



# Πολλαπλασιασμός 3 Πινάκων

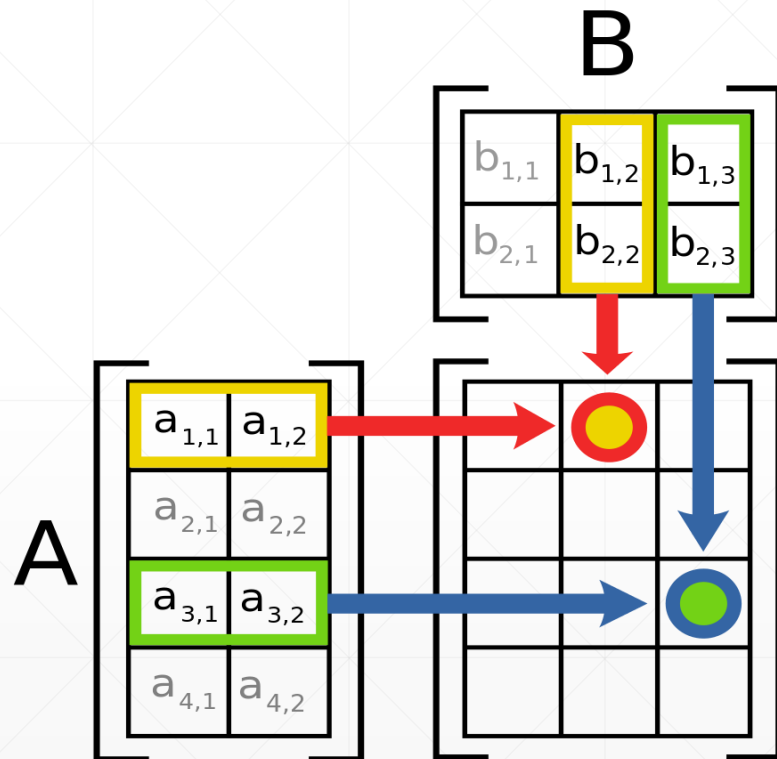
---

Συστήματα σε Ολοκληρωμένα Κυκλώματα.  
Εργασία εξαμήνου: Κωνσταντίνος Τσικρίκης (ΑΜ:60598)

# Περιεχόμενα παρουσίασης

- Γενικά για τον πολλαπλασιασμό πινάκων
- Ανάλυση κώδικα
- HLS optimizations

# Πολλαπλασιασμός πινάκων



- Πρέπει η 2<sup>η</sup> διάσταση του 1<sup>ου</sup> πίνακα να είναι ίδια με την 1<sup>η</sup> διάσταση του 1<sup>ου</sup> πίνακα.
- $A_{N,M} * B_{M,L} = C_{N,L}$
- Δεν υπάρχει απευθείας τρόπος να κάνουμε τριπλό πολλαπλασιασμό, γι' αυτό χρησιμοποιούμε την επιμεριστική ιδιότητα του πολλαπλασιασμού
- $(A*B)*C$

# Ανάλυση κώδικα (γενικά)

- Χρησιμοποιήσαμε τις βιβλιοθήκες: `iostream`, `cstdlib`, `ctime`, `mc_scverify.h`
- Η χρήση τους έγινε για την εκτύπωση μηνυμάτων στο `terminal`, για την δημιουργία τυχαίων πινάκων για την επιβεβαίωση της ορθής λειτουργίας της συνάρτησης.
- Η `mc_scverify.h` είναι για το RTL verification.
- Οι διαστάσεις των πινάκων ορίζονται σαν στατικές μεταβλητές αφού κατά τη διάρκεια της συνάρτησης δεν αλλάζουν.
- Οι πίνακες έχουