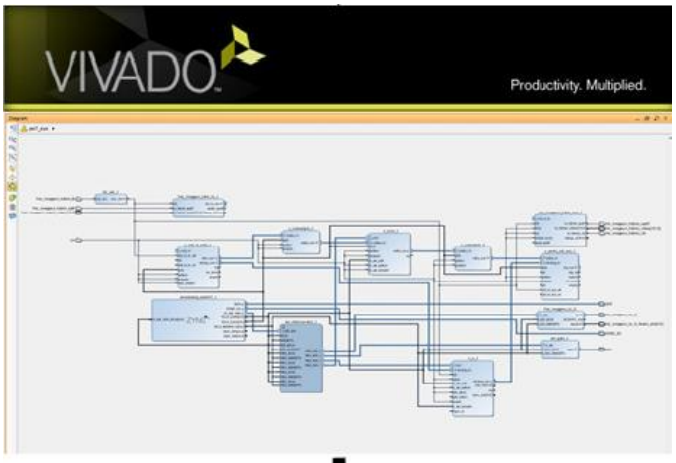
Σωτήριος Καραμπέτσος
Αναπληρωτής Καθηγητής

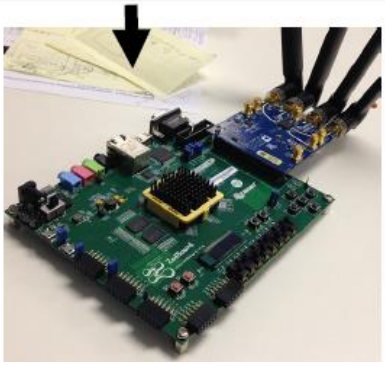
ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2021

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Development of telecommunication Sud-Systems with ZedBoard Zynq-7000 ARM/FPGA SoC Development Board and RF Evaluation Board AD-FMCOMMS2/3/4/5 FMC.





Student: Andreas Paterakis
Registration Number: 50106895

Supervisor

Sotiris Karabetsos
Associate Professor

ATHENS-EGALEO, OCTOBER 2021

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Ευάγγελος Ζέρβας, Καθηγητής	Γρηγόριος Κουλούρας, Επίκουρος Καθηγητής	Σωτήριος Καραμπέτσος, Αναπληρωτής Καθηγητής
Evangelos Zervas Digitally signed by Evangelos Zervas Date: 2021.10.04 18:12:59 +03'00'	 Digitally signed by Grigorios Koulouras DN: c=GR, o=University of West Attica, ou=Class A - Private Key created and stored in hardware CSP, sn=Koulouras, givenName=Grigorios, serialNumber=5281068459, cn=Grigorios Koulouras, email=gregkoul@uniwa.gr Date: 2021.10.04 22:00:44 +03'00'	Sotirios Karampetsos Digitally signed by Sotirios Karampetsos Date: 2021.10.04 17:57:11 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος **Ανδρέας Πατεράκης** του **Αλκιβιάδη**, με αριθμό μητρώου **50106895** φοιτητής του **Πανεπιστημίου Δυτικής Αττικής** της Σχολής **ΜΗΧΑΝΙΚΩΝ** του Τμήματος **ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ**,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.
Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Ο Δηλών

Andreas Paterakis
Digitally signed by Andreas Paterakis
Date: 2021.10.04 16:04:35 +03'00'

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, σηματοδοτείται η ολοκλήρωση του προπτυχιακού κύκλου σπουδών μου στο τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών του Πανεπιστημίου Δυτικής Αττικής. Έτσι θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν με τον τρόπο τους, για την επιτυχή και έγκαιρη περάτωση των σπουδών μου .

Αρχικά, θα ήθελα να ευχαριστήσω τους καθηγητές μου, που σε αυτό το ταξίδι των πέντε ετών, μου προσέφεραν τις κατάλληλες γνώσεις, συμβουλές και εφόδια, ώστε να ξεκινήσω την επαγγελματική μου σταδιοδρομία.

Συγκεκριμένα, επιθυμώ να ευχαριστήσω τον υπεύθυνο καθηγητή μου κ. Σωτήριο Καραμπέτσο, για την θαυμάσια συνεργασία μας όλα αυτά τα χρόνια, όπως επίσης και για την εξαιρετική και συνεχή καθοδήγηση που μου παρείχε στην παρούσα εργασία.

Τέλος, οφείλω ένα μεγάλο «ευχαριστώ» στην οικογένειά μου, διότι δίχως αυτούς όσα έχω καταφέρει μέχρι σήμερα δεν θα ήταν πραγματικότητα. Ειδικά, ευχαριστώ την γυναίκα μου Παναγιώτα-Παρασκευή, που αποδέχτηκε όλες τις επιλογές μου και με στήριξε ψυχολογικά και για τη διαρκή συμπαράσταση κάθε είδους που μου πρόσφερε από την αρχή ως το τέλος στην ολοκλήρωση του προπτυχιακού κύκλου σπουδών μου στο τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών του Πανεπιστημίου Δυτικής Αττικής .

Περίληψη

Η παρούσα διπλωματική εργασία απαρτίζεται από πέντε κύρια κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού - Software Defined Radio (SDR), καθώς και οι παράμετροι που χρειάζονται να ληφθούν υπόψιν για την ανάπτυξη και υλοποίηση τηλεπικοινωνιακών υποσυστημάτων βασισμένων επάνω σε αυτά. Στο δεύτερο κεφάλαιο εξετάζεται η μεθοδολογία σχεδίασης συστημάτων SDR, αναλύονται οι διάφορες κατηγορίες αναπτυξιακών πλατφορμών που αξιοποιούνται σε συστήματα SDR, αλλά και οι διαφορετικές κατηγορίες λογισμικού που απαιτούνται για την σχεδίαση και υλοποίηση ενός συστήματος SDR. Αναφέρονται τα πλεονεκτήματα και τα μειονεκτήματα που προσφέρουν και επιπλέον περιγράφονται κάποιες χαρακτηρίστηκες περιπτώσεις εφαρμογών, στις οποίες χρησιμοποιούνται συστήματα SDR. Στο τρίτο κεφάλαιο γίνεται περιγραφή της αρχιτεκτονικής και της αρχής λειτουργίας τόσο των επί μέρους εξαρτημάτων του εργαστηριακού εξοπλισμού (ZedBoard, AD-FMCOMMS4 FMC), όσο και του λογισμικού (Simulink, Vivado) που χρησιμοποιήθηκε για αυτήν την εργασία. Επίσης, παρουσιάζεται ένας αναλυτικός οδηγός της εγκατάστασης των αναγκαίων προγραμμάτων και εργαλείων, που είναι απαραίτητα για την σύνδεση και την αποκατάσταση της επικοινωνίας μεταξύ του εξοπλισμού και ενός προγραμματιστικού περιβάλλοντος σε ηλεκτρονικό υπολογιστή. Στο τέταρτο κεφάλαιο περιγράφονται τα διαφορετικά καθεστώτα στα οποία λειτουργεί μια αναπτυξιακή πλακέτα FPGA και δίνονται κάποια χαρακτηριστικά παραδείγματα υλοποίησης ενός πλήρους συστήματος SDR για καθένα από τα καθεστώτα αυτά. Τέλος, θα γίνει μια λεπτομερείς παρουσίαση, σχεδίαση, και ανάπτυξη ενός πλήρους αυτόνομου συστήματος SDR, που είναι και το αντικείμενο αυτής της διπλωματικής εργασίας. Στο τελευταίο κεφάλαιο της εργασίας αυτής, αναφέρονται τα προβλήματα που προέκυψαν κατά τη διάρκειά της, γίνονται διάφοροι σχολιασμοί για τα αποτελέσματα των μετρήσεων που πάρθηκαν καθώς και τα συμπεράσματα που εξάγονται σχετικά με την λειτουργικότητα και την αξιοπιστία της συνολικής κατασκευής.

Λέξεις – κλειδιά

SDR, ZedBoard, Vivado, AD-FMCOMMS4-EBZ, AD9364, Simulink, MatLab.

Abstract

This current diploma thesis consists of five main chapters. The first chapter presents the theoretical background of Software Defined Radio (SDR), as well as the parameters that need to be taken into account for the development and implementation of telecommunication subsystems based on them. The second chapter examines the design methodology of SDR systems, analyzes the various categories of development platforms used for SDR systems, but also the different categories of software required to design and implement a SDR system. The advantages and disadvantages of SDR are mentioned and in addition some typical applications, in which SDR systems are used. The third chapter describes the architecture and the principle of operation of both the individual components of the laboratory equipment (ZedBoard, AD-FMCOMMS4 FMC), as well as the software (Simulink, Vivado) used for this diploma thesis. It also presents a detailed guide to installing the necessary programs and tools, which are necessary for the connection and restoration of communication between the equipment and a programming environment on a computer. The fourth chapter describes the different schemes in which an FPGA development board operates and gives some typical examples of implementing a complete SDR system for each of these schemes. Finally, there will be a detailed presentation. design, and development of a fully autonomous SDR system, which is the subject of this diploma thesis. In the last chapter, the problems that arose during it are mentioned, various comments are made on the results of the measurements taken as well as the conclusions that are drawn regarding the functionality and the reliability of the overall construction.

Keywords

SDR, ZedBoard, Vivado, AD-FMCOMMS4-EBZ, AD9364, Simulink, MatLab.

Περιεχόμενα

Κατάλογος Εικόνων 11
Αλφαβητικό Ευρετήριο 17
1 ΚΕΦΑΛΑΙΟ 1ο :Εισαγωγή 20
1.1 Αντικείμενο της διπλωματικής εργασίας 20
1.2 Σκοπός και στόχοι 20
1.3 Μεθοδολογία 20
1.4 Καινοτομία 21
1.5 Δομή 21
2 ΚΕΦΑΛΑΙΟ 2ο : Συστήματα Ραδιοεπικοινωνίας Μέσω Λογισμικού - Software Defined Radio (SDR) 22
2.1 Εισαγωγή SDR 23
2.1.1 Η σύντομη ιστορία των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού (SDR) 23
2.2 Υλοποίηση 24
2.2.1 Αρχιτεκτονική SDR 24
2.2.1.1 Hardware SDR 25
2.2.1.1.1 Κεραία. 25
2.2.1.1.2 RF Front End. 25
2.2.1.1.3 Αναλογική σε ψηφιακή (ADC) και ψηφιακή σε αναλογική μετατροπή (DAC). 26
2.2.1.1.4 Ψηφιακό Front - End. 26
2.2.1.1.5 Επεξεργασία σήματος – Signal Processing. 27
2.2.1.2 Software SDR 27
2.2.1.2.1 Επεξεργαστές Γενικής Χρήσης - General Purpose Processor (GPP) 27
2.2.1.2.2 Επεξεργαστές Ψηφιακού Σήματος - Digital Signal Processor (DSP) 29
2.2.1.2.3 Συστοιχίες επιτόπιας προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA) 30
2.2.1.2.4 Υβριδικές Σχεδιάσεις 30
2.2.2 Software 31
2.2.2.1 Αλγόριθμος Σύνθεσης 32
2.2.2.1.1 Σύνθεση Υψηλού Επιπέδου - High Level Synthesis (HLS) 32
2.2.2.1.2 Γλώσσες Σύνθεσης Υψηλού Επιπέδου - High Level Synthesis Languages 32
2.2.2.1.2.1 C 33
2.2.2.1.2.2 C++ 33
2.2.2.1.2.3 SystemC 33
2.2.2.2 Αναπτυξιακά Εργαλεία 34
2.2.2.2.1 MatLab and Simulink 34
2.2.2.2.2 Vivado HLS 34
2.2.2.2.3 SDSoC 34
2.2.2.2.4 LegUP 35
2.2.2.2.5 GNU Radio 35
2.2.2.2.6 LabVIEW 35
2.2.2.2.7 CUDA 35
2.3 Εφαρμογές 35
2.3.1 Κινητές επικοινωνίες 35
2.3.1.1 Κατασκευή δικτύου κινητής Τηλεφωνίας 36
2.3.2 Έρευνα και ανάπτυξη 36
2.3.3 Στρατιωτική χρήση 36
2.3.3.1 Παρακολούθηση πορείας πλοίων σε πραγματικό χρόνο 36
2.3.3.2 Παρακολούθηση πορείας αεροσκαφών σε πραγματικό χρόνο μέσω της εκπομπής Mode S 36
2.3.4 Χρήση από την Ραδιοερασιτεχνική Κοινότητα 36
2.3.4.1 Λήψη Δορυφορικών Εκπομπών 37

2.3.4.2 Ραδιοερασιτέχνες....37
2.3.4.3 Ψηφιακό ραδιόφωνο - Digital Radio Mondiale (DRM)....37
2.4 Πλεονεκτήματα – Μειονεκτήματα....37
2.4.1 Πλεονεκτήματα....37
2.4.2 Μειονεκτήματα....38
3 ΚΕΦΑΛΑΙΟ 3ο : Αναπτυξιακή πλακέτα και λογισμικό υλοποίησης SDR. 40
3.1 Αναπτυξιακή πλακέτα ZedBoard....41
3.1.1 ZedBoard....41
3.1.2 Zynq – 7000 SoC Αρχιτεκτονική....43
3.1.3 Απλοποιημένη Αρχιτεκτονική SoC....45
3.1.4 Μονάδα Επεξεργασίας Εφαρμογών - Application Processing Unit (APU)....46
3.1.5 Πρότυπο Διαύλου Advanced eXtensible Interface (AXI)....47
3.1.6 Επικοινωνία με την Αναπτυξιακή πλακέτα ZedBoard....47
3.1.7 Προγραμματισμός ZedBoard....48
3.2 AD-FMCOMMS4-EBZ....49
3.2.1 Τεχνικά χαρακτηριστικά AD9364....50
3.2.2 Περιγραφή AD9364....51
3.2.3 Δέκτης AD9364....52
3.2.4 Πομπός AD9364....53
3.3 Λογισμικό....54
3.3.1 Vivado Design Suite....54
3.3.1.1 Διάγραμμα Ροής Vivado HLS....54
3.3.1.2 Είσοδοι της διαδικασίας HLS....55
3.3.1.3 Συνάρτηση Επαλήθευσης....55
3.3.1.4 Σύνθεση Υψηλού Επιπέδου HLS....55
3.3.1.5 Συνδυαστική εξομοίωση C/RTL....55
3.3.1.6 Αξιολόγηση της Ενσωμάτωσης....56
3.3.1.7 Αξιολόγηση της Ενσωμάτωσης....56
3.3.1.8 Εξαγωγή RTL....56
3.3.1.9 Υλοποίηση της λογικής Intellectual Property (IP) στο Vivado....56
3.3.2 MatLab/Simulink....57
3.3.2.1 MatLab....57
3.3.2.2 Simulink....58
3.3.2.3 MatLab HDL Coder....59
3.3.3 Διασύνδεση εξοπλισμού με το προγραμματιστικό περιβάλλον MatLab....61
3.3.3.1 Communication Toolbox Support Package for Xilinx Zynq-Based Radio....61
3.3.3.2 HDL Coder Support Package for Xilinx Zynq....74
3.3.3.3 Embedded Coder Support Package for Xilinx Zynq....77
3.3.4 Εγκατάσταση Vivado Design Suite....79
4 ΚΕΦΑΛΑΙΟ 4ο : Υλοποίηση Τηλεπικοινωνιακού Συστήματος SDR.... 91
4.1 Καθεστώτα Λειτουργίας Αναπτυξιακής Πλακέτας FPGA....92
4.1.1 Μοντέλο Διεπαφής Λογισμικού – Streaming....92
4.1.1.1 Παραδείγματα Μοντέλου Διεπαφής Λογισμικού – Streaming με χρήση κώδικα MatLab (script)....92
4.1.1.1.1 Εκπομπή και λήψη Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9361/AD9364 - Receive Tone Signal Using Analog Devices AD9364 [100]. 92
4.1.1.1.2 QPSK Εκπομπή Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9361/AD9364 - QPSK Transmit Repeat Using Analog Devices AD9364 [102]....93
4.1.1.1.3 Ανάπτυξη τηλεπικοινωνιακών υποσυστημάτων OFDM με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC. [103]....94

4.1.1.2 Παραδείγματα Μοντέλου Διεπαφής Λογισμικού – Streaming με χρήση έτοιμων μπλοκ του Simulink……95
4.1.1.2.1 Υλοποίηση πομπού QPSK με την χρήση του Analog Devices AD9361/AD9364 - QPSK Transmitter Using Analog Devices AD9364 [104] ……95
4.1.1.2.2 Υλοποίηση Πομπού Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με συνολικό φέρον (Double Side Band-Amplitude Modulation-Total Carrier - DSB-AM-TC) ……98
4.1.1.2.3 Υλοποίηση Πομπού Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC) ……99
4.1.1.2.4 Υλοποίηση Πομπού Διαμόρφωση Μονής Πλευρικής Ζώνης με την χρήση φίλτρων (Single Side Band - SSB) ……101
4.1.1.2.5 Υλοποίηση Πομπού Διαμόρφωση Μονής Πλευρικής Ζώνης με την χρήση μετασχηματισμού Hilbert (Hilbert transform Single Side Band - SSB) [105] ……104
4.1.1.2.6 Υλοποίηση Πομποδέκτη Διαμόρφωσης Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC) ……106
4.1.2 Πλήρης Αυτόνομη Ανάπτυξη- Stand Alone ……108
4.1.2.1 Παράδειγμα Υλοποίησης πομποδέκτη QPSK, χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9361/AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης - Stand Alone. ……108
4.1.2.2 Υλοποίηση πομπού QPSK χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης – Stand Alone. ……114
4.1.2.2.1 Βήματα υλοποίησης Τηλεπικοινωνιακού Συστήματος SDR. ……114
4.1.2.2.2 Σχεδίαση Τηλεπικοινωνιακού συστήματος SDR στο Simulink. ……114
4.1.2.2.3 Δημιουργία Μοντέλου υλικού - Hardware Generation Model. ……115
4.1.2.2.4 Διάγραμμα Ροής IP Core - IP Core Generation Workflow. ……118
4.1.2.2.5 Δημιουργία Μοντέλου Διεπαφής και Βιβλιοθήκης Μπλοκ…. ……122
4.1.2.2.6 Δημιουργία Bitstream και ενσωμάτωση του στο FPGA…. ……124
4.1.2.2.7 Επεξεργαστής σε λειτουργία - Processor-in-the-Loop…. ……128
4.1.2.3 Υλοπόιηση Πλήρης Αυτόνομη Ανάπτυξη- Stand Alone…. ……133
4.1.2.3.1 Δημιουργία εκτελέσιμου αρχείου στο καθεστώς Πλήρης Αυτόνομης Ανάπτυξης – Stand Alone……134
5 ΚΕΦΑΛΑΙΟ 5ο : Συμπεράσματα…… 135
Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές …… 136

Κατάλογος Εικόνων

Εικόνα 2.1.1.1 Μπλοκ διάγραμμα αναλογικού ράδιο-τηλεπικοινωνιακού συστήματος ……[23]
Εικόνα 2.1.1.2 Μπλοκ διάγραμμα ενός «ιδανικού» SDR ……..……[24]
Εικόνα 2.2.1.1 Μπλοκ διάγραμμα SDR. Υπό εικόνα (a) πομπός SDR και (b) δέκτης SDR .……[25]
Εικόνα 2.2.2.1.1 Διάγραμμα ροής της HLS ……[32]
Εικόνα 2.2.2.2.1 Διάγραμμα ροής της MathWork για SoC πλατφόρμες ……...……[34]
Εικόνα 3.1.1.1 Διάταξη διεπαφών πλακέτας ZedBoard [69] ……..……[42]
Εικόνα 3.1.1.2 Μπλοκ διάγραμμα πλακέτας ZedBoard[69] ……..……..…[42]
Εικόνα 3.1.1.2 Μπλοκ διάγραμμα επεξεργαστών Zynq – 7000 SoC [69] ……..……[43]
Εικόνα 3.1.1.3. Αρχιτεκτονικής συστήματος επεξεργασίας Zynq – 7000 SoC [69] …..……[44]
Εικόνα 3.1.1.4. Απλοποιημένη Αρχιτεκτονικής Zynq – 7000 SoC [69] ……..……...……[44]
Εικόνα 3.1.3.1. Απλοποιημένη Αρχιτεκτονική ενός SoC [69] ……..……..……[45]
Εικόνα 3.1.3.2. Συσχέτιση μεταξύ Συστήματος Υλικού, Συστήματος Λογισμικού και Αρχιτεκτονικής Zynq [69] ……..……[45]
Εικόνα 3.1.4.1. Μπλοκ διάγραμμα την APU [69] ……..……[46]
Εικόνα 3.1.5.1. Θύρες σύνδεσης της αναπτυξιακής πλακέτας ZedBoard [69] ……..…...…..…[48]
Εικόνα 3.1.6.1. Ακίδες jumpers της πλακέτας ZedBoard [69] ……..……..……...…..…[49]
Εικόνα 3.1.6.2. Θέσεις ακίδων jumpers της πλακέτας ZedBoard [69] ……..……..……[49]
Εικόνα 3.2.1 Πρόσθια όψη πλακέτας AD-FMCOMMS4-EBZ [73] ……..……..……[50]
Εικόνα 3.2.2 Οπίσθια όψη πλακέτας AD-FMCOMMS4-EBZ [73] ……..……..……[50]
Εικόνα 3.2.3 Διάταξη κύριων εξαρτημάτων της πλακέτας AD-FMCOMMS4-EBZ [73] .……[50]
Εικόνα 3.2.2.1 Λειτουργικό Μπλοκ διάγραμμα ολοκληρωμένου AD9364 [73] ..……..…[52]
Εικόνα 3.2.2.2 Μπλοκ διάγραμμα δομής ολοκληρωμένου AD9364 [83] ……..……..……[52]
Εικόνα 3.3.1.1 Διάγραμμα Ροής Vivado [69] ……..……..……...……[54]
Εικόνα 3.3.1.1.1 Στάδια Σύνθεσης Υψηλού επιπέδου HLS [69] ……..……..……[54]
Εικόνα 3.3.3.1 Παράδειγμα Ολοκληρωτή IP- IP Integrator [69] ……..……..……[57]
Εικόνα 3.3.2.1.1 Επιφάνεια εργασίας MatLab ……..……..……[58]
Εικόνα 3.3.2.2.1 Παράδειγμα υλοποίηση QPSK πομπού και δέκτη στο Simulink [98] ……[59]
Εικόνα 3.3.2.3.1 Παράδειγμα HDL Workflow Advisor [95] ……..……...……[60]
Εικόνα 3.3.2.3.2 Διάγραμμα ροής HDL Coder [95] ……..……..……[61]
Εικόνα 3.3.3.1.1 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (1/23) ……..……..……..……[62]
Εικόνα 3.3.3.1.2 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (2/23) ……..……..……..……[62]
Εικόνα 3.3.3.1.3 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (3/23) ……..……..……..……[63]
Εικόνα 3.3.3.1.4 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (4/23) ……..……..……..……[63]
Εικόνα 3.3.3.1.5 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (5/23) ……..……..……..……[64]
Εικόνα 3.3.3.1.6 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (6/23) ……..……..……..……[64]
Εικόνα 3.3.3.1.7 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (7/23) ……..……..……..……[65]
Εικόνα 3.3.3.1.8 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (8/23) ……..……..……..……[65]
Εικόνα 3.3.3.1.9 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (9/23) ……..……..……..……[66]
Εικόνα 3.3.3.1.10 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (10/23) ……..……..……..…..……[66]
Εικόνα 3.3.3.1.11 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (11/23)……..……..……..……..……[67]

Εικόνα 3.3.3.1.12 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (12/23) …..……..…..……..……[67]
Εικόνα 3.3.3.1.13 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (13/23) …….……..…..……..……[68]
Εικόνα 3.3.3.1.14 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (14/23) …..……..…..……..……[68]
Εικόνα 3.3.3.1.15 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (15/23) ...……..…..……..……[69]
Εικόνα 3.3.3.1.16 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (16/23) …….……..…..……[69]
Εικόνα 3.3.3.1.17 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (17/23) …….……..…..……[70]
Εικόνα 3.3.3.1.18 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (18/23) …….……..…..……[70]
Εικόνα 3.3.3.1.19 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (19/23) …….……..…..……[71]
Εικόνα 3.3.3.1.20 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (20/23) ……..…..……[71]
Εικόνα 3.3.3.1.21 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (21/23) ……..…..……[72]
Εικόνα 3.3.3.1.22 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (22/23) …….……..…..……[72]
Εικόνα 3.3.3.1.23 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (23/23) …….……..…..……[72]
Εικόνα 3.3.3.1.24 Επίτευξη ορθής επικοινωνία με την πλακέτα ZedBoard ……...…[73]
Εικόνα 3.3.3.1.25 Επίτευξη ορθής επικοινωνία με την πλακέτα ZedBoard..……..…[74]
Εικόνα 3.3.3.1.26 Αδυναμία επικοινωνίας με την πλακέτα ZedBoard ..……...…[74]
Εικόνα 3.3.3.2.1 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (1/7) ..[74]
Εικόνα 3.3.3.2.2 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (2/7) ..[75]
Εικόνα 3.3.3.2.3 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (3/7) ..[75]
Εικόνα 3.3.3.2.4 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (4/7) ..[75]
Εικόνα 3.3.3.2.5 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (5/7) ..[76]
Εικόνα 3.3.3.2.6 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (6/7) ..[76]
Εικόνα 3.3.3.2.7 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (7/7) ..[77]
Εικόνα 3.3.3.3.1 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (1/7) ..[77]
Εικόνα 3.3.3.3.2 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (2/7) ..[77]
Εικόνα 3.3.3.3.3 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (3/7) ..[78]
Εικόνα 3.3.3.3.4 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (4/7) ..[78]
Εικόνα 3.3.3.3.5 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (5/7) ..[78]
Εικόνα 3.3.3.3.6 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (6/7) ..[79]
Εικόνα 3.3.3.3.7 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (7/7) ..[79]
Εικόνα 3.3.4.1 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (1/32) ……[80]
Εικόνα 3.3.4.2 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (2/32) ……[80]
Εικόνα 3.3.4.3 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (3/32) ……[80]
Εικόνα 3.3.4.4 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (4/32) ……[81]
Εικόνα 3.3.4.5 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (5/32) ……[81]
Εικόνα 3.3.4.6 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (6/32) ……[81]
Εικόνα 3.3.4.7 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (7/32) ……[82]
Εικόνα 3.3.4.8 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (8/32) ……[82]
Εικόνα 3.3.4.9 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (9/32) ……[82]
Εικόνα 3.3.4.10 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (10/32) ……[83]

Εικόνα 3.3.4.11 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (11/32) .....[83]
Εικόνα 3.3.4.12 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (12/32) .....[83]
Εικόνα 3.3.4.13 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (13/32) .....[84]
Εικόνα 3.3.4.14 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (14/32) .....[84]
Εικόνα 3.3.4.15 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (15/32) .....[84]
Εικόνα 3.3.4.16 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (16/32) .....[85]
Εικόνα 3.3.4.17 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (17/32) .....[85]
Εικόνα 3.3.4.18 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (18/32) .....[85]
Εικόνα 3.3.4.19 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (19/32) .....[86]
Εικόνα 3.3.4.20 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (20/32) .....[86]
Εικόνα 3.3.4.21 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (21/32) .....[86]
Εικόνα 3.3.4.22 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (22/32) .....[87]
Εικόνα 3.3.4.23 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (23/32) .....[87]
Εικόνα 3.3.4.24 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (24/32) .....[87]
Εικόνα 3.3.4.25 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (25/32) .....[87]
Εικόνα 3.3.4.26 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (26/32) .....[88]
Εικόνα 3.3.4.27 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (27/32) .....[88]
Εικόνα 3.3.4.28 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (28/32) .....[88]
Εικόνα 3.3.4.29 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (29/32) .....[89]
Εικόνα 3.3.4.30 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (30/32) .....[89]
Εικόνα 3.3.4.31 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (31/32) .....[89]
Εικόνα 3.3.4.32 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (32/32) .....[90]
Εικόνα 4.1.1.1.1.1 Φάσμα εκπομπής Σύνθετου Ημιτονοειδούς Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard και την RF πλακέτα AD-FMCOMMS4 FMC, σε αναλυτή φάσματος του MatLab. .....[93]
Εικόνα 4.1.1.1.1.2 Φάσμα εκπομπής Σύνθετου Ημιτονοειδούς Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard και την RF πλακέτα AD-FMCOMMS4 FMC σε εργαστηριακό αναλυτή φάσματος . .....[93]
Εικόνα 4.1.1.1.2.1 Φάσμα QPSK Εκπομπής με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της Analog Devices AD9364 σε εργαστηριακό αναλυτή φάσματος .....[94]
Εικόνα 4.1.1.1.3.1 Φάσμα OFDM εκπομπής 32-QAM με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, από αναλυτή φάσματος του MatLab .....[95]
Εικόνα 4.1.1.1.3.2 Φάσμα OFDM εκπομπής 32-QAM με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, από εργαστηριακό αναλυτή φάσματος .....[95]
Εικόνα 4.1.1.2.1.1 Command Window του MatLab .....[96]
Εικόνα 4.1.1.2.1.2 Παράδειγμα υλοποίησης QPSK Εκπομπής με την χρήση έτοιμων μπλοκ του Simulink για την αναπτυξιακή πλακέτας ZedBoard και την Analog Devices AD9364 .....[96]
Εικόνα 4.1.1.2.1.3 QPSK Εκπομπή με την χρήση έτοιμων μπλοκ του Simulink για την αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[97]
Εικόνα 4.1.1.2.1.4 Ρυθμίσεις της Analog Devices AD9364 .....[97]
Εικόνα 4.1.1.2.1.5 Φέρουσα της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος σε κατάσταση ηρεμίας .....[97]
Εικόνα 4.1.1.2.2.1 Υλοποίηση πομπού DSB-AM-TC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364 .....[98]
Εικόνα 4.1.1.2.2.2 Φάσμα πομπού DSB-AM-TC, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices .....[98]

Εικόνα 4.1.1.2.2.3 Φάσμα πομπού DSB-AM-TC στην έξοδο της RF αναπτυξιακής πλακέτας AD - FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[99]
Εικόνα 4.1.1.2.2.4 Φάσμα δέκτη DSB-AM-TC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[99]
Εικόνα 4.1.1.2.3.1 Υλοποίηση πομπού DSB-AM-SC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364 .....[100]
Εικόνα 4.1.1.2.3.2 Φάσμα πομπού DSB-AM-SC, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[100]
Εικόνα 4.1.1.2.3.3 Φάσμα πομπού DSB-AM-SC στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος. .....[100]
Εικόνα 4.1.1.2.3.4 Φάσμα δέκτη DSB-AM-SC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[101]
Εικόνα 4.1.1.2.4.1 Υλοποίηση πομπού SSB με την χρήση φίλτρων, μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364 .....[101]
Εικόνα 4.1.1.2.4.2 Φάσμα πομπού USB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[102]
Εικόνα 4.1.1.2.4.3 Φάσμα πομπού USB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[102]
Εικόνα 4.1.1.2.4.4 Φάσμα δέκτη USB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[103]
Εικόνα 4.1.1.2.4.5 Φάσμα πομπού LSB, πριν την διαμόρφωση του από την φέρουσα (Fc της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[103]
Εικόνα 4.1.1.2.4.6 Φάσμα πομπού LSB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[103]
Εικόνα 4.1.1.2.4.7 Φάσμα δέκτη LSB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink. .....[103]
Εικόνα 4.1.1.2.5.1 Υλοποίηση πομπού SSB με την χρήση μετασχηματισμού Hilbert, μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364 .....[104]
Εικόνα 4.1.1.2.5.2 Φάσμα πομπού LSB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[104]
Εικόνα 4.1.1.2.5.3 Φάσμα πομπού LSB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[105]
Εικόνα 4.1.1.2.5.4 Φάσμα δέκτη LSB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[105]
Εικόνα 4.1.1.2.5.5 Φάσμα πομπού USB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink .....[105]
Εικόνα 4.1.1.2.5.6 Φάσμα πομπού USB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .....[106]

Εικόνα 4.1.1.2.5.7 Φάσμα δέκτη USB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink ……[106]
Εικόνα 4.1.1.2.6.1 Υλοποίηση πομποδέκτη DSB-AM-SC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364 ……[107]
Εικόνα 4.1.1.2.6.2 Φάσμα πομπού DSB-AM-SC, πριν την διαμόρφωση του από την φέρουσα (Fc) της της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink ……[107]
Εικόνα 4.1.1.2.6.3 Φάσμα πομπού DSB-AM-SC στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος. ……[107]
Εικόνα 4.1.1.2.6.4 Φάσμα δέκτη DSB-AM-SC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink ……[108]
Εικόνα 4.1.1.2.6.5 Φάσμα στην είσοδο του μεγάφωνου, σε αναλυτή φάσματος του Simulink ……[108]
Εικόνα 4.1.2.1.1 Μπλοκ διάγραμμα πομποδέκτη QPSK χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης - Stand Alone. [107] ……[110]
Εικόνα 4.1.2.2.1.1 Διάγραμμα ροής σχεδίασης και υλοποίησης τηλεπικοινωνιακού συστήματος SDR …[110]
Εικόνα 4.1.2.2.2.1 Μπλοκ Διάγραμμα QPSK πομπού στο Simulink ……[111]
Εικόνα 4.1.2.2.2.2 Μπλοκ Διάγραμμα Αλγόριθμου QPSK πομπού στην Προγραμματιζόμενη Λογική - Logic Programmable (PL) του FPGA, στο Simulink ……[111]
Εικόνα 4.1.2.2.2.3 Εξομοίωση QPSK πομπού στο Simulink πριν την διαμόρφωση του από την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC ……[111]
Εικόνα 4.1.2.2.3.1 HDL Core του Simulink ……[112]
Εικόνα 4.1.2.2.3.2 HDL Core Advisor του Simulink ……[112]
Εικόνα 4.1.2.2.3.3 Έλεγχος μοντέλου με το HDL Core Advisor του Simulink ……[113]
Εικόνα 4.1.2.2.3.4 HDL Core Advisor του Simulink ……[113]
Εικόνα 4.1.2.2.3.5 Προειδοποίηση (Warning) του HDL Core Advisor στο εξεταζόμενο μοντέλο ……[113]
Εικόνα 4.1.2.2.3.6 Επιδιόρθωση προειδοποίησης (Warning) του HDL Core Advisor στο εξεταζόμενο μοντέλο ……[114]
Εικόνα 4.1.2.2.3.7 Μοντέλο χωρίς σφάλματα (Errors) ή προειδοποιήσεις (Warnings) του HDL Core Advisor ……[114]
Εικόνα 4.1.2.2.4.1 Ενεργοποίηση του εργαλείου Vivado της Xilinx ……[114]
Εικόνα 4.1.2.2.4.2 Ενεργοποίηση του IP HDL Core repositories ……[115]
Εικόνα 4.1.2.2.4.3 HDL Workflow Advisor(1/2) ……[115]
Εικόνα 4.1.2.2.4.4 HDL Workflow Advisor(2/2) ……[115]
Εικόνα 4.1.2.2.4.5 Βήμα 1.1 ……[116]
Εικόνα 4.1.2.2.4.6 Βήμα 1.2 ……[116]
Εικόνα 4.1.2.2.4.7 Βήμα 1.3 ……[117]
Εικόνα 4.1.2.2.4.8 Βήμα 1.4 ……[117]
Εικόνα 4.1.2.2.4.9 Βήμα 2 ……[117]
Εικόνα 4.1.2.2.4.10 Βήμα 3 ……[118]
Εικόνα 4.1.2.2.4.11 Αναφορά Βήματος 3 ……[118]
Εικόνα 4.1.2.2.4.12 Αναφορά Δημιουργίας Κώδικα Βήματος 3. ……[118]
Εικόνα 4.1.2.2.5.1 Βήμα 4.1 ……[119]
Εικόνα 4.1.2.2.5.2 Βήμα 4.2 ……[119]
Εικόνα 4.1.2.2.5.3 Μοντέλο διεπαφής της αναπτυξιακής πλακέτας Zynq ……[119]
Εικόνα 4.1.2.2.5.4 Βιβλιοθήκη μπλοκ του αναπτυσσόμενο μοντέλου ……[120]
Εικόνα 4.1.2.2.5.5 Μοντέλο Διεπαφής QPSK πομπού ……[120]
Εικόνα 4.1.2.2.6.1 Βήμα 4.3 ……[121]
Εικόνα 4.1.2.2.6.2 Εξωτερικό Παράθυρο δημιουργίας Bitstream ……[121]

Εικόνα 4.1.2.2.6.3 Επιτυχημένη ολοκλήρωση δημιουργίας του Bitstream στο Εξωτερικό Παράθυρο …..[121]
Εικόνα 4.1.2.2.6.4 Επιτυχημένη λήψη του Bitstream από την αναπτυξιακή πλακέτα ZedBoard ……..[122]
Εικόνα 4.1.2.2.6.5 Επιτυχημένη λήψη του Bitstream από την αναπτυξιακή πλακέτα ZedBoard χωρίς την χρήση του Workflow Advisor ……..[123]
Εικόνα 4.1.2.2.7.1 Αλγόριθμος Καθεστώτος Εξωτερικής Λειτουργίας Επεξεργαστή ARM [109]……..[123]
Εικόνα 4.1.2.2.7.2 Model Configuration Parameters ……..[124]
Εικόνα 4.1.2.2.7.3 Commonly Used Parameters ……..[124]
Εικόνα 4.1.2.2.7.4 Hardware Implementations ……..[124]
Εικόνα 4.1.2.2.7.5 Επαλήθευση φυσικής διεύθυνσης IP της αναπτυξιακής πλακέτας ZedBoard ……..[124]
Εικόνα 4.1.2.2.7.6 Monitor & Tune ……..[125]
Εικόνα 4.1.2.2.7.7 Έναρξη διαδικασίας Monitor & Tune ……..[125]
Εικόνα 4.1.2.2.7.8 Έναρξη κατασκευής μοντέλου από τον ενσωματωμένο κωδικοποιητή ……..[125]
Εικόνα 4.1.2.2.7.9 Έναρξη διαδικασίας εγκατάστασης του εκτελέσιμου αρχείο του ARM στην αναπτυξιακή πλακέτα ZedBoard, σύνδεση του με το μοντέλο και εκτέλεση του από αυτή ……..[126]
Εικόνα 4.1.2.2.7.10 Οι ρυθμιζόμενες παράμετροι του μοντέλου ……..[126]
Εικόνα 4.1.2.2.7.11 Κεντρική συχνότητα εκπομπής 2,4 GHz ……..[127]
Εικόνα 4.1.2.2.7.12 Κεντρική συχνότητα εκπομπής 2,45 GHz ……..[127]
Εικόνα 4.1.2.2.7.13 Συντελεστής ενίσχυσης 0 (-15,67 dBm) ……..[128]
Εικόνα 4.1.2.2.7.14 Συντελεστής ενίσχυσης -10 (-25,83 dBm) ……..[128]
Εικόνα 4.1.2.2.7.15 Συντελεστής ενίσχυσης -30 (-45,17 dBm) ……..[128]
Εικόνα 4.1.2.2.7.16 Συντελεστής ενίσχυσης -60 (-76,50 dBm) ……..[128]
Εικόνα 4.1.2.2.8.17 Διάταξη υλικού QPSK Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς «Επεξεργαστής σε λειτουργία - Processor-in-the-Loop» ……..[129]
Εικόνα 4.1.2.3.1.1 Επιλογή διαδιακασίας «Build Stand-Alone» ……..[130]
Εικόνα 4.1.2.3.1.2 Έναρξη διαδιακασίας «Build Stand-Alone» ……..[130]
Εικόνα 4.1.2.3.1.3 Επιτυχής ολοκλήρωσης διαδιακασίας «Build Stand-Alone» ……..[130]
Εικόνα 4.1.2.3.1.4 Δημιουργία εκτελέσιμου αρχείου «gm_QPSK_interface.elf» του μοντέλου ……..[131]
Εικόνα 4.1.2.3.1.5 Φάκελος στον οποίο αποθηκεύεται το εκτελέσιμο αρχείο «gm_QPSK_interface.elf» του μοντέλου ……..[131]
Εικόνα 4.1.2.3.1.6 Αντιγραφή του εκτελέσιμου αρχείου «gm_QPSK_interface.elf» του μοντέλου στην κάρτα SD της αναπτυξιακής πλακέτας ZedBoard ……..[132]
Εικόνα 4.1.2.3.1.7 Άνοιγμα του αρχείου «init.sh» στην κάρτα SD, με την χρήση του του εργαλείου «Σημειωματάριο» των Windows ……..[132]
Εικόνα 4.1.2.3.1.8 Επεξεργασία του αρχείου «init.sh» στην κάρτα SD, με την χρήση του του εργαλείου «Σημειωματάριο» των Windows ……..[133]
Εικόνα 4.1.2.3.1.9 QPSK εκπομπή Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς Πλήρης Αυτόνομης Ανάπτυξης ……..[133]
Εικόνα 4.1.2.3.1.10 Διάταξη υλικού QPSK Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς «Πλήρης Αυτόνομης Ανάπτυξης» ……..[134]

Αλφαβητικό Ευρετήριο

ACP: Accelerator Coherency Port
ADC: Analog to Digital Converter
AGC: Automatic Gain Control
AIS: Automatic Identification System
ALU: Arithmetic and Logic Unit
AM: Amplitude Modulation
ANSI: American National Standards Institute
APT: Automatic Picture Transmission
APU: Application Processing Unit
ARM: Advanced RISC Machine
ASIC: Application-specific integrated circuit
ATTN: Attenuator
AXI: Advanced eXtensible Interface
BBP: Baseband Processor
BER: Bit Error Rate
BPF Band Pass Filter
DAC: Digital to Analog Converter
DDC: Digital Down Converter
DF: Decimation Filter
DMA: Direct memory access
DRM: Digital Radio Mondiale
DSB-AM-SC: Double Side Band-Amplitude Modulation-Suppressed Carrier
DSB-AM-TC: Double Side Band-Amplitude Modulation-Total Carrier
DSP: Digital Signal Processor
DUC: Digital Up Converter
DVB: Digital Video Broadcasting
E2E: End-to-End
ESL Electronic System Level
EVM: Error Vector Magnitude
FDD: Feature-Driven Development
FDD: Frequency Division Duplex
FDM: Frequency Division Multiplexing
FFT: Fast Fourier Transform
FIFO: First in First Out
FIR: Finite Impulse Response
FM: Frequency Modulation
FMC: Fixed Mobile Convergence
FPGA: Field Programmable Gate Array
FPU: Floating Point Unit
GOPS: Giga Operations per Second
GPIO: General Purpose Input/Output
GPP: General Purpose Processor
GPU: Graphic Processing Unit
HDL: Hardware Description Language
HLS: High Level Synthesis
HPC: High Performance Computing
HPSDR: High Performance Software Defined Radio
I: In-phase
ICI: Intercarrier Interference
IDE: Intergrated Design Enviroment
IDFT: Inverse Discrete Fourier

IF: Intermediate Frequency
IF: Interpolation Filter
IFFT: Inverse Fast Fourier Transform
IoT: Internet of Things
IP: Intellectual Property
ISI: Intersymbol Interference
ISO: International Organization for Standardization
JTAG: Joint Test Action Group
LDPC: Low Density Parity Check
LNA: Low Noise Amplifier
LO: Local Oscillator
LSB: Lower Side Band
LTE: Long Term Evolution
LTE-A: Long Term Evolution Advanced
LVDS: Low Voltage Differential Signaling
MAC: Multiply Accumulates
MCM: Multi Carrier Modulation
MMU: Memory Management Unit
MPE: Media Processing Engine
MPSoC: Multi-processor SoC
NOAA: National Oceanic and Atmospheric Administration
OCM: On Chip Memory
OFDM: Orthogonal Frequency Division Multiplexing
OLED: Organic Light Emitting Diode
OS: Operation System
PA: Power Amplification
PC: Personal Computer
PL: Programmable Logic
PLL: Phase-Locked Loop
PS: Processing System
Q: Quadrature
QAM: Quadrature Modulation
RF: Radio Frequency
RSSI: Received Signal Strength Indication
RTL: Register-Transfer Level
SCU: Snoop Control Unit
SDK: Software Development Kit
SDR: Software Defined Radio
SFDR: Spurious-free Dynamic Range
SNR: Signal to Noise Ratio
SoC: System on a Chip
SPMD: Single Program Multiple Data
SRC: Sample Rate Conversion
SSB: Single Side Band
TDD: Test-Driven Development
TDD: Time Division Duplex
TIA: Transimpedance Amplifier
TLM: Transaction Level Modeling
UART: Universal Asynchronous Receiver-Transmitter
USB: Universal Serial Bus
USB: Upper Side Band
USPR: Universal Software Radio Peripheral

XADC: Xilinx Analog Mixed Signal Module

1 ΚΕΦΑΛΑΙΟ 1^{Ο} :Εισαγωγή

Με την εκθετική ανάπτυξη στους τρόπους και τα μέσα με τα οποία οι άνθρωποι χρειάζεται να επικοινωνήσουν - μετάδοση δεδομένων, φωνητικές επικοινωνίες, μετάδοση βίντεο, ανταλλαγή γραπτών μηνυμάτων , μετάδοση εντολών ελέγχου, επικοινωνίες απόκρισης έκτακτης ανάγκης κ.λπ. – η ανάγκη τροποποίησης των ήδη υπάρχων συσκευών τηλεπικοινωνίας , με τρόπο εύκολο και όσο το δυνατό πιο οικονομικά, έχει καταστεί άκρως επιτακτική για τις επιχειρήσεις. Η τεχνολογία των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού - Software Defined Radio (SDR) φέρνει την ευελιξία, την αποδοτικότητα του κόστους και την ισχύ για να οδηγήσει τις επικοινωνίες προς τα εμπρός, με ευρεία οφέλη τόσο στους πάροχους των εν λόγω υπηρεσιών και στους προγραμματιστές/κατασκευαστές των τηλεπικοινωνιακών προϊόντων όσο και στους τελικούς χρήστες.

Τα τελευταία χρόνια, οι τεχνολογίες που απαιτούνται για την εφαρμογή της έννοιας των συστημάτων SDR έχουν ωριμάσει και πλέον προσφέρονται συστήματα που παρέχουν στους προγραμματιστές συστημάτων, ευελιξία και προσαρμοστικότητα, σημαντικά πλεονεκτήματα για την αντιμετώπιση ζητημάτων όπως το περιορισμένο διαθέσιμο φάσμα, η ολοένα και πιο γρήγορη αλλαγή στα πρότυπα των ασύρματων επικοινωνιών και η ανάγκη για την δημιουργία οικονομικά αποδοτικότερων τηλεπικοινωνιακών προϊόντων για εξειδικευμένες αγορές. Πλέον, βλέπουμε την χρήση συστημάτων SDR σε εφαρμογές που απαιτούν υψηλή ποιότητα υπηρεσιών, όπως σε επικοινωνίες δεδομένων κινητής τηλεφωνίας. Στο μέλλον μπορεί να φανταστεί κανείς ότι οι αρχιτεκτονικές SDR θα χρησιμοποιούνται όλο και περισσότερο για την παροχή τηλεπικοινωνιακών υπηρεσιών όπως η κινητή τηλεφωνία και οι ψηφιακές τηλεοπτικές και ραδιοφωνικές εκπομπές.

1.1 Αντικείμενο της διπλωματικής εργασίας

Στην παρούσα διπλωματική εργασία παρουσιάζεται το θεωρητικό υπόβαθρο των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού (Software Defined Radio - SDR), η αναλυτική περιγραφή του SDR εξοπλισμού που χρησιμοποιήθηκε καθώς και η διασύνδεσή μεταξύ των επιμέρους τμημάτων του. Επίσης θα σχεδιαστεί και θα υλοποιηθεί ένα αυτόνομο σύστημα SDR, συγκεκριμένων χαρακτηριστικών, το οποίο θα αποτελεί τον κεντρικό άξονα ενός τηλεπικοινωνιακού συστήματος. Το σύστημα αυτό θα απαρτίζεται από διάφορα επιμέρους τμήματα που θα εξηγηθούν στην πορεία όπως και θα εξηγηθούν ειδικές περιπτώσεις οι οποίες προέκυψαν κατά την δημιουργία του συστήματος και οι τρόποι που αυτές αντιμετωπίστηκαν.

1.2 Σκοπός και στόχοι

Σκοπός της εργασίας αυτής είναι η κατανόηση της μεθοδολογίας και της λειτουργίας των SDR συστημάτων και των δυνατοτήτων που αυτά προσφέρουν, με πεδίο εφαρμογής τις αναπτυξιακές πλατφόρμες FPGA. Κατά την διάρκεια της εργασίας αυτής, ο αναγνώστης θα μπορέσει εύκολα, χωρίς την απαίτηση κάποιων προ απαιτούμενων εξειδικευμένων γνώσεων στον τομέα των ψηφιακών επικοινωνιών, να προβεί στην σχεδίαση, στην υλοποίηση και στην κατασκευή, ο ίδιος ενός τηλεπικοινωνιακού συστήματος με την χρήση SDR ή τουλάχιστον να καταλάβει την δομή και τον τρόπο σκέψης που θα πρέπει να ακολουθηθεί για την δημιουργία τέτοιων συστημάτων. Ξεκινώντας από την υλοποίηση απλών, βασικών, αναλογικών διαμορφώσεων, αφού κατανοήσει τον μηχανισμό που διέπει τα συστήματα SDR, μετέπειτα θα είναι σε θέση να προχωρήσει στην υλοποίηση περισσότερο εξειδικευμένων διαμορφώσεων (ψηφιακών διαμορφώσεων), που κατά κόρον χρησιμοποιούνται, σε όλα τα σύγχρονα τηλεπικοινωνιακά πρότυπα και συστήματα.

1.3 Μεθοδολογία

Για την σχεδίαση και υλοποίηση ενός αυτονόμου τηλεπικοινωνιακού συστήματος SDR, θα πρέπει να εξηγηθούν αναλυτικά τόσο το θεωρικό υπόβαθρο στο οποίο στηρίζονται όσο και το υλικό (hardware) και το λογισμικό (software), στοιχεία τα οποία κάνουν εφαρμόσιμο ένα τέτοιο σύστημα, όπως επίσης και τα πλεονεκτήματα και μειονεκτήματα που ένα τέτοιο σύστημα προσφέρει. Στη συνέχεια θα εξηγηθεί ο τρόπος με τον οποίο υλοποιείται ένα σύστημα SDR, η δομή του πομπού και

του δέκτη καθώς και οι διασυνδέσεις που απαιτούνται για να λειτουργήσουν. Έπειτα θα αναλυθεί ο τρόπος με τον οποίο το υλικό (hardware) συνδέεται με κάποιο συγκεκριμένο προγραμματιστικό περιβάλλον (software) και ο τρόπος λειτουργίας τους. Το επόμενο βήμα είναι η σχεδίαση και υλοποίηση ενός αλγορίθμου που θα εξομοιώνει την βασική λειτουργία ενός αυτόνομου τηλεπικοινωνιακού συστήματος SDR. Βασική επιδίωξη είναι η υλοποίηση κάποιας βασικής αναλογικής διαμόρφωσης (AM, SSB,κλπ.) και στην συνέχεια κάποια από τις σύγχρονες ψηφιακές διαμορφώσεις (QPSK, QAM, κλπ.). Επίσης θα πρέπει να εξετάσει στον αλγόριθμο αυτόν κάθε πιθανό σενάριο λειτουργίας του εξοπλισμού και θα πρέπει επίσης για αυτά τα σενάρια να υπάρχουν τρόποι οι οποίοι θα τα αντιμετωπίζουν και θα τα παρακάμπτουν ή στην περίπτωση που δεν θα μπορούν να αντιμετωπιστούν το σύστημα θα πρέπει να αυτοπροσασμόζεται.

1.4 Καινοτομία

Μέσω της παρούσας εργασίας έχει επιτευχθεί η δημιουργία και η κατασκευή ενός πραγματικού, αξιόπιστου και αυτόνομου τηλεπικοινωνιακού συστήματος SDR, το οποίο μπορεί εύκολα να προσαρμοστεί λειτουργικά σε κάθε είδους περιβάλλον. Επιπλέον αυτό το σύστημα μπορεί να παραμετροποιηθεί κατάλληλα χωρίς να απαιτείται άλλος εξοπλισμός πέρα από αυτόν που χρησιμοποιείται ήδη ώστε να αλλάξει η λειτουργία του και προσαρμοστεί ευκολά και γρηγορά σε νέα τηλεπικοινωνιακά πρότυπα ή μοντέλα και σε νέες διαμορφώσεις που θα προκύψουν στο μέλλον.

1.5 Δομή

Η συγκεκριμένη διπλωματική εργασία χωρίζεται σε τρία βασικά κεφάλαια. Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στο θεωρητικό υπόβαθρο και στην αρχιτεκτονική των συστημάτων SDR. Στο ίδιο κεφάλαιο, γίνεται αναφορά στις βασικές κατηγορίες υλικού (Hardware) και λογισμικού (Software), που χρησιμοποιούνται σε διάφορα συστήματα SDR, οι ανάγκες/απαιτήσεις που κλήθηκαν αυτά τα συστήματα να βρουν εφαρμογή, όπως και τα πλεονεκτήματα-μειονεκτήματά, που αυτά τα συστήματα προσφέρουν. Στο δεύτερο κεφάλαιο γίνεται μια εισαγωγή στα τεχνικά χαρακτηριστικά και στην αρχιτεκτονικής της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, όπως και στα χαρακτηριστικά του λογισμικού (MatLab, Simulink, Vivado), που χρησιμοποιούνται στην υλοποίηση ενός αυτόνομου τηλεπικοινωνιακού συστήματος SDR. Στη συνέχεια δίνονται αναλυτικές οδηγίες την παραμετροποίηση/ρύθμιση του υλικού (Hardware), για την εγκατάσταση του αναγκαίου λογισμικού και των αναγκαίων εργαλείων (Add-ons), και πως το υλικό αλληλοεπιδρά και συνδέεται με το λογισμικό, σε ένα συγκεκριμένο προγραμματιστικό περιβάλλον, με σκοπό την σχεδίαση και υλοποίηση ενός πλήρως λειτουργικού αυτόνομου τηλεπικοινωνιακού συστήματος SDR. Στο τρίτο κεφάλαιο αναλύονται τα τρία κύρια καθεστώτα στα μπορεί να λειτουργήσει μια αναπτυξιακή πλακέτα FPGA και δίνονται κάποια χαρακτηριστικά παραδείγματα και καθένα από αυτά τα καθεστώτα. Το μεγαλύτερο βάρος δίνεται στην ανάλυση, σχεδίαση ενός αυτόνομου τηλεπικοινωνιακού συστήματος SDR, που είναι και το θέμα τις παρούσας διπλωματικής εργασίας. Τέλος συνοψίζονται όλα αυτά τα κεφάλαια, σχολιάζονται τα αποτελέσματα που πάρθηκαν και προτείνονται τρόποι για την βελτίωση ολόκληρου του τηλεπικοινωνιακού SDR συστήματος.

2 ΚΕΦΑΛΑΙΟ 2ο : Συστήματα Ραδιοεπικοινωνίας Μέσω Λογισμικού - Software Defined Radio (SDR)

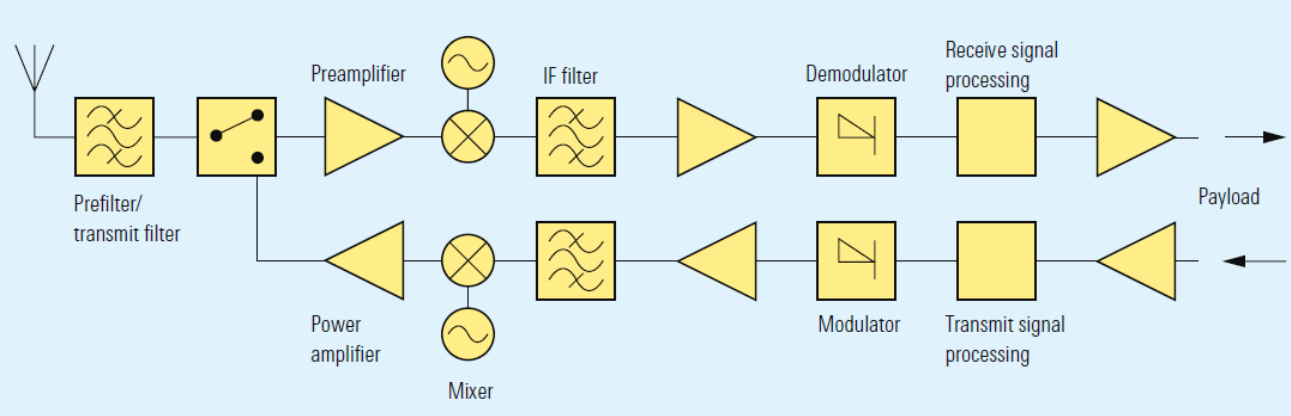
Τα συστήματα Ραδιοεπικοινωνίας (Radios) είναι βασικό μέρος της καθημερινής ανθρώπινης επικοινωνίας, είτε οι άνθρωποι το συνειδητοποιούν είτε όχι. Όταν οι περισσότεροι άνθρωποι σκέφτονται συστήματα Ραδιοεπικοινωνίας, σκέφτονται τα ραδιόφωνα AM / FM των αυτοκινήτων τους ή τα φορητά ραδιόφωνα τύπου MP3/4 τους. Ωστόσο, τα συστήματα Ραδιοεπικοινωνίας είναι κάτι πολύ περισσότερο από ότι οι περισσότεροι συνειδητοποιούν. Για παράδειγμα, οι προσαρμογείς Wi-Fi (Wi-Fi adapters) μέσα σε έναν προσωπικό υπολογιστή (PC) ή τα ακουστικά Bluetooth που χρησιμοποιούνται για συνομιλία στο κινητό τηλέφωνο, είναι συστήματα Ραδιοεπικοινωνίας. Γενικά, ένα σύστημα Ραδιοεπικοινωνίας είναι οποιαδήποτε συσκευή που μεταδίδει ή λαμβάνει πληροφορίες ασύρματα μέσω τη χρήση ηλεκτρομαγνητικών κυμάτων γνωστών ως ραδιοκύματα.
Αρχικά, τα συστήματα Ραδιοεπικοινωνίας κατασκευάζονταν με την χρήση διακριτών εξαρτημάτων (Hardware), προκειμένου να χρησιμοποιηθούν για ένα συγκεκριμένο σκοπό, για την εκτέλεση μίας συγκεκριμένης λειτουργίας. Τα συστήματα Ραδιοεπικοινωνίας αυτά, μπορούν να αναφέρονται ως Συστήματα Ραδιοεπικοινωνίας που καθορίζονται από το υλικό (Hardware Defined Radios), επειδή ,στην περίπτωση αυτή, το σύστημα εξαρτάται πλήρως από τις διατάξεις υλοποίησης του όπως ηλεκτρικά κυκλώματα και ηλεκτρονικές μονάδες ή μπλοκ (Hardware). Αντίθετα, τα Συστήματα Ραδιοεπικοινωνίας που καθορίζονται από λογισμικό (Software Defined Radios), που αναπτύχθηκαν τα τελευταία χρόνια, είναι νέας τύπος συστημάτων Ραδιοεπικοινωνίας, ο οποίος καθορίζεται από ένα τύπο λογισμικού. Αυτά τα συστήματα Ραδιοεπικοινωνίας που καθορίζονται από λογισμικό (SDR), είναι μια αναπτυσσόμενη τεχνολογία με πολλά πλεονεκτήματα που τα καθιστούν ελκυστικά για ερευνητές και προγραμματιστές που ασχολούνται με την ανάπτυξη συστημάτων Ραδιοεπικοινωνίας.
Στο κεφάλαιο αυτό θα μελετηθεί ο τομέας των συστημάτων ραδιοεπικοινωνίας μέσω λογισμικού - Software Defined Radio (SDR) και η ευελιξία που παρέχουν στην κατασκευή, τροποποίηση και δημιουργία διάφορων επιμέρους αλλά και εξ’ ολοκλήρων τηλεπικοινωνιακών υποσυστημάτων και πρωτοκόλλων.

2.1 Εισαγωγή SDR

Ένα Συστήμα Ραδιοεπικοινωνίας Μέσω Λογισμικού (SDR) είναι ένα σύστημα τηλεπικοινωνίας όπου εξαρτήματα που παραδοσιακά χρησιμοποιούνταν σε επίπεδο πλακέτας – υλικού (hardware) (π.χ. μίκτες, φίλτρα, ενισχυτές, διαμορφωτές/αποδιαμορφωτές, φωρατές, κ.λπ.) αντικαθίστανται με λογισμικό (software), μέσω της χρήσης ενός προσωπικού υπολογιστή (Personal Computer – PC) ή ενός ενσωματωμένου συστήματος. Αν και η έννοια του SDR δεν είναι καινούργια, οι ταχέως εξελισσόμενες δυνατότητες των ψηφιακών ηλεκτρονικών καθιστούν πρακτικά υλοποιήσιμες πολλές διαδικασίες που κάποτε ήταν μόνο θεωρητικά δυνατές [1]. Ο στόχος του SDR είναι η σχεδίαση ενός τηλεπικοινωνιακού συστήματος που θα μπορεί να υλοποίει, κάθε νέου πρωτοκόλλου τηλεπικοινωνίας, μόνο με την σχεδίαση και εκτέλεση λογισμικού.

2.1.1 Η σύντομη ιστορία των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού (SDR)

Οι ραδιοεπικοινωνίες υπάρχουν και επηρεάζουν την ζωή μας περισσότερο από 100 χρόνια. Όλα ξεκίνησαν με τα πειράματα του Marconi και άλλων επιστημόνων. Τα αρχικά συστήματα ραδιοεπικοινωνιών στηρίχτηκαν στην ιδέα του υπερετερόδινο δέκτη. Για δεκαετίες, τα συστήματα αυτά χρησιμοποιήθηκαν στην πλειονότητα των ράδιο-τηλεπικοινωνιακών συστημάτων. Μέχρι και πριν από λίγα χρόνια, αυτά τα αναλογικά συστήματα αποτελούσαν ότι καλύτερο μπορούσε να συναντήσει κανείς, από άποψη τεχνολογίας , στον τομέα των ράδιο-τηλεπικοινωνίων.



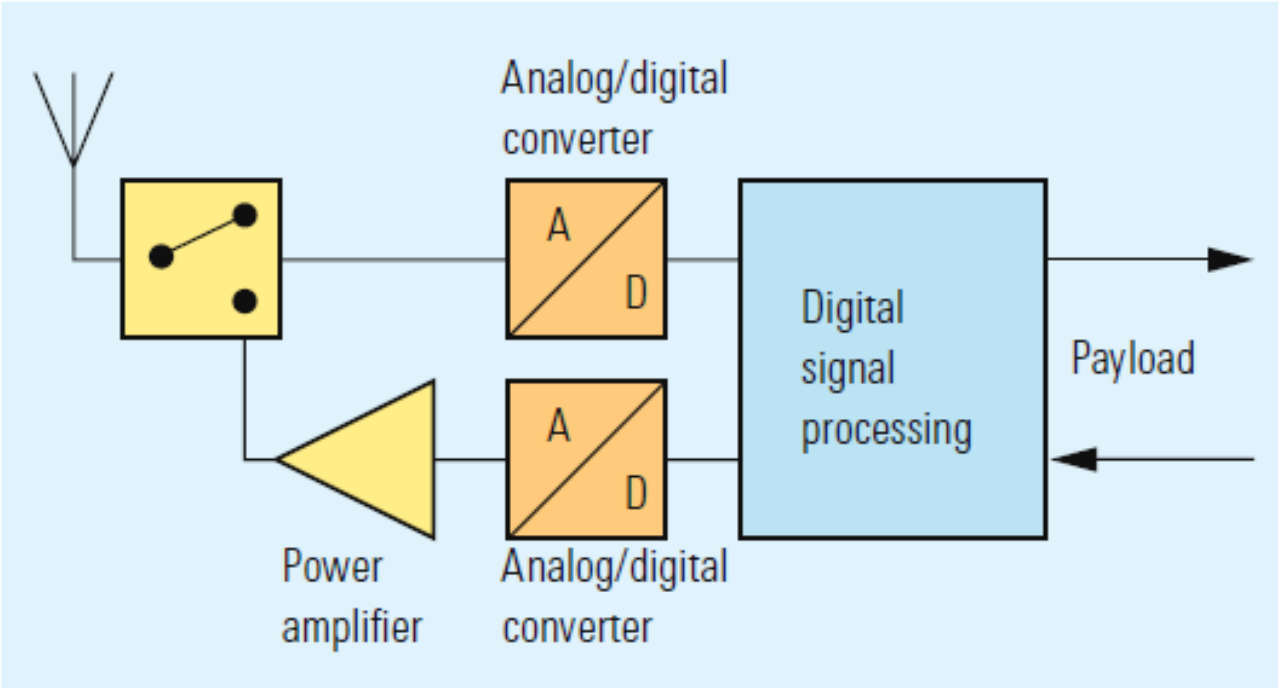
Εικόνα 2.1.1.1 Μπλοκ διάγραμμα αναλογικού ράδιο-τηλεπικοινωνιακού συστήματος

Η ψηφιακή τεχνολογία βρήκε αρχικά το δρόμο της στις τηλεπικοινωνίες στις αρχές της δεκαετίας του 1992, όταν ο Joseph Mitola πρότεινε τη βασική ιδέα των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού – Software Defined Radio (SDR). Τα πρώτα εξαρτήματα που άρχισαν να ελέγχονται ψηφιακά ήταν κυρίως οι μονάδες επεξεργασίας σημάτων ήχου, οι διαμορφωτές, οι αποδιαμορφωτές, τα φίλτρα και οι μικτές. Σήμερα, οι όλες κύριες λειτουργίες των τηλεπικοινωνιακών συστημάτων ελέγχονται από λογισμικό.

Ένα «ιδανικό» SDR περιέχει ένα μικρό ποσοστό αναλογικών μερών(front-end). Όπως είναι γνωστό στους τηλεπικοινωνιακούς διαύλους δεν μεταδίδονται ψηφιακά σήματα αλλά αναλογικά Μία κεραία και έναν μεγάλης ταχύτητας (GHz) δειγματολήπτη (ADC), ο οποίος θα είναι ικανός να λαμβάνει και να δειγματοληπτεί ένα μεγάλό εύρος φάσματος συχνοτήτων, είναι απαραίτητα για την μετατροπή των αναλογικών σημάτων σε ψηφιακά. Από την μεριά του δέκτη, οποιαδήποτε αποδιαμόρφωση, συγχρονισμός, αποκωδικοποίηση απαραίτητη για την ανάκτηση της πληροφορίας που περιέχεται στο λαμβανόμενο σήμα, πραγματοποιείται με την χρήση λογισμικού, το οποίο εκτελείται σε μία αφιερωμένη για αυτόν το λόγο υπολογιστική συσκευή(PC, ενσωματωμένο επεξεργαστή, συστοιχία επιτόπια προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA)).

Η αντίθεση διαδικασία ακολουθείται προς την μεριά του πομπού. Η πληροφορία κωδικοποιείται, διαμορφώνεται και ακολουθεί η μετατροπή των ψηφιακών σημάτων σε αναλογικών, με την χρήση

κατάλληλων μετατροπέων (Digital to Analog Converters – DAC) , καθώς στους τηλεπικοινωνιακούς διαύλους, δεν μεταδίδονται ψηφιακά σήματα αλλά αναλογικά [2].



Εικόνα 2.1.1.2 Μπλοκ διάγραμμα ενός «ιδανικού» SDR

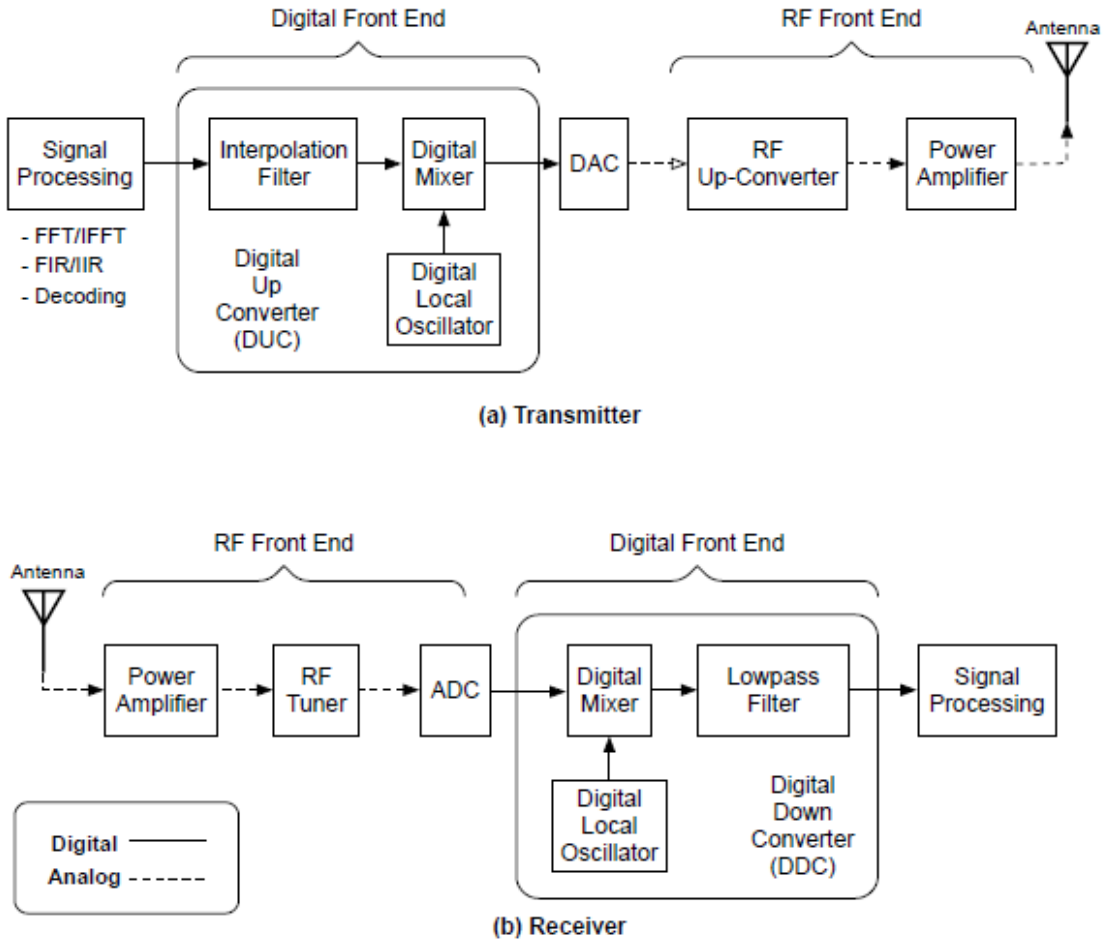
Ένα Συστήμα SDR είναι ένα τηλεπικοινωνιακό σύστημα, στο οποίο τα χαρακτηριστικά της συχνότητας φορέα, του εύρους ζώνης σήματος, της διαμόρφωσης, καθορίζονται μέσω λογισμικού. Ένα σύγχρονο σύστημα SDR εφαρμόζει επίσης ,οποιαδήποτε απαραίτητη κρυπτογραφία, κωδικοποίηση διόρθωσης σφαλμάτων και κωδικοποίηση πηγής, φωνής ή δεδομένων με την χρήση λογισμικού [3].

2.2 Υλοποίηση

Όπως εξηγήθηκε προηγουμένως, ένα σύστημα SDR αποτελείται από μια διεπαφή υλικού (Hardware front-end) και το κομμάτι του λογισμικού (software). Κατά συνέπεια, η ανάπτυξη ενός συστήματος SDR αποτελεί τον επιτυχημένο συνδυασμό μίας μονάδας υλικού (front-end) και την κατάλληλη ανάπτυξη λογισμικού (software). Σε πολλές περιπτώσεις, το ίδιο hardware είναι συμβατό με πολλαπλά ,διαφορετικά πακέτα λογισμικού SDR. Ωστόσο, τα πακέτα λογισμικού SDR δημιουργούνται συχνά ειδικά για χρήση με μια συγκεκριμένη μονάδα Hardware Front-end.
Σε αυτήν την ενότητα, θα εξεταστεί η γενική αρχιτεκτονική του SDR, τα κύρια συστατικά τους και οι απαιτήσεις επεξεργασίας τους. Τα SDR παίζουν ζωτικής σημασίας ρόλο στην ανάπτυξη νέων πρωτόκολλων επικοινωνίας, λόγω της ευελιξίας τους και την ευκολία προγραμματισμού τους. Αυτό οφείλεται κυρίως στο γεγονός ότι οι λειτουργίες της Ψηφιακής Επεξεργασίας Σήματος (Digital Signal Processing -DSP) και το ψηφιακό Front-end, τα οποία περιλαμβάνουν την επιλογή καναλιού και την διαμόρφωση/αποδιαμόρφωση, πραγματοποιούνται στο ψηφιακό κομμάτι του SDR. Αυτές οι λειτουργίες υλοποιούνται με την εκτέλεση ενός προγράμματος σε επεξεργαστές όπως DSPs ή επεξεργαστές γενικής χρήσης - General Purpose Processors – GPPs καθώς και σε προγραμματιζόμενο υλικό όπως συστοιχίες επιτόπιας προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA)).

2.2.1 Αρχιτεκτονική SDR

Στην Εικόνα 2.2.1.1 (a) και (b), παρουσιάζεται το τυπικό μπλοκ διάγραμμα ενός πομποδέκτης SDR, ο οποίος αποτελείται από τα ακόλουθα μπλοκ: Μπλοκ Επεξεργασίας Σήματος, Digital Front End, Analog RF Front End και κεραία.



Εικόνα 2.2.1.1 Μπλοκ διάγραμμα SDR. Υπόεικονα (a) πομπός SDR και (b) δέκτης SDR.

2.2.1.1 Hardware SDR

2.2.1.1.1 Κεραία.

Τα SDR, συνήθως χρησιμοποιούν κεραίες, κατάλληλα σχεδιασμένες έτσι ώστε να είναι σε θέση να καλύπτουν ένα ευρύ μέρος του ηλεκτρομαγνητικού φάσματος. Οι κεραίες αυτές, συχνά αναφέρονται ως «ευφυείς» ή «έξυπνες» κεραίες, λόγω της ικανότητας τους να συντονίζονται σε πολλές διαφορετικές μπάντες συχνοτήτων, ενώ την ίδια στιγμή, να παρουσιάζουν μεγάλη απόρριψη των παρεμβολών.

2.2.1.1.2 RF Front End.

Περιλαμβάνει το κύκλωμα RF, η κύρια λειτουργία του οποίου είναι, η μετάδοση και η λήψη του σήματος σε διαφορετικές συχνότητες. Μια άλλη λειτουργία του είναι να αλλάξει το σήμα στην/από την ενδιάμεση συχνότητα (Intermediate Frequency - IF). Η λειτουργία της βαθμίδας αυτής χωρίζεται σε δύο, ανάλογα με την κατεύθυνση του σήματος (δηλαδή, Λειτουργία Εκπομπής Tx ή Λήψης Rx):

- Στη διαδρομή της εκπομπής, τα ψηφιακά δείγματα μετατρέπονται σε αναλογικό σήμα από το DAC, το οποίο με τη σειρά του τροφοδοτεί το RF Front-end. Αυτό το αναλογικό σήμα αναμιγνύεται με μια προκαθορισμένη συχνότητα RF, διαμορφώνεται, και μετά μεταδίδεται μέσω του τηλεπικοινωνιακού καναλιού. Με το τρόπο αυτό μεταβαίνουμε από την ζώνη βασικών συχνοτήτων (Baseband) σε μία ζώνη υψηλότερων συχνοτήτων εκπομπής (Passband).
- Στη διαδρομή της λήψης, η κεραία συλλαμβάνει το σήμα RF. Η είσοδος της κεραίας συνδέεται στο RF Front-end χρησιμοποιώντας ένα κύκλωμα προσαρμογής (RF Coupler),

προκειμένου να επιτευχθεί η βέλτιστη μεταφορά ισχύος τους σήματος. Στη συνέχεια περνάει από έναν ενισχυτή χαμηλού θορύβου (Low Noise Amplifier - LNA), ο οποίος βρίσκεται πολύ κοντά την κεραία, για την ενίσχυση των αδύναμων σημάτων και την ελαχιστοποίηση του επίπεδού του θορύβου. Αυτό το ενισχυμένο σήμα, μαζί με ένα σήμα από έναν τοπικό ταλαντωτή (Local Oscillator - LO), τροφοδοτούν έναν μικτή προκειμένου να μεταβούμε από την ζώνη υψηλών συχνοτήτων (Passband) , στην ζώνη χαμηλών συχνοτήτων (Baseband) του αρχικού σήματος.

2.2.1.1.3 Αναλογική σε ψηφιακή (ADC) και ψηφιακή σε αναλογική μετατροπή (DAC).

Ο DAC, είναι υπεύθυνος για την παραγωγή του αναλογικού σήματος, που θα μεταδοθεί, από τα παραγόμενα ψηφιακά δείγματα. Στην πλευρά του δέκτη, ο ADC είναι υπεύθυνός για μετατροπή του σήματος, το οποίο είναι συνεχές στο πεδίο του χρόνου, σε διακριτά, δυαδικά κωδικοποιημένα σήματα. Η απόδοση του ADC εξαρτάται από διάφορες παραμέτρους:

(i) Σηματοθορυβική αναλογία, Signal-to-Noise Ratio (SNR): η αναλογία της ισχύς του σήματος ως προς την ισχύ του θορύβου στην έξοδο,

(ii) Ανάλυση: τον αριθμό των παραγόμενων bit ανά δείγμα

(iii) Δυναμικό εύρος χωρίς αλλοίωση (Spurious-free Dynamic Range - SFDR): ο λόγος της ισχύος του του φέροντος ως προς το επόμενο ισχυρότερο σήμα θορύβου ή αρμονικής παρεμβολής [5], και

(iv) Διάχυση ισχύος.

Η πρόοδος που έχει επιτευχθεί στην ανάπτυξη συστημάτων SDR, έχει δώσει ορμή για περαιτέρω βελτιώσεις στην απόδοση των ADC/DAC μετατροπέων. Καθώς η ενέργεια που καταναλώνει ένας ADC/DAC μετατροπέας, επηρεάζει τη διάρκεια ζωής ενός συστήματος SDR με μπαταρία, έχει γίνει σημαντική πρόοδος στην ανάπτυξη ενεργειακά αποδοτικότερων μετατροπέων με σκοπό την αύξηση της αυτονομίας του.

2.2.1.1.4 Ψηφιακό Front - End.

Το Ψηφιακό Front-end έχει δύο κύριες λειτουργίες:

- Ρυθμός μετατροπής δειγμάτων - Sample Rate Conversion (SRC), είναι η διαδικασία αλλαγής του ρυθμού δειγματοληψίας ενός διακριτού σήματος για να ληφθεί μια νέα διακριτή αναπαράσταση του υποκείμενου συνεχούς σήματος. Αυτό είναι απαραίτητο καθώς ο πομπός και ο δέκτης θα πρέπει να συγχρονιστούν προκειμένου να επιτευχθεί επικοινωνία μεταξύ τους[6].
- Καναλοποίηση - Channelization, η διαδικασία κατά την οποία, το τηλεπικοινωνιακό κανάλι, μέσω του οποίου πραγματοποιείται η επικοινωνία μεταξύ πομπού και δέκτη, διαχωρίζεται με κάποιον τρόπο έτσι ώστε να μπορούν να αποσταλούν πάνω από ένα μηνύματα. Οι τυπικές μέθοδοι καναλιοποίησης περιλαμβάνουν πακετοποίηση (Packetizing), πολυπλεξία διαίρεσης συχνότητας (Frequency Division Multiplexing) και πολυπλεξία διαίρεσης χρόνου (Time Division Multiplexing).

Σε έναν σύστημα SDR, εκτελούνται οι ακόλουθες εργασίες στο ψηφιακό Front-end:

- Στην πλευρά του πομπού, ο Προς τα Πάνω Ψηφιακός Μετατροπέας - Digital Up Converter (DUC) μετατρέπει το βασικό σήμα (Baseband) σήμα σε IF σήμα. Ο DAC που συνδέεται με τον DUC, μετατρέπει τα ψηφιακά δείγματα της IF συχνότητας σε αναλογικό σήμα IF. Στη συνέχεια, ο αντίστοιχος RF DUC μετατροπέας, μετατρέπει το αναλογικό IF σήμα σε RF αναλογικό σήμα (Passband).
- Στην πλευρά του δέκτη, ο ADC μετατρέπει το σήμα IF σε ψηφιακά δείγματα. Αυτά τα ψηφιακά δείγματα, τροφοδοτούν το επόμενο μπλοκ, το οποίο είναι ο Προς τα Κάτω Ψηφιακός Μετατροπέας – Digital Down Converter (DDC). Ο DDC μετατροπέας περιλαμβάνει έναν ψηφιακό μίκτη και έναν αριθμητικά ελεγχόμενο ταλαντωτή. Ο DDC μετατροπέας εξάγει το ψηφιακό σήμα βασικής ζώνης (Baseband) από το ADC και μετά την επεξεργασία του από το Ψηφιακό Front-end μπλοκ του SDR, το ψηφιακό σήμα βασικής ζώνης (Baseband) προωθείται σε ψηφιακό υψηλής ταχύτητας μπλοκ επεξεργασίας σήματος.

2.2.1.1.5 Επεξεργασία σήματος – Signal Processing.

Οι λειτουργίες επεξεργασίας σημάτων, όπως η κωδικοποίηση/αποκωδικοποίηση, η διαμόρφωση/αποδιαμόρφωση, και το ανακάτεμα/αναδιάταξη των bits, πραγματοποιείται σε αυτό το μπλοκ. Η κωδικοποίηση του καναλιού χρησιμεύει ως μέσο διόρθωσης σφαλμάτων των λαμβανόμενων bits. Συγκεκριμένα, το κωδικοποιημένο σήμα περιλαμβάνει κάποια επιπλέον bits που χρησιμοποιούνται από τον αποκωδικοποιητή του δέκτη, για την ανακατασκευή του αρχικού σήματος, στην περίπτωση που το λαμβανόμενο σήμα φτάνει αλλοιωμένο. Παράδειγμα Κωδικών Διόρθωσης Σφαλμάτων, αποτελούν οι Turbo Κώδικες και ο κώδικας Ελέγχου Ισοτιμίας Χαμηλής Πυκνότητας - Low Density Parity Check (LDPC). Ο αποκωδικοποιητής αποτελεί το πιο υπολογιστικά απαιτητικό κομμάτι του μπλοκ επεξεργασίας σήματος. Το δεύτερο κομμάτι που θεωρείται εξίσου περίπλοκο υπολογιστικά και ακριβό (όσον αφορά την ισχύ) είναι ο Γρήγορος Μετασχηματισμός Φουριέ - Fast Fourier Transform (FFT) και το Ανάστροφος Γρήγορος Μετασχηματισμός Φουριέ - Inverse FFT (IFFT), ως μέρος της διαδικασίας διαμόρφωσης**[7].**

2.2.1.2 Software SDR

Το επόμενο κύριο συστατικό ενός συστήματος SDR, είναι το λειτουργικό σύστημα (Software), με την βοήθεια του οποίου θα σχεδιαστεί/υλοποιηθεί, ένας χαμηλοπερατός πομποδέκτης. Ανάλογα με την χρησιμοποιουμένη σχεδιαστική μεθοδολογία/προσέγγιση που ακολουθείται για την υλοποίηση ενός συστήματος SDR, οι κυριότερες κατηγορίες αρχιτεκτονικής λειτουργικού, είναι οι ακόλουθες:

(i) Επεξεργαστές Γενικής Χρήσης - General Purpose Processor (GPP)
(ii) Μονάδα επεξεργασίας γραφικών - Graphic Processing Unit (GPU)
(iii) Επεξεργαστές Ψηφιακού Σήματος - Digital Signal Processor (DSP)
(iv) Συστοιχίες επιτόπιας προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA)
(v) Υβριδικές Σχεδιάσεις

Τα κυριότερα κριτήρια που διαχωρίζουν τις παραπάνω σχεδιάσεις/προσεγγίσεις είναι:

- Ευελιξία και ευκολία στην επαναδιαμόρφωση του χρησιμοποιούμενου λογισμικού: Η ικανότητα τροποποίησης της χρησιμοποιούμενης διαμόρφωσης ,του αλγόριθμου και του πρωτόκολλου, απλώς φορτώνοντας νέο λογισμικό στην πλατφόρμα.
- Ικανότητα προσαρμογής. Το σύστημα SDR θα πρέπει να είναι σε θέση να προσαρμόζεται εύκολα στις ιδιαιτερότητες και ανάγκες του κάθε δικτύου, καθώς οι απαιτήσεις λειτουργίας του αλλάζουν.
- Υπολογιστική ισχύς. Ο ρυθμός επεξεργασίας ενός συστήματος SDR, η οποία μετριέται σε εκατομμύρια εκτελούμενες λειτουργίες ανά δευτερόλεπτο - Giga Operations per Second (GOPS).
- Ενεργειακής απόδοσης. Η συνολική κατανάλωση ισχύος (συνήθως της τάξης μερικών εκατοντάδων milli watt), ειδικά για κινητές συσκευές στις οποίες εκτελούνται εφαρμογές Διαδίκτυο των Πραγμάτων – Internet of Things (IoT).
- Κόστος. Το συνολικό κόστος ενός συστήματος SDR, συμπεριλαμβανομένου του κόστους ανάπτυξης, αγοράς του υλικού και του χρόνου. που η εφαρμογή φτάνει ολοκληρωμένη στην αγορά.

2.2.1.2.1 Επεξεργαστές Γενικής Χρήσης - General Purpose Processor (GPP)

Μία από τις πρώτες προσεγγίσεις για την υλοποίηση συστημάτων SDR είναι χρησιμοποιώντας έναν επεξεργαστή γενικού σκοπού (GPP), ή όπως είναι γενικά γνωστοί οι μικροεπεξεργαστές υπολογιστών x86/64 και Αρχιτεκτονικής Προχωρημένη Μηχανή RISC – Advance RISC Machine (ARM) [9]. Παραδείγματα συστημάτων SDR που χρησιμοποιούν την αρχιτεκτονική GPP είναι: Microsoft Research Software Radio (Sora) [10], KUAR [11] και Universal Software Radio Peripheral (USRP) [12] .

Ένας GPP είναι ένα ψηφιακό κύκλωμα που οδηγείται από ένα ρολόι και είναι σε θέση να επεξεργάζεται διαφορετικές λειτουργίες. Οι GPP μπορούν να χρησιμοποιηθούν σε πλήθος

εφαρμογών, εξαλείφοντας την ανάγκη δημιουργίας συγκεκριμένων ηλεκτρονικών κυκλωμάτων, μειώνοντας έτσι το συνολικό κόστος λειτουργίας εφαρμογών. Οι GPP είναι το υλικό που προτιμούν αρκετοί ακαδημαϊκοί ερευνητές λόγω της ευελιξίας τους, της αφθονία των παραλλαγών τους και της ευκολίας προγραμματισμού τους, η οποία είναι μία από τις βασικές απαιτήσεις στις πλατφόρμες SDR. Επιπλέον, οι ερευνητές προτιμούν τους GPP, καθώς είναι πιο εξοικειωμένοι με τα πρωτόκολλα λειτουργίας τους, σε σύγκριση με τα DSP και FPGA. Από την άποψη της απόδοσης, οι GPP βελτιώνονται συνεχώς, όχι μόνο λόγω της τεχνολογικής πρόοδού που έχει σημειωθεί τα τελευταία χρόνια στον τομέα των CMOS, αλλά και λόγω της αύξησης του μέσου όρου των εκτελούμενων λειτουργιών ανά δευτερόλεπτο - Giga Operations per Second (GOPS) ανά κύκλο ρολογιού. Το τελευταίο επιτυγχάνεται με διάφορούς τρόπους, και ειδικότερα, αξιοποιώντας την μέθοδο του παραλληλισμού, εντός και μεταξύ των χρησιμοποιουμένων επεξεργαστών. Αυτό οδήγησε στην εξέλιξη των πολυπύρηνων GPP.

Αρχιτεκτονικά, ένας GPP περιλαμβάνει οδηγίες για διαφορετικές λειτουργίες όπως Αριθμητικές και Λογικές Μονάδες - Arithmetic and Logic Unit (ALU), μεταφορά δεδομένων και I/O. Ένας GPP επεξεργάζεται αυτές τις οδηγίες σε διαδοχική σειρά. Λόγω αυτής της διαδοχικής επεξεργασίας, οι GPP δεν είναι κατάλληλοι για την επίτευξη υψηλής υπολογιστής απόδοσης όταν καλούνται να εκτελέσουν σύνθετες εφαρμογές, με μεγάλες απαιτήσεις στο υπολογιστικό κομμάτι ,οι οποίες εκτελούνται σε πραγματικό χρόνο. Επιπλέον, τα ασύρματα πρωτόκολλα απαιτούν προβλέψιμη απόδοση προκειμένου να επιτευχθεί ο αναγκαίος συγχρονισμός μεταξύ τους. Ωστόσο, μέρος των εντολών που καλούνται να εκτελέσουν οι GPP, εκτελούνται έκτος της αναμενόμενης λογικής σειράς, γεγονός που καθιστά ανέφικτη την επίτευξη της αναγκαίας προβλεψιμότητας.

Προκειμένου να διορθωθεί αυτός ο περιορισμός των GPP, οι ερευνητές έχουν προτείνει πολλές λύσεις, μία από τις οποίες είναι η προσθήκη συν-επεξεργαστών, όπως Μονάδες Επεξεργασίας Γραφικών - Graphic Processing Unit (GPU). Οι GPU είναι επεξεργαστές ειδικά σχεδιασμένοι για να χειρίζονται εργασίες που σχετίζονται με γραφικά και επεξεργάζονται αποτελεσματικά μεγάλα τμήματα παράλληλης ροής δεδομένων. Τα συστήματα SDR που υλοποιούνται με GPP και με GPU, είναι περισσότερο ευέλικτα και έχουν υψηλότερη υπολογιστή ισχύ. Ωστόσο, αυτό οδηγεί σε χαμηλότερα επίπεδα αποδοτικότητας όσο αφορά την ισχύ. Οι GPU ενεργούν ως συν-επεξεργαστές των GPP καθώς απαιτείται ένας GPP να ενεργεί ως μονάδα ελέγχου, με σκοπό να μεταφέρει δεδομένα από μία εξωτερική μνήμη του συστήματος. Μετά την ολοκλήρωση της μεταφοράς δεδομένων από την εξωτερική μνήμη, ο GPU εκτελεί τον αναγκαίο αλγόριθμο επεξεργασίας σήματος.

Ένα άλλο πλεονέκτημα των GPU και GPP είναι η αποδοτικότητα ισχύος τους, η οποία βελτιώνεται με κάθε νέο μοντέλο που παρουσιάζεται στην αγορά. Από αρχιτεκτονική άποψη, οι GPUs έχουν πολλά πλεονεκτήματα που τα καθιστούν τις προτιμώμενες λύσεις σε εφαρμογές όπως η επεξεργασία βίντεο. Συγκεκριμένα, οι GPU χρησιμοποιούν μια μεθοδολογία που ονομάζεται Μοναδικό Πρόγραμμα Πολλαπλά Δεδομένα - Single Program Multiple Data (SPMD) που επιτρέπει την εκτέλεση πολλαπλών ροών εντολών από το ίδιο πρόγραμμα [14].

Ωστόσο, οι GPP και GPU παρουσιάζουν και κάποια μειονέκτημα. Οι σύγχρονες πλατφόρμες GPP και GPU, όπως το Sora και το USRP, χρησιμοποιούν επιτραπέζιους υπολογιστές (PC) για να υλοποιήσουν τα συστήματα SDR. Ωστόσο, αυτές οι πλατφόρμες καταναλώνουν σημαντική ποσότητα υπολογιστικής ισχύος για την επίτευξη του στόχου τους, γεγονός που τα καθιστά ακατάλληλα για πραγματικές εφαρμογές.

Όταν χρησιμοποιούνται οι GPP και οι GPU για τον σχεδιασμό συστημάτων SDR, η μεταφορά δεδομένων μεταξύ τους, μπορεί να οδηγήσει σε σημεία συμφόρησης, γεγονός που προκαλεί την μείωση της απόδοσης του συστήματος, ειδικά σε εφαρμογές που καλούνται να εκτελεστούν σε πραγματικό χρόνο. Ωστόσο, υπάρχουν συνεχείς προσπάθειες για τη μείωση ή την εξάλειψη του χρόνου επιβάρυνσης κατά την μεταφορά των δεδομένων, εισάγοντας κατάλληλους αλγορίθμους που υλοποιούν παράλληλες εργασίες αντιγραφής δεδομένων από και προς την μνήμη. Ωστόσο, η επεξεργαστική ισχύς των μικροεπεξεργαστών βελτιώνεται συνεχώς, ισορροπώντας μεταξύ επαρκούς υπολογιστικής ισχύς, κατανάλωσης ενέργειας και κόστους.

2.2.1.2.2 Επεξεργαστές Ψηφιακού Σήματος - Digital Signal Processor (DSP)

Οι Επεξεργαστές Ψηφιακού Σήματος - Digital Signal Processor (DSP) μπορούν να θεωρηθούν ως μία ειδική περίπτωση των Επεξεργαστών Γενικής Χρήσης - General Purpose Processor (GPP).

Ο DSP είναι ένας συγκεκριμένος τύπος μικροεπεξεργαστή που έχει βελτιστοποιηθεί για την επεξεργασία ψηφιακών σημάτων. Τόσο οι DSP όσο και οι GPP είναι ικανοί να εκτελέσουν πολύπλοκες αριθμητικές λειτουργίες/εφαρμογές. Λειτουργίες όπως η διαμόρφωση/ αποδιαμόρφωση, το φιλτράρισμα και η κωδικοποίηση/αποκωδικοποίηση, συχνά χρησιμοποιούνται σε εφαρμογές που περιλαμβάνουν αναγνώριση ομιλίας, επεξεργασία εικόνας και σε τηλεπικοινωνιακά συστήματα. Ωστόσο, οι DSP είναι σε θέση να τα εφαρμόζουν πιο γρήγορα και πιο αποτελεσματικά λόγω της αρχιτεκτονικής τους (π.χ. αρχιτεκτονική τύπου RISC, παράλληλη επεξεργασία), η οποία είναι ειδικά βελτιστοποιημένη έτσι ώστε να χειρίζονται με ευκολία αριθμητικές πράξεις. Καθώς, οι DSP είναι ικανοί να επιτυγχάνουν υψηλή υπολογιστική απόδοση, καταναλώνοντας μικρά επίπεδα ενέργειας, είναι καλύτεροι για την ανάπτυξη συστημάτων SDR, σε σύγκριση με τους GPP.

Όπως έχει ήδη αναφερθεί παραπάνω, οι GPP επιτυγχάνουν μία μέση απόδοση για ένα ευρύ φάσμα εφαρμογών. Αν και αυτό το επίπεδο απόδοσης είναι επαρκές για εφαρμογές που αφορούν την έρευνα και τον ακαδημαϊκό χώρο, εάν το υπό ανάπτυξη σύστημα SDR, προορίζεται για μία εμπορική εφαρμογή, πρέπει να πληρούνται ορισμένες συγκεκριμένες απαιτήσεις απόδοσης. Για τον λόγω αυτό, σε σύγκριση με τους GPP, οι DSP είναι περισσότερο προσαρμοσμένοι για την αποτελεσματική επεξεργασία ψηφιακών σημάτων, χρησιμοποιώντας χαρακτηριστικά όπως συνδυασμένες λειτουργίες πολλαπλής συσσώρευσης (μονάδες MAC) και παραλληλισμό. Οι κατασκευαστές DSP συνήθως τα ταξινομούν σε δύο κατηγορίες: βελτιστοποιημένα όσο αφορά την απόδοση και όσο αφορά την κατανάλωση ενέργειας.

Όσον αφορά την αρχιτεκτονική των εντολών τους, οι DSP μπορούν να κατηγοριοποιηθούν σε δύο ομάδες:

(i) Αρχιτεκτονική μοναδικής εντολής - πολλαπλών δεδομένων (SIMD) [15], και

(ii) Αρχιτεκτονική Πολλαπλών εντολών - πολλαπλών δεδομένων (MIMD) [16]

Αυτή η ταξινόμηση είναι μία μέθοδος ταξινόμησης διαφόρων αρχιτεκτονικών ανάλογα με το αριθμό ταυτόχρονων εντολών και ροών δεδομένων, ως εξής:

- Ένα DSP που βασίζεται σε SIMD, μπορεί να εκτελέσει μια εντολή σε πολλαπλές ροές δεδομένων ταυτόχρονα. Ένα σύστημα SDR που χρησιμοποιεί SIMD είναι η αρχιτεκτονική SODA (Service-oriented Development of Applications).
- Αντίθετα, τα DSP που βασίζονται σε MIMD έχουν τη δυνατότητα να εκτελούν πολλές εντολές σε πολλαπλές ροές δεδομένων ανά πάσα χρονική στιγμή. Αυτό είναι ουσιαστικά μια επέκταση της αρχιτεκτονικής SIMD, όπου υπάρχουν διαφορετικές οδηγίες ή προγράμματα εκτελούνται ταυτόχρονα σε πολλούς πυρήνες.

Αν και τα DSP γενικά καταναλώνουν αρκετή ισχύ, υπάρχουν ωστόσο DSP που είναι βελτιστοποιημένα για ασύρματές υλοποιήσεις χαμηλής ισχύος. Μία μέθοδος για την μείωση της κατανάλωσης ενέργειας είναι η χρησιμοποίηση πολλαπλών διαύλων δεδομένών μνήμης (π.χ. ένα για εγγραφή και δύο για ανάγνωση). Αυτό οδηγεί σε υψηλό εύρος ζώνης μνήμης και επιτρέπει την εκτέλεση πολλαπλών εντολών, οι οποίες με τη σειρά τους οδηγούν σε λιγότερους κύκλους λειτουργίας. Αυτές οι τεχνικές, σε συνδυασμό με άλλες τεχνικές εξοικονόμησης ενέργειας και βάζοντας τμήματα ή ολόκληρο το σύστημα SDR σε κατάσταση αναστολής λειτουργίας, επιτυγχάνουν περαιτέρω μείωση στην κατανάλωση ενέργειας.

Παρόλο που τα DSP έχουν ευρέως υιοθετηθεί για υλοποιήσεις συστημάτων SDR, παρουσιάζουν κάποια μειονεκτήματα. Καθώς όλο και περισσότερες εφαρμογές καλούνται να χειριστούν υπολογιστικά απαιτητικές εργασίες, τα DSP μπορεί να είναι πλέον ανεπαρκή. Αυτό οδήγησε στην υιοθέτηση διαφορετικών αρχιτεκτονικών όπως τα FPGA ή πολυπύρηνα GPP, ή ακόμη και ένα υβριδικό συνδυασμό των δύο, για την υλοποίηση ενός συστήματος SDR. Επιπλέον, η κατανάλωση ισχύος των DSP είναι γενικά υψηλότερη από ένα αντίστοιχο FPGA, λόγω λειτουργίας τους σε υψηλότερες συχνότητες.

2.2.1.2.3 Συστοιχίες επιτόπιας προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA)

Μια άλλη προσέγγιση για την πραγματοποίηση των συστημάτων SDR είναι η χρησιμοποίηση προγραμματιζόμενων υλικών όπως οι Συστοιχίες Επιτόπιας Προγραμματιζόμενων Πυλών - Field Programmable Gate Array (FPGA).

Ένα FPGA είναι μια σειρά προγραμματιζόμενων μπλοκ λογικής, όπως μπλοκ γενικής λογικής, μπλοκ μνήμης και μπλοκ πολλαπλασιαστών. Ένα FPGA έχει την ικανότητα της εφαρμογής οποιουδήποτε σχεδίου ή λειτουργίας, με κύριο χαρακτηριστικό του την ευκολία ενημέρωσης του. Αν και τα FPGA καταναλώνουν περισσότερη ισχύς από τα Ολοκληρωμένα Κυκλώματα Συγκεκριμένης Εφαρμογής - Application-specific integrated circuit (ASIC) [17], η ευκολία στον προγραμματισμό τους είναι ο λόγος πίσω από την αυξανόμενη υιοθέτησή τους σε ένα ευρύ φάσμα εφαρμογών.

Μια άλλη μεγάλη διαφορά είναι ότι, η κατασκευή ενός ASIC είναι ακριβή και απαιτεί μερικούς μήνες, σε αντίθεση με τα FPGA, που είναι σε θέση να επαναπρογραμματιστούν γρήγορα με αισθητά μικρότερο κόστος. Τα FPGA είναι σε θέση να προσφέρουν πλεονεκτήματα, όπως απόδοση υψηλής ταχύτητας, χαμηλή κατανάλωση ισχύος και φορητότητα, σε σύγκριση με άλλους επεξεργαστές όπως τα GPP και τα DSP.

Κατά την τελευταία δεκαετία, έχουν σημειωθεί σημαντικά βήματα στην εξέλιξη των FPGA και ειδικότερα στην επίτευξη υψηλότερης υπολογιστικής ισχύς. Επιπλέον, η διαθεσιμότητα διαφόρων εργαλείων software, κατέστησε τα FPGA ευκολότερο προσβάσιμα σε όλους. Αυτό οφείλεται στην ευρεία διαθεσιμότητα μεταγλωττιστών που έχουν τη δυνατότητα δημιουργίας κώδικα επιπέδου μεταφοράς μητρώου - Generating Register-Transfer Level (RTL) [18], όπως το Verilog [19] και η γλώσσα περιγραφής υλικού VHSIC - Very High Speed Integrated Circuit [20], που απαιτείται για την υλοποίηση μια εφαρμογής σε ένα FPGA, από υψηλού επιπέδου γλώσσες προγραμματισμού. Αυτή η διαδικασία αναφέρεται συνήθως ως σύνθεση υψηλού επιπέδου- High Level Synthesis (HLS) [21]. Παραδείγματα τέτοιων μεταγλωττιστών συμπεριλαμβάνουν, τον μεταγλωττιστή HDL Coder για το πρόγραμμα MatLab, τον μεταγλωττιστή HLS της Xilinx και τον μεταγλωττιστή Altera Nios II για την γλώσσα προγραμματισμού C / C++.

Ο μεταγλωττιστής HLS επιτρέπει στους μηχανικούς λογισμικού να σχεδιάζουν και να εκτελούν εφαρμογές, όπως SDR, σε FPGAs χρησιμοποιώντας μία οικεία γλώσσα προγραμματισμού, όπως η C/C ++, η SystemC και το MatLab, για την υλοποίηση της εφαρμογής. Αυτοί οι μεταγλωττιστές μπορούν επίσης να χρησιμοποιηθούν για την βελτιστοποίηση, όσο φορά την ταχύτητα όλου ή μέρος του κώδικα που εκτελείται σε ένα GPP ή DSP. Επιπλέον, ένα FPGA μπορεί να επιτύχει υψηλή απόδοση ενώ ταυτόχρονα καταναλώνει λιγότερη ενέργεια από τους επεξεργαστές που συζητήθηκαν προηγουμένως.

Μία από τις προκλήσεις της χρήσης των FPGA, είναι η γνώση που πρέπει να διαθέτει ένας προγραμματιστής, σχετικά με την αρχιτεκτονική του υλικού που σκοπεύει να προγραμματίσει και πόρους που χρειάζεται, για να σχεδιάσει μια εφαρμογή αποτελεσματικά για FPGAs. Όσο αφορά την υλοποίηση ενός συστήματος SDR, ο σχεδιασμός της πλατφόρμας ήταν συνήθως η δουλειά των μηχανικών λογισμικού, και είναι μία διαδικασία που μπορεί να είναι αρκετά χρονοβόρα. Ωστόσο, η υιοθέτηση λύσεων FPGA μπορεί να καταστεί περισσότερο εφικτή μέσω της χρήση εργαλείων HLS.

2.2.1.2.4 Υβριδικές Σχεδιάσεις

Η τέταρτη προσέγγιση για την υλοποίηση των SDRs είναι οι υβριδικές σχεδιάσεις, που βασίζονται στον συνδυασμό υλικού και λογισμικού, σε μία πλατφόρμα. Αυτό αναφέρεται συνήθως ως συν-σχεδίαση ή υβριδική προσέγγιση.

Ο σχεδιασμός υλικού/λογισμικού ως έννοια υπάρχει εδώ και πάνω από μια δεκαετία, και έχει εξελιχθεί με ταχύ ρυθμό τα τελευταία χρόνια, λόγω του αυξανόμενου ενδιαφέροντος στην επίλυση προβλημάτων σχεδιασμού ολοκληρωμένων κυκλωμάτων. Αν και οι GPP γίνονται πιο ισχυροί από ποτέ με την χρήση πολυπύρηνων σχεδίων, είναι σαφές ότι για να επιτευχθεί υψηλότερη απόδοση, σε εφαρμογές που απαιτούν επεξεργασία σε πραγματικό χρόνο, οι σχεδιαστές έπρεπε να στρέψουν την προσοχή τους, σε νέες σχεδιάσεις που χρησιμοποιούν λύσεις υλικού, δηλαδή, FPGA και ASICs. Η

προσέγγιση αυτή, αφορά τη μεθοδολογία σχεδιασμού υλικού, που εκπροσωπείται από τα FPGA και τη μεθοδολογία λογισμικού, που αντιπροσωπεύονται από τους επεξεργαστές.
Καθώς οι εφαρμογές που χρησιμοποιούνται μεγαλώνουν σε πολυπλοκότητα και μέγεθος, έχει γίνει κοινή πρακτική, η σχεδίαση συστημάτων που ενσωματώνουν τόσο λογισμικό (όπως firmware και λειτουργικό σύστημα) όσο και υλικό. Αυτό έγινε εφικτό τα τελευταία χρόνια χάρη στην πρόοδο στη σύνθεση υψηλού επιπέδου και στην ανάπτυξη εργαλείων, που όχι μόνο έχουν την ικανότητα να παράγουν αποδοτικό RTL από κώδικες λογισμικού, αλλά επίσης καθορίζουν τη διεπαφή μεταξύ των δύο πλευρών. Η βιομηχανία έχει αναπτύξει μία τεράστια γκάμα από προϊόντα συν-σχεδιασμού, όπως διάφορές αναπτυξιακές πλατφόρμες που περιλαμβάνουν Ολοκληρωμένα Συστήματα σε ένα Τσιπ - System on Chip (SoC).
Εκτός από τα προαναφερθέντα πλεονεκτήματα, υπάρχουν και άλλοι λόγοι που κάνουν τον συν-σχεδιασμό περισσότερο ενδιαφέρουσα σαν σχεδίαση, όπως ο ταχύτερος χρόνος διάθεσης της σχεδίασης στην αγορά, η χαμηλότερη κατανάλωση ισχύος, η ευελιξία και η υψηλότερη ταχύτητα επεξεργασίας.
Η μεθοδολογίας του συν-σχεδιασμού, αφορά στην ουσία, την κατάτμηση του συστήματος σε συνθέσιμο υλικό και εκτελέσιμα μπλοκ λογισμικού. Αυτή η διαδικασία εξαρτάται από αυστηρά κριτήρια, που αναπτύσσονται από τον σχεδιαστή του συστήματος. Ο σχεδιαστής θα πρέπει να είναι σε θέση να γνωρίζει κάποιες πληροφορίες, όσο αφορά το αναπτυσσόμενο σύστημα, πληροφορίες που θα συμβάλουν στην καλύτερη απόφαση, για το τι θα εφαρμοστεί στο υλικό και τι θα πρέπει να διατηρηθεί στο λογισμικό.
Τα υβριδικά συστήματα SDR, όπως έχει αναφερθεί και παραπάνω, απαιτούν την χρήση τόσο μπλοκ υλικού όσο και λογισμικού. Αυτό οφείλεται στο γεγονός ότι το μέρος του ελέγχου του συστήματος συνήθως εκτελείται από έναν γενικό επεξεργαστή και οι άλλες λειτουργίες, όπως η επεξεργασία σήματος, από έναν εξειδικευμένο επεξεργαστή όπως οι DSPs, που επιταχύνεται χρησιμοποιώντας ειδικό υλικό όπως τα FPGA. Αυτή η προσέγγιση σχεδιασμού ταιριάζει άριστα με τα συστήματα SDR και μπορεί να χρησιμοποιηθεί πλήρως, για την κάλυψη ορισμένων απαιτήσεων που σχετίζονται στα ελκυστικά χαρακτηριστικά τους. Επιπλέον, μέσω προσεκτικής εφαρμογής της τεχνικής βελτιστοποίησης RTL, επιτυγχάνεται η ανάπτυξη συστημάτων αποδοτικής ισχύος, για εφαρμογές για κινητά και IoT.
Αξίζει να σημειωθεί ότι οι προμηθευτές FPGA, όπως η Intel και η Xilinx, διευρύνουν τη βάση προϊόντων τους με περισσότερα SoCs και SoCs πολλαπλών επεξεργαστών – Multi-Processor SoCs (MPSoCs), λόγω της αυξανόμενης ζήτησης για τέτοιες συσκευές.
Ένα μειονέκτημα της υιοθέτησης SoCs στα υβριδικά συστήματα είναι ότι οι τιμές τους είναι γενικά υψηλότερες, σε σύγκριση σε προηγουμένως αναφερθείσες προσεγγίσεις σχεδιασμού, λόγω της ύπαρξης πολλαπλών εξαρτημάτων στην ίδια αναπτυξιακή πλακέτα, δηλαδή, επεξεργαστή, FPGA, επιπλέον μονάδες μνήμης και εξελιγμένες διεπαφές. Μια άλλη πρόκληση, όσο αφορά την χρήση υβριδικών συστημάτων είναι η κοινόχρηστη πρόσβαση στη μνήμη, π.χ. εξωτερική μνήμη DDR, μεταξύ του επεξεργαστή και του FPGA. Ο αριθμός των λειτουργιών ανάγνωσης και εγγραφής μνήμης που εκτελείται από ένα GPP είναι υψηλότερο από αυτό των FPGA. Αυτό οφείλεται στο γεγονός ότι οι επεξεργαστές εκτελούν λειτουργίες σε καταχωρητές, ενώ τα FPGA λειτουργούν σε ενδιάμεσες μνήμες. Η πρόσβαση στη μνήμη όμως έχει ως αποτέλεσμα την επιπλέον επιβάρυνση, στην συνολική καθυστέρηση του συστήματος και στην συνολική του απόδοση. Παρόλο αυτά, έχει αναπτυχθεί μια μεθοδολογία για την πρόβλεψη του εύρους ζώνης της κοινόχρηστης μνήμης, κάνοντας την χρήση λειτουργικά ισοδύναμο λογισμικό. Αυτό επιτρέπει στους σχεδιαστές να γνωρίζουν τυχόν εμπόδια πριν από την εφαρμογή μιας σχεδίασης, σε ένα υβριδικό σύστημα.

2.2.2 Software

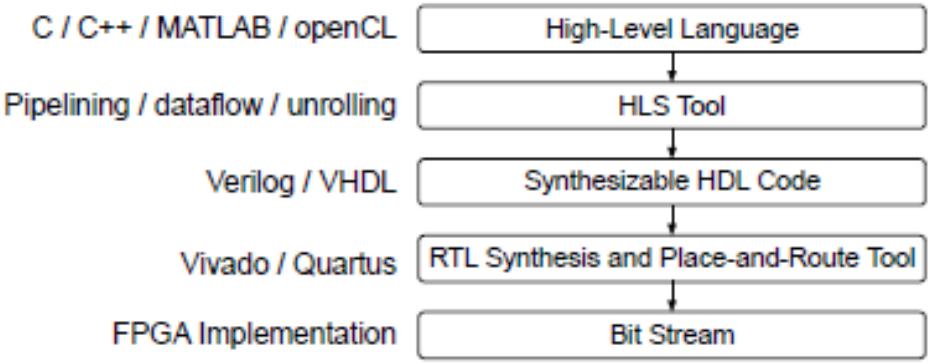
Στην ενότητα αυτή θα αναφερθούν οι κυριότεροι αλγόριθμοι σύνθεσης και τα κυριότερα αναπτυξιακά εργαλεία για την υλοποίησης ενός συστήματος SDR.

2.2.2.1 Αλγόριθμος Σύνθεσης

Στην ενότητα αυτή θα αναφερθεί ο κυριότερος αλγόριθμός Σύνθεσης Υψηλού Επιπέδου - High Level Synthesis (HLS). Ο HLS είναι μια αφηρημένη μέθοδος σχεδιασμού υλικού χρησιμοποιώντας μία γλώσσα. προγραμματισμό υψηλού επιπέδου. Οι προγραμματιστές ανάπτυξης FPGA και συστημάτων SDR, μπορεί να επωφεληθούν από τον HLS, καθώς δεν απαιτεί προηγούμενη εμπειρία στον σχεδιασμό υλικού, σε αντίθεση με τα υπόλοιπα εργαλεία ανάπτυξης.

2.2.2.1.1 Σύνθεση Υψηλού Επιπέδου - High Level Synthesis (HLS)

Η HLS είναι τη διαδικασία μετατροπής μιας αλγοριθμικής προδιαγραφής, μίας σχεδίασης που περιγράφεται από μια γλώσσα προγραμματισμού υψηλού επιπέδου σε μία εφαρμογή RTL. Η HLS παρέχει ένα νέο επίπεδο σχεδίασης μέσω της εξερεύνησης της μικρό-αρχιτεκτονικής και πολλών περιορισμών του υλικού. Το RTL που παράγεται μέσω αυτής της διαδικασίας είναι βελτιστοποιημένο στους τομής της κατανάλωσης ισχύος, της απόδοσης και της χρονικής καθυστέρησης, συγκρινόμενο με έναν κώδικα που έχει παραχθεί με τον κλασσικό τρόπο. Το σχήμα της εικόνας 2.2.2.1.1, απεικονίζει το διάγραμμα ροής αυτής της διαδικασίας [22]. Η κύρια διαφορά μεταξύ της RTL και της γλώσσας C είναι η απουσία χρονικών περιορισμών στο μοντέλο υψηλού επιπέδου, το οποίο είναι απλώς μια συμπεριφορική περιγραφή του συστήματος χωρίς λεπτομέρειες σχετικά με το υποκείμενο υλικό. Η δεύτερη διαφορά είναι η αρχιτεκτονική της διαδικασίας – Processing Architecture [23]. Ενώ η αρχιτεκτονική GPP είναι σταθερή, η αρχιτεκτονική της διαδικασίας έχει δημιουργηθεί από τον μεταγλωττιστή για το FPGA. Επιπλέον, η HLS μπορεί να επιταχύνει τον κύκλο ανάπτυξης (χρόνος στον οποίο η σχεδίαση θα καταλήξει στην διάθεση της στην αγορά) από αρκετές εβδομάδες, έως αρκετούς μήνες. Αυτό συμβαίνει επειδή το καθήκον της παραγωγής ενός βελτιστοποιημένου RTL, υλοποιείται με ευθύνη της HLS και οι αρμοδιότητες του προγραμματιστή επικεντρώνονται στην περιγραφή της αλγοριθμικής περιγραφής του συστήματος SDR.



Εικόνα 2.2.2.1.1 Διάγραμμα ροής της HLS.

Οι μεταγλωττιστές της HLS έχουν την ικανότητά να παρέχουν μία ακριβής εκτίμηση της λειτουργικής περιοχής και του συγχρονισμού του συστήματος, συγκρίσιμη σε αποτελέσματα με αντίστοιχα συστήματα που έχουν υλοποιηθεί μέσω πιο παραδοσιακών τρόπων σχεδίασης.
Παραδείγματα εργαλείων HLS περιλαμβάνουν το Xilinx Vivado HLS [24] και SDSoC [25], Intel FPGA SDK για OpenCL [26], Cadence Stratus High-level Synthesis [27], Synopsys Synphony C Compiler [28], Maxeler MaxCompiler [29], MatLab HDL Coder, και LegUP [30], που σε αντίθεση με άλλα εργαλεία είναι ανεξάρτητα από τον προμηθευτή (λειτουργούν ανεξάρτητα από τον προμηθευτή της αναπτυξιακής πλακέτας FPGA).

2.2.2.1.2 Γλώσσες Σύνθεσης Υψηλού Επιπέδου - High Level Synthesis Languages

Στο κεφάλαιο αυτό θα γίνει μια αναφορά στις κυριότερες Γλώσσες Σύνθεσης Υψηλού Επιπέδου C, C++, και SystemC.

2.2.2.1.2.1 C

Η δημοφιλής γλώσσα γενικής χρήσης C είναι μια διαδικαστική γλώσσα προγραμματισμού που αναπτύχθηκε αρχικά από την εταιρία Bell Labs και χρησιμοποιείται από τη δεκαετία του 1970. Τυποποιήθηκε από το Αμερικανικό Ινστιτούτο Εθνικών Προτύπων - American National Standards Institute (ANSI) το 1989 και στη συνέχεια υιοθετήθηκε από τον Διεθνή Οργανισμό Τυποποίησης - International Organization for Standardization ((ISO) ως πρότυπο ISO 9899. Έκτοτε, έχουν γίνει πολλές αναθεωρήσεις για να ενσωματωθούν νέα χαρακτηριστικά στη γλώσσα, τα πιο πρόσφατα από τα οποία ήταν το 2011. Οι τυποποιημένες εκδόσεις του C μπορούν να αναφέρονται ως «ANSI-C», «ISO-C» ή απλά «Standard C».
Η γλώσσα C επέτρεψε την εγγραφή προγραμμάτων όχι σε γλώσσα Assembly, αλλά χρησιμοποιώντας δομές και εντολές υψηλότερου επιπέδου. Αυτό είχε τα σημαντικά πλεονεκτήματα της απλοποίησης της εργασίας προγραμματισμού, καθώς απαιτούνται λιγότερες γραμμές κώδικα, με λιγότερες πιθανότητες σφάλματος, σε σχέση με τον αντίστοιχο κώδικα γραμμένο σε γλώσσα Assembly, επιτρέποντας ταχύτερες διαδικασίες εντοπισμού σφαλμάτων και επαλήθευσης. Έκανε επίσης τον κώδικα πιο φορητό, καθώς μπορούσε να χρησιμοποιηθεί σε διαφορετικές πλατφόρμες.
Εκτός από την Standard C, η οποία καθορίζει τη βασική λειτουργικότητα της γλώσσας, η C μπορεί να επεκταθεί συμπεριλαμβάνοντας βιβλιοθήκες για συγκεκριμένες εφαρμογές.

2.2.2.1.2.2 C++

Η γλώσσα C ++ είναι μια αντικειμενοστρεφής γλώσσα που βασίζεται στην γλώσσα C. Το επίπεδο αφαίρεσης της γλώσσας C ++ είναι συνήθως υψηλότερο από την γλώσσα C, έχει χαρακτηριστικά που κρύβουν λεπτομέρειες και επιτρέπουν την ανάπτυξη πιο περίπλοκου και ευέλικτου κώδικα. Από την άλλη πλευρά, τα χαρακτηριστικά και το στυλ προγραμματισμού της γλώσσας C είναι συμβατά με την γλώσσα C ++ και κατά συνέπεια, η C ++ μπορεί να θεωρηθεί υπερσύνολο της C. Συνοπτικά, η γλώσσα C ++ αντιπροσωπεύει μια γλώσσα υψηλότερου επιπέδου από την C, αλλά διατηρεί επίσης υποστήριξη για χαμηλού επίπεδου λειτουργίες της γλώσσας C.
Όπως και η γλώσσα C, η γλώσσα C ++ αναπτύχθηκε επίσης από την εταιρία Bell Labs. Το πρότυπο για την C ++ δημοσιεύθηκε αρχικά το 1998 ως ISO 14882 και έχει υποβληθεί σε αναθεώρηση για να παράγει μεταγενέστερες εκδόσεις.

2.2.2.1.2.3 SystemC

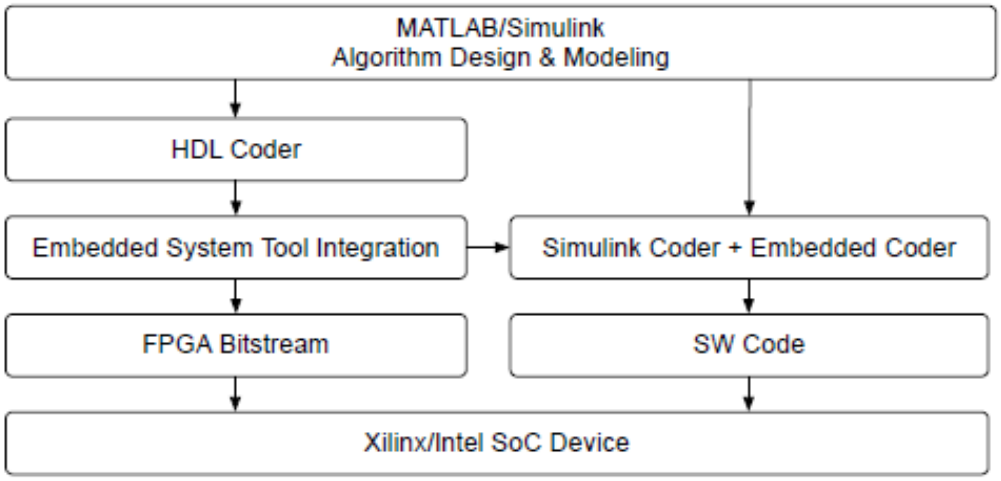
Αν και έχουμε την τάση να αντιμετωπίζουμε την SystemC ως γλώσσα από μόνο της, ωστόσο, αποτελεί επέκταση της γλώσσας C ++, δηλαδή μια συγκεκριμένη κατηγορία βιβλιοθήκης που περιλαμβάνεται σε αναπτυξιακά εργαλεία, ως το αρχείο κεφαλίδας, "systemc.h" Η SystemC επιτρέπει τη σύνταξη κώδικα τύπου C ++, χρησιμοποιώντας πολλές από τις αρχές που βασίζονται στο υλικό των HDL, οι οποίες δεν αποτελούν μέρος της Standard C ++.
Η SystemC χρησιμοποιείται συνήθως για την Μοντελοποίηση σε Επίπεδο Συναλλαγής - Transaction Level Modeling (TLM) και τον σχεδιασμός και την Επαλήθευση Επιπέδου Ηλεκτρονικού Συστήματος - Electronic System-Level (ESL), στο πλαίσιο του SoC. Το TLM είναι η ανάπτυξη περιγραφών πολύ υψηλού επιπέδου που μοντελοποιούν τις αλληλεπιδράσεις μεταξύ των στοιχείων του συστήματος, ενώ το ESL αναφέρεται στο σχεδιασμό ενός συστήματος με πολύ υψηλό επίπεδο αφαίρεσης, δηλαδή τη λειτουργικότητά του και τη βασική αρχιτεκτονική του. Ως εκ τούτου, και τα δύο αυτά αντιπροσωπεύουν μεθόδους για την ανάπτυξη και τον εξευγενισμό του υψηλού επιπέδου πλαισίου για το SoC, προτού ξεκινήσουν τις εργασίες για τη λειτουργική εφαρμογή μεμονωμένων στοιχείων. Η γλώσσα μπορεί επίσης να χρησιμοποιηθεί για επαλήθευση ESL ως ολοκληρωμένο μέρος της διαδικασίας ανάπτυξης.
Οι πρώτες εργασίες για την SystemC υποστηρίχθηκαν από την κοινοπραξία Open SystemC Initiative. Αυτό αντιπροσώπευε την αρχική φάση ανάπτυξης του SystemC. το πρότυπο SystemC επικυρώθηκε από το IEEE το 2005 ως IEEE 1666, και αργότερα αναθεωρήθηκε ως IEEE 1666-2011.

2.2.2.2 Αναπτυξιακά Εργαλεία

Σε αυτήν την ενότητα θα εξεταστούν τα υπάρχοντα εργαλεία λογισμικού για την ανάπτυξη ενός συστήματος SDR.

2.2.2.2.1 MatLab and Simulink

Οι περισσότεροι σχεδιαστές ενός συστήματος SDR, ξεκινούν με την μοντελοποίηση και την προσομοίωση του συστήματος χρησιμοποιώντας τα προγράμματα της MathWorks, MatLab και Simulink [31]. Μέσω της διαθεσιμότητας μίας ευρείας γκάμας ενσωματωμένων λειτουργιών και εργαλείων (add-ons), ειδικά για την επεξεργασία σημάτων και την επικοινωνία, ανάπτυξη και δοκιμή εφαρμογών, έγινε ένα πολύ κοινό και ευρέως υιοθετημένο λειτουργικό. Ωστόσο, για την χρησιμοποίηση αυτών των μοντέλων SDR σε διαφορετικές πλατφόρμες, οi προγραμματιστές θα χρειαστεί να χρησιμοποιήσουν τον MatLab Coder [32] και τον Simulink Coder [33] για τη δημιουργία κώδικα C/C++. Ο παραγόμενος κώδικας μπορεί να χρησιμοποιηθεί με τον ενσωματωμένο κωδικοποιητή Embedded Coder [34], για την βελτιστοποίηση και δημιουργία διεπαφών λογισμικού με προγράμματα οδήγησης Advanced eXtensible Interface (AXI) [35] προκειμένου ο κώδικας να μπορεί να εκτελεστεί σε ενσωματωμένους επεξεργαστές και μικροεπεξεργαστές. Εναλλακτικά, οι προγραμματιστές μπορούν να χρησιμοποιήσουν τον HDL Coder [36] για την δημιουργία κώδικα RTL (Verilog ή VHDL), προκειμένου να ενσωματωθεί σε FPGA ή ASIC. Επίσης, παρέχεται από την MathWorks, υποστήριξη για SoC συσκευές της Xilinx [37] και Intel, παρέχοντας πληροφορίες και βελτιστοποιήσεις, σχετικά με τη χρήση του παραγόμενου μοντέλου στις εν λόγο συσκευές. Στην εικόνα 2.2.2.2.1,παρουσιάζεται το σχέδιο ροής για πλατφόρμες SoC που προσφέρουν τα προαναφερθέντα εργαλεία και πώς συνδέονται [38].



Εικόνα 2.2.2.2.1 Διάγραμμα ροής της MathWork για SoC πλατφόρμες.

2.2.2.2.2 Vivado HLS

Το Xilinx Vivado HLS είναι ένα περιβάλλον σχεδίασης για σύνθεση υψηλού επιπέδου. Αυτό το εργαλείο προσφέρει μία ποικιλία δυνατοτήτων για την τροποποίηση και βελτιστοποίηση του παραγόμενου RTL, το οποίο είναι συμβατό και βελτιστοποιημένο για αναπτυξιακές πλακέτες FPGA της εταιρίας Xilinx. Σαν είσοδο δέχεται κώδικα γραμμένο σε πολλές γλώσσες (π.χ. C, C ++, SystemC και OpenCL) και δημιουργεί μπλοκ υλικού σε Verilog ή VHDL. Οι προγραμματιστές έχουν πολλές επιλογές για την βελτιστοποίηση της παραγόμενης λύσης, όσο αφορά τον χρονισμό, μέσω της χρήσης οδηγιών (οδηγίες για τη διαδικασία βελτιστοποίησης) για την βελτιστοποίηση του RTL.

2.2.2.2.3 SDSoC

Ένα άλλο εργαλείο της Xilinx είναι το SDSoC. Η κύρια διαφορά μεταξύ των δύο εργαλείων είναι ότι το τελευταίο έχει την ικανότητα παροχής λύσεων, συγκεκριμένα για SoCs. Το SDSoC είναι χτισμένο έχοντας σαν βάση το Vivado HLS και έχει την ίδια δυνατότητα μετατροπής της γλώσσας

C σε RTL. Το κύριο πλεονέκτημα της χρήσης του SDSoC είναι ότι δημιουργεί αυτόματα μετακινητές δεδομένων, οι οποίοι είναι υπεύθυνοι για τη μεταφορά δεδομένα μεταξύ του λογισμικού στον επεξεργαστή και του υλικού στο FPGA.

2.2.2.2.4 LegUP

Ένα παρόμοιο εργαλείο με το SDSoC είναι το ανοιχτού κώδικα LegUP[39]. Αναπτύχθηκε στο Πανεπιστήμιο του Τορόντο, ως μέρος μιας ακαδημαϊκής ερευνητικής προσπάθειας για το σχεδιασμό ενός εργαλείου HLS που είναι ικανό να λάβει έναν κωδικό C ως είσοδο και να παρέχει τρεις πιθανές εξόδους: ένας συνθέσιμος κωδικός RTL για ένα FPGA, ένα καθαρό εκτελέσιμο λογισμικό και την λύση ενός συν-σχεδιασμού υλικού/λογισμικού για ένα SoC.

2.2.2.2.5 GNU Radio

Είναι μια ανάπτυξη λογισμικού ανοιχτού κώδικα που παρέχει μπλοκ επεξεργασίας σήματος για εφαρμογές SDR [40], [41]. Τρέχει σε επιτραπέζιους ή φορητούς υπολογιστές, και με την προσθήκη απλού υλικού όπως το USRP B200 [42] και μπορεί να δημιουργήσει ένα βασικό σύστημα SDR. Χρησιμοποιείται συχνά από τον ακαδημαϊκό χώρο και την ερευνητική κοινότητα για προσομοίωση καθώς και την γρήγορη υλοποίηση πλατφορμών SDR. Η αρχή λειτουργίας του πλησιάζει αρκετά με αυτή του Simulink, καθώς περιλαμβάνει διαφορετικά είδη μπλοκ όπως αποκωδικοποιητές, αποδιαμορφωτές και φίλτρα. Επιπλέον, υποστηρίζει συστήματα USRP [43]. Ένα από τα ελκυστικά χαρακτηριστικά του ραδιοφώνου GNU είναι η δυνατότητα να οριστεί και να προστεθεί ένα νέο μπλοκ. Αυτό μπορεί να γίνει μέσω προγραμματισμού σε γλώσσα C ++ ή Python.

2.2.2.2.6 LabVIEW

Ένα ευρέως χρησιμοποιούμενο εργαλείο από την εταιρία National Instruments [44], [45], που προσφέρει ένα οπτικό περιβάλλον προγραμματισμού για την δοκιμή εφαρμογών και αυτοματισμών που χρησιμοποιούνται από την βιομηχανία και τον ακαδημαϊκό τομέα. Είναι παρόμοιο σε λειτουργεία με το GNU Radio και το Simulink, όπου ο σχεδιασμός ενός συστήματος μπορεί να πραγματοποιηθεί σχηματικά με τη σύνδεση μιας αλυσίδα διαφόρων μπλοκ, καθένα από τα οποία εκτελεί μια συγκεκριμένη συνάρτηση/λειτουργία. Προσφέρει επίσης πλήρη υποστήριξη του USRP, επιτρέποντας την ταχεία δημιουργία πρωτοτύπων συστημάτων επικοινωνιών. Ο σχεδιασμός διαφορετικών μπλοκ ενός συστήματος, μπορεί να επιτευχθεί χρησιμοποιώντας γλώσσες υψηλού επιπέδου όπως η C και το MatLab.

2.2.2.2.7 CUDA

Το εργαλείο CUDA[46], [47] αναπτύχθηκε από την εταιρεία NVIDIA. Οι προγραμματιστές συνήθως χρησιμοποιούν CUDA, όταν οι GPU αποτελούν μέρος της αρχιτεκτονικής επεξεργασίας ως συν-επεξεργαστές και θέλουν να εκμεταλλευτούν πλήρως τη δύναμή τους, επιταχύνοντας με αυτόν τον τρόπο τις εφαρμογές. Οι γλώσσες που μπορούν να χρησιμοποιηθούν στο CUDA, περιλαμβάνουν τις C, C++, Python, Fortran και MatLab. Η εργαλειοθήκη του CUDA περιλαμβάνει, μία πλούσια βιβλιοθήκη για λειτουργίες επιτάχυνσης που σχετίζονται με το GPU, έναν μεταγλωττιστή και διάφορα άλλα εργαλεία ανάπτυξης, για την ανάπτυξη και βελτιστοποίηση εφαρμογών συστημάτων που ενσωματώνουν GPUs.

2.3 Εφαρμογές

Οι κυριότεροι τομείς στους οποίους βρίσκουν εφαρμογή το SDR, παρουσιάζονται παρακάτω.

2.3.1 Κινητές επικοινωνίες

Το SDR είναι χρήσιμα στις κινητές επικοινωνίες. Με μία απλή αναβάθμιση/τροποποίηση του εφαρμοζόμενου λογισμικού, είναι δυνατή η εφαρμογή αλλαγών σε οποιαδήποτε πρότυπο, ακόμη και η προσθήκη νέων διαμορφώσεων, χωρίς την ανάγκη αλλαγών στο κόμματι του υλικό. Αυτό μπορεί να εφαρμοστεί ακόμη και εξ αποστάσεως, παρέχοντας έτσι σημαντική εξοικονόμηση κόστους.

2.3.1.1 Κατασκευή δικτύου κινητής Τηλεφωνίας

Το πρόγραμμα gr-lte [48] είναι ένα πακέτο λογισμικού ανοιχτού κώδικα που στοχεύει στην υλοποίηση ενός δέκτη GNU Radio LTE για λήψη, συγχρονισμό και αποκωδικοποίηση σημάτων LTE. Η ανάπτυξή του επικεντρώθηκε στην αρθρωτότητα. Αυτό καθιστά το πρόγραμμα εύκολο στην επέκταση και επαναχρησιμοποίηση των μπλοκ του. Αυτό το πρόγραμμα ξεκίνησε ως πτυχιακή εργασία στο Communication Engineering Lab (CEL) στο Karlsruhe Institute of Technology (KIT), Γερμανία, http://www.cel.kit.edu.
Το gr-lte παρέχει ένα αρθρωτό περιβάλλον για την υλοποίηση ενός δέκτη LTE Downlink. Όλα τα μπλοκ μπορούν να αντικαθίστανται. π.χ. να εισαχθεί ένα νέο μπλοκ εκτιμητή καναλιών με πιο προηγμένους αλγόριθμους. Από την άλλη πλευρά, είναι αρκετά εύκολο να προστεθούν νέα μπλοκ για την αποκωδικοποίηση επιπλέον καναλιών. Ο σχεδιασμός του μπλοκ είναι αρθρωτός και επαναχρησιμοποιήσιμος. Επιπλέον, οι θύρες μηνυμάτων χρησιμοποιούνται για διαμόρφωση κατά το χρόνο εκτέλεσης. Αυτό υποτίθεται ότι καθιστά τη ροή ελέγχου ορατή στους χρήστες και επομένως καλύτερα κατανοητή.

2.3.2 Έρευνα και ανάπτυξη

Το SDR είναι πολύ χρήσιμο σε πολλά ερευνητικά προγράμματα, καθώς μπορεί να διαμορφωθεί έτσι ώστε να ικανοποιεί επακριβώς τις απαιτήσεις του δέκτη και του πομπού σε οποιαδήποτε εφαρμογή ,χωρίς την ανάγκη συνολικού επανασχεδιασμού του υλικού.

2.3.2.1 Ράδιο-Αστρονομία

Ο δέκτης ραδιοαστρονομικού λογισμικού – Radio Astronomy Software Defined Receiver (RASDR) [49], [50], είναι μία εφαρμογή ανοικτού λογισμικού/υλικού, για την ανάπτυξη ενός δέκτη SDR χαμηλού κόστους και υψηλής απόδοσης. Ο στόχος του RASDR είναι να παρέχει ένα υλικό χαμηλού κόστους που επιτρέπει στους αστρονόμους και εκπαιδευτικά ιδρύματα να πραγματοποιούν αστρονομικές μετρήσεις υψηλής ποιότητας.

2.3.3 Στρατιωτική χρήση

Ο στρατός κάνει μεγάλη χρήση της τεχνολογίας SDR, επιτρέποντάς του ,την επαναχρησιμοποίηση του υλικό και την τροποποίηση των χρησιμοποιουμένων διαμορφώσεων, όπου απαιτείται

2.3.3.1 Παρακολούθηση πορείας πλοίων σε πραγματικό χρόνο

Το Αυτόματο Σύστημα Αναγνώρισης – Automatic Identification System (AIS)[51], είναι ένα αυτόματο σύστημα παρακολούθησης που χρησιμοποιείται για τον εντοπισμό σκαφών και την πορεία αυτών, το οποίο συμπληρώνει τα θαλάσσια ραντάρ. Υπάρχουν πολλές διαθέσιμες επιλογές για τη λήψη και την αποκωδικοποίηση δεδομένων AIS και μία που χρησιμοποιεί υλικό RTL-SDR, με βάση το ραδιόφωνο GNU. Τα μηνύματα AIS μπορούν να καταγραφούν, να αποθηκευτούν και να τροφοδοτηθούν στην υπηρεσία aprs.fi που βασίζεται στους Χάρτες Google ,για την οπτική απεικόνιση της πορείας και των χαρακτηριστικών (όνομα, εθνικότητας, τονάζ) ενός πλοίου.

2.3.3.2 Παρακολούθηση πορείας αεροσκαφών σε πραγματικό χρόνο μέσω της εκπομπής Mode S

Το Mode S[52] είναι παρόμοιο σκοπό με το AIS, αν και για αεροσκάφη. Για άλλη μια φορά το ταπεινό υλικό RTLSDR μπορεί να χρησιμοποιηθεί για τη λήψη μεταδόσεων και με την χρήση του λογισμικού Gr-Air-Modes [53], οι θέσεις των αεροσκαφών μπορούν να απεικονισθούν στο Google Earth.

2.3.4 Χρήση από την Ραδιοερασιτεχνική Κοινότητα

Οι ραδιοερασιτέχνες χρησιμοποιούν με επιτυχία την τεχνολογία SDR, λόγω της βελτιωμένης απόδοσης και ευελιξίας, που αυτή παρέχει.

2.3.4.1 Λήψη Δορυφορικών Εκπομπών

Καθημερινά καλύπτουν όλοι την επιφάνεια της Γης, πολλοί δορυφόροι καιρού της Εθνικής Υπηρεσίας Ωκεανών και Ατμόσφαιρας της Αμερικής - National Oceanic and Atmospheric Administration (NOAA) [54]. Κάθε δορυφόρος καιρού NOAA μεταδίδει ένα αυτόματο σήμα μετάδοσης εικόνας – Automatic Picture Transmission (APT)[55], το οποίο περιέχει μια ζωντανή εικόνα καιρού της περιοχής που καλύπτει εκείνη την χρονική στιγμή. Ένα rtl-sdr dongle σε συνδυασμό με μια καλή κεραία, SDRSharp και ένα πρόγραμμα αποκωδικοποίησης μπορεί να χρησιμοποιηθεί για την λήψη και εμφάνιση των ζωντανών αυτών εικόνων, πολλές φορές κατά την διάρκεια της ημέρας.

2.3.4.2 Ραδιοερασιτέχνες

Όπως είναι αναμενόμενο, οι ραδιοερασιτέχνες, κάνουν πολλή δουλειά με το SDR και υπάρχουν πολλές διαθέσιμες επιλογές, ξεκινώντας από το δημοφιλές, χαμηλού κόστους και απλό στην κατανόηση υλικό SoftRock SDR[56], έως το αρθρωτό και απίστευτα ευέλικτο υψηλής απόδοσης SDR - High Performance Software-Defined Radio (HPSDR)[57], [58]. Εκτός από τη χρήση υλικού που έχει σχεδιαστεί λαμβάνοντας υπόψη την ερασιτεχνική χρήση ασυρμάτων, είναι επίσης δυνατό να χρησιμοποιηθεί, κάτι όπως ένα δέκτη USRP ή RTL-SDR, ανάλογα με το αν υπάρχει η ανάγκη για την χρήση ενός πλήρους πομποδέκτη SDR ή μόνο λήψης, καθορίζοντας την απαιτούμενη κάλυψη συγκεκριμένης ζώνης συχνοτήτων ή το απαιτούμενο δυναμικό εύρος. Όσον αφορά το λογισμικό, υπάρχουν εφαρμογές που βασίζονται στο GNU Radio, όπως ο δημοφιλής δέκτης Gqrx[59], καθώς και πολλές άλλες εφαρμογές που βασίζονται στο πολύ πιο απλό πρόγραμμα DttSP[60].

2.3.4.3 Ψηφιακό ραδιόφωνο - Digital Radio Mondiale (DRM)

Το ψηφιακό ραδιόφωνο Digital Radio Mondiale Mondiale (DRM) [61],[62] (το Mondiale είναι ιταλικά και γαλλικά για "παγκοσμίως") είναι το καθολικό, ανοιχτά τυποποιημένο σύστημα ψηφιακής μετάδοσης για όλες τις συχνότητες μετάδοσης, συμπεριλαμβανομένων των ζωνών AM (LW, MW, SW), καθώς και των ζωνών VHF I, II - (ζώνη FM) και III. Το DRM διασφαλίζει την αποτελεσματική και πλήρη ψηφιοποίηση των χωρών που δεσμεύονται για την ανάπτυξη του ψηφιακού ραδιοφώνου. Η μεγάλη ευελιξία του DRM υποστηρίζει όλους τους τύπους αναγκών κάλυψης - από τοπικό, περιφερειακό, εθνικό και διεθνές επίπεδο. Το DRM επιτρέπει την απρόσκοπτη μετάβαση σε ψηφιακό ραδιόφωνο με την αναβάθμιση της υπάρχουσας υποδομής πομπού καθώς και με αναλογικές-ψηφιακές διαμορφώσεις ταυτόχρονης εκπομπής. Το ψηφιακό ραδιόφωνο DRM μπορεί να εξοικονομήσει έως και 80% σε κόστος ενέργειας και συντήρησης.

2.4 Πλεονεκτήματα – Μειονεκτήματα

Όπως συμβαίνει με οποιαδήποτε τεχνολογία, υπάρχουν κάποια πλεονεκτήματα και μειονεκτήματα της χρήσης συστημάτων SDR.

2.4.1 Πλεονεκτήματα

Η ικανότητα των συστημάτων SDR να αλλάζουν τη φυσική συμπεριφορά τους παρέχει πολλές πλεονεκτήματα σε σχέση με παραδοσιακά αναλογικά τηλεπικοινωνιακά συστήματα. Κατά κύριο λόγο, τα SDR μπορούν εύκολα να τροποποιηθούν και να εφαρμόσουν διαφορετικά πρωτόκολλα φυσικού επιπέδου, σε αντίθεση με παραδοσιακά αναλογικά τηλεπικοινωνιακά συστήματα. Με μία απλή επεξεργασία του εφαρμοζόμενου κώδικα, ο σχεδιαστής μπορεί να αλλάξει τη λειτουργικότητα ενός συστήματος SDR, χωρίς να είναι αναγκαία οποιαδήποτε αλλαγή σε επίπεδο υλικού.

Αυτή η προσαρμοστικότητα είναι χρήσιμη για διάφορους λόγους. Ένα σύστημα SDR, μπορεί γρήγορα να αλλάζει, έτσι ώστε να είναι σε θέση να υποστηρίξει διαφορετικά πρωτόκολλα υλικού. Αυτό θα μπορούσε να χρησιμοποιηθεί σε ένα σύστημα, που θα πρέπει να υποστηρίζει πολλά διαφορετικά πρωτόκολλα επικοινωνίας. Με αυτόν τον τρόπο, αντί να χρειάζονται διαφορετικές βαθμίδες υλικού για κάθε υποστηριζόμενο πρωτόκολλο, χρειάζεται απλώς μία βαθμίδα υλικού, με

διαφορετικό λογισμικό εγκατεστημένο σε αυτήν, για κάθε απαραίτητο πρωτόκολλο επικοινωνίας. Επιπλέον, οι προγραμματιστές είναι σε θέση να επεξεργάζονται και να ενημερώνουν γρήγορα το τηλεπικοινωνιακό σύστημα, απλώς αλλάζοντας τον εγκατεστημένο κώδικα, αντί να χρειάζεται να αναπτύξουν και να αντικαταστήσουν μία νέα μονάδα υλικού. Αυτό οδηγεί στην μείωση της φυσικής πολυπλοκότητας, του μεγέθους και του κόστους του τηλεπικοινωνιακού συστήματος και στην ανάπτυξη μίας συσκευής, που θα είναι σε θέση να εκτελεί πολλαπλές λειτουργίες.

Ένα δεύτερο πλεονέκτημα των SDR είναι ότι μπορούν να είναι φθηνότερα από τα παραδοσιακά τηλεπικοινωνιακά συστήματα, από ορισμένες απόψεις. Σε ένα παραδοσιακό τηλεπικοινωνιακό σύστημα, όποτε χρειάζεται να ενημερωθεί ή να προβούμε σε οποιαδήποτε επεξεργασία σε ένα από τα χαρακτηριστικά του (ισχύς, διαμόρφωση, ζώνη συχνοτήτων, κλπ.), πρέπει να δημιουργηθεί μια εντελώς νέα πλακέτα ηλεκτρονικού κυκλώματος, που ενδέχεται να κοστίσει αρκετά. Αντίθετα, τα SDR χρειάζονται απλώς μια ενημέρωση του εγκατεστημένου λογισμικού τους ,προκειμένου να επιτύχουμε την προσθήκη ενός νέου χαρακτηριστικού ή την βελτίωση ενός ήδη υπάρχον. Οι εταιρείες είναι σε θέση να επωφεληθούν από τη δυνατότητα γρήγορης τροποποίηση υπάρχον σχεδίων, αλλάζοντας απλώς κάποιες γραμμές κώδικα αντί να αλλάζουν κάποιο κομμάτι του ηλεκτρονικού εξοπλισμού τους. Αυτό μειώνει το κόστος, εξαλείφοντας την ανάγκη για την ανάπτυξη νέου ηλεκτρονικού εξοπλισμού κατά την αναβάθμιση των τηλεπικοινωνιακών συστημάτων. Το χαμηλότερο κόστος των συσκευών SDR, σε σύγκριση με τα παραδοσιακά τηλεπικοινωνιακά συστήματα, έχει οδηγήσει όλο και περισσότερους καταναλωτές και προγραμματιστές στην υιοθέτηση συστημάτων SDR.

Η ευκολία των δοκιμών και της εφαρμογής νέων προτύπων επικοινωνίας παρουσιάζει ένα τρίτο πλεονέκτημα των συστημάτων SDR σε σχέση με τα παραδοσιακά τηλεπικοινωνιακά συστήματα. Για την ανάπτυξη ενός νέου ασύρματου πρωτόκολλου επικοινωνίας, απαιτούνται πολλές δοκιμές για τον προσδιορισμό του πρότυπου και των προδιαγραφών του υπό ανάπτυξη πρωτοκόλλου. Για τα παραδοσιακά τηλεπικοινωνιακά συστήματα, νέα κυκλώματα πρέπει να σχεδιαστούν και να δημιουργηθούν για κάθε νέα δοκιμή. Στη συνέχεια, αν πρέπει να γίνουν επιπλέον αλλαγές, νέα ηλεκτρονικά κυκλώματα θα πρέπει να σχεδιαστούν ή να ενσωματωθούν. Αντίθετα, με τα συστήματα SDR, οι δοκιμές και η εφαρμογή νέων προτύπων είναι πολύ απλούστερη, φθηνότερη και ταχύτερη. Κατά τη δοκιμή, ο κωδικός μπορεί να αλλάξει, προκειμένου να δοκιμαστεί μια νέα προδιαγραφή. Αυτό επιτρέπει στους ερευνητές και τους προγραμματιστές να διαθέτουν ένα ισχυρό και ευέλικτο εργαλείο δοκιμής και ανάπτυξης νέων συστημάτων επικοινωνίας.

Τα συστήματα SDR παρουσιάζουν κάποιες λειτουργίες, που τους επιτρέπουν να ξεπερνούν κάποιους από τους περιορισμούς των παραδοσιακών τηλεπικοινωνιακών συστημάτων. Τα συστήματα SDR είναι σε θέση να αναλύουν το ασύρματο φάσμα σε μια περιοχή και να προσαρμόζουν τις παραμέτρους λειτουργίας τους, προκειμένου να επιτύχουν την αποτελεσματικότερη χρήση του[63].

2.4.2 Μειονεκτήματα

Το πιο κοινό επιχείρημα εναντίον των συστημάτων SDR είναι το κόστος. Το επιχείρημα αυτό είναι ιδιαίτερα σημαντικό για καταναλωτικά προϊόντα, τα οποία παράγονται σε μεγάλο αριθμό και έχουν χαμηλό κόστος παραγωγής. Τα προϊόντα αυτά είναι συνήθως μία εξαιρετικά απλή συσκευή, η οποία έχει μία και μοναδική λειτουργία, που αποκλείει την προσθήκη νέων λειτουργιών στο μέλλον. Εκατομμύρια τέτοιες πανομοιότυπες συσκευές πωλούνται κάθε χρόνο, αποσβένοντας το κόστος ανάπτυξης ενός ASIC. Το κόστος ενός ASIC είναι ανάλογο με το μέγεθος του ολοκληρωμένου, το οποίο με τη σειρά του είναι συνάρτηση της πολυπλοκότητας του. Ένα τσιπ για ένα σύστημα SDR, θα μπορούσε να χρησιμοποιηθεί και σε πολλές άλλες συσκευές, αλλά ο αυξημένος όγκος της παραγωγής του συγκεκριμένου ολοκληρωμένου, δεν θα οδηγούσε στην μείωση του κόστους του. Αντίθετα, δεδομένου ότι ένα σύστημα SDR είναι αναγκαστικά πιο πολύπλοκο από ένα παραδοσιακό σύστημα τηλεπικοινωνίας, το κόστος ενός ολοκληρωμένου για ένα σύστημα SDR θα ήταν υψηλότερο.

Το δεύτερο πιο κοινό επιχείρημα εναντίον των συστημάτων SDR είναι η αυξημένη κατανάλωση ενέργειας. Οι λόγοι που συμβάλλουν στην υψηλότερη κατανάλωση ενέργειας σε ένα σύστημα SDR

είναι η αυξημένη πολυπλοκότητα του DSP και το υψηλότερο εύρος ζώνης του RF σήματος. Η κατανάλωση ισχύος σε ένα FPGA ή GPP που χρησιμοποιείται για την εκτέλεση μιας εφαρμογή επεξεργασίας σήματος είναι υψηλότερη από ότι σε αντίστοιχα ASIC. Οι ADC, οι DAC και οι RF βαθμίδες που απαιτούνται σε ένα σύστημα SDR, καταναλώνουν αρκετή ισχύς. Η κατανάλωση ισχύος στους ADC ποικίλλει περίπου γραμμικά σε σχέση με το εύρος ζώνης, αλλά είναι λιγότερο γραμμική για τους DAC. Ένας ευρείας ζώνης ADC, συχνά απαιτεί υψηλότερο δυναμικό εύρος (περισσότερα bits) για να είναι σε θέση να απαλείφει τις παρεμβολές. Η αύξηση του δυναμικού εύρους ενός ADC, αυξάνει σημαντικά την κατανάλωση ισχύος.

Ένα άλλο επιχείρημα εναντίον του SDR είναι η πολυπλοκότητα που εισάγει σε ένα σύστημα. Η πολυπλοκότητα που εισάγεται σε ένα σύστημα καθορίζεται από τρία στοιχεία:

- *Αυξημένος χρόνος και κόστος για την εφαρμογή ενός συστήματος SDR.* Χρειάζεται περισσότερη σχεδιαστική προσπάθεια για την ανάπτυξη λογισμικού και υλικού, για την υποστήριξη πολλαπλών διαμορφώσεων σε ένα σύστημα, σε σχέση με ένα σύστημα που υποστηρίζει μόνο μία.
- *Μεγαλύτερος χρόνος καθορισμού των προδιαγραφών και απαιτήσεων ενός συστήματος SDR και ακριβότερη υλοποίηση τους.* Ένα σύστημα SDR καλείται να υποστηρίζει ένα σύνολο βασικών διαμορφώσεων αλλά και να προβλέπει την ενσωμάτωση επιπλέον μελλοντικών διαμορφώσεων. Πρέπει δηλαδή να παρέχεται κάποιο περιθώριο στους διαθέσιμους πόρους του DSP, για την υποστήριξη μελλοντικών διαμορφώσεων.
- *Αυξημένος κίνδυνος μη ολοκλήρωσης της σχεδίασης ενός συστήματος SDR.* Πρέπει να ληφθούν υπόψη τουλάχιστον δύο αιτίες μη ολοκλήρωσης της σχεδίασης ενός συστήματος SDR: η αδυναμία ολοκλήρωσης της σχεδίασης εγκαίρως και επί του καθορισμένου προϋπολογισμού, δεδομένου ότι το SDR εξακολουθεί να είναι μια σχετικά νέα τεχνολογία και επομένως είναι πιο δύσκολο να προβλεφθούν προβλήματα προγραμματισμού.

Η ουσιαστική πλήρης δοκιμή ενός συστήματος SDR είναι επίσης ένα ισχυρότερο επιχείρημα κατά των συστημάτων SDR. Είναι σαφές ότι δεν είναι δυνατό να δοκιμαστεί κάθε συνδυασμός υποστηριζόμενων παραμέτρων ενός συστήματος SDR (διαμορφώσεις, κώδικες, ρυθμοί μετάδοσης δεδομένων κ.λπ.). Πιθανά σφάλμα στη σχεδίαση ενός συστήματος SDR, μπορούν να επηρεάσουν όχι μόνο το ίδιο το σύστημα, αλλά και ολόκληρο το υποστηριζόμενο δίκτυο. Στην πραγματικότητα, ένα ευρείας ζώνη σύστημα SDR θα μπορούσε πιθανώς να μεταδώσει ένα σήμα σε μια μη εξουσιοδοτημένη ζώνη, μπλοκάροντας έτσι άλλα σημαντικά, για άλλες εφαρμογές, σήματα. Η δοκιμή ενός SDR σε σταθερή κατάσταση λειτουργίας, όταν δηλαδή, έχει επιλεγεί μια συγκεκριμένη διαμόρφωση, δεν επαρκεί. Τα πιθανά προβλήματα ή σφάλματα, μπορούν να παρατηρηθούν μόνο κατά τη μετάβαση από μια συγκεκριμένη διαμόρφωση, σε μία άλλη [64].

3 ΚΕΦΑΛΑΙΟ 3ο : Αναπτυξιακή πλακέτα και λογισμικό υλοποίησης SDR.

Στο κεφάλαιο αυτό θα αναφερθούν τα χαρακτηριστικά του εξοπλισμού και του λογισμικού, που χρησιμοποιήθηκε για την υλοποίηση του συστήματος SDR της παρούσας διπλωματικής εργασίας, οι αρχιτεκτονικές κατασκευής των τμημάτων που το απαρτίζουν, τα χαρακτηριστικά με τις δυνατότητες που προσφέρει, καθώς και οι αναγκαίες ρυθμίσεις/παραμετροποιήσεις, τόσο του υλικού (Hardware) όσο και του λογισμικού (Software) αυτής. Στο τέλος του κεφαλαίου αυτού, παρουσιάζεται ένας αναλυτικός οδηγός για την διασύνδεση του εξοπλισμού με τον ηλεκτρονικό υπολογιστή και η επικοινωνία αυτού, με το προγραμματιστικό περιβάλλον του MatLab/Simulink.

3.1 Αναπτυξιακή πλακέτα ZedBoard

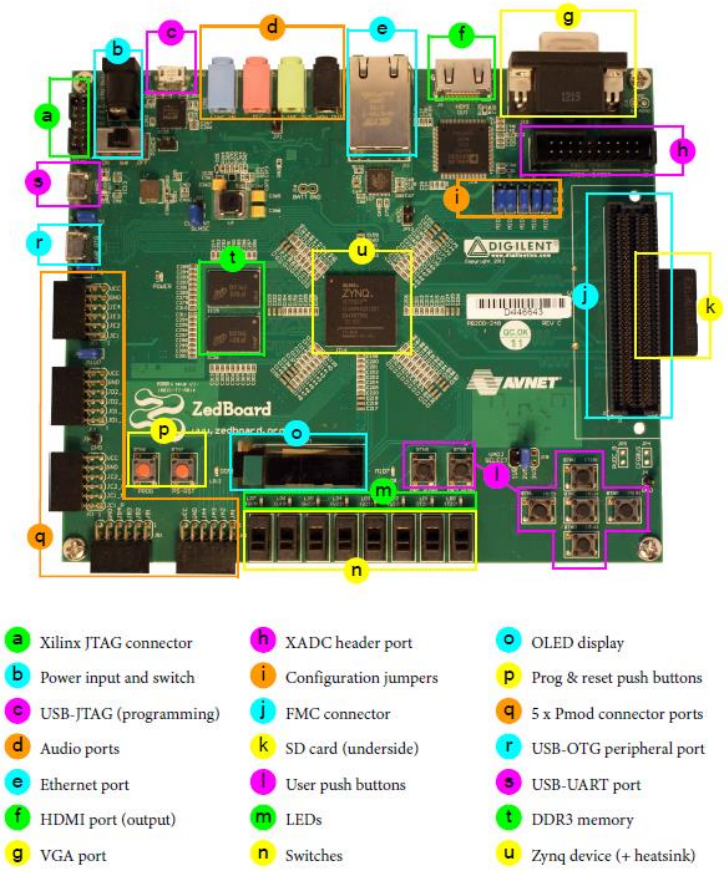
Η αναπτυξιακή πλακέτα ZedBoard είναι μια πλακέτα χαμηλού κόστους, που βασίζεται στον επεξεργαστή XC7Z020 Zynq. Είναι προϊόν της κοινοπραξίας μεταξύ της Xilinx, της Avnet [65] (του διανομέα) και της Digilent [66](του κατασκευαστή της πλακέτα). Αν και προορίζεται κυρίως ως πλατφόρμα ανάπτυξης για τη βιομηχανία, η ZedBoard απευθύνεται επίσης σε φοιτητές, ακαδημαϊκούς και χομπίστες. Ένας ιστότοπος που διαχειρίζεται η Avnet, είναι αφιερωμένος στην υποστήριξη χρηστών (ZedBoard.org)[67], μέσω της ανταλλαγής απόψεως, ιδεών και εφαρμογών. Αυτές οι εφαρμογές μπορεί είτε να είναι ολοκληρωμένα σχέδια που θα ήθελαν να μοιραστούν οι προγραμματιστές ή ελλιπή συστήματα, στα οποία θα ήθελαν, να ζητήσουν βοήθεια ή να ενθαρρύνουν τη συμμετοχή της κοινότητας των χρηστών. Κάθε μέλος της κοινότητας αυτής, μπορεί να ζητήσει να δημοσιεύσει ένα πρότζεκτ πάνω στο οποίο εργάζεται, το οποίο περιλαμβάνει την υποβολή μιας σύντομης αίτησης, για επακόλουθο έλεγχο και έγκριση.

3.1.1 ZedBoard

Όπως έχει ήδη αναφερθεί, η ZedBoard βασίζεται στον επεξεργαστή ZC7Z020 Zynq. Πρόκειται για έναν από τους μικρότερους επεξεργαστές της σειράς Zynq-7000 της εταιρίας Xilinx, και βασίζεται στην λογική Artix-7[68]. Αυτή η πλακέτα περιέχει ότι χρειάζεται κάποιος προκειμένου να σχεδιάσει κάποια εφαρμογή, η οποία θα είναι συμβατή με Linux, Android, Windows ή και κάποιο άλλο λειτουργικό σύστημα.

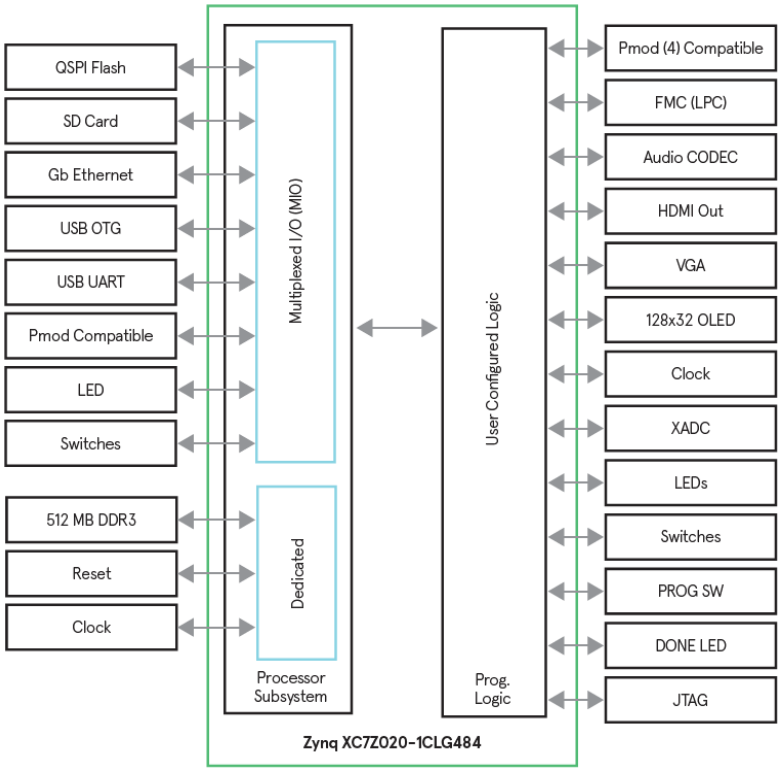
Μερικά από τα χαρακτηριστικά της πλακέτας είναι:

- Μνήμες
  512 MB DDR3
  256 Mb Quad-SPI Flash
  4 GB SD card
- Θύρα Gigabit Ethernet
- Θύρα FMC
- Οθόνη Organic Light Emitting Diode (OLED) 128x32
- Κωδικοποιητή ήχου I2S
- Θύρες Γενικής Χρήσης Εισόδου/Εξόδου - General Purpose Input/Output (GPIO): 9 x LEDs, 8 x Αναλογικοί διακόπτες, 7 x Πιεστικοί διακόπτες
- Κωδικοποιητή ήχου - Audio codec (Analog Devices ADAU1761): υποστηρίζει line in, line out, μικρόφωνο (in), και ακουστικό (out))
- Θύρα Video (HDMI)
- Θύρα Video (VGA)
- Οθόνη Organic Light Emitting Diode (OLED)
- Διεπαφή Pmod (x 5)
- Θύρα USB-JTAG (προγραμματισμός)
- Θύρα USB-UART (επικοινωνία)
- Υποδοχή κάρτας SD (εντοπίζεται στο κάτω μέρος της πλακέτας)
- Βαθμίδα Xilinx Analog Mixed Signal Module (XADC)
- Θύρα Xilinx JTAG
- Δύο ρολόγια, στα 100MHz και 33.3333MHz αντίστοιχα.



Εικόνα 3.1.1.1 Διάταξη διεπαφών πλακέτας ZedBoard [69].

Το μπλοκ διάγραμμα της πλακέτας παρουσιάζεται στην εικόνα 3.1.1.2. που ακολουθεί:



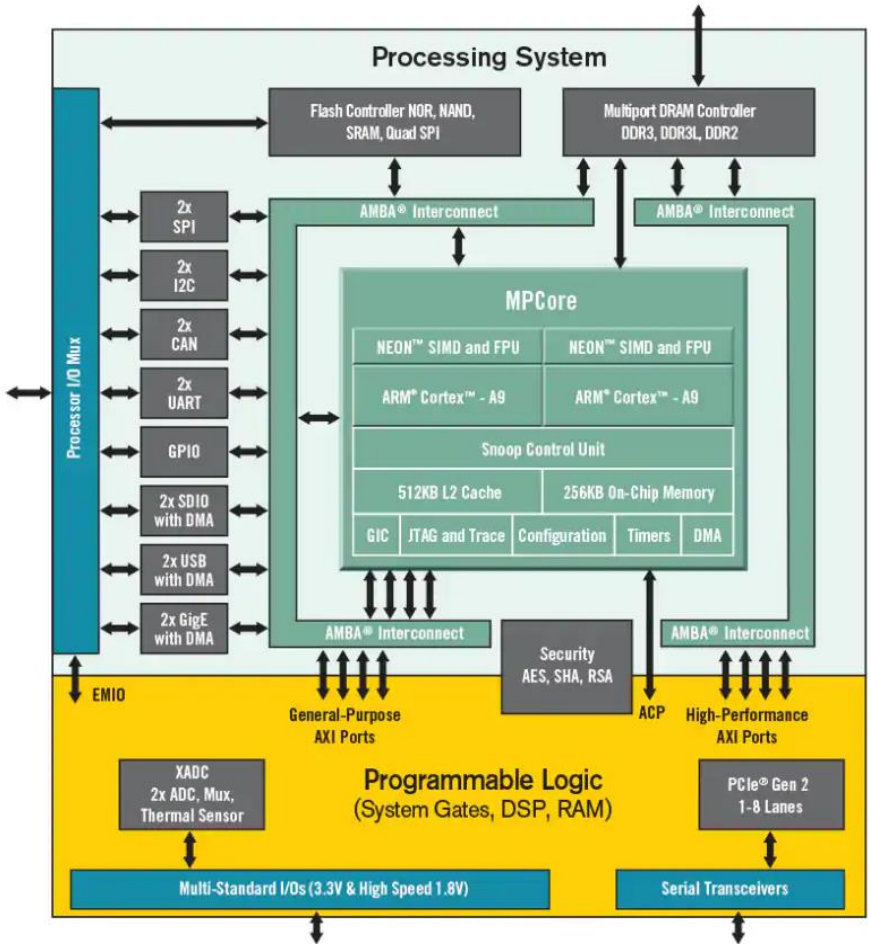
Εικόνα 3.1.1.2 Μπλοκ διάγραμμα πλακέτας ZedBoard [69].

Η αναπτυξιακή πλακέτα ZedBoard μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές μεταξύ των οποίον είναι:

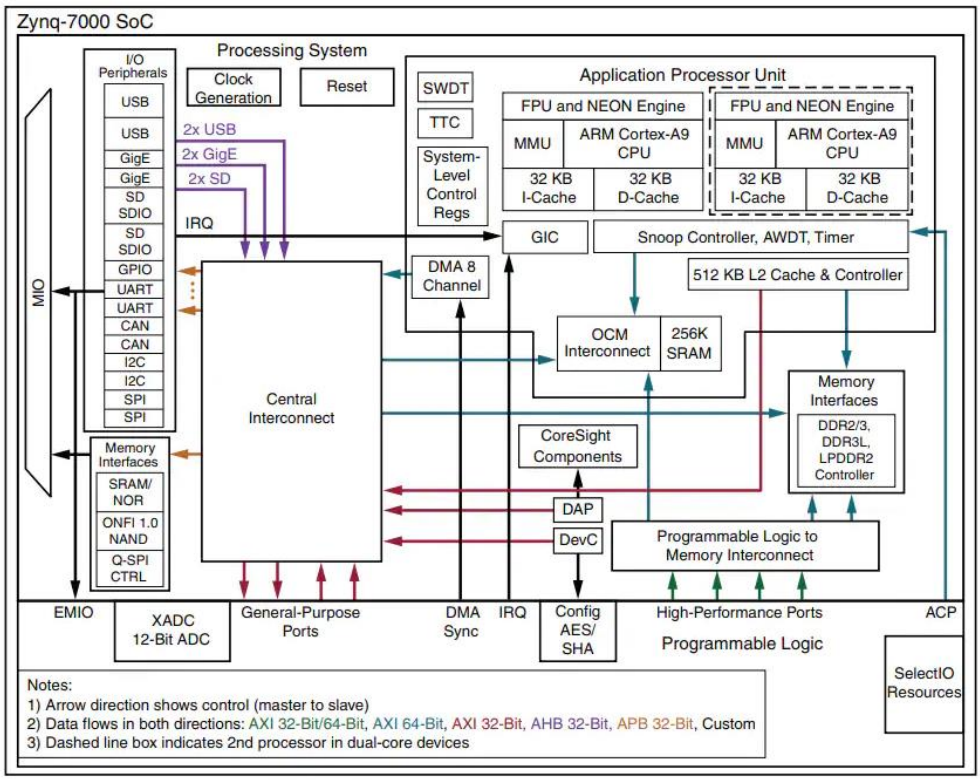
- Ακαδημαϊκή Έρευνα
- Αμυντικές εφαρμογές και διάστημα
- Ανάπτυξη λειτουργικών Linux/Android/RTOS
- Βελτιστοποίηση/επιτάχυνση λογισμικού – High Performance Computing (HPC)
- Έλεγχος μοτέρ
- Επεξεργασία Video και ήχου
- Επεξεργασία Ενσωματωμένων ARM
- Επικοινωνίες
- Ιατρική
- Προγραμματισμός και δημιουργία πρωτοτύπων SoC Zynq-7000
- Ρομποτική

3.1.2 Zynq – 7000 SoC Αρχιτεκτονική

Η Zynq – 7000 είναι μια οικογένεια ολοκληρωμένων ενιαίων συστημάτων προγραμματιζόμενων επεξεργαστών της εταιρίας Xilinx, που συνήθως χρησιμοποιείται ως πυρήνας πλακετών βασιζόμενων επάνω σε αυτή όπως είναι και η ZedBoard.
Όλες οι συσκευές Zynq έχουν την ίδια βασική αρχιτεκτονική και όλες περιέχουν, ως βάση του συστήματος επεξεργασίας τους, έναν επεξεργαστή διπλού πυρήνα ARM Cortex-A9.
Είναι σημαντικό να αναφερθεί ότι το σύστημα επεξεργασίας της Zynq δεν περιλαμβάνει μόνο τον επεξεργαστή ARM, αλλά ένα σύνολο πόρων επεξεργασίας που αποτελούν την Μονάδα Επεξεργασία Εφαρμογής – Application Processing Unit (APU) και επιπλέον κάποιων περιφερειακών διεπαφών, μνήμη cache, διεπαφές μνήμης και ρολόγια χρονισμού. Το μπλοκ διάγραμμα και η αρχιτεκτονική της οικογένειας επεξεργαστών Zynq – 7000 SoC φαίνεται στις εικόνες που ακολουθούν.

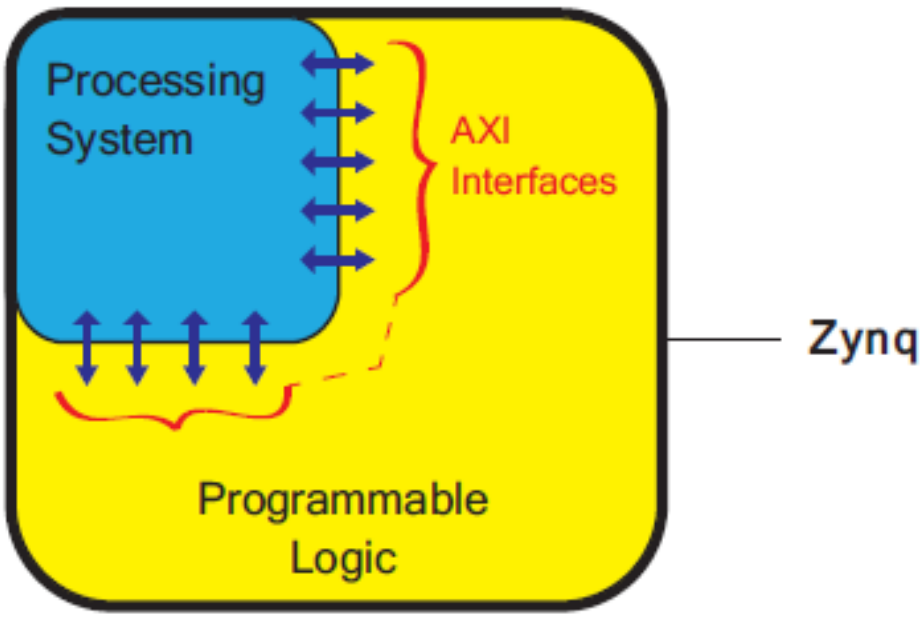


Εικόνα 3.1.1.2 Μπλοκ διάγραμμα επεξεργαστών Zynq – 7000 SoC [69].



Εικόνα 3.1.1.3. Αρχιτεκτονικής συστήματος επεξεργασίας Zynq – 7000 SoC [69].

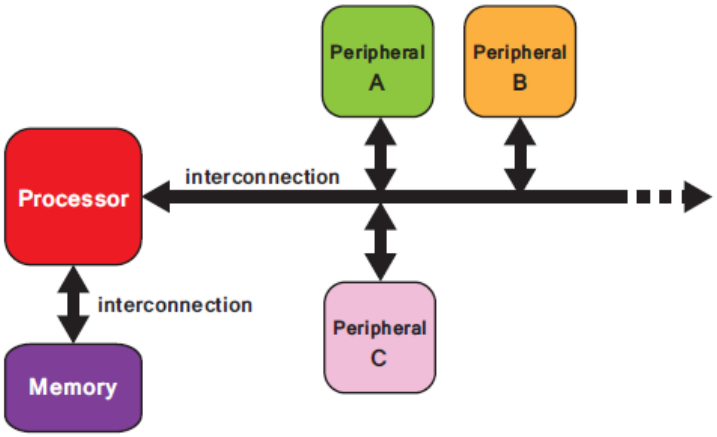
Η αρχιτεκτονική της οικογένεια Zynq, περιλαμβάνει δύο κύρια μέρη: ένα Σύστημα Επεξεργασίας - Processing System (PS), που υλοποιείται γύρω από έναν επεξεργαστή ARM Cortex-A9 διπλού πυρήνα και την Προγραμματιζόμενη Λογική - Logic Programmable (PL), η οποία είναι ισοδύναμη με αυτήν ενός FPGA. Διαθέτει επίσης ενσωματωμένη μνήμη, μια ποικιλία περιφερειακών και κάποιες διεπαφές επικοινωνίας υψηλής ταχύτητας.
Η PL είναι ιδανική για την εφαρμογή υποσυστημάτων λογικής, αριθμητικής και υψηλής ταχύτητας ροής δεδομένων, ενώ το PS υποστηρίζει ρουτίνες λογισμικού ή/και λειτουργικά συστήματα, πράγμα που σημαίνει ότι η συνολική λειτουργικότητα, οποιουδήποτε σχεδιασμένου συστήματος μπορεί να διαχωριστεί κατάλληλα μεταξύ υλικού και λογισμικού. Η επικοινωνία μεταξύ PL και PS γίνονται χρησιμοποιώντας τυπικές βιομηχανικές συνδέσεις Advanced eXtensible Interface (AXI).



Εικόνα 3.1.1.4. Απλοποιημένη Αρχιτεκτονικής Zynq – 7000 SoC [69].

3.1.3 Απλοποιημένη Αρχιτεκτονική SoC

Ο επεξεργαστής μπορεί να θεωρηθεί ως το κεντρικό στοιχείο του συστήματος υλικού (Hardware System). Το σύστημα λογισμικού (Software System)(ένα λογισμικό «στοίβας») εκτελείται στον επεξεργαστή, που περιλαμβάνει εφαρμογές (συνήθως βασίζονται σε λειτουργικό σύστημα (Operation System - OS)) και με ένα χαμηλότερο επίπεδο λειτουργικότητας λογισμικό για την διασύνδεση με το σύστημα υλικού. Η επικοινωνία μεταξύ των διαφόρων στοιχείων του συστήματος πραγματοποιείται μέσω διασυνδέσεων. Αυτές μπορεί να είναι άμεσοι σύνδεσμοι, από σημείο σε σημείο ή αρτηρίες, που εξυπηρετούν πολλά εξαρτήματα. Στην τελευταία περίπτωση, απαιτείται ένα πρωτόκολλο για τη διαχείριση της πρόσβασης στην αρτηρία.



Εικόνα 3.1.3.1. Απλοποιημένη Αρχιτεκτονική ενός SoC [69].

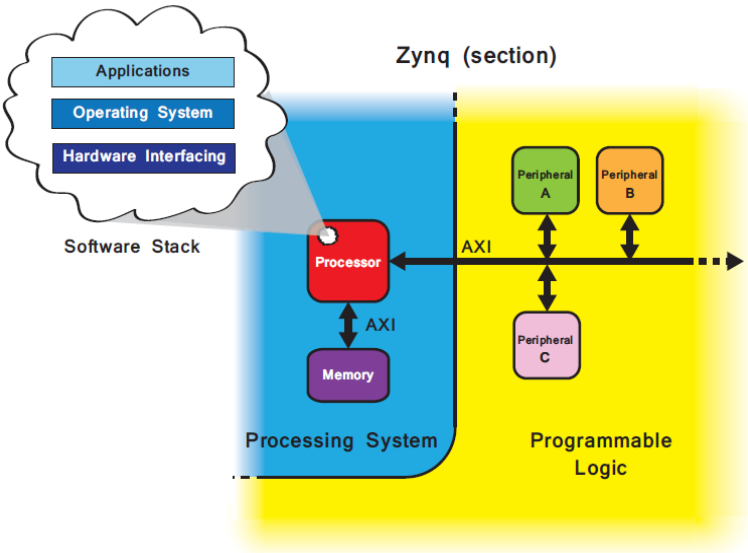
Τα περιφερειακά συστήματα είναι λειτουργικά συστατικά που βρίσκονται μακριά από τον επεξεργαστή και ανήκουν μία από τις τρεις ακόλουθες κατηγορίες:

(i) στοιχεία που συμπληρώνουν τον κύριο επεξεργαστή, συνήθως βελτιστοποιημένα για μια συγκεκριμένη εργασία.

(ii) πυρήνες για αλληλεπίδραση με εξωτερικές διεπαφές, π.χ. σύνδεση με LED και διακόπτες, κωδικοποιητές κ.λπ. και

(iii) πρόσθετα στοιχεία μνήμης.

Τα περιφερειακά, μπορεί να θεωρηθούν ως διακριτά λειτουργικά μπλοκ που μπορούν να σχεδιαστούν, να δοκιμαστούν και να ενσωματωθούν σε ένα σύστημα.

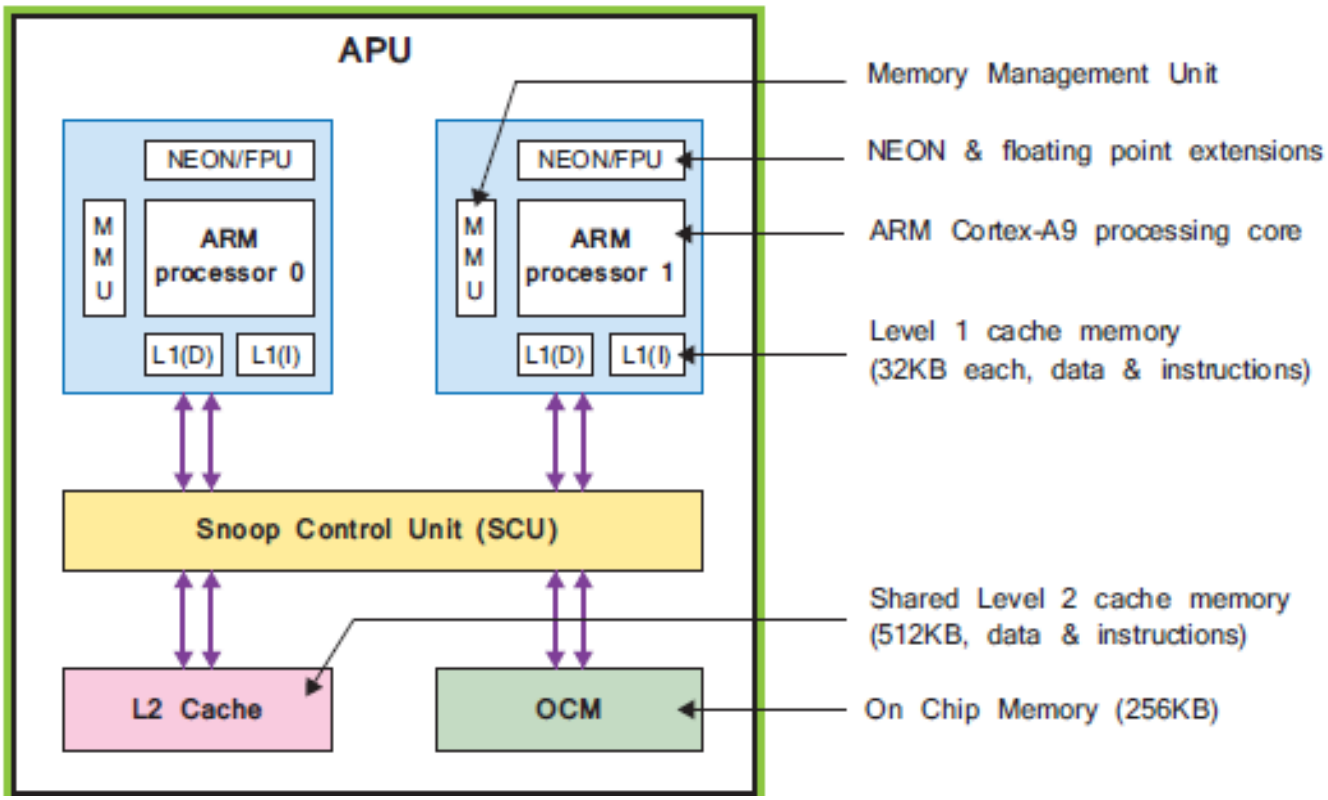


Εικόνα 3.1.3.2. Συσχέτιση μεταξύ Συστήματος Υλικού, Συστήματος Λογισμικού και Αρχιτεκτονικής Zynq [69].

Στην εικόνα 3.1.3.2 παρουσιάζεται μια όψη του συστήματος υλικού, όπως είναι χαρτογραφημένο σε μία συσκευή της οικογένειας Zynq. Οι αρχιτεκτονικές και των δύο συστημάτων (PS και PL) έχουν ουσιαστικά απλουστευθεί, αλλά παρέχουν μια υψηλού επιπέδου απεικόνιση, για το πώς ενσωματώνονται τα SoCs σε συσκευές της οικογένειας Zynq. Το PS έχει σταθερή αρχιτεκτονική και φιλοξενεί τη μνήμη του επεξεργαστή και του συστήματος, ενώ το PL είναι απόλυτα ευέλικτο, δίνοντας στον σχεδιαστή έναν «κενό καμβά» για τη δημιουργία προσαρμοσμένων περιφερειακών ή για την χρησιμοποίηση τυπικών περιφερειακών. Οι διασυνδέσεις που συνδέουν το PS και το PL, υλοποιούνται μέσω διεπαφών AXI.
Το σύστημα λογισμικού φαίνεται στην αριστερή πλευρά της εικόνας 3.1.3.2. Το λογισμικό φιλοξενείται στον επεξεργαστή, ο οποίος είναι ο ARM Cortex-A9, που βρίσκεται στο Zynq PS και περιλαμβάνει μια ιεραρχία στοιχείων λογισμικού

3.1.4 Μονάδα Επεξεργασίας Εφαρμογών - Application Processing Unit (APU)

Ένα απλοποιημένο μπλοκ διάγραμμα της Μονάδα Επεξεργασίας Εφαρμογών - Application Processing Unit (APU) φαίνεται στην εικόνα 3.1.4.1 που ακολουθεί. Η APU αποτελείται κυρίως από δύο πυρήνες επεξεργασίας ARM, ο καθένας με διάφορες υπολογιστικές μονάδες: μια Μηχανή Επεξεργασίας Πολυμέσων NEON ™ - Media Processing Engine (MPE), μία Μονάδα Κινητής Υποδιαστολής – Floating Point Unit(FPU), μια Μονάδα Διαχείρισης Μνήμης Memory Management Unit (MMU) και μία μνήμη cache επιπέδου 1 (σε δύο ενότητες για οδηγίες και δεδομένα). Η APU περιέχει επίσης μνήμη cache επιπέδου 2 και μια επιπλέον μνήμη On Chip Memory (OCM). Τέλος, μια Μονάδα Ελέγχου Επιτήρησης- Snoop Control Unit (SCU) [70], σχηματίζει μια γέφυρα μεταξύ των πυρήνων ARM και της μνήμης cache επιπέδου 2 και των OCM. Αυτή η μονάδα έχει επίσης κάποια ευθύνη για διασύνδεση με το PL.



Εικόνα 3.1.4.1. Μπλοκ διάγραμμα την APU [69].

Ο ARM Cortex-A9 μπορεί να λειτουργήσει σε συχνότητες έως 1GHz, ανάλογα με τη συσκευή της οικογενείας Zynq. Καθένας από τους δύο πυρήνες έχει ξεχωριστές προσωρινές μνήμες 32KB επιπέδου 1, για δεδομένα και οδηγίες. Οι δύο πυρήνες μοιράζονται επιπλέον μια μεγαλύτερη μνήμη cache επιπέδου 2, 512 KB για οδηγίες και δεδομένα ενώ υπάρχει και μία επιπλέον μνήμη 256 KB

on-chip εντός της APU. Ο πρωταρχικός ρόλος του MMU είναι η μετάφραση μεταξύ εικονικών και φυσικών διευθύνσεων.
Η Μονάδα Ελέγχου Επιτήρησης αναλαμβάνει πολλές εργασίες που σχετίζονται με τη διασύνδεση μεταξύ των επεξεργαστών και τις μνήμες προσωρινής μνήμης επιπέδου 1 και 2. Το SCU είναι υπεύθυνο για τη διατήρηση της συνοχής της μνήμης μεταξύ της προσωρινής μνήμης δεδομένων επεξεργαστή και της κοινόχρηστης προσωρινής μνήμης επιπέδου 2. Επίσης, η SCU διαχειρίζεται τις συναλλαγές που πραγματοποιούνται μεταξύ του PS και του PL μέσω της Θύρας Επιτάχυνσης Συνοχής - Accelerator Coherency Port (ACP).
Από πλευράς προγραμματισμού, παρέχεται υποστήριξη για τον επεξεργαστή ARM, μέσω του κιτ ανάπτυξης λογισμικού Xilinx - Software Development Kit (SDK), που περιλαμβάνει όλα τα απαραίτητα στοιχεία για την ανάπτυξη λογισμικού σε αυτόν. Ο μεταγλωττιστής υποστηρίζει τα σύνολα εντολών ARMand Thumb® (16-bit ή 32-bit), μαζί με 8-bit Java bytecodes (χρησιμοποιούνται για Java Virtual Machines).

3.1.5 Πρότυπο Διαύλου Advanced eXtensible Interface (AXI)

Το πρότυπο διαύλου Advanced eXtensible Interface και η τρέχουσα έκδοση AXI4, αποτελεί μέρος του ανοιχτού προτύπου ARM AMBA® 3.0. Πολλές συσκευές και μπλοκ IP που παράγονται από τρίτους κατασκευαστές και προγραμματιστές, βασίζονται σε αυτό το πρότυπο.
Οι δίαυλοι AXI χρησιμοποιούνται για τη σύνδεση των επεξεργαστών και άλλων μπλοκ IP, σε ένα ενσωματωμένο σύστημα. Στην πραγματικότητα υπάρχουν τρεις επιλογές του AXI4, καθεμία από τις οποίες αντιπροσωπεύει ένα διαφορετικό πρωτόκολλο διαύλου. Η επιλογή του κατάλληλου πρωτοκόλλου διαύλου AXI για μια συγκεκριμένη σύνδεση, εξαρτάται από τις επιθυμητές ιδιότητες αυτής της σύνδεσης.

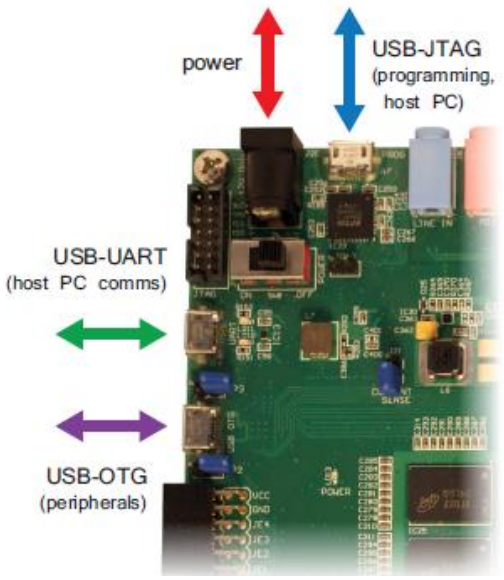
- AXI4 – Κατάλληλο για συνδέσμους που χαρτογραφούνται στην μνήμη του συστήματος (memory-mapped links) και οι οποίοι επιτυγχάνουν υψηλή απόδοση: παρέχεται μια διεύθυνση ακολουθούμενη από μία ριπή δεδομένων έως και 256 λέξεις δεδομένων (words data).
- AXI4-Lite - Ένας απλοποιημένος σύνδεσμος που υποστηρίζει μόνο μία μεταφορά δεδομένων ανά σύνδεση. Το AXI4-Lite επίσης έχουν χαρτογραφηθεί σε θέσεις μνήμης του συστήματος και αποτελούν υποσύνολο το AXI4: σε αυτήν την περίπτωση μεταφέρονται μια διεύθυνση και μία λέξη δεδομένων(word data).
- AXI4-Stream Χρησιμοποιείται σε μη χαρτογραφημένα στη μνήμη περιφερειακά, όπου απαιτείται υψηλής ταχύτητας συνεχής ροή δεδομένων. Δεν υπάρχει μηχανισμός διευθύνσεων. Αυτός ο τύπος διαύλου είναι πιο κατάλληλος για την άμεση ροή δεδομένων μεταξύ πηγής και προορισμού (δεν έχει αντιστοιχιστεί μνήμη).

Ο όρος «χαρτογράφηση μνήμης - memory-mapped» που χρησιμοποιείται στις παραπάνω περιγραφές, είναι χρήσιμο να επεξηγηθεί η σημασία του, εν συντομία. Εάν ένα πρωτόκολλο έχει χαρτογραφηθεί στην μνήμη του συστήματος, μια διεύθυνση μνήμης καθορίζεται μέσα στη συναλλαγή που εκδίδεται από τον master (ανάγνωση ή εγγραφή), η οποία αντιστοιχεί σε μια διεύθυνση στο χώρο μνήμης του συστήματος. Στην περίπτωση του AXI4-Lite, το οποίο υποστηρίζει μία μεταφορά δεδομένων ανά συναλλαγή, τα δεδομένα στη συνέχεια γράφονται ή διαβάζονται από μια καθορισμένη διεύθυνση μνήμης. Στην περίπτωση ριπών AXI4, η καθορισμένη διεύθυνση μνήμης είναι για την πρώτη λέξη δεδομένων που θα μεταφερθεί και ο slave πρέπει στη συνέχεια να υπολογίσει τις διευθύνσεις για τις λέξεις δεδομένων που ακολουθούν.

3.1.6 Επικοινωνία με την Αναπτυξιακή πλακέτα ZedBoard

Η ZedBoard πρέπει να είναι αρχικά συνδεδεμένη στο τροφοδοτικό της και για την προεπιλεγμένη διαμόρφωση, στον υπολογιστή μέσω θύρας USB (δηλ. USB-A σε micro-USB-B σύνδεση), με την θύρα JTAG (Joint Test Action Group) [71]. Μια πρόσθετη σύνδεση θα πρέπει να γίνει εάν πρόκειται να χρησιμοποιηθεί η θύρα UART (Universal Asynchronous Receiver-Transmitter) [72] για να διευκολυνθεί η απλή επικοινωνία μέσω υπολογιστή-PC. Οι θέσεις αυτών των δύο

συνδέσεων στην ZedBoard φαίνονται στην εικόνα 3.1.5.1 που ακολουθεί, όπως και μία τρίτη θύρα micro-USB που προορίζεται για τη σύνδεση περιφερειακών.



Εικόνα 3.1.5.1. Θύρες σύνδεσης της αναπτυξιακής πλακέτας ZedBoard [69].

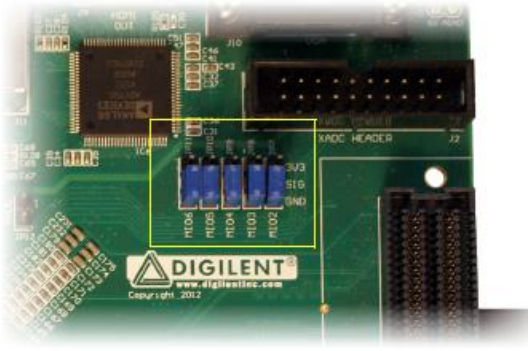
3.1.7 Προγραμματισμός ZedBoard

Η ZedBoard μπορεί να προγραμματιστεί με τέσσερις διαφορετικούς τρόπους. Αυτοί είναι οι ακόλουθοι:

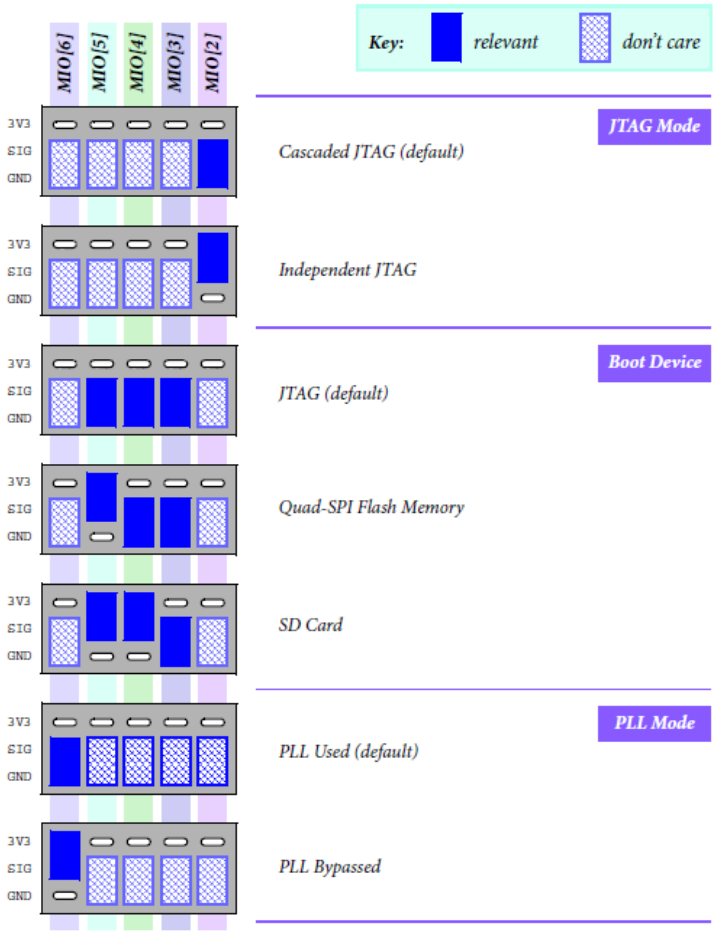
- USB-JTAG: Αυτή είναι η προεπιλεγμένη και η πιο απλή μέθοδος προγραμματισμού της ZedBoard, δεδομένου ότι μπορεί να γίνει απευθείας μέσω του καλωδίου USB-micro-USB.
- JTAG: Ένας σύνδεσμος Xilinx JTAG διατίθεται στην πλακέτα και μπορεί να χρησιμοποιηθεί αντί της σύνδεσης USB-JTAG. Αυτό απαιτεί ένα καλώδιο USB Xilinx Platform ή ένα καλώδιο προγραμματισμού Diligent USB-JTAG.
- Quad-SPI μνήμη flash: Η μνήμη flash στην πλακέτα δεν είναι πτητική (Non-Volatile) και ως εκ τούτου μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων διαμόρφωσης, που παραμένουν όταν η πλακέτα είναι απενεργοποιημένη. Η χρήση αυτής της μεθόδου καταργεί την απαίτηση για ενσύρματη σύνδεση για τον προγραμματισμό της συσκευής Zynq.
- Κάρτα SD: Υπάρχει μια υποδοχή SD στην κάτω πλευρά της ZedBoard. Αυτή η δυνατότητα μπορεί να χρησιμοποιηθεί για τον προγραμματισμό της Zynq με αρχεία που είναι αποθηκευμένα στην κάρτα SD, χωρίς να απαιτείται καμία ενσύρματη σύνδεση για προγραμματισμό.

Οι μέθοδοι JTAG είναι ιδανικές κατά την φάση ανάπτυξης μίας εφαρμογής, καθώς είναι εύκολο και βολικό, να δημιουργηθεί σύνδεση USB/JTAG μεταξύ του υπολογιστή και της ZedBoard.
Ο χρήστης της ZedBoard καθορίζει τη μέθοδο εκκίνησης/προγραμματισμού, μέσω ενός συνόλου πέντε βραχυκυκλωτήρων (jumpers), που βρίσκονται πάνω στην πλακέτα. Από τους πέντε βραχυκυκλωτήρας, οι τρεις κεντρικοί χρησιμοποιούνται για τον καθορισμό της πηγής για τον προγραμματισμό της πλακέτας (JTAG, μνήμη flash ή κάρτα SD), ο βραχυκυκλωτήρας στην άκρη δεξιά ελέγχει τη λειτουργία JTAG και ο βραχυκυκλωτήρας στην αριστερή καθορίζει εάν χρησιμοποιείται το εσωτερικό PLL.
Η λειτουργία JTAG αναφέρεται στη μέθοδο εντοπισμού σφαλμάτων χρησιμοποιώντας το πρωτόκολλο JTAG. Αυτό μπορεί είτε να είναι διαδοχικά, όταν μία μόνο σύνδεση JTAG χρησιμοποιείται για διασύνδεση με τις θύρες πρόσβασης εντοπισμού σφαλμάτων τόσο στο PS όσο και στο PL, είτε ανεξάρτητα, όπου οι θύρες πρόσβασης εντοπισμού σφαλμάτων στο PS και PL έχουν πρόσβαση ξεχωριστά, απαιτώντας ένα καλώδιο για κάθε μία από αυτές. Η λειτουργία PLL καθορίζει εάν η διαδικασία διαμόρφωσης της συσκευής, περιλαμβάνει μια φάση αναμονής για το

κλείδωμα του PLL, πριν ξεκινήσει η διαδικασία εκκίνησης. Η εναλλακτική λύση είναι να παρακαμφθεί το PLL, αλλά σε αυτήν την περίπτωση, η εκκίνηση διαρκεί περισσότερο.



Εικόνα 3.1.6.1. Ακίδες jumpers της πλακέτας ZedBoard [69].

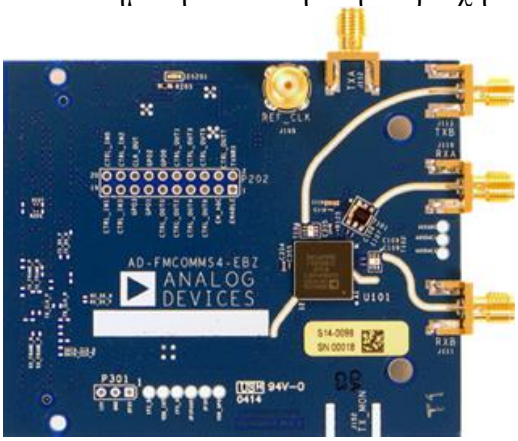


Εικόνα 3.1.6.2. Θέσεις ακίδων jumpers της πλακέτας ZedBoard [69].

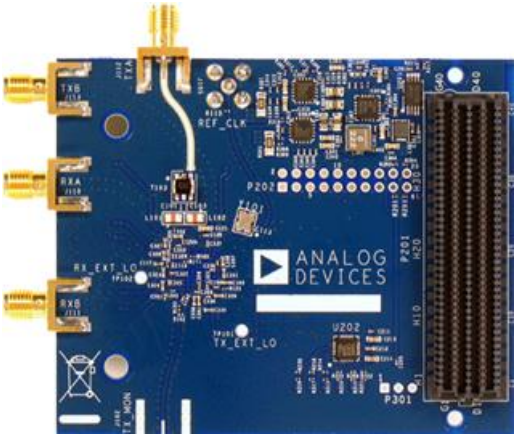
3.2 AD-FMCOMMS4-EBZ

Η πλακέτα AD-FMCOMMS4-EBZ[73] είναι μια υψηλής ταχύτητας μονού καναλιού αναλογική RF διάταξη της εταιρίας Analog Devices [74] με ένας κονέκτορα τύπου FMC (Fixed Mobile Convergence) [75], βασισμένη στο ολοκληρωμένο σύστημα AD9364[76], της ίδιας εταιρίας. Όταν συνδυαστεί με μία αναπτυξιακή πλακέτα FPGA, μπορεί να υλοποιήσει και να εξομοιώσει μία ποικιλία από αναλογικά και ψηφιακά τηλεπικοινωνιακά συστήματα 1x1, στην περιοχή συχνοτήτων από 70 MHz ως 6GHz ή για μέγιστή RF απόδοση στην περιοχή των 2.4GHz με 2.5GHz, με δυνατότητα επιλογής εύρους καναλιών από 200KHz μέχρι 56MHz.

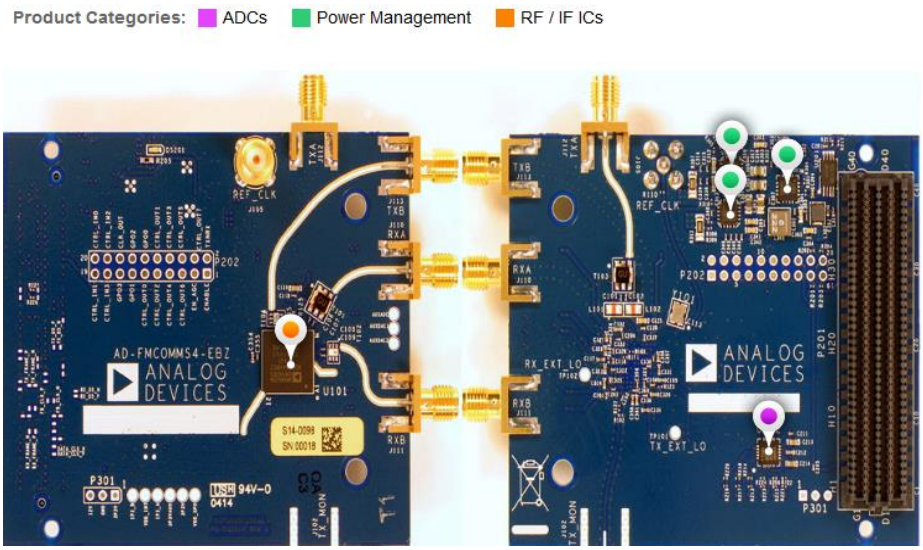
Η πλακέτα AD-FMComms4-EBZ περιλαμβάνει δύο τύπους εξωτερικών RF μετασχηματιστών (Baluns) [77], έναν στοχευμένο για εφαρμογές μεγάλου εύρους συντονισμού (Mini-circuits TCM1-63AX +) [78] και έναν με βελτιστοποιημένη απόδοση στην περιοχή των 2.4 GHz με 2.5GHz.



Εικόνα 3.2.1 Πρόσθια όψη πλακέτας AD-FMCOMMS4-EBZ [73].



Εικόνα 3.2.2 Οπίσθια όψη πλακέτας AD-FMCOMMS4-EBZ [73].



Εικόνα 3.2.3 Διάταξη κύριων εξαρτημάτων της πλακέτας AD-FMCOMMS4-EBZ [73].

3.2.1 Τεχνικά χαρακτηριστικά AD9364

Τα τεχνικά χαρακτηριστικά του AD9364 παρουσιάζονται συνοπτικά παρακάτω:

- RF 1× 1 πομποδέκτης με ενσωματωμένο 12-bit DAC και ADC
- Ζώνη συχνοτήτων εκπομπής: 70 MHz με 6.0 GHz
- Ζώνη συχνοτήτων λήψης: 70 MHz με 6.0 GHz

- Υποστηρίζει λειτουργίες Ανάπτυξης Οδηγούμενη από Έλεγχο - Test Driven Development (TDD) [79] και Ανάπτυξης Οδηγούμενη από Χαρακτηριστικό - Feature-driven development (FDD) [80]
- Ρυθμιζόμενο εύρος καναλιού: <200 kHz to 56 MHz
- Ευαισθησία δέκτη με στάθμη θορύβου <2.5 dB
- Ανεξάρτητο Αυτόματο σύστημα ελέγχου της ενίσχυσης
- Γραμμικός Πομπός Ευρείας Ζώνης Highly
- Μέτρο Σφάλματος Διανύσματος Εκπομπής - TX Error Vector magnitude (EVM) [81]: ≤−40 dB
- Θόρυβος εκπομπής: ≤−157 dBm/Hz από το επίπεδο της στάθμη θορύβου
- Εκπομπή: ≥66 dB δυναμικό εύρος με 1 dB ακρίβεια
- 2.4 Hz μέγιστο βήμα τοπικού ταλαντωτή - Local Oscillator (LO)
- CMOS/LVDS [82] ψηφιακή διεπαφή

3.2.2 Περιγραφή AD9364

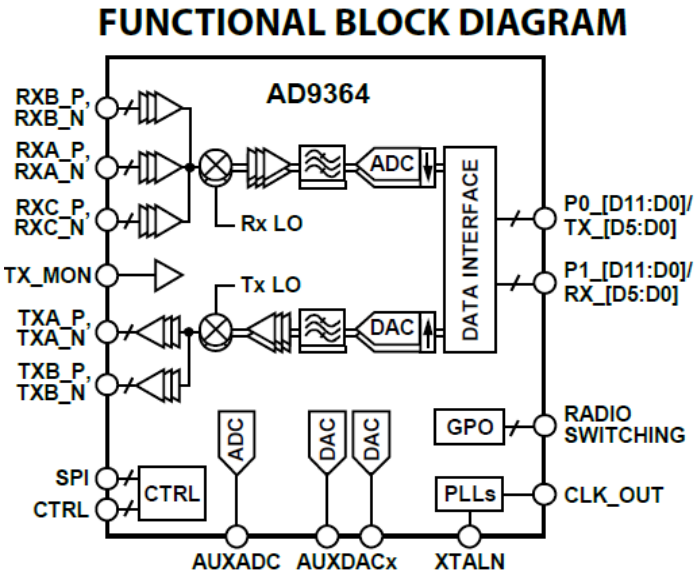
Το ολοκληρωμένο AD9364 είναι ένα υψηλής απόδοσης, RF πομποδέκτης, σχεδιασμένος για χρήση σε ένα ευρύ φάσμα εφαρμογών. Το ολοκληρωμένο ενσωματώνει όλα τις απαραίτητα RF μονάδες και ψηφιακά μπλοκ για την παροχή όλων των λειτουργιών ενός; πομποδέκτη σε μία μόνο συσκευή. Η ευκολία προγραμματισμού του καθώς και η μεγάλη συχνοτική περιοχή λειτουργίας, του επιτρέπουν να προσαρμόζεται για χρήση σε διάφορα πρότυπα επικοινωνίας και διαμορφώσεις, συμπεριλαμβανομένων των συστημάτων Διπλεξίας Διαίρεσης Συχνότητας Frequency Division Duplex (FDD) και Χρονοδιαιρετικής Αμφίδρομη Επικοινωνία - Time Division Duplex (TDD) Αυτή η ευκολία προγραμματισμού του ολοκληρωμένου, επιτρέπει τη διασύνδεση του, με διάφορους επεξεργαστές βασικής ζώνης – Baseband Processors (BBP), χρησιμοποιώντας μια μόνο παράλληλη θύρα δεδομένων 12-bit ή μια διεπαφή Διαφορικής Σηματοδότησης Χαμηλής Τάσης 12-bit – Low Voltage Differential Signaling (LVDS).

Το ολοκληρωμένο AD9364 παρέχει επίσης συστήματα αυτόματης βαθμονόμησης και βρόγχους Αυτόματου Ρυθμιζόμενου Κέρδους - Automatic Gain Control (AGC), για τη διατήρηση υψηλού επιπέδου απόδοσης, υπό διαφορετικές θερμοκρασίες και συνθήκες σήματος εισόδου. Επιπλέον, η συσκευή περιλαμβάνει αρκετές λειτουργίες δοκιμής, που επιτρέπουν στους σχεδιαστές συστημάτων, να εισάγουν κάποια σήματα δοκιμής και να δημιουργούν εσωτερικές λειτουργίες ελέγχου, που μπορούν να χρησιμοποιηθούν, για να διορθώσουν τα σχέδιά τους, κατά τη διάρκεια της δημιουργίας του πρωτοτύπου και να βελτιστοποιήσουν τη σχεδίαση τους, για μια συγκεκριμένη εφαρμογή.

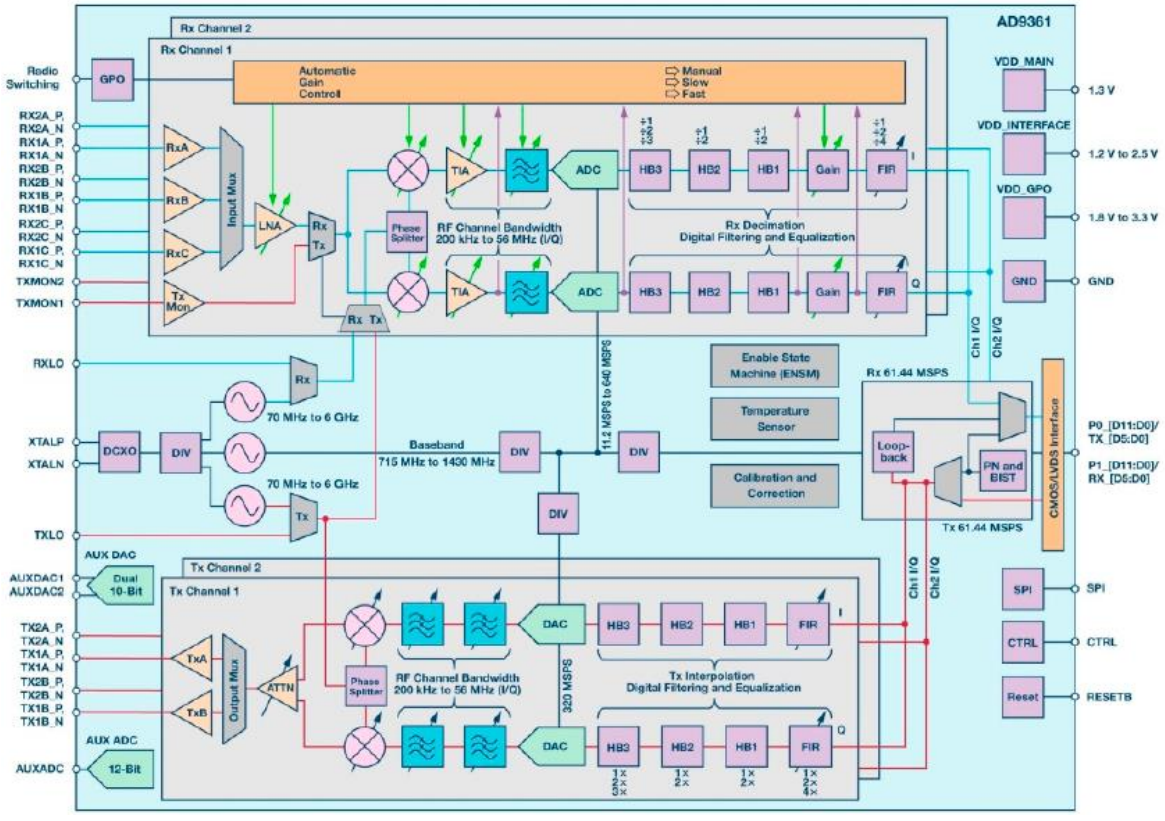
Στον δέκτη του AD9364 ο τοπικός ταλαντωτής λειτουργεί από 70 MHz έως 6.0 GHz και ο αντίστοιχος στον πομπό από 47 MHz έως 6.0 GHz, καλύπτοντας με αυτόν τον τρόπο, όλες τις αδειοδοτημένες και μη περιοχές συχνοτήτων. Το εύρος κάθε καναλιού είναι δυνατό να οριστεί από 200 KHz έως 56 MHz.

Οι δύο ανεξάρτητοι άμεσης μετατροπής δέκτες (RX) παρέχουν υψηλή ικανότητα καταστολής του θορύβου και γραμμικότητα. Κάθε υποσύστημα λήψης (RX) περιλαμβάνει αυτόνομο ανεξάρτητο Βρόγχο Ρυθμιζόμενου Κέρδους– Automatic Gain Control (AGC), Διόρθωση DC Συνιστώσας (DC Offset Correction), Διόρθωση Πλάτους (Quadrature Correction) και ψηφιακό φιλτράρισμα, εξαλείφοντας έτσι την ανάγκη εκτέλεσης των λειτουργιών αυτών, κατά την φάση της αποδιαμόρφωση των Pass-Band σημάτων που λαμβάνονται από αυτό. Επίσης, το AD9364 έχει ευέλικτες Λειτουργίες Χειροκίνητου Ελέγχου του Κέρδους που μπορούν να είναι ρυθμιστούν εξωτερικά. Δύο υψηλού δυναμικού εύρους Μετατροπείς Αναλογικού Σήματος σε Ψηφιακό (ADC) ανά κανάλι, ψηφιοποιούν τα λαμβανόμενα σήματα I και Q και περνούν μέσω Ρυθμιζόμενων Φίλτρων Αποδεκατισμού (Decimation Filters) και Φίλτρων Πεπερασμένης Χρονικής Απόκρισης με συντελεστή καθυστέρησης 128 σημείων - 128-tap Finite Impulse Response (FIR), για την παραγωγή σήματος εξόδου 12-bit με την κατάλληλη συχνότητα δειγματοληψίας. Οι πομποί χρησιμοποιούν αρχιτεκτονική άμεσης μετατροπής, που επιτυγχάνει υψηλή ακρίβεια διαμόρφωσης

με ελάχιστο θόρυβο. Αυτή η αρχιτεκτονική του πομπού παράγει το μικρότερο Μέτρο Σφάλματος Διανύσματος Εκπομπής (EVM), της τάξης των <−40 dB, εισάγοντας διευρυμένα περιθώρια για την επιλογή κατάλληλης εξωτερικής Ενίσχυσης Ισχύος (Power Amplification - PA).



Εικόνα 3.2.2.1 Λειτουργικό Μπλοκ διάγραμμα ολοκληρωμένου AD9364 [73].



Εικόνα 3.2.2.2 Μπλοκ διάγραμμα δομής ολοκληρωμένου AD9364 [83].

3.2.3 Δέκτης AD9364

Ο δέκτης του AD9364 περιλαμβάνει όλα εκείνα τα απαραίτητα μπλοκ για τη λήψη σημάτων RF και τη μετατροπή τους σε ψηφιακά δεδομένα, που μπορούν να χρησιμοποιηθούν από ένα BBP. Έχει τρεις εισόδους, που μπορούν να πολυπλεχθούν, καθιστώντας το AD9364 κατάλληλο για χρήση σε συστήματα με πολλαπλές εισόδους κεραίας.

Η αρχιτεκτονική του δέκτη στηρίζεται στην Άμεση Μετατροπή (Direct Conversion). Αυτή η αρχιτεκτονική περιλαμβάνει έναν Ενισχυτή Χαμηλής Στάθμης Θορύβου – Low Noise Amplifier (LNA), από δύο ενισχυτές I (In-phase) και Q (Quadrature), μίκτες και φίλτρα διαμόρφωσης ζώνης, έτσι ώστε το λαμβανόμενο σήμα να αποδιαμορφωθεί με την χρήση ενός Τοπικού Ταλαντωτή (Local Oscillator – LO) και να εξαχθούν τα φανταστικά (I) και τα πραγματικά (Q) μέρη του σήματος. Αφού τα σήματα I – Q ενισχυθούν από έναν Τελεστικό Ενισχυτή Διαντίστασης (Transimpedance Amplifier – TIA) ,θα απομονωθεί το προκαθορισμένο εύρος ζώνης τους μέσω φιλτραρίσματος από ένα Φίλτρο Χαμηλής Ζώνης (Band Pass Filter – BPF)
Ο έλεγχος της ενίσχυσης του δέκτη, επιτυγχάνεται ακολουθώντας έναν Προκαθορισμένο Χάρτη Κέρδους (Reprogrammed Gain Index Map), που κατανέμει το κέρδος μεταξύ των μπλοκ για βέλτιστη απόδοση σε κάθε επίπεδο. Αυτό μπορεί να επιτευχθεί ενεργοποιώντας το εσωτερικό AGC είτε σε γρήγορη είτε σε αργή λειτουργία ή χρησιμοποιώντας χειροκίνητο έλεγχο κέρδους, επιτρέποντας στον BBP να κάνει τις ρυθμίσεις κέρδους ανάλογα με τις ανάγκες της εφαρμογής. Επιπλέον, κάθε κανάλι περιέχει ανεξάρτητη ικανότητα Μέτρησης της Ισχύς του Λαμβανόμενου Σήματος – Received Signal Strength Indication (RSSI) [84], παρακολούθηση της συνιστώσας DC του σήματος και όλα τα απαραίτητα κυκλώματα για αυτορρύθμιση του.
Ο δέκτης τέλος, περιλαμβάνει 12-bit, Σ-Δ ADC μετατροπείς, ώστε να παραχθεί μια ψηφιακή λέξη των 12 bits. Τα δύο ξεχωριστά ψηφιακά σήματα, διέρχονται από μία συστοιχία Φίλτρων Αποδεκατισμού (Decimation Filter – DF) [85], ώστε να γίνει η Υποδειγματοληψία αυτών (Down Converting) και από ένα πλήρως προγραμματιζόμενο ψηφιακό Φίλτρο Πεπερασμένης Απόκρισης Παλμών (Finite Impulse Response – FIR) 128 σημείων, ώστε να επιτευχθεί η αντιστάθμιση, το φιλτράρισμα καθώς και ο επιθυμητός Ρυθμός Δειγματοληψίας (Sample Rate). Ο ρυθμός δειγματοληψίας κάθε ψηφιακού φίλτρου είναι ρυθμιζόμενος, αλλάζοντας τους συντελεστές αποδεκατισμού με τέτοιον τρόπο ώστε να παράγει τον επιθυμητό ρυθμό δειγματοληψία (Sample Rate) [73].

3.2.4 Πομπός AD9364

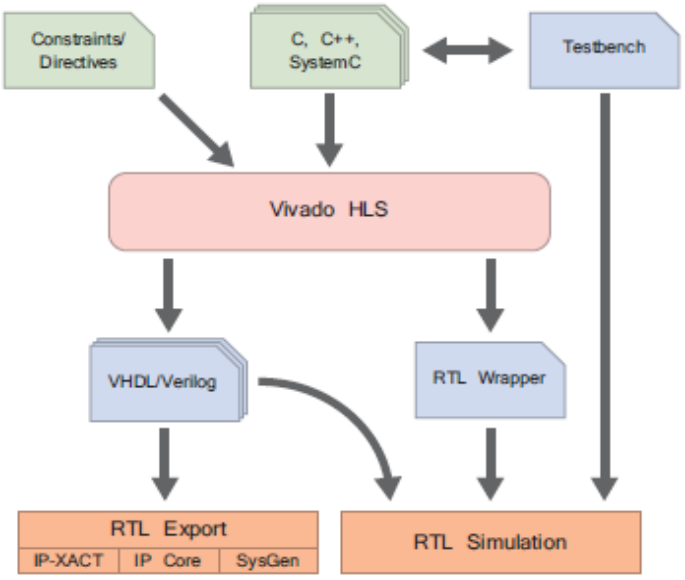
Η αρχιτεκτονική του πομπού επίσης στηρίζεται στην Άμεση Μετατροπή (Direct Conversion). Τα I και Q σήματα που λαμβάνονται από το BBP, διέρχονται μέσω ενός πλήρως προγραμματιζόμενου φίλτρου FIR 128 σημείων. Η έξοδος του FIR οδηγείται σε μια συστοιχία Φίλτρων Παρεμβολής (Interpolation Filter – IF), με σκοπό το φιλτράρισμα, καθώς και την Υπερδειγματοληψία (Upconverting) των δύο ψηφιακών σημάτων ώστε να είναι συμβατά με τους μετατροπείς DAC 12-bits. Στην συνέχεια, τα ψηφιακά σήματα I και Q, τροφοδοτούνται στο μπλοκ RF, για την μετατροπή τους σε αναλογικά Σήματα Βασικής Ζώνης (Baseband Analog Signals).
Όταν μετατρέπονται σε αναλογικά σήματα βασικής ζώνης, τα σήματα I και Q φιλτράρονται από μία συστοιχία Χαμηλοπερατών Φίλτρων – Low Pass Filters (LPF), για την αφαίρεση του άπειρου αριθμού νέων συχνοτήτων/αρμονικών (remove sampling artifacts), που δημιουργούνται κατά την διάρκεια της δειγματοληψίας και που δεν υπήρχαν στο αρχικό σήμα [86]. Το φιλτραρισμένο σήμα, στην συνέχεια, τροφοδοτείται στους μίκτες, προκειμένου να διαμορφωθούν στην επιλεγμένη συχνότητα μέσω του τοπικού ταλαντωτή (LO) και στη συνέχεια συνθέτονται ώστε να προκύψει ένα τελικό σήμα προς αποστολή στην έξοδο. Το συνδυασμένο σήμα περνά μέσω μίας σειράς αναλογικών φίλτρων που παρέχουν μία πρόσθετη διαμόρφωση ζώνης και μεταφέρεται στον ενισχυτή εξόδου. Τέλος αυτό το τελικό διαμορφωμένο αναλογικό σήμα διέρχεται από έναν Προγραμματιζόμενο Εξασθενητή Πλάτους (Attenuator – ATTN) ο οποίος μειώνει την δυναμική περιοχή του και στη συνέχεια μέσω του πολυπλέκτη επιλέγεται η κεραία με την οποία θα αποσταλεί. Το κανάλι μετάδοσης παρέχει την δυνατότητα προσαρμογής του εύρους της εξασθένισης, για να βοηθήσει τους σχεδιαστές, να βελτιστοποιήσουν την Συματοθορυβική Αναλογία – Signal to Noise Ratio (SNR).
Ο πομπός επίσης περιλαμβάνει ένα Μπλοκ Παρακολούθησης της Εκπομπής (Tx Monitor Block). Αυτό το μπλοκ παρακολουθεί την έξοδο του πομπού και το δρομολογεί, μέσω του καναλιού δέκτη, πίσω στο BBP ,με σκοπό την παρακολούθηση του εκπεμπόμενου σήματος. Το μπλοκ Tx Monitor είναι διαθέσιμο μόνο σε λειτουργία TDD, ενώ ο δέκτης είναι αδρανής [73].

3.3 Λογισμικό

Στην ενότητα αυτή θα γίνει μία εκτενής αναφορά στο λογισμικό που χρησιμοποιήθηκε για την υλοποίηση ενός συστήματος SDR και συγκεκριμένα του λογισμικού Vivado Design Suite της Xilinx.

3.3.1 Vivado Design Suite

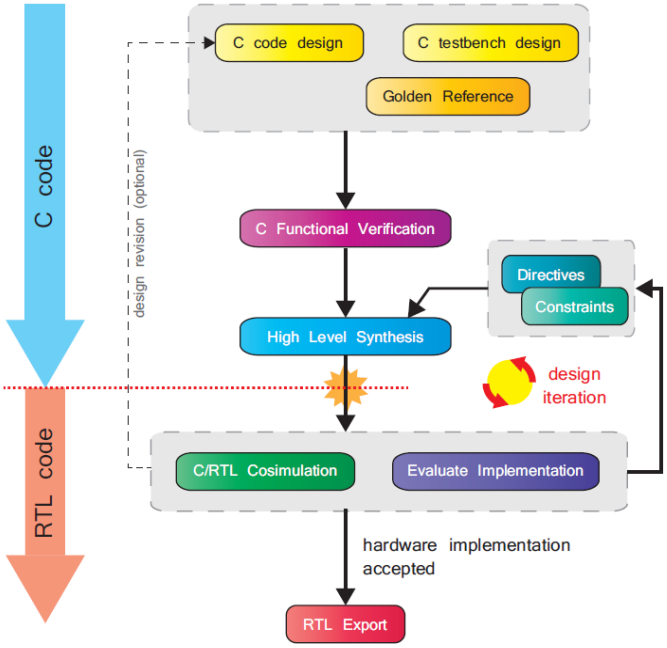
Το Vivado Design Suite[87],[88] κυκλοφόρησε τον Απρίλιο του 2012 από την εταιρία Xilinx και είναι ένα Ολοκληρωμένο Περιβάλλον Σχεδίασης – Integrated Designed Environment (IDE)[89] με εργαλεία επιπέδου συστήματος σε IC, που βασίζονται σε ένα κοινόχρηστο μοντέλο κλιμάκωσης δεδομένων και ένα κοινό περιβάλλον εντοπισμού σφαλμάτων. Το Vivado περιλαμβάνει εργαλεία σχεδίασης Επιπέδου Ηλεκτρονικού Συστήματος- Electronic System Level (ESL)[90] για τη σύνθεση και την επαλήθευση αλγοριθμικού IP με βάση την γλώσσα C. Αυτή η διαδικασία αποτελεί το βήμα σύνθεσης υψηλού επιπέδου, όπως απεικονίζεται στην εικόνα 3.3.1.1 που ακολουθεί.



Εικόνα 3.3.1.1 Διάγραμμα Ροής Vivado [69].

3.3.1.1 Διάγραμμα Ροής Vivado HLS

Το διάγραμμα ροής σχεδιασμού για το HLS περιλαμβάνει αρκετά στάδια, συμπεριλαμβανομένων της εκτέλεσης του αλγόριθμού HLS, της παραγωγής του RTL, καθώς και των στοιχείων επαλήθευσης. Το πλήρες διάγραμμα ροή σχεδιασμού παρουσιάζεται στην εικόνα 3.3.1.1.1 που ακολουθεί.



Εικόνα 3.3.1.1.1 Στάδια Σύνθεσης Υψηλού επιπέδου HLS [69].

3.3.1.2 Είσοδοι της διαδικασίας HLS

Η κύρια είσοδος στη διαδικασία HLS είναι ένας κώδικας γραμμένος στην γλώσσα C/C++/SystemC, μαζί με μία εφαρμογή ελέγχου (testbench) που στηρίζεται στην γλώσσα C και η οποία έχει αναπτυχθεί για την εκτέλεση της συνάρτησης και την επαλήθευση της σωστής λειτουργίας της. Αυτό θα περιλαμβάνει μια ειδική αναφορά ελέγχου, με την οποία θα συγκριθούν τα αποτελέσματα που παράγονται από τον κώδικα που προορίζεται για σύνθεση. Η αναφορά αυτή μπορεί να έχει τη μορφή ενός έτοιμου συνόλου τιμών εξόδου ή μπορεί να αποτελεί μέρος της ίδιας της εφαρμογής ελέγχου (testbench).

3.3.1.3 Συνάρτηση Επαλήθευσης

Πριν ξεκινήσει η διαδικασία σύνθεσής του κώδικα RTL, είναι απαραίτητη η επαλήθευση της λειτουργικής ακεραιότητας του κώδικα C/C++/ SystemC, που αποτελεί την είσοδο στην διαδικασία HLS. Αυτό μπορεί να επιτευχθεί γράφοντας μία εφαρμογή ελέγχου (testbench) στην ίδια γλώσσα υψηλού επιπέδου και ελέγχοντας τα αποτελέσματα που παράγονται με κάποια ειδική αναφορά ελέγχου.

3.3.1.4 Σύνθεση Υψηλού Επιπέδου HLS

Το επόμενο βήμα είναι η ίδια η διαδικασία HLS, η οποία περιλαμβάνει ανάλυση και επεξεργασία του κώδικα C, μαζί με οδηγίες και περιορισμούς που παρέχονται από τον χρήστη, για τη δημιουργία μιας περιγραφής RTL του κυκλώματος. Μόλις ολοκληρωθεί η διαδικασία HLS, παράγεται ένα σύνολο αρχείων εξόδου, συμπεριλαμβανομένων αρχείων σχεδίασης στην επιθυμητή γλώσσα RTL. Δημιουργούνται επίσης διάφορα άλλα αρχεία καταγραφής και αναφοράς, testbench κ.λπ.

3.3.1.5 Συνδυαστική εξομοίωση C/RTL

Μόλις ολοκληρωθεί η διαδικασία HLS και παραχθεί το αντίστοιχο μοντέλο RTL, μπορεί να ελεγχθεί έναντι του αρχικού κώδικα C/C++/SystemC μέσω της διαδικασίας συνδυαστικής εξομοίωσης C/RTL στο Vivado HLS. Αυτή η διαδικασία επαναχρησιμοποιεί την αρχική εφαρμογή ελέγχου (testbench), για να παρέχει εισόδους στην έκδοση RTL που δημιουργείται από την διαδικασία HLS και ελέγχει τις εξόδους που παράγονται σε σχέση με τις αναμενόμενες τιμές. Η έξοδος SystemC είναι ιδιαίτερα χρήσιμη, καθώς παρέχει έναν μηχανισμό για την επαλήθευση του σχεδιασμού σε περιβάλλοντα όπου δεν υπάρχει διαθέσιμος προσομοιωτής HDL.

3.3.1.6 Αξιολόγηση της Ενσωμάτωσης

Μαζί με την επαλήθευση της ακεραιότητας του σχεδιασμού, είναι επίσης απαραίτητο να αξιολογηθεί η έξοδος RTL όσον αφορά την εφαρμογή και την απόδοσή της. Για παράδειγμα, οι αριθμοί πόρων που απαιτούνται στο PL, η καθυστέρηση λόγω σχεδιασμού, η μέγιστη υποστηριζόμενη συχνότητα ρολογιού και ούτω καθεξής.

3.3.1.7 Αξιολόγηση της Ενσωμάτωσης

Η ενσωμάτωση του RTL αξιολογείται και, όπου απαιτείται, οι περιορισμοί και οι οδηγίες βελτιώνονται. Είναι επίσης πιθανό ότι τα αποτελέσματα της αξιολόγησης της ενσωμάτωσης, να αποτελέσουν την αιτία για μία πιο θεμελιώδη αναθεώρηση και βελτίωση του αρχικού κώδικα C.
Στην εικόνα 3.3.2.1, παρουσιάζονται τα βήματα που έγιναν από το σχεδιασμό στην γλώσσα C, έως τη δημιουργία εξόδων για σύνθεση RTL. Ενδεχομένως να χρειαστεί να πραγματοποιηθούν πολλαπλές επαναλήψεις της διαδικασίας HLS, χρησιμοποιώντας τροποποιημένες οδηγίες και περιορισμούς, προκειμένου να βρεθεί μια «καλύτερη» λύση. Αυτή η διαδικασία της επανάληψης, αποτυπώνεται στη διαδρομή ανατροφοδότησης που εμφανίζεται στη δεξιά πλευρά του διαγράμματος.
Στην περίπτωση που ζητηθεί από τον σχεδιαστή, να αλλάξει τον κώδικα C, εμπλέκεται ένα πιο σημαντικό βήμα προς τα πίσω στη διαδικασία σχεδιασμού, και αυτό υποδεικνύεται από το βέλος στην αριστερή πλευρά του διαγράμματος. Οποιεσδήποτε αλλαγές στον κώδικα C απαιτούν λειτουργική εκ νέου επαλήθευση, πριν από την επακόλουθη επαλήθευση και οι διαδικασίες αξιολόγησης της ενσωμάτωσης εκτελούνται ξανά.

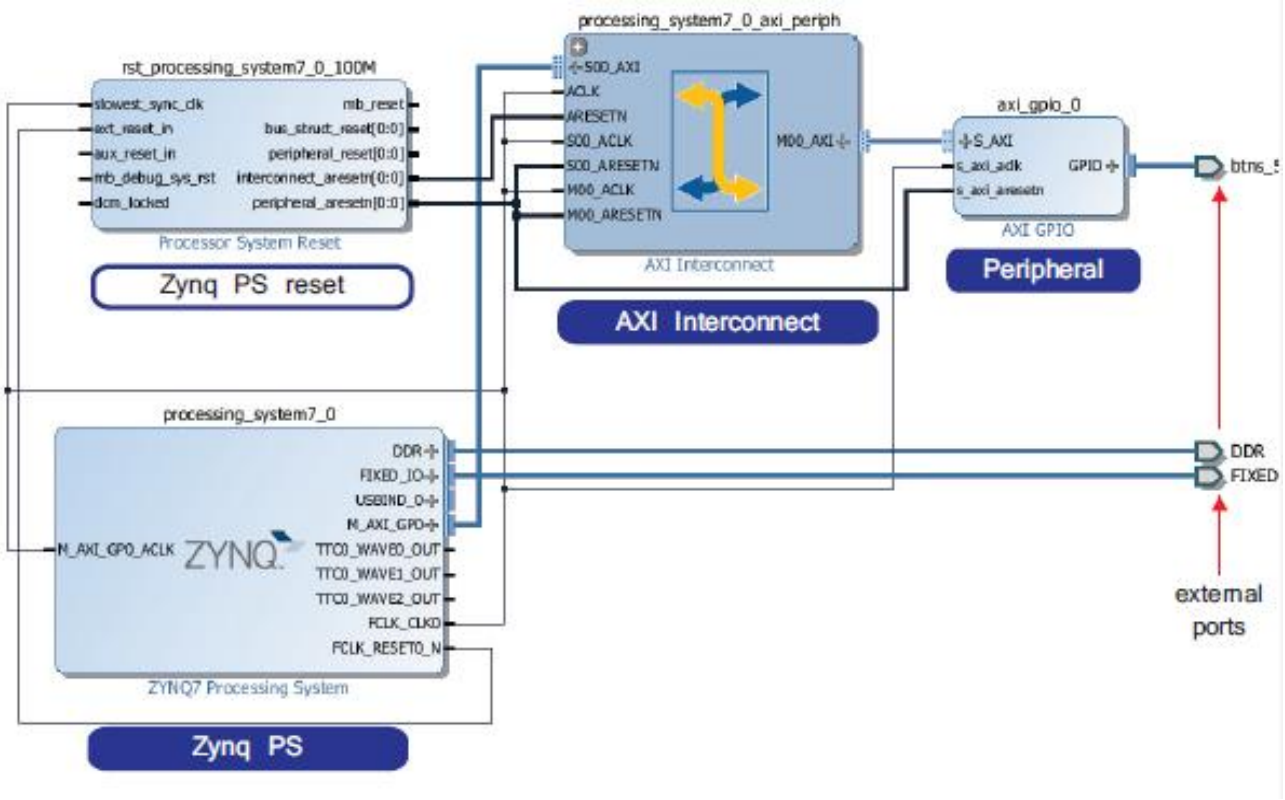
3.3.1.8 Εξαγωγή RTL

Μόλις ολοκληρωθεί η αξιολόγηση της ενσωμάτωσης και είμαστε σίγουροι για την επίτευξη των επιδιωκόμενων σχεδιαστικών στόχων, θα προχωρήσουμε στην ενσωμάτωση σε ένα μεγαλύτερο σύστημα. Αυτό μπορεί να επιτευχθεί απευθείας χρησιμοποιώντας τα παραγόμενα αρχεία RTL που δημιουργούνται αυτόματα από τη διαδικασία HLS.

3.3.1.9 Υλοποίηση της λογικής Intellectual Property (IP) στο Vivado

Ένας από τους σημαντικότερους παράγοντες στο σχεδιασμό σύγχρονων συστημάτων είναι η δυνατότητα επαναχρησιμοποίησης/επανασχεδιασμού μιας σχεδίασης και η ταχεία ανάπτυξη της. Ο χρόνος στον οποίο μία σχεδίαση καταλήγει στην αγορά είναι κρίσιμος και τα εργαλεία σχεδιασμού που επιταχύνουν τη διαδικασία ανάπτυξης, χωρίς να διακυβεύεται η ανθεκτικότητα της σχεδίασης ή η ποιότητας των αποτελεσμάτων, φέρνουν σαφή πλεονεκτήματα.
Η ροή σχεδιασμού του Vivado βασίζεται στην υπόθεση ότι πολλά δομικά στοιχεία του συστήματος είναι έτοιμα κομμάτια Intellectual Property (IP) [91] που μπορούν να ενσωματωθούν σε μία σχεδίαση. Σε αντίθεση με τις παλαιότερες μεθόδους σχεδιασμού, οι οποίες συνήθως καλύπτουν την υλοποίηση συστημάτων/σχεδιάσεων εξ ολοκλήρου από το μηδέν, το Vivado επικεντρώνεται στην εκμετάλλευση της προ-επαληθευμένης IP που υπάρχει στις βιβλιοθήκες του Vivado, τρίτων κατασκευαστών IP ή των προηγούμενων δραστηριοτήτων του σχεδιαστής. Σε μεγάλο βαθμό, η υλοποίηση μίας νέας σχεδίασης/εφαρμογής, έχει μετατοπιστεί προς ανώτερα επίπεδα, παρά στη σχεδίαση υλικού χαμηλού επιπέδου.
Ο ολοκληρωτής IP (IP Integrator) είναι η κύρια ενσωμάτωση της μεθόδου σχεδιασμού εστιασμένη στο IP. Ο IP Integrator είναι ένα χαρακτηριστικό του Vivado Design Suite[92]. Οι IP μπορούν είτε να εισαχθούν από τον υπάρχοντα κατάλογο, είτε να δημιουργηθούν ως μαύρα κουτιά για την μεταγενέστερη ενσωμάτωση λειτουργικών υποσυστημάτων και να δημιουργηθούν οι διεπαφές μεταξύ των διαφόρων στοιχείων. Αυτή η προσέγγιση προσφέρεται για ταχεία ανάπτυξη του συστήματος υλικού.
Το Vivado Design Suite περιλαμβάνει μια δυνατότητα, το IP Packager, το οποίο επιτρέπει την ενοποίηση των IP σε τυποποιημένα πακέτα (με βάση το πρότυπο IP-XACT [93]) με σκοπό τη διευκόλυνση της μελλοντικής επαναχρησιμοποίησης της υφιστάμενης σχεδίασης. Από την πλευρά της ομάδας σχεδιασμού, η δημιουργία ενός εσωτερικού χώρου αποθήκευσης εύκολα

επαναχρησιμοποιήσιμων IP, που διαμοιράζονται σε όλους τους, είναι μια συναρπαστική μέθοδος επιτάχυνσης των κύκλων σχεδιασμού προϊόντων.



Εικόνα 3.3.3.1 Παράδειγμα Ολοκληρωτή IP- IP Integrator [69].

3.3.2 MatLab/Simulink

Με την ταχεία πρόοδο στις τεχνολογίες σχεδιασμού FPGA, οι πάροχοι υλικού και λογισμικού εργάζονται για την παραγωγή πιο προηγμένων και φιλικών προς τον χρήστη εργαλείων για το σχεδιασμό και τη δοκιμή προγραμμάτων FPGA. Η έμφαση δίνεται στο γρήγορο πρωτότυπο του σχεδιασμού FPGA, ακόμη και με ελάχιστη γνώση των γλωσσών προγραμματισμού Verilog και VHDL.

Οι διάφοροι μηχανικοί λογισμικού και υλικού χρησιμοποιούν τα MatLab και Simulink για την ανάπτυξη πρωτοτύπων και εφαρμογών, για την ανάπτυξη τους σε συσκευές Xilinx FPGA και Zynq SoC. Με τα MatLab και Simulink, είναι δυνατή η υλοποίηση ενός μοντέλου αρχιτεκτονικής υλικού σε επίπεδο συστήματος, ο προγραμματισμός του FPGA ή του SoC χωρίς να γραφτεί κώδικας, η προσομοίωση και διόρθωση του FPGA ή του SoC χρησιμοποιώντας τα εργαλεία (add-ons) που περιέχονται σε αυτά, η εκτέλεση του σχεδιασμού FPGA και SoC παραγωγής [94].

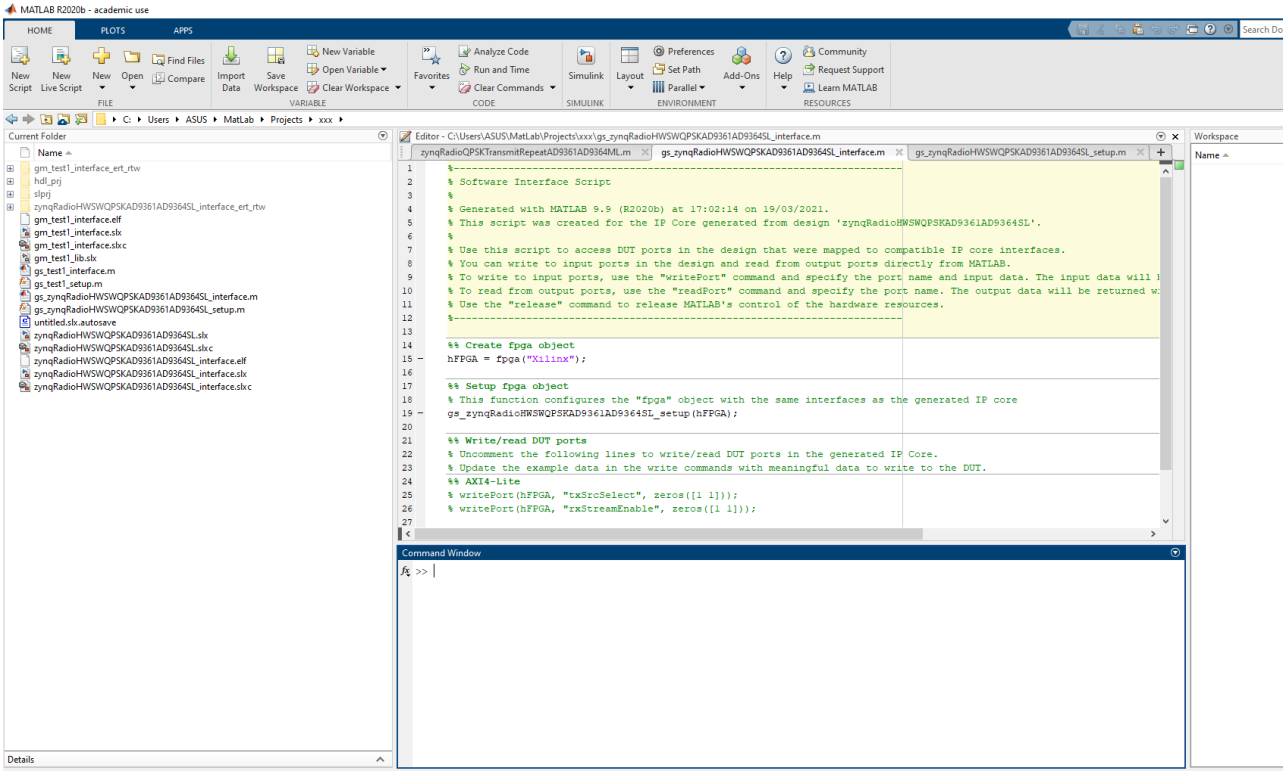
Και τα δύο αυτά εργαλεία επιτρέπουν στους προγραμματιστές να χρησιμοποιούν το περιβάλλον του Simulink, με χρήση drag and drop μπλοκ, για το σχεδιασμό των αλγορίθμων τους χωρίς να χρειάζεται να γράψουν ούτε μια γραμμή κώδικα VHDL, ακόμη και για πολύ μεγάλα και περίπλοκα σχέδια [95].

3.3.2.1 MatLab

Το MatLab (Matrix Laboratory) είναι ένα περιβάλλον αριθμητικής υπολογιστικής και μια προγραμματιστική γλώσσα τέταρτης γενιάς. Αποθηκεύει και κάνει τις πράξεις με βάση την άλγεβρα μητρών. Η τρέχουσα έκδοσή του είναι η R2020b η οποία κυκλοφόρησε στις 17 Σεπτεμβρίου του 2020.

Χρησιμοποιείται κατά κύριο λόγο για την επίλυση μαθηματικών προβλημάτων, ωστόσο είναι πολύ "ισχυρό" και μπορεί να χρησιμοποιηθεί και για προγραμματισμό, καθώς περιέχει εντολές από την C++ όπως την while, την switch και την if. Στον τομέα των γραφικών, όσον αφορά τον μαθηματικό

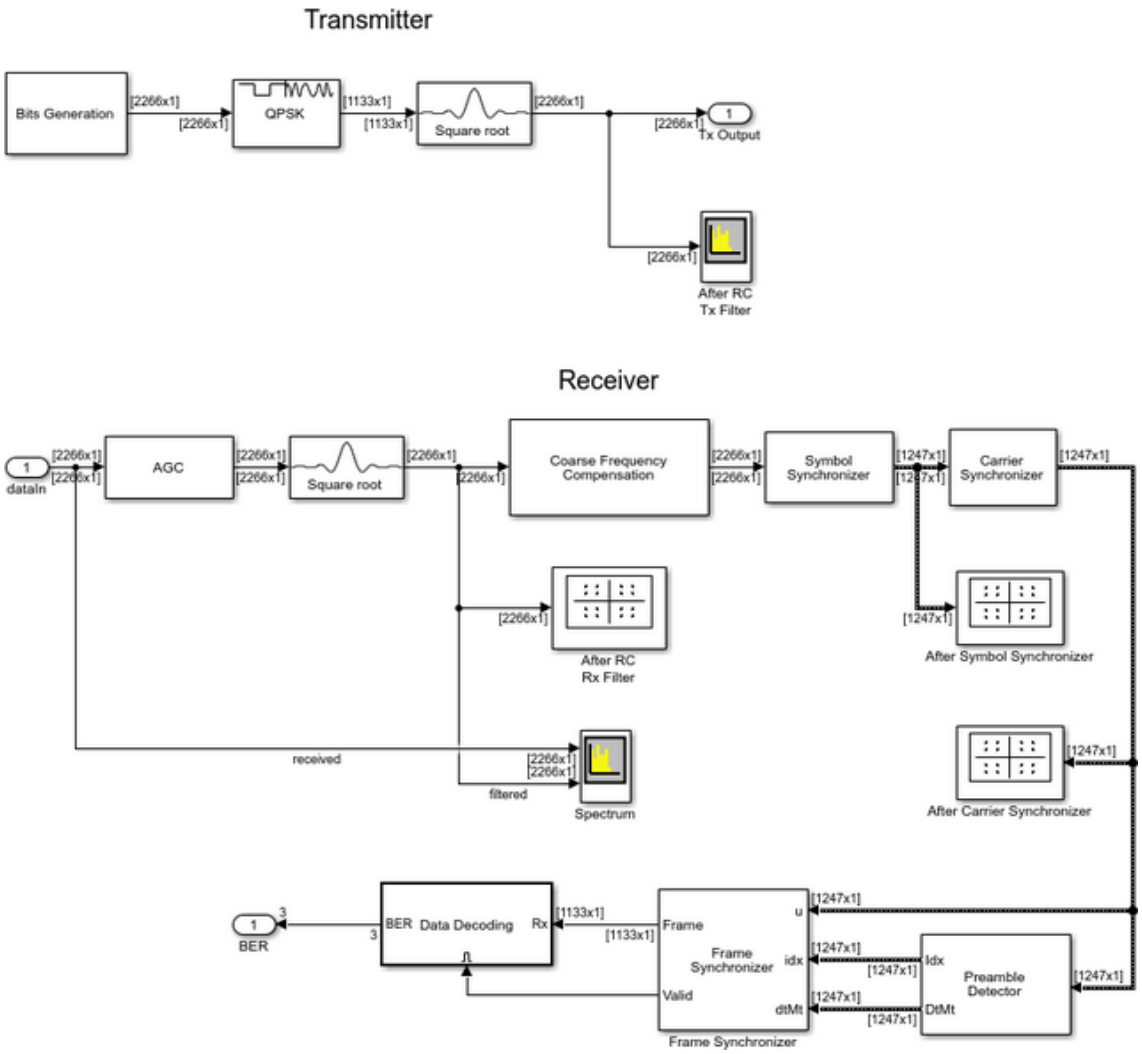
κλάδο, μπορεί να υλοποιήσει συναρτήσεις πραγματικές, μιγαδικές, πεπλεγμένες συναρτήσεις δύο μεταβλητών και άλλες. Όσον αφορά τον στατιστικό κλάδο μπορεί να υλοποιήσει ιστογράμματα, τομεογράμματα, ραβδοδιαγράμματα, εμβαδογράμματα και άλλα [96].



Εικόνα 3.3.2.1.1 Επιφάνεια εργασίας MatLab.

3.3.2.2 Simulink

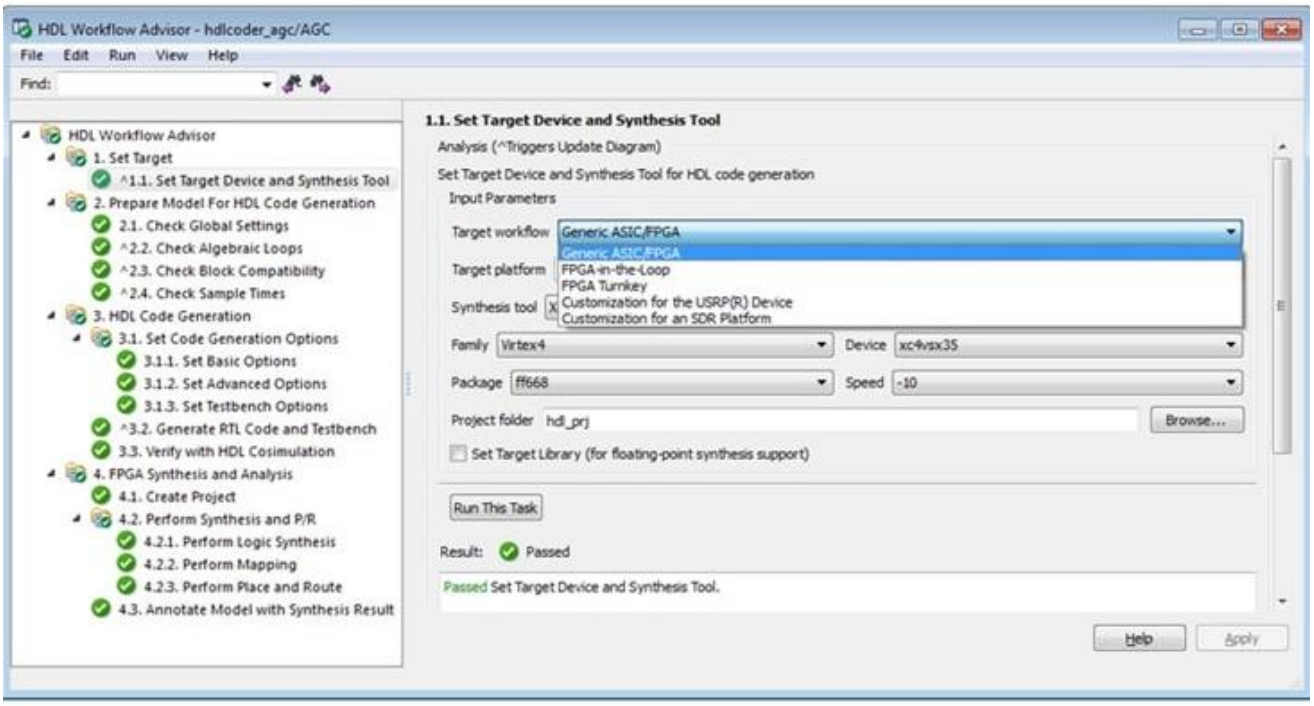
Το Simulink είναι ένα περιβάλλον γραφικού προγραμματισμού, βασισμένο στην γλώσσα προγραμματισμού MatLab. Χρησιμοποιείται για τη μοντελοποίηση, την προσομοίωση και την ανάλυση πληθώρας δυναμικών συστημάτων και εφαρμογών. Η βασική του διεπιφάνεια αποτελείται από ένα χώρο γραφικής απεικόνισης συστημάτων μέσω μεμονωμένων μπλοκ, τα οποία εισάγονται με μια διαδικασία «drag and drop» από ενσωματωμένες βιβλιοθήκες. Επικοινωνεί πλήρως και αμφίδρομα με το MatLab και τις υπόλοιπες εφαρμογές του. Παρέχεται, δηλαδή, η δυνατότητα αποστολής δεδομένων και εκτέλεσης κώδικα MatLab μέσω του Simulink, ενώ αντίθετα μοντέλα αυτού μπορούν να δημιουργηθούν προγραμματίζοντάς τα στο MatLab. Το Simulink χρησιμοποιείται ευρέως σε εφαρμογές αυτομάτου ελέγχου και ψηφιακής επεξεργασίας σήματος
Πολλά προϊόντα υλικού (hardware) και λογισμικού (software), τόσο από την MathWorks όσο και από τρίτους, μπορούν να χρησιμοποιηθούν και να αξιοποιηθούν μέσω Simulink.
Μέσω του Simulink Coder, το Simulink μπορεί να μετατρέψει ένα μοντέλο σε πηγαίο κώδικα γραμμένο σε C, κάτι που είναι χρήσιμο σε εφαρμογές πραγματικού χρόνου. Η ποιότητα στην αυτοματοποιημένη ανάπτυξη κώδικα αυξάνεται συνεχώς και πλέον το συγκεκριμένο εργαλείο χρησιμοποιείται συχνά από την βιομηχανία. Επιπλέον ο κώδικας είναι τόσο αποτελεσματικός που μπορεί να χρησιμοποιηθεί και σε ενσωματωμένα συστήματα.
Το Simulink Real Time, μαζί με x86-based συστήματα πραγματικού χρόνου, είναι ένα περιβάλλον για την προσομοίωση και τον έλεγχο Simulink και Stateflow μοντέλων σε πραγματικό χρόνο, πάνω στο φυσικό σύστημα. Όταν χρησιμοποιούνται μαζί με την ενσωματωμένη εφαρμογή HDL Coder, το Simulink και το Stateflow μπορούν να δημιουργήσουν αυτόματα VHDL και Verilog κώδικα.
Γενικά κάθε είδους προσομοίωση μπορεί να πραγματοποιηθεί στο λογισμικό και μάλιστα όλα είναι παραμετροποιήσιμα. Σημαντικά μεγάλος αριθμός μπλοκ είναι διαθέσιμα μέσω του Simulink Library Browser αλλά δίνεται η δυνατότητα δημιουργίας νέων μπλοκ εξολοκλήρου από τον χρήστη[97].



Εικόνα 3.3.2.2.1 Παράδειγμα υλοποίηση QPSK πομπού και δέκτη στο Simulink [98].

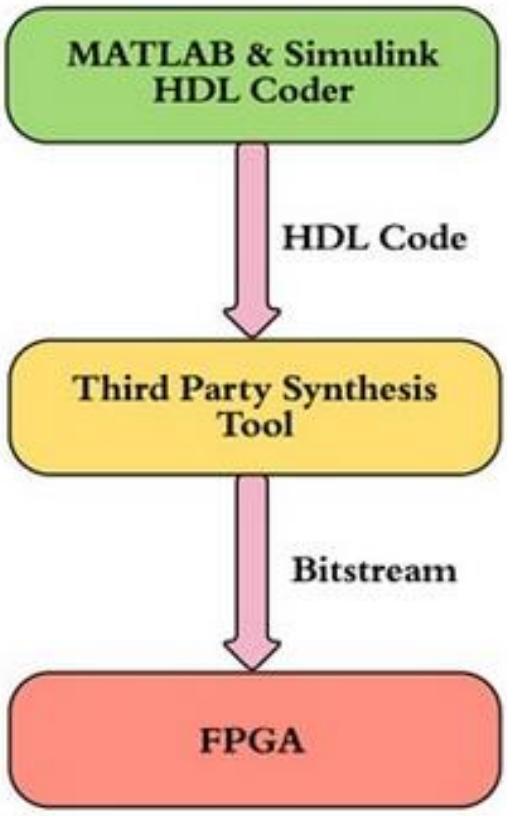
3.3.2.3 MatLab HDL Coder

Ο HDL Coder, που παρέχεται από την MathWorks, είναι μια εργαλειοθήκη του MatLab που δημιουργεί ανεξάρτητους από το υλικό, φορητούς και συνθετικούς κωδικούς Verilog και VHDL, οι οποίοι μπορούν στη συνέχεια να χρησιμοποιηθούν για προγραμματισμό FPGA και σχεδιασμό ASIC. Η εργαλειοθήκη λειτουργεί με διαφορετικές λειτουργίες MatLab, μοντέλα του Simulink και διαγράμματα κατάστασης (Stateflow charts), μετατρέποντας μία σχεδίαση του MatLab σε αντίστοιχους κωδικούς Verilog και HDL. Στο πλαίσιο των SDR, το περιβάλλον σχεδίασης Simulink συνιστάται για την γρήγορη δημιουργία ενός πρωτότυπου. Κατά την διάρκεια της σχεδίασης με τον HDL Coder στο Simulink, το πρώτο βήμα είναι να φιλτραριστεί το πρόγραμμα περιήγησης βιβλιοθήκης Simulink, έτσι ώστε να εμφανίζει μόνο τα μπλοκ μοντέλου που είναι συμβατά με τον HDL Coder. Για το σκοπό αυτό, πληκτρολογώντας "!hdllib" στη γραμμή εντολών MatLab, κάποιος παίρνει το πρόγραμμα περιήγησης βιβλιοθήκης Simulink που δείχνει μόνο τα υποστηριζόμενα μπλοκ. Ο HDL Coder υποστηρίζει πάνω από 200 μπλοκ Simulink, συμπεριλαμβανομένων γραφημάτων Stateflow. Χρησιμοποιώντας αυτά τα μπλοκ, μπορεί κανείς να υλοποιήσει μία σχεδίαση ως αρχείο μοντέλου Simulink, σύροντας και ρίχνοντας διάφορα μπλοκ στο σχεδιασμό. Ο HDL Coder μετατρέπει αυτόματα τους Αριθμούς Κινητής Υποδιαστολής (Floating Point Numbers) σε Αριθμούς με Σταθερό Σημείο (Fixed-Point). Επιπλέον, οι λειτουργίες που είναι γραμμένες σε MatLab mcode (Script), μπορούν επίσης να ενσωματωθούν στο σχεδιασμό. Για τον έλεγχο της σχεδίασης μέσω προσομοιώσεων, μπορούν να χρησιμοποιηθούν διαφορετικά εργαλεία του MatLab, συμπεριλαμβανομένων Παλμογράφων, Αναλυτών Φάσματος, κ.λπ. Μπορούν επίσης να χρησιμοποιηθούν μπλοκ από τη βιβλιοθήκη Simulink Sources, για τη δημιουργία δοκιμαστικών σημάτων για τον έλεγχο της σχεδίασης.



Εικόνα 3.3.2.3.1 Παράδειγμα HDL Workflow Advisor [95].

Μόλις τα αποτελέσματα του σχεδίασης επαληθευτούν μέσω προσομοιώσεων, το επόμενο βήμα είναι η δημιουργία του κώδικα HDL, ο οποίος γίνεται μέσω του Workflow Advisor, ο οποίος αποτελεί κομμάτι του HDL Coder. Μέσω του Workflow Advisor, μπορεί κανείς να επιλέξει διαφορετικές παραμέτρους της σχεδίασης, συμπεριλαμβανομένης της ροής εργασίας, της χρησιμοποιούμενης πλατφόρμας (Target Workflow), του χρησιμοποιούμενου FPGA, της δήλωσης των εξωτερικών θυρών της σχεδίασης κ.λπ. Στο πλαίσιο της υλοποίησης ενός συστήματος SDR, δίνεται η δυνατότητα επιλογής μια προσαρμόσιμη πλατφόρμα SDR, υπό την προϋπόθεση ότι η συγκεκριμένη πλατφόρμα υποστηρίζεται από τον HDL Coder [95].
Εάν ο στόχος είναι απλώς η δημιουργία κωδικών HDL, ο HDL Coder μπορεί να το κάνει ως αυτόνομο εργαλείο εντός του MatLab. Ωστόσο, εάν κάποιος πρέπει να συνθέσει τον κώδικα HDL και να δημιουργήσει ένα bitstream για το FPGA, απαιτούνται εργαλεία σύνθεσης τρίτων, όπως το Xilinx ISE και το Vivado ή το Altera Quartus. Στην εικόνα 3.3.2.3.2 παρουσιάζεται το διάγραμμα ροής για την δημιουργία του bitstream ,σε μια διαδικασία τριών βημάτων. Το πρώτο βήμα αναφέρεται στη σχεδίαση του μοντέλου χρησιμοποιώντας λειτουργίες MatLab και μπλοκ Simulink. Στο τέλος αυτού του βήματος, δημιουργούνται οι κωδικοί HDL οι οποίοι στη συνέχεια μπορούν να συντεθούν χρησιμοποιώντας εργαλεία τρίτων, όπως αυτά που αναφέρθηκαν παραπάνω, και δημιουργείται του bitstream. Τέλος, το bitstream μεταφορτώνεται στο FPGA. Το HDL Coder έχει τη δυνατότητα να ενσωματώνει τα εργαλεία τρίτων στο Workflow Advisor, χρησιμοποιώντας την επιλογή Synthesis Tool όπως φαίνεται στην εικόνα 3.3.2.3.1, παρέχοντας ένα ομοιόμορφο και ολοκληρωμένο περιβάλλον (Simulink Environment) για όλες τις διαδικασίες, ξεκινώντας από το σχεδιασμό μοντέλου μέχρι τη δημιουργία bitstream. Σε αυτό το σημείο, είναι επίσης επιτακτική ανάγκη να αναφερθεί ότι εάν κάποιος ήθελε να προσομοιώσει και να επαληθεύσει τους δημιουργημένους κωδικούς HDL, ο HDL Coder από μόνος του δεν επαρκεί και απαιτείται μια πρόσθετη εργαλειοθήκη, δηλαδή το HDL Verifier [99]. Με το HDL Verifier, θα απαιτηθούν κάποια επιπλέον εργαλεία προσομοίωσης προερχόμενα από άλλες εταιρίες όπως το Questa και το ModelSim. Επιπλέον, ο HDL Verifier παρέχει επίσης τη δυνατότητα "FPGA in the loop", επιτρέποντας στους προγραμματιστές την δυνατότητα να δοκιμάσουν τη σχεδίαση σε πραγματικό υλικό από το περιβάλλον Simulink. Η συνένωση HDL με HDL Verifier επιτρέπει σε κάποιον να επαληθεύσει ότι ο κωδικός HDL ταιριάζει με τους αλγόριθμους του MatLab και τα μοντέλα του Simulink.



Εικόνα 3.3.2.3.2 Διάγραμμα ροής HDL Coder [95].

3.3.3 Διασύνδεση εξοπλισμού με το προγραμματιστικό περιβάλλον MatLab

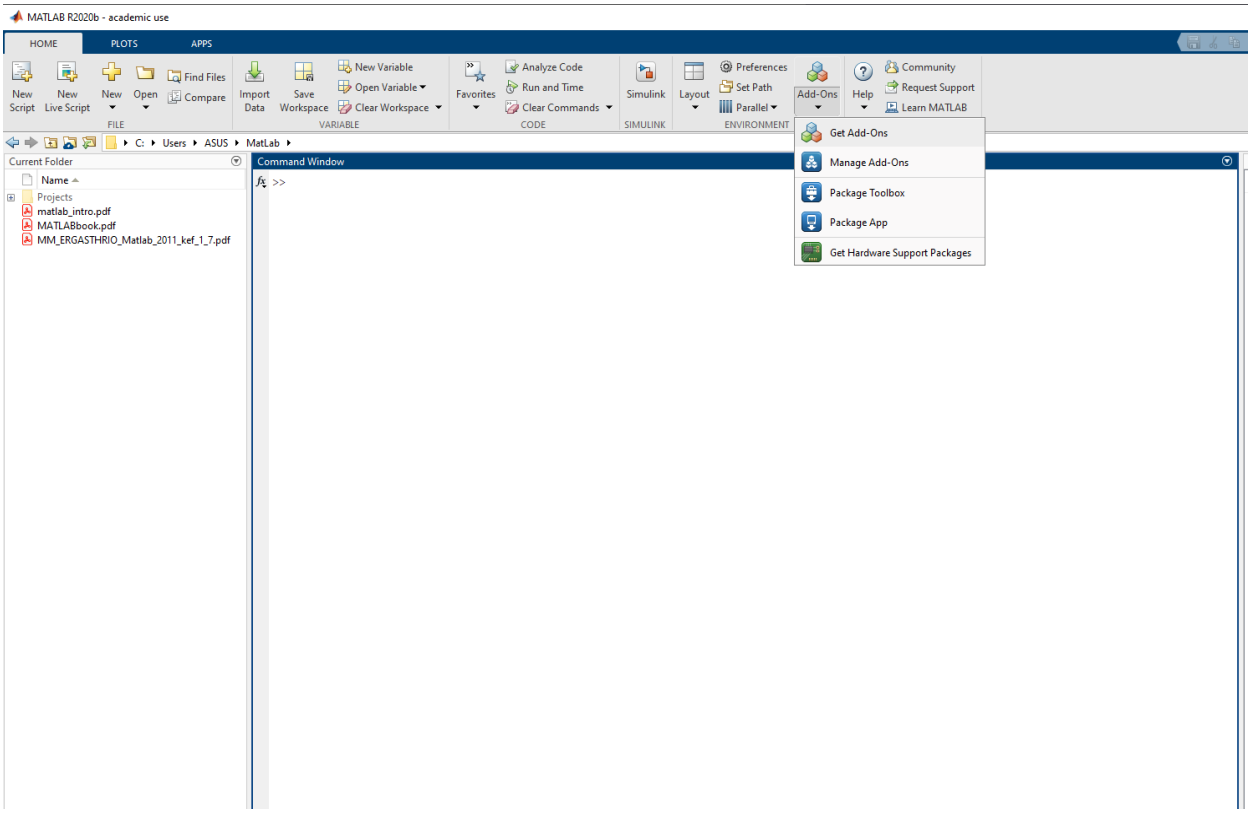
Για την διασύνδεση του εξοπλισμού με το προγραμματιστικό περιβάλλον του MatLab όπου θα δημιουργηθούν και θα υλοποιηθούν οι αλγόριθμοι των τηλεπικοινωνιακών υποσυστημάτων, θα χρειαστεί η εγκατάσταση μιας σειράς από επιπρόσθετων προσθηκών/εργαλείων (Add-Ons) με σκοπό την επίτευξης της επικοινωνίας μεταξύ του PC και της αναπτυξιακής πλακέτας, για τον προγραμματισμό της. Τα Add-Ons που είναι απαραίτητα για την επίτευξη των παραπάνω λειτουργιών είναι τα ακόλουθα:

- Simulink
- Communication Toolbox Support Package for Xilinx Zynq-Based Radio
- Communications Toolbox
- HDL Coder
- HDL Coder Support Package for Xilinx Zynq
- Embedded Coder
- Simulink Coder
- Embedded Coder Support Package for Xilinx Zynq

Στην παρούσα εργασία χρησιμοποιήθηκε η έκδοση MatLab 2020b σε λογισμικό Windows . Η έκδοση αυτή του MatLab παρέχει προ εγκατεστημένα τα ακόλουθα Add-ons: Simulink, Communications Toolbox, HDL Coder, Embedded Coder, Simulink Coder. Τα υπόλοιπα χρειάζεται να εγκατασταθούν και να ρυθμιστούν κατάλληλα από τον χρήστη.

3.3.3.1 Communication Toolbox Support Package for Xilinx Zynq-Based Radio

1. Αρχικά, μεταβαίνουμε στην αρχική σελίδα της MathWorks[31] και δημιουργούμε ένα λογαριασμό χρήστη, προκειμένου να μας επιτραπεί η λήψη των απαραίτητων εργαλείων (Add-ons).
2. Στην καρτέλα “Home” του MatLab, πρέπει να επιλεχθεί “Add Ons > Get Hardware Support Packages”.

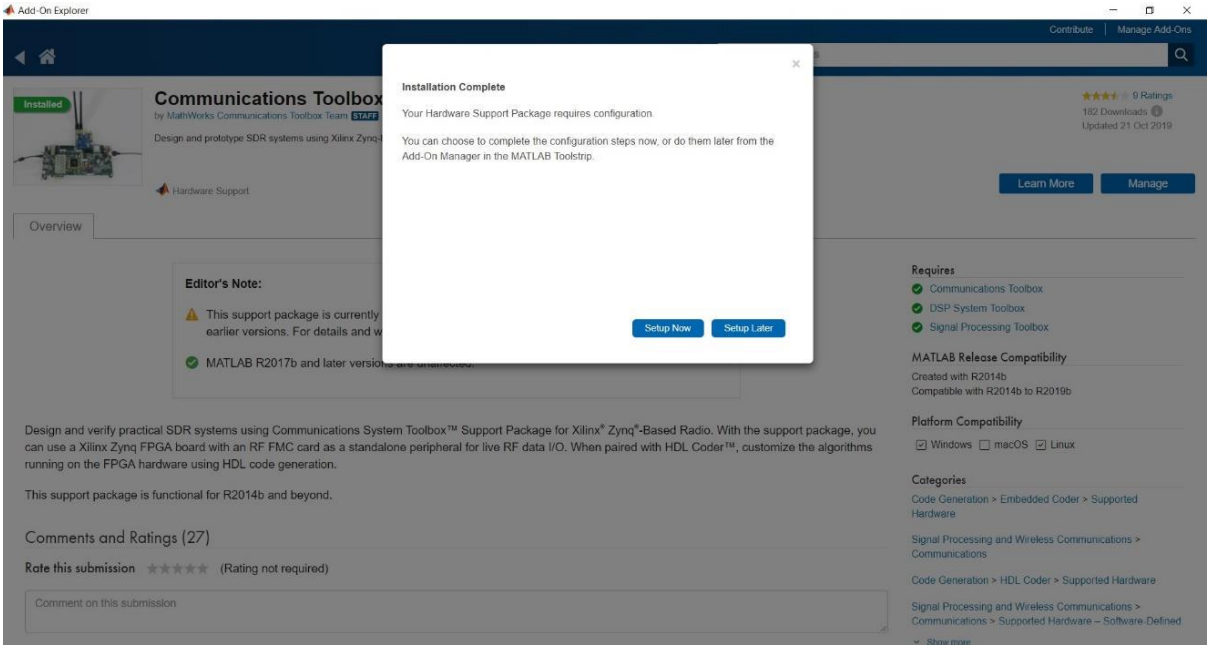


Εικόνα 3.3.3.1.1 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (1/23).

3. Στο πεδίο της αναζήτηση επάνω δεξιά του παραθύρου που άνοιξε, θα πρέπει να αναζητηθεί το "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" και αφού το επιλέξουμε προχωράμε στην εγκατάσταση του. Πριν την εγκατάσταση του εργαλείου, θα μας ζητήσει να εισάγουμε τα στοιχεία του λογαριασμού που έχουμε δημιουργήσει στην MathWorks (User Name , Password), διαφορετικά η εγκατάσταση δεν θα προχωρήσει.

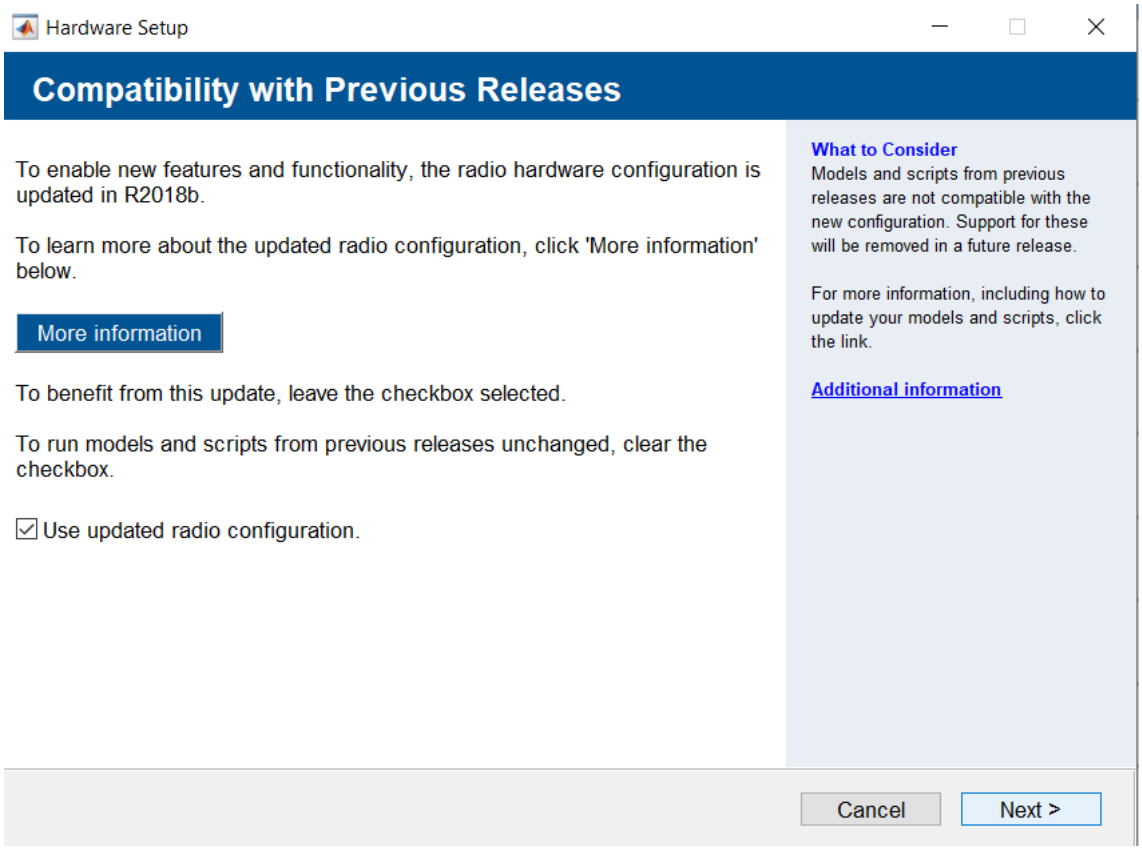
Εικόνα 3.3.3.1.2 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (2/23).

4. Μετά την ολοκλήρωση της εγκατάστασης για να ανοίξει ο οδηγός ο οποίος θα συνδέσει τις πλακέτες με το περιβάλλον του MatLab θα πρέπει να επιλεχθεί "Set Up Now".



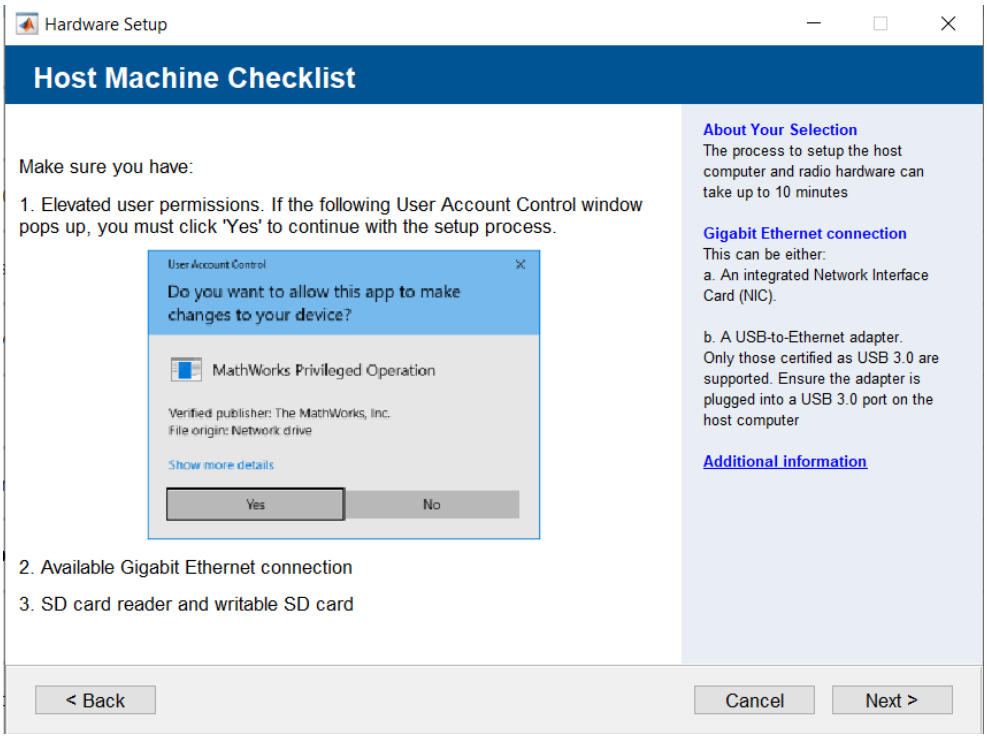
Εικόνα 3.3.3.1.3 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (3/23).

5. Στο παράθυρο που αφορά την ενημέρωση της συμβατότητας παλαιών προγραμματιστικών μοντέλων με τις νεότερες εκδόσεις του MatLab επιλέγεται "Next".



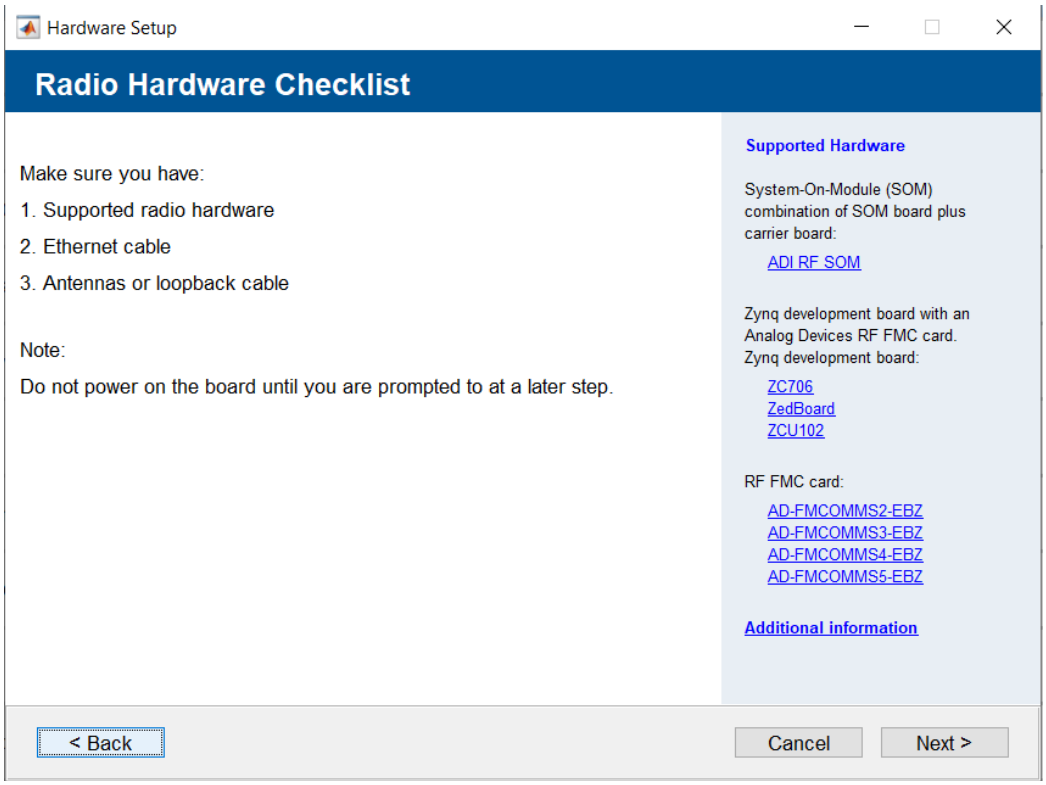
Εικόνα 3.3.3.1.4 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (4/23).

6. Στη συνέχεια επιλέγεται "Yes" στο αναδυόμενο παράθυρο των Windows που ζητάει την άδεια στο να γίνουν κάποιες αλλαγές σχετικά με την κάρτα δικτύου του υπολογιστή και έπειτα επιλέγεται "Next" στο παράθυρο διαλόγου του MatLab.



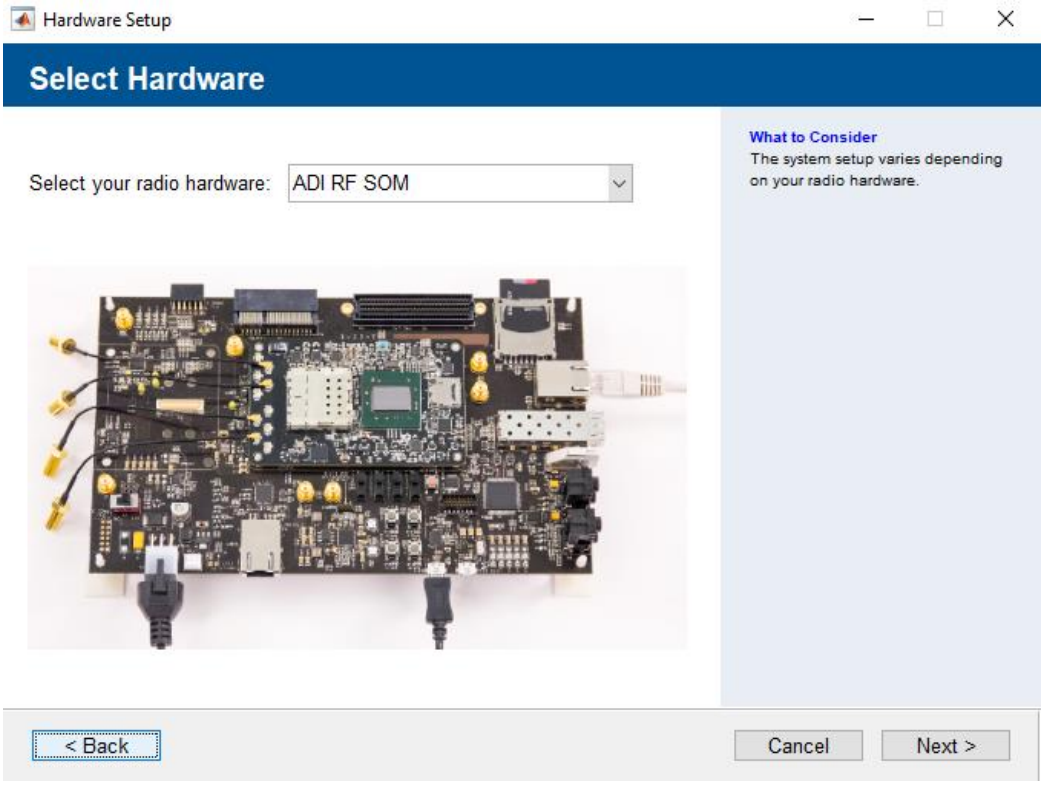
Εικόνα 3.3.3.1.5 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (5/23).

7. Σε αυτό το παράθυρο το MatLab ενημερώνει για τον προαπαιτούμενο εξοπλισμό, έτσι επιλέγεται "Next".

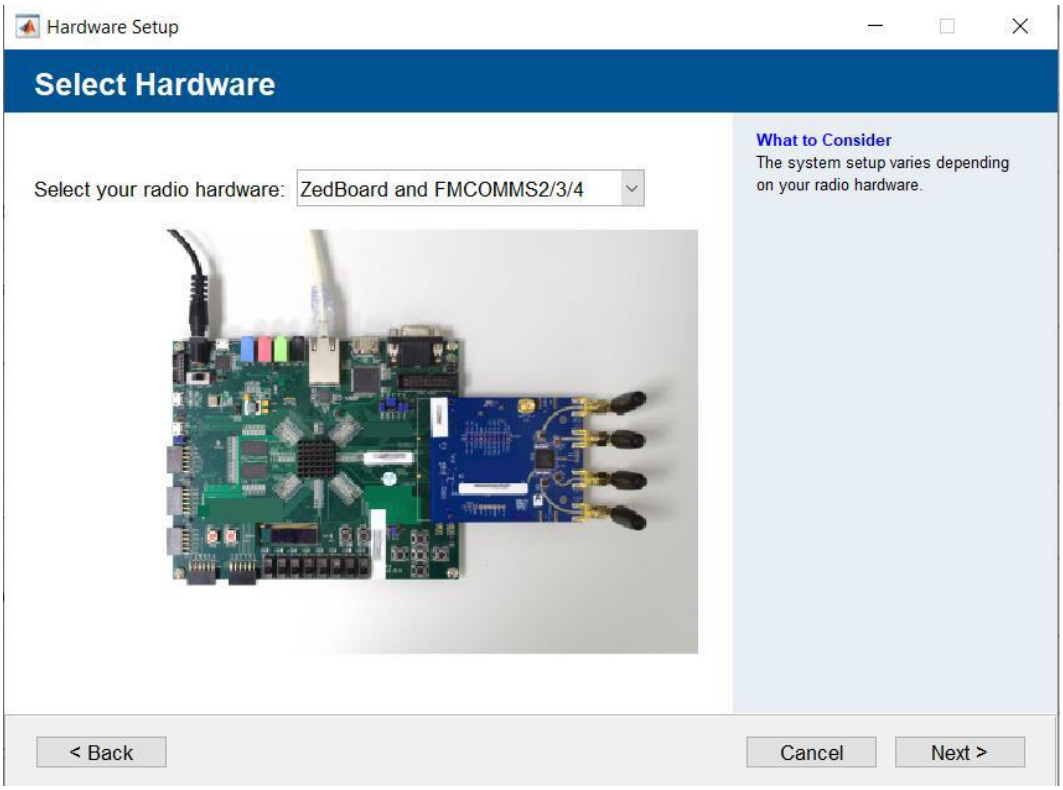


Εικόνα 3.3.3.1.6 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (6/23).

8. Αφού βρεθεί ο συνδυασμός των πλακετών που θα χρησιμοποιηθούν στο συγκεκριμένο παράθυρο όπως φαίνεται και στην παρακάτω εικόνα επιλέγεται "Next".

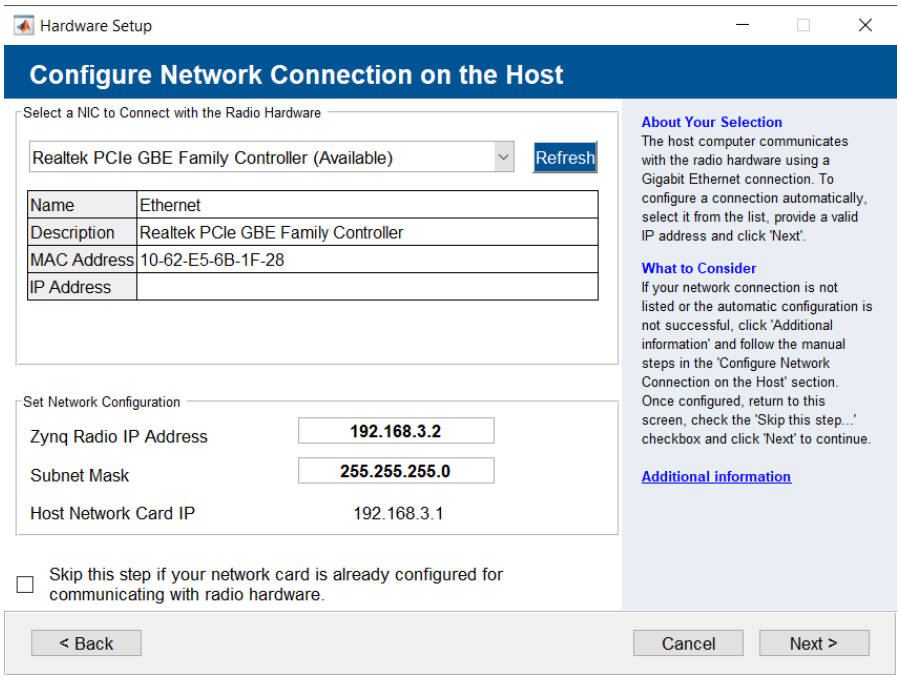


Εικόνα 3.3.3.1.7 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (7/23).



Εικόνα 3.3.3.1.8 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (8/23).

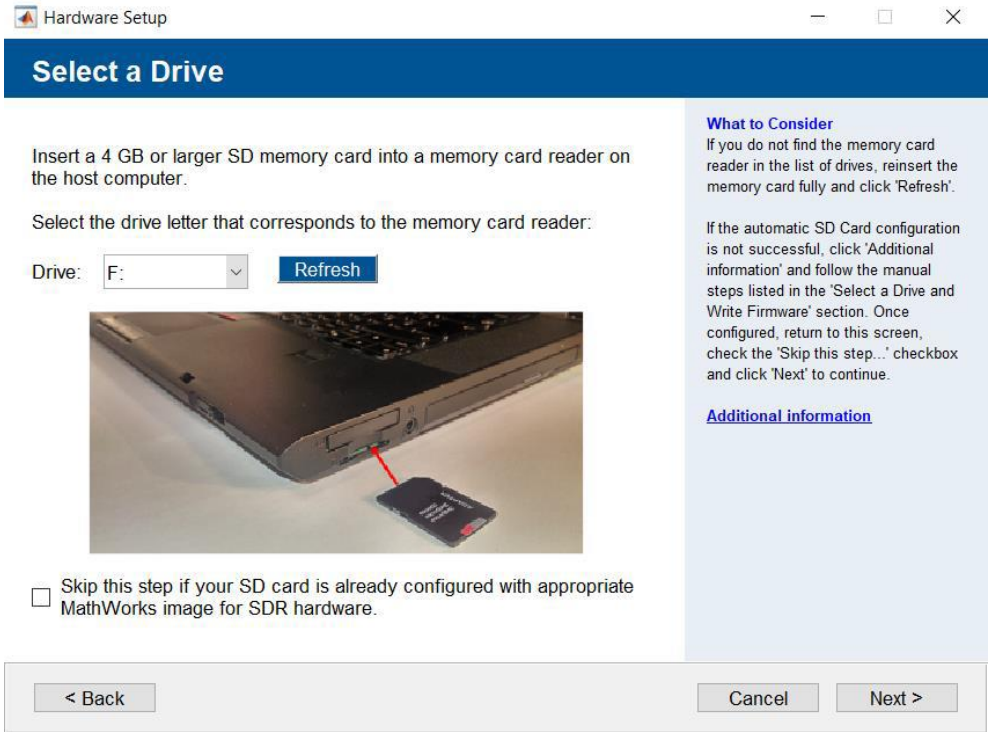
9. Σε αυτό το παράθυρο ο οδηγός που εκτελείτε, προσπαθεί να απονέμει την IP διεύθυνση "192.168.3.1" στον υπολογιστή, έτσι ώστε να μπορέσει να επικοινωνήσει ο υπολογιστής με την ZedBoard, όταν αυτή συνδεθεί. Έτσι εντοπίζεται η διαθέσιμη κάρτα δικτύου στην οποία ανήκει η χρησιμοποιούμενη θύρα Ethernet και στη συνέχεια θα επιλεχθεί "Next".



Εικόνα 3.3.3.1.9 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (9/23).

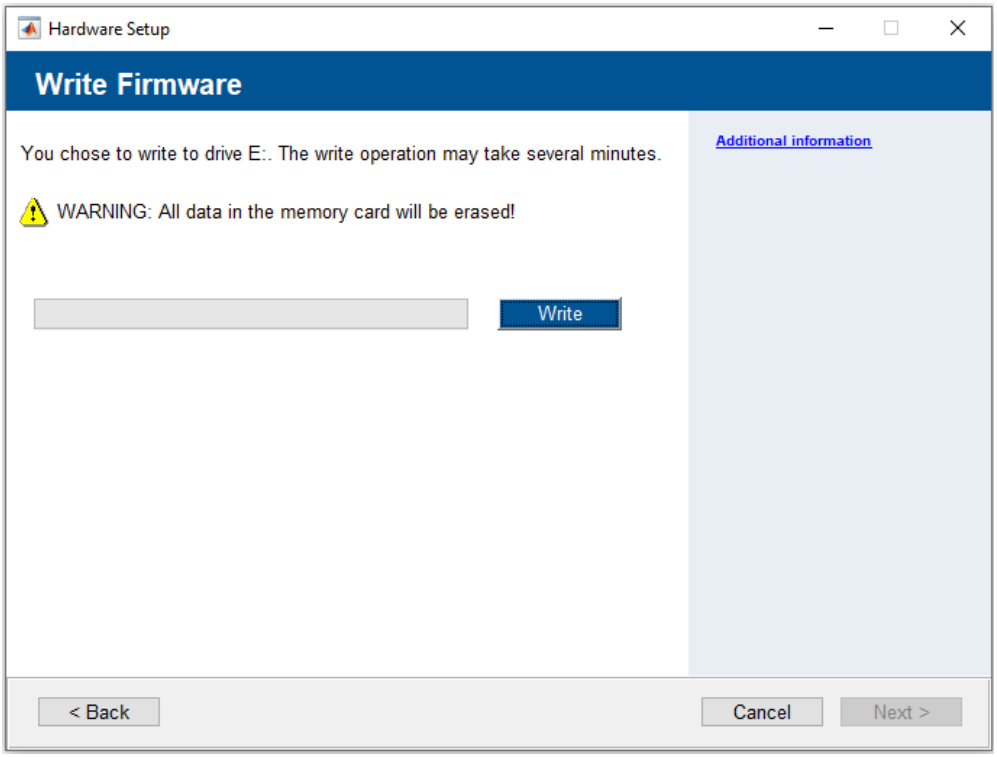
Σε περίπτωση που μέσα στην παρένθεση δίπλα από το όνομα της επιλεγόμενης κάρτας δικτύου αναγράφει "In Use" ,σημαίνει ότι η κάρτα αυτή έχει ήδη κάποια διεύθυνση IP χωρίς να χρησιμοποιείται. Αν επιλεγεί τέτοιου είδους κάρτα, συχνά ο οδηγός δεν είναι ικανός να αλλάξει αυτόματα αυτή την διεύθυνση κι έτσι θα πρέπει να σβηστεί η υπάρχουσα IP της κάρτας χειροκίνητα.

10. Τώρα θα χρειαστεί να συνδεθεί η SD κάρτα μνήμης στον υπολογιστή και να εντοπιστεί στο συγκεκριμένο παράθυρο ώστε στο επόμενο βήμα να γίνει η εγγραφή του απαιτούμενο firmware σε αυτή. Επιλέγεται "Next".

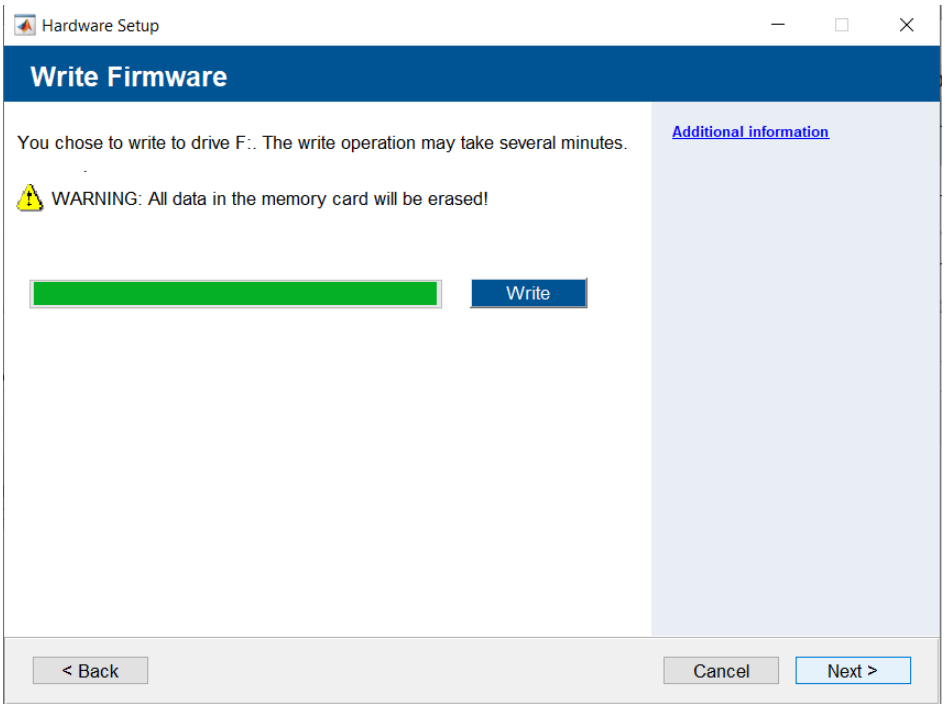


Εικόνα 3.3.3.1.10 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (10/23).

11. Σε αυτό το σημείο επιλέγοντας “Write” το firmware που θα είναι υπεύθυνο για την οδήγηση και σύνδεση των πλακετών με τον υπολογιστή θα φορτωθεί στην SD κάρτα μνήμης και μετά την ολοκλήρωσή του θα πρέπει να συνεχιστεί ο οδηγός επιλέγοντας “Next” .

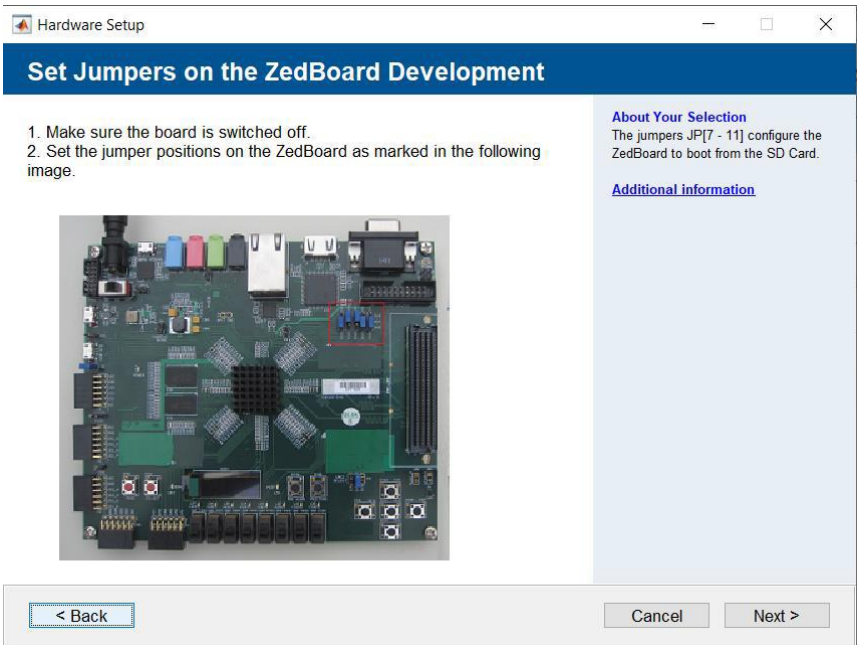


Εικόνα 3.3.3.1.11 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (11/23).



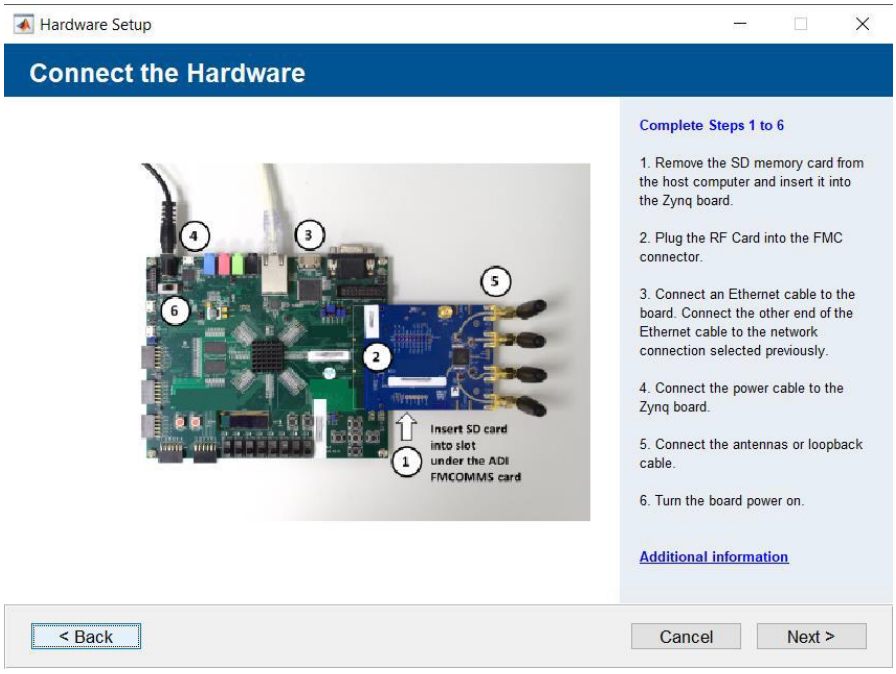
Εικόνα 3.3.3.1.12 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (12/23).

12. Επιλέγεται “Next” εφόσον ότι οι βραχυκυκλωτήρες (Jumper) του εικονιζόμενου σημείου της πλακέτας είναι σε αυτήν την θέση. Αυτό συμβαίνει με σκοπό την ρύθμιση της ZedBoard για να πραγματοποιήσει boot μέσω της κάρτας SD.



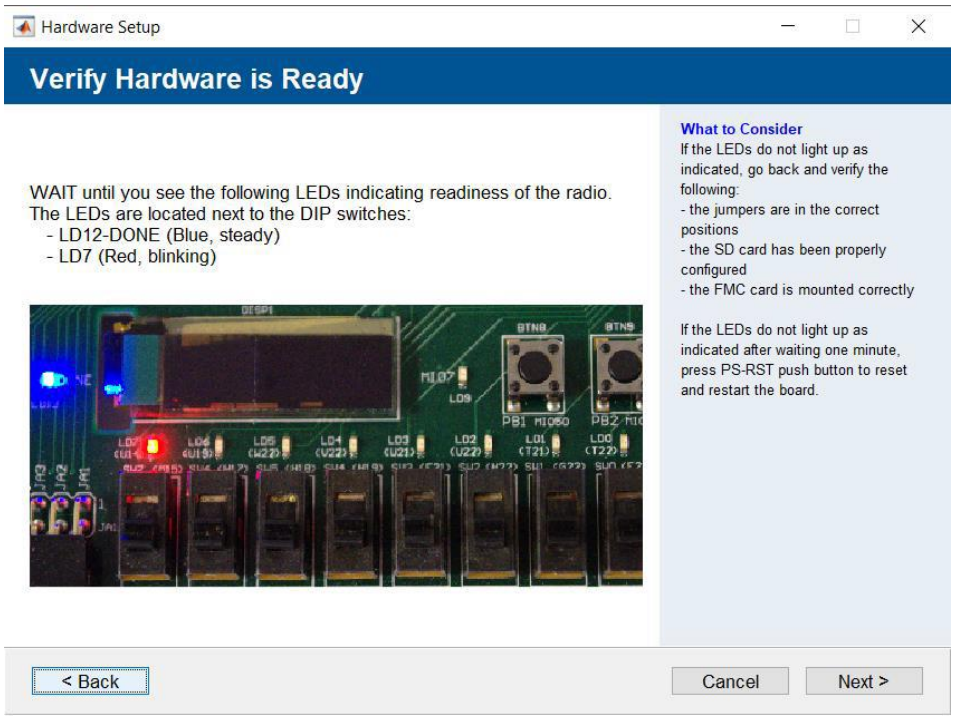
Εικόνα 3.3.3.1.13 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (13/23).

13. Η πλακέτα πλέον είναι έτοιμη να συνδεθεί με τον υπολογιστή. Έτσι αρχικά πρέπει πρώτα η SD κάρτα να συνδεθεί στην υποδοχή SD της ZedBoard και έπειτα οι θύρες Ethernet του υπολογιστή και της πλακέτας, θα πρέπει να συνδεθούν απευθείας μεταξύ τους με την χρήση Ethernet καλωδίου. Στη συνέχεια θα πρέπει να βιδωθούν επάνω στην RF διάταξη, συγκεκριμένα στις υποδοχές TxA και RxA, οι κατάλληλες κεραίες που θα χρησιμοποιηθούν και τέλος η πλακέτα μαζί με το τροφοδοτικό της συνδέεται στο ρεύμα. Έπειτα ενεργοποιείτε η πλακέτα μέσω του On/Off διακόπτη της και επιλέγεται "Next" στο παράθυρο του MatLab.



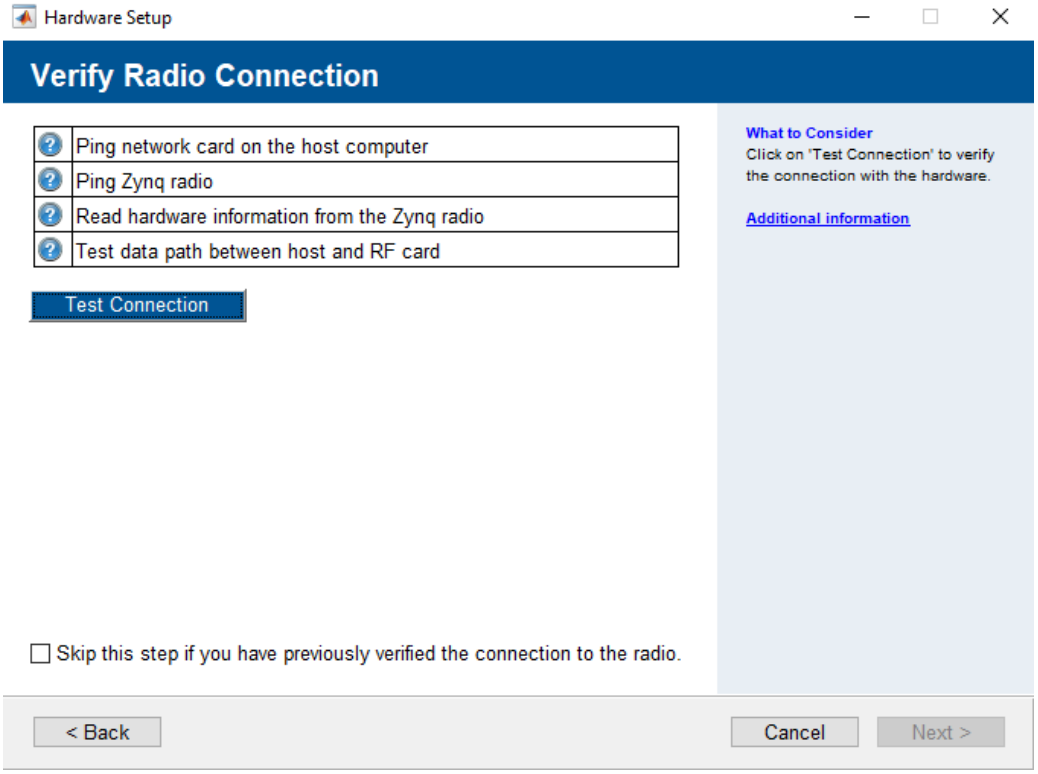
Εικόνα 3.3.3.1.14 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (14/23).

14. Επιλέγεται "Next" εφόσον το κόκκινο LED (LD7) της πλακέτας ξεκινήσει να αναβοσβήνει διαρκώς. Αυτό σημαίνει ότι η πλακέτα κατάφερε επιτυχώς να πραγματοποιήσει boot μέσω της SD κάρτας και να διαβάσει το φορτωμένο.

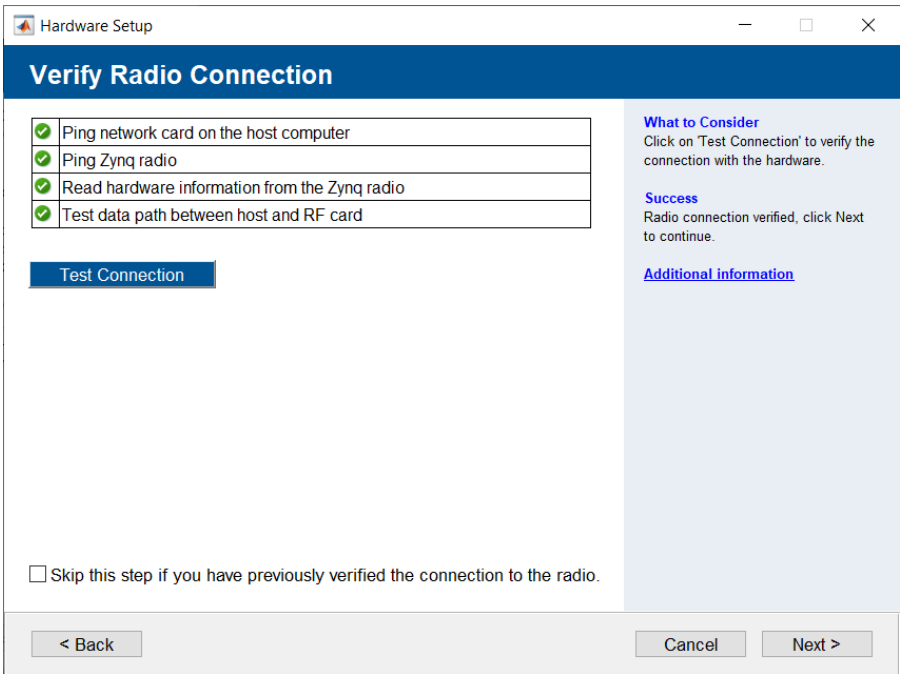


Εικόνα 3.3.3.1.15 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (15/23).

15. Με σκοπό, τον έλεγχο αρχικά της σύνδεσης της ZedBoard με τον υπολογιστή, της επικοινωνίας μεταξύ ZedBoard και AD-FMCOMMS4-EBZ πλακέτας, της επικοινωνίας μεταξύ της AD-FMCOMMS4-EBZ πλακέτας και του υπολογιστή καθώς και τον έλεγχο της λήψης και αποστολής δεδομένων μέσω των πλακετών και των κεραιών, θα πρέπει να επιλεχθεί το "Test Communication" και αφού πραγματοποιηθούν οι τέσσερις έλεγχοι με επιτυχία, επιλέγεται "Next".

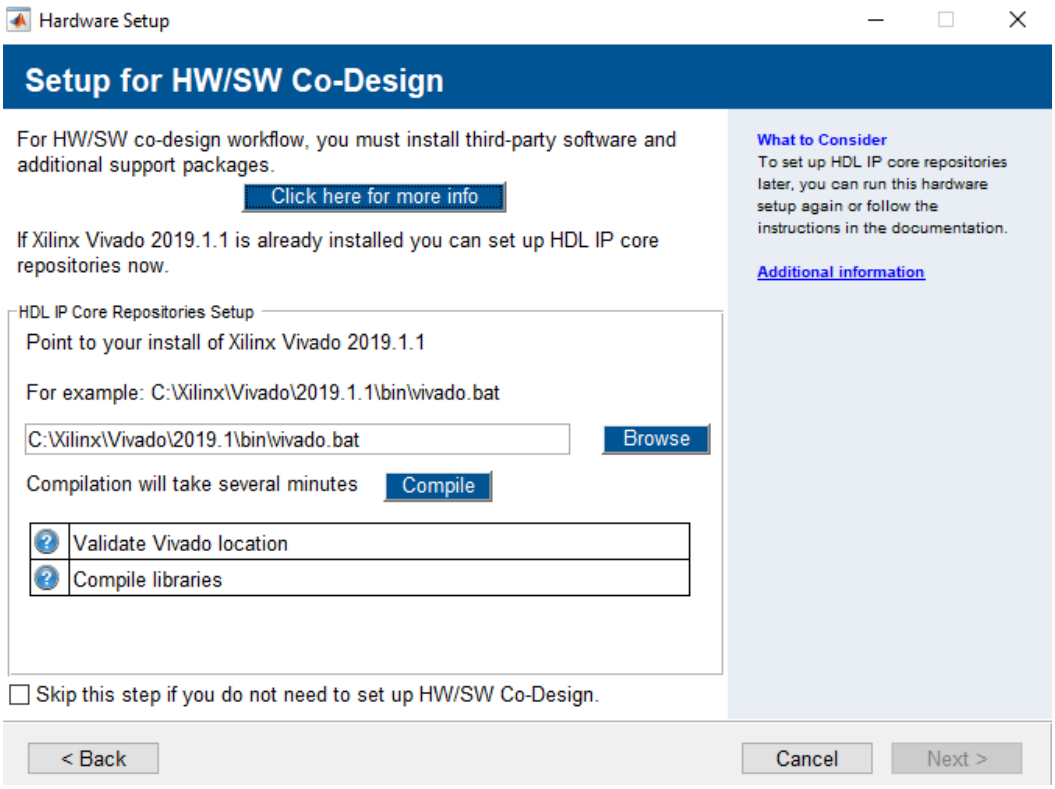


Εικόνα 3.3.3.1.16 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (16/23).

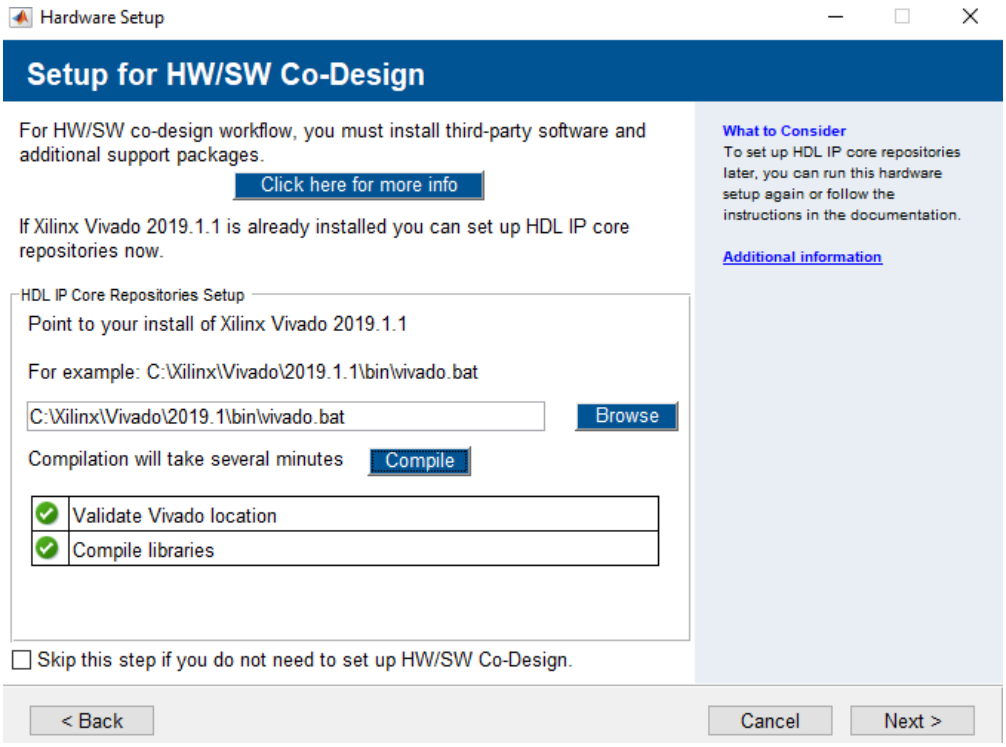


Εικόνα 3.3.3.1.17 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (17/23).

16. Στην συνέχεια θα πρέπει να καθοριστεί η διαδρομή στην οποία έχει γίνει εγκατάσταση του Vivado στον υπολογιστή. Η έκδοση του Vivado που απαιτείται να έχει εγκατασταθεί στον υπολογιστή για την πλακέτα ZedBoard είναι η 2019.1.1. Στην περίπτωση που έχει εγκατασταθεί διαφορετική έκδοση, η εγκατάσταση θα προχωρήσει αλλά με μία προειδοποίηση "Warning" ,ότι ενδεχομένως να υπάρξουν προβλήματα συμβατότητας με την πλακέτα ZedBoard, κατά την φάση υλοποίησης του παραγόμενου μοντέλου, σε επόμενα στάδια.

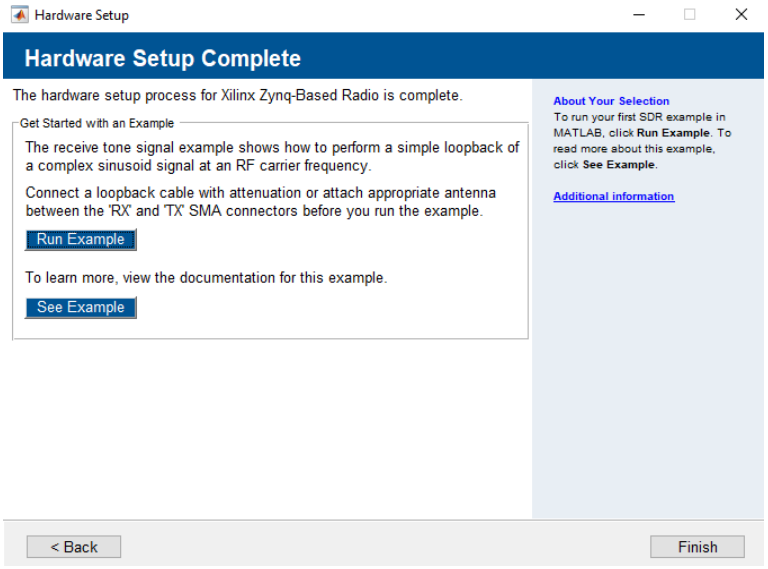


Εικόνα 3.3.3.1.18 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (18/23).



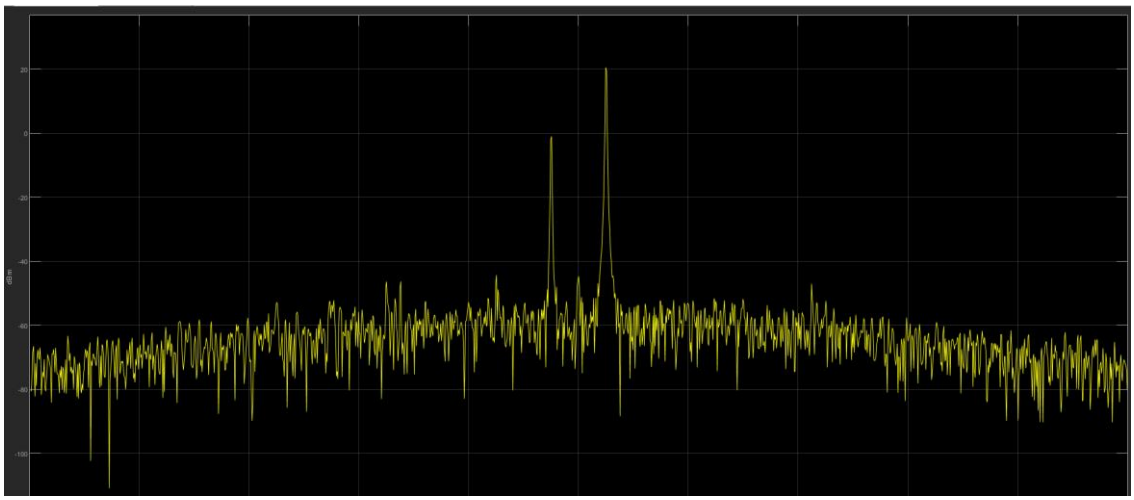
Εικόνα 3.3.3.1.19 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (19/23).

Στην συνέχεια επιλέγοντας το "Next", η εγκατάσταση έχει ολοκληρωθεί επιτυχώς και παρέχονται κάποια παραδείγματα υλοποίησης συστημάτων SDR με την χρήση της πλακέτας ZedBoard και του συγκεκριμένου Add-on.
Επιπλέον, στην καρτέλα αυτή, παρέχονται κάποια έτοιμα παραδείγματα και επιπλέον πληροφορίες σχετικά με την αναπτυξιακή πλακέτα ZedBoard.

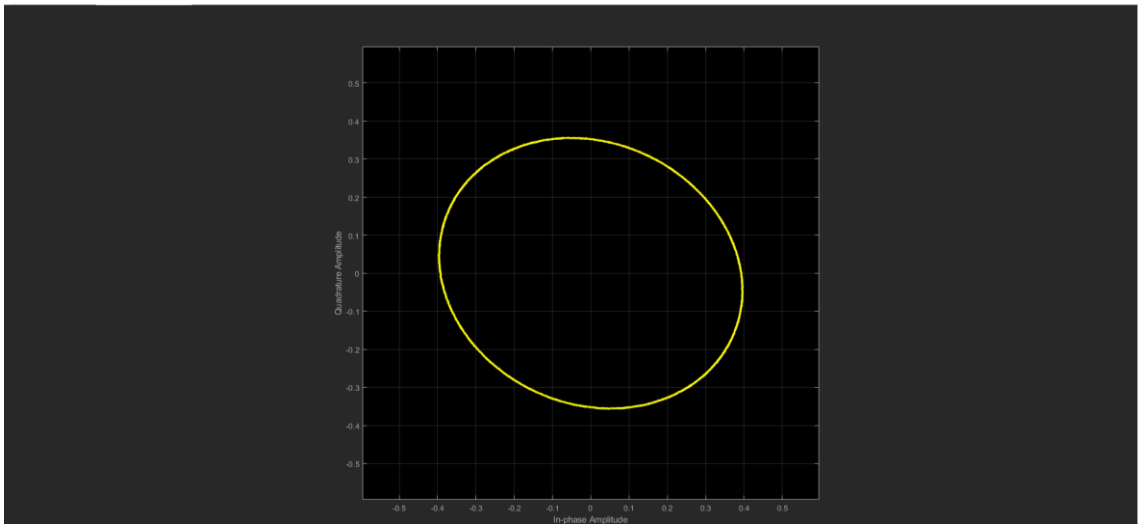


Εικόνα 3.3.3.1.20 Οδηγός εγκατάστασης "Communication Toolbox Support Package for Xilinx Zynq-Based Radio" (20/23).

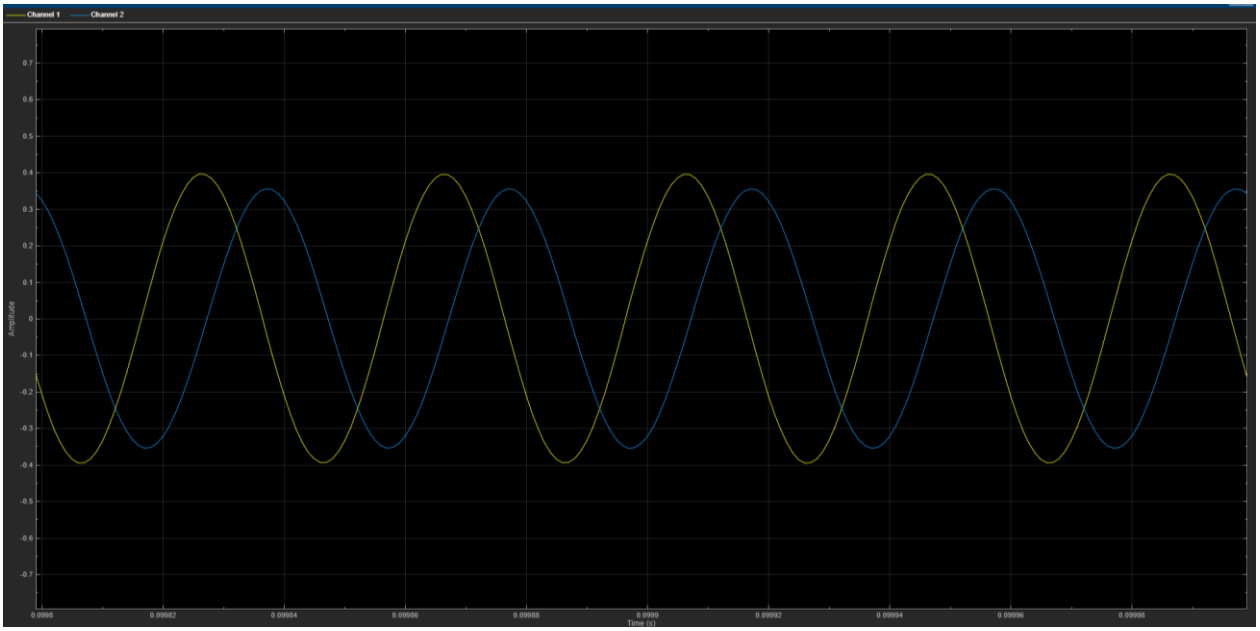
Ακολουθώντας τα βήματα που περιγράφονται αναλυτικά στο συγκεκριμένο παράδειγμα, λαμβάνουμε κάποιες κυματομορφές που επιβεβαιώνουν την ορθή λειτουργία του συγκεκριμένου Add-on, την σωστή επικοινωνία του ηλεκτρονικού υπολογιστή και του περιβάλλον του MatLab, με την πλακέτα ZedBoard και την ορθή λειτουργία της RF πλακέτας AD-FMCOMMS4-EBZ (ορθή λειτουργία πομπού και δέκτη).



Εικόνα 3.3.3.1.21 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (21/23).



Εικόνα 3.3.3.1.22 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (22/23).



Εικόνα 3.3.3.1.23 Οδηγός εγκατάστασης “Communication Toolbox Support Package for Xilinx Zynq-Based Radio” (23/23).

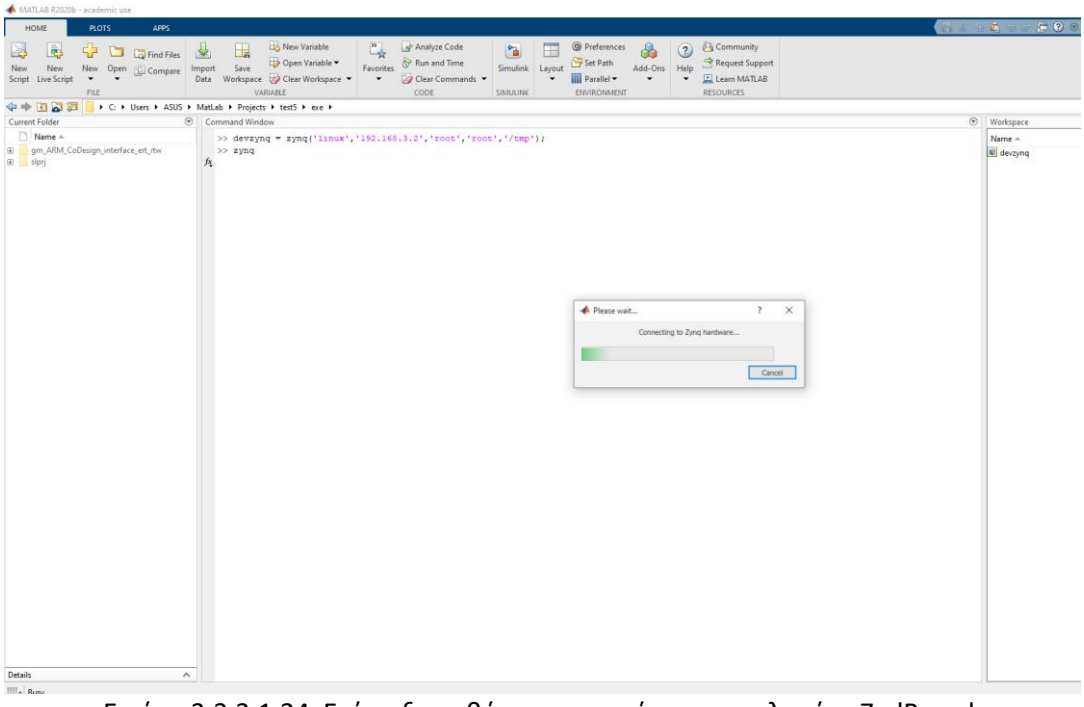
Πλέον, μέσω της εκτέλεσης κώδικα MatLab (script) ή μοντέλων του Simulink και συγκεκριμένων εντολών που καθορίζουν την συμπεριφορά των πλακετών, μπορούν να δημιουργηθούν οποιαδήποτε τηλεπικοινωνιακά πρωτόκολλα καθώς και συστήματα, χωρίς την αλλαγή κάθε φορά Hardware.
Προκειμένου να επιβεβαιώσουμε ότι όντως η πλακέτα ZedBoard επικοινωνεί με τον υπολογιστή και το MatLab, μεταβαίνουμε στην επιφάνια εργασίας του MatLab και στο Command Window πληκτρολογούμε τις εντολές:

```
devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
zynq;
```

Η εντολή «zynq» εμφανίζει τη διεύθυνση IP ή το όνομα του υπολογιστή, το όνομα χρήστη και τον κωδικό πρόσβασης της πιο πρόσφατης επιτυχούς σύνδεσης του MatLab σε την πλακέτα Zynq. Αυτές οι πληροφορίες δεν προέρχονται από το τρέχουσα πλατφόρμα Xilinx® Zynq που είναι συνδεδεμένη στον υπολογιστή. Επομένως, εάν έχει γίνει οποιαδήποτε αλλαγή στις ρυθμίσεις της πλατφόρμας Xilinx Zynq από την τελευταία επιτυχημένη σύνδεση, οι πληροφορίες δεν θα είναι ενημερωμένες.
Η εντολή «devzynq = zynq» δημιουργεί μια σύνδεση με την πλατφόρμα Xilinx Zynq, devzynq. Αυτή η σύνδεση επαναχρησιμοποιεί τη διεύθυνση IP, το όνομα χρήστη και τον κωδικό πρόσβασης από την πιο πρόσφατη σύνδεση στην πλατφόρμα Xilinx Zynq.
Η ορθή σύνταξη της εντολής είναι:

```
devzynq= zynq(operatingsystem,ipaddress,username,password,builddir)
```

και επιτρέπει τον καθορισμό μη προεπιλεγμένων τιμών για διεύθυνση IP ή όνομα του υπολογιστή, όνομα χρήστη, κωδικό πρόσβασης, φάκελο στην πλατφόρμα Xilinx Zynq.
Εφόσον έχουμε ολοκληρώσει με επιτυχία τα παραπάνω βήματα και η πλακέτα είναι ενεργοποιημένη και συνδεμένη με τον υπολογιστή, με την σωστή διεύθυνση δικτύου IP, θα λάβουμε μια απάντηση επιβεβαίωσης, όπως φαίνεται στις εικόνες 3.3.3.1.24 και 3.3.3.1.25.
Η εντολή «zynq» μπορεί να χρησιμοποιηθεί σε οποιαδήποτε χρονική στιγμή, εφόσον έχουμε κάποια αμφιβολία, για την ορθή επικοινωνία του υπολογιστή με την αναπτυξιακή πλακέτα ZedBoard.



Εικόνα 3.3.3.1.24 Επίτευξη ορθής επικοινωνία με την πλακέτα ZedBoard.

```
Command Window
>> devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
>> zynq

ans =

  LinuxShell with properties:

    IPAddress: '192.168.3.2'
     Username: 'root'
         Port: 22

fx >>
```

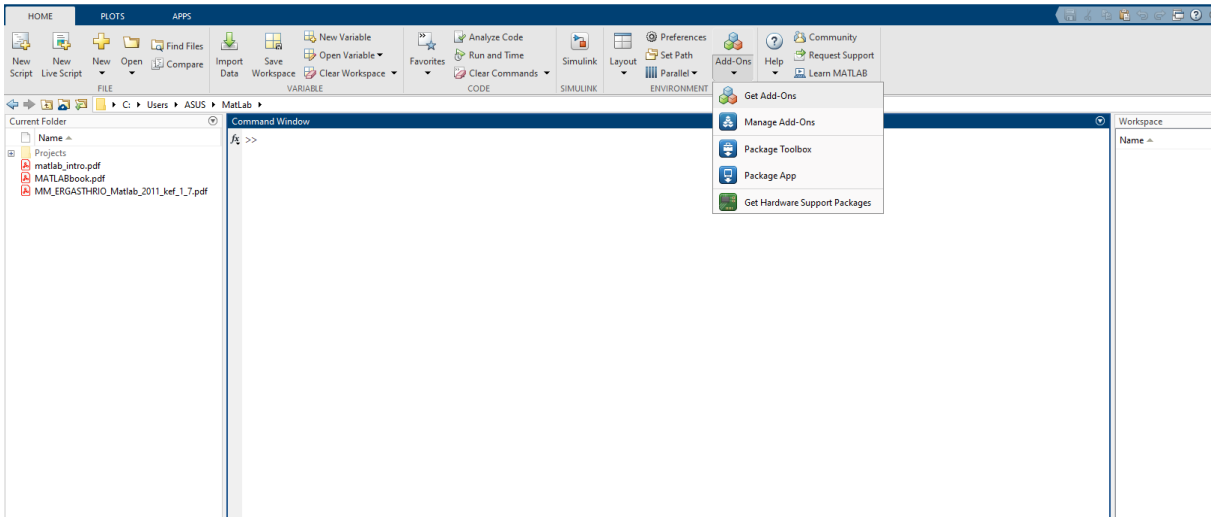
Εικόνα 3.3.3.1.25 Επίτευξη ορθής επικοινωνία με την πλακέτα ZedBoard.

```
Command Window
>> devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
>> zynq
Cannot connect to Zynq Hardware at IP Address "192.168.3.2".
Click here to view the troubleshooting guide and verify the IP Address settings of your hardware.
Error using codertarget.zynq.internal.LinuxShell/setupZynqHardware (line 548)
Could not connect to the hardware over Serial Port.
Check that USB cable is plugged into 'USB-to-UART' port of the hardware and the power switch is turned on.
If this error persists, switch off the hardware, wait for 10 seconds, switch on the hardware and then repeat this step.

Error in codertarget.zynq.internal.LinuxShell (line 227)
                    obj.setupZynqHardware(username,password,remotedir);

Error in zynq (line 113)
    h = codertarget.zynq.internal.LinuxShell('ZC702', varargin{:});

fx >>
```

Εικόνα 3.3.3.1.26 Αδυναμία επικοινωνίας με την πλακέτα ZedBoard.

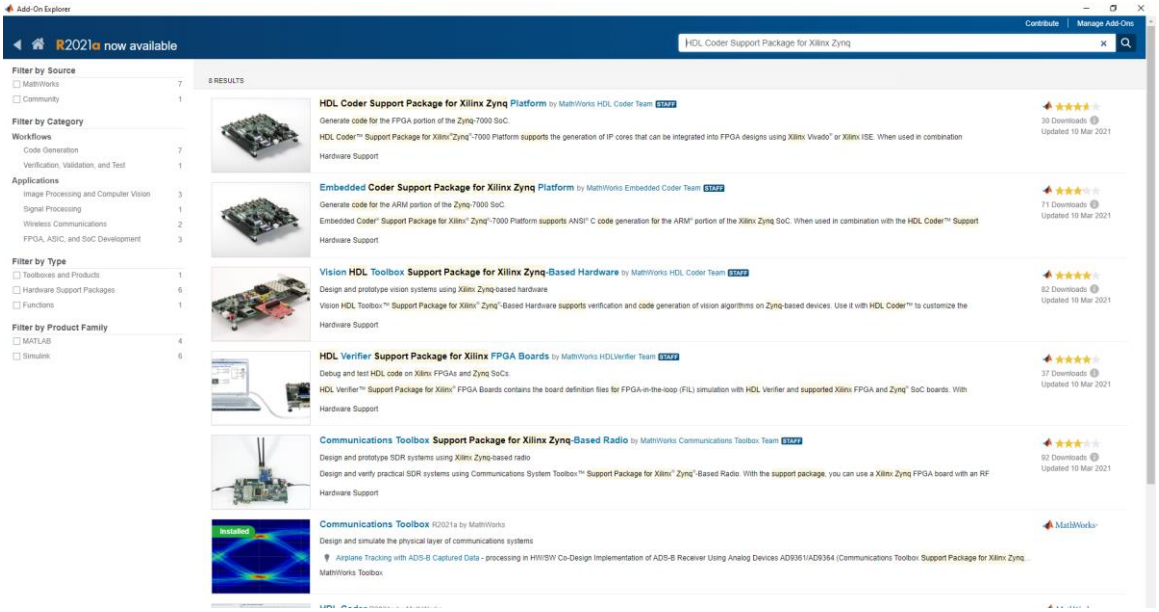
3.3.3.2 HDL Coder Support Package for Xilinx Zynq

1. Αρχικά στην καρτέλα “Home” του MatLab, πρέπει να επιλεχθεί “Add Ons > Get Hardware Support Packages”.

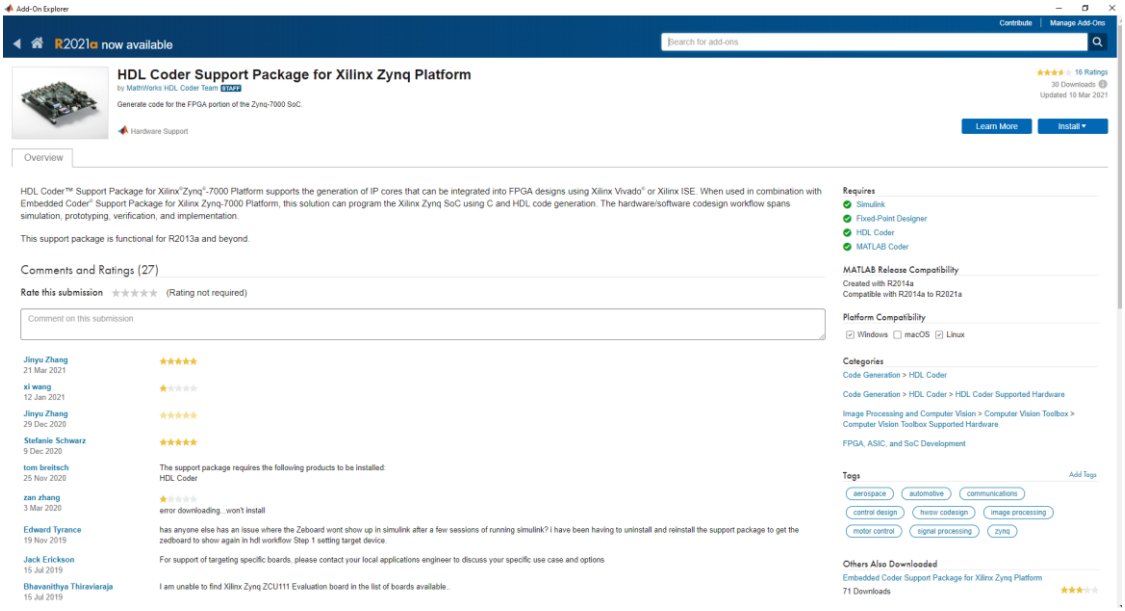


Εικόνα 3.3.3.2.1 Οδηγός εγκατάστασης “HDL Coder Support Package for Xilinx Zynq” (1/7).

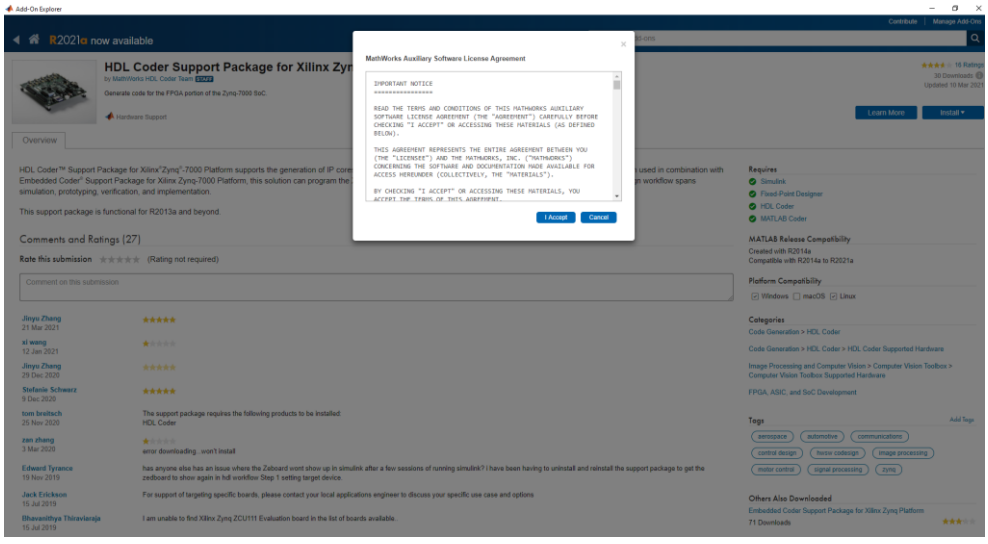
2. Στην αναζήτηση επάνω δεξιά του παραθύρου που ανοίξει, θα πρέπει να αναζητηθεί το “HDL Coder Support Package for Xilinx Zynq” και να εγκατασταθεί.



Εικόνα 3.3.3.2.2 Οδηγός εγκατάστασης “HDL Coder Support Package for Xilinx Zynq” (2/7).

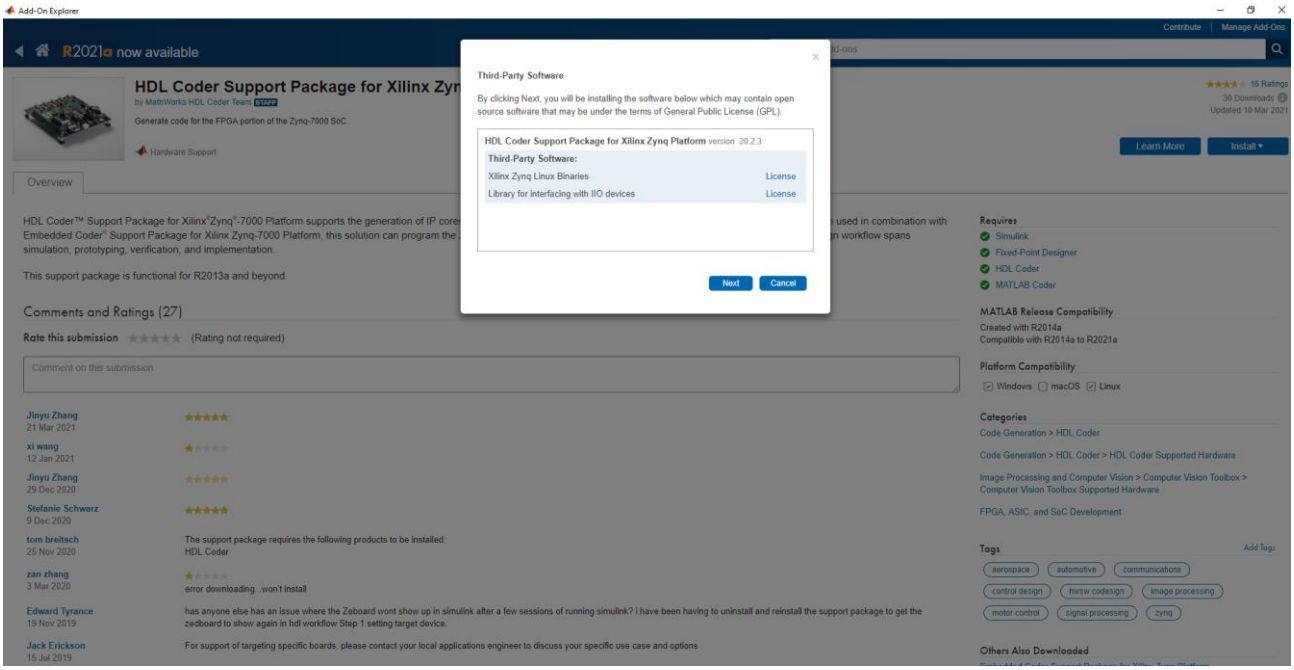


Εικόνα 3.3.3.2.3 Οδηγός εγκατάστασης “HDL Coder Support Package for Xilinx Zynq” (3/7).

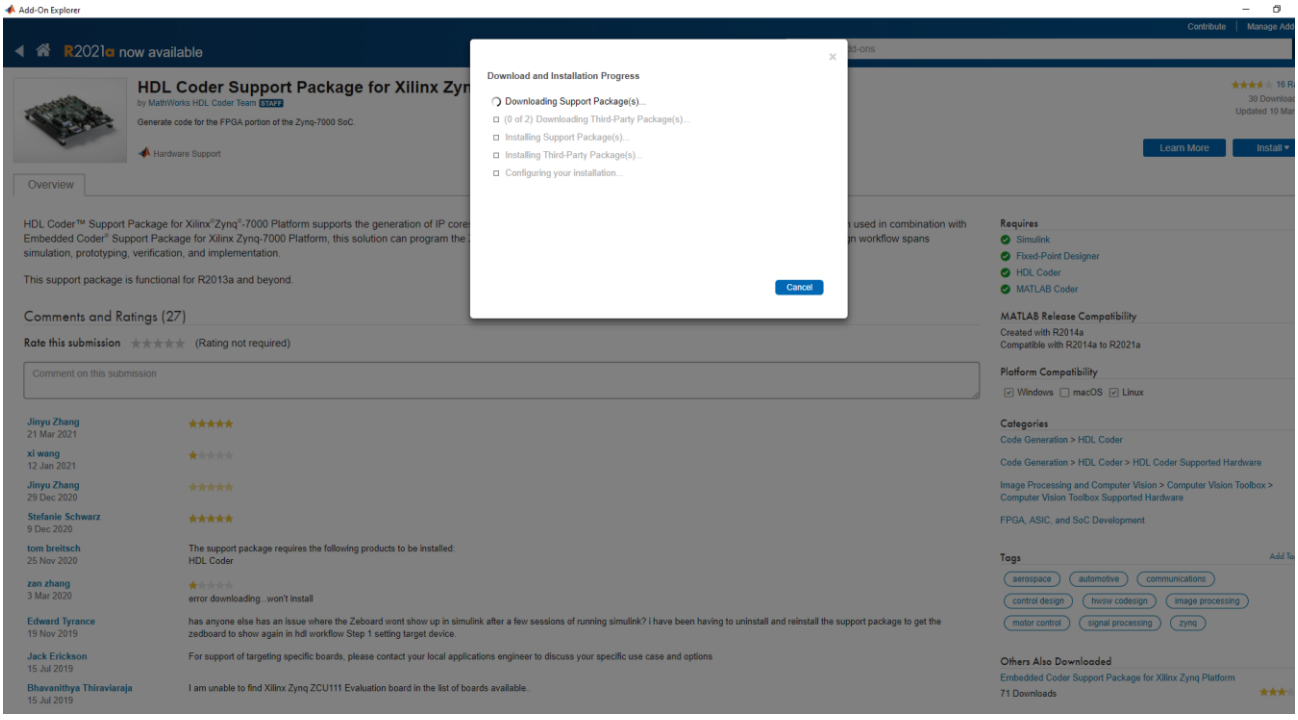


Εικόνα 3.3.3.2.4 Οδηγός εγκατάστασης “HDL Coder Support Package for Xilinx Zynq” (4/7).

3. Επιλέγοντας το "Next", θα εγκατασταθούν κάποια προγράμματα άλλων κατασκευαστών (Third-Party Software), τα οποία είναι απαραίτητη για την εύρυθμη λειτουργία του συγκεκριμένου εργαλείου (Add-on).

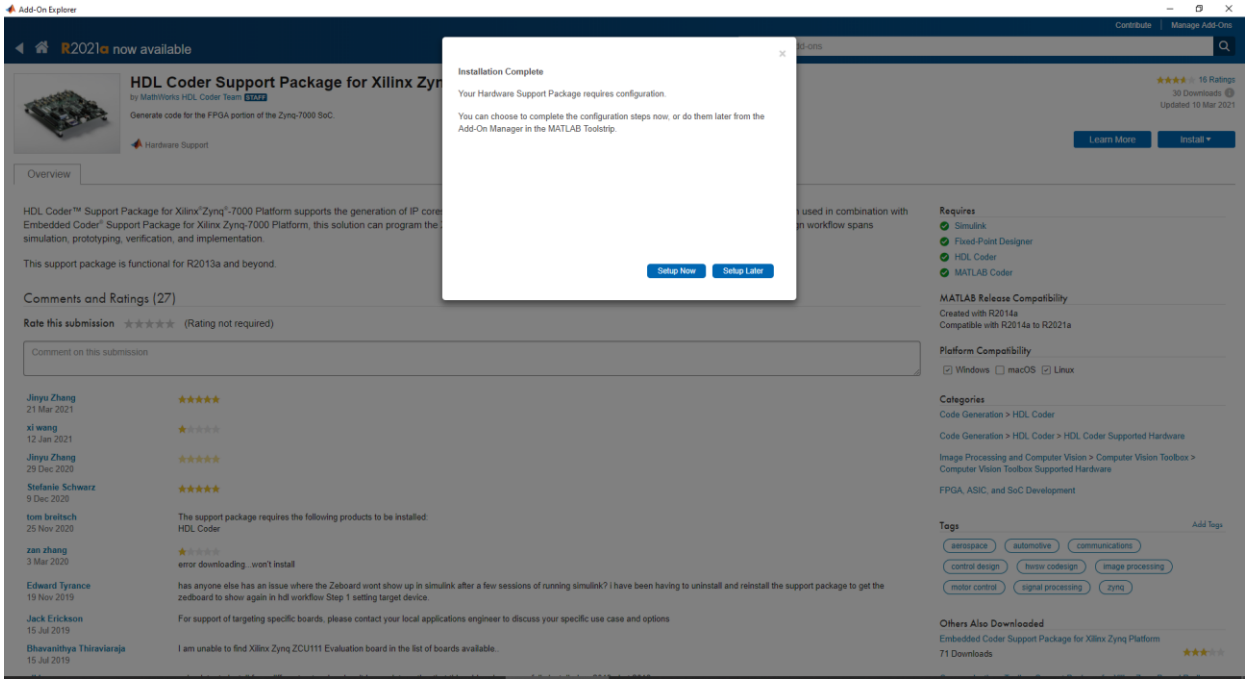


Εικόνα 3.3.3.2.5 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (5/7).



Εικόνα 3.3.3.2.6 Οδηγός εγκατάστασης "HDL Coder Support Package for Xilinx Zynq" (6/7).

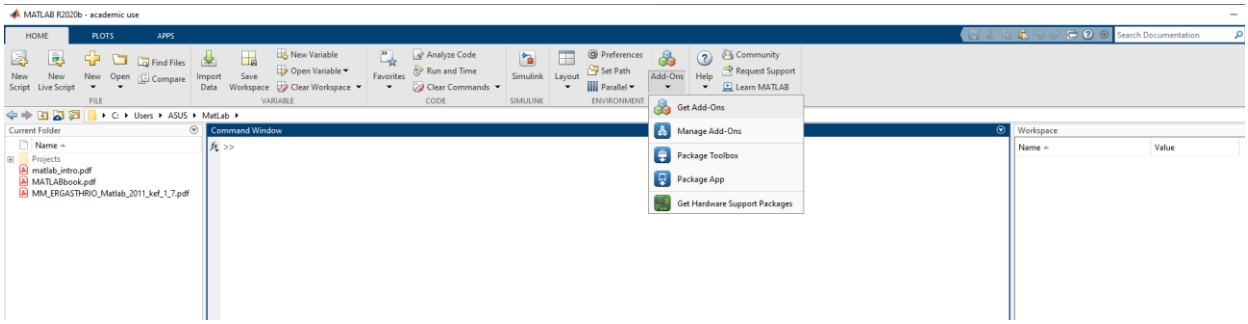
4. Μόλις ολοκληρωθεί η λήψη και η εγκατάσταση των προγραμμάτων άλλων κατασκευαστών (Third-Party Software), η εγκατάσταση του " HDL Coder Support Package for Xilinx Zynq" έχει ολοκληρωθεί. Στο σημείο αυτό θα ζητηθεί να επιλέξουμε αν θα προχωρήσουμε στην ρύθμιση/παραμετροποίηση του συγκεκριμένου εργαλείου τώρα ή αργότερα. Αν έχει πραγματοποίηση η ρύθμιση/παραμετροποίηση του "Communication Toolbox Support Package for Xilinx Zynq-Based Radio ", όπως παρουσιάστηκε στο κεφάλαιο 3.3.3.1, τότε αυτό δεν απαιτείται, και επομένως επιλέγουμε " SetUp Later".



Εικόνα 3.3.3.2.7 Οδηγός εγκατάστασης “HDL Coder Support Package for Xilinx Zynq” (7/7).

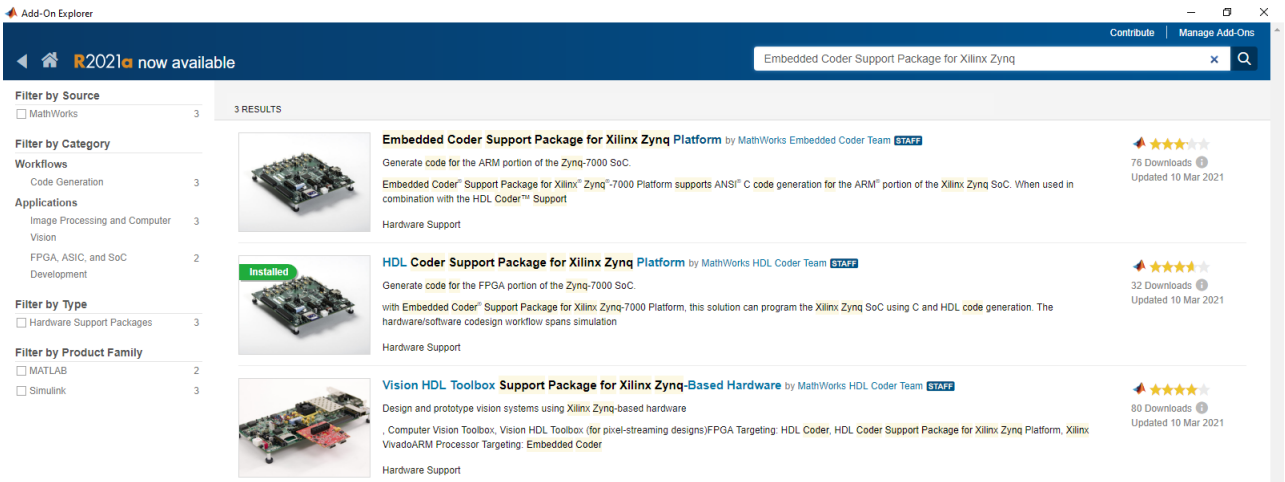
3.3.3.3 Embedded Coder Support Package for Xilinx Zynq

1. Αρχικά στην καρτέλα “Home” του MatLab, πρέπει να επιλεχθεί “Add Ons > Get Hardware Support Packages”.

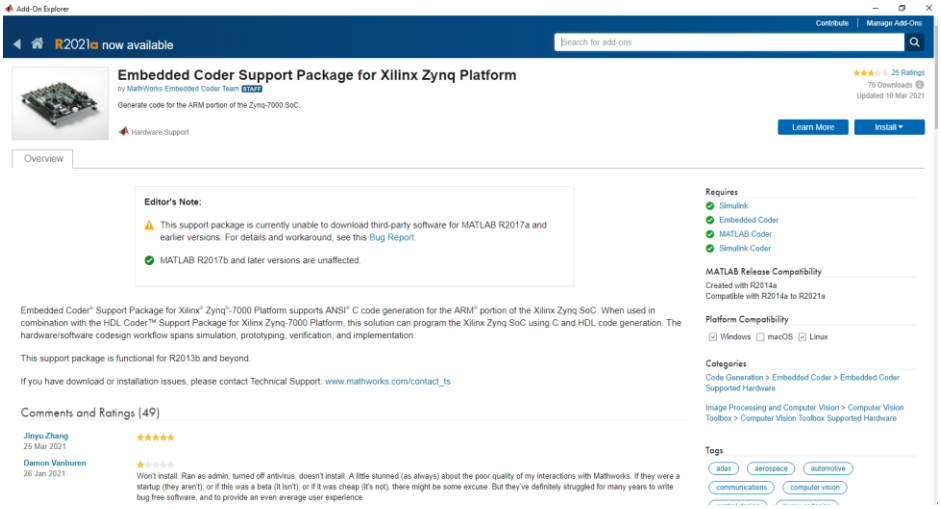


Εικόνα 3.3.3.3.1 Οδηγός εγκατάστασης “Embedded Coder Support Package for Xilinx Zynq” (1/7).

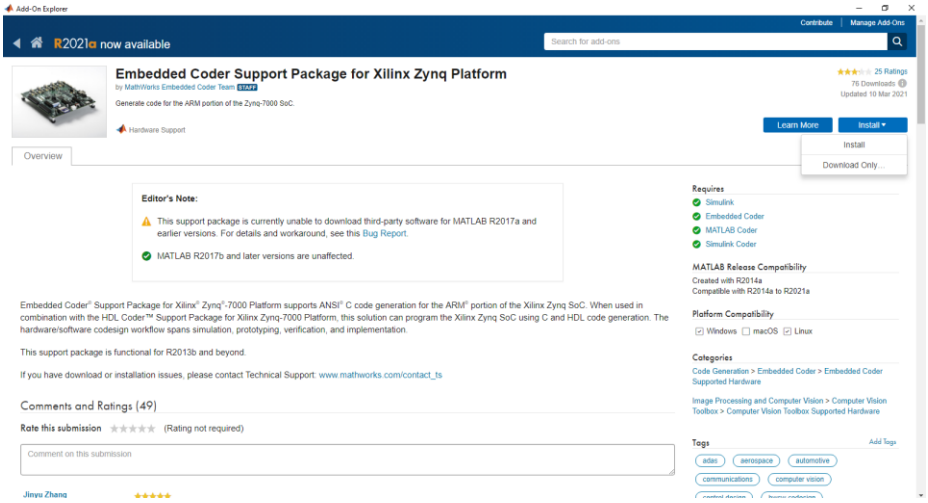
2. Στην αναζήτηση επάνω δεξιά του παραθύρου που άνοιξε, θα πρέπει να αναζητηθεί το “Embedded Coder Support Package for Xilinx Zynq” και να εγκατασταθεί.



Εικόνα 3.3.3.3.2 Οδηγός εγκατάστασης “Embedded Coder Support Package for Xilinx Zynq” (2/7).

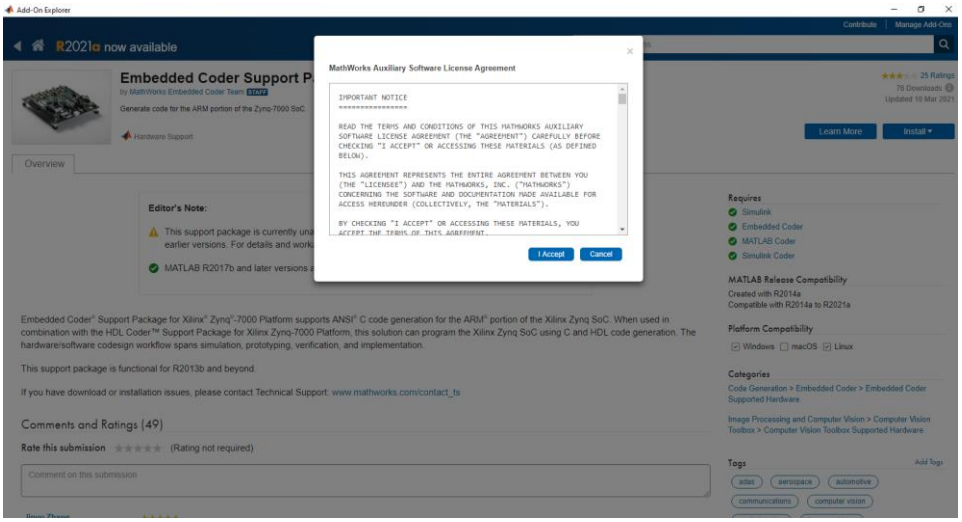


Εικόνα 3.3.3.3.3 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (3/7).



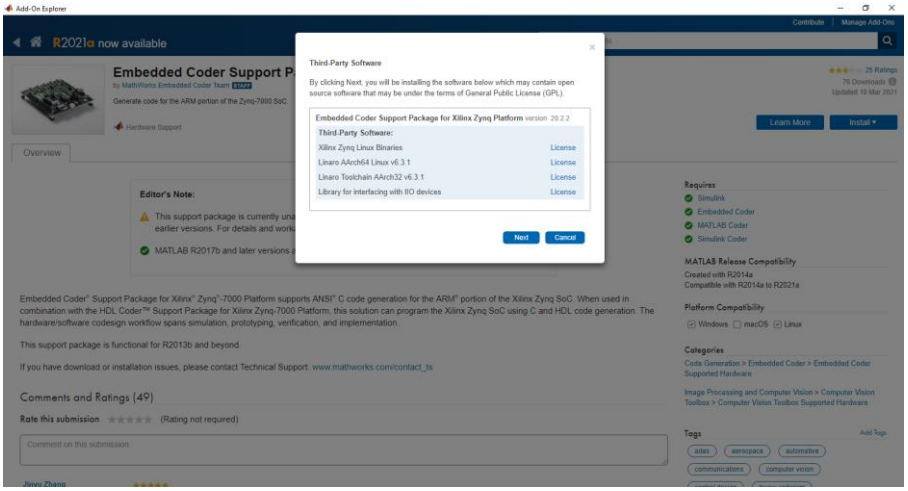
Εικόνα 3.3.3.3.4 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (4/7).

3. Πατώντας την επιλογή "Accept", ξεκινάει η εγκατάσταση.

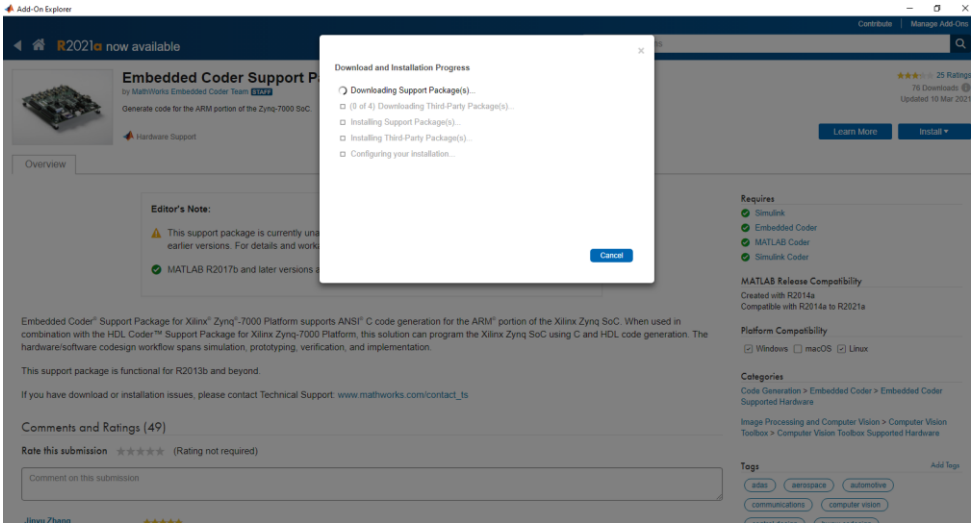


Εικόνα 3.3.3.3.5 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (5/7).

5. Επιλέγοντας το "Next", θα εγκατασταθούν κάποια προγράμματα άλλων κατασκευαστών (Third-Party Software), τα οποία είναι απαραίτητη για την εύρυθμη λειτουργία του συγκεκριμένου εργαλείου (Add-on).



Εικόνα 3.3.3.3.6 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (6/7).



Εικόνα 3.3.3.3.7 Οδηγός εγκατάστασης "Embedded Coder Support Package for Xilinx Zynq" (7/7).

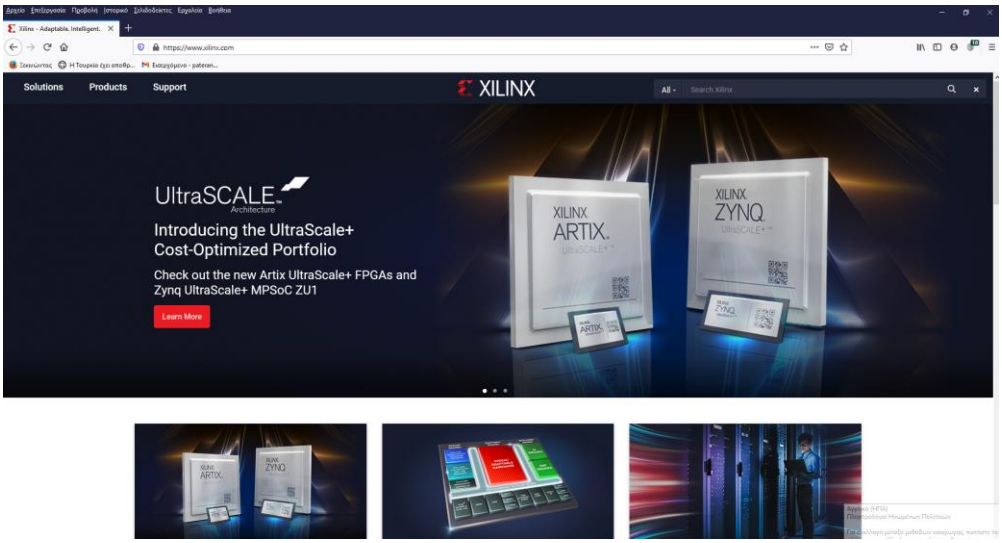
6. Μόλις ολοκληρωθεί η λήψη και η εγκατάσταση των προγραμμάτων άλλων κατασκευαστών (Third-Party Software), η εγκατάσταση του " Embedded Coder Support Package for Xilinx Zynq " έχει ολοκληρωθεί. Στο σημείο αυτό θα ζητηθεί να επιλέξουμε αν θα προχωρήσουμε στην ρύθμιση/παραμετροποίηση του συγκεκριμένου εργαλείου τώρα ή αργότερα. Αν έχει πραγματοποίηση η ρύθμιση/παραμετροποίηση του "Communication Toolbox Support Package for Xilinx Zynq-Based Radio ", όπως παρουσιάστηκε στο κεφάλαιο 3.3.3.1, τότε αυτό δεν απαιτείται, επομένως επιλέγουμε " SetUp Later".

3.3.4 Εγκατάσταση Vivado Design Suite

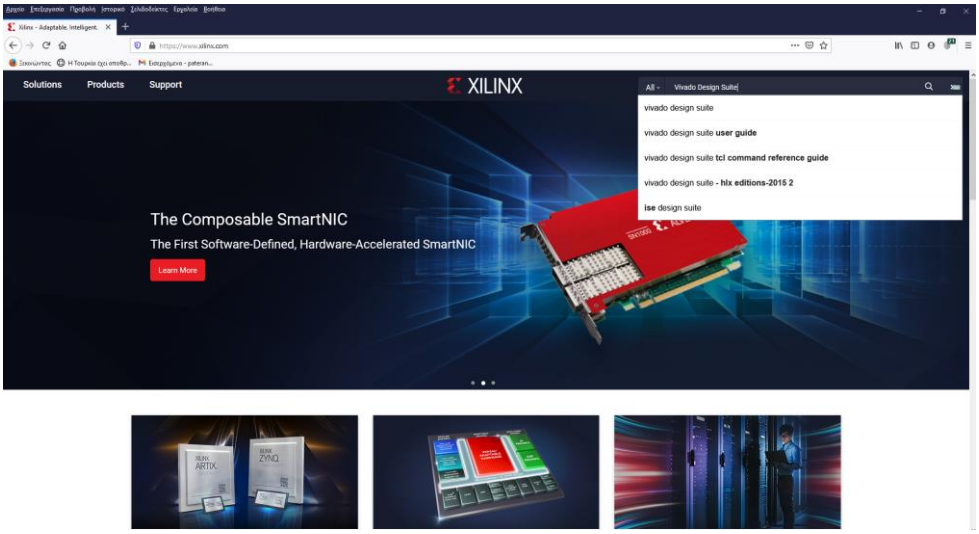
Για την εγκατάσταση του Vivado Design Suite, μεταβαίνουμε στην αρχική σελίδα της Xilinx και κατεβάζουμε τον installer του Vivado Design Suite. Η έκδοση που είναι συμβατή με την αναπτυξιακή πλακέτα ZedBoard της Xilinx είναι η 2019.1.1. Οποιαδήποτε άλλη έκδοση του Vivado Design Suite, μπορεί να οδηγήσει σε σφάλματα συμβατότητας με την αναπτυξιακή πλακέτα ZedBoard ή αποτυχία υλοποίησης του μοντέλου. Αρχικά, γίνετε η εγκατάσταση της βασικής έκδοσης του Vivado Design Suite 2019.1 και στην συνέχεια προχωράμε στην αναβάθμιση της, στην πρώτη ενημέρωση (Update 1).

Τα βήματα εγκατάστασης του Vivado Design Suite περιγράφονται αναλυτικά παρακάτω:

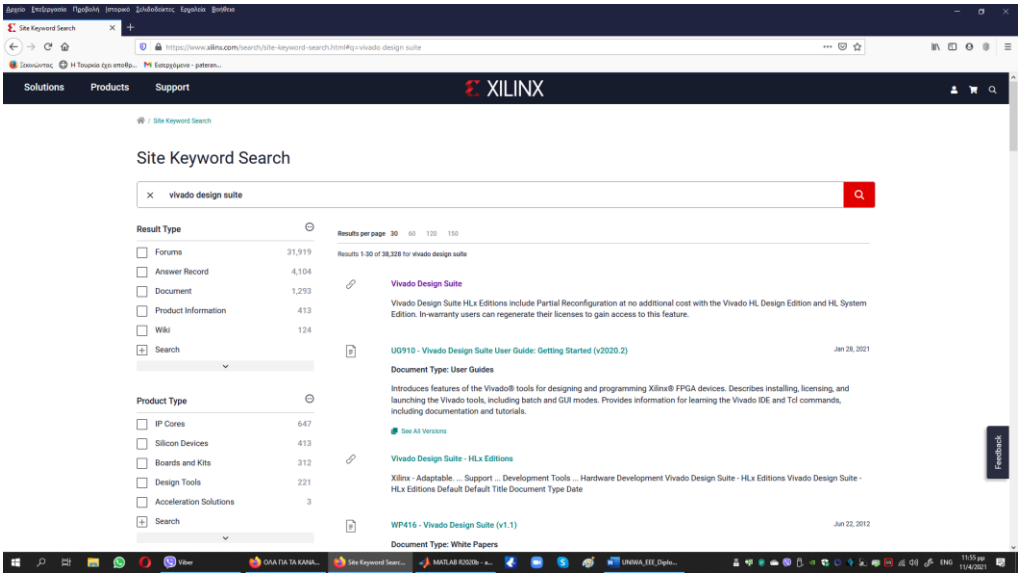
1. Μεταβαίνουμε στην αρχική σελίδα της Xilinx και στο πεδίο αναζήτησης δακτυλογραφούμε "Vivado Design Suite". Από της διαθέσιμες επιλογές , επιλέγουμε του "Vivado Design Suite" και μεταβαίνουμε στην αντίστοιχη καρτέλα.



Εικόνα 3.3.4.1 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (1/32).

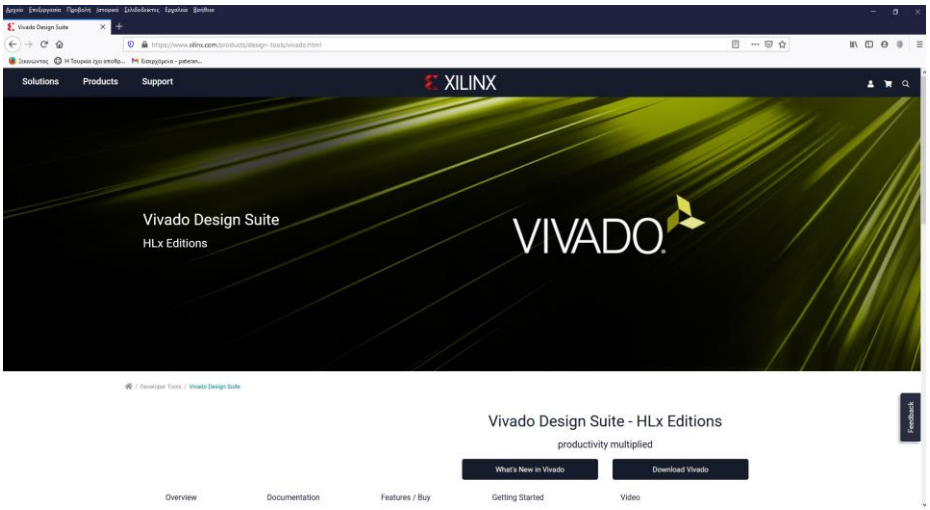


Εικόνα 3.3.4.2 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (2/32).



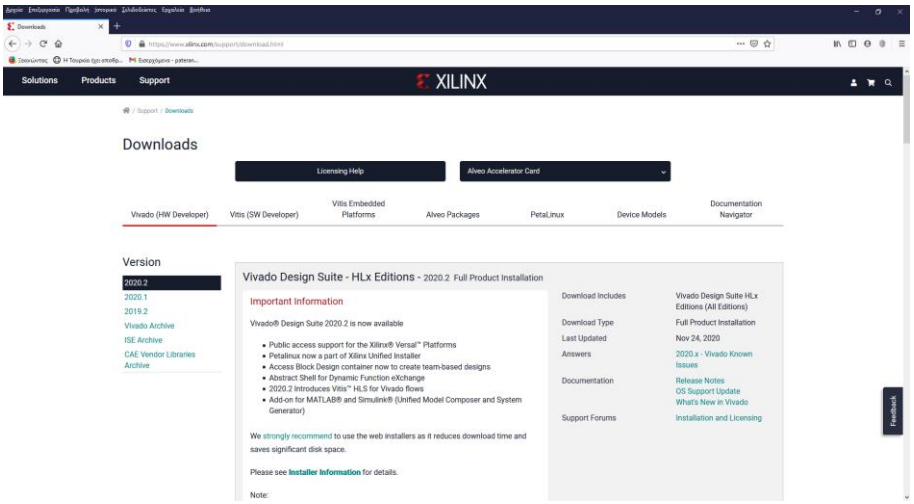
Εικόνα 3.3.4.3 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (3/32).

2. Στην καρτέλα του “Vivado Design Suite” επιλέγουμε το “DownLoad Vivado” και μεταβαίνουμε στην καρτέλα λήψης .

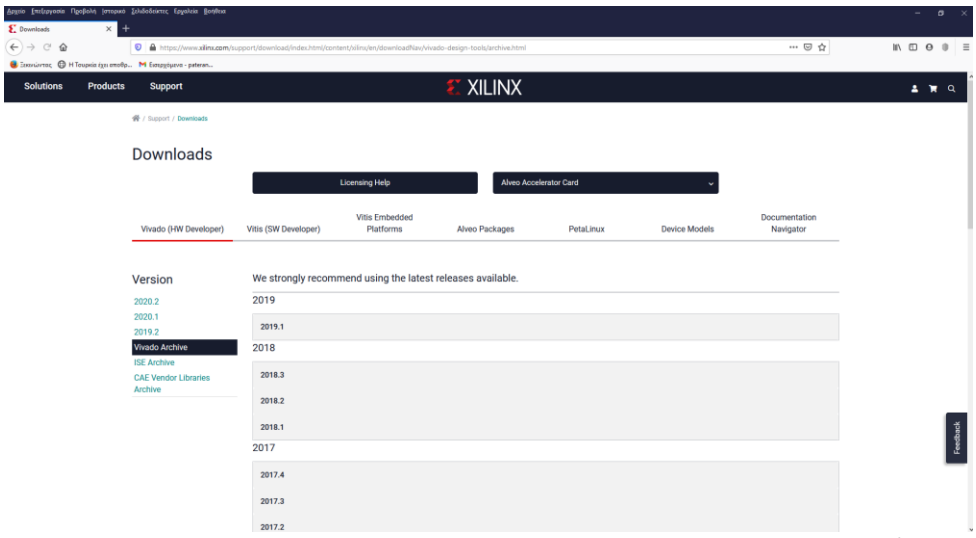


Εικόνα 3.3.4.4 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (4/32).

3. Στην καρτέλα λήψης του "Vivado Design Suite", επιλέγουμε την καρτέλα "Vivado Archive" και στην συνέχεια την επιλογή 2019.1.

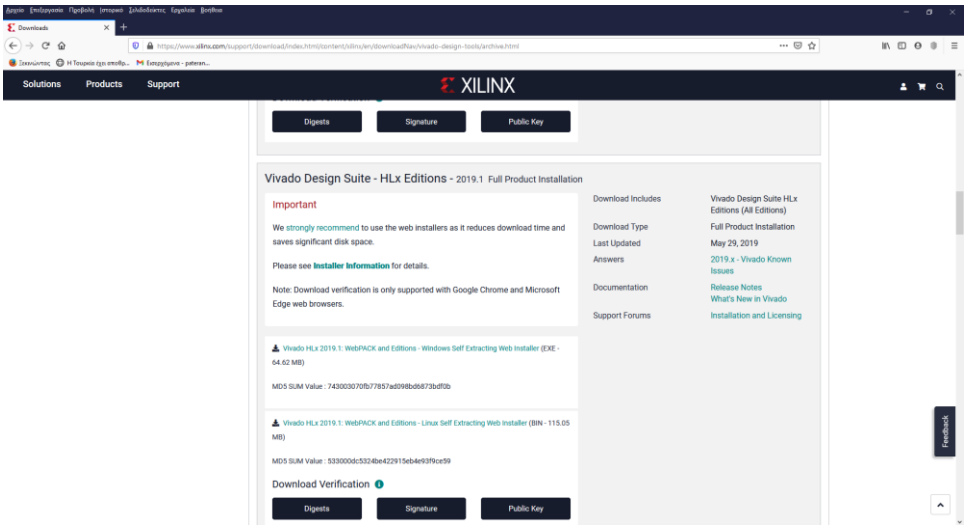


Εικόνα 3.3.4.5 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (5/32).

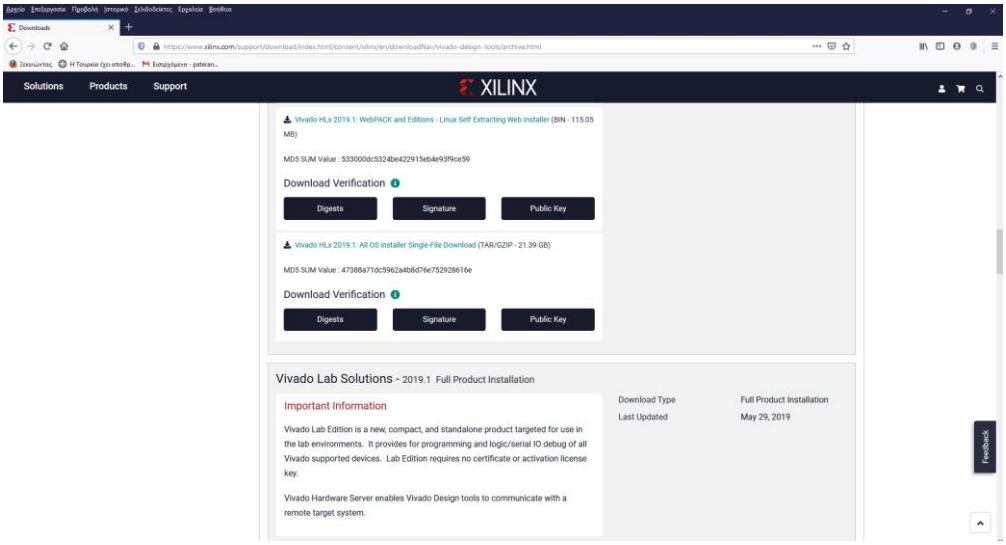


Εικόνα 3.3.4.6 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (6/32).

4. Στην καρτέλα του "Vivado 2019.1" επιλέγουμε το "Vivado HLx 2019.1: All OS installer Single-File Download".

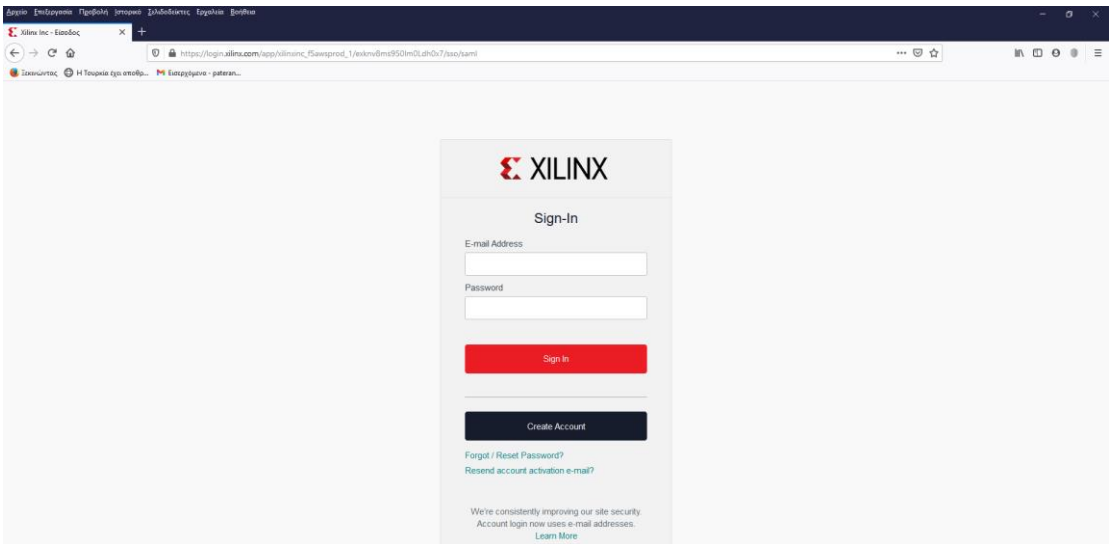


Εικόνα 3.3.4.7 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (7/32).

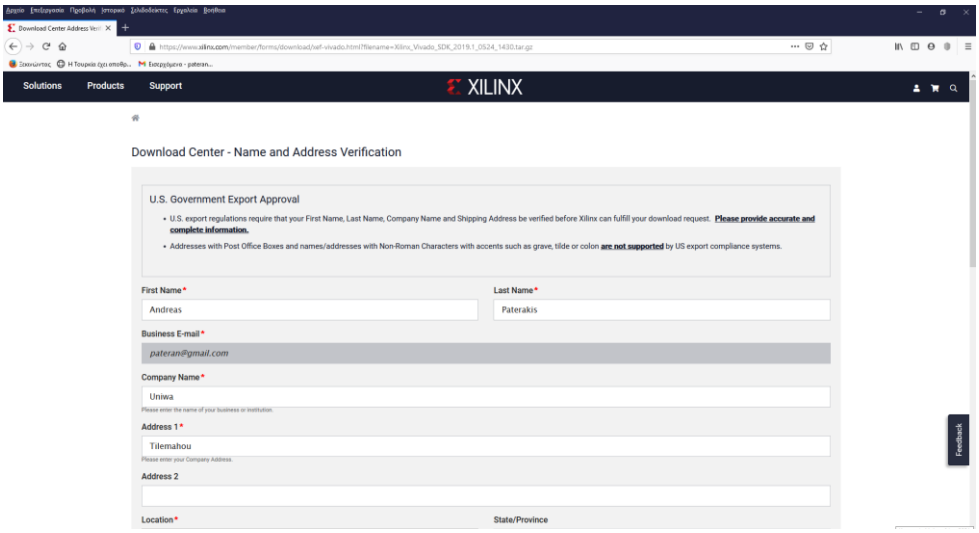


Εικόνα 3.3.4.8 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (8/32).

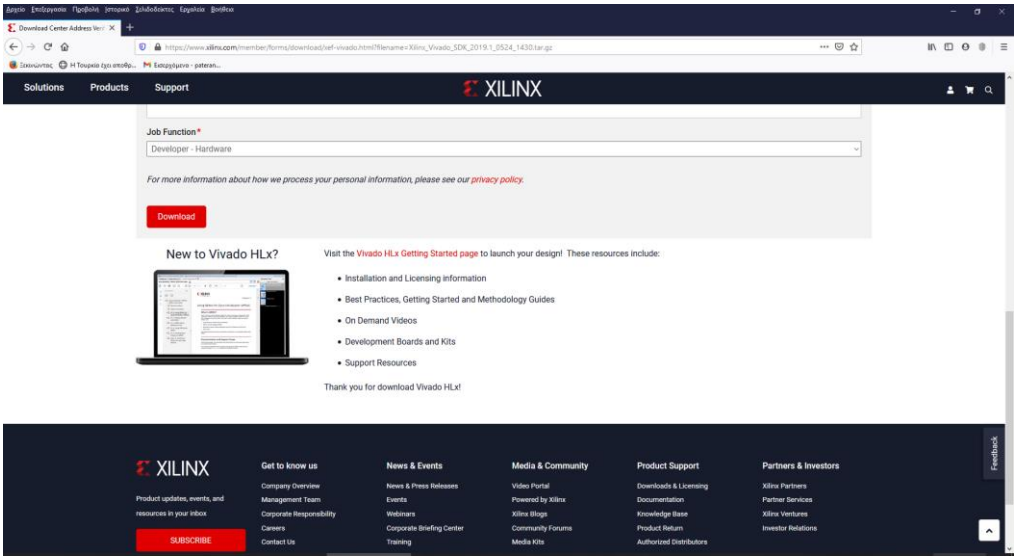
5. Πριν την έναρξη της λήψης του "Vivado Design Suite" θα μας ζητηθεί να δημιουργήσουμε έναν λογαριασμό στην Xilinx. Μετά την δημιουργία του λογαργιασμού , μπορεί να ξεκινήσει η λήψη.



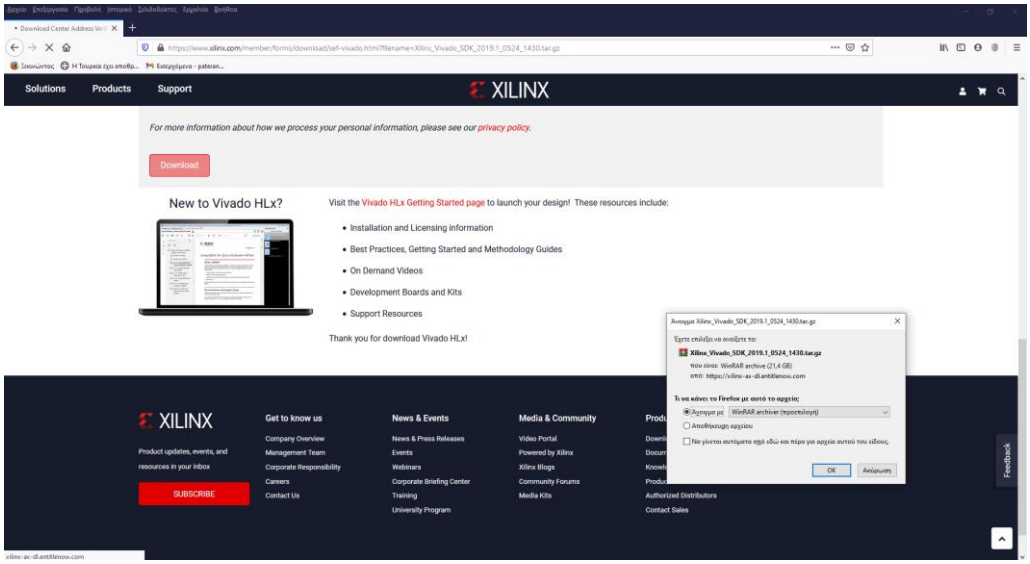
Εικόνα 3.3.4.9 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (9/32).



Εικόνα 3.3.4.10 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (10/32).

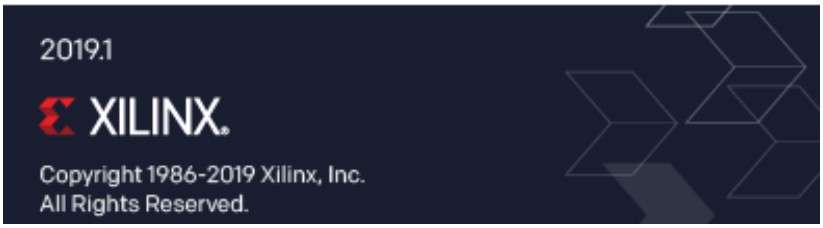


Εικόνα 3.3.4.11 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (11/32).

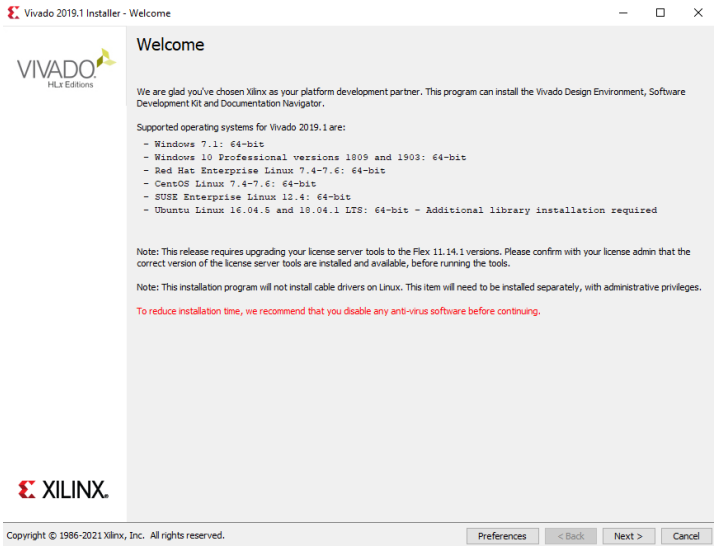


Εικόνα 3.3.4.12 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (12/32).

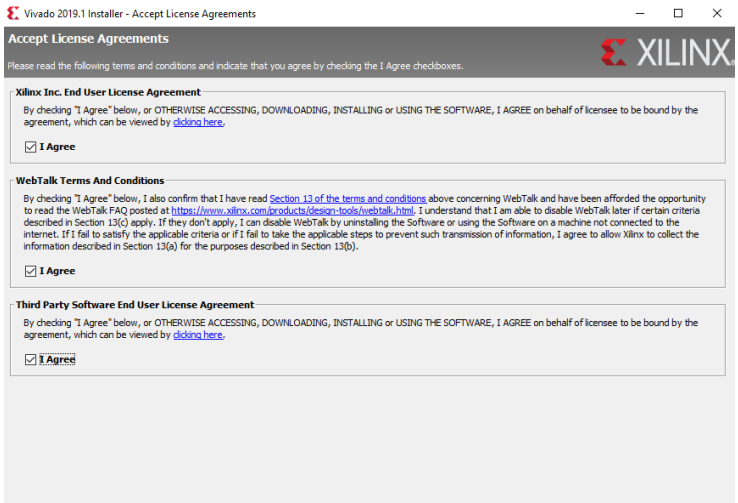
6. Με την ολοκλήρωση της λήψης προχωράμε στην εγκατάσταση του "Vivado Design Suite".



Εικόνα 3.3.4.13 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (13/32.

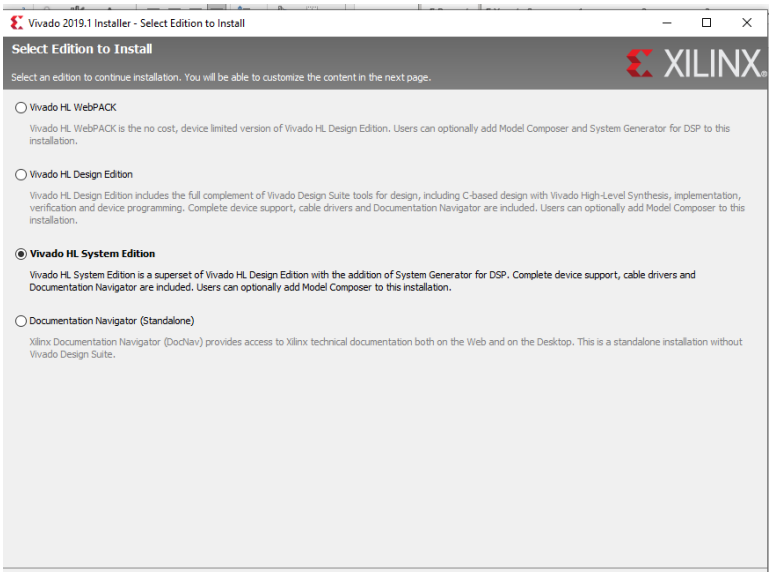


Εικόνα 3.3.4.14 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (14/32).

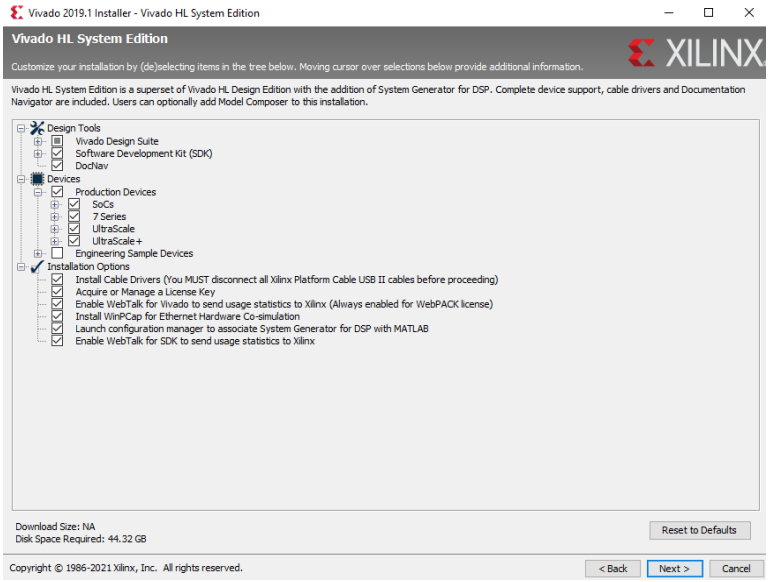


Εικόνα 3.3.4.15 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (15/32).

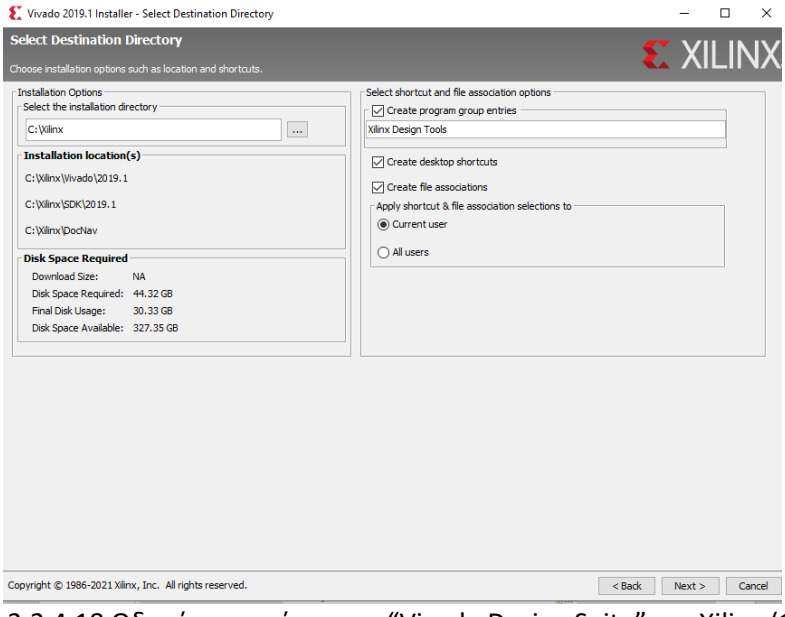
7. Στην αντίστοιχη καρτέλα, επιλέγουμε την έκδοση του Vivado , που επιθυμούμε να εγκατασταθεί. Στην συγκεκριμένη περίπτωση, επιλέγουμε την έκδοση "Vivado System Edition". Μετά από καποιες καρτέλες (επιλογή θέσης εγκατάστασης, δέσμευση χώρου στον σκληρό δίσκο) ξεκινάει η εγκατάσταση του προγράμματος.



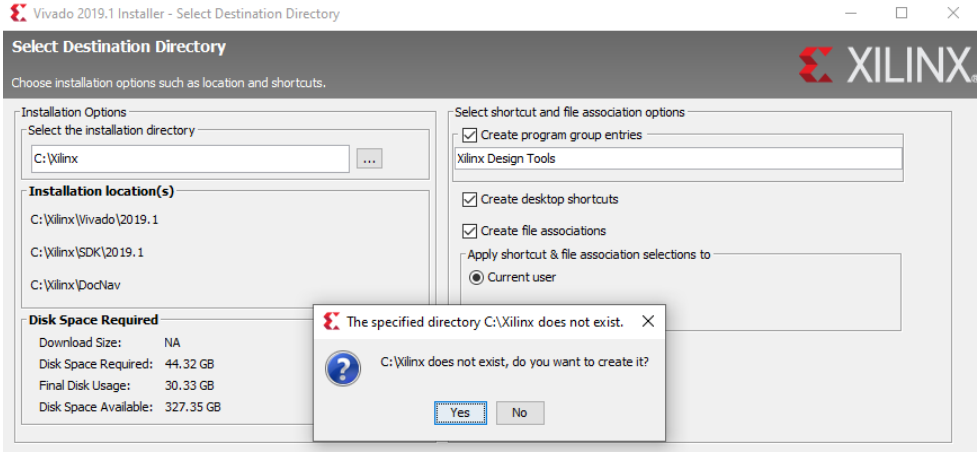
Εικόνα 3.3.4.16 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (16/32).



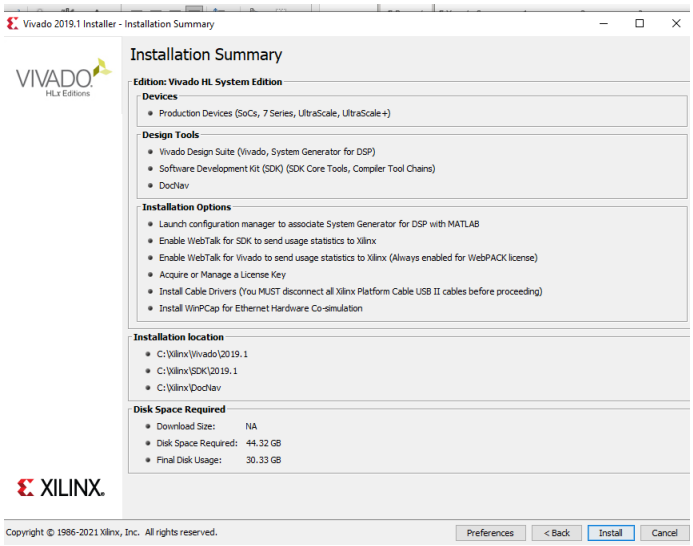
Εικόνα 3.3.4.17 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (17/32).



Εικόνα 3.3.4.18 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (18/32).



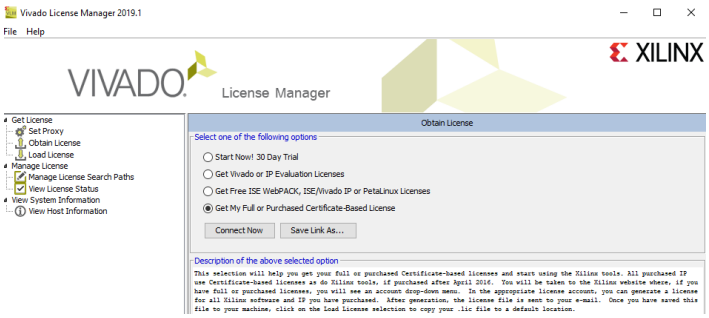
Εικόνα 3.3.4.19 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (19/32).



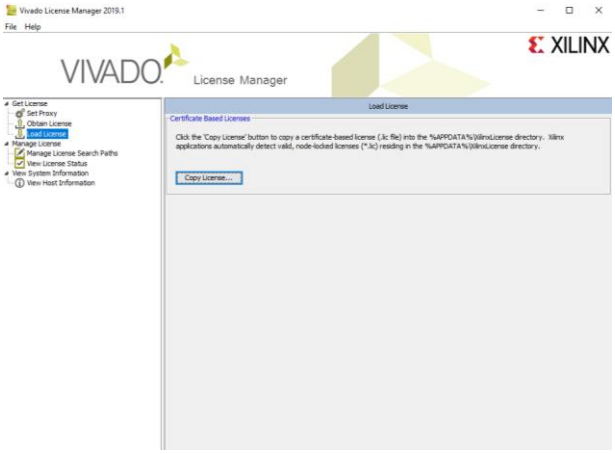
Εικόνα 3.3.4.20 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (20/32).

Εικόνα 3.3.4.21 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (21/32).

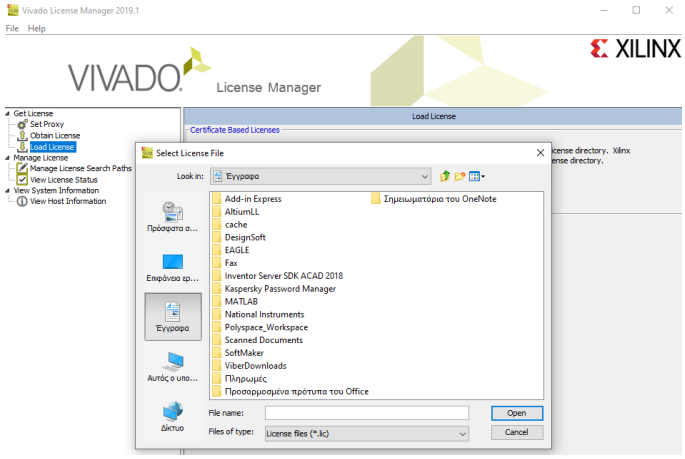
8. Μετά την επιτυχή ολοκλήρωση της εγκατάστασης του Vivado, θα μας ζητηθεί να εισάγουμε την άδεια χρήσης του Vivado, που μας έχει δοθεί από την Xilinx. Η άδεια του χρησιμοποίηθηκε για την ολοκλήρωση της συγκεκριμένη εργασίας, δοθηκε μαζί με την αγορά του εξοπλισμού (αναπτυξιακη πλακέτα ZedBoard, RF Module AD-FMCOMMS4 FMC).



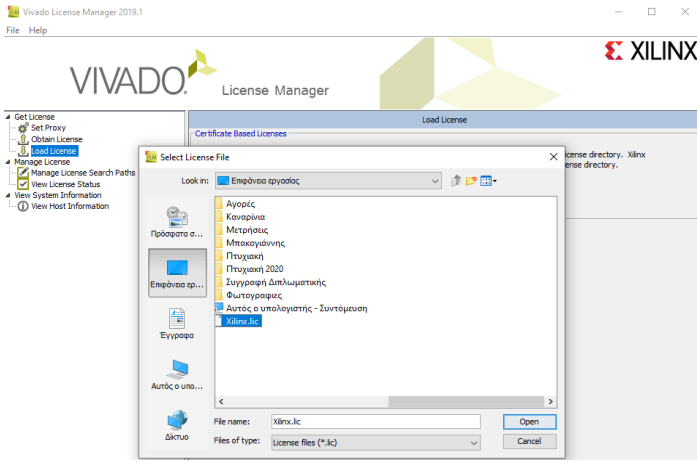
Εικόνα 3.3.4.22 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (22/32).



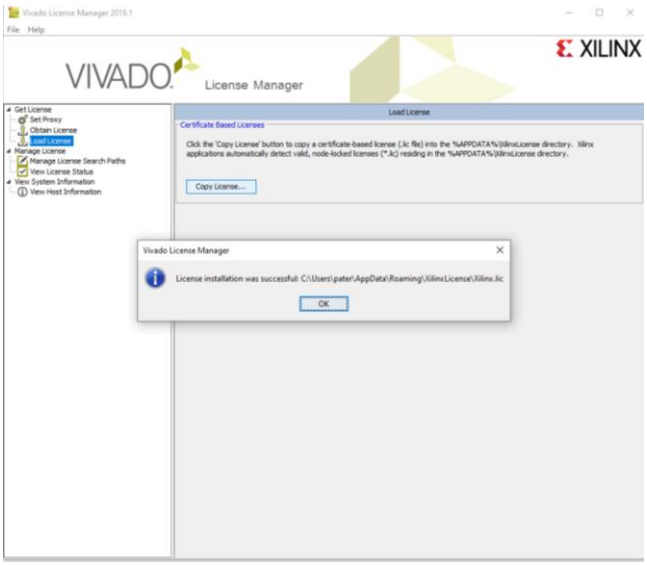
Εικόνα 3.3.4.23 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (23/32).



Εικόνα 3.3.4.24 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (24/32).

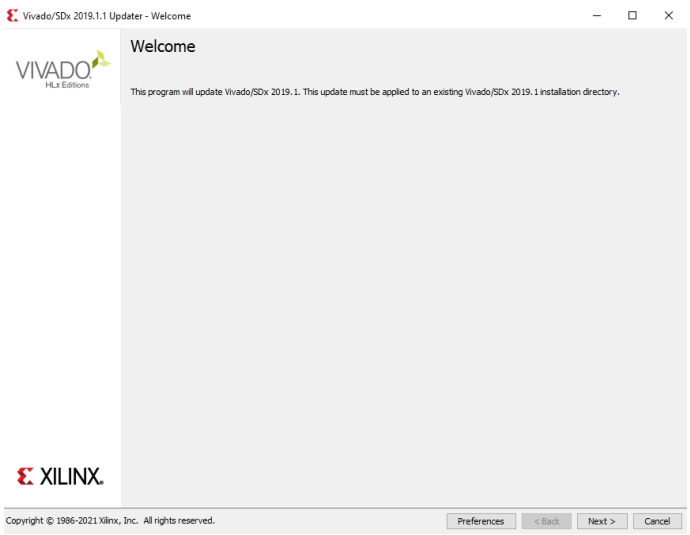


Εικόνα 3.3.4.25 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (25/32).

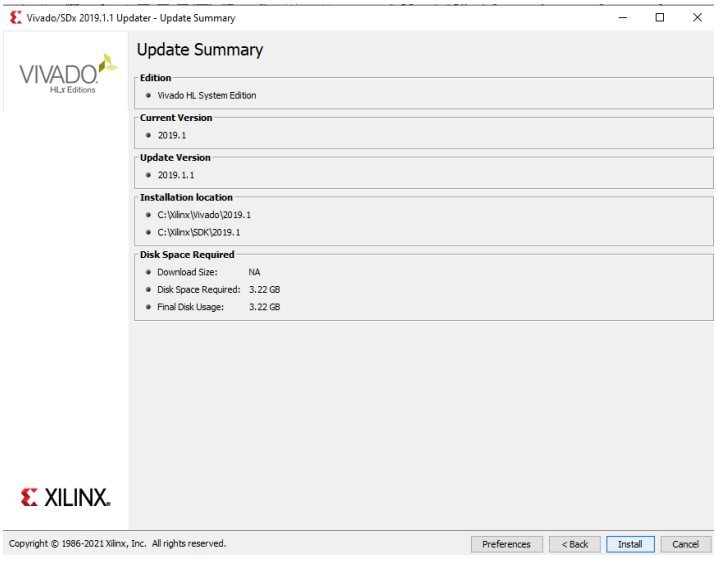


Εικόνα 3.3.4.26 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (26/32).

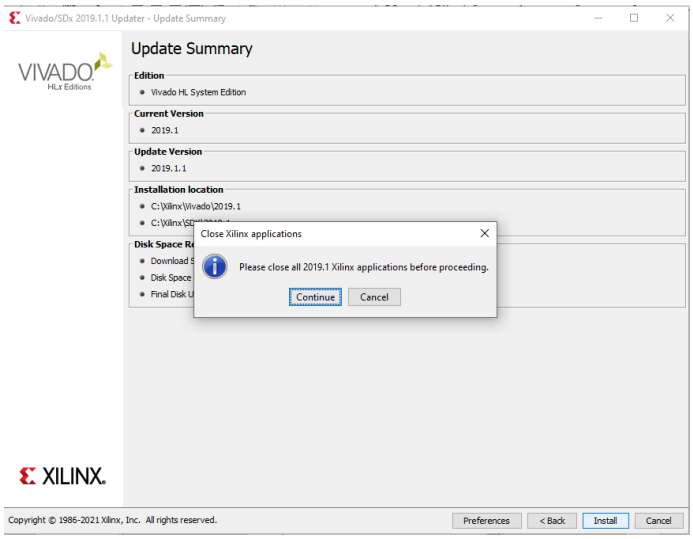
9. Με την ίδια διαδικασία που περιεγράφηκε και πριν, κατεβάζουμε την ενημέρωση (Update 1) του Vivado 2019.1 και προχωράμε στην εγκατάσταση του.



Εικόνα 3.3.4.27 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (27/32).

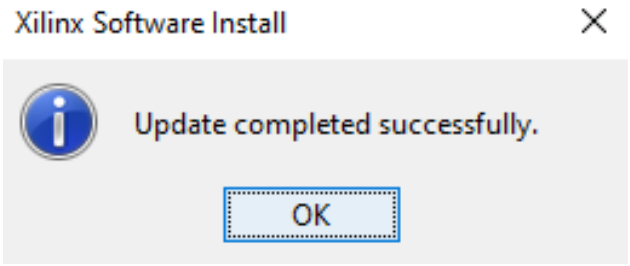


Εικόνα 3.3.4.28 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (28/32).



Εικόνα 3.3.4.29 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (29/32).

Εικόνα 3.3.4.30 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (30/32).

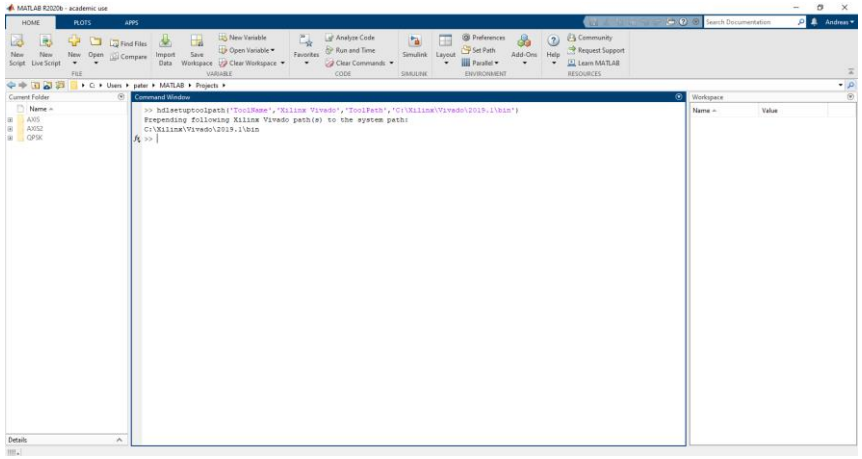


Εικόνα 3.3.4.31 Οδηγός εγκατάστασης "Vivado Design Suite" της Xilinx (31/32).

10. Μετά την ολοκλήρωση της εγκατάστασης μεταβαίνουμε στην επιφάνεια εργασίας του MatLab στο Command Line ,δακτυλογραφούμε την εντολή:

```
"hdlsetuptoolpath('ToolName','Xilinx
Vivado','ToolPath','C:\Xilinx\Vivado\2019.1\bin')"
```

Η εντολή «hdlsetuptoolpath ('ToolName', TOOLNAME, 'ToolPath', TOOLPATH)» προσθέτει ένα εργαλείο σύνθεσης FPGA τρίτου μέρους(Third Party Tool) στη διαδρομή του συστήματός και ρυθμίζει τις μεταβλητές περιβάλλοντος συστήματος για το εργαλείο σύνθεσης. Πριν την χρήση του HDL Workflow Advisor, θα πρέπει να εκτελεστή η εντολή αυτή για να προστεθεί το εργαλείο στη διαδρομή του συστήματός.



Εικόνα 3.3.4.32 Οδηγός εγκατάστασης “Vivado Design Suite” της Xilinx (32/32).

4 ΚΕΦΑΛΑΙΟ 4ο : Υλοποίηση Τηλεπικοινωνιακού Συστήματος SDR

Σε αυτό το κεφάλαιο παρουσιάζονται τα διάφορα καθεστώτα, στα οποία μπορεί να λειτουργήσει μια αναπτυξιακή πλακέτα FPGA, προκειμένου να υλοποιηθεί/εξομοιωθεί/αξιολογηθεί ένα σύστημα SDR. Επιγραμματικά θα αναφερθούν, τα επιμέρους χαρακτηριστικά σε καθένα από τα καθεστώτα λειτουργίας των αναπτυξιακών πλακετών FPGA, ενώ θα δοθούν και θα περιγράφουν κάποια χαρακτηριστικά παραδείγματα υλοποίησης συστημάτων SDR, τόσο αναλογικών όσο και ψηφιακών διαμορφώσεων, στα καθεστώτα αυτά. Στο μεγαλύτερο μέρος του κεφαλαίου αυτού, θα αναλυθεί και θα σχεδιαστεί ένα πλήρως αυτόνομο τηλεπικοινωνιακό σύστημα SDR. Θα αναφερθούν όλα τα απαραίτητα βήματα υλοποίησης του, αρχίζοντας από την σχεδίασή του τηλεπικοινωνιακό σύστημα SDR χρησιμοποιώντας τα έτοιμα μπλοκ του Simulink, την χρησιμοποίηση του HDL Coder και του προγράμματος Vivado, μέχρι την παραμετροποίηση που πρέπει να γίνει στον εξοπλισμό, προκειμένου αυτός, να είναι σε θέση να λειτουργήσει σε αυτόνομο καθεστώς (Stand Alone).

4.1 Καθεστώτα Λειτουργίας Αναπτυξιακής Πλακέτας FPGA.

Κατά την φάση της ανάπτυξης ενός συστήματος SDR, ο σχεδιαστής καλείται να προβεί στην εξομοίωση του σχεδίου που έχει υλοποιήσει, προκειμένου να πιστοποιήσει ότι αυτό λειτουργεί εντός των απαιτήσεων/προδιαγραφών, που αρχικά είχαν καθοριστεί. Στην περίπτωση αυτή σημαντικό ρόλο παίζουν οι αναπτυξιακές πλακέτες FPGA. Οι αναπτυξιακές πλακέτες FPGA επιτρέπουν την υλοποίηση/εξομοίωση των σχεδιάσεων σε τρία διαφορετικά καθεστώτα:

- Μοντέλο Διεπαφής Λογισμικού – Streaming
- Επεξεργαστής σε βρόχο - Processor-in-the-Loop
- Πλήρης Αυτόνομη Ανάπτυξη- Stand Alone

Η παρούσα εργασία έχει σκοπό την υλοποίηση ενός Πλήρης Αυτόνομου Συστήματος SDR- Stand Alone SDR System. Παρόλα, αυτά θα γίνει αναφορά και στα υπόλοιπα δύο καθεστώτα.

4.1.1 Μοντέλο Διεπαφής Λογισμικού – Streaming

Στο καθεστώς Μοντέλου Διεπαφής Λογισμικού – Streaming, η πλακέτα ZedBoard και η RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC, χρησιμοποιούνται για την εκπομπή και λήψη του σήματος του αναπτυσσόμενου/σχεδιαζόμενου συστήματος SDR, χωρίς να είναι αναγκαίος οπουδήποτε είδους προγραμματισμός σε αυτές. Με την χρήση του MatLab και του Simulink, μπορεί να σχεδιαστεί οποιοδήποτε τηλεπικοινωνιακό σύστημα, το οποίο θα εκπέμψει και θα λάβει την πληροφορία. διαμέσου της πλακέτα ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC. Στην περίπτωση αυτή, το υλικό (η πλακέτα ZedBoard και η RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC), χρησιμοποιείται σαν διεπαφή ανάμεσα στο λογισμικό (MatLab, Simulink) και το κανάλι.

Η σχεδίαση του συστήματος SDR, το οποίο θέλουμε να δοκιμάσουμε μέσω της αναπτυξιακής FPGA πλακέτας, μπορεί να αφορά μία υλοποίηση με την μορφή κώδικα MatLab (script) ή με την μορφή μιας σχεδίασης με την χρήση έτοιμων μπλοκ του Simulink. Στα παραδείγματα που ακολουθούν, περιγράφονται και τα δύο είδη υλοποίησης.

4.1.1.1 Παραδείγματα Μοντέλου Διεπαφής Λογισμικού – Streaming με χρήση κώδικα MatLab (script)

4.1.1.1.1 Εκπομπή και λήψη Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 - Receive Tone Signal Using Analog Devices AD9364 [100].

Αυτό το παράδειγμα παρουσιάζει την χρήση του εργαλείου Xilinx Zynq-based Radio Support Package and Communications Toolbox ™, για την υλοποίηση μίας εκπομπής και λήψης ενός σύνθετου ημιτονοειδούς σήματος χρησιμοποιώντας την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364. Περιέχει έναν έτοιμο κώδικά MatLab (script), όπου ένας Ψηφιακό Συνθέτη Συχνοτήτων - Direct Digital Synthesizer (DDS) [101] στο FPGA δημιουργεί ένα σύνθετο ημιτονοειδές σήμα και το μεταδίδει χρησιμοποιώντας την κάρτα RF.

