

Ψηφιακή Σχεδίαση 2

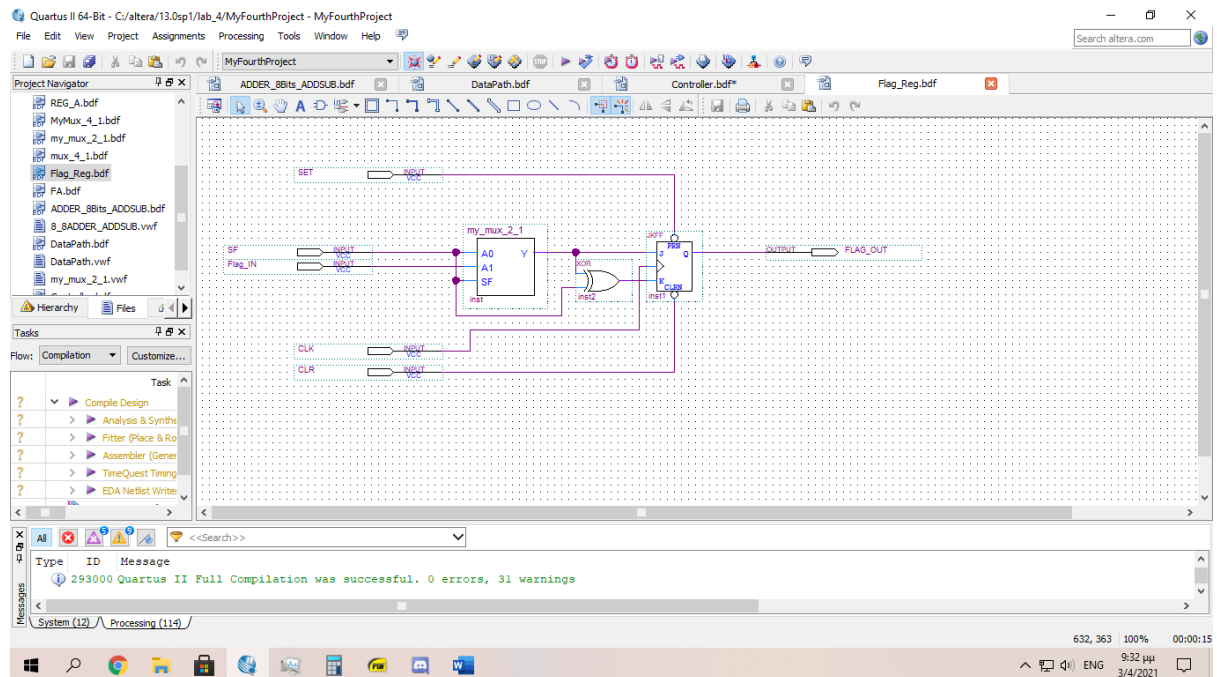
3^η Εργαστηριακή Άσκηση

Γκότσης Βασίλης 3206

Πρίφτη Ιωάννης 3321

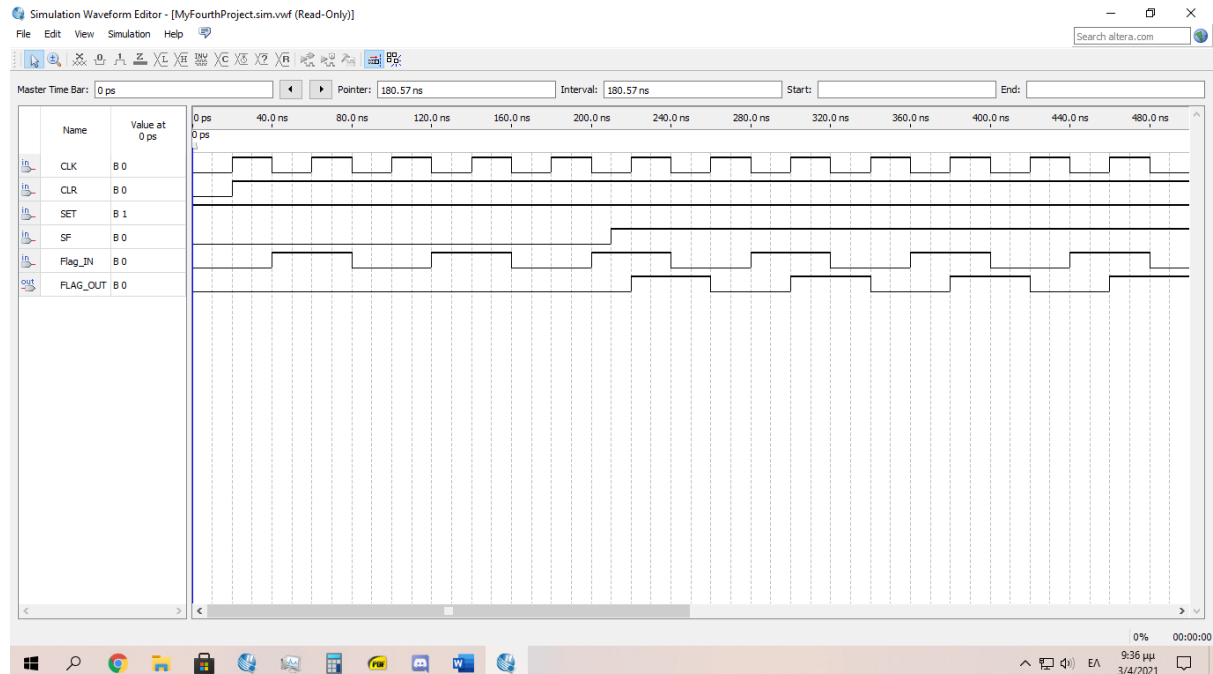
Αρχικά ξεκινήσαμε φτιάχνοντας ένα πολυπλέκτη 2 σε 1 για τις ανάγκες που αφορούν την σημαία μηδενισμού με το ίδιο τρόπο που τον έχουμε υλοποιήσει στις προηγούμενες ασκήσεις τον οποίο και βάλαμε σε κουτάκι με επιλογή το σήμα SF (Store Flag) και εξίσωση $Y = (SF' * SF) + SF * \text{Flag_IN}$.

Την έξοδο του πολυπλέκτη την διασυνδέσαμε στο JK-FF με τέτοιον τρόπο ώστε να υλοποιεί την λειτουργία που αναφέρεται στην εκφώνηση και πιο συγκεκριμένα $J = Y = (SF' * SF) + SF * \text{Flag_IN}$ και $K = (Y' * SF + Y * SF')$, όπως ακριβώς φαίνεται στην παρακάτω εικόνα:



Όπου η έξοδος FLAG_OUT ενεργοποιείται στην περίπτωση όπου τα σήματα εισόδου SF=1 και Flag_IN=1.

Εκτελώντας functional simulation το κύκλωμα φαίνεται να λειτουργεί σωστά όπως φαίνεται και στην παρακάτω εικόνα:



Όσον αφορά τους καταχωρητές A και B τους υλοποιήσαμε με παρόμοιο τρόπο όπως και στις προηγούμενες εργαστηριακές ασκήσεις και κάποιες αλλαγές φυσικά.

Αποτελείται από δύο επίπεδα πολυπλεκτών και ένα επίπεδο dff.

Το κατώτερο επίπεδο αποτελείται από καταχωρητές 2 σε 1 όπως τους υλοποιήσαμε προηγουμένως με κοινή είσοδο επιλογής SEL_A για τον καταχωρητή A και SEL_B για τον καταχωρητή B, εισόδους IO_7-IO_0 και I1_7-I1_0 και έξοδο $Y = (SEL_X' * IO_X) + (SEL_X * IO_X')$, όπου X είναι από 0..7.

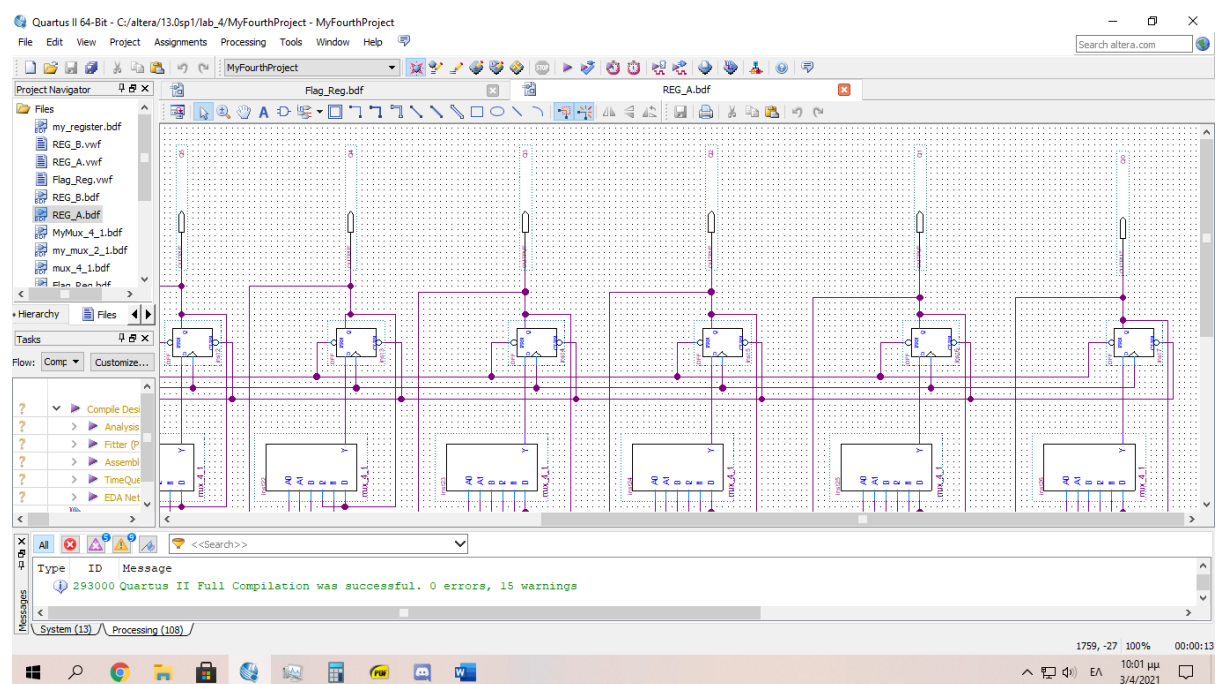
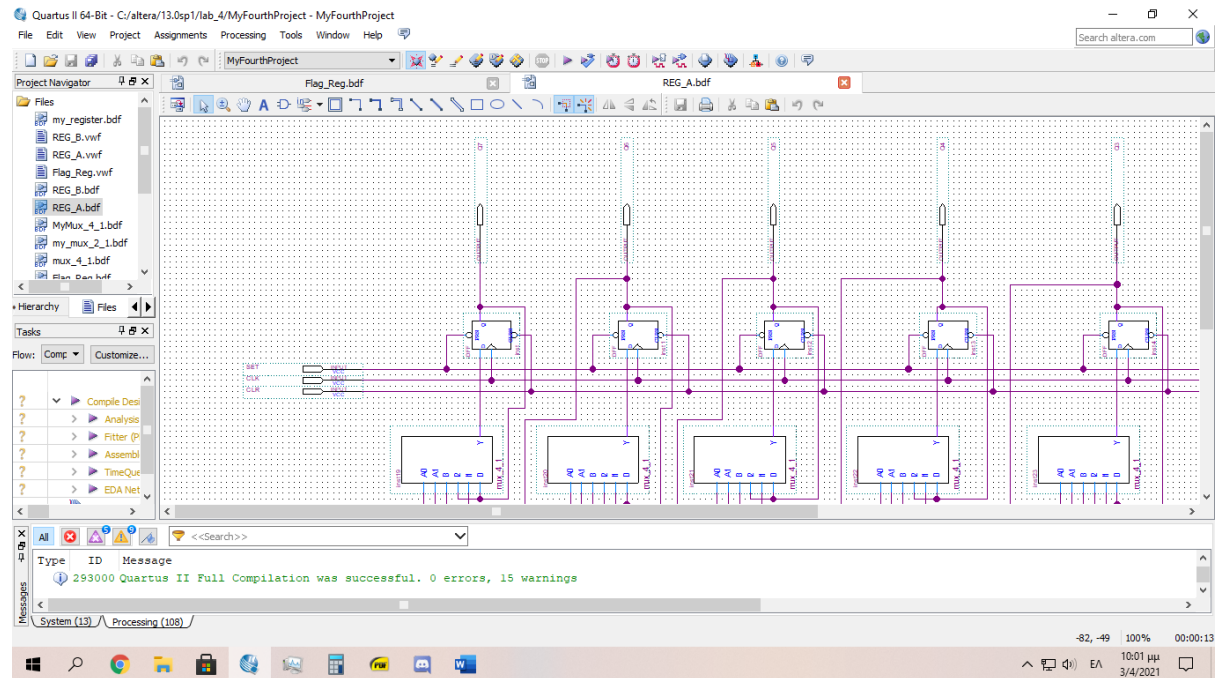
Αυτό το επίπεδο το υλοποιήσαμε ώστε να μπορούμε να επιλέγουμε ανάμεσα σε παράλληλη φόρτωση των δεδομένων από τις εξωτερικές εισόδους που διασυνδέονται με dip switches (όταν SEL_X=0) ή φόρτωση των δεδομένων από τις εσωτερικές εισόδους του συστήματος (πχ του αθροιστή).

Το δεύτερο επίπεδο πολυπλεκτών αποτελείται από πολυπλέκτες 4 σε 1 όπως τους υλοποιήσαμε και σε προηγούμενες εργαστηριακές ασκήσεις με κοινές εισόδους επιλογής A1=LS και A0=EN. Όσον αφορά τις εισόδους των πολυπλεκτών χρησιμοποιήσαμε τις I0 και I2 να δέχονται τις εξόδους του καταχωρητή ώστε το κύκλωμα να παραμένει αδρανές όταν το EN=0. Στις εισόδους I3 υλοποιείται η αριστερή ολίσθηση όταν EN=1 και L_S=1 και τέλος στις εισόδους I1 μπαίνει η είσοδος που προέρχεται από το κατώτερο επίπεδο πολυπλεκτών που είναι ουσιαστικά η φόρτωση δεδομένων από τα dip switches ή από το εσωτερικό του συστήματος όπως προαναφέρθηκε όταν EN=1 και L_S=0

Τέλος έχουμε τα dff που παίρνουν ως είσοδο τις εξόδους των πολυπλεκτών 4 σε 1 και τις αποθηκεύουν και βγάζουν αντίστοιχα τις εξόδους Q0-Q7.

