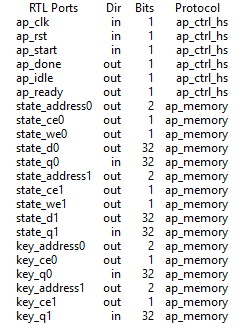
Κύκλωμα 1:

Ports:

Το κύκλωμα 1 έχει 2 ports, το key και το state και βλέπουμε από την ανάλυση, ότι το key διαβάζεται ενώ το state το χρησιμοποιήσουμε για να γράψουμε δεδομένα.

Αν ανοίξουμε τον κώδικα vhdl και επικεντρωθούμε μόνο σε αυτά που είναι 32 bit το μέγεθος του array.



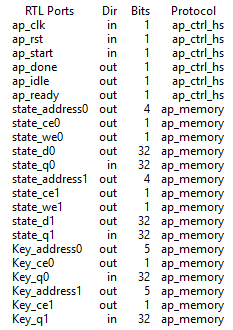
Βλέπουμε ότι στην σύνθεση όσον αφορά το state, έχει φτιάξει ports τα οποία είναι και για είσοδο και για έξοδο(state\_d0,state\_q0,state\_d1,state\_q1) σε σύγκριση με το key , που έχει φτιάξει port μόνο για είσοδο(key\_q0,key\_q1).

Κύκλωμα 2:

Ports:

Το κύκλωμα 2 έχει 2 ports, το key και το state και βλέπουμε από την ανάλυση, ότι το key διαβάζεται ενώ το state το χρησιμοποιήσουμε για να γράψουμε δεδομένα.

Αν ανοίξουμε τον κώδικα vhdl και επικεντρωθούμε μόνο σε αυτά που είναι 32 bit το μέγεθος του array.

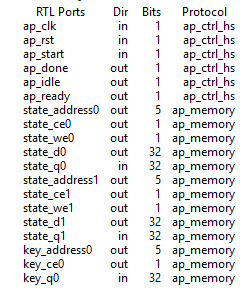


Βλέπουμε ότι στην σύνθεση όσον αφορά το state, έχει φτιάξει ports τα οποία είναι και για είσοδο και για έξοδο(state\_d0,state\_q0,state\_d1,state\_q1) σε σύγκριση με το key , που έχει φτιάξει port μόνο για είσοδο(key\_q0,key\_q1).

Κύκλωμα 3:

Το κύκλωμα 3 έχει 2 ports, το key και το statemt και βλέπουμε από την ανάλυση, ότι το key διαβάζεται ενώ το state το χρησιμοποιήσουμε για να γράψουμε δεδομένα.

Αν ανοίξουμε τον κώδικα vhdl και επικεντρωθούμε μόνο σε αυτά που είναι 32 bit το μέγεθος του array.

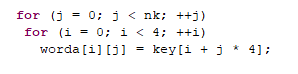


Βλέπουμε ότι στην σύνθεση όσον αφορά το state, έχει φτιάξει ports τα οποία είναι και για είσοδο και για έξοδο(state\_d0,state\_q0,state\_d1,state\_q1) σε σύγκριση με το key , που έχει φτιάξει port μόνο για είσοδο(key\_q0).

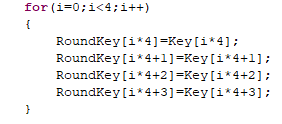
Όπως βλέπουμε από τα παραπάνω, το κυκλώματα έχουν παρόμοια εισόδους και εξόδους με εξαίρεση το 3ο και αυτό γίνεται επειδή και τα τρία κυκλώματα χρησιμοποιούν ορίσματα τύπο word τα οποία μεταφράζονται σε STD\_LOGIC\_VECTOR των 32 bit.

Η διαφορά του 3ου με τα υπόλοιπα είναι ότι αποθηκεύει το key σε ένα δυσδιάστατο πίνακα και όχι σε ένα μονοδιάστατο όπως το κύκλωμα 1 και 2 και για αυτόν τον λόγο δεν φτιάχνει και άλλο port στο 3ο για να χωρέσει το κλειδί.

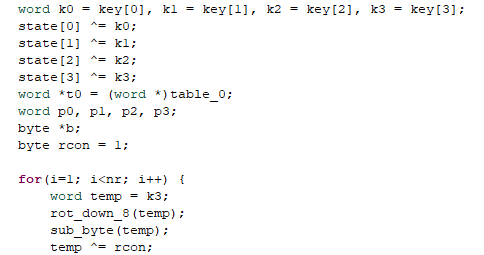
3ο κύκλωμα:



2ο κύκλωμα



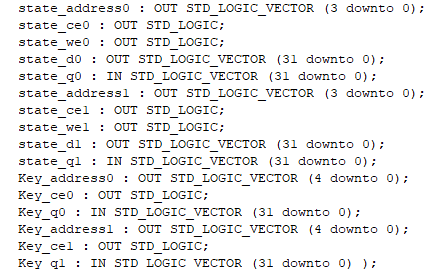
1ο κύκλωμα:



Εκτός αυτού τα ports τα φτιάχνει σύμφωνα με τις ανάγκες των συναρτήσεων. Πιο συγκεκριμένα, όσον αφορά το 2ο κύκλωμα η κύρια συνάρτηση **Cipher** χρειάζεται ως ορίσματα το state και το key.



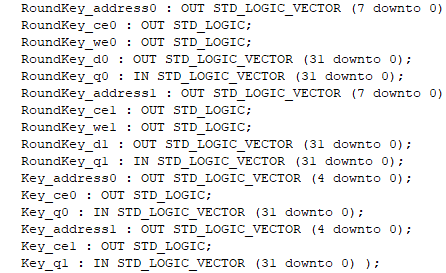
Και βλέπουμε ότι έχει φτιάξει τα ανάλογα:



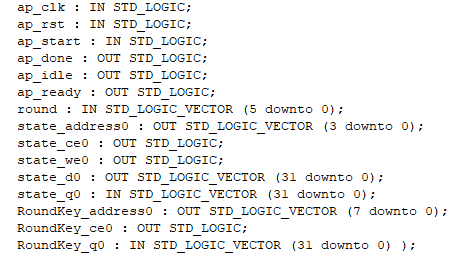
Στην συνέχεια, έχουμε την συνάρτηση **KeyExpansion** λαμβάνει ως ορίσματα το RoundKey,Key και Rcon. Τα οποία όπως βλέπουμε, έχουν μετατραπεί σε ports τα RoundKey και Key καθώς το Rcon ενώ το Rcon σε μνήμη ROM. Ο λόγος που μετατράπηκε σε auto Rom είναι επειδή, υπάρχει στην συνάρτηση το Cipher, που είναι η κύρια συνάρτηση, και αποτελεί έναν πίνακα.

Αν ήταν σε μια άλλη συνάρτηση θα είχε μετατραπεί σε block Rom, καθώς κάθε η συνάρτηση έχει το δικό της Block.

INFO: [RTMG 210-279] Implementing memory 'KeyExpansion\_sbox\_rom' using block ROMs.



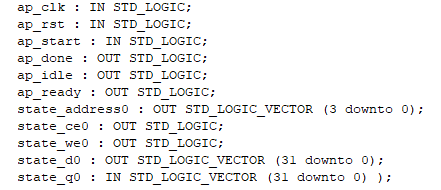
Όσον αφορά την συνάρτηση AddRoundKey, λαμβάνει ως ορίσματα το round, το state, και RoundKey.



Όπως βλέπουμε παραπάνω το round και το roundKey το λαμβάνει ως είσοδο, ενώ το state για είσοδο και έξοδο.

Η συνάρτηση SubBytes λαμβάνει ως ορίσματα από τι βλέπουμε μόνο το state, το οποίο στην αρχή το διαβάζει και μετά γράφει. Άρα συμπεραίνουμε ότι θα έχουμε μια μεταβλητή STD\_LOGIC\_VECTOR για IN και OUT.





Ομοίως και η **ShiftRows** εφόσον διαβάζει και γράφει στο state



Θα είχε το ίδιο, αλλά ζητά ακριβώς τα ίδια ports, το εργαλείο δεν χρειάζεται να κάνει σύνθεση και για αυτά, για εξοικονόμηση χώρου.

Ομοίως και η **MixColumns**:



Με τον ίδιο τρόπο γίνονται και τα άλλα δύο κυκλώματα.