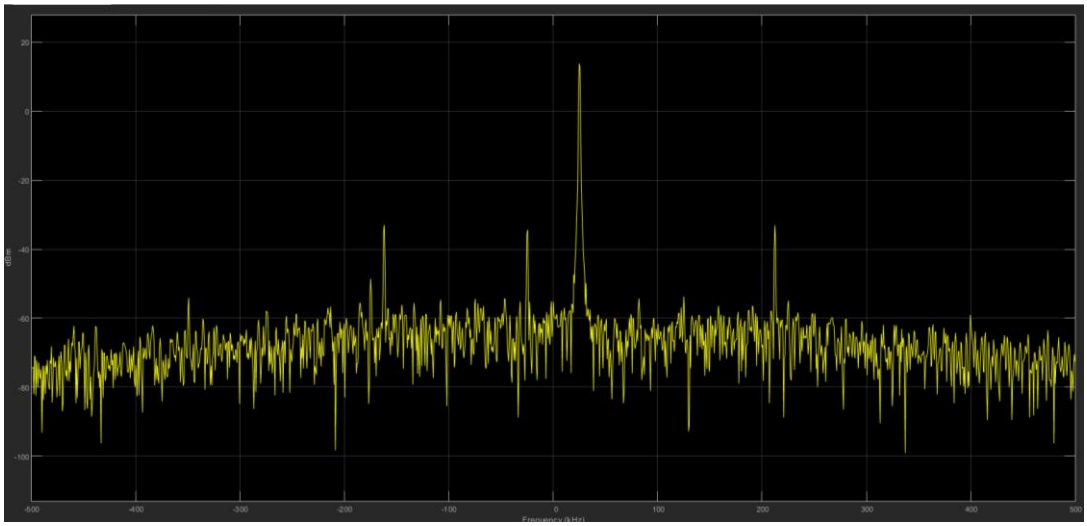
Προκειμένου να εκτελέσουμε αυτόν τον κώδικα του MatLab, μεταβαίνουμε στο Command Window του MatLab και πληκτρολογούμε την εντολή:

```
«edit zynqRadioToneReceiverAD9361AD9364ML»,
```

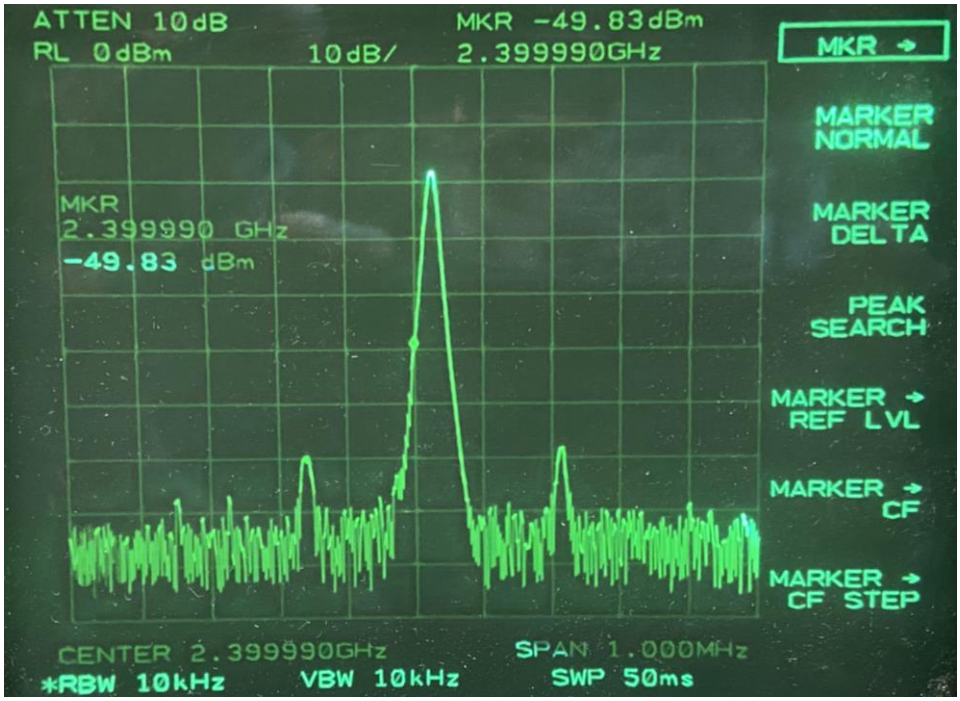
όπου θα ανοίξει η καρτέλα Editor του MatLab, με το script του συγκεκριμένου παραδείγματος.

Στην συνέχεια θα πρέπει να ενεργοποιηθεί η πλακέτα ZedBoard, να συνδεθεί η θύρα Ethernet της κάρτα δικτύου του υπολογιστή, με την αντίστοιχη της πλακέτας και να συνδεθεί μία θύρα USB του υπολογιστή, με την θύρα UART της πλακέτας χρησιμοποιώντας τα κατάλληλα (καλώδιο Ethernet και καλώδιο USB σε micro USB, αντίστοιχα). Αφού ολοκληρωθούν οι προηγούμενες ενέργειες, προχωράμε στην εκτέλεση του παραδείγματος, με την χρήση του πλήκτρου «Play», στην καρτέλα Editor του MatLab.

Αλλάζοντας τις παραμέτρους του κώδικα, μπορούμε να διαφοροποιήσουμε τόσο την συχνότητα του μεταδιδόμενου Σύνθετου Ημιτονοειδούς Σήματος, όσο και την κεντρική συχνότητα της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC (25KHz και 2,4GHz αντίστοιχα). Το μεταδιδόμενο σήμα στη συνέχεια λαμβάνεται από την κάρτα RF και το σήμα βασικής ζώνης απεικονίζεται στο MatLab. Αυτό το απλό παράδειγμα επιβεβαιώνει ότι το υλικό SDR έχει ρυθμιστεί σωστά και δείχνει τον τρόπο λήψης δεδομένων RF από υλικό SDR χρησιμοποιώντας MatLab.



Εικόνα 4.1.1.1.1.1 Φάσμα εκπομπής Σύνθετου Ημιτονοειδούς Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard και την RF πλακέτα AD-FMCOMMS4 FMC, σε αναλυτή φάσματος του MatLab.



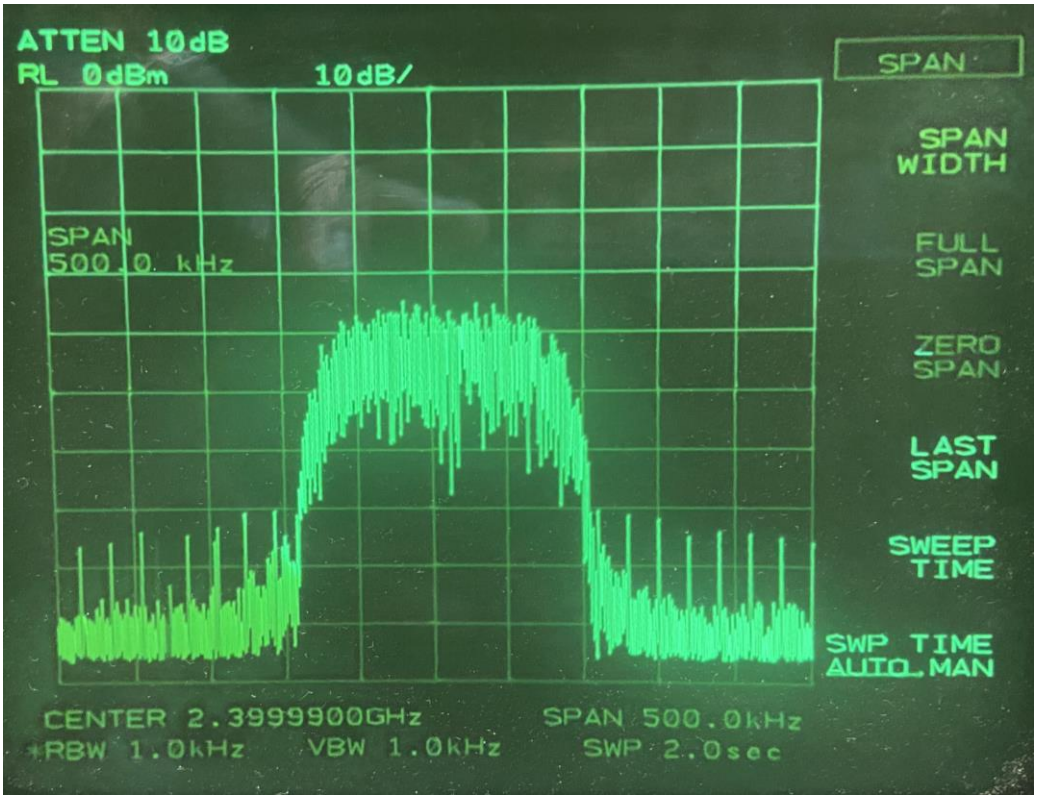
Εικόνα 4.1.1.1.1.2 Φάσμα εκπομπής Σύνθετου Ημιτονοειδούς Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard και την RF πλακέτα AD-FMCOMMS4 FMC σε εργαστηριακό αναλυτή φάσματος .

4.1.1.1.2 QPSK Εκπομπή Σήματος χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 - QPSK Transmit Repeat Using Analog Devices AD9364 [102]

Αυτό το παράδειγμα δείχνει την υλοποίηση ενός πομπού QPSK με την χρήση έτοιμου κώδικα MatLab (script), με την βοήθεια του εργαλείου Xilinx® Zynq-based Radio Support Package και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, για συνεχή μετάδοση δεδομένων QPSK.
Προκειμένου να εκτελέσουμε αυτόν τον κώδικα του MatLab, μεταβαίνουμε στο Command Window του MatLab και πληκτρολογούμε την εντολή:

```
«edit zynqRadioQPSKTransmitRepeatAD9361AD9364ML»,
```

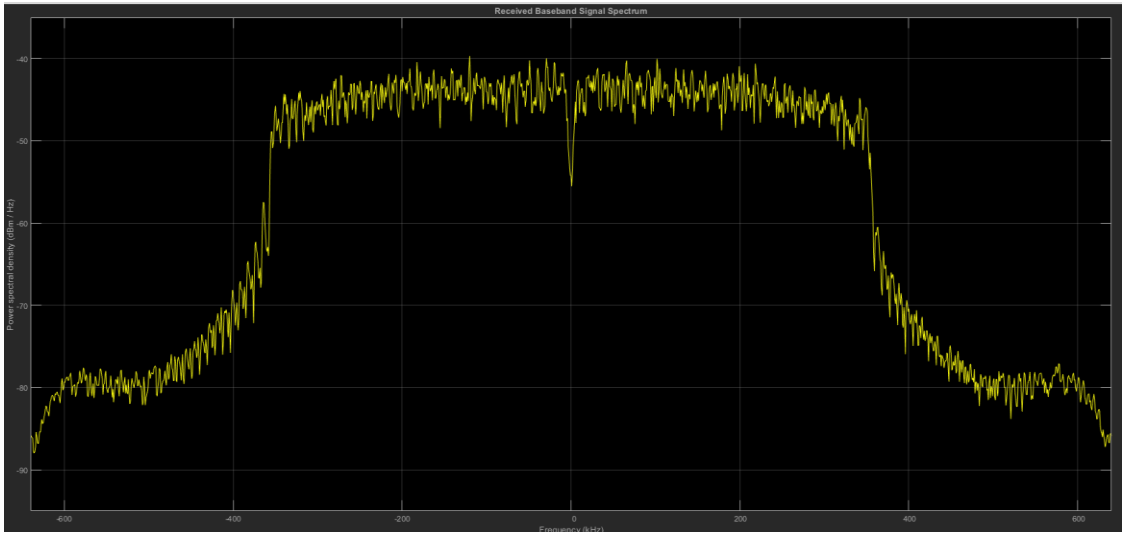
όπου θα ανοίξει η καρτέλα Editor του MatLab, με το script του συγκεκριμένου παραδείγματος.
Στην συνέχεια θα πρέπει να ενεργοποιηθεί η πλακέτα ZedBoard, να συνδεθεί η θύρα Ethernet της κάρτα δικτύου του υπολογιστή, με την αντίστοιχη της πλακέτας και να συνδεθεί μία θύρα USB του υπολογιστή, με την θύρα UART της πλακέτας χρησιμοποιώντας τα κατάλληλα (καλώδιο Ethernet και καλώδιο USB σε micro USB, αντίστοιχα). Αφού ολοκληρωθούν οι προηγούμενες ενέργειες, προχωράμε στην εκτέλεση του παραδείγματος, με την χρήση του πλήκτρου «Play», στην καρτέλα Editor του MatLab.
Αλλάζοντας τις παραμέτρους του κώδικα, μπορούμε να διαφοροποιήσουμε, τόσο την κεντρική συχνότητα του μεταδιδόμενου Σύνθετου Ημιτονοειδούς Σήματος της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, όσο και τον χρόνο εκπομπής (2,4GHz και 20 sec αντίστοιχα).



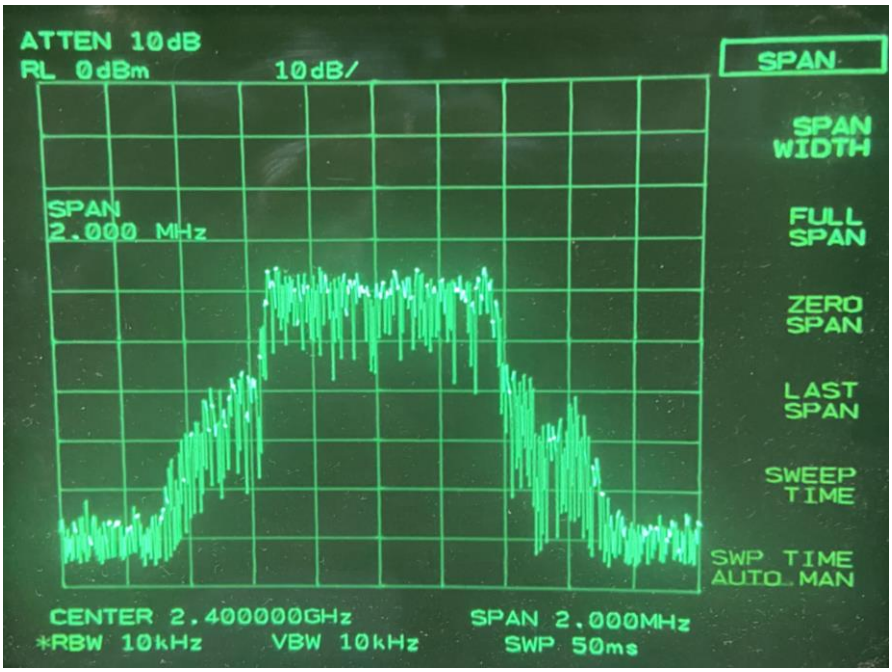
Εικόνα 4.1.1.1.2.1 Φάσμα QPSK Εκπομπής με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της Analog Devices AD9364 σε εργαστηριακό αναλυτή φάσματος .

4.1.1.1.3 Ανάπτυξη τηλεπικοινωνιακών υποσυστημάτων OFDM με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC. [103]

Στο παράδειγμα που ακολουθεί, χρησιμοποιήθηκε ο κώδικας MatLab, που δημιουργήθηκε για την ανάπτυξη τηλεπικοινωνιακών υποσυστημάτων OFDM με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC. Ακολουθώντας τα βήματα όπως περιγράφονται στην συγκεκριμένη διπλωματική μπορούμε να υλοποιήσουμε ένα τηλεπικοινωνιακό υποσύστημα OFDM, με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364.



Εικόνα 4.1.1.1.3.1 Φάσμα OFDM εκπομπής 32-QAM με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, από αναλυτή φάσματος του MatLab.



Εικόνα 4.1.1.1.3.2 Φάσμα OFDM εκπομπής 32-QAM με την χρήση της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, από εργαστηριακό αναλυτή φάσματος .

4.1.1.2 Παραδείγματα Μοντέλου Διεπαφής Λογισμικού – Streaming με χρήση έτοιμων μπλοκ του Simulink

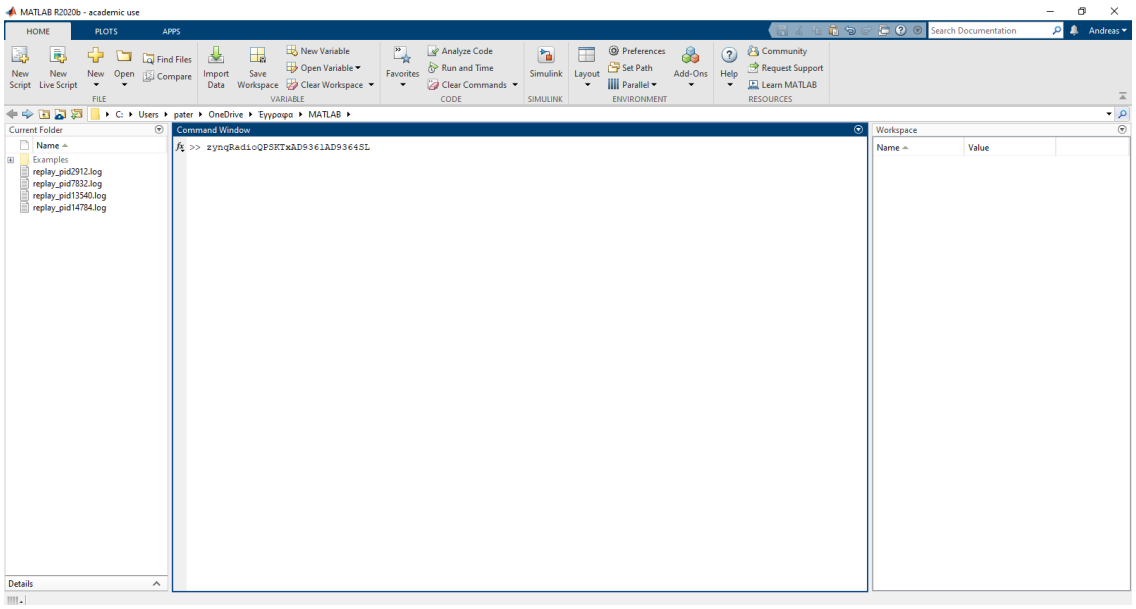
4.1.1.2.1 Υλοποίηση πομπού QPSK με την χρήση του Analog Devices AD9364 - QPSK Transmitter Using Analog Devices AD9364 [104]

Αυτό το παράδειγμα δείχνει τον τρόπο χρήσης της αναπτυξιακής πλακέτας ZedBoard, για την υλοποίηση ενός πομπού QPSK, με την χρήση έτοιμων μπλοκ του Simulink σε καθεστώς διεπαφής - Streaming. Η συσκευή SDR σε αυτό το μοντέλο θα μεταδίδει συνεχώς το μήνυμα «Hello world» που διαμορφώνονται σε QPSK σε φορέα με καθορισμένη κεντρική συχνότητα στα 2.4GHz. Το συγκεκριμένο μεταδιδόμενο μήνυμα μπορεί να αποδιαμορφωθεί, χρησιμοποιώντας το μοντέλο QPSK Receiver Using Analog Devices AD9364 [105] με την χρήση μιας δεύτερη πλατφόρμα SDR.

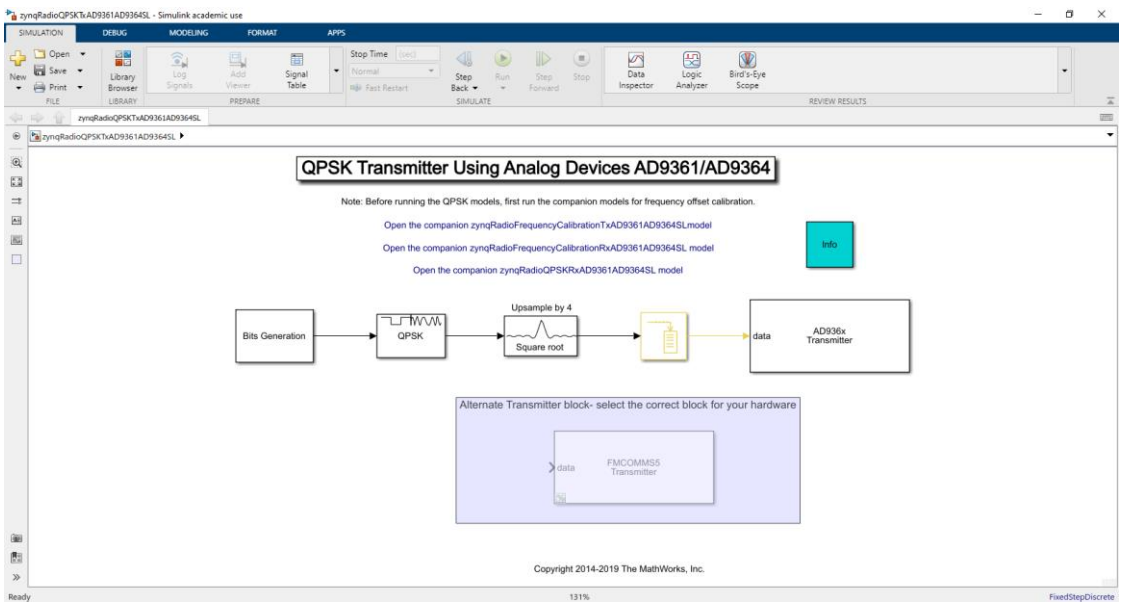
Με την πληκτρολόγηση της εντολής:

```
«zynq RadioQPSKTxAD9361AD9364SL»
```

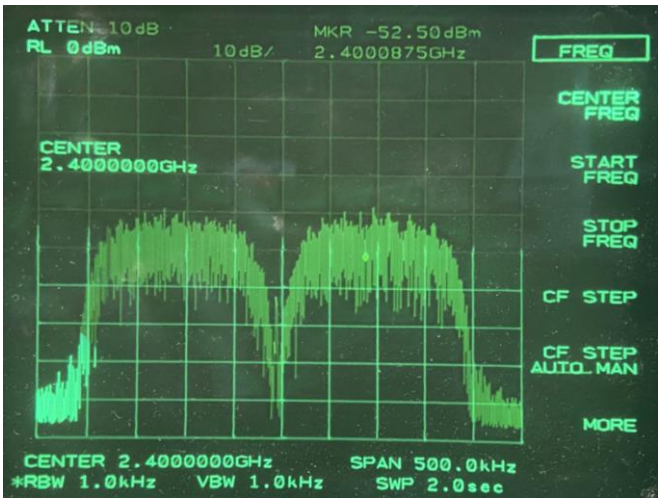
στο Command Window του MatLab, μεταβαίνουμε στο Simulink, στο έτοιμο μοντέλο του QPSK πομπού του παραδείγματος.
Αφού ενεργοποιήσουμε την πλακέτα ZedBoard, και υλοποιήσουμε τις κατάλληλες συνδέσεις όπως και στα προηγούμενα παραδείγματα, προχωράμε στην εκτέλεση του παραδείγματος με την χρήση του πλήκτρου «Play», στην καρτέλα Simulation του Simulink. Στο μπλοκ του πομπού AD9364, μπορούμε να αλλάξουμε τις ρυθμίσεις της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC (ισχύς, συχνότητα εκπομπής, συχνότητα δειγματοληψίας, κ.τ.λ.). Σε κατάσταση ηρεμίας της πλακέτας ZedBoard (χρονική στιγμή κατά την οποία δεν έχουμε κάποια εκπομπή ή λήψη), μπορούμε να δούμε στον αναλυτή φάσματος, την φέρουσα της πλακέτας. Η φέρουσα της πλακέτας είναι ένα σήμα στο 2.4GHz, με ισχύς -64,17dBm(αν δεν έχουν αλλαχθεί οι ρυθμίσεις της πλακέτας ZedBoard). Η φέρουσα αυτή επιβεβαιώνει την σωστή κατάσταση λειτουργίας της πλακέτα.



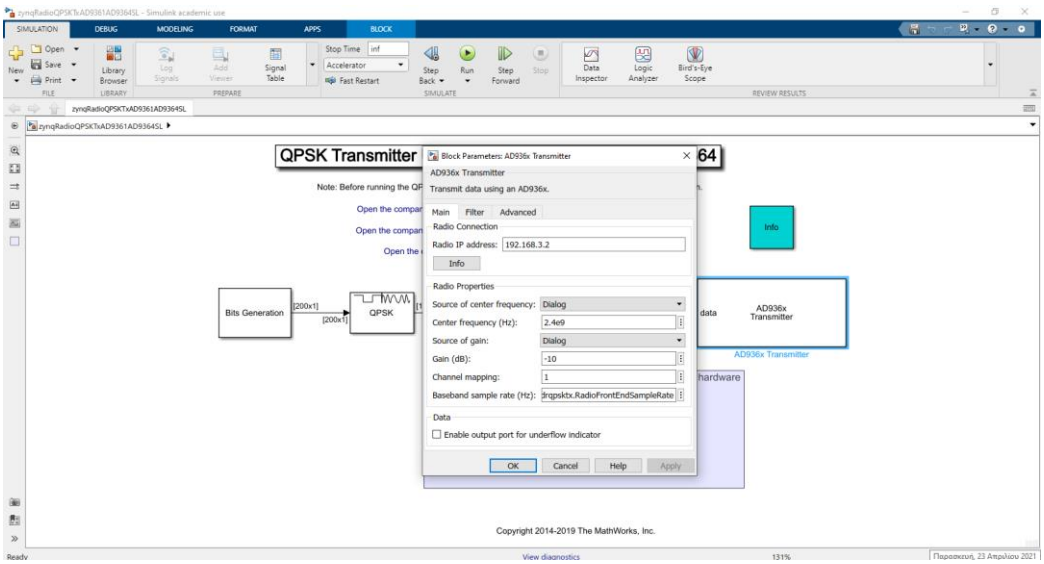
Εικόνα 4.1.1.2.1.1 Command Window του MatLab.



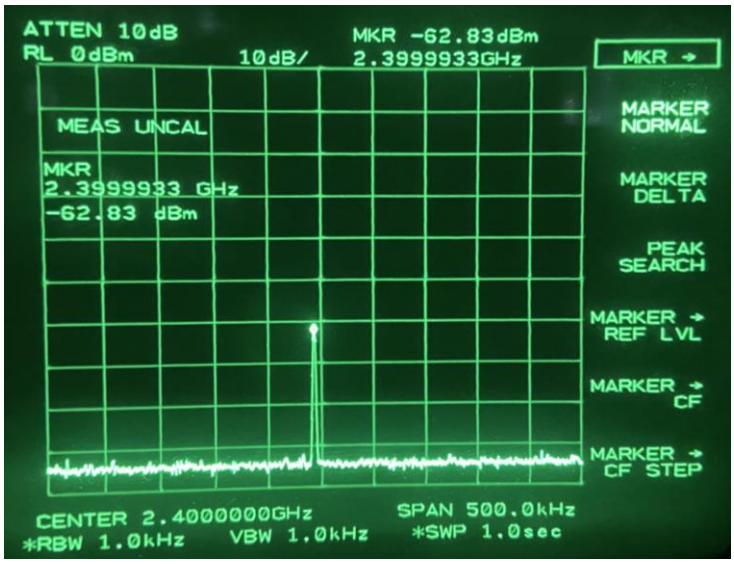
Εικόνα 4.1.1.2.1.2 Παράδειγμα υλοποίησης QPSK Εκπομπής με την χρήση έτοιμων μπλοκ του Simulink για την αναπτυξιακή πλακέτας ZedBoard και την Analog Devices AD9364.



Εικόνα 4.1.1.2.1.3 QPSK Εκπομπή με την χρήση έτοιμων μπλοκ του Simulink για την αναπτπυξιακή πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος .



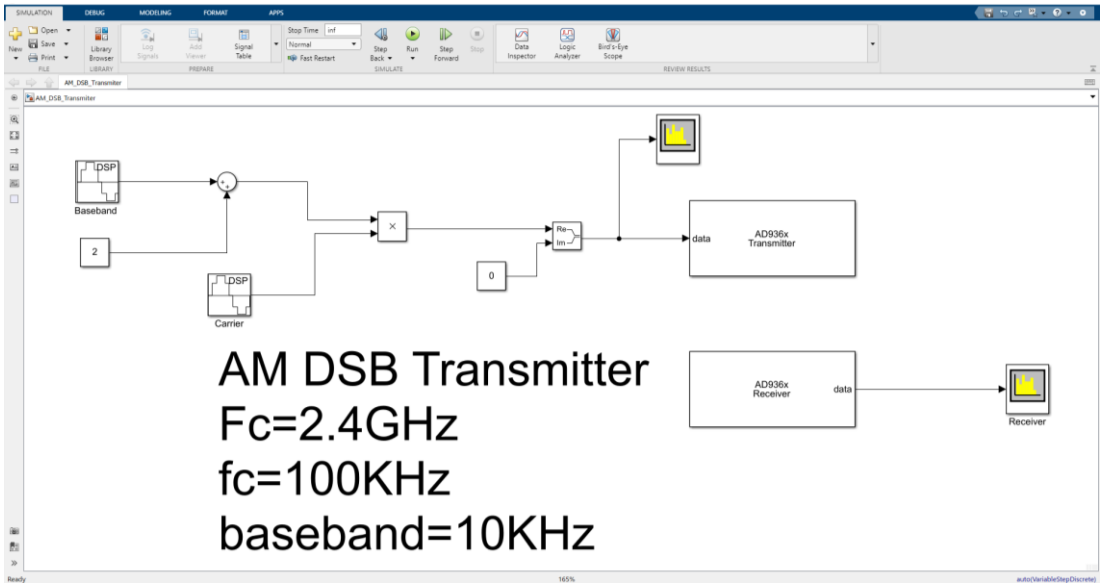
Εικόνα 4.1.1.2.1.4 Ρυθμίσεις της Analog Devices AD9364 .



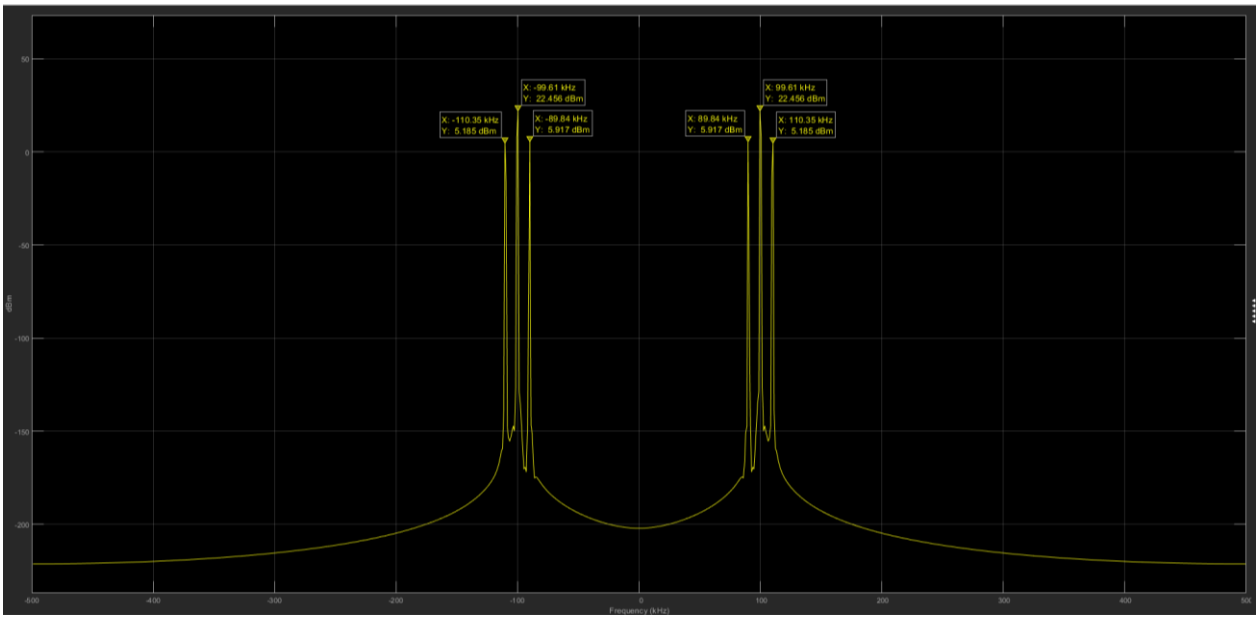
Εικόνα 4.1.1.2.1.5 Φέρουσα της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος σε κατάσταση ηρεμίας.

4.1.1.2.2 Υλοποίηση Πομπού Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με συνολικό φέρον (Double Side Band-Amplitude Modulation-Total Carrier - DSB-AM-TC)

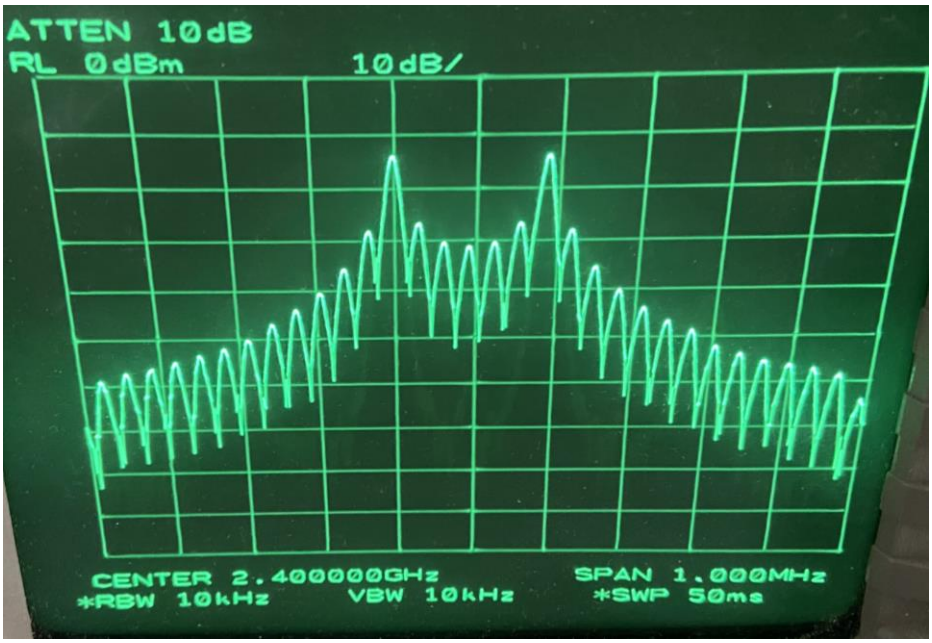
Στα πλαίσια του εργαστηρίου του μαθήματος «Τηλεπικοινωνίες» δημιουργήθηκε ένας πομπός Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με συνολικό φέρον (Double Side Band-Amplitude Modulation-Total Carrier - DSB-AM-TC), με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364. Ένα ημίτονο με συχνότητα 10KHz (Baseband), διαμορφώνεται από ένα φέρον (fc) με συχνότητα 100KHz. Το σήμα που παράγεται στην συνέχεια διαμορφώνεται με την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, με συχνότητα 2.4GHz και εκπέμπεται μέσω της κεραίας.



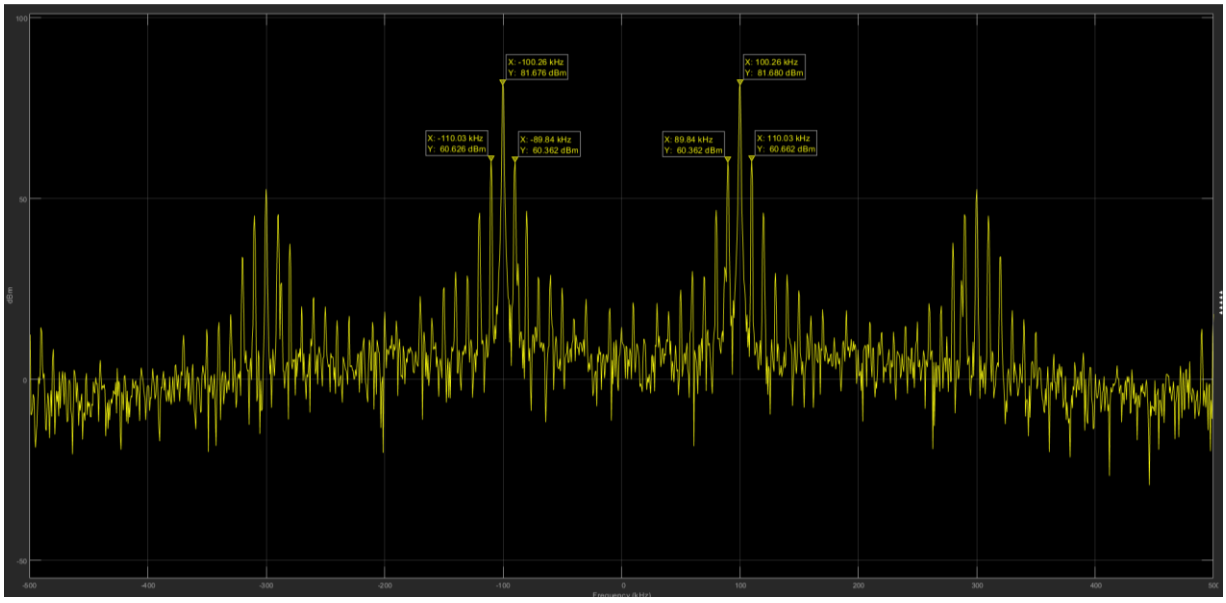
Εικόνα 4.1.1.2.2.1 Υλοποίηση πομπού DSB-AM-TC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364.



Εικόνα 4.1.1.2.2.2 Φάσμα πομπού DSB-AM-TC, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



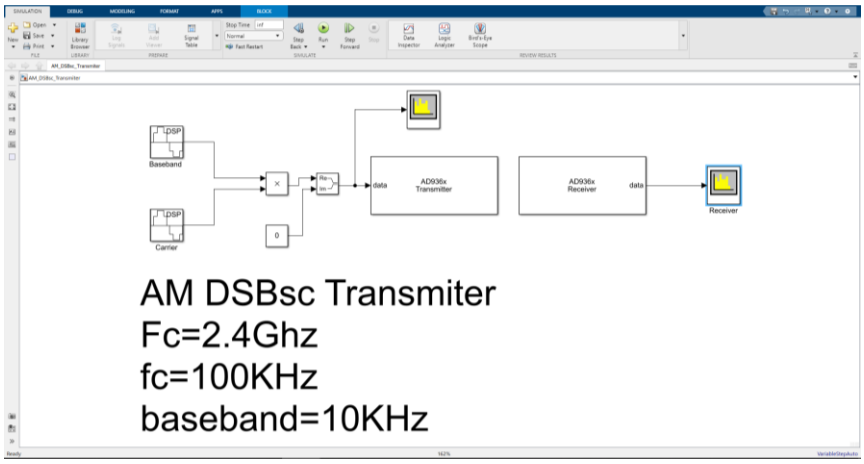
Εικόνα 4.1.1.2.2.3 Φάσμα πομπού DSB-AM-TC στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



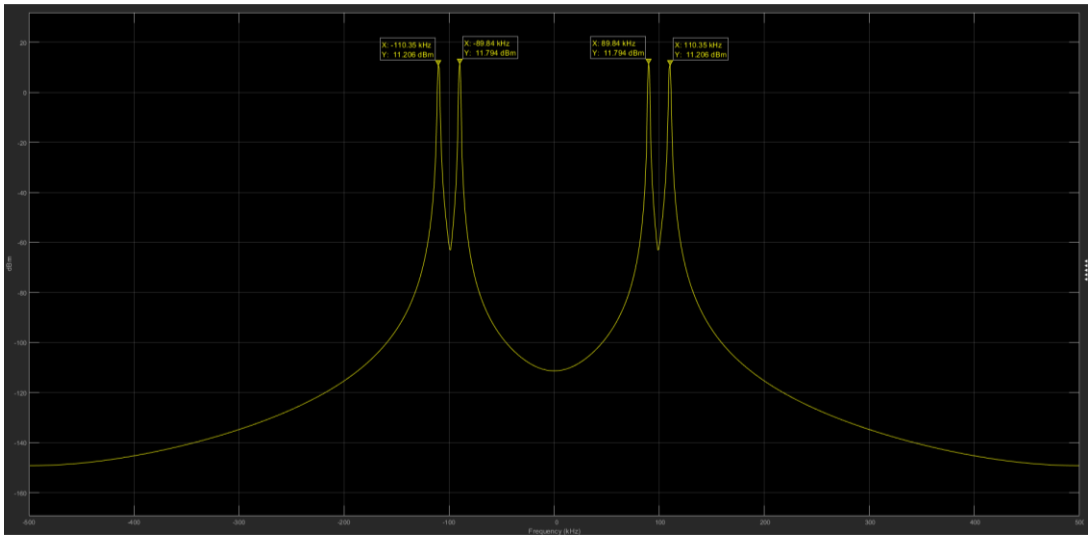
Εικόνα 4.1.1.2.2.4 Φάσμα δέκτη DSB-AM-TC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.

4.1.1.2.3 Υλοποίηση Πομπού Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC)

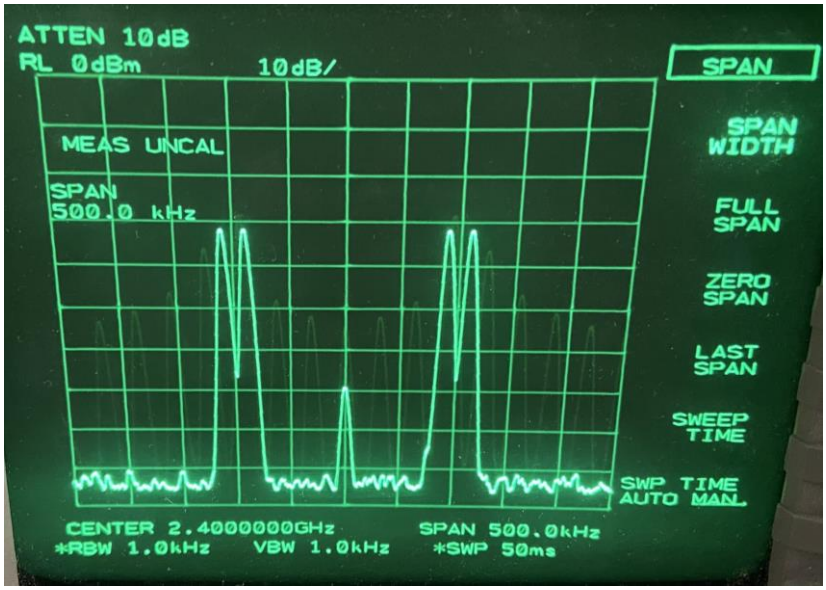
Στα πλαίσια του εργαστηρίου του μαθήματος «Τηλεπικοινωνίες» δημιουργήθηκε ένας πομπός Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC), με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364. Ένα ημίτονο με συχνότητα 10KHz (Baseband), διαμορφώνεται από ένα φέρον (fc) με συχνότητα 100KHz. Το σήμα που παράγεται στην συνέχεια διαμορφώνεται με την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, με συχνότητα 2.4GHz και εκπέμπεται μέσω της κεραίας.



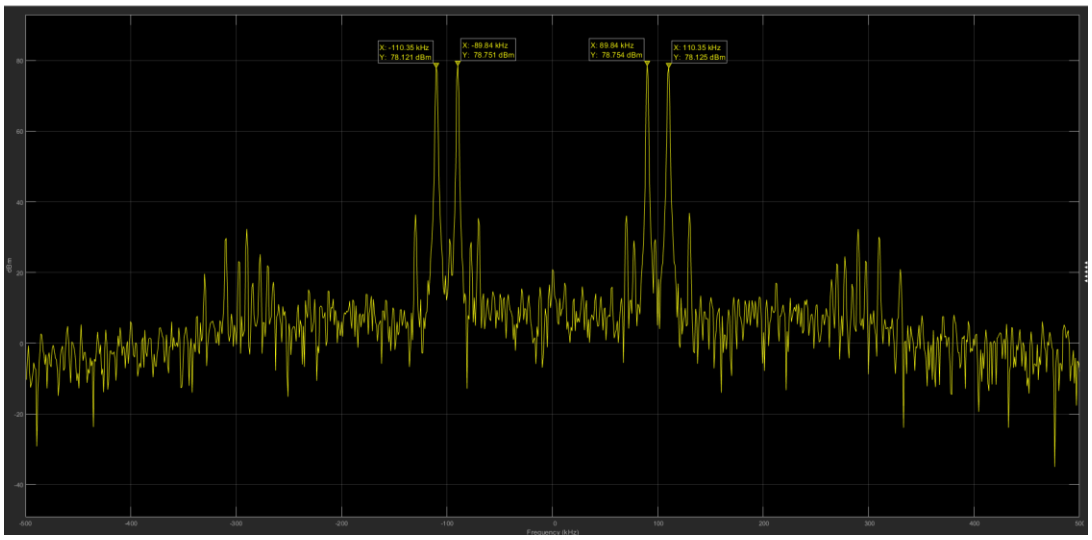
Εικόνα 4.1.1.2.3.1 Υλοποίηση πομπού DSB-AM-SC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364.



Εικόνα 4.1.1.2.3.2 Φάσμα πομπού DSB-AM-SC, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



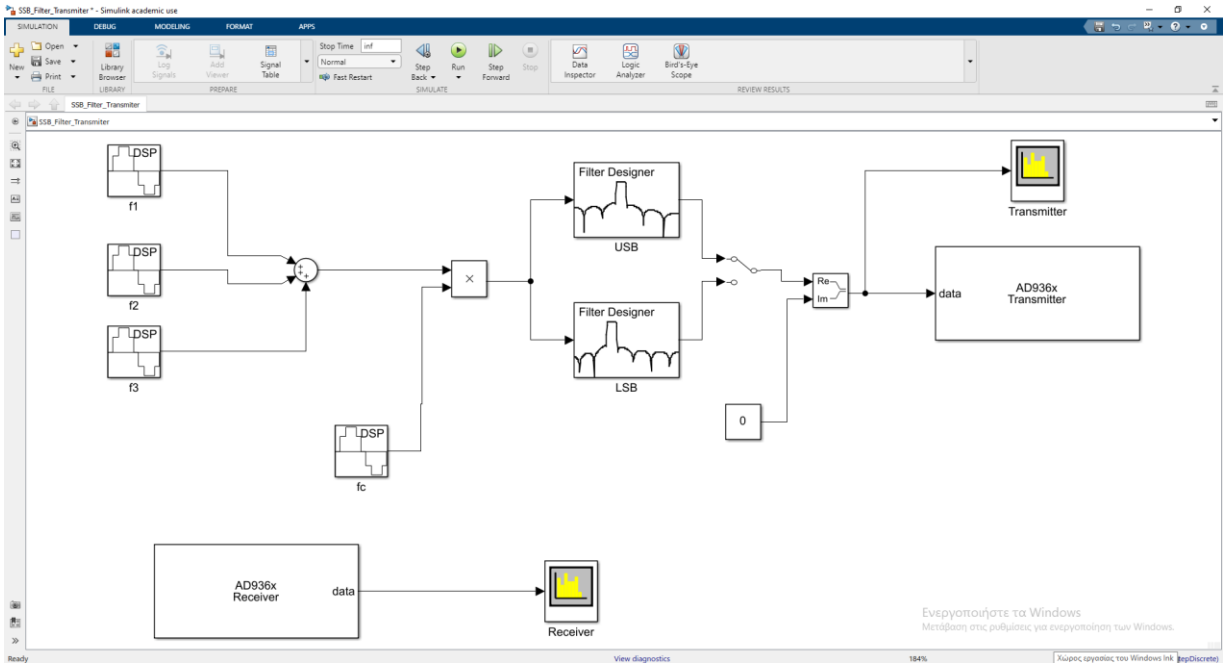
Εικόνα 4.1.1.2.3.3 Φάσμα πομπού DSB-AM-SC στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



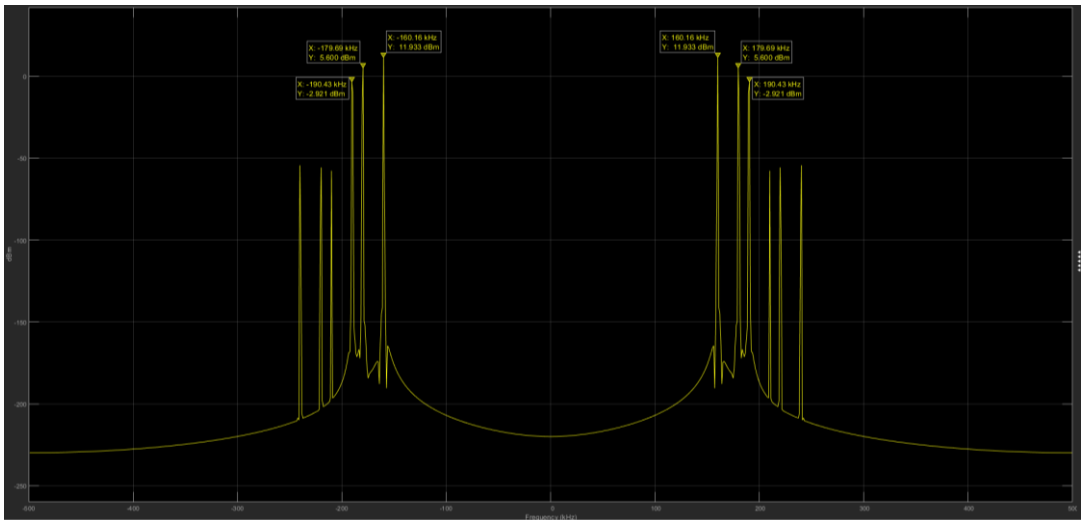
Εικόνα 4.1.1.2.3.4 Φάσμα δέκτη DSB-AM-SC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.

4.1.1.2.4 Υλοποίηση Πομπού Διαμόρφωση Μονής Πλευρικής Ζώνης με την χρήση φίλτρων (Single Side Band - SSB)

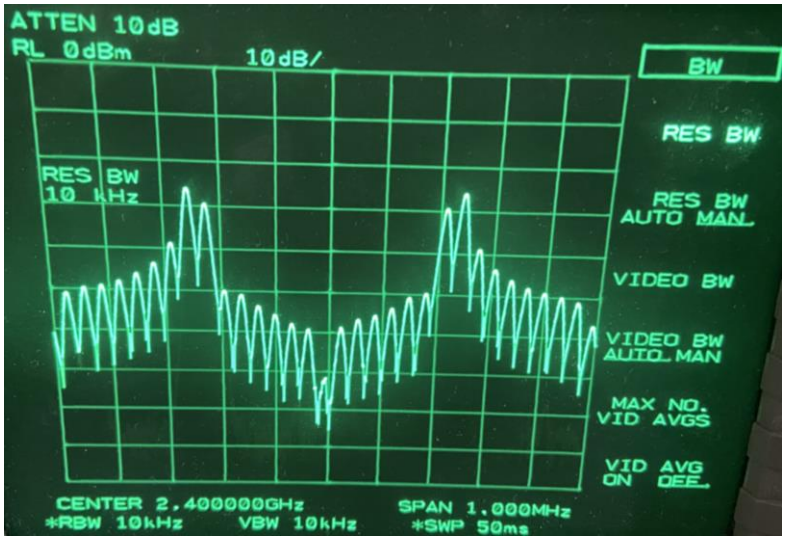
Στα πλαίσια του εργαστηρίου του μαθήματος «Τηλεπικοινωνίες» δημιουργήθηκε ένας πομπός Διαμόρφωση Απλής Πλευρικής Ζώνης (Single Side Band - SSB), με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364. Τρία ημίτονα με συχνότητες f_1=10KHz, f_2=20KHz, f_3=40KHz (Baseband), διαμορφώνονται από ένα φέρον (fc) με συχνότητα 200KHz. Το σήμα που παράγεται με την χρήση κατάλληλων φίλτρων , μετατρέπεται σε σήμα Άνω και Κάτω Πλευρικής Ζώνης (Upper Side Band – USB, Lower Side Band - LSB). Στην συνέχεια, με την χρήση ενός διακόπτη, επιλέγουμε ποια από τις δύο πλευρικές ζώνες (USB, LSB), θα διαμορφωθεί με την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, με συχνότητα 2.4GHz και θα εκπεμφθεί μέσω της κεραίας.



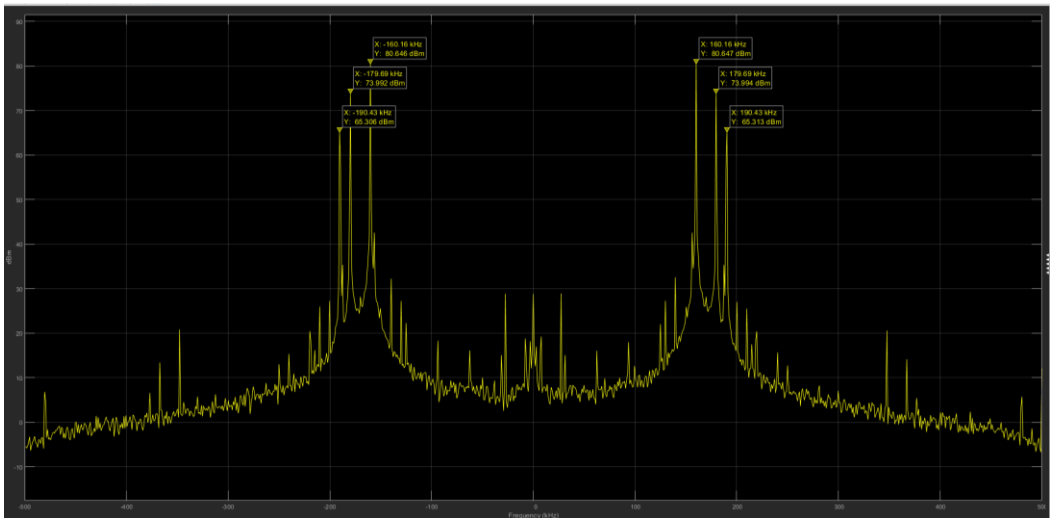
Εικόνα 4.1.1.2.4.1 Υλοποίηση πομπού SSB με την χρήση φίλτρων, μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364.



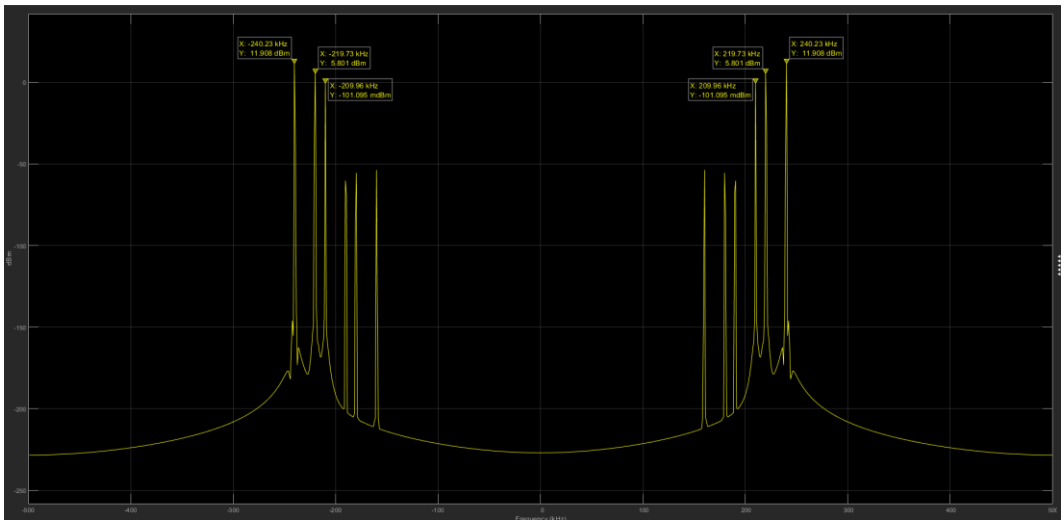
Εικόνα 4.1.1.2.4.2 Φάσμα πομπού USB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



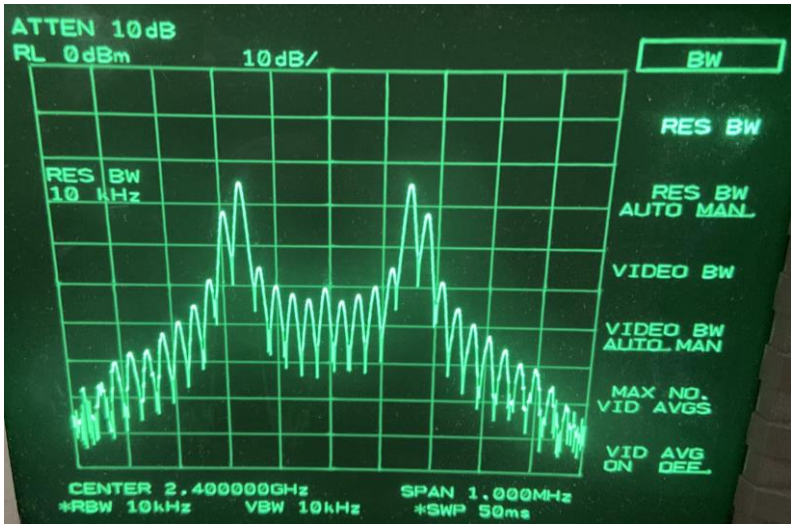
Εικόνα 4.1.1.2.4.3 Φάσμα πομπού USB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



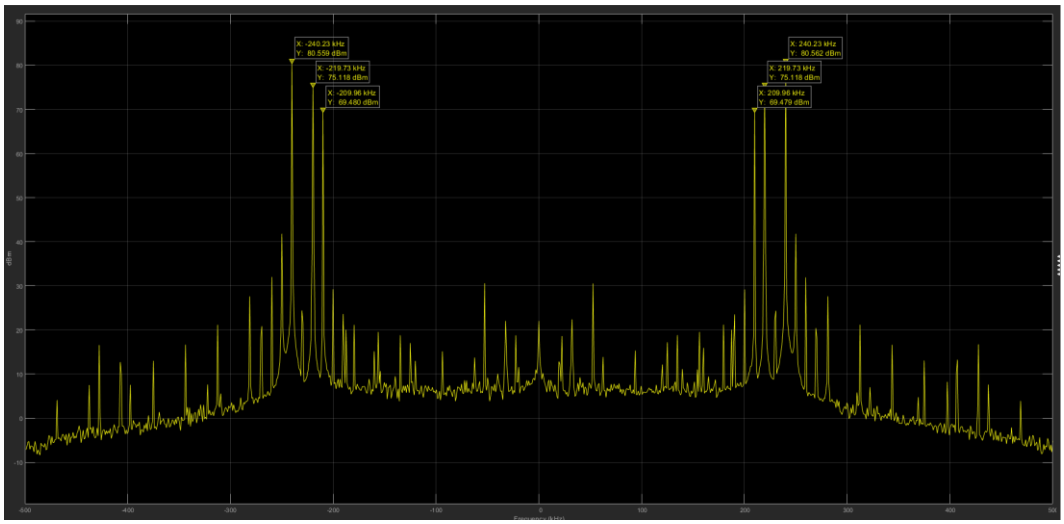
Εικόνα 4.1.1.2.4.4 Φάσμα δέκτη USB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



Εικόνα 4.1.1.2.4.5 Φάσμα πομπού LSB, πριν την διαμόρφωση του από την φέρουσα (Fc της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



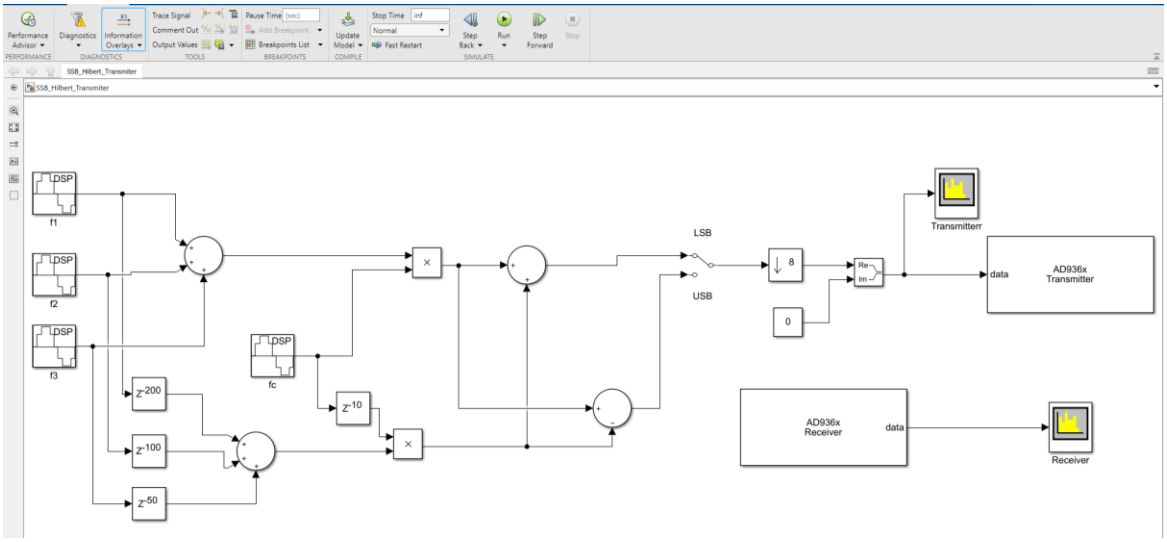
Εικόνα 4.1.1.2.4.6 Φάσμα πομπού LSB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



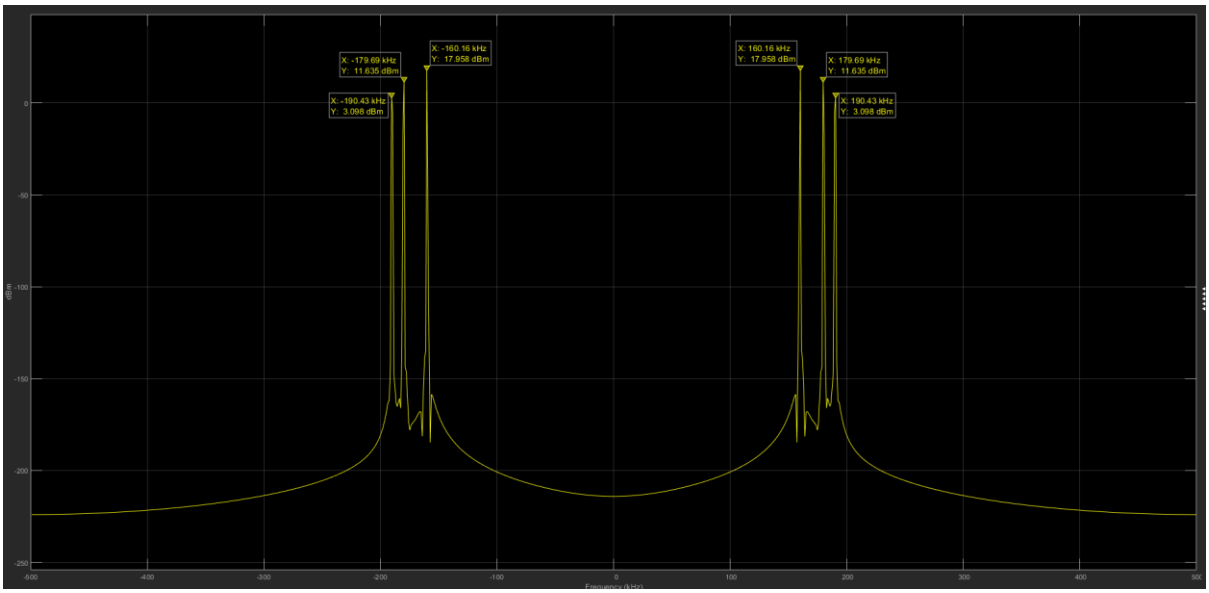
Εικόνα 4.1.1.2.4.7 Φάσμα δέκτη LSB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.

4.1.1.2.5 Υλοποίηση Πομπού Διαμόρφωση Μονής Πλευρικής Ζώνης με την χρήση μετασχηματισμού Hilbert (Hilbert transform Single Side Band - SSB) [105]

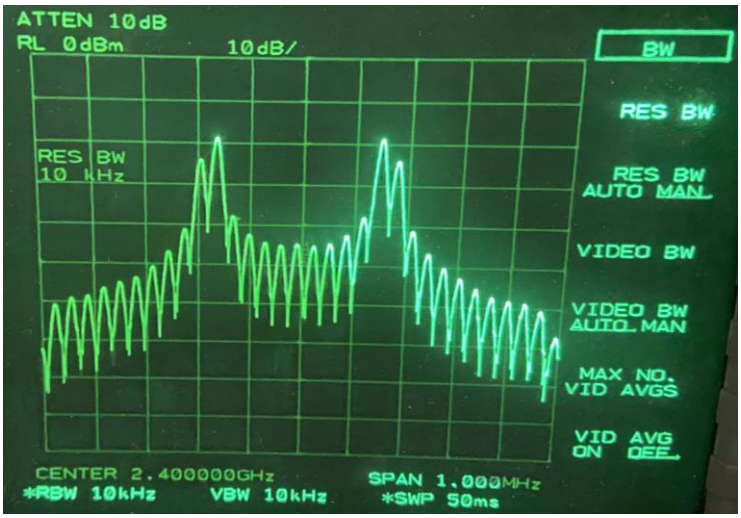
Στα πλαίσια του εργαστηρίου του μαθήματος «Τηλεπικοινωνίες» δημιουργήθηκε ένας πομπός Διαμόρφωση Απλής Πλευρικής Ζώνης (Single Side Band - SSB), με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, αξιοποιώντας τις ιδιότητες του μετασχηματισμού Hilbert. Τρία ημίτονα με συχνότητες f_1=10KHz, f_2=20KHz, f_3=40KHz (Baseband), διαμορφώνονται από ένα φέρον (fc) με συχνότητα 200KHz. Το σήμα που παράγεται με την χρήση του μετασχηματισμού Hilbert, μετατρέπεται σε σήμα Άνω και Κάτω Πλευρικής Ζώνης (Upper Side Band – USB, Lower Side Band - LSB). Στην συνέχεια, με την χρήση ενός διακόπτη, επιλέγουμε ποια από τις δύο πλευρικές ζώνες (USB, LSB), θα διαμορφωθεί με την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, με συχνότητα 2.4GHz και θα εκπεμφθεί μέσω της κεραίας.



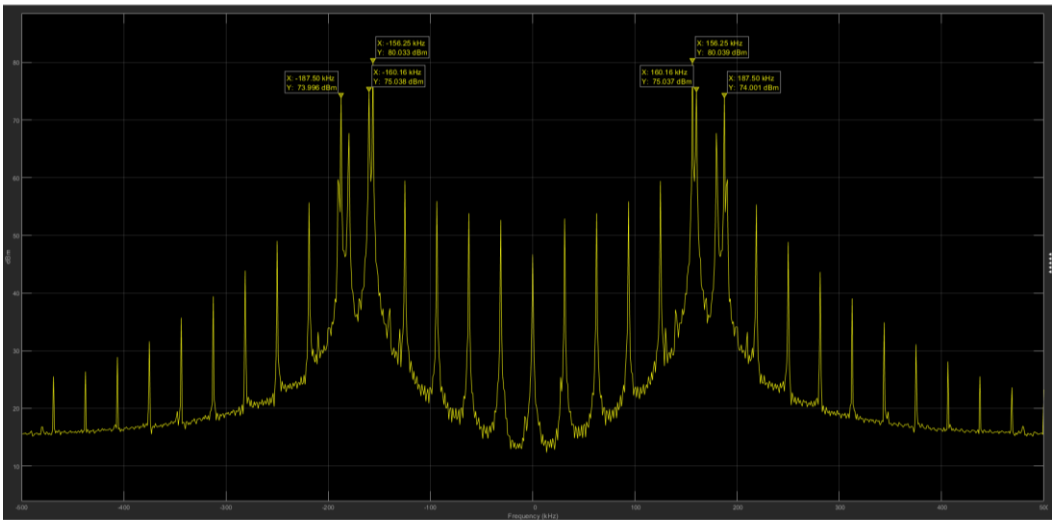
Εικόνα 4.1.1.2.5.1 Υλοποίηση πομπού SSB με την χρήση μετασχηματισμού Hilbert, μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364.



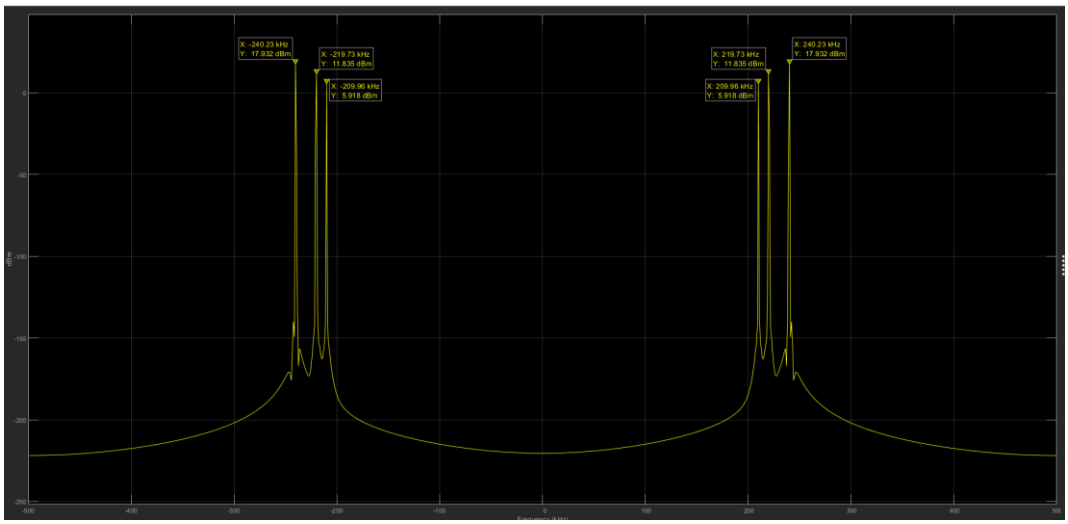
Εικόνα 4.1.1.2.5.2 Φάσμα πομπού LSB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



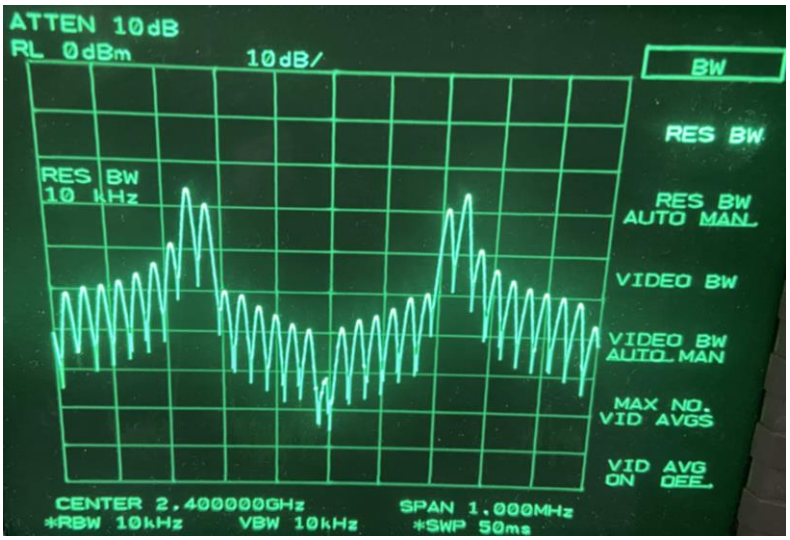
Εικόνα 4.1.1.2.5.3 Φάσμα πομπού LSB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



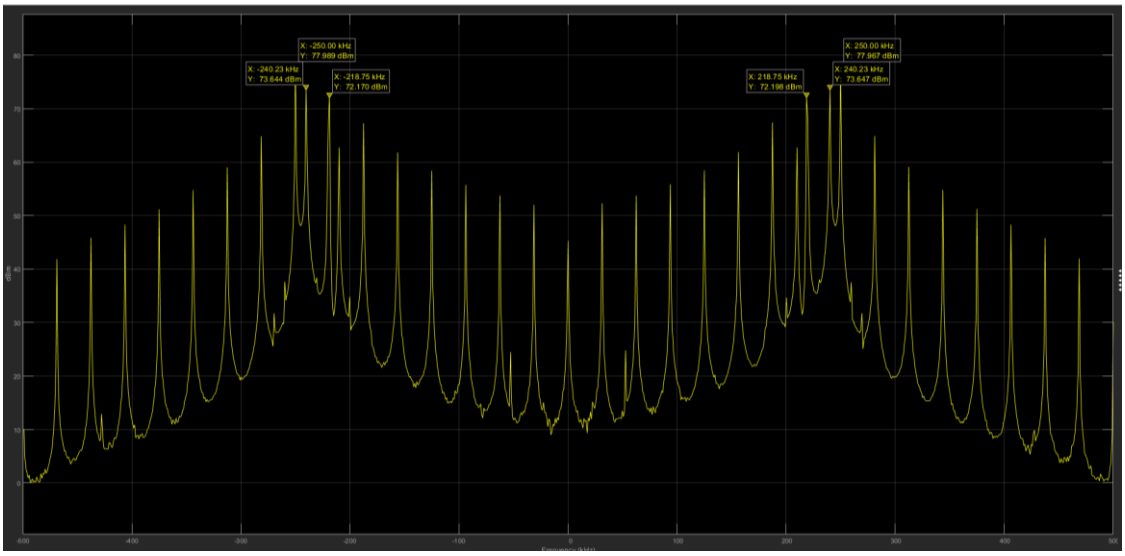
Εικόνα 4.1.1.2.5.4 Φάσμα δέκτη LSB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



Εικόνα 4.1.1.2.5.5 Φάσμα πομπού USB, πριν την διαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



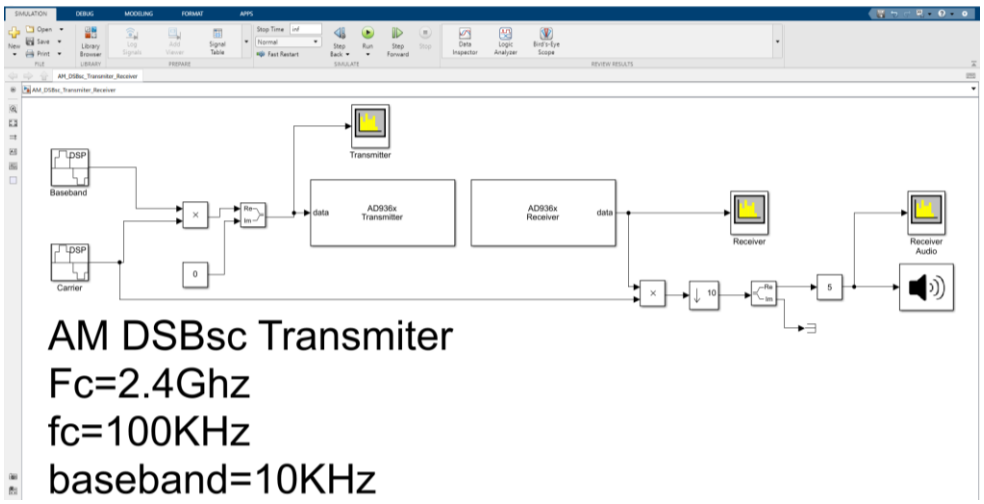
Εικόνα 4.1.1.2.5.6 Φάσμα πομπού USB, στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



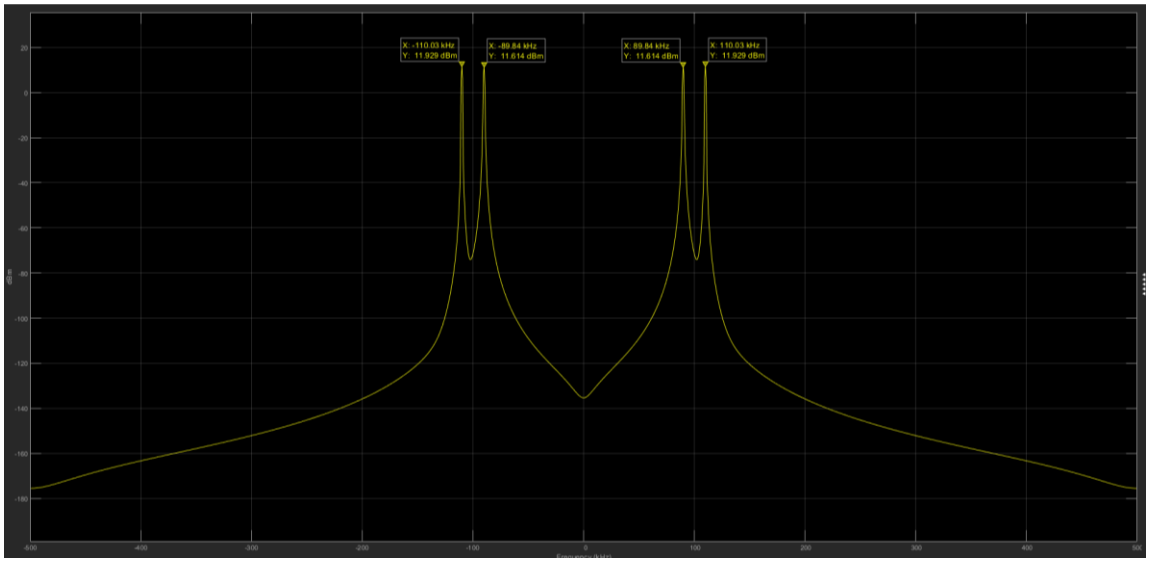
Εικόνα 4.1.1.2.5.7 Φάσμα δέκτη USB, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.

4.1.1.2.6 Υλοποίηση Πομποδέκτη Διαμόρφωσης Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC)

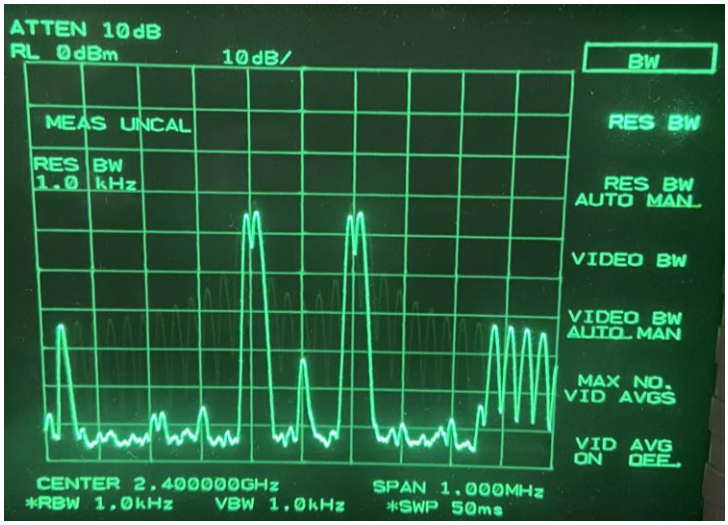
Στα πλαίσια του εργαστηρίου του μαθήματος «Τηλεπικοινωνίες» δημιουργήθηκε ένας πομποδέκτης Διαμόρφωση Πλάτους Διπλής Πλευρικής Ζώνης με κατεσταλμένο φέρον (Double Side Band-Amplitude Modulation-Suppressed Carrier - DSB-AM-SC), με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9361/AD9364. Ένα ημίτονο με συχνότητα 10KHz (Baseband), διαμορφώνεται από ένα φέρον (fc) με συχνότητα 100KHz. Το σήμα που παράγεται στην συνέχεια διαμορφώνεται με την φέρουσα (Fc) της πλακέτα Analog Devices AD9361/AD9364 με συχνότητα 2.4GHz και εκπέμπεται μέσω της κεραίας. Στην συνέχεια γίνεται λήψη από τον δέκτη της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, αποδιαμόρφωση του λαμβανόμενου σήματος από την φέρουσα της πλακέτα Fc=2.4GHz, αποδιαμόρφωση από την φέρουσα fc=100KHz και λήψη του αρχικού ημιτόνου με συχνότητα 10KHz (baseband σήμα). Ο σήμα αυτό αναπαράγεται από τα ηχεία του υπολογιστή, μετά από κατάλληλη επεξεργασία.



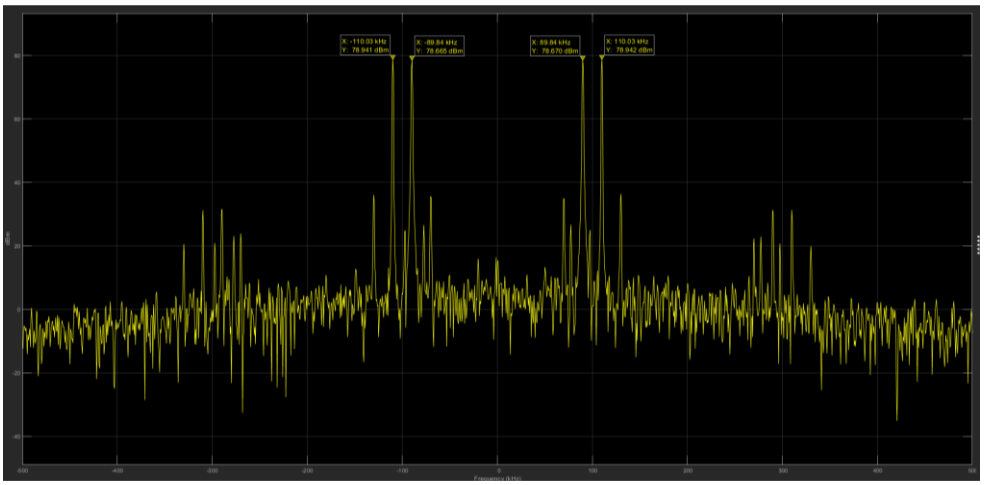
Εικόνα 4.1.1.2.6.1 Υλοποίηση πομποδέκτη DSB-AM-SC με την χρήση με την χρήση μπλοκ του Simulink, της αναπτυξιακής πλακέτας ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364.



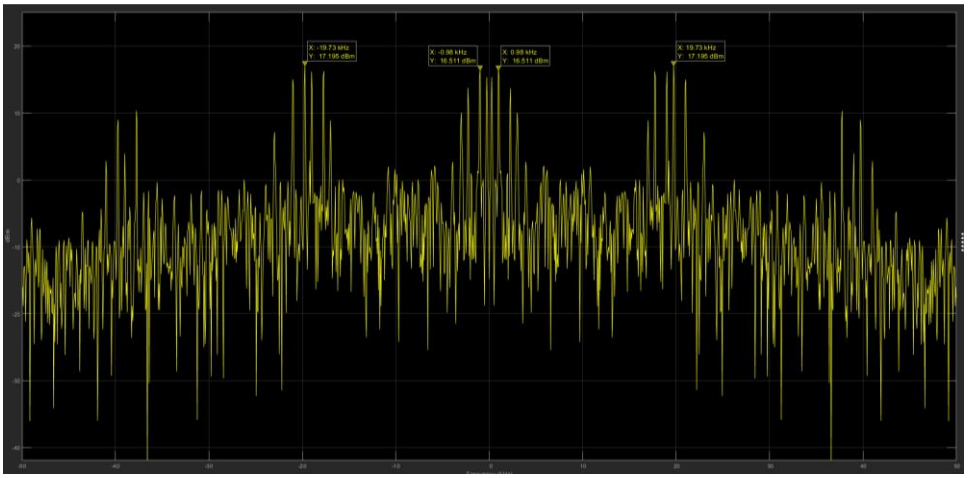
Εικόνα 4.1.1.2.6.2 Φάσμα πομπού DSB-AM-SC, πριν την διαμόρφωση του από την φέρουσα (Fc) της της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



Εικόνα 4.1.1.2.6.3 Φάσμα πομπού DSB-AM-SC στην έξοδο της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε εργαστηριακό αναλυτή φάσματος.



