**2η Εργαστηριακή Άσκηση**

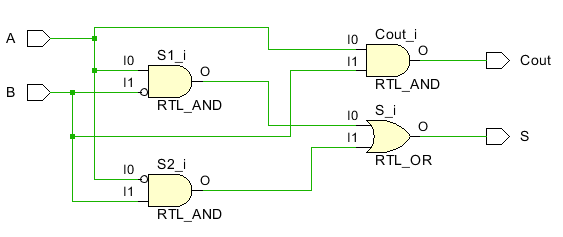
**VLSI**

Παντελαίος Δημήτριος Α.Μ.: 03118049

Μοίρας Αλέξανδρος Α.Μ.: 03118081

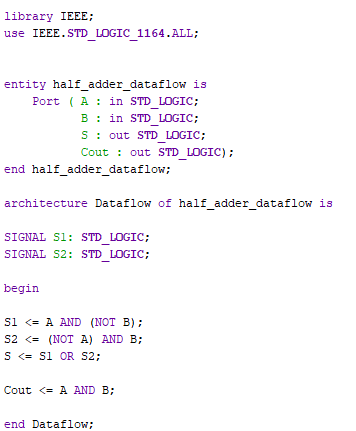
Ζήτημα 1:

Το RTL schematic του ημιαθροιστή που υλοποιήσαμε:

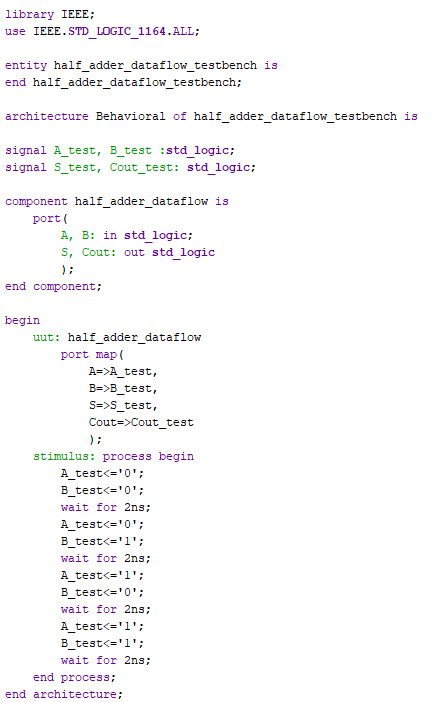


Πρόκειται για ένα απλό συνδυαστικό κύκλωμα με μία πύλη AND μεταξύ των δύο bit εισόδου ώστε να παράγεται κρατούμενο αν και τα δύο bits εισόδου είναι 1 και το λογικό άθροισμα των γινομένων Α’Β και ΑΒ’ με δύο πύλες AND και μία OR καθώς άθροισμα 1 προκύπτει αν ένα από τα δύο bits της εισόδου είναι 1. Το κύκλωμα δημιουργήθηκε από το Vivado βασισμένο στη dataflow περιγραφή μας.

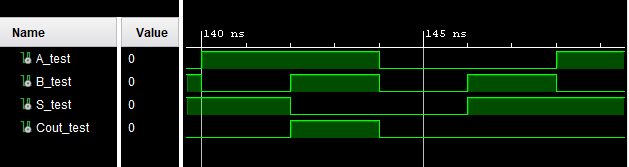
O VHDL κώδικας για την dataflow περιγραφή του half adder:



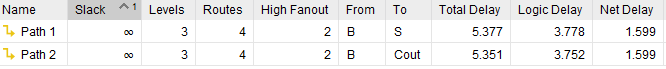
O VHDL κώδικας για τo testbench της dataflow περιγραφής του half adder:



Η προσομοίωση που επιβεβαιώνει την ορθή λειτουργία του κυκλώματός μας όπως περιεγράφη και παραπάνω:

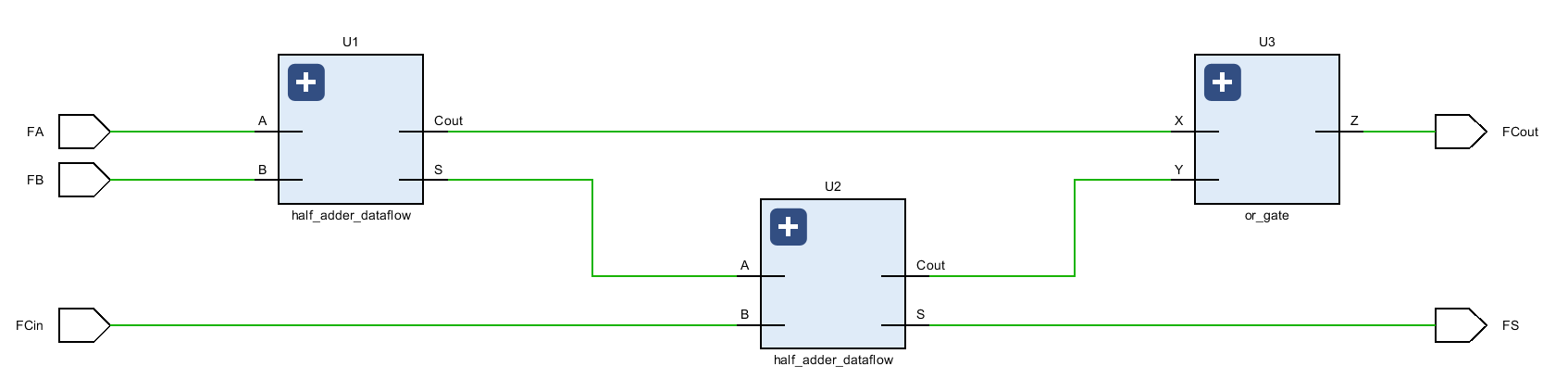


Το κρίσιμο μονοπάτι του κυκλώματος είναι αυτό από το B στο S που είναι λογικό καθώς για να παραχθεί το άθροισμα S από τις εισόδους μεσολαβούν δύο επίπεδα πυλών, ενώ στο κρατούμενο 1. Η χρονική του καθυστέρηση είναι 5.377ns.



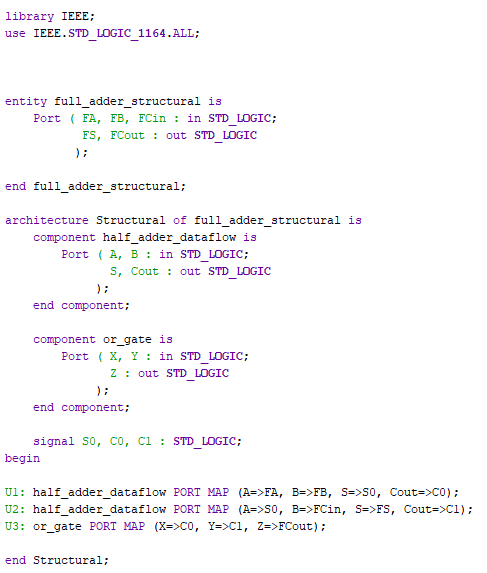
Ζήτημα 2:

Τώρα εκτός από τα FΑ, FΒ bits εισόδου έχουμε και κρατούμενο εισόδου. Το RTL Schematic του Full Adder που υλοποιήσαμε:

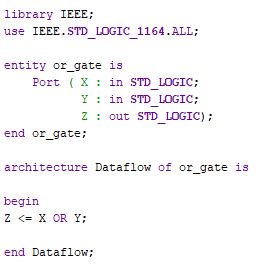


Εδώ εμείς ορίζουμε τη δομή του κυκλώματος χρησιμοποιώντας structural περιγραφή. Χρησιμοποιούμε δύο ημιαθροιστές όπου στον πρώτο αθροίζουμε τα δύο bits εισόδου Α και Β, ενώ στον δεύτερο αθροίζουμε το αποτέλεσμα της πρώτης άθροισης με το κρατούμενο εισόδου, ώστε να προκύψει το τελικό άθροισμα εξόδου, ενώ κρατούμενο στην έξοδο έχουμε αν οποιοσδήποτε από τους 2 ημιαθροιστές δώσει κρατούμενο (μόνο ένας από τους δύο μπορεί να δίνει) για αυτό και διασυνδέουμε τις εξόδους κρατουμένου τους με μια πύλη OR (την οποία υλοποιήσαμε με dataflow περιγραφή ώστε να τη χρησιμοποιήσουμε στη structural) η έξοδος της οποίας είναι το κρατούμενο του Full Adder.

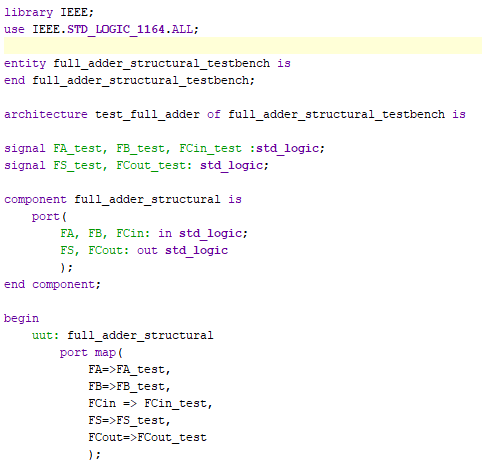
O VHDL κώδικας για την structural περιγραφή του full adder:

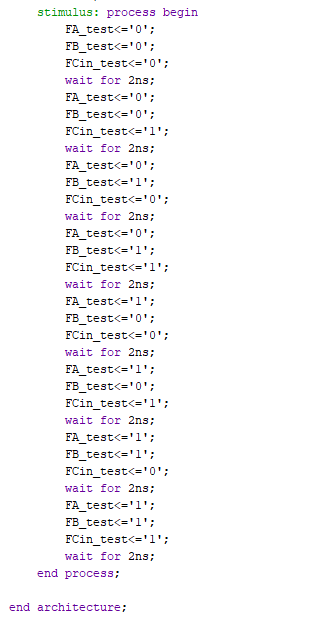


O VHDL κώδικας για την dataflow περιγραφή της πύλης OR:

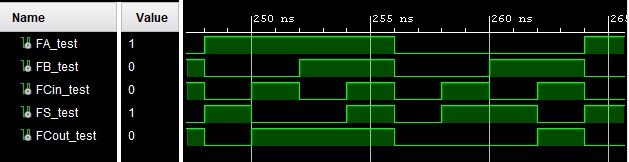


O VHDL κώδικας για τo testbench της structural περιγραφής του full adder:

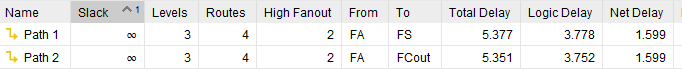




Η προσομοίωση που επιβεβαιώνει την ορθή λειτουργία του κυκλώματός μας για τις 8 δυνατές διαφορετικές εισόδους:

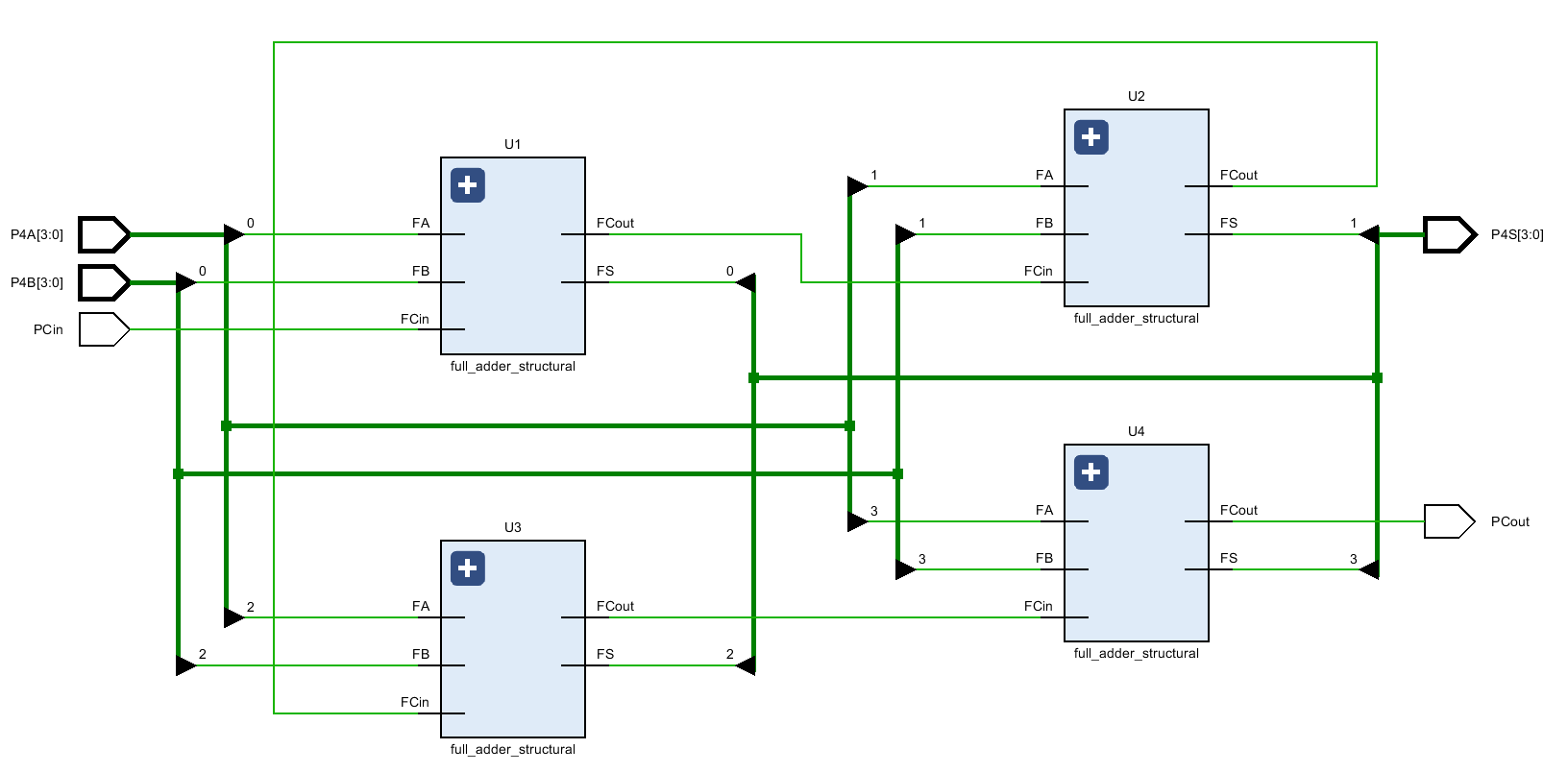


Το κρίσιμο μονοπάτι του κυκλώματος είναι από το FA στο FS καθώς χρειάζεται να υπολογιστούν δύο αθροίσματα από τους ημιαθροιστές που όπως εξηγήθηκε παραπάνω είναι χρονοβόρες διαδικασίες ενώ η πύλη OR δε φαίνεται να αυξάνει τόσο την καθυστέρηση του μονοπατιού FA-FCout. Η συνολική καθυστέρηση του πιο κρίσιμου μονοπατιού είναι 5.377ns.

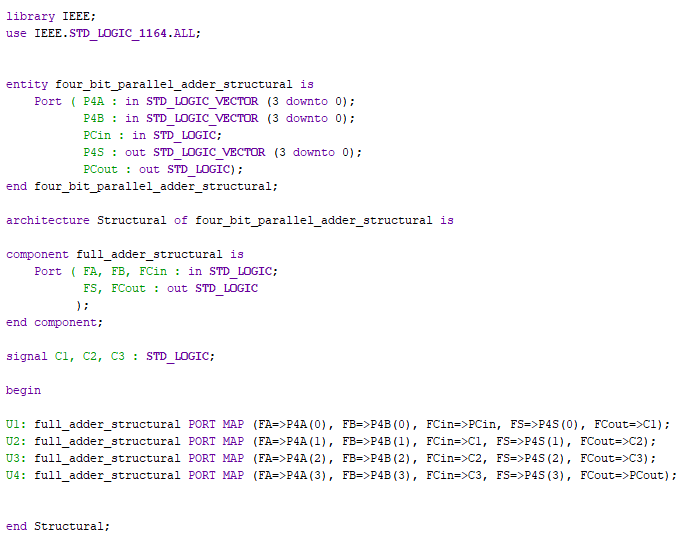


Ζήτημα 3:

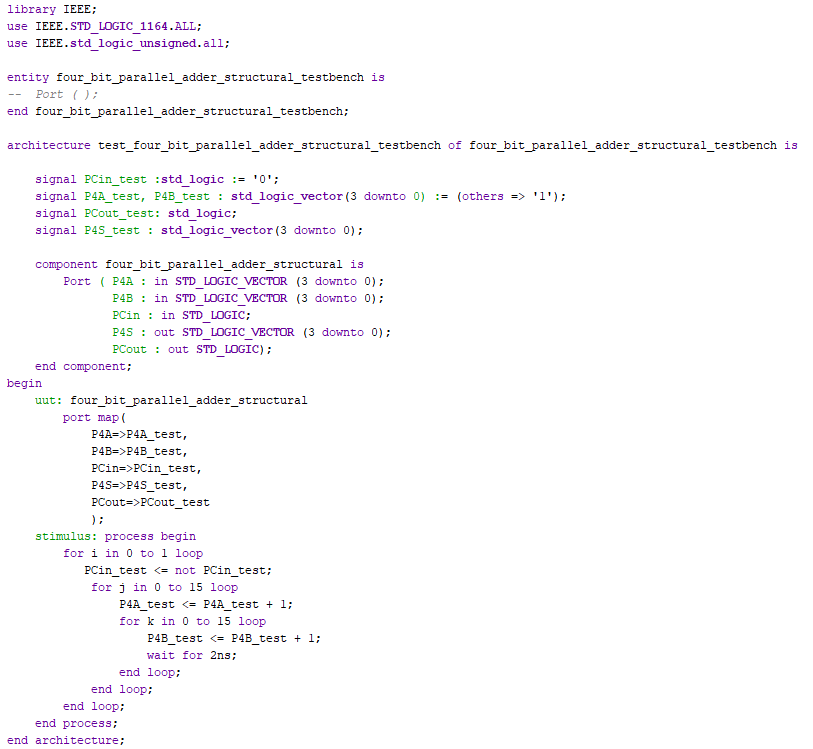
Για τον παράλληλο αθροιστή των 4 bits θα διασυνδέσουμε 4 πλήρεις αθροιστές έναν για κάθε bit, με τον πρώτο (LSB) να δέχεται από την είσοδο το κρατούμενο εισόδου του, τον 2ο από να το δέχεται από το κρατούμενο εξόδου της 1ης βαθμίδας κλπ. Το RTL Schematic στο οποίο φαίνεται η παραπάνω ιδέα είναι το εξής:



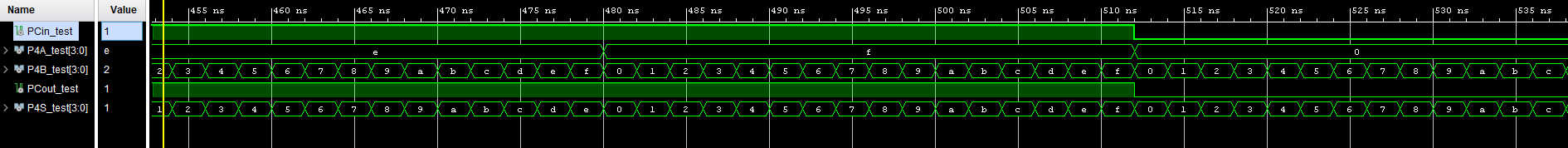
O VHDL κώδικας για την structural περιγραφή του 4 bit parallel adder:



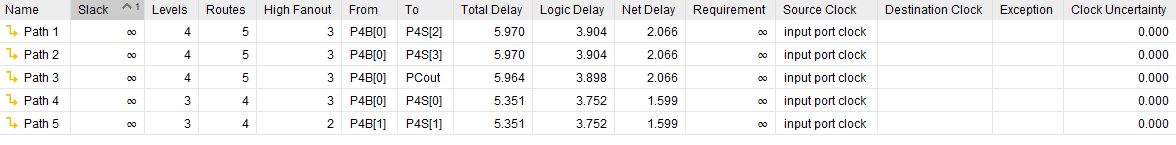
O VHDL κώδικας για τo testbench της structural περιγραφής του 4 bit parallel adder:



Ένα τμήμα της προσομοίωσης (σημαντικό καθώς εξετάζει και περιπτώσεις με κρατούμενο εισόδου και χωρίς αλλά και περιπτώσεις όπου προκύπτει κρατούμενο εξόδου) η οποία εκτελείται εξαντλητικά για όλες τις περιπτώσεις (ωστόσο δε χωράει σε ένα screenshot για να συμπεριληφθεί εξ ολοκλήρου στην αναφορά):