Οι καταχωρητές A και B είναι ακριβώς ίδιοι αλλά όσον αφορά τον καταχωρητή A, επειδή δεν επιθυμούμε να κάνει ολίσθηση αυτό που σκεφτήκαμε είναι να έχουμε συνεχώς τον διακόπτη $L_S_A=0$ (Κάνει μόνο Load και όχι Shift).

Το κύκλωμα φαίνεται στις παρακάτω εικόνες:



Quartus II 64-Bit - C:/altera/13.0sp1/lab_4/MyFourthProject - MyFourthProject

File Edit View Project Assignments Processing Tools Window Help

MyFourthProject

Project Navigator

- Files
 - my_register.bdf
 - REG_B.vwf
 - REG_A.vwf
 - Flag_Reg.vwf
 - REG_B.bdf
 - REG_A.bdf
 - MyMux_4_1.bdf
 - my_mux_2_1.bdf
 - mux_4_1.bdf
 - Elan Dan.bdf
- Hierarchy
 - Files
- Tasks
 - Comp
 - Customize...
- Flow: Comp

Messages

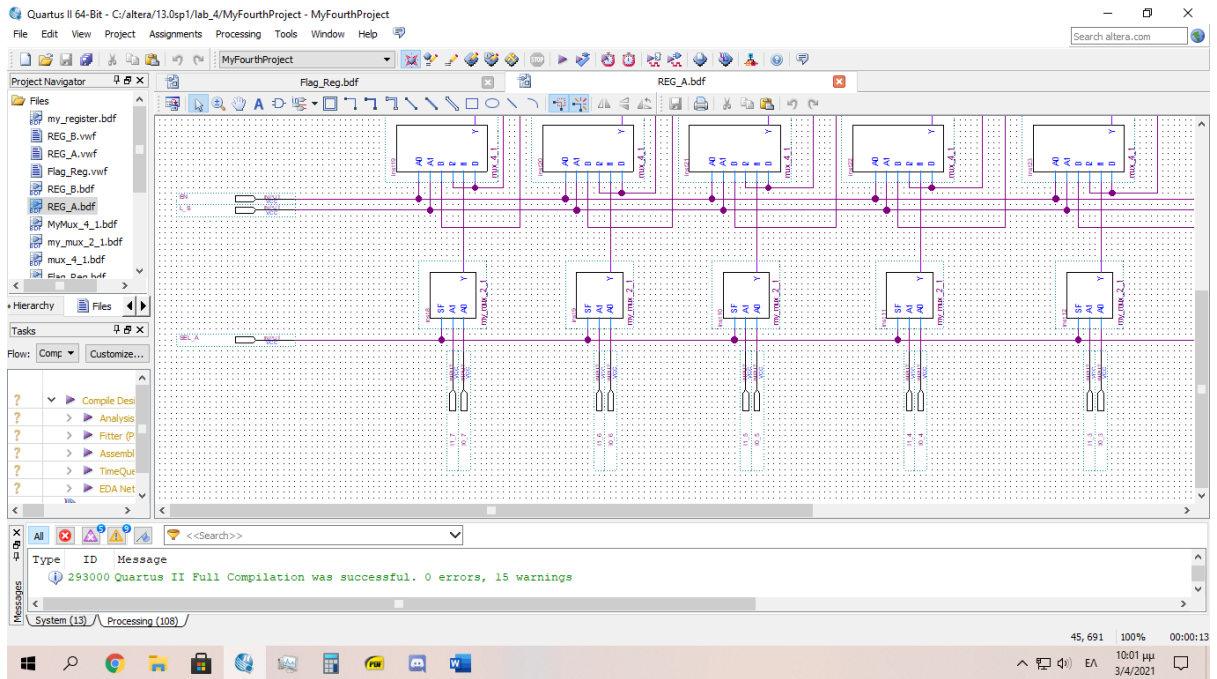
Type ID Message

293000 Quartus II Full Compilation was successful. 0 errors, 15 warnings

System (13) / Processing (108)

45,691 100% 00:00:13

10:01 μm 3/4/2021



Quartus II 64-Bit - C:/altera/13.0sp1/lab_4/MyFourthProject - MyFourthProject

File Edit View Project Assignments Processing Tools Window Help

MyFourthProject

Project Navigator

- Files
 - my_register.bdf
 - REG_B.vwf
 - REG_A.vwf
 - Flag_Reg.vwf
 - REG_B.bdf
 - REG_A.bdf
 - MyMux_4_1.bdf
 - my_mux_2_1.bdf
 - mux_4_1.bdf
 - Elan Dan.bdf
- Hierarchy
 - Files
- Tasks
 - Comp
 - Customize...
- Flow: Comp