Εικόνα 4.1.1.2.6.4 Φάσμα δέκτη DSB-AM-SC, μετά την αποδιαμόρφωση του από την φέρουσα (Fc) της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC με το Analog Devices AD9364, σε αναλυτή φάσματος του Simulink.



Εικόνα 4.1.1.2.6.5 Φάσμα στην είσοδο του μεγαφώνου, σε αναλυτή φάσματος του Simulink.

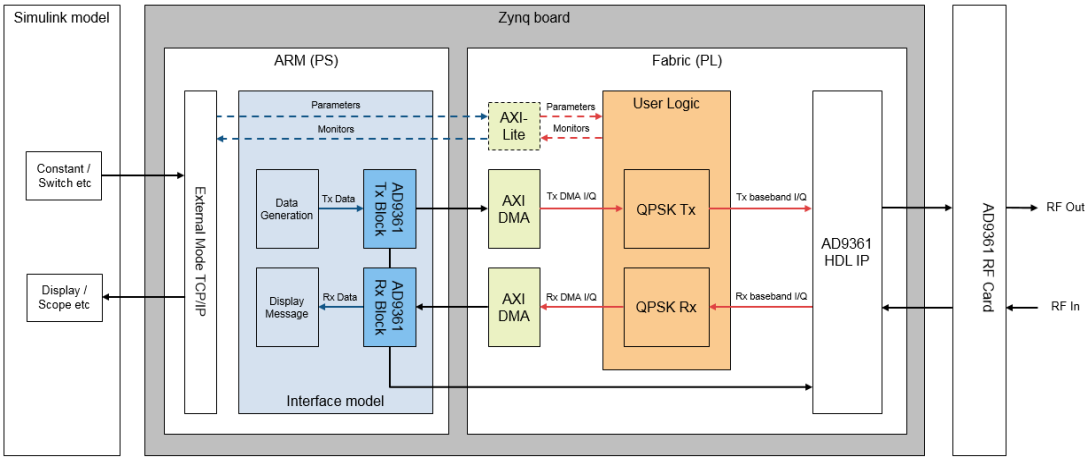
4.1.2 Πλήρης Αυτόνομη Ανάπτυξη- Stand Alone

Στο καθεστώς Αυτόνομης Ανάπτυξης - Stand Alone, η πλακέτα ZedBoard και η RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC, χρησιμοποιούνται για την εκπομπή και λήψη του σήματος του αναπτυσσόμενου/σχεδιαζόμενου τηλεπικοινωνιακού συστήματος SDR, χωρίς να είναι αναγκαία οπουδήποτε επικοινωνία με το MatLab ή το Simulink. Το αναπτυσσόμενο τηλεπικοινωνιακό σύστημα SDR, αρχίζει να εκπέμπει και να λαμβάνει. διαμέσου της πλακέτα ZedBoard και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC, από την χρονική στιγμή που ενεργοποιηθεί και για όσο χρονικό διάστημα ενεργοποιημένη, εκτελώντας τον αλγόριθμο με τον οποίο έχει προγραμματιστεί. Στην περίπτωση αυτή, το MatLab και το Simulink, χρησιμοποιούνται για τον αρχικό προγραμματισμό της αναπτυξιακής πλακέτα ZedBoard, αλλά με την ανάπτυξη του αλγόριθμού σε αυτή, δεν απαιτείται οποιοδήποτε σύνδεση του υλικού με το λογισμικό. Το υλικό (η πλακέτα ZedBoard και η RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC), λειτουργούν εντελώς αυτόνομα από το λογισμικό (MatLab, Simulink).

Στο παράδειγμα που ακολουθεί, θα παρουσιαστεί η αναλυτική ανάπτυξη ενός πλήρως αυτόνομου τηλεπικοινωνιακού συστήματος SDR.

4.1.2.1 Παράδειγμα Υλοποίησης QPSK πομπού, χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης - Stand Alone.

Για την υλοποίηση ενός πλήρως Αυτόνομου πομπού, χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364, θα χρησιμοποιηθεί σαν βάση, το αντίστοιχο παράδειγμα της MathWorks [107].
Αυτό το παράδειγμα δείχνει τον τρόπο χρήσης της αναπτυξιακής πλακέτας ZedBoard, για την υλοποίηση ενός πομποδέκτη QPSK, με την χρήση έτοιμων μπλοκ του Simulink σε καθεστώς Αυτόνομης Ανάπτυξης – Stand Alone. Η συσκευή SDR σε αυτό το μοντέλο θα μεταδίδει συνεχώς δεδομένα, τα οποία στην συνέχεια διαμορφώνονται σε QPSK με καθορισμένη κεντρική συχνότητα στα 2.4GHz. Στην συνέχεια, τα δεδομένα λαμβάνονται και αφού αποδιαμορφωθούν, προβάλλονται στην επιφάνεια εργασίας του MatLab.
Ο αλγόριθμους υλοποίησης ενός πομποδέκτη QPSK, παρουσιάζεται στην εικόνα 4.1.2.1.1. Σε αυτήν την εικόνα, παρουσιάζεται η χαρτογράφηση του επεξεργαστή ARM (Processing System-PS), είναι υπεύθυνος για την εκτέλεση των εργασιών της κωδικοποίησης, αποκωδικοποίησης και χαμηλού ρυθμού εργασίες επεξεργασίας. Επίσης παρουσιάζεται η χαρτογράφηση της Προγραμματιζόμενης Λογικής - Programmable Logic (PL), η οποία αποτελείται από Συστοιχίες επιτόπιας προγραμματιζόμενων πυλών - Field Programmable Gate Array (FPGA), στην οποία ενσωματώνεται το αναπτυσσόμενο τηλεπικοινωνιακό σύστημα SDR. Τα μεταδιδόμενα και τα λαμβανόμενα σήματα βασικής ζώνης, αποστέλλονται και λαμβάνονται χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364.



Εικόνα 4.1.2.1.1 Μπλοκ διάγραμμα πομποδέκτη QPSK χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης - Stand Alone. [107].

Η Προγραμματιζόμενη Λογική - Logic Programmable (PL) του FPGA, χρησιμοποιείται για την επεξεργασία σημάτων υψηλής ταχύτητας ενώ το Σύστημα Επεξεργασίας - Processing System (PS) του ARM, χρησιμοποιείται για χαμηλότερους ρυθμούς, λειτουργίες ελέγχου. Σε αυτό το παράδειγμα, οι λειτουργίες του πομποδέκτη QPSK, υλοποιούνται σύμφωνα με την προγραμματιζόμενη λογική (PL), καθώς αυτές περιλαμβάνουν λειτουργίες υψηλού ρυθμού όπως έλεγχος κέρδους, φιλτράρισμα και αντιστάθμιση συχνότητας. Η κωδικοποίηση και η αποκωδικοποίηση δεδομένων είναι πολύ πιο αργές σαν διαδικασίες και εφαρμόζονται στον ARM (PS), ο οποίος αποκωδικοποιεί το μήνυμα και το στέλνει στην επιφάνεια εργασίας του MatLab. Το IP Core Generation Work του Simulink, χρησιμοποιείται για την υλοποίηση του FPGA IP Core και τη δημιουργία του μοντέλου διεπαφής λογισμικού (Interface Model).
Παρόλο αυτά, το παραγόμενο μοντέλο θα πρέπει να ικανοποίει κάποιες συγκεκριμένες απαιτήσεις.

Σύνθετες είσοδοι και έξοδοι (Complex Input/Output) δεν υποστηρίζονται στις θύρες του υποσυστήματος HDL. Για τον λόγο αυτόν, τα πραγματικά και φανταστικά σήματα (Image – Real), πρέπει να διαμορφωθούν κατάλληλα πριν την χρησιμοποίηση τους στο εκάστοτε υποσύστημα.

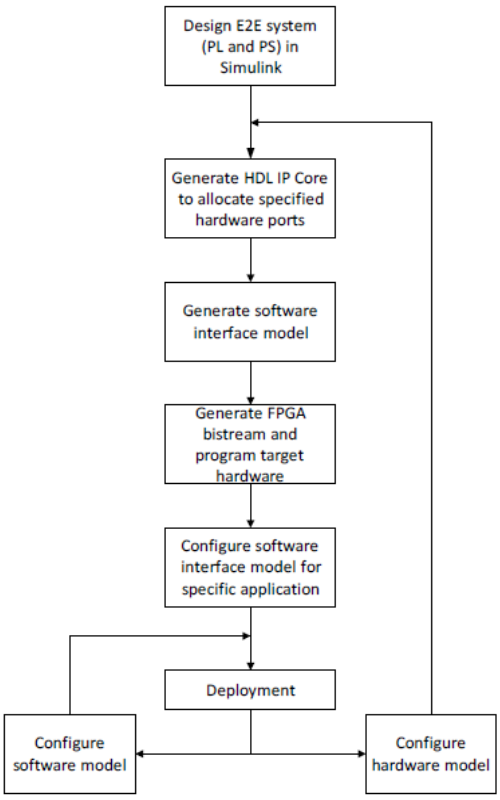
Τα δεδομένα εισόδου και εξόδου στο υποσύστημα μοντελοποιούνται χρησιμοποιώντας ξεχωριστά σήματα δεδομένων (Data) και ελέγχου (Valid).

Οι ρυθμοί ρολογιού εισόδου και εξόδου κάθε υποσυστήματος πρέπει να είναι ίσοι. Ειδικότερά στο Simulink τα σήματα δεδομένων και ελέγχου πρέπει να κινούνται με τον ίδιο ρυθμό δείγματος.
Στο συγκεκριμένο παράδειγμα έχει προστεθεί μία Θύρα ελέγχου AXI4-lite txSrcSelect, προκειμένου να υπάρχει κάποιος έλεγχος στα μεταδιδόμενα δεδομένα. Η θύρα txSrcSelect στο υποσύστημα HDL χρησιμοποιείται για την επιλογή μεταξύ δύο διαφορετικών πηγών δεδομένων για τον πομπό. Εάν η θύρα txSrcSelect βρίσκεται σε υψηλό δυναμικό (True), η πηγή δεδομένων για τον πομπό είναι ένας πίνακας αναζήτησης που είναι αποθηκευμένος στο FPGA και τα λαμβανόμενα δεδομένα θα μοιάζουν με τις συμβολοσειρές "Hello World 0XX". Εάν η θύρα txSrcSelect είναι σε χαμηλό δυναμικό (False), η πηγή δεδομένων για τα ληφθέντα δεδομένα θα είναι ο επεξεργαστής ARM, ο οποίος θα δημιουργήσει δείγματα σε πραγματικό χρόνο και θα τα στείλει στον πομπό στο FPGA. Το μήνυμα σε αυτήν την περίπτωση θα είναι "*Zynq HW/SW Co-Design*".
Τέλος, στο παράδειγμα έχει προστεθεί μία Θύρα Ελέγχου AXI4-lite rxStreamEnable, η οποία διασφαλίζει ότι ο αλγόριθμος HDL επεξεργάζεται δεδομένα μόνο όταν αυτή η είσοδος είναι αληθής. Αυτό συμβαίνει, μόνο όταν ο αλγόριθμος HDL πομπού έχει αρχίσει να μεταδίδει έγκυρα δεδομένα, διασφαλίζοντας ότι ο δέκτης κλειδώνει σωστά.

4.1.2.2 Υλοποίηση πομπού QPSK χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364 στο καθεστώς Αυτόνομης Ανάπτυξης – Stand Alone.

4.1.2.2.1 Βήματα υλοποίησης Τηλεπικοινωνιακού Συστήματος SDR.

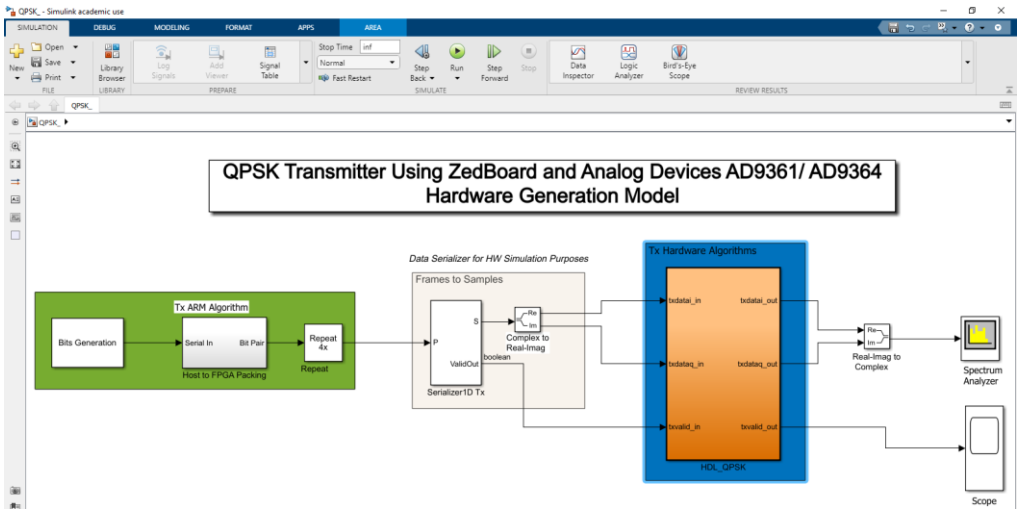
Το διάγραμμα ροής εργασίας, για την υλοποίηση ενός τηλεπικοινωνιακού συστήματος SDR- End-to-End Digital Wireless Communication System (E2E), χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9361/AD9364 παρουσιάζεται στην εικόνα 4.1.2.2.1.1, που ακολουθεί.



Εικόνα 4.1.2.2.1.1 Διάγραμμα ροής σχεδίασης και υλοποίησης τηλεπικοινωνιακού συστήματος SDR. [108].

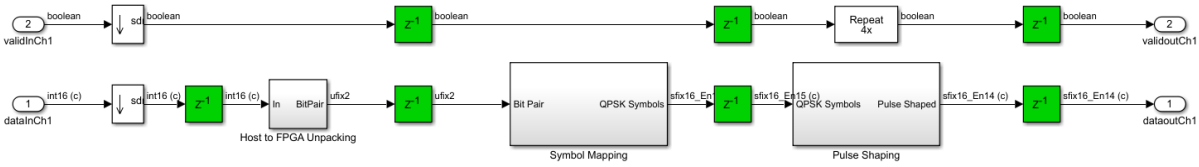
4.1.2.2.2 Σχεδίαση Τηλεπικοινωνιακού συστήματος SDR στο Simulink.

Για την υλοποίηση ενός πλήρως Αυτόνομου πομπού, χρησιμοποιώντας την αναπτυξιακή πλακέτα ZedBoard της Xilinx και την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC με το Analog Devices AD9364, σχεδιάστηκε ένας πομπός QPSK στο Simulink, χρησιμοποιώντας σαν βάση, το αντίστοιχο παράδειγμα της MathWorks [107]. Το τελικό μοντέλο μου δημιουργήθηκε, παρουσιάζεται στην εικόνα 4.1.2.2.2.1.

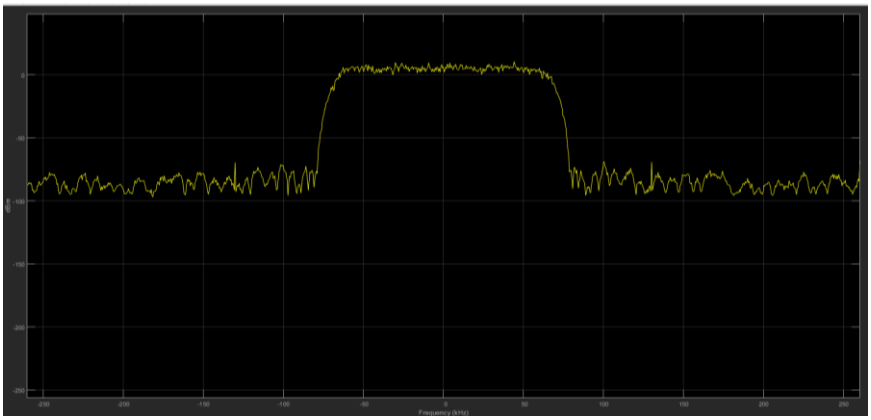


Εικόνα 4.1.2.2.2.1 Μπλοκ Διάγραμμα QPSK πομπού στο Simulink.

Το πράσινο πλαίσιο περιέχει τα υποσυστήματα που συγκροτούν το Σύστημα Επεξεργασίας - Processing System (PS) του ARM, ενώ το μπλε πλαίσιο τα αντίστοιχα υποσυστήματα που συγκροτούν την Προγραμματιζόμενη Λογική - Logic Programmable (PL) του FPGA. Το παραπάνω μπλοκ διάγραμμα περιέχει κάποια επιπλέον μπλοκ που χρησιμοποιούνται στο στάδιο της εξομοίωσης του αναπτυσσόμενου υπολογιστικού συστήματος SDR (παλμογράφος, αναλυτής φάσματος, Serializer).



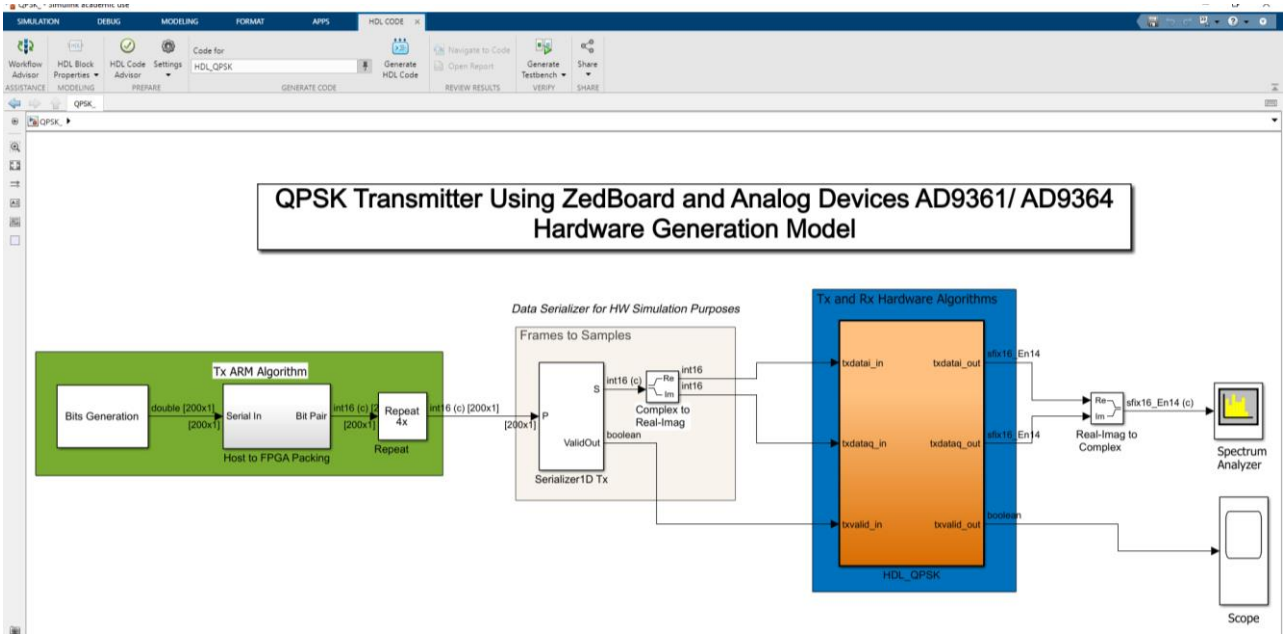
Εικόνα 4.1.2.2.2.2 Μπλοκ Διάγραμμα Αλγόριθμου QPSK πομπού στην Προγραμματιζόμενη Λογική - Logic Programmable (PL) του FPGA, στο Simulink.



Εικόνα 4.1.2.2.2.3 Εξομοίωση QPSK πομπού στο Simulink πριν την διαμόρφωση του από την RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC.

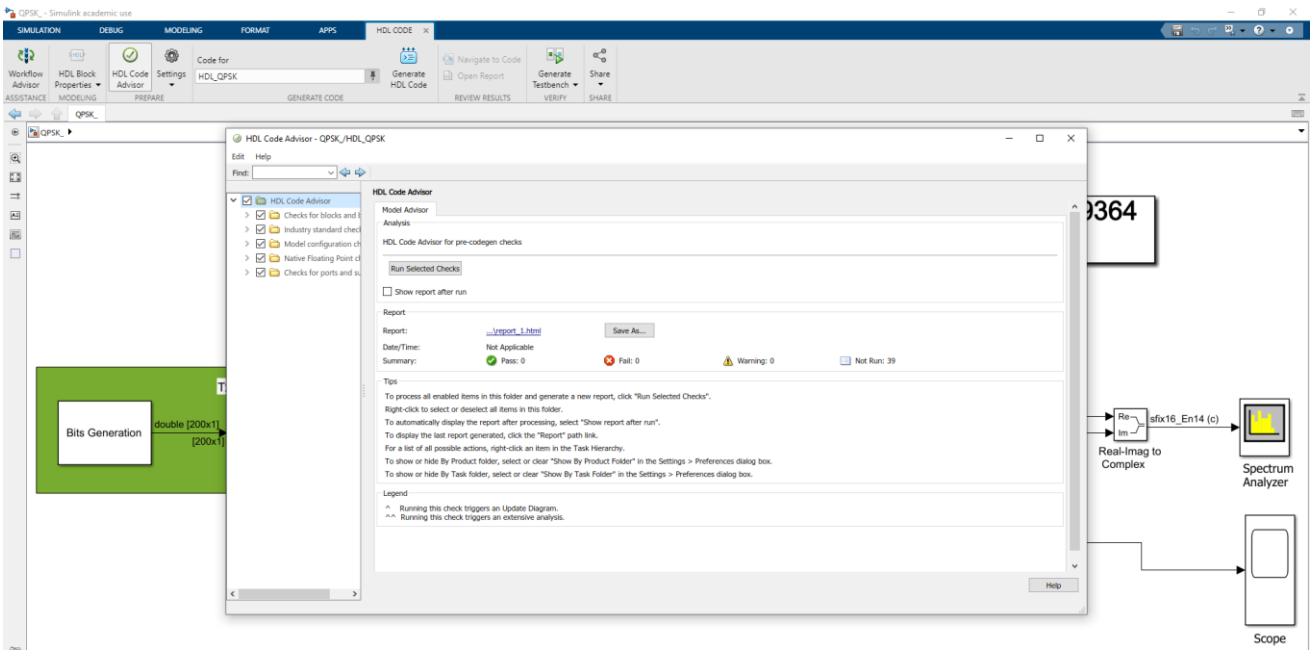
4.1.2.2.3 Δημιουργία Μοντέλου υλικού - Hardware Generation Model.

Για την υλοποίηση του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR της εικόνας 4.1.2.2.2.1, είναι αναγκαίο να μετατραπεί σε μοντέλο συμβατό με την γλώσσα HDL, προκειμένου να μπορέσει να ενσωματωθεί στο FPGA, χρησιμοποιώντας την αντίστοιχη ροή εργασίας του Simulink HDL Core. Το Simulink HDL Core απλοποιεί την διαδικασία μετάφρασης του μοντέλου σχεδιασμού υψηλού επιπέδου σε μοντέλο συμβατό με την γλώσσα HDL, το οποίο στην συνέχεια θα ενσωματωθεί ως αρχείο bitstream στο FPGA.
Προκειμένου να γίνει αυτό μεταβαίνουμε στην καρτέλα APPS του Simulink και επιλέγουμε την εφαρμογή HDL Core.

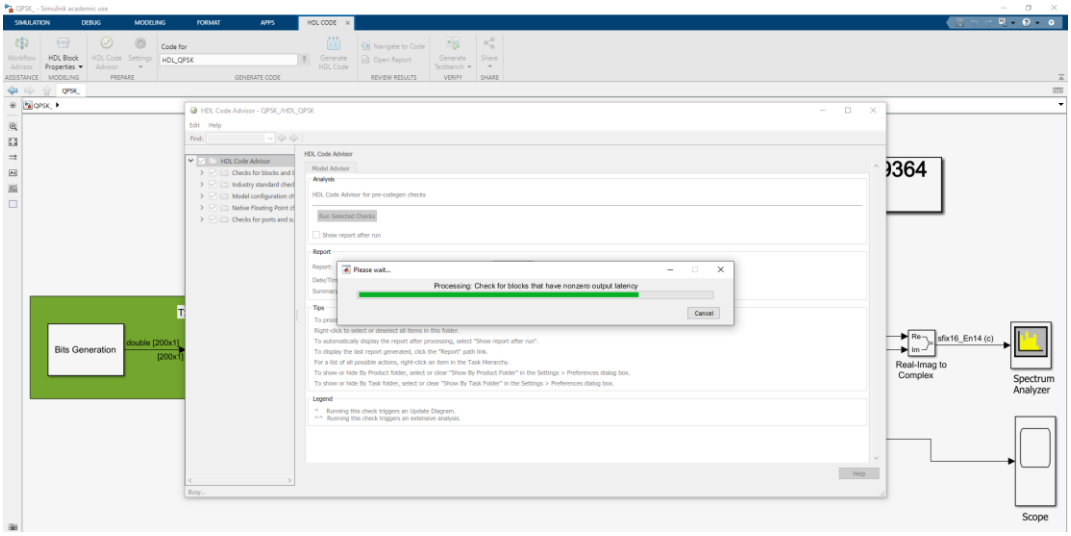


Εικόνα 4.1.2.2.3.1 HDL Core του Simulink.

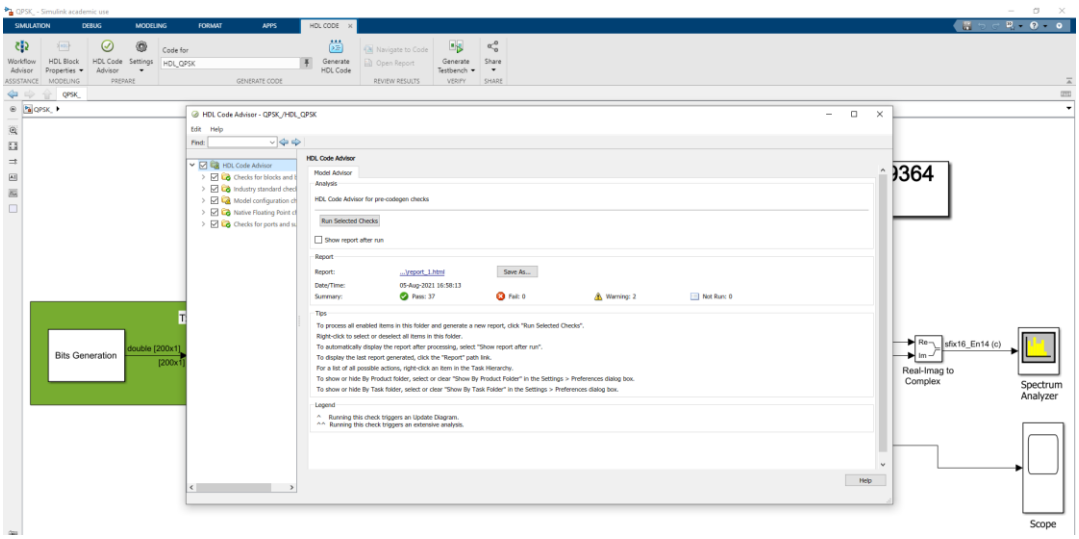
Στην καρτέλα αυτή επιλέγουμε το HDL Core Advisor ξεκινάμε τον έλεγχο του σχεδιαζόμενου μοντέλου με την γλώσσα HDL.



Εικόνα 4.1.2.2.3.2 HDL Core Advisor του Simulink.

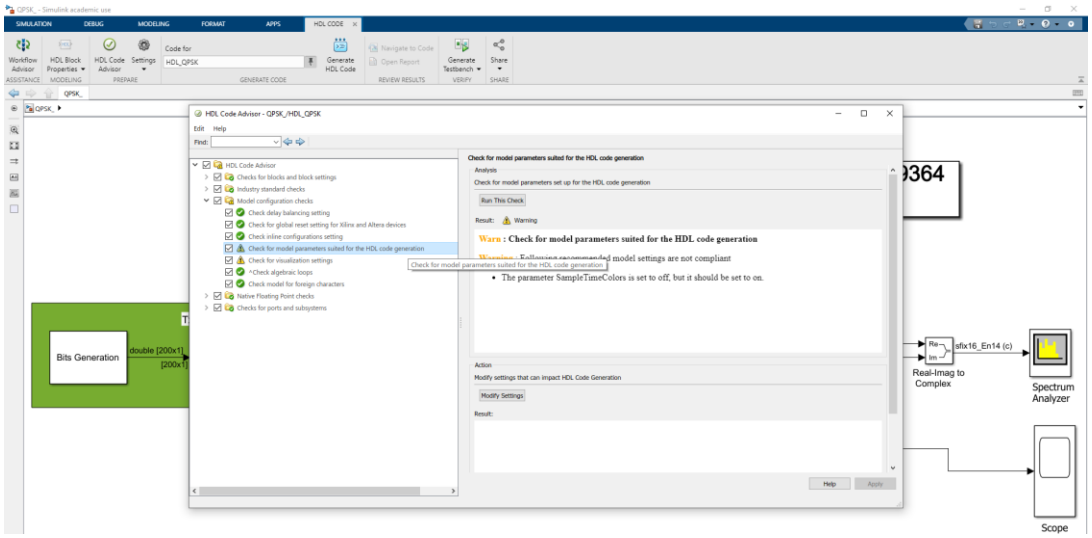


Εικόνα 4.1.2.2.3.3 Έλεγχος μοντέλου με το HDL Core Advisor του Simulink.

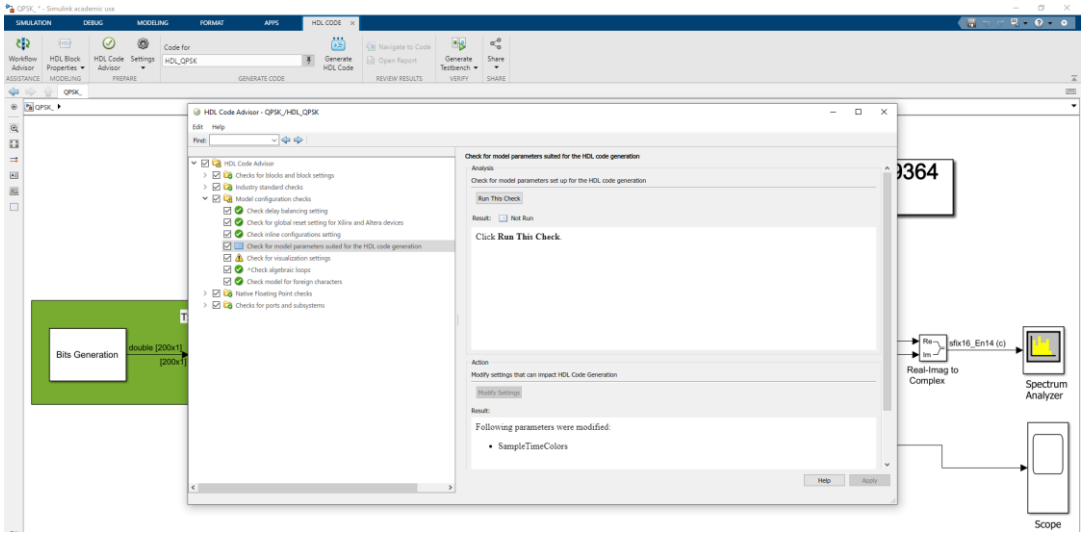


Εικόνα 4.1.2.2.3.4 HDL Core Advisor του Simulink.

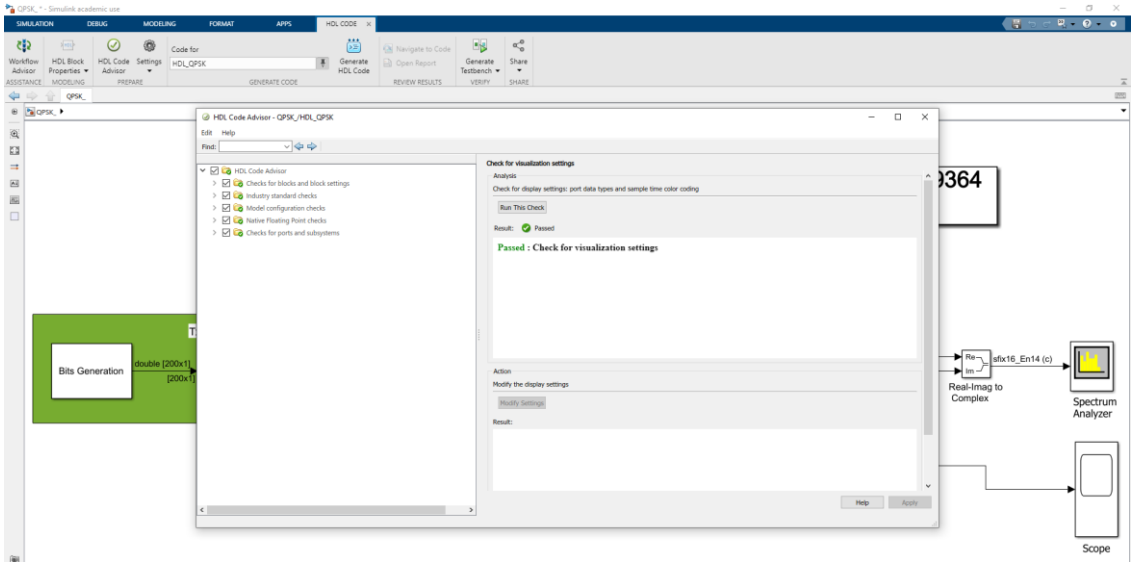
Αν κατά την διάρκεια του ελέγχου εμφανιστούν στο μοντέλο σφάλματα (Errors) ή προειδοποιήσεις (Warnings), αφού τα επιλέξουμε, προσπαθούμε να τα επιλύσουμε. Σημαντικό είναι να μην υπάρχει στο μοντέλο μας κανένα σφάλμα και όσο τον δυνατό λιγότερες προειδοποιήσεις.



Εικόνα 4.1.2.2.3.5 Προειδοποίηση (Warning) του HDL Core Advisor στο εξεταζόμενο μοντέλο.



Εικόνα 4.1.2.2.3.6 Επιδιόρθωση προειδοποίησης (Warning) του HDL Core Advisor στο εξεταζόμενο μοντέλο.

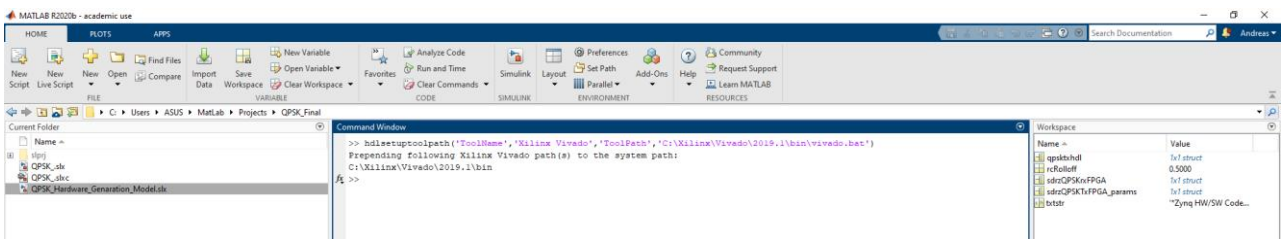


Εικόνα 4.1.2.2.3.7 Μοντέλο χωρίς σφάλματα (Errors) ή προειδοποιήσεις (Warnings) του HDL Core Advisor.

4.1.2.2.4 Διάγραμμα Ροής IP Core - IP Core Generation Workflow.

Μετά την ολοκλήρωση του ελέγχου συμβατότητας του εξεταζόμενου μοντέλου με την γλώσσα HDL, μπορεί να ξεκινήσει η διαδικασία δημιουργίας του HDL IP Core, δημιουργώντας το λογισμικό που θα ενσωματωθεί στο FPGA και θα εκτελεστεί στον ARM.
Θα πρέπει προηγουμένως να έχει ρυθμιστεί το εργαλείο Vivado της Xilinx. Αυτό θα μπορεί να γίνει εκτελώντας την ακόλουθη εντολή στην επιφάνεια εργασίας του MatLab:

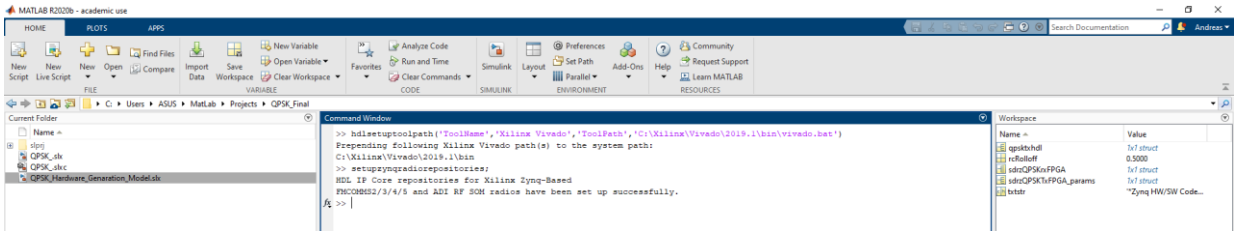
```
>> hdlsetuptoolpath('ToolName','Xilinx
Vivado','ToolPath','C:\Xilinx\Vivado\2019.1\bin\vivado.bat');
```



Εικόνα 4.1.2.2.4.1 Ενεργοποίηση του εργαλείου Vivado της Xilinx.

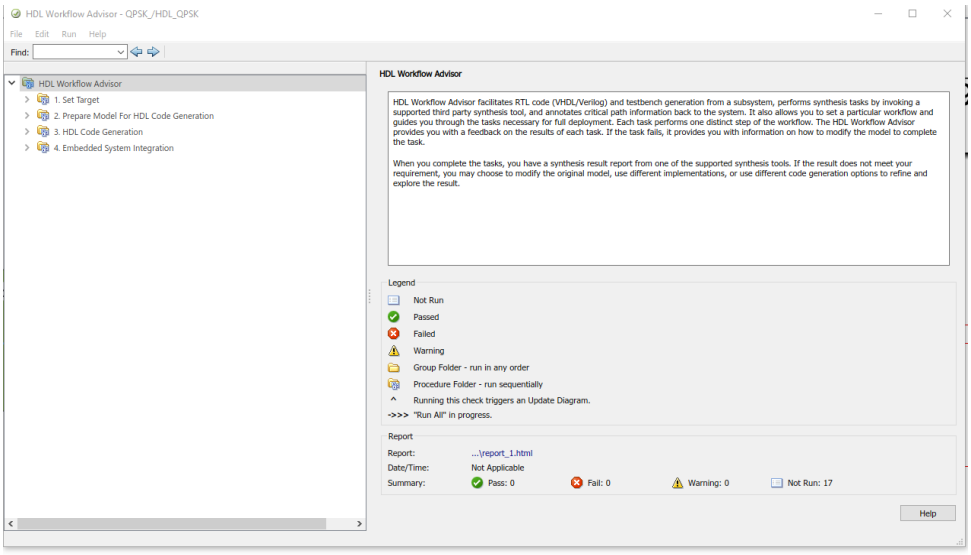
Αν δεν έχει ενεργοποιηθεί το εργαλείο Vivado της Xilinx, στον έλεγχο 1.1 του IP Core Generation, δεν θα είναι δυνατή η επιλογή «Synthesis Tool» και η διαδικασία θα οδηγήσει σε σφάλμα (Error) και θα σταματήσει.
Επίσης, θα πρέπει να πρέπει να έχει δημιουργηθεί το IP HDL Core repositories της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC. Αυτό μπορεί να γίνει εκτελώντας την εντολή:

```
>> setupzynqradiorepositories;
```

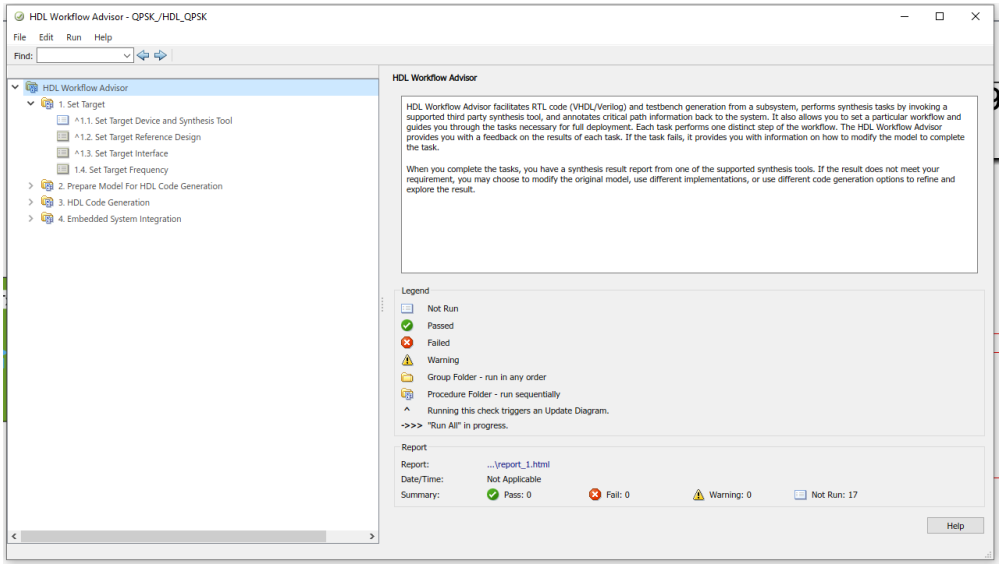


Εικόνα 4.1.2.2.4.2 Ενεργοποίηση του IP HDL Core repositories.

Στην συνέχεια ξεκινάει η διαδικασία δημιουργίας του HDL IP Core, επιλέγοντας την καρτέλα HDL Code / HDL Workflow Advisor.

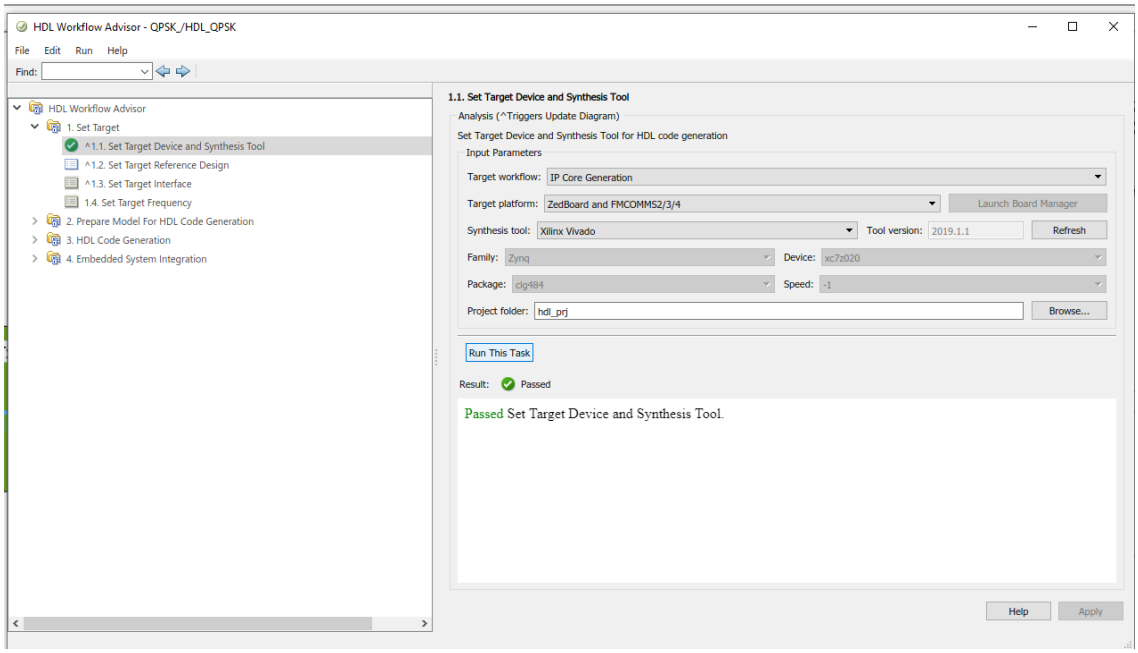


Εικόνα 4.1.2.2.4.3 HDL Workflow Advisor (1/2).



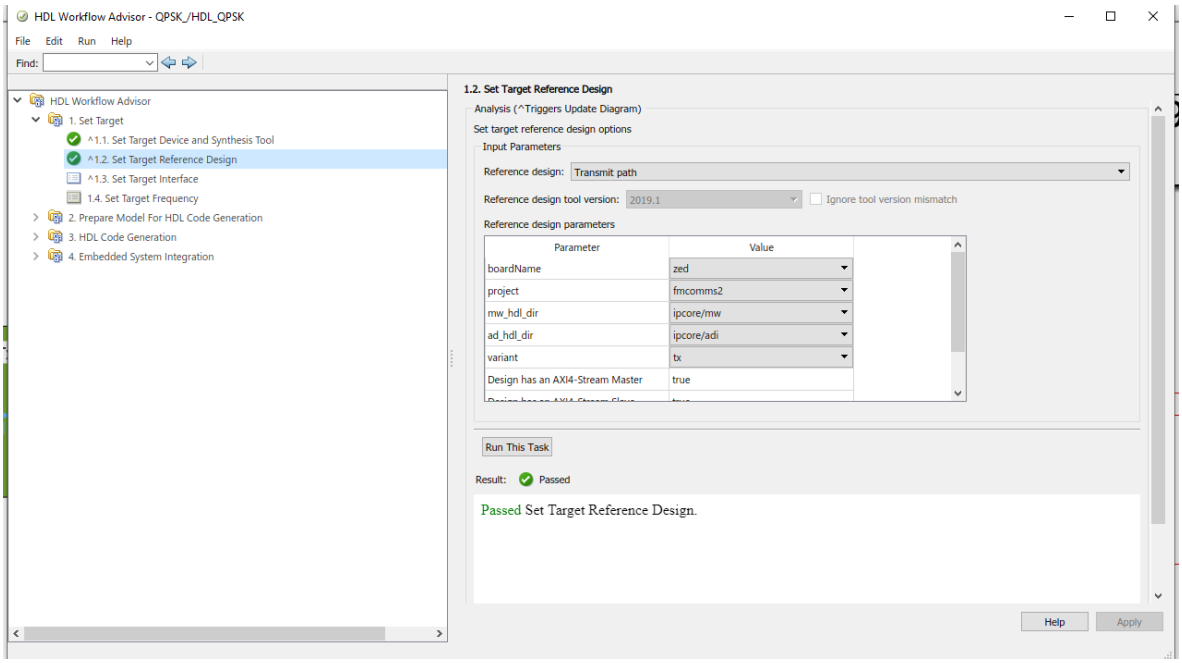
Εικόνα 4.1.2.2.4.4 HDL Workflow Advisor (2/2).

- Στο Βήμα 1.1, επιλέγετε η ροή εργασιών IP Core Generation και την κατάλληλη αναπτυξιακή πλακέτα Zynq. Στην προκειμένη περίπτωση επιλέγουμε το ZedBoard και AD-FMCOMMS4 FMC. Αν έχει ενεργοποιηθεί το εργαλείο Vivado της Xilinx, το σύστημα θα το επιλέξει αυτόματο στην επιλογή «Synthesis Tool». Αν δεν εμφανίζεται αυτόματα, χρειάζεται να το ενεργοποιήσουμε, όπως αναφέρθηκε προηγουμένως. Στην συνέχεια, τρέχουμε τις ρυθμίσεις. Μετά την εκτέλεση των ρυθμίσεων, θα πρέπει να μην υπάρχει σφάλμα ή προειδοποίηση (Error – Warning) και να εμφανιστεί η ένδειξη Passed.



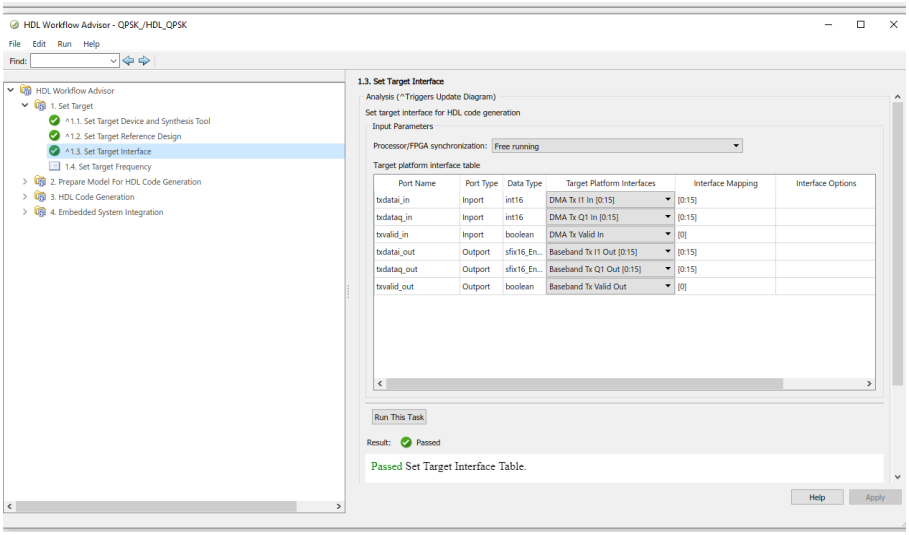
Εικόνα 4.1.2.2.4.5 Βήμα 1.1.

- Στο βήμα 1.2, επιλέγετε η επιλογή Εκπομπή (Transmitter).



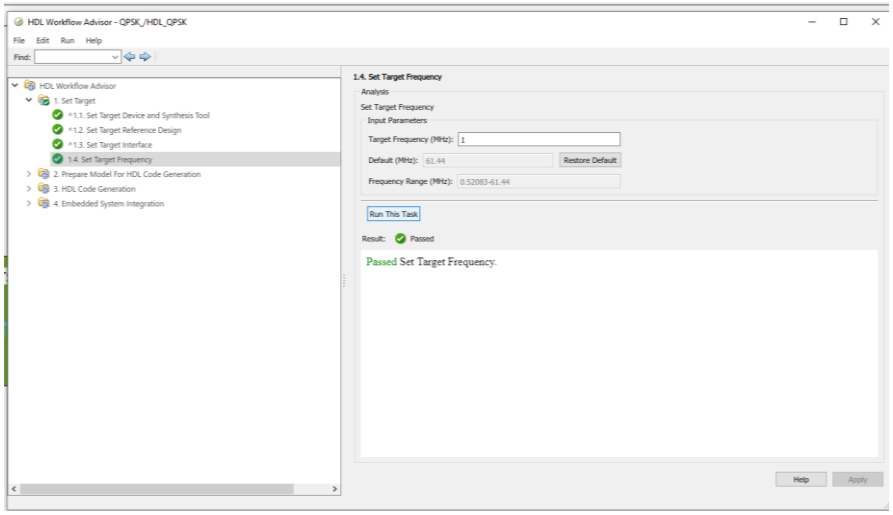
Εικόνα 4.1.2.2.4.6 Βήμα 1.2.

- Στο βήμα 1.3, δημιουργείται ο πίνακας διασύνδεσης, που θα χρησιμοποιηθεί για τη χαρτογράφηση των λογικών σημάτων του χρήστη με τα αντίστοιχα σήματα διεπαφής στο εξεταζόμενο μοντέλο.



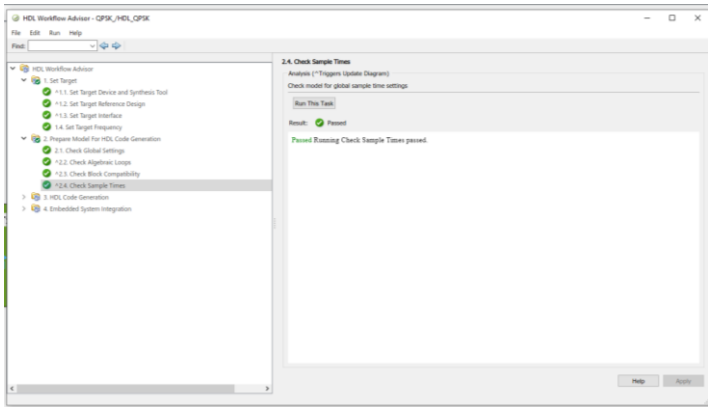
Εικόνα 4.1.2.2.4.7 Βήμα 1.3.

- Στο βήμα 1.4, ρυθμίζεται ότι η συχνότητα σύνθεσης του εξεταζόμενου μοντέλου (Device Under Test – DUT), ανάλογα με την συχνότητα δειγματοληψίας του συστήματος. Στο συγκεκριμένο μοντέλο, η συχνότητα δειγματοληψίας είναι λίγο υψηλότερη από 520ksps, οπότε μια συχνότητα σύνθεσης 1MHz είναι αρκετή.



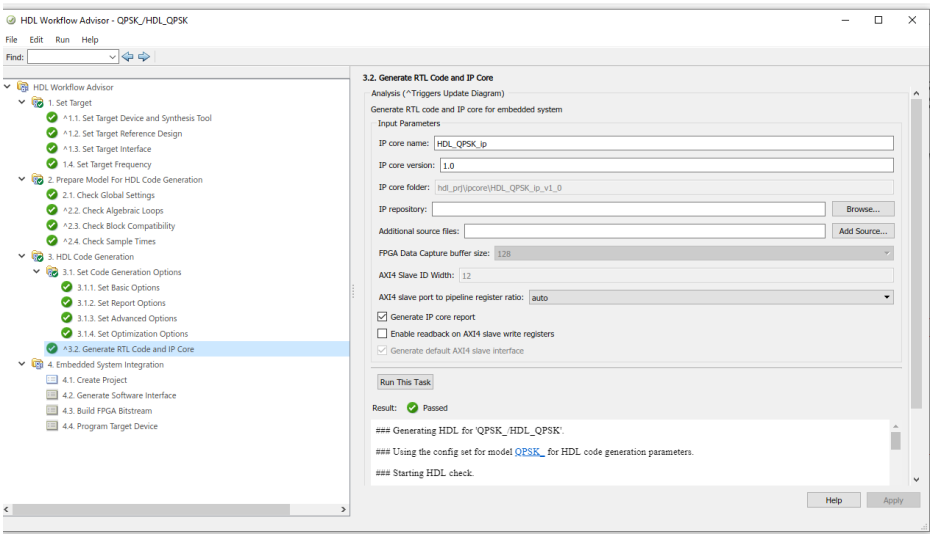
Εικόνα 4.1.2.2.4.8 Βήμα 1.4.

- Το βήμα 2 προετοιμάζει το μοντέλο για τη δημιουργία κώδικα HDL κάνοντας κάποιους ελέγχους σχεδιασμού.

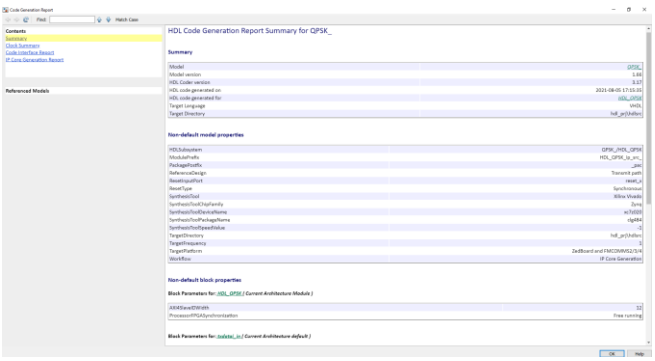


Εικόνα 4.1.2.2.4.9 Βήμα 2.

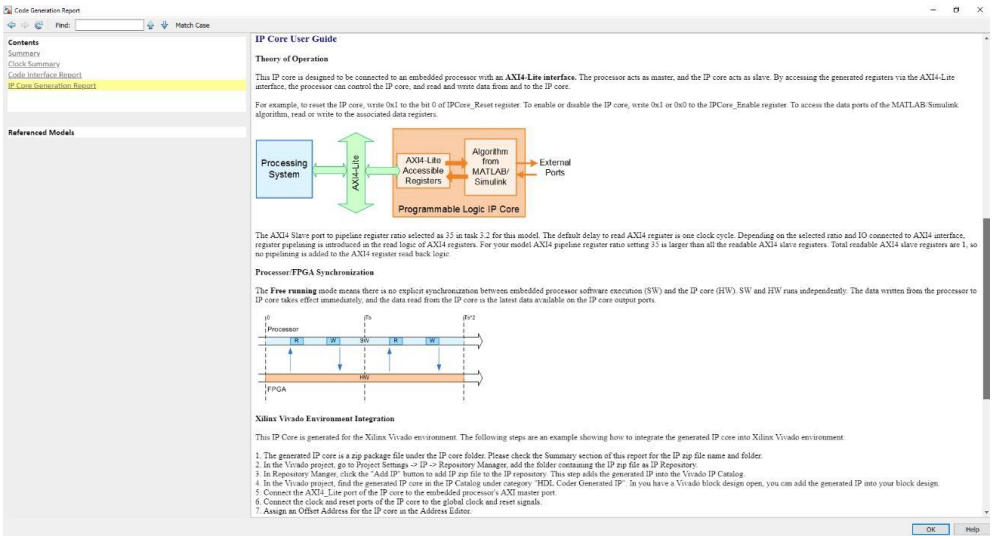
- Στο βήμα 3 πραγματοποιείται η δημιουργία του κώδικα HDL για το HDL IP Core και δημιουργείται η αντίστοιχη αναφορά.



Εικόνα 4.1.2.2.4.10 Βήμα 3.



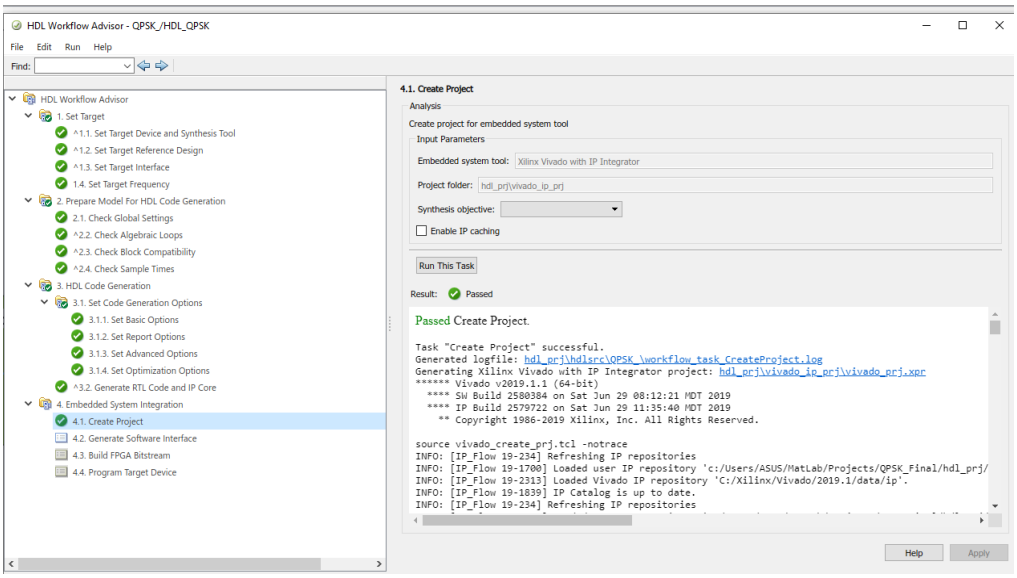
Εικόνα 4.1.2.2.4.11 Αναφορά Βήματος 3.



Εικόνα 4.1.2.2.4.12 Αναφορά Δημιουργίας Κώδικα Βήματος 3.

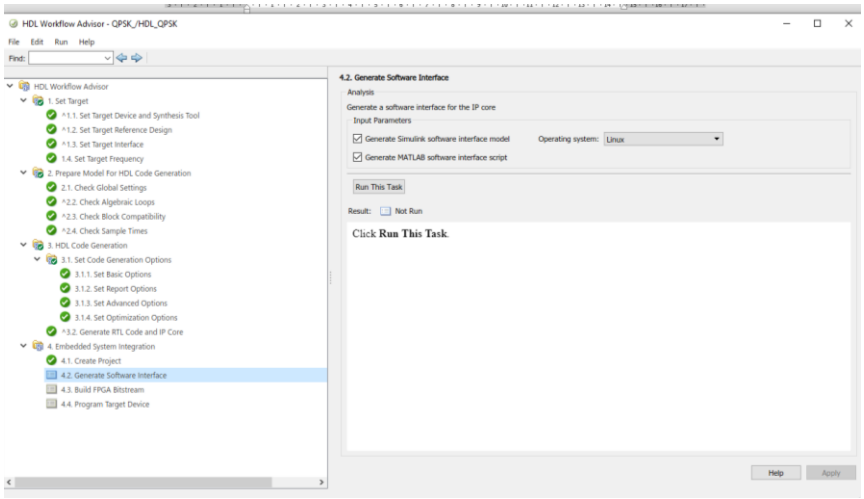
4.1.2.2.5 Δημιουργία Μοντέλου Διεπαφής και Βιβλιοθήκης Μπλοκ

- Στο βήμα 4 προχωράμε στην ενσωμάτωση του HDL IP Core, που δημιουργήθηκε στα προηγούμενα βήματα, στην αναπτυξιακή πλακέτα Zynq ZedBoard,καθώς και στην δημιουργία του bitstream που θα φορτωθεί και θα ενσωματωθεί σε αυτήν.

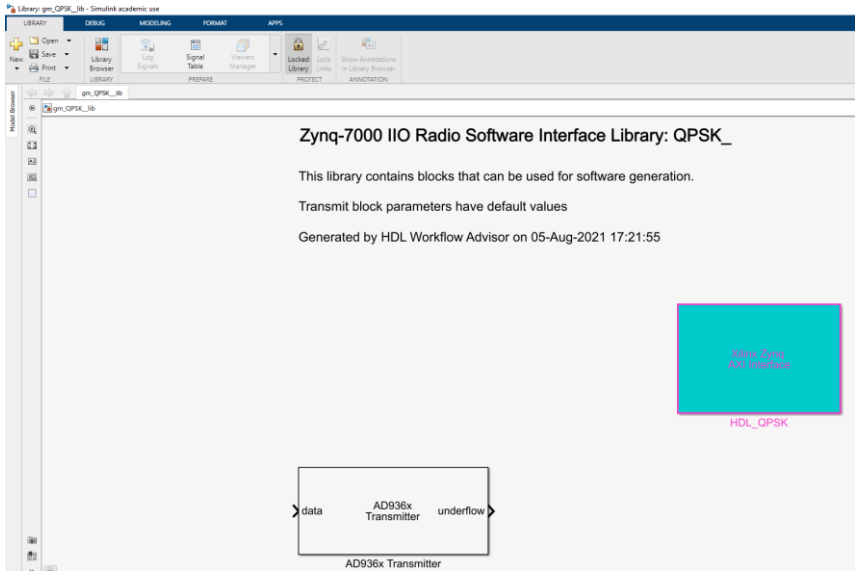


Εικόνα 4.1.2.2.5.1 Βήμα 4.1.

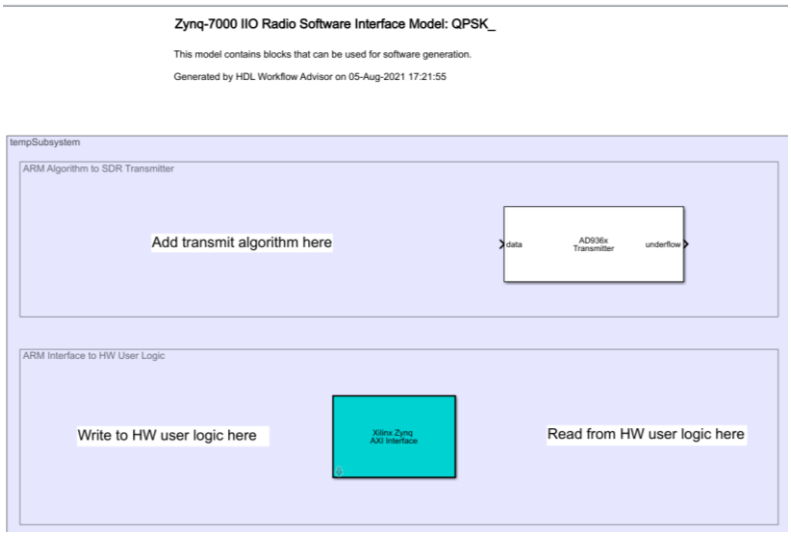
- Στο βήμα 4.2, η ροή εργασίας δημιουργεί ένα μοντέλο διεπαφής της αναπτυξιακής πλακέτας Zynq και μια βιβλιοθήκη μπλοκ.



Εικόνα 4.1.2.2.5.2 Βήμα 4.2.



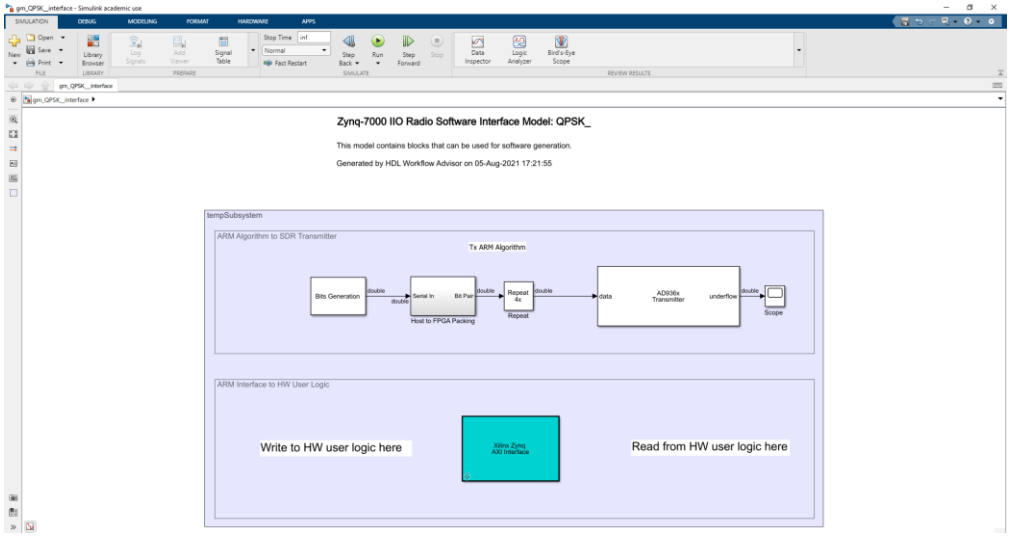
Εικόνα 4.1.2.2.5.3 Μοντέλο διεπαφής της αναπτυξιακής πλακέτας Zynq.



Εικόνα 4.1.2.2.5.4 Βιβλιοθήκη μπλοκ του αναπτυσσόμενο μοντέλου.

Η βιβλιοθήκη περιέχει το μπλοκ διεπαφής AXI που έχει δημιουργηθεί από το υποσύστημα HDL_QPSK. Το μπλοκ αυτό εμφανίζει μόνο τις θύρες ελέγχου AXI4-lite και όχι τις θύρες δεδομένων που υπάρχουν στο αρχικό μοντέλο του συστήματος στο Simulink. Οι θύρες δεδομένων υπάρχουν στο μπλοκ του πομπού που αντιπροσωπεύει τη διεπαφή δεδομένων μεταξύ του FPGA και του ARM.
Χρησιμοποιώντας το παραγόμενο μοντέλο της διεπαφής, προχωράμε στην δημιουργία του μοντέλου του πομπού QPSK που επιθυμούμε να ενσωματώσουμε στο FPGA. Το μοντέλο αυτό μπορεί να χρησιμοποιηθεί σε τρία διαφορετικά καθεστώτα:

- Εξομοίωση εξωτερικής λειτουργίας – External Mode Simulation
- Επεξεργαστής σε λειτουργία - Processor-in-the-Loop
- Πλήρης ανάπτυξη – Full Deployment

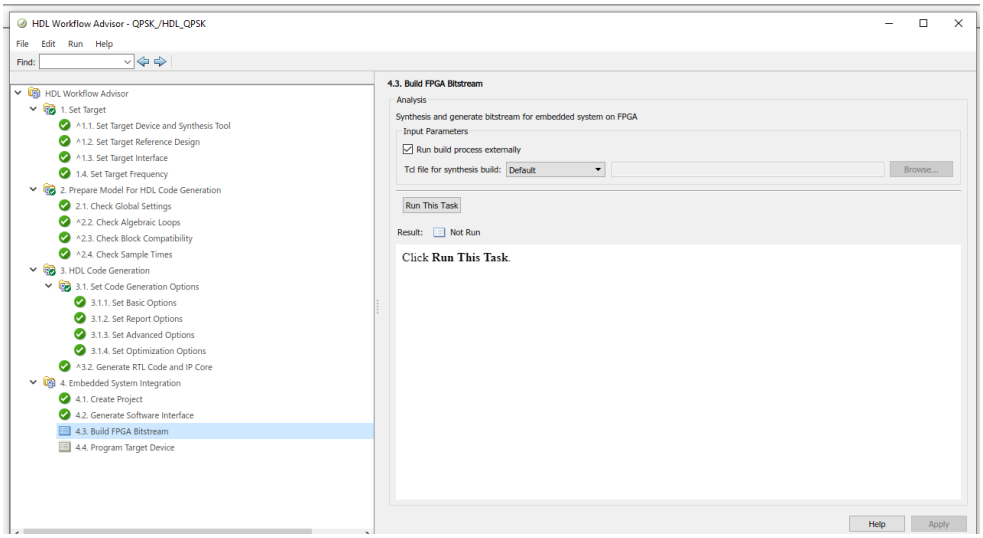


Εικόνα 4.1.2.2.5.5 Μοντέλο Διεπαφής QPSK πομπού.

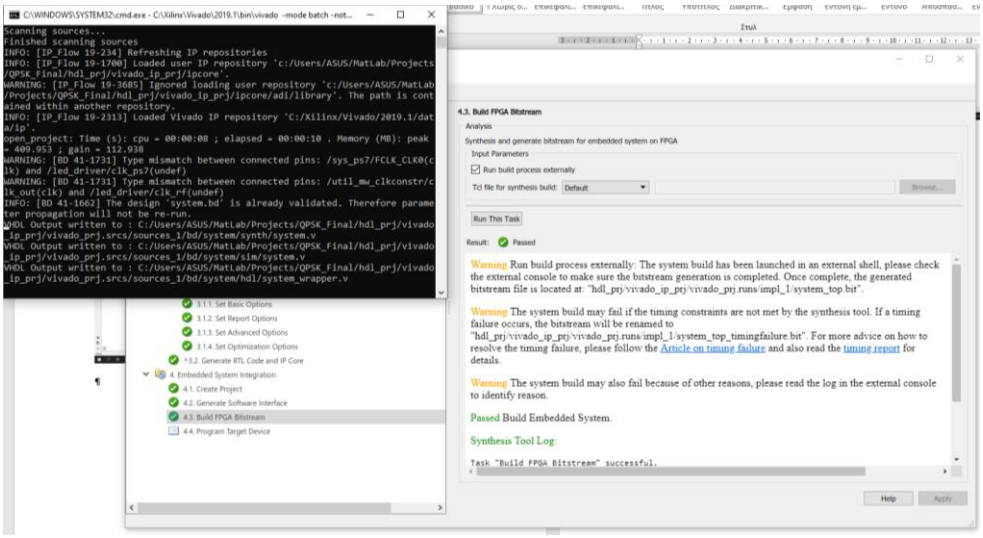
4.1.2.2.6 Δημιουργία Bitstream και ενσωμάτωση του στο FPGA

- Στο βήμα 4.3, ο σύμβουλος ροής εργασίας δημιουργεί ένα bitstream για το FPGA. Εκτελώντας αυτό το βήμα ανοίγει ένα εξωτερικό παράθυρο στο οποίο παρουσιάζονται τα διάφορα στάδια δημιουργίας bitstream, εφόσον έχει επιλέγει η αντίστοιχη επιλογή (Εκτέλεση διαδικασίας κατασκευής εξωτερικά - Run build process externally). Το βήμα αυτό θα ολοκληρωθεί σε λίγα λεπτά, αφού ολοκληρωθούν ορισμένοι βασικοί έλεγχοι

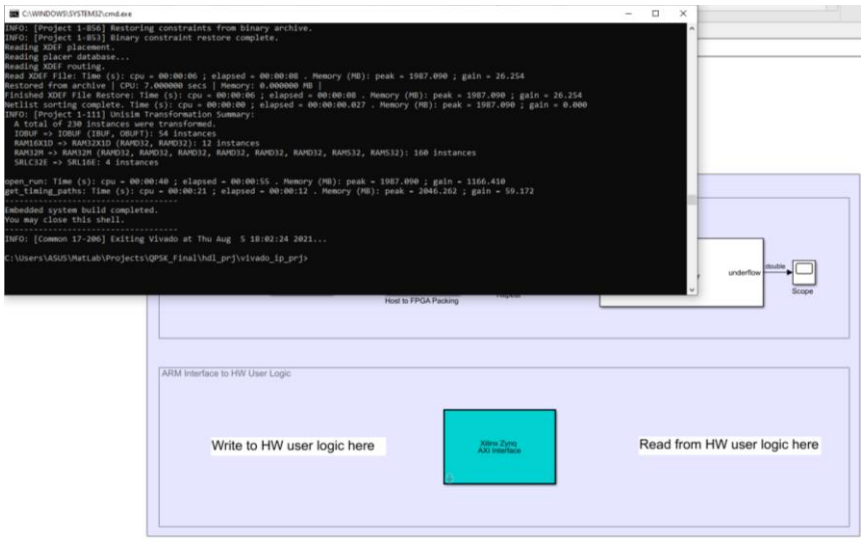
έργου και θα επισημανθεί με πράσινο σημάδι επιλογής. Ωστόσο, θα πρέπει να περιμένουμε έως ότου και το εξωτερικό παράθυρο, εμφανίσει μια αντίστοιχη επιτυχημένη ένδειξη δημιουργίας bitstream, πριν προχωρήσουμε στο επόμενο βήμα.



Εικόνα 4.1.2.2.6.1 Βήμα 4.3.



Εικόνα 4.1.2.2.6.2 Εξωτερικό Παράθυρο δημιουργίας Bitstream.



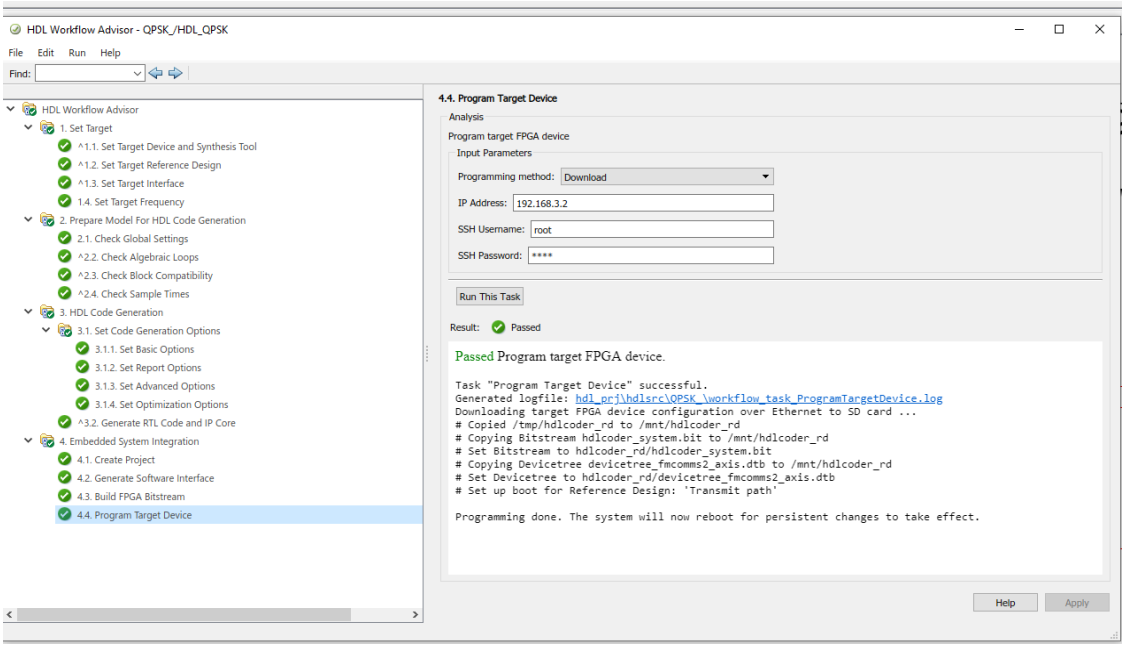
Εικόνα 4.1.2.2.6.3 Επιτυχημένη ολοκλήρωση δημιουργίας του Bitstream στο Εξωτερικό Παράθυρο.

- Στο βήμα 4.4 πραγματοποιεί η λήψη του bitstream από την συσκευή. Δίνονται δύο επιλογές για λήψη της ροής bitstream.
  - Με τη χρήση της επιλογής «Λήψη – Download", που είναι η προτεινόμενη από τον Advisor και αυτή που θα χρησιμοποιηθεί.
  - Με την χρήση του βύσματος JTAG και του αντίστοιχου καλωδίου.

  Πριν εκτελεστεί αυτό το βήμα, προτείνεται να εκτελεστεί στην επιφάνεια εργασίας του MatLab η συνάρτηση zynq με την ακόλουθη σύνταξη:

```
>> devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
```

προκειμένου να βεβαιωθούμε ότι το MatLab έχει ρυθμιστεί με τη σωστή IP διεύθυνση της αναπτυξιακής πλακέτας ZedBoard. Η προκαθορισμένη «default» διεύθυνση IP της αναπτυξιακής πλακέτας ZedBoard είναι η «192.168.3.2». Εάν έχει αλλαχτεί η διεύθυνση IP της πλακέτα, κατά τη διαδικασία ρύθμισης του υλικού, θα πρέπει να δοθεί αυτήν η διεύθυνση.



Εικόνα 4.1.2.2.6.4 Επιτυχημένη λήψη του Bitstream από την αναπτυξιακή πλακέτα ZedBoard.

Εναλλακτικά, το bitstream μπορεί να φορτωθεί εκτός του περιβάλλοντος εργασίας του Workflow Advisor. Σε αυτή την περίπτωση χρειάζεται να δημιουργηθεί ένα αντικείμενο SDR (SDR Object) στο περιβάλλον εργασίας του MatLab, καλώντας την συνάρτηση downloadImage στο αντικείμενο αυτό. Η διαδικασία αυτή , περιγράφεται στα ακόλουθα βήματα:

- Στο συγκεκριμένο μοντέλο επειδή χρησιμοποιείται η αναπτυξιακή πλακέτα ZedBoard της Xilinx και η RF αναπτυξιακή πλακέτα AD-FMCOMMS2/3/4/5 FMC, η εντολή δημιουργίας του SDR αντικειμένου συντάσσεται ακολούθως:

```
>> radio = sdrdev('AD936x');
```

- Στην συνέχεια γίνεται λήψη του bitstream από την ZedBoard, χρησιμοποιώντας την ακόλουθη συνάρτηση:

```
>> downloadImage(radio,'FPGAImage',
...'hdl_prj\vivado_ip_prj\vivado_prj.runs\impl_1\system_top.bit')
    % Η διαδρομή του αρχείου στο οποίο δημιουργήθηκε το bitstream
```

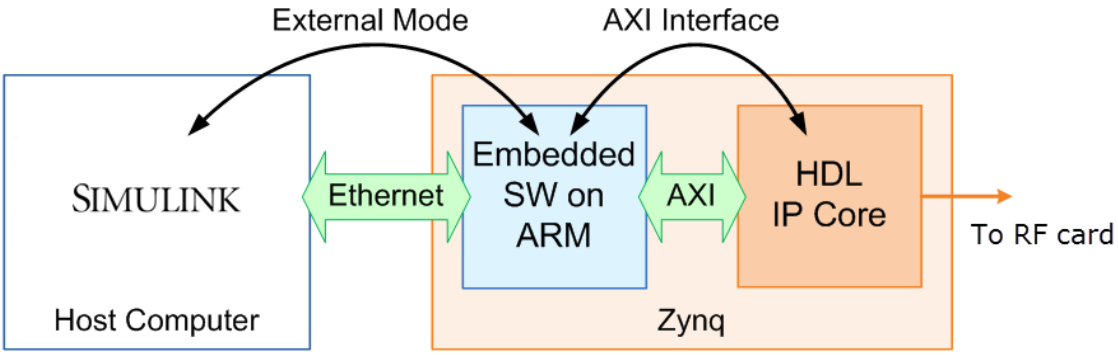
```
Command Window
>> devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
>> radio = sdrdev('AD936x');
>> downloadImage(radio,'FPGAImage','C:\Users\pater\MATLAB\Projects\QPSK_Final\hdl_prj\vivado_ip_prj\vivado_
## Loading bitstream file.
## Rebooting board.
## Reboot complete.
>>
```

Εικόνα 4.1.2.2.6.5 Επιτυχημένη λήψη του Bitstream από την αναπτυξιακή πλακέτα ZedBoard χωρίς την χρήση του Workflow Advisor.

4.1.2.2.7 Επεξεργαστής σε λειτουργία - Processor-in-the-Loop

Σε αυτό το καθεστώς, πραγματοποιείται η εκτέλεση του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR, στον επεξεργαστή ARM, χρησιμοποιώντας το καθεστώς εξωτερικής λειτουργίας (External Mode), που παρέχεται από το Simulink. Το καθεστώς External Mode χρησιμοποιείται για την δόκιμή, ρύθμιση και βελτιστοποίηση (Tunning) του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR.

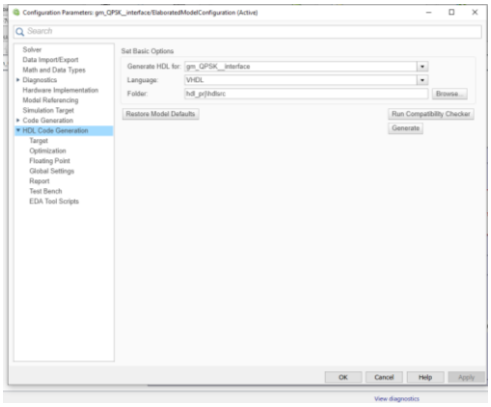
Αρχικά αναπτύσσεται/φορτώνεται ο αλγόριθμος του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR, στον επεξεργαστή ARM στην αναπτυξιακή πλακέτα Zynq ZedBoard, όπως προγράφτηκε στο προηγούμενο βήμα. Στη συνέχεια, συνδέετε ο αλγόριθμος με το μοντέλο του Simulink που δημιουργήθηκε σε προηγούμενα βήματα (Interface) στον κεντρικό υπολογιστή μέσω της σύνδεσης Ethernet. Ο κύριος σκοπός του μοντέλου Simulink είναι να συντονίζει και να παρακολουθεί, πώς ο αλγόριθμος αυτός λειτουργεί και συμπεριφέρεται στο υλικό, στην προκειμένη περίπτωση στην αναπτυξιακή πλακέτα ZedBoard. Αυτό είναι εφικτό καθώς ο επεξεργαστής ARM είναι συνδεδεμένος στον πυρήνα HDL IP μέσω της διεπαφής AXI.



Εικόνα 4.1.2.2.7.1 Αλγόριθμος Καθεστώτος Εξωτερικής Λειτουργίας Επεξεργαστή ARM [109].

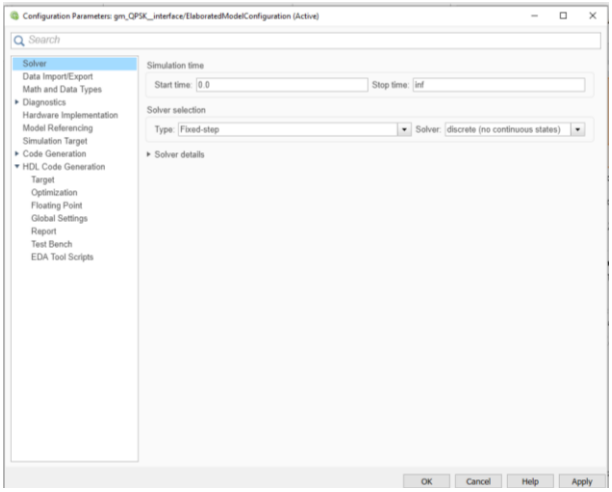
Για να εκτελεστεί ο αλγόριθμος στο καθεστώς εξωτερικής λειτουργίας θα πρέπει να ακολουθηθούν τα παρακάτω βήματα:

- Κάνοντας δεξί κλικ στο δημιουργημένο μοντέλο (Interface), επιλέγεται το «Model Configuration Parameters».



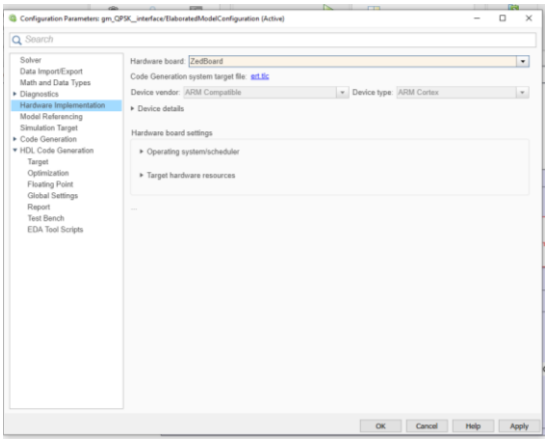
Εικόνα 4.1.2.2.7.2 Model Configuration Parameters.

- Στο παράθυρο «Commonly Used Parameters», επιλέγετε «Solver», ορίζεται η χρονική διάρκεια εκτέλεσης του αλγόριθμου «Stop Time» σε «inf» και κάνουμε κλικ στο OK.



Εικόνα 4.1.2.2.7.3 Commonly Used Parameters.

- Στο παράθυρο «Hardware Implementations», επιλέγουμε «Hardware board», ορίζουμε την αναπτυξιακή πλακέτα σε «ZedBoard» και κάνουμε κλικ στο «Apply».

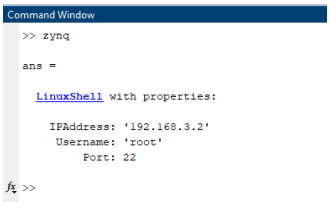


Εικόνα 4.1.2.2.7.4 Hardware Implementations.

- Πριν από την εκτέλεση του μοντέλου, γίνεται κλήση της συνάρτηση «zynq», με την ακόλουθη σύνταξη:

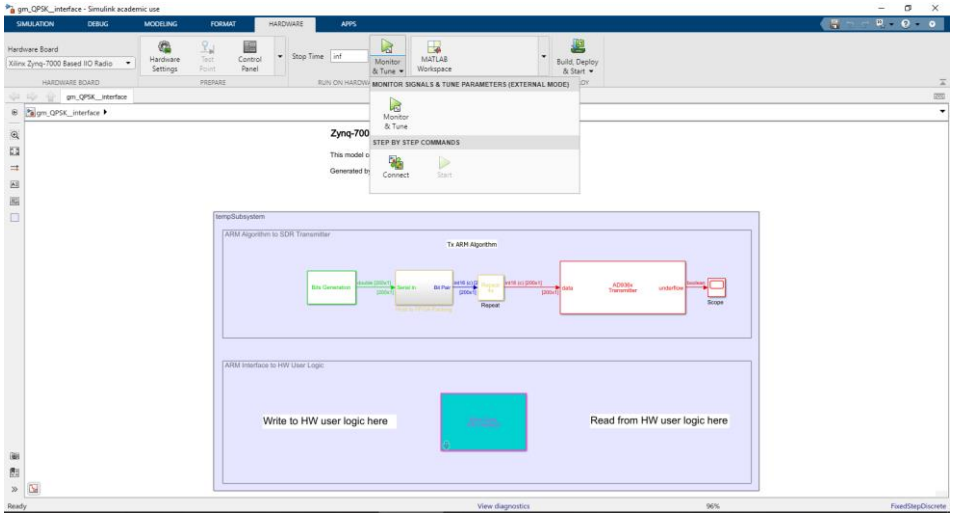
```
devzynq = zynq('linux','192.168.3.2','root','root','/tmp');
```

προκειμένου να βεβαιωθούμε ότι το MatLab έχει ρυθμιστεί με τη σωστή φυσική διεύθυνση IP της αναπτυξιακής πλακέτας ZedBoard. Από προεπιλογή, η φυσική διεύθυνση IP της αναπτυξιακής πλακέτας ZedBoard είναι η «192.168.3.2». Εάν έχει αλλαχτεί η διεύθυνση IP κατά τη διαδικασία εγκατάστασης πλακέτας, πρέπει αντ' αυτού να οριστεί η νέα αυτή διεύθυνση.

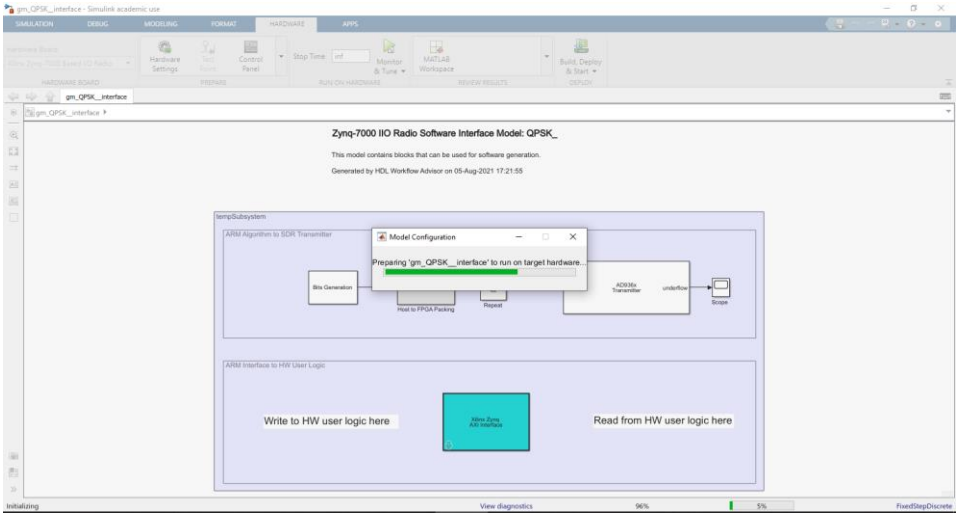


Εικόνα 4.1.2.2.7.5 Επαλήθευση φυσικής διεύθυνσης IP της αναπτυξιακής πλακέτας ZedBoard.

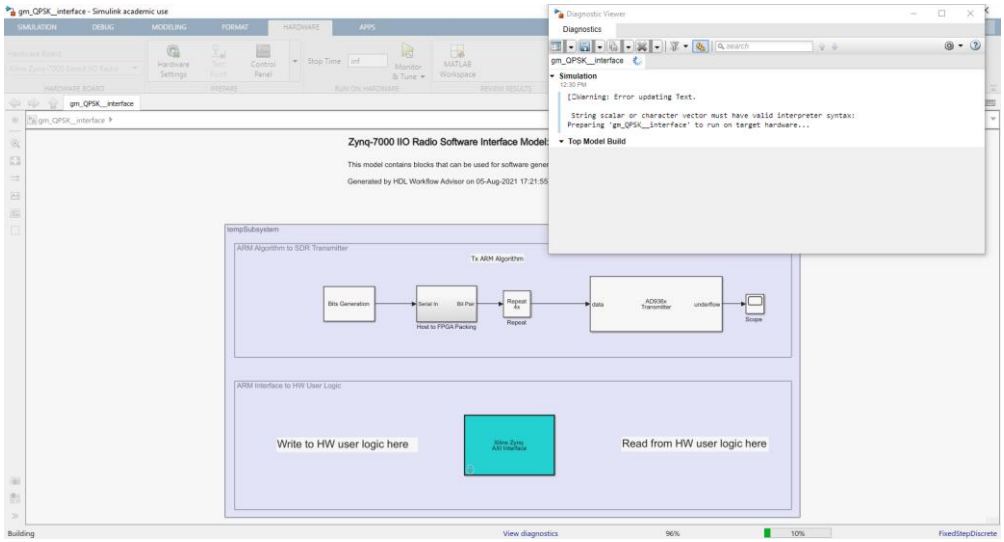
- Για να εκτελεστεί το μοντέλο, στην καρτέλα «Hardware», κάνουμε κλικ στην επιλογή «Monitor & Tune». Ο ενσωματωμένος κωδικοποιητής κατασκευάζει το μοντέλο, και κατεβάζει το εκτελέσιμο ARM στην αναπτυξιακής πλακέτας ZedBoard, το εκτελεί και συνδέει το μοντέλο με το εκτελέσιμο, που εκτελείται σε αυτήν.



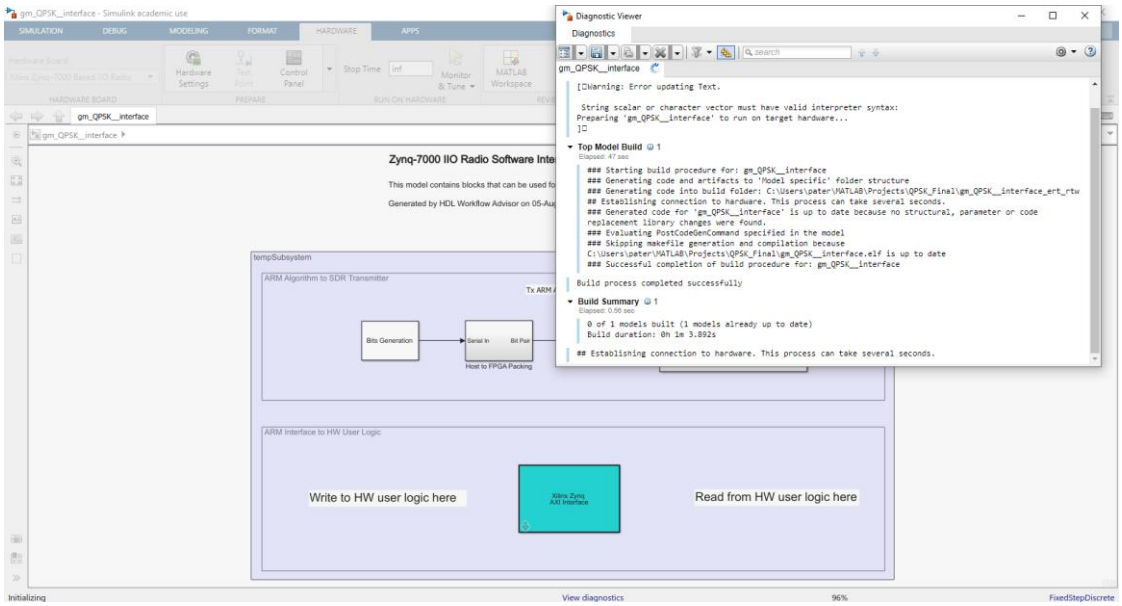
Εικόνα 4.1.2.2.7.6 Monitor & Tune.



Εικόνα 4.1.2.2.7.7 Έναρξη διαδικασίας Monitor & Tune.

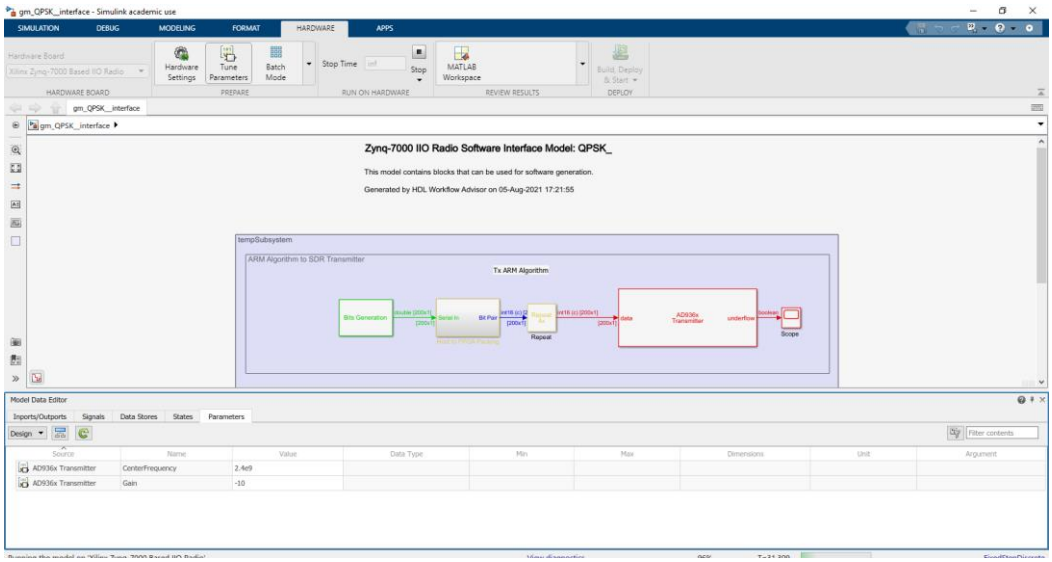


Εικόνα 4.1.2.2.7.8 Έναρξη κατασκευής μοντέλου από τον ενσωματωμένο κωδικοποιητή.



Εικόνα 4.1.2.2.7.9 Έναρξη διαδικασίας εγκατάστασης του εκτελέσιμου αρχείο του ARM στην αναπτυξιακή πλακέτα ZedBoard, σύνδεση του με το μοντέλο και εκτέλεση του από αυτή.

- Αλλάζοντας τις ρυθμιζόμενες παραμέτρους στο μοντέλο του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR, επιβεβαιώνεται ότι η συμπεριφορά του ανταποκρίνεται στις δικές μας προδιαγραφές και απαιτήσεις. Στο συγκεκριμένο μοντέλο οι παράμετροί που μπορούν να ρυθμιστούν είναι :
    - Η κεντρική συχνότητα εκπομπής (AD936x Transmitter Central Frequency) της RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMCH. Το εύρος συχνοτήτων είναι από 70 MHz έως 6 GHz.
    - Ο συντελεστής ενίσχυσης (AD936x Transmitter Gain) της RF αναπτυξιακή πλακέτα AD-FMCOMMS4 FMC. Το κέρδους, καθορίζεται από ένα βαθμωτό μέγεθος. Το εύρος κέρδους είναι -89,75 dB έως 0 dB, με βήμα των 0,25 dB.

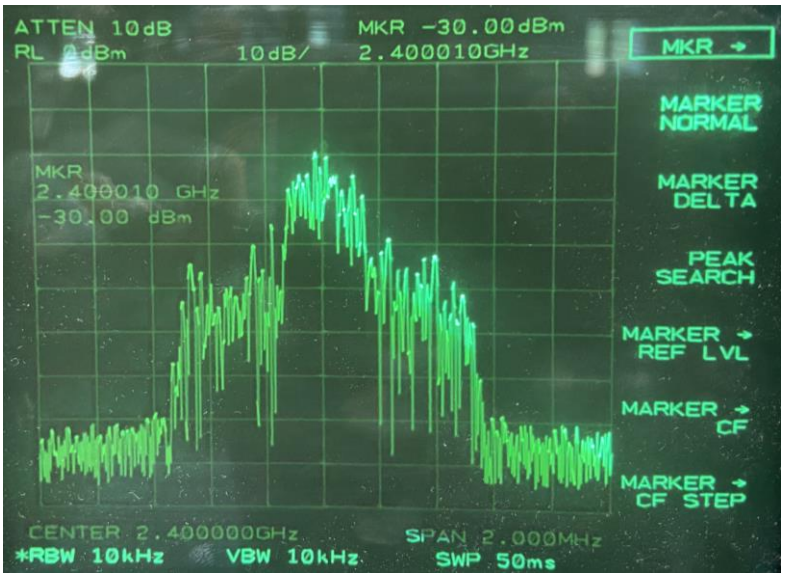


Εικόνα 4.1.2.2.7.10 Οι ρυθμιζόμενες παράμετροι του μοντέλου.

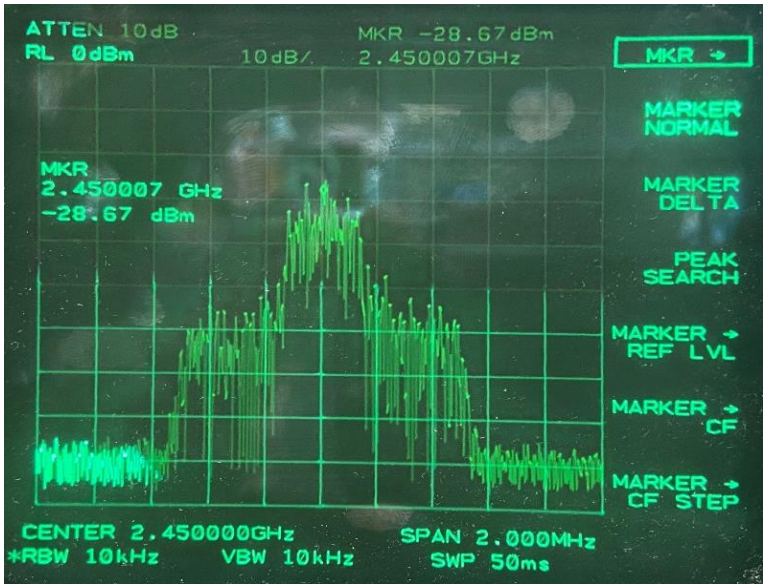
Οι προκαθορισμένες τιμές στο συγκεκριμένο μοντέλο είναι:

- AD936x Transmitter Central Frequency: 2.4GHz.
- AD936x Transmitter Gain: -10.

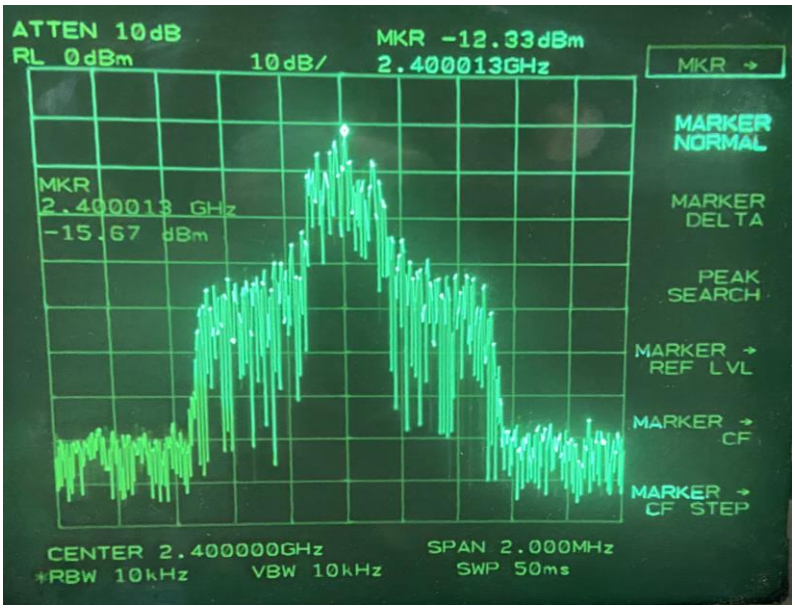
Αλλάζοντας τις ρυθμιζόμενες παραμέτρους του μοντέλου, μπορούμε να δούμε αντίστοιχες μεταβολές της εκπεμπόμενης κυματομορφής του μοντέλου στον αναλυτή φάσματος.



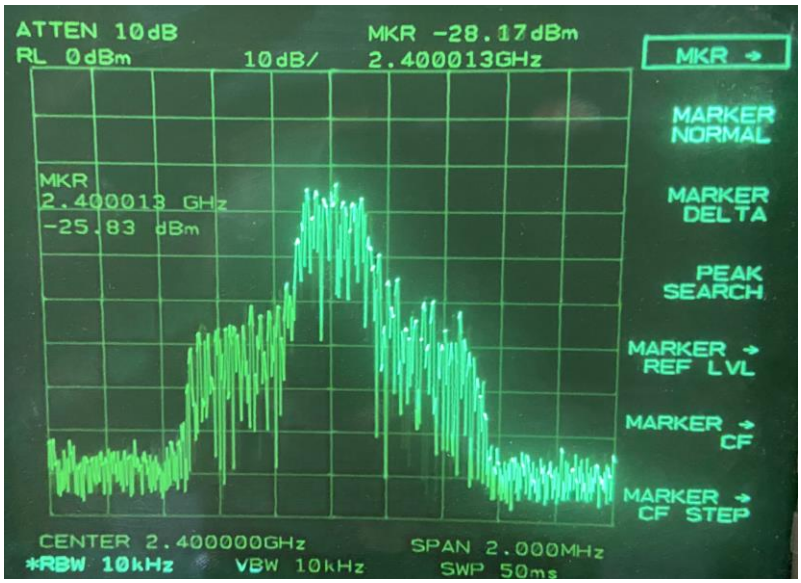
Εικόνα 4.1.2.2.7.11 Κεντρική συχνότητα εκπομπής 2,4 GHz.



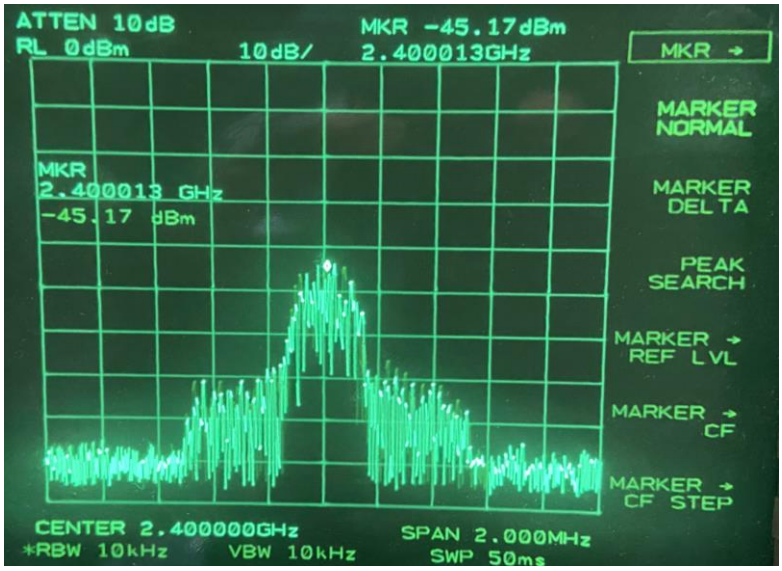
Εικόνα 4.1.2.2.7.12 Κεντρική συχνότητα εκπομπής 2,45 GHz.



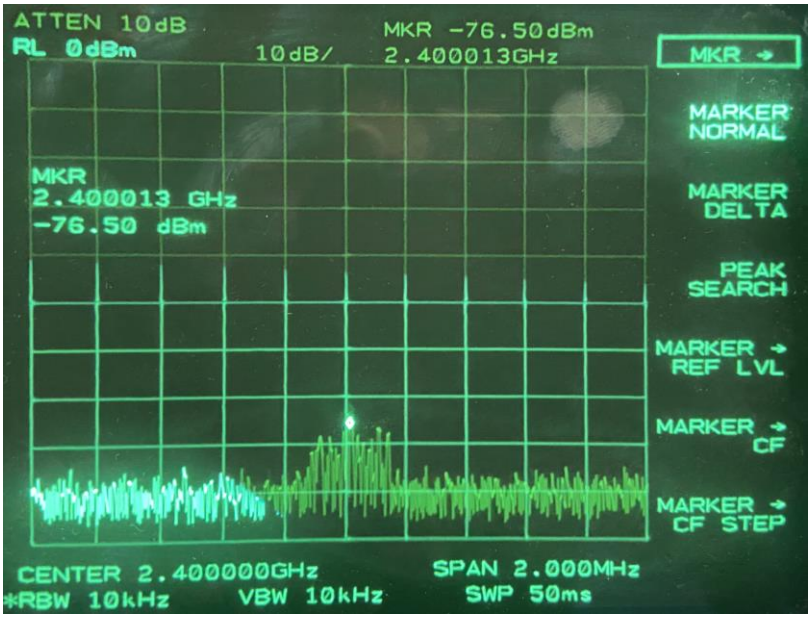
Εικόνα 4.1.2.2.7.13 Συντελεστής ενίσχυσης 0 (-15,67 dBm) .



Εικόνα 4.1.2.2.7.14 Συντελεστής ενίσχυσης -10 (-25,83 dBm).

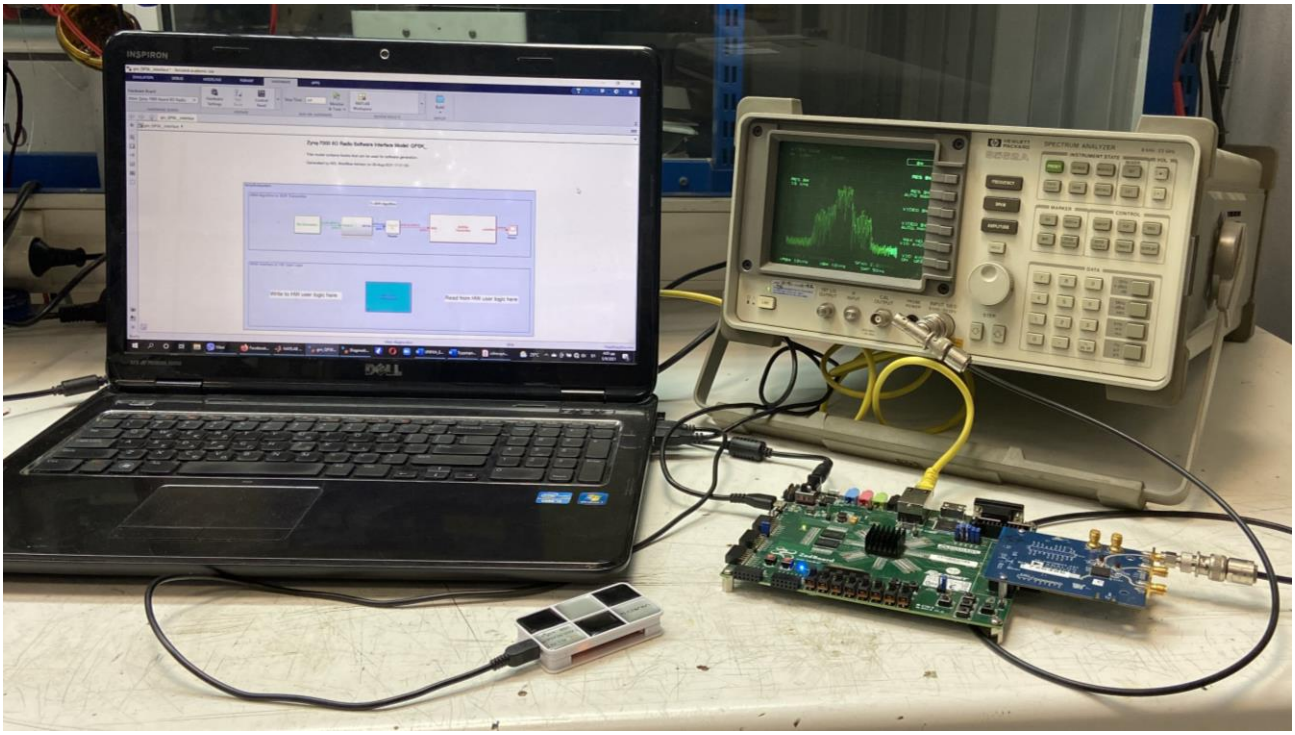


Εικόνα 4.1.2.2.7.15 Συντελεστής ενίσχυσης -30 (-45,17 dBm).



Εικόνα 4.1.2.2.7.16 Συντελεστής ενίσχυσης -60 (-76,50 dBm).

- Μετά τις αναγκαίες αλλαγές στις ρυθμιζόμενες παραμέτρους, στο μοντέλο του αναπτυσσόμενου τηλεπικοινωνιακού μοντέλου SDR, και εφόσον έχει επιβεβαιωθεί ότι η συμπεριφορά του ανταποκρίνεται στις δικές μας προδιαγραφές και απαιτήσεις, κάνοντας κλικ στο κουμπί «**Stop**» στο μοντέλο, το παράθυρο εντολών του συστήματος κλείνει.



Εικόνα 4.1.2.2.7.17 Διάταξη υλικού QPSK Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς «Επεξεργαστής σε λειτουργία - Processor-in-the-Loop».

4.1.2.3 Υλοποίηση Πλήρης Αυτόνομη Ανάπτυξη- Stand Alone

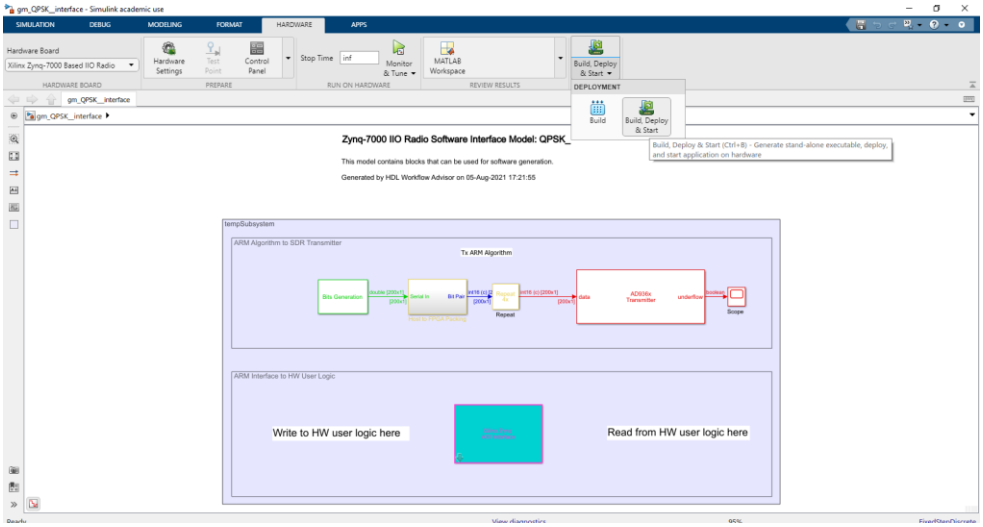
Στο καθεστώς Πλήρης Αυτόνομης Ανάπτυξης - Stand Alone, προχωράμε στην ανάπτυξη ενός πλήρως λειτουργικού τηλεπικοινωνιακού συστήματος SDR, το οποίο θα εκτελείται στην αναπτυξιακή πλακέτα ZedBoard ακόμα και αφού έχει αποσυνδεθεί από το Simulink. Σε αυτήν τη λειτουργία ενώ το μοντέλο βρίσκεται σε λειτουργία, δεν μπορεί να αλληλοεπιδράσει με το Simulink, προκειμένου να προβούμε σε ρύθμιση των παραμέτρων λειτουργίας του.

4.1.2.3.1 Δημιουργία εκτελέσιμου αρχείου στο καθεστώς Πλήρης Αυτόνομης Ανάπτυξης – Stand Alone

Έχοντας ολοκληρώσει επιτυχώς τα προηγούμενα βήματα στο καθεστώς «Επεξεργαστής σε λειτουργία - Processor-in-the-Loop» και έχοντας ρυθμίσει τις παραμέτρους του τηλεπικοινωνιακού συστήματος SDR, έτσι ώστε να επιβεβαιωθεί ότι η συμπεριφορά του, ανταποκρίνεται στις δικές μας προδιαγραφές και απαιτήσεις, προχωράμε στην δημιουργία ενός αυτόνομου εκτελέσιμου αρχείου, το οποίο θα φορτωθεί στην αναπτυξιακή πλακέτα ZedBoard και θα εκτελείται κάθε φορά που είναι ενεργοποιημένη.

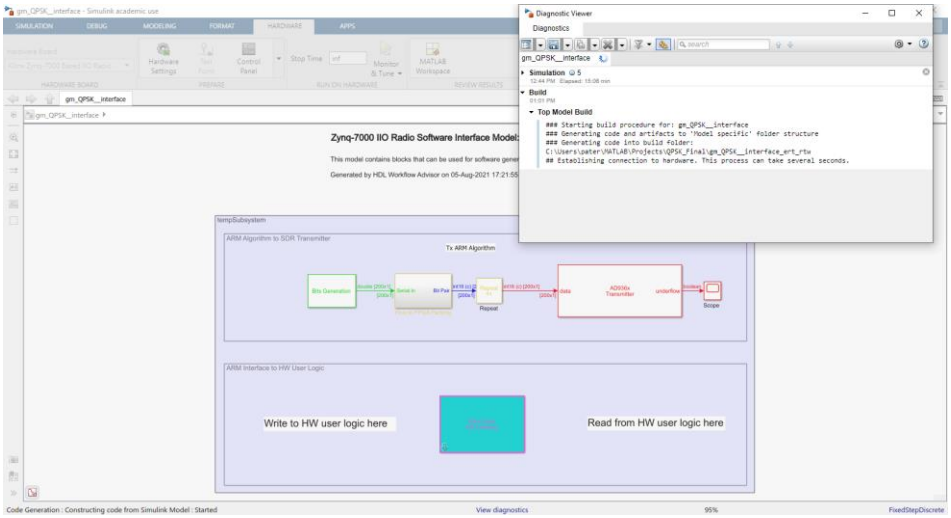
Αυτή η διαδικασία περιγράφεται, στα βήματα που ακολουθούν:

- Με ενεργοποιημένη την αναπτυξιακή πλακέτα ZedBoard, επιλέγουμε στην καρτέλα «Hardware», στην επιλογή «Deploy», επιλέγουμε «Build Stand-Alone».

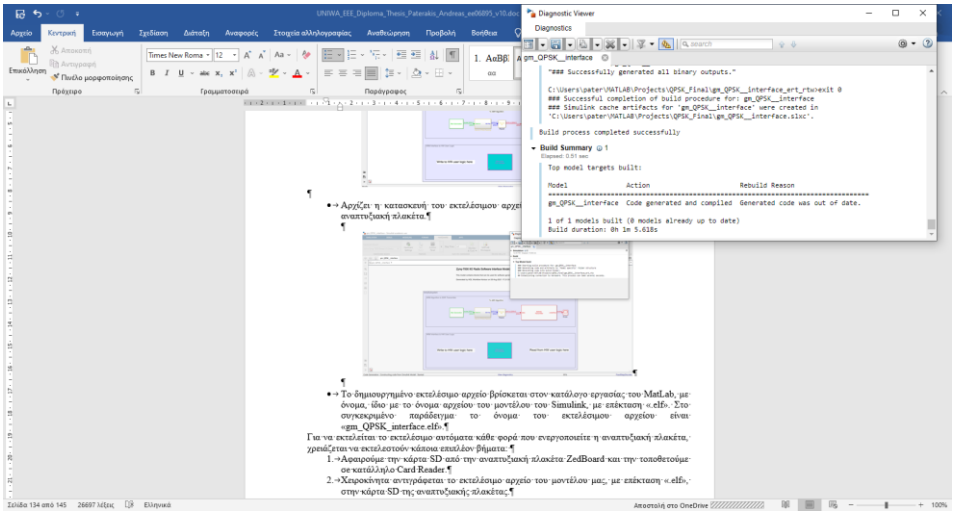


Εικόνα 4.1.2.3.1.1 Επιλογή διαδιακασίας «Build Stand-Alone».

- Αρχίζει η κατασκευή του εκτελέσιμου αρχείου που θα φορτωθεί και θα τρέξει στην αναπτυξιακή πλακέτα. Η παραπάνω διαδικασία δημιουργίας του εκτελέσιμου αρχείου παρακολουθείται, ανοίγοντας το παράθυρο «View Diagnostics» στο κάτω μέρος της οθόνης του Interface του μοντέλου μας.

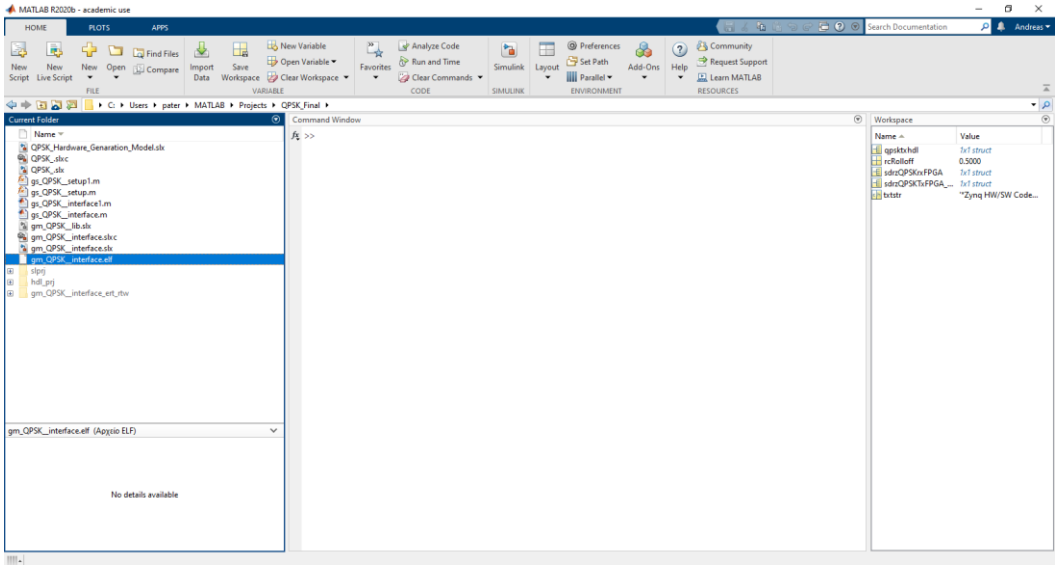


Εικόνα 4.1.2.3.1.2 Έναρξη διαδιακασίας «Build Stand-Alone».



Εικόνα 4.1.2.3.1.3 Επιτυχής ολοκλήρωσης διαδιακασίας «Build Stand-Alone».

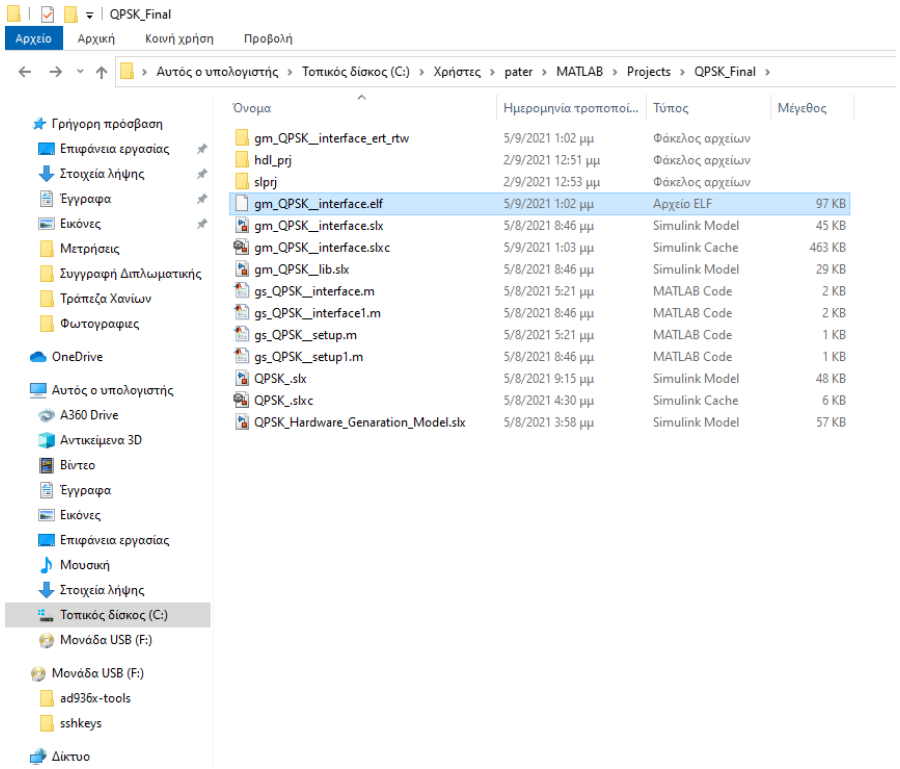
- Το δημιουργημένο εκτελέσιμο αρχείο βρίσκεται στον κατάλογο εργασίας του MatLab, με όνομα, ίδιο με το όνομα αρχείου του μοντέλου του Simulink, με επέκταση «.elf». Στο συγκεκριμένο παράδειγμα το όνομα του εκτελέσιμου αρχείου είναι «gm_QPSK_interface.elf».



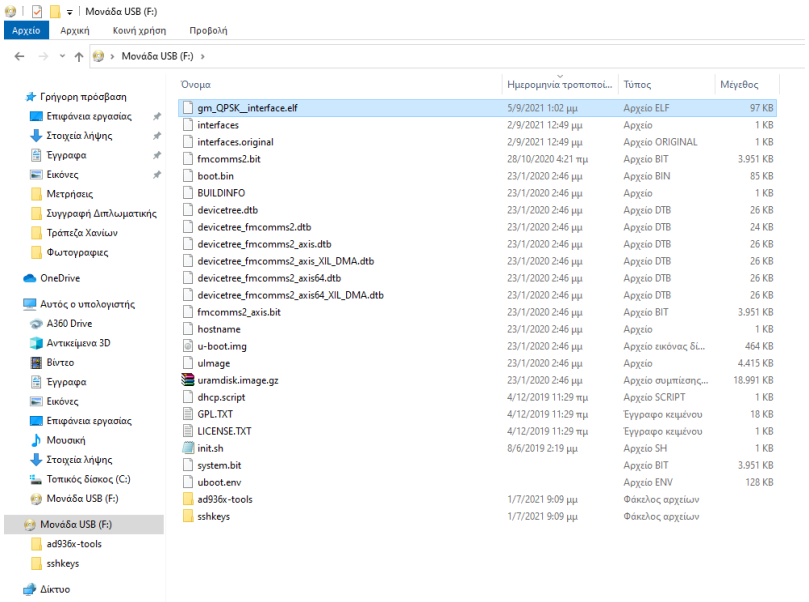
Εικόνα 4.1.2.3.1.4 Δημιουργία εκτελέσιμου αρχείου «gm_QPSK_interface.elf» του μοντέλου.

Για να εκτελείται το εκτελέσιμο αυτόματα κάθε φορά που ενεργοποιείτε η αναπτυξιακή πλακέτα ZedBoard, χρειάζεται να εκτελεστούν κάποια επιπλέον βήματα:

1. Απενεργοποιούμε την αναπτυξιακή πλακέτα ZedBoard, αφαιρούμε την κάρτα SD και την τοποθετούμε σε κατάλληλο Card Reader.
2. Χειροκίνητα αντιγράφεται το εκτελέσιμο αρχείο του μοντέλου μας, με επέκταση «.elf», στην κάρτα SD της αναπτυξιακής πλακέτας.



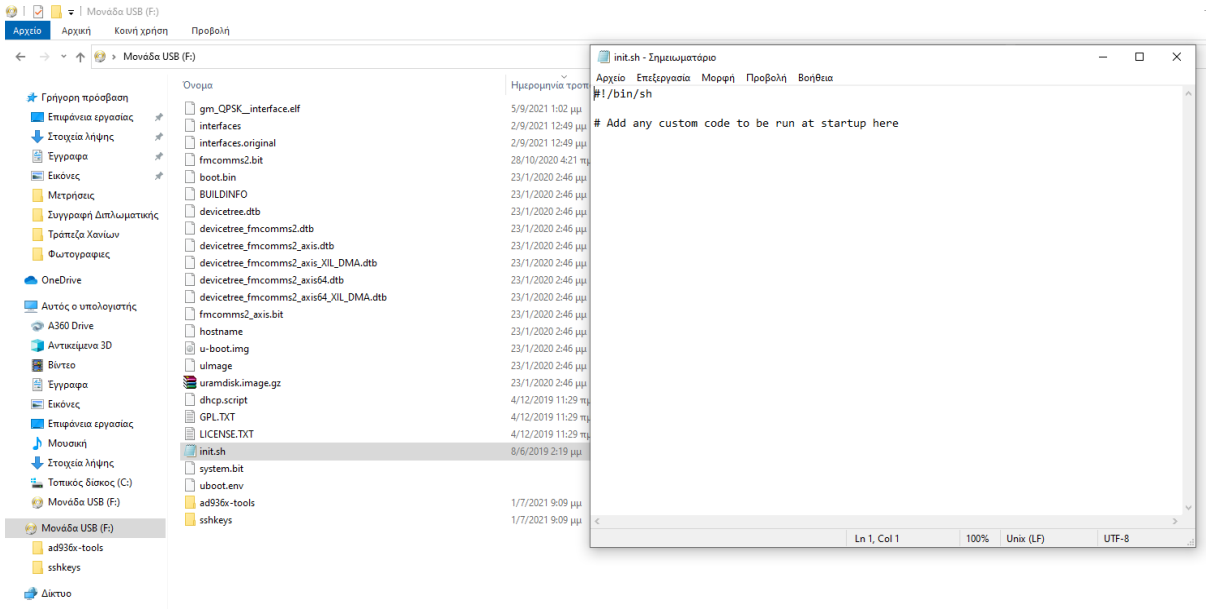
Εικόνα 4.1.2.3.1.5 Φάκελος στον οποίο αποθηκεύεται το εκτελέσιμο αρχείο «gm_QPSK_interface.elf» του μοντέλου.



Εικόνα 4.1.2.3.1.6 Αντιγραφή του εκτελέσιμου αρχείου «gm_QPSK_interface.elf» του μοντέλου στην κάρτα SD της αναπτυξιακής πλακέτας ZedBoard.

3. Με την χρήση του εργαλείου «Σημειωματάριο» των Windows, ανοίγουμε το δημιουργημένο όνομα αρχείου «init.sh» στην κάρτα SD. Στο αρχείο αυτό υπάρχει ο ακόλουθος κώδικας εντολών.

```
#!/bin/sh
# Add any custom code to be run at startup here
```



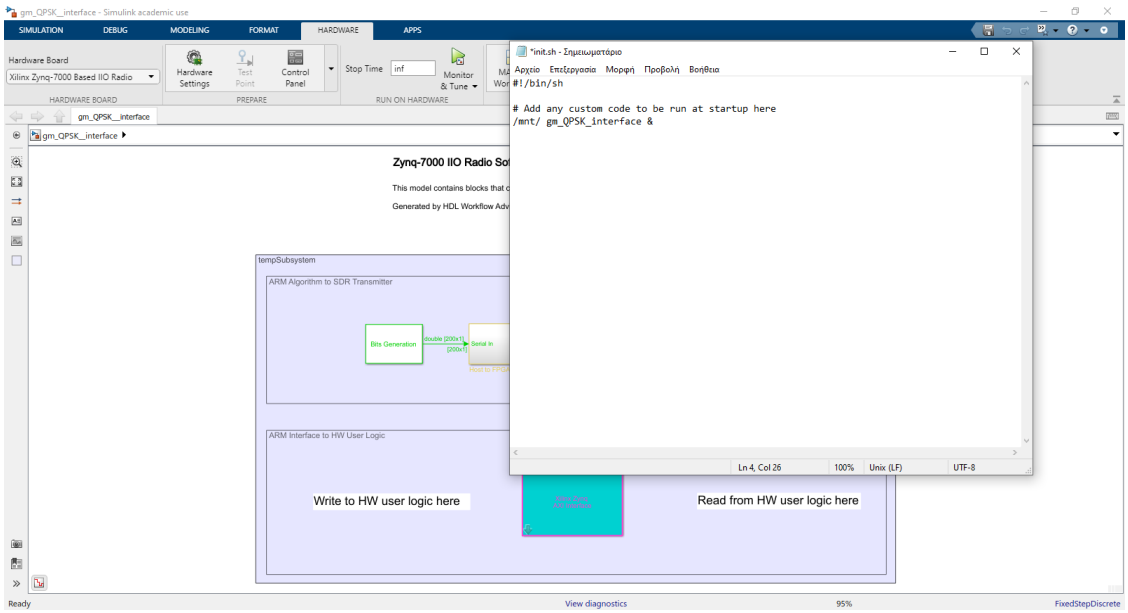
Εικόνα 4.1.2.3.1.7 Άνοιγμα του αρχείου «init.sh» στην κάρτα SD, με την χρήση του του εργαλείου «Σημειωματάριο» των Windows .

4. Για να μπορέσει να εκτελεστεί το εκτελέσιμο αρχείο «gm_QPSK_interface.elf», κάθε φορά που ενεργοποιούμε την αναπτυξιακή πλακέτα ZedBoard, θα πρέπει ο κώδικας εντολών που περιέχεται στο αρχείο «init.sh», να έχει την μορφή :

```
#!/bin/sh
# Add any custom code to be run at startup here
/mnt/mymodel &
```

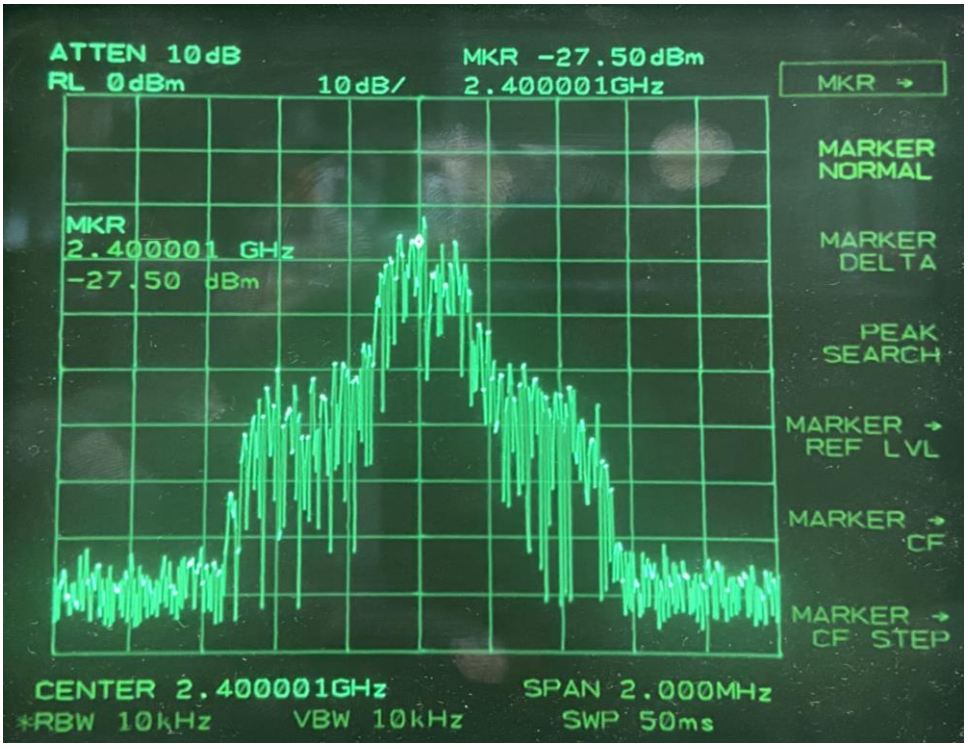
5. Ενημερώνουμε το αρχείο «init.sh», αντικαθιστώντας όπου «mymodel», το όνομα του εκτελέσιμου αρχείου του μοντέλου μας και στην συνέχεια το αποθηκεύουμε.

```
#!/bin/sh
# Add any custom code to be run at startup here
/mnt/gm_QPSK_interface.elf &
```

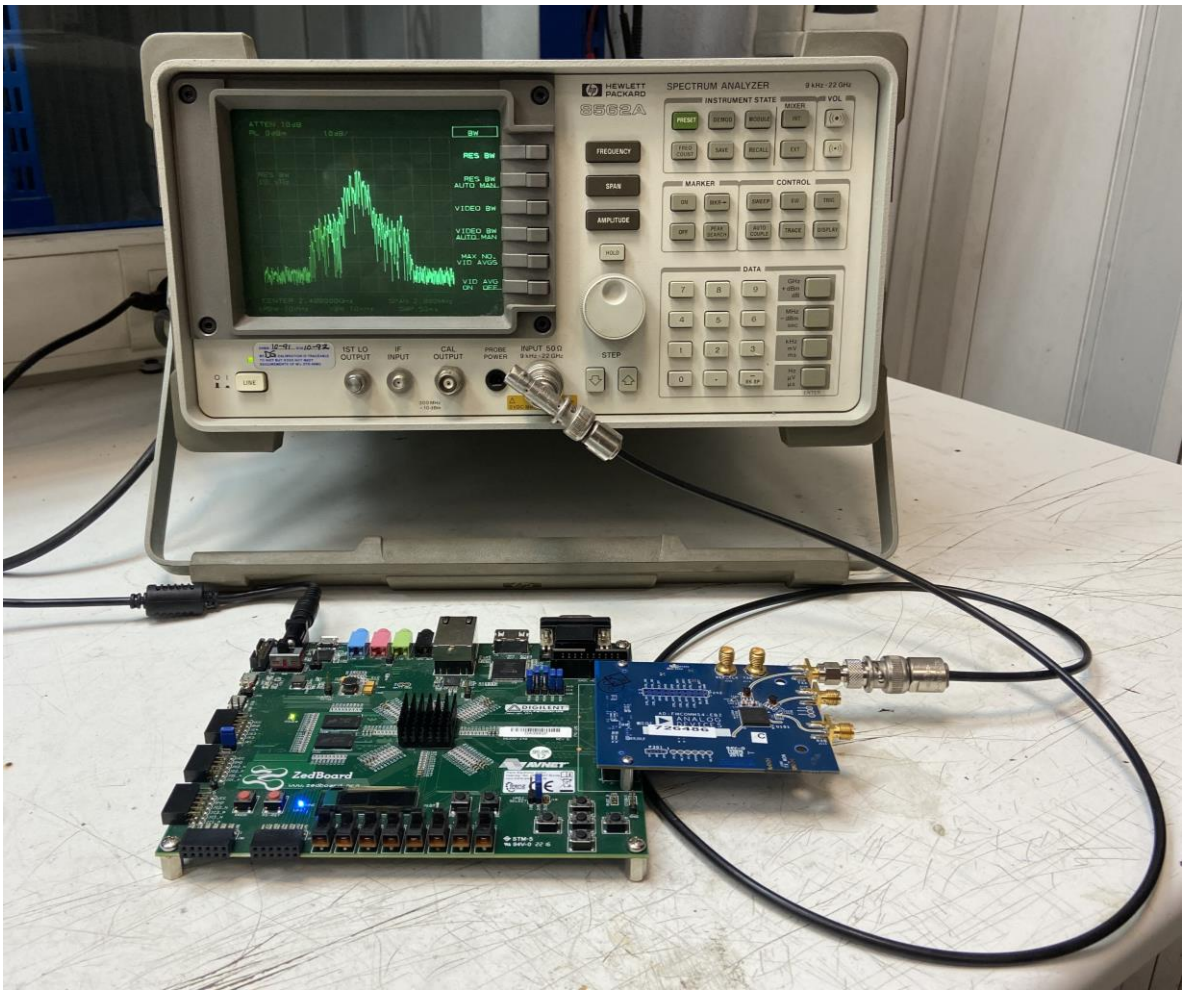


Εικόνα 4.1.2.3.1.8 Επεξεργασία του αρχείου «init.sh» στην κάρτα SD, με την χρήση του του εργαλείου «Σημειωματάριο» των Windows .

6. Τοποθετούμε την κάρτα SD, στην αναπτυξιακή πλακέτα ZedBoard και την ενεργοποιούμε.
7. Στον αναλυτή φάσματος βλέπουμε την εκπομπή του Αναπτυσσόμενου Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς Πλήρης Αυτόνομης Ανάπτυξης.



Εικόνα 4.1.2.3.1.9 QPSK εκπομπή Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς Πλήρης Αυτόνομης Ανάπτυξης .



Εικόνα 4.1.2.3.1.10 Διάταξη υλικού QPSK Τηλεπικοινωνιακού Συστήματος SDR, σε καθεστώς «Πλήρης Αυτόνομης Ανάπτυξης» .

5 ΚΕΦΑΛΑΙΟ 5ο : Συμπεράσματα

Σε αυτή την Διπλωματική εργασία έγινε μία εισαγωγή στις Συστοιχίες Επιτόπια Προγραμματιζόμενων Πυλών - Field Programmable Gate Array (FPGA), μια σειρά προγραμματιζόμενων λογικών μπλοκ, ως πλατφόρμα για την εφαρμογή των Συστημάτων Ραδιοεπικοινωνίας Μέσω Λογισμικού - Software Defined Radio (SDR).

Τα συστήματα SDR αποτελούν τον τρόπο σχεδίασης και υλοποίησης Ψηφιακών Τηλεπικοινωνιακών Συστημάτων, με την χρήση κατάλληλου λογισμικού και υλικού, τεχνολογία η οποία παρουσιάζει πολλά πλεονεκτήματα καθώς και μεγάλη ευελιξία. Χρησιμοποιώντας συστήματα SDR που βασίζονται σε αναπτυξιακές πλακέτες FPGA, είναι δυνατή η υλοποίηση ενός ψηφιακού πομπού και δέκτη, εφαρμόζοντας οποιασδήποτε πρωτόκολλο επικοινωνίας, δημιουργώντας κατάλληλο αλγόριθμο. Ο αλγόριθμος αυτός μπορεί να επαναχρησιμοποιηθεί και να εκτελεστεί, σε οποιαδήποτε άλλη πλατφόρμα FPGA, μετά από γρήγορη αναδιαμόρφωση του και με κατάλληλες αλλαγές στο λογισμικό που εκτελείται στο FPGA. Με αυτόν τον τρόπο, ένα σύστημα SDR, μπορεί να ανταποκρίνεται στις εκάστοτε τεχνικές προδιαγραφές και απαιτήσεις τόσο του υλικού στο οποίο καλείται να εκτελεστεί όσο και στις ιδιαιτερότητες του ίδιου του πρωτόκολλου επικοινωνίας που καλείται να υλοποίηση.

Ξεκινώντας από την σχεδίαση και υλοποίηση, βασικών αναλογικών διαμορφώσεων (AM, SSB, κτλ.), στο καθεστώς «Μοντέλο Διεπαφής Λογισμικού – Streaming», έγιναν εμφανής τα πλεονεκτήματα που προσφέρουν τα συστήματα SDR. Με την χρήση της αναπτυξιακής πλατφόρμας ZedBoard Zynq-7000 ARM/FPGA SoC και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC έγινε υλοποίηση των βασικών αναλογικών διαμορφώσεων και παραμετροποίηση - βελτιστοποίηση της απόδοσης τους ,σε πραγματικό χρόνο, έτσι ώστε να λειτουργούν αξιόπιστα και εντός των αρχικών προδιαγραφών ή απαιτήσεων του εκάστοτε τηλεπικοινωνιακού συστήματος.

Στην συνέχεια έγινε η σχεδίαση και υλοποίηση κάποιων βασικών ψηφιακών διαμορφώσεων (QAM, OFDM, QPSK), τόσο στο καθεστώς «Μοντέλο Διεπαφής Λογισμικού – Streaming», όσο και στο καθεστώς «Επεξεργαστής σε βρόχο - Processor-in-the-Loop». Σε αυτά τα παραδείγματα, έγινε εμφανές ότι ένα τηλεπικοινωνιακό σύστημα SDR, μπορεί εύκολα να παραμετροποιηθεί και να λειτουργήσει με πολλούς τρόπους και κυρίως αξιόπιστα. Αλλάζοντας τις βασικές παραμέτρους λειτουργίας του (π.χ. κεντρική συχνότητα εκπομπής, συντελεστής ενίσχυσης), όπως αυτές καθορίζονται κατά τα αρχικά στάδια της σχεδίαση του αλγόριθμου του τηλεπικοινωνιακού μοντέλου, και στην συνέχεια, εκτέλεση του προγράμματος σε πραγματικό χρόνο, είναι δυνατή η προσαρμογή του, στις εκάστοτε ανάγκες, χωρίς να απαιτείται η αφαίρεση ή η προσθήκη κάποιων άλλων κυκλωματικών διατάξεων.

Βασική επιδίωξη της παρούσας Διπλωματικής εργασίας ήταν η ανάπτυξη ενός πλήρως αυτόνομου Τηλεπικοινωνιακού Συστήματος SDR, με την χρήση της αναπτυξιακής πλατφόρμας ZedBoard Zynq-7000 ARM/FPGA SoC και της RF αναπτυξιακής πλακέτας AD-FMCOMMS4 FMC. Σχεδιάστηκε και υλοποιήθηκε ένας πομπός QPSK, με την χρήση του Simulink και ακολουθώντας μία σειρά από βήματα, έγινε εφικτός ο προγραμματισμός της αναπτυξιακής πλακέτας, έτσι ώστε, με την ενεργοποίηση της να εκτελεί αυτόνομα τον αλγόριθμο αυτό.

Σαν μελλοντικός στόχος αυτής της διπλωματικής, είναι η σχεδίαση του αλγορίθμου ενός δέκτη QPSK και η ανάπτυξη του σε μία δεύτερη αναπτυξιακή πλακέτα, με σκοπό την ανάπτυξη του πομπού και του δέκτη, ενός αυτόνομού Τηλεπικοινωνιακού Συστήματος SDR. Με αυτόν το τρόπο θα μπορεί να μελετηθούν, κάτω από πραγματικές συνθήκες, οι δυνατότητες τόσο του αλγόριθμου όσο και του ίδιου του υλικού, για την περαιτέρω βελτίωση του Τηλεπικοινωνιακού Συστήματος SDR. Έχοντας δύο διαφορετικές αναπτυξιακές πλακέτες, θα μπορούν να μελετηθούν στην πράξη η απόδοση του αλγόριθμού όσο αφορά τον Ρυθμό Μετάδοσης Εσφαλμένου Bit - Bit Error Rate (BER) [110] και την Διασυμβολική Παρεμβολή - Intersymbol Interference (ISI) [111] παράγοντες κρίσιμοί για την ανάπτυξη ενός πραγματικού Τηλεπικοινωνιακού Συστήματος SDR.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] *Software defined radio: architectures, systems, and functions. Markus Dillinger, Kambiz Madani, Nancy Alonistioti, 2003.*

[2] *Software defined radios – overview and hardware, News from Rohde & Schwarz 182,2004/II, pp 58-60.*

[3] *History and Background of Cognitive Radio Technology, Bruce A. Fette, in Cognitive Radio Technology (Second Edition), 2009.*

[4] *Luis F. Chaparro, Aydin Akan, in Signals and Systems Using MatLab (Third Edition), 2019.*

[5] *Pozar, David M. (2001). Microwave and RF Design of Wireless Systems. Wiley. pp. 103 – 104.*

[6] *Antoniou, Andreas (2006). Digital Signal Processing. McGraw-Hill. p. 830.*

[7] *Channelization for Network Coding in Wireless Networks, Raju Kumar∗, Heesook Choi†, JaeSheung Shin‡ and Thomas La Porta∗∗Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, †Sprint ATL, 1 Adrian Court, Burlingame, CA 94010, Converged Access Network Research Team, Electronics and Telecommunications Research Institute, DaeJeon, Korea (South), 2008.*

[8] *Software-defined Radios: Architecture, State-of-the-art, and Challenges Rami Akeela, and Behnam Dezfouli Internet of Things Research Lab, Department of Computer Engineering, Santa Clara University, USA*

[9] *Montanaro, James et al. (1997). "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor". Digital Technical Journal, vol. 9, no. 1. pp. 49–62.*

[10] *Sora: High Performance Software Radio Using General Purpose Multi-Core Processors: Kun Tan, Jiansong Zhang, Ji Fang, He Liu, Yusheng Ye, Shen Wang§ Yongguang Zhang, Haitao Wu, Wei Wang† Geoffrey M. Voelker, Microsoft Research Asia, Beijing, China, Tsinghua University, Beijing, China, Beijing Jiaotong University, Beijing, China, UCSD, La Jolla, USA.*

[11] *KUAR: A Flexible Software-Defined Radio Development Platform, 2007 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 17-20 April 2007, IEEE, Dublin, Ireland.*

[12] *Journal of Physics: Conference Series, Physical principles of work and basic features of Universal Software Radio Peripheral, I V Ryabov et al 2021 J. Phys.: Conf. Ser. 1728 012013.*

[13] *Barron, E. T.; Glorioso, R. M. (September 1973). "A micro controlled peripheral processor". MICRO 6: Conference Record of the 6th Annual Workshop on Microprogramming. Micro 6: 122–128).*

[14] *Lee, R.B. (1995). "Realtime MPEG video via software decompression on a PA-RISC processor". digest of papers Compcon '95. Technologies for the Information Superhighway. pp. 186–192.*

[15] *Flynn, Michael J. (September 1972). "Some Computer Organizations and Their Effectiveness" (PDF). IEEE Transactions on Computers. C-21.*

[16] *Flynn, Michael J. (September 1972). "Some Computer Organizations and Their Effectiveness" (PDF). IEEE Transactions on Computers. C-2.*

[17] *Barr, Keith (2007). ASIC Design in the Silicon Sandbox: A Complete Guide to Building Mixed-Signal Integrated Circuits. McGraw Hill Professional.*

[18] *Frank Vahid (2010). Digital Design with RTL Design, Verilog and VHDL (2nd ed.). John Wiley and Sons. p. 247.*

[19] *Huang, Chi-Lai; Su, S.Y.H. "Approaches for Computer-Aided Logic System Design Using Hardware Description Language". Proceedings of International Computer Symposium 1980, Taipei, Taiwan, December 1980. pp. 772–790.*

[20] *Peter J. Ashenden, "The Designer's Guide to VHDL, Third Edition (Systems on Silicon)", 2008.*

[21] *McFarland, M.C.; Parker, A.C.; Camposano, R. (February 1990). "The high-level synthesis of digital systems". Proceedings of the IEEE. 78 (2): pp. 301–318.*

[22] *Software-defined Radios: Architecture, State-of-the-art, and Challenges Rami Akeela, and Behnam Dezfouli Internet of Things Research Lab, Department of Computer Engineering, Santa Clara University, USA, σελ. 9.*

[23] *Dawis, E. P. (2001). Architecture of an SS7 Protocol Stack on a Broadband Switch Platform using Dualistic Petri Nets. Communications, Computers and signal Processing, 2001. PACRIM. 2001 IEEE Pacific Rim Conference on Volume 1, 2001 Page(s):323 - 326 vol.1*

[24] Vivado Design Suite User Guide, Getting Started, G910 (v2018.2) June 6, 2018, by Xilinx.

[25] *SDSoC Environment User Guide UG1027 (v2019.1), May 22, 2019, From Xilinx.*

[26] *OpenCL-based Design Methodology for Application-Specific: Processors Pekka O. J¨a¨askel¨ainen, Carlos S. de La Lamay, Pablo Huertay and Jarmo H. Takala, Tampere University of Technology, Department of Computer Systems, Tampere, Finland, Rey Juan Carlos, Department of Computer Architecture, Mostoles, Madrid, Spain.*

[27] *Improving the performance of high-level synthesis, Microprocessing and Microprogramming, 1989, Peter Marwedel.*

[28] *System-on-Chip Design Using High-Level Synthesis Tools: Erdal Oruklu1*, Richard Hanley1, Semih Aslan2, Christophe Desmouliers1, Fernando M. Vallina3, Jafar Saniie1, 1Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, USA, 2Ingram School of Engineering, Texas State University, San Marcos, USA, 3Xilinx Inc., San Jose, November 20, 2011.*

[29] *Using MaxCompiler for the high-level synthesis of trigger algorithms :S. Summers (Imperial Coll., London), A. Rose (Imperial Coll., London), P. Sanders (Parsys, London), Feb 6, 2017, Topical Workshop on Electronics for Particle Physics, Karlsruhe, Germany, Published in: JINST 12 (2017) 02, C02015.*

[30] *LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems Share on: Andrew Canis, Jongsok Choi, Mark Aldham, Victor Zhang, Ahmed Kammoona, Tomasz Czajkowski, Stephen D. Brown, Jason H. Anderson: ACM Transactions on Embedded Computing Systems Volume 13Issue 2September 2013 Article No.: 24pp 1–27.*

[31] MATLAB for Artificial Intelligence, From MathWorks, https://www.mathworks.com/?s_tid=gn_logo (accessed Feb. 27, 2021).

[32] *MATLAB Coder™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[33] *Simulink Coder™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[34] *Embedded Coder™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[35] *Arm Holdings. "AMBA AXI and ACE Protocol Specification" (PDF). developer.arm.com. pp. 109–118. Retrieved 5 July 2019.*

[36] *HDL Coder™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[37] Xilinx, From Xilinx, https://www.xilinx.com/ (accessed Feb. 27, 2021).

[38] *Software-defined Radios: Architecture, State-of-the-art, and Challenges Rami Akeela, and Behnam Dezfouli Internet of Things Research Lab, Department of Computer Engineering, Santa Clara University, USA, σελ. 11.*

[39] *MetaCL: Automated "Meta" OpenCL Code Generation for High-Level Synthesis on FPGA: Paul Sathre ,Dept. of CS Virginia Tech, Atharva Gondhalekar, Dept. of ECE Virginia Tech, Mohamed Hassan, Dept. of ECE Virginia Tech, Wu-chun Feng, Depts. of CS & ECE, Virginia Tech, HPEC 2020, September, Waltham, MA.*

[40] *Using GNU Radio for Analog Communications, Hackspace Brussels - January 31, 2019.*

[41] *GNU Radio Tutorials, Labs 1 – 5, Balint Seeber, Ettus Research. Version 1.0 (18th April 2014).*

[42] *Implementation of FM Transceiver using Software Defined Radio (SDR), Devidas Kushnure, Murtaza Jiniyawala, Sushama Molawade, Snehal Patil 1Assistant Professor,2Student,3Student, 4Student Department of Electronics and Telecommunication Engineering, VPKBIET, Baramati, India.*

[43] *The Universal Software Radio Peripheral (USRP) Family of Low-Cost SDRs, Matt Ettus, Martin Braun, Book Editor(s): Oliver Holland, Hanna Bogucka, Arturas Medeisis, 2015, ch.-1.*

[44] *Jeffrey., Travis (2006). LabVIEW for everyone: graphical programming made easy and fun. Kring, Jim. (3rd ed.). Upper Saddle River, NJ: Prentice Hall.*

[45] *LabVIEW for Everyone: Graphical Programming Made Easy and Fun: Jeffrey Travis, J. Kring, Published 2006, Computer Science.*

[46] *Vasiliadis, Giorgos; Antonatos, Spiros; Polychronakis, Michalis; Markatos, Evangelos P.; Ioannidis, Sotiris (September 2008). "Gnort: High Performance Network Intrusion Detection Using*

Graphics Processors" (PDF). Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID). Lecture Notes in Computer Science. 5230: pp. 116–134.

[47] *NVIDIA CUDA software and GPU parallel computing architecture, January 2007, Conference: Proceedings of the 6th International Symposium on Memory Management, ISMM 2007, Montreal, Quebec, Canada, October 21-22, 2007, David B. KirkDavid B. Kirk.*

[48] GNU Radio LTE Receiver. From GitHub, https://github.com/kit-cel/gr-lte (accessed Mar 03, 2021).

[49] *Radio Astronomy Software Defined Receiver Project, by Bogdan Vacaliuc, Marcus Leech, Paul Oxley, Richard Flagg, David E Fields, June, 2011, Washington, Virginia, United States of America, Annual meeting of the Society of Amateur Radio Astronomers, Jun 25, 2011 - Jun 28, 2011.*

[50] *Radio Astronomy with RASDR2 and RASD Rviewer1, D. Fields2, P. Oxley, B. Vacaliuc3, C. Lyster, S. Kurtz4, Z. Tamosevicius5 and S. Berl, Tamke-Allan Observatory6, Roane State Community College, 276 Patton Lane, Harriman, TN 37748.*

[51] *The Nautical Institute. "Automatic Identification System (AIS): A Human Factors Approach" (PDF). www.nautinst.org. The Nautical Institute. Archived from the original (PDF) on August 12, 2011. Retrieved 25 January 2015.*

[52] *Peppler, I.L.: From the Ground Up, pages 238–239. Aviation Publishers Co. Limited, Ottawa Ontario, Twenty Seventh Revised Edition, 1996.*

[53] Aviation transponder interrogation modes, From RTL-SDR.COM, https://www.rtl-sdr.com/tag/gr-air-modes/ (accessed Mar 03, 2021).

[54] *Introduction to the Software-defined: Radio Approach, A. F. B. Selva, A. L. G. Reis, K. G. Lenzi, L. G. P. Meloni, Member, IEEE and S. E. Barbin, Member, IEEE, IEEE LATIN AMERICA TRANSACTIONS, VOL. 10, NO. 1, JAN. 2012.*

[55] *NOAA Weather Radio, NOAA's KLM User's Guide: Section 4.2 – APT System.*

[56] *The design and benefits of an automatic picture transmission (APT) network J. Clodman, To cite this article: J. Clodman (1969) The design and benefits of an automatic picture transmission (APT) network, Atmosphere, 7:1, 11-32, DOI: 10.1080/00046973.1969.9676565,To link to this article: https://doi.org/10.1080/00046973.1969.9676565,Published online: 18 Apr 2011.*

[57] *OpenHPSDR (High Performance Software Defined Radio) for VHF/UHF/Microwave, version 3.0, by Roger Rehr W3SZ.*

[58] *High Performance Software Defined Radio, Overview and Current Status, Warren Pratt, NR0V, John Westmoreland, AJ6BC, Pacificon 2014.*

[59] *Setting up Gqrx Software Defined Radio on Ubuntu 18.04 Using Remote Raspberry Pi SDR with rtl_tcp, Date Sun 17 March 2019 Tags SDR / linux / blog).*

[60] *Commentary on DttSP By Jonathan Naylor ON/G4KLX.*

[61] *DRM Introduction and Implementation Guide" (PDF; 6.7 MB). DRM. pp. 22.*

[62] *Digital Radio Mondiale and How to Receive it: By Prof. K. Padmanabhan, Dr S. Ananthi and Dr N. Mohan, November 19, 2018.*

[63] *Studies in Software-Defined Radio System Implementation Harold A. Haldren, III A Senior Thesis submitted in partial fulfillment of the requirements for graduation in the Honors Program Liberty University Spring 2014, σελ. 8.*

[64] *Implementing Software Defined Radio, Eugene Grayver, Springer Science+Business Media New York 2013, Springer, New York, NY, σελ. 37-41.*

[65] Avnet, https://www.avnet.com/wps/portal/us (accessed Mar 12, 2021).

[66] Diligent, https://store.digilentinc.com/ (accessed Mar 12, 2021).

[67] *ZedBoard™ Getting Started Guide, Version 7.0, From Avnet.*

[68] *XA Artix-7 FPGAs Data Sheet: Overview, DS197 (v1.3) November 15, 2017, From Xilinx.*

[69] *The Zynq Book Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC, Xilinx, University of Strathclyde Glasgow, σελ. 134-135.*

[70] *Hennessy, John L; Patterson, David A. (2011). Computer Architecture: A Quantitative Approach. pp. 355–356.*

[71] *1149.7-2009 - IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture, Publisher: IEEE, 10 Feb. 2010.*

[72] *Adam Osborne, An Introduction to Microcomputers Volume 1: Basic Concepts, Osborne-McGraw Hill Berkeley California USA, 1980, pp. 116–126.*

[73] *AD-FMCOMMS4-EBZ User Guide, From Analog Devices.*

[74] Analog Devices, https://www.analog.com/en/index.html (accessed Mar 15, 2021).

[75] *GLOBAL SYMPOSIUM FOR REGULATORS, Dubai World Trade Center, Dubai, United Arab Emirates, 5-7 February 2007, Work in progress, for discussion purposes, FIXED-MOBILE CONVERGENCE, PREPARED BY EWAN SUTHERLAND, TELECOMMUNICATION POLICY ANALYST.*

[76] *AD9361 Reference Manual UG-570, From Analog Devices.*

[77] *Sevick, Jerry (W2FMI). Transmission Line Transformer, The American Radio Relay League, 1990, cp 1-4.*

[78] *TCM1-63AX+, Surface Mount RF Transformers, 50Ω / 10 to 6000MHz Data Sheet, From Mini-circuits.*

[79] *Müller, Matthias M.; Padberg, Frank. "About the Return on Investment of Test-Driven Development" (PDF). Universität Karlsruhe, Germany.*

[80] *Developing Secure Websites Using Feature Driven Development (FDD): A Case Study: Adila Firdaus, I. Ghani, Norizan Binti Mohd. Yasin, Published 2013, Computer Science, Journal of Clean Energy Technologies.*

[81] *Vigilante, McCune, Reynaert. "To EVM or Two EVMs?". doi:10.1109/MSSC.2017.2714398. S2CID 6849707.*

[82] *LVDS Application and Data Book, SLLD009, Texas Instruments, November 2002.*

[83] *RF Agile Transceiver, Data Sheet AD9361, From Analog Devices.*

[84] *Martin Sauter (2010). "3.7.1 Mobility Management in the Cell-DCH State". From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband. John Wiley & Sons. p. 160.*

[85] *Lyons, Richard (2001). Understanding Digital Signal Processing. Prentice Hall. pp. 304.*

[86] *Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XIX. Edited by Holst, Gerald C. Proceedings of the SPIE, Volume 6941, article id. 694102, 11 pp. (2008).*

[87] Xilinx Vivado, From Xilinx, https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2020-2.html (accessed Mar 24, 2021).

[88] *White Paper: Vivado Design Suite. WP416 (v1.1) June 22, 2012, Vivado Design Suite, By: Tom Feist, Xilinx.*

[89] *Kimmig, Markus; Monperrus, Martin; Mezini, Mira (2011). "Querying source code with natural language". 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). pp. 376–379.*

[90] *Adamov, Alexander (2007). "Electronic System Level Models for Functional Verification of System-on-Chip". 2007 9th International Conference - the Experience of Designing and Applications of CAD Systems in Microelectronics. CAD Systems in Microelectronics. pp. 348–350.*

[91] *Intellectual Property (IP) Cores for Home Networking: Author: Amit Dhir, Xilinx, Mar, 2005.*

[92] *Vivado Design Suite User Guide, Getting Started, G910 (v2018.2) June 6, 2018, by Xilinx.*

[93] *Kruijtzer, W.; Vaumorin, E.; Van Der Wolf, P.; De Kock, E.; Stuyt, J.; Ecker, W.; Mayer, A.; Hustin, S.; Amerijckx, C.; De Paoli, S. (2008). Industrial IP integration flows based on IP-XACT™ standards. DATE'08. Proceedings of the conference on Design, automation and test in Europe. pp. 32–37.*

[94] *Deep Learning HDL Toolbox™ Support Package for Xilinx® FPGA and SoC Devices, User's Guide, COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[95] *Model-Based Design with Simulink, HDL Coder, and Xilinx System Generator for DSP By Kiran Kintali and Yongfeng Gu, by The MathWorks, Inc.*

[96] *Getting Started with MATLAB®, Version 7, COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[97] *Simulink™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[98] QPSK Transmitter and Receiver in Simulink, From MathWorks, https://www.mathworks.cn/help/comm/ug/qpsk-transmitter-and-receiver-in-simulink.html (accessed Mar 25, 2021).

[99] *HDL Verifier™ Getting Started Guide © COPYRIGHT 2003–2021 by The MathWorks, Inc.*

[100] Receive Tone Signal Using Analog Devices AD9361/AD9364, From MathWorks, https://www.mathworks.com/help//supportpkg/xilinxzynqbasedradio/ug/receive-tone-signal-using-analog-devices-ad9361-ad9364.html (accessed Apr 20, 2021).

[101] L. *Cordesses, Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 2) IEEE Signal Processing Magazine, DSP Tips & Tricks column, pp. 110–117, Vol. 21, No. 5, Sep. 2004.*

[102] QPSK Transmit Repeat Using Analog Devices AD9361/AD9364, From MathWorks, https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/qpsk-transmit-repeat-using-analog-devices-ad9361-ad9364.html?requestedDomain=us (accessed Apr 20, 2021).

[103] *ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ (ΠΑΔΑ), ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ, ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ, Διπλωματική Εργασία «Ανάπτυξη τηλεπικοινωνιακών υποσυστημάτων OFDM με την αναπτυξιακή πλατφόρμα ZedBoard Zynq-7000 ARM/FPGA SoC», Φοιτητής: Μπακογιάννης Δημήτριος, ΑΜ: 50106861, Επιβλέπων Καθηγητής: Καραμπέτσος Σωτήριος, Επίκουρος Καθηγητής, ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2020.*

[104] QPSK Transmitter Using Analog Devices AD9361/AD9364, From MathWorks, https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/qpsk-transmitter-using-analog-devices-ad9361-ad9364.html (accessed Apr 23, 2021).

[105] QPSK Receiver Using Analog Devices AD9361/AD9364, From MathWorks, https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/qpsk-receiver-using-analog-devices-ad9361-ad9364.html (accessed Apr 23, 2021).

[106] *The use of the Hilbert transforms to double the information rate in spread-spectrum communications, D. Shklarsky; L.B. Milstein; P.K. Das. IEEE Transactions on Vehicular Technology (Volume: 30, Issue: 1, Feb 1981), 29 – 36.*

[107] HW/SW Co-Design QPSK Transmit and Receive Using Analog Devices AD9361/AD9364, From MathWorks, https://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio/ug/hw-sw-co-design-qpsk-transmit-and-receive-using-analog-devices-ad9361-ad9364.html (accessed Aug 7, 2021).

[108] *End-to-End Wireless Digital Communication System of FPGA Based Software Defined Radio, Tuan M. Dang, Thomas Gonnot and Jafar Saniie, Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, Illinois, USA*

[109] *Communications Toolbox™ Support Package for Xilinx® Zynq®-Based Radio, User's Guide, MatLab & Simulink, R2021a, MathWorks®, σελ. 6-17.*

[110] *Καραμπέτσος. (2020). Τεχνολογίες διαμόρφωσης πολλαπλών φερόντων [OnLine]. Available: https://eclass.uniwa.gr/modules/document/index.php?course=EEE224&openDir=/5e70fe09uieJ.*

[111] *Theory and Design of Digital Communication Systems, Tri T. Ha, Naval Postgraduate School, Monterey, California, Cambridge University Press, June 2012, pp. 473 – 521.*