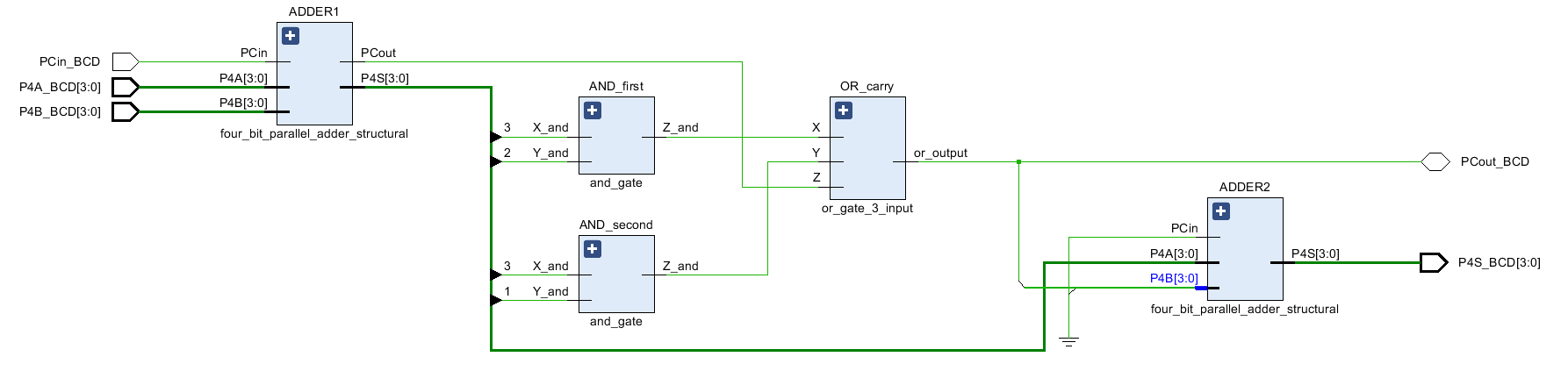


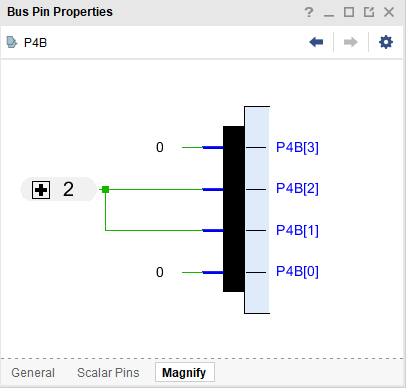
Το critical path του κυκλώματος είναι από το P4B[0] στα P4S[2], P4S[3] δηλαδή από το LSB του δεύτερου προσθετέου στα MSB του τελικού αθροίσματος που είναι λογικό γιατί για να υπολογιστεί το MSB αυτού του αθροίσματος πρέπει να υπολογιστούν ένα ένα και τα 4 αθροίσματα του 1 bit ξεκινώντας από το LSB, με κάθε ένα από αυτά να περιμένει έως ότου το προηγούμενο παράξει το κρατούμενο εξόδου του. Οπότε το MSB εξαρτάται από το κρατούμενο που θα παράξει η πρώτη βαθμίδα και σιγά σιγά θα γίνει propagate ως αυτό. Η συνολική του καθυστέρηση είναι 5.970ns.

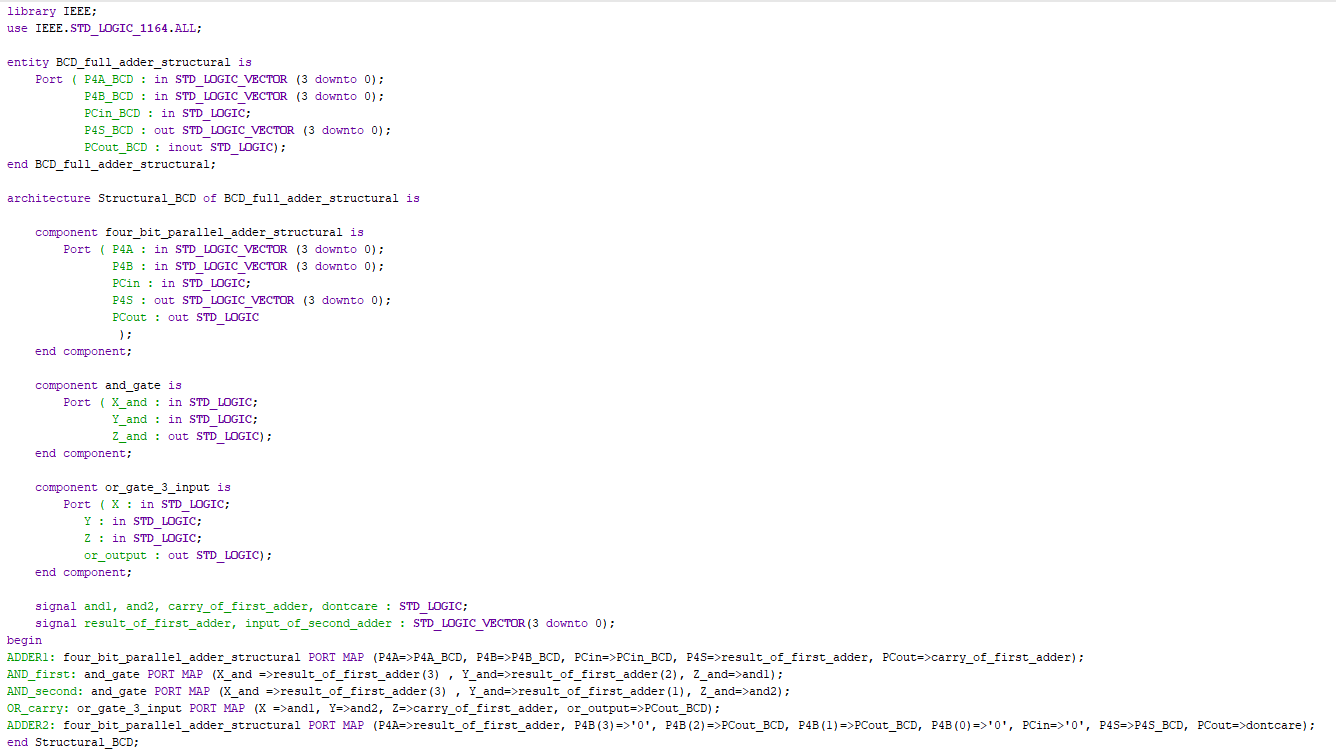


Ζήτημα 4:

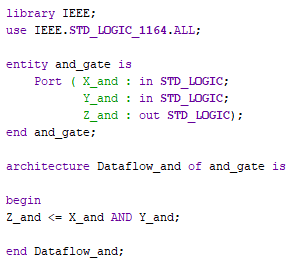
Για να κατασκευάσουμε BCD πλήρη αθροιστή θα χρησιμοποιήσουμε παράλληλους αθροιστές των τεσσάρων bit σε συνδυασμό με λογικές πύλες AND και μια OR 3 εισόδων τις οποίες κατασκευάζουμε με dataflow περιγραφή. Συγκεκριμένα θεωρούμε ότι οι δύο προσθετέοι είναι σε BCD μορφή και τους δίνουμε ως είσοδο στον πρώτο παράλληλο αθροιστή των 4 bit μαζί με το κρατούμενο εισόδου. Αν προκύψει κρατούμενο εξόδου ή είναι 1 τα bit 3 και 2 ή 3 και 1 του αθροίσματος εξόδου (εδώ χρησιμοποιούνται οι 2 AND και η OR 3 εισόδων) τότε το αποτέλεσμα της άθροισης είναι μεγαλύτερο ή ίσο του 10 άρα το κρατούμενο εξόδου τίθεται σε 1. Αν το άθροισμα είναι μεγαλύτερο ή ίσο του 10 σε αυτό προστίθεται το 6 (μέσω ενός δεύτερου παράλληλου αθροιστή 4 bit ο οποίος προσθέτει στο άθροισμα του πρώτου αθροιστή το 6 αν Cin=1 αλλιώς το 0) εκτελώντας με αυτόν τον τρόπο δεκαδική διόρθωση. Το κρατούμενο εξόδου αυτού του αθροιστή δεν έχει κάποια σημασία οπότε αγνοείται. Το RTL Schematic του κυκλώματος που περιεγράφη παραπάνω είναι το εξής:



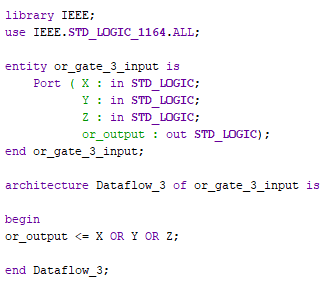


O VHDL κώδικας για την structural περιγραφή του BCD adder:

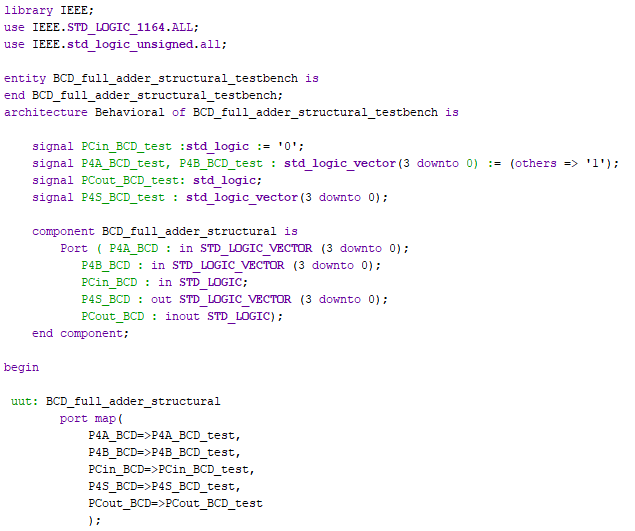
Ο VHDL κώδικας για την dataflow περιγραφή της πύλης ΑΝD:

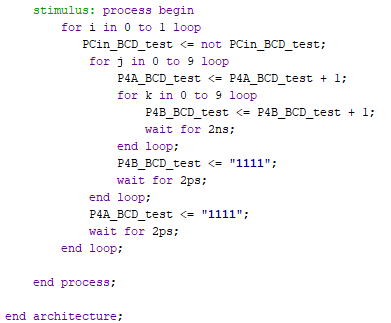


Ο VHDL κώδικας για την dataflow περιγραφή της πύλης OR 3 εισόδων:

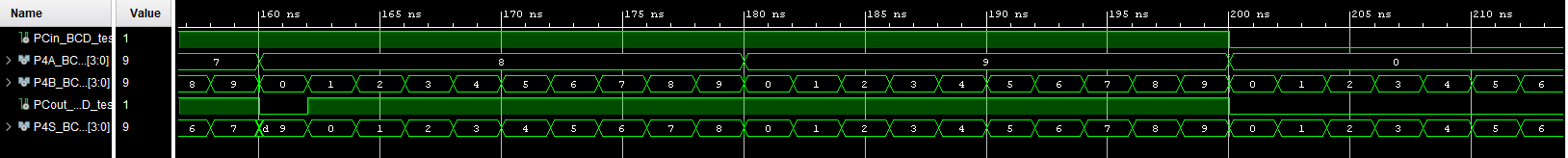


O VHDL κώδικας για τo testbench της structural περιγραφής του BCD adder:

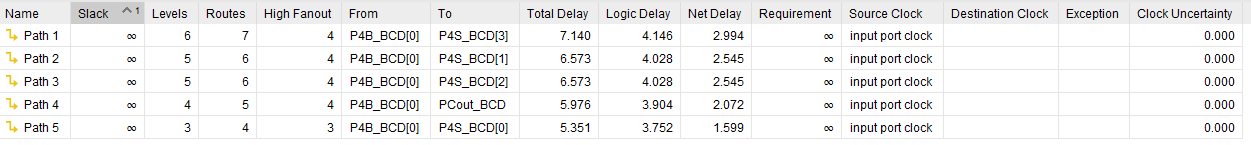




Πάλι παρουσιάζουμε ένα κρίσιμο τμήμα της προσομοίωσης καθώς η εξαντλητική είναι πολύ μεγάλη:

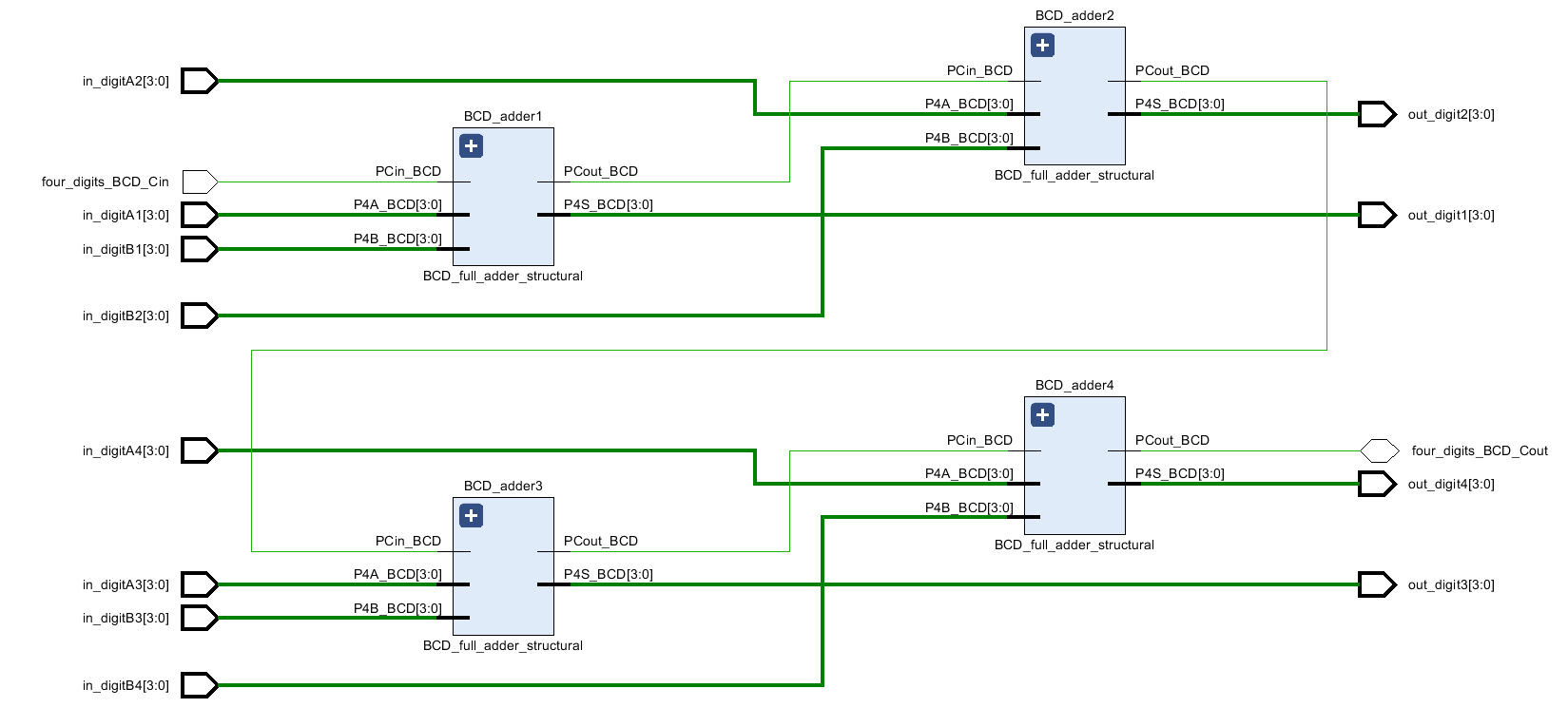


Το critical path του κυκλώματος είναι το μονοπάτι από το P4B\_BCD[0] (LSB δεύτερου προσθετέου) στο P4S\_BCD[3] (MSB αθροίσματος εξόδου) που είναι λογικό καθώς για τον υπολογισμό του αθροίσματος εξόδου μεσολαβούν δύο παράλληλοι αθροιστές των 4 bit που για κάθε έναν το πιο χρονοβόρο μονοπάτι είναι ο υπολογισμός του τελευταίου bit του αθροίσματος όπως εξηγήθηκε παραπάνω ενώ πχ το κρατούμενο εξόδου εδώ υπολογίζεται διατρέχοντας μόνο έναν παράλληλο αθροιστή. Η συνολική του καθυστέρηση είναι 7.14ns.

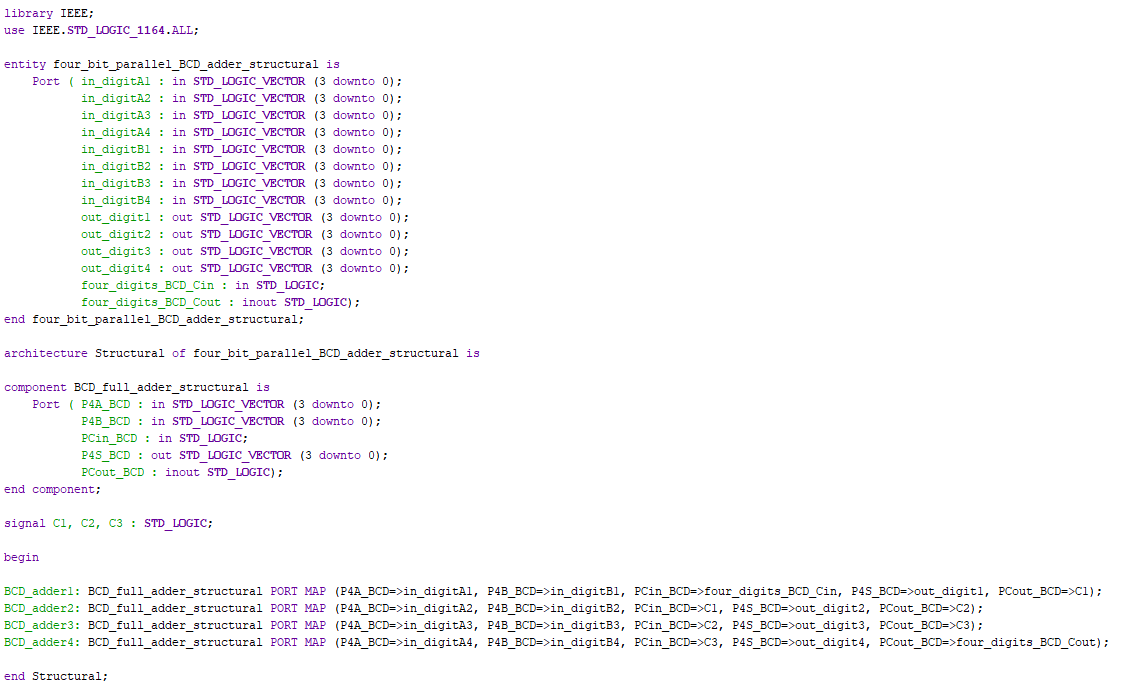


Ζήτημα 5:

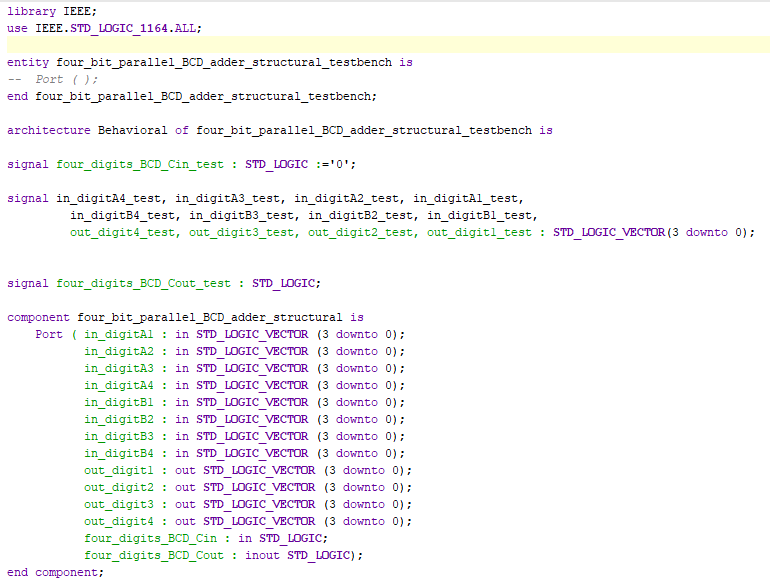
Για να υλοποιήσουμε τον παράλληλο BCD αθροιστή των 4 ψηφίων απλώς διασυνδέουμε 4 BCD Full Adders του προηγούμενου ερωτήματος με το κρατούμενο εξόδου του LSB να είναι κρατούμενο εισόδου του 2ου LSB κ.ο.κ. εντελώς αντίστοιχα με τη λογική του ζητήματος 3. Το RTL schematic αυτής της δομής:

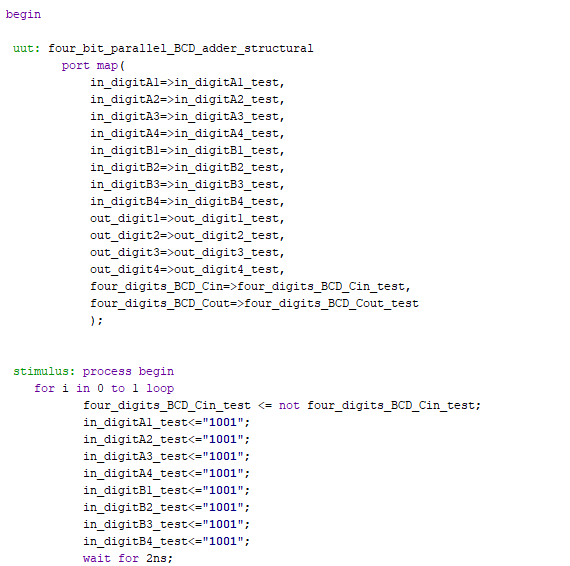


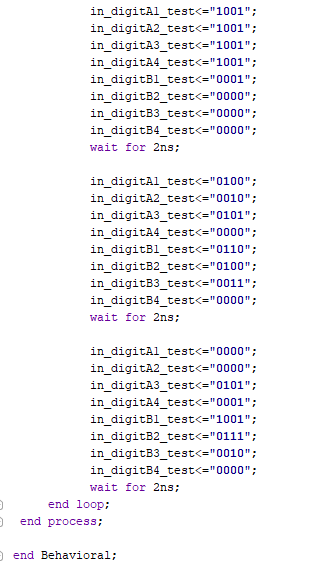
O VHDL κώδικας για την structural περιγραφή του four digits BCD adder:



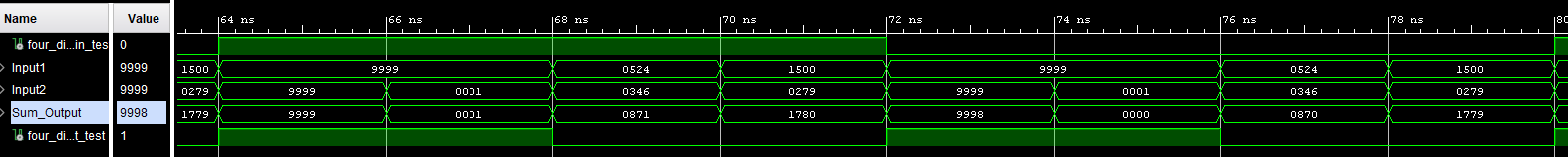
O VHDL κώδικας για τo testbench της structural περιγραφής του four digits BCD adder:







Τώρα για να ελέγξουμε το κύκλωμα, επειδή εξαντλητικά οι περιπτώσεις θα ήταν υπερβολικά πολλές (10000\*10000\*2), προσομοιώσαμε μόνο ορισμένες που θεωρήθηκαν κρίσιμες, όπως την άθροιση αριθμών που το αποτέλεσμα δε χωρά σε 4 δεκαδικά ψηφία και κάποιες περιπτώσεις αναμενόμενης λειτουργίας, με και χωρίς κρατούμενο εισόδου.



Το critical path του κυκλώματος είναι από το in\_digitB1[1] (2ο LSB του πρώτου ψηφίου του δεύτερου προσθετέου) ως τα out\_digit4[2], out\_digit4[3] (MSB του τελευταίου ψηφίου του αθροίσματος εξόδου) καθώς τόσο εντός των BCD Full Adders όσο και μεταξύ τους πρέπει το κρατούμενο να διαδίδεται από τα λιγότερο σημαντικά bits ή ψηφία στα περισσότερο. Η χρονική του καθυστέρηση είναι 11.262ns.