Messages

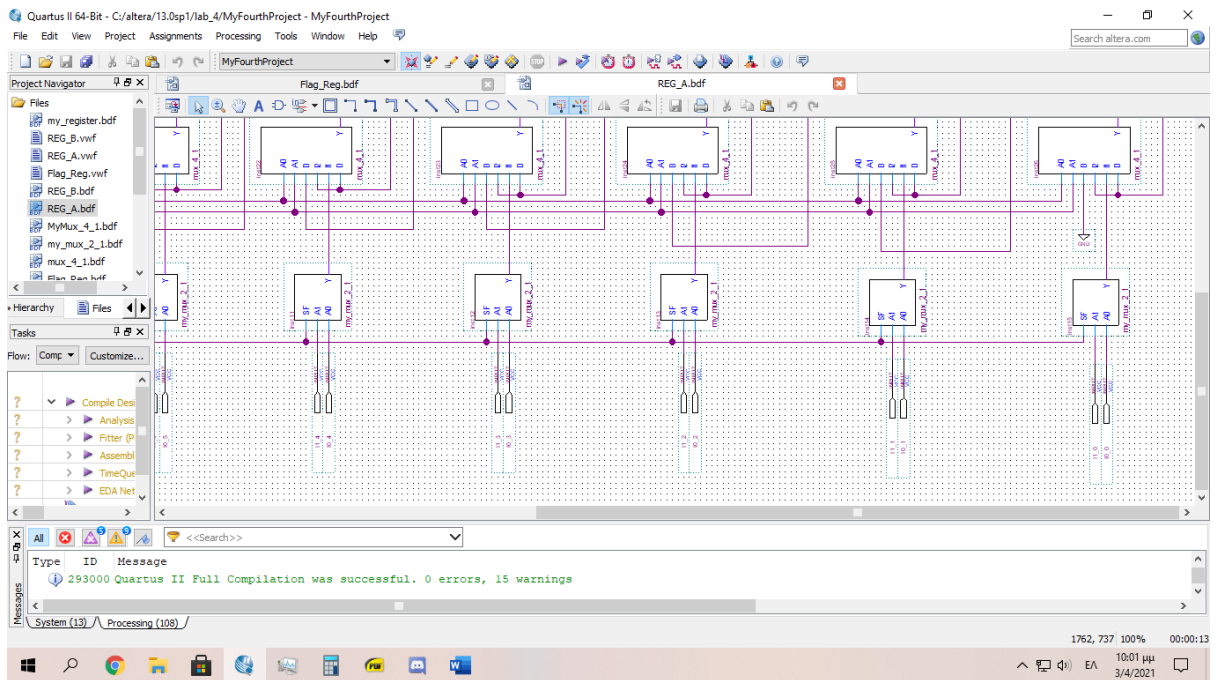
Type ID Message

293000 Quartus II Full Compilation was successful. 0 errors, 15 warnings

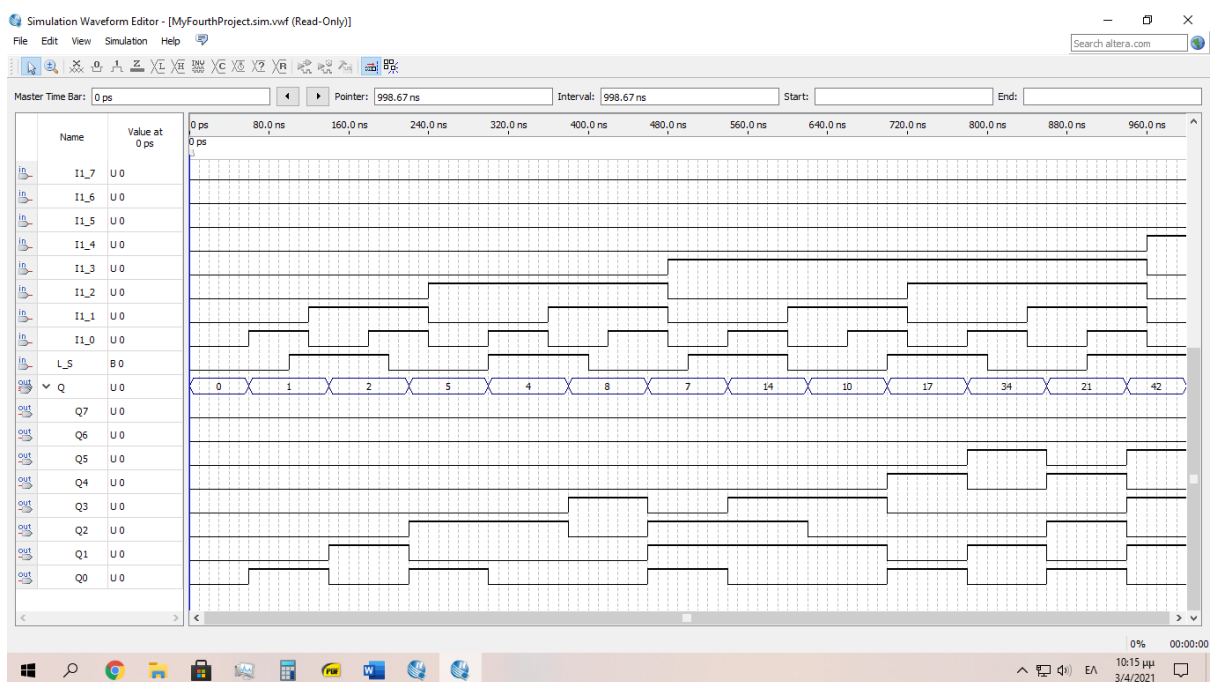
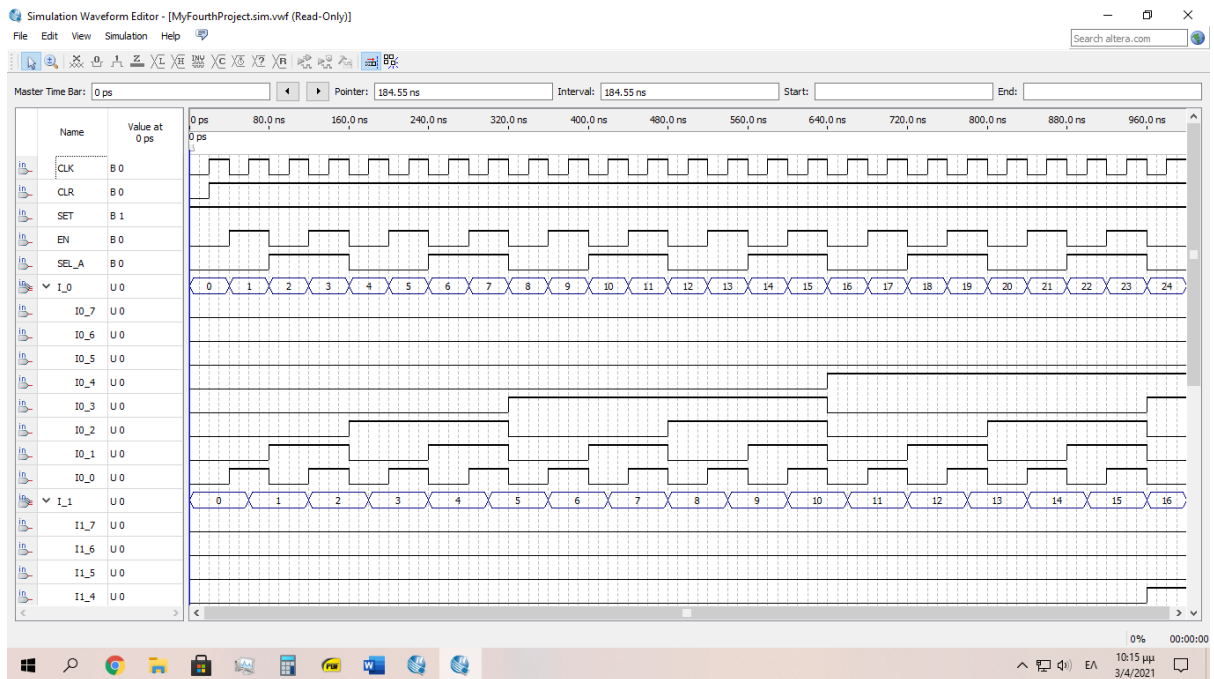
System (13) / Processing (108)

1762,737 100% 00:00:13

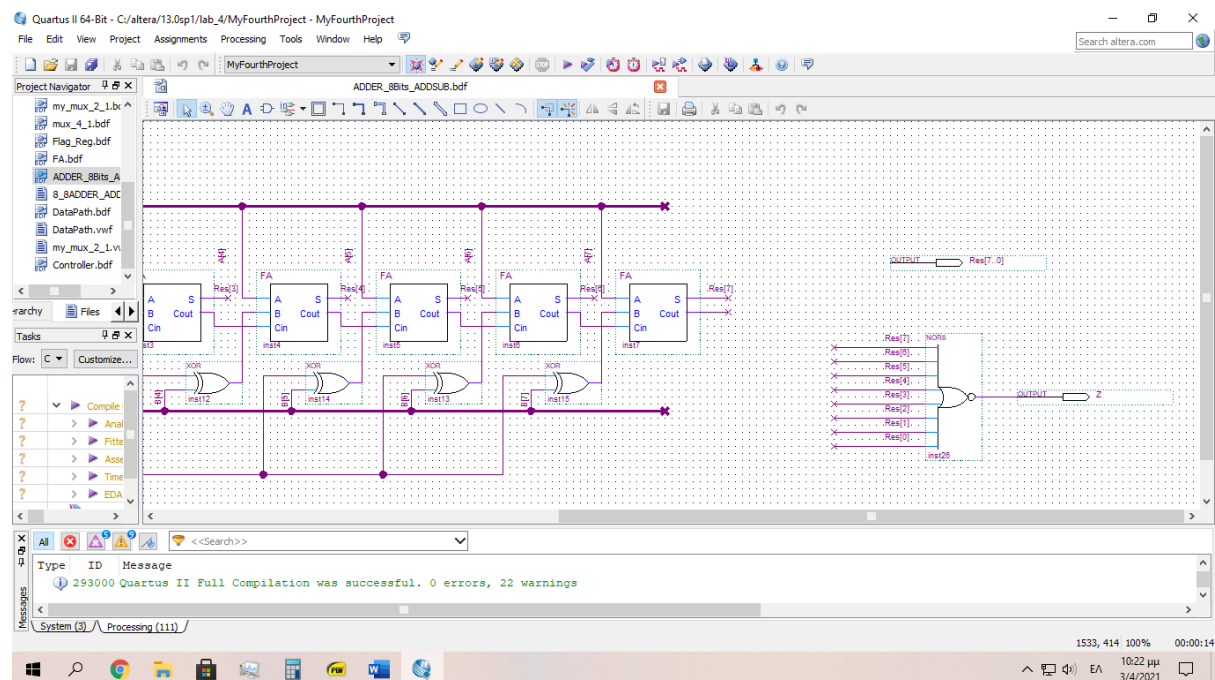
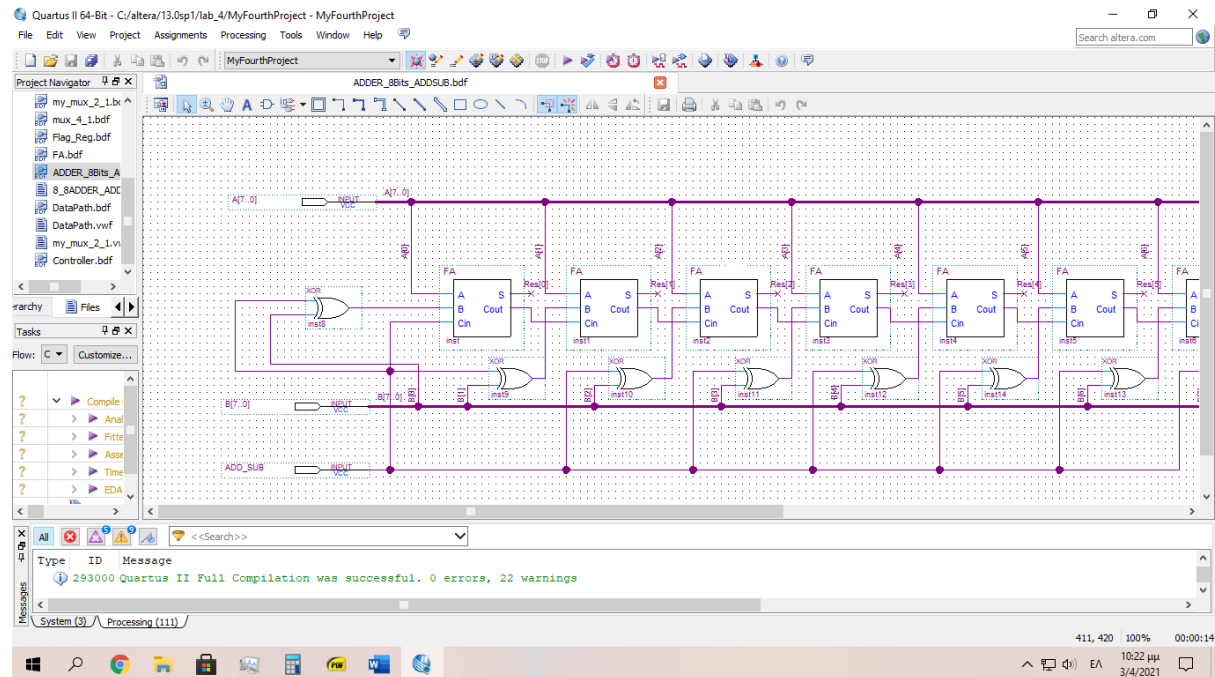
10:01 μm 3/4/2021



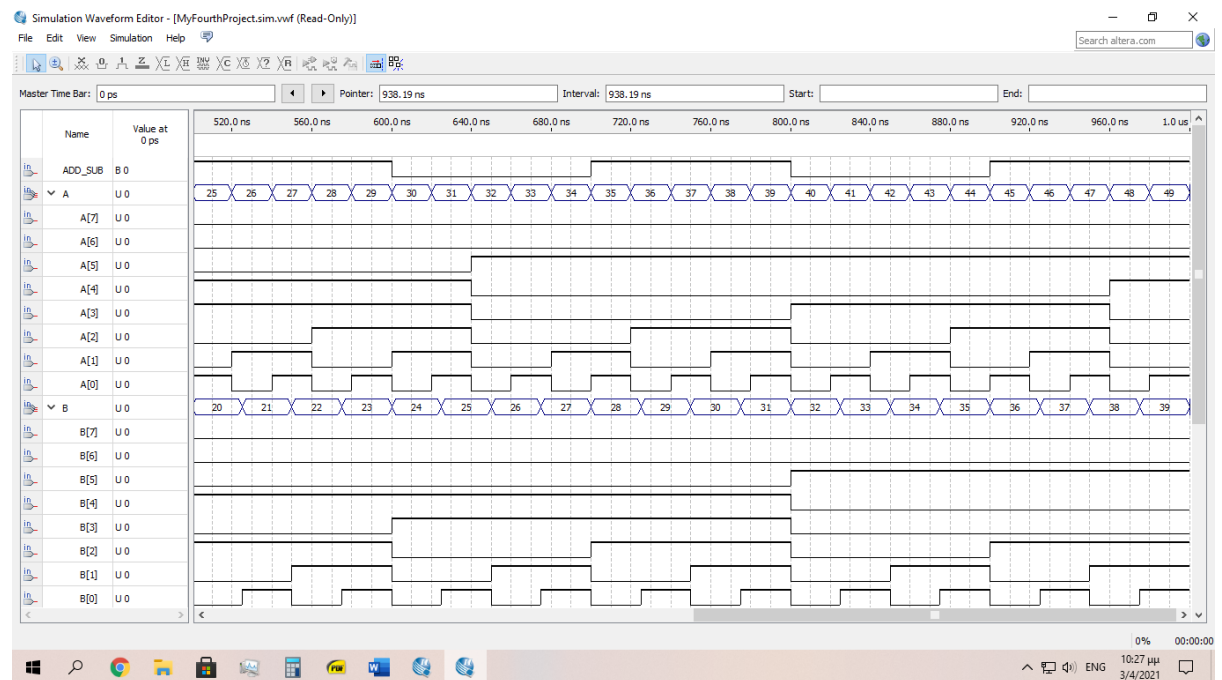
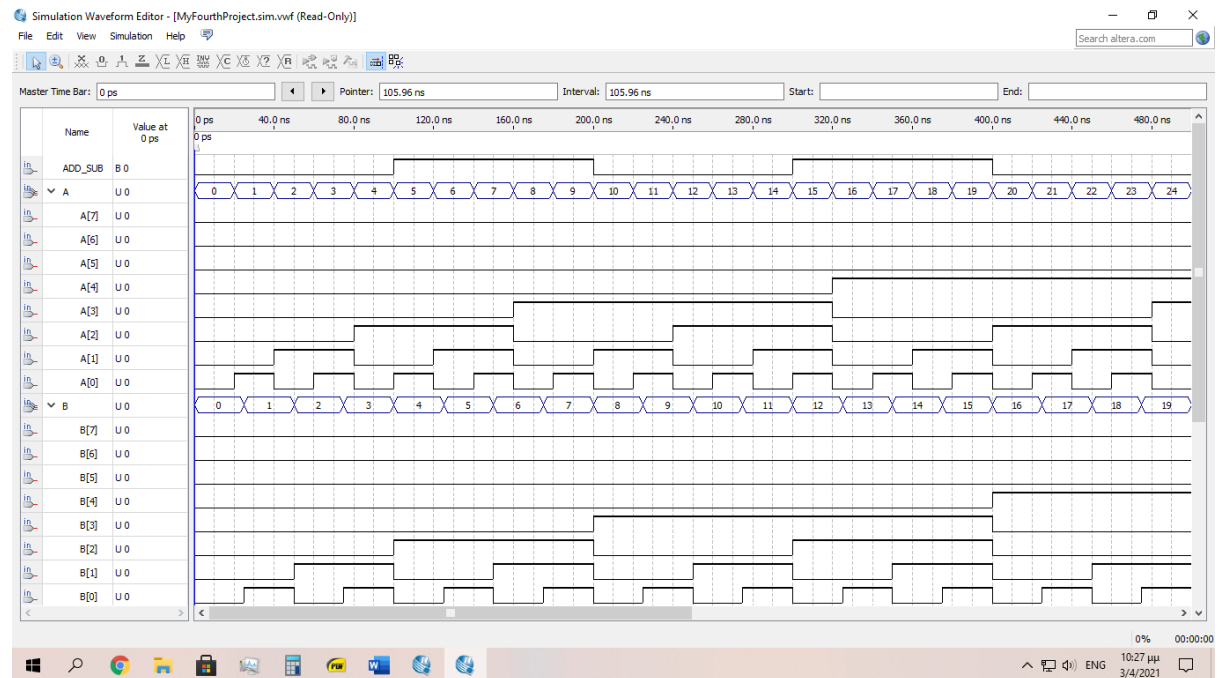
Εκτελώντας functional simulation το κύκλωμα φαίνεται να λειτουργεί σωστά, όπως φαίνεται στις παρακάτω εικόνες:

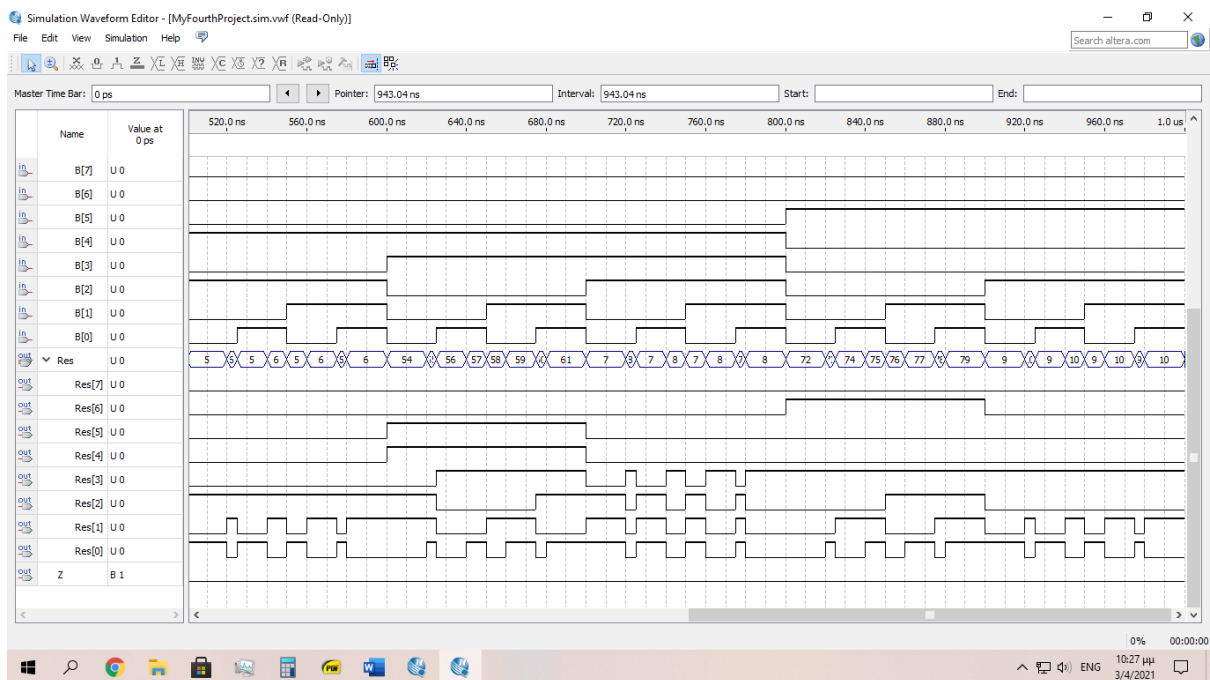
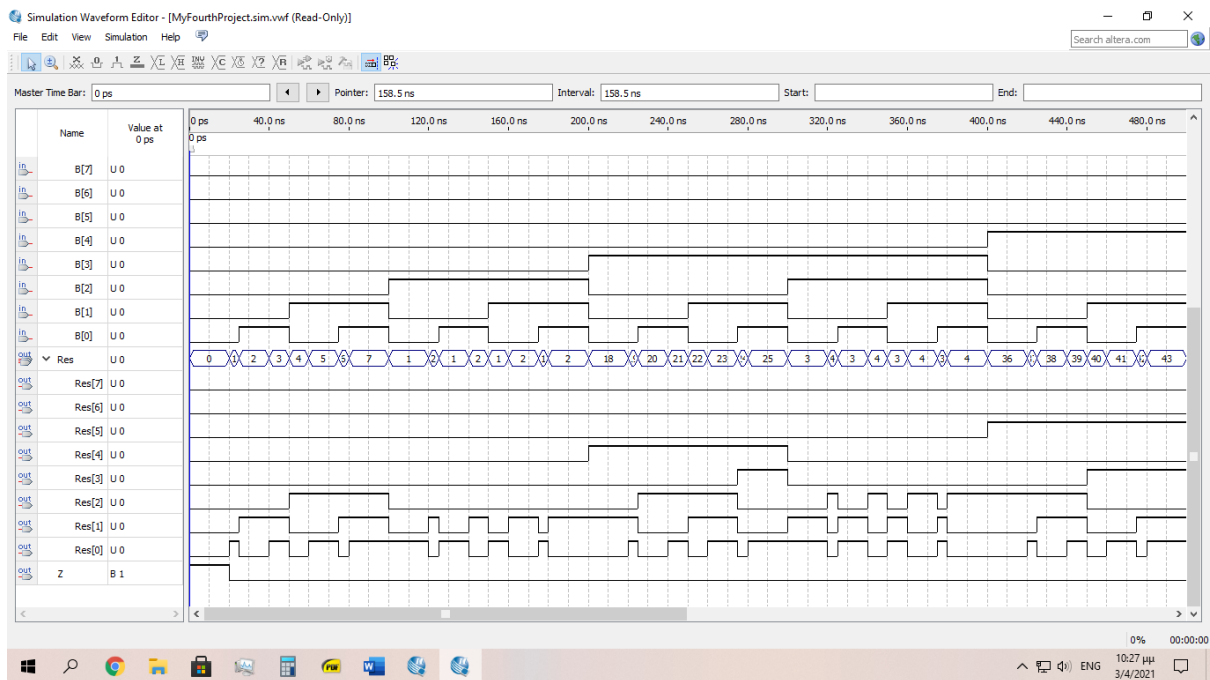


Το αθροιστή-αφαιρέτη τον υλοποιήσαμε ακριβώς όπως και σε προηγούμενη εργαστηριακή άσκηση ακολουθώντας το παράδειγμα από την ψηφιακή σχεδίαση 1 με αριθμητική συμπληρώματος ως προς 2 προσθέτοντας επιπλέον μια πορ με 8 εισόδους ώστε να βλέπουμε πότε το αποτέλεσμα είναι $Res[7..0]=0$ (όταν $A=B$ τότε $Z=1$), επιπλέον διαθέτει και μία είσοδο ADD_SUB που ουσιαστικά όταν είναι 0 τότε γίνεται πρόσθεση ενώ όταν είναι 1 τότε γίνεται αφαίρεση. Το κύκλωμα φαίνεται στις παρακάτω εικόνες με εξόδους $Res[7..0]$ και Z :



Τέλος, τοποθετήσαμε το κύκλωμα σε κουτάκι και εκτελέσαμε functional simulation που φαίνεται να λειτουργεί σωστά όπως φαίνεται στις παρακάτω εικόνες:





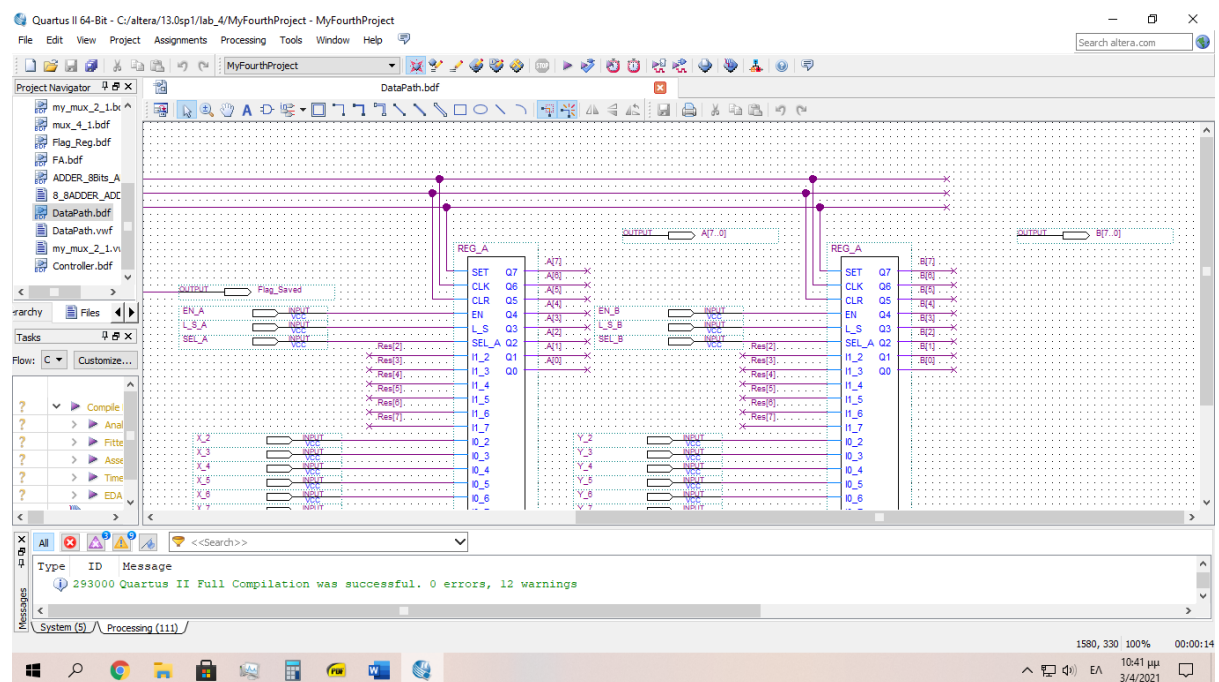
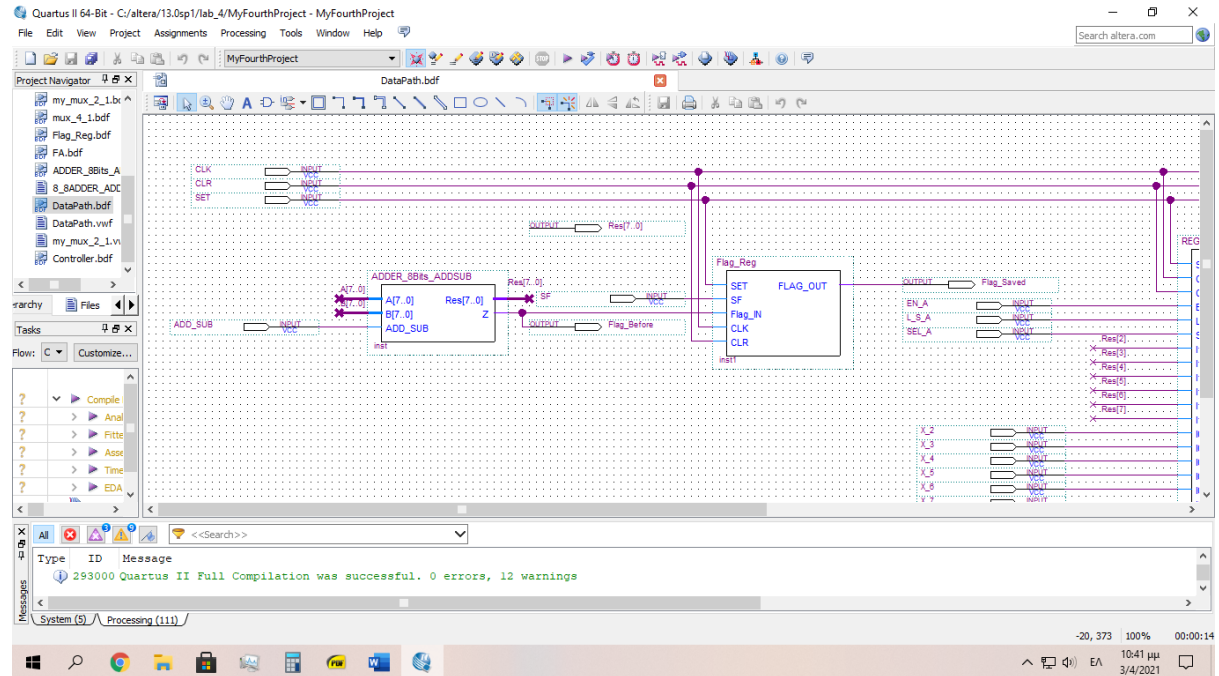
Όσον αφορά το DataPath η διασύνδεση έγινε όπως αναγράφεται στην εκφώνηση.

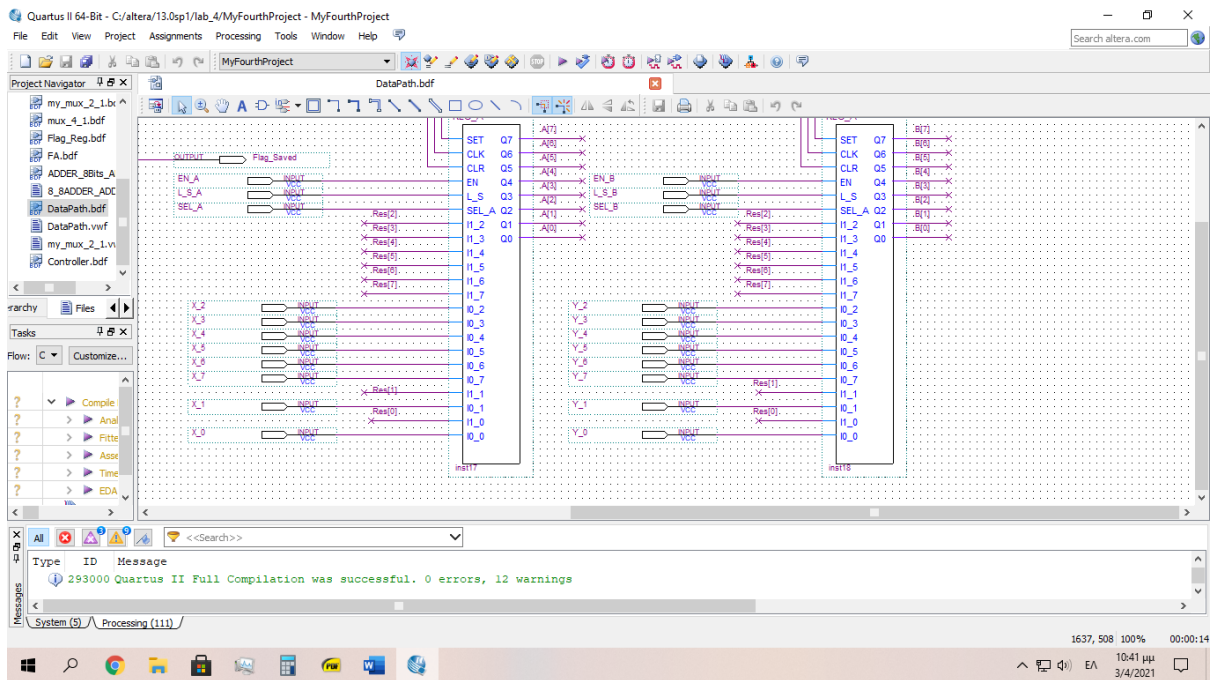
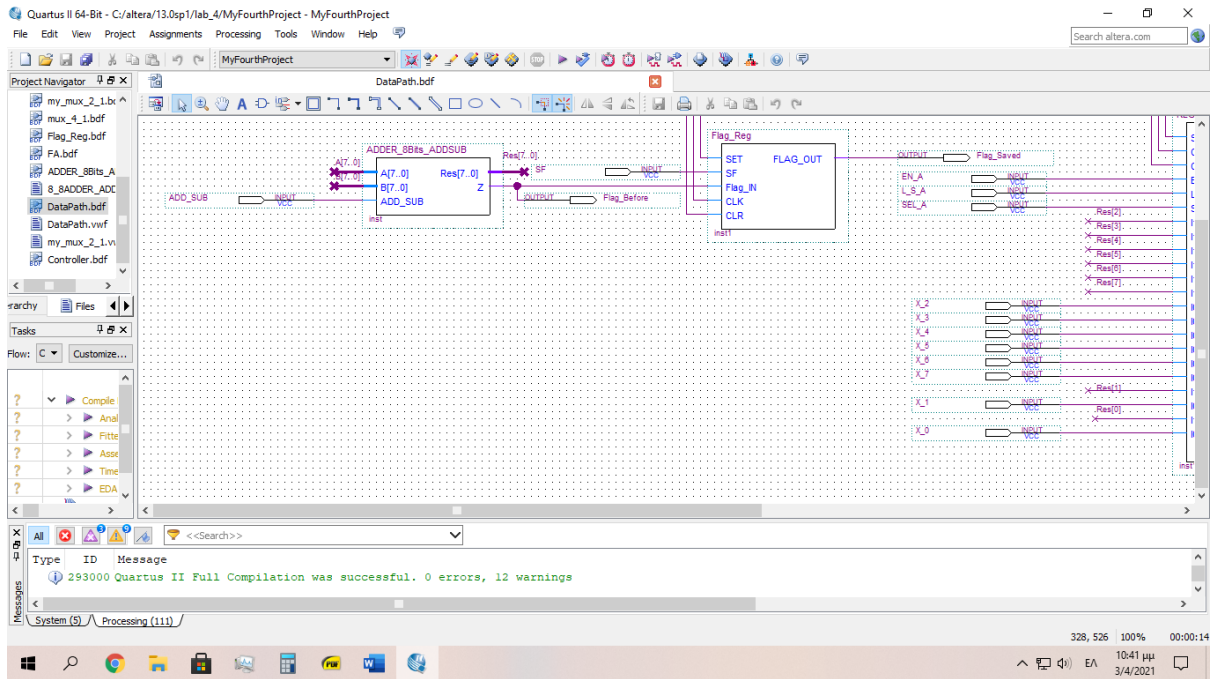
Αρχικά όλα τα σύγχρονα κυκλώματα έχουν κοινές εισόδους CLK, CLR, SET. Ο αθροιστής παίρνει ως εισόδους τις εξόδους από του δύο καταχωρητές(A και B) και την επιλογή για πρόσθεση ή για αφαίρεση και έχει δύο εξόδους, το αποτέλεσμα της πράξης που υλοποιεί και την σημαία Z που μπαίνει ως είσοδος στο κύκλωμα Flag_Reg ώστε από εκεί και μετά να την διαχειριστεί όπως αναγράφεται στην εκφώνηση.

Οι παράλληλες εισοδοι(IO_0-IO_7) των καταχωρητών A και B διασυνδέονται με εξωτερικούς διακόπτες ώστε να μπορέσει ο χρήστης να φορτώσει τις τιμές που επιθυμεί.

Όσον αφορά τις άλλες εισόδους (I1_0-I1_7) παίρνουν τιμές από την έξοδο του αθροιστή.

Τέλος, οι εισόδους EN_A, EN_B, L_S_A, L_S_B, SEL_A, SEL_B συνδέονται με εξωτερικούς διακόπτες ώστε να μπορεί να επιλεγεί η κατάλληλη λειτουργία από τον χρήστη πριν την προσθήκη του control unit. Το κύκλωμα φαίνεται στις παρακάτω εικόνες:





Εκτελώντας στην αρχή functional simulation το κύκλωμα φαίνεται να λειτουργεί σωστά όπως φαίνεται στις παρακάτω εικόνες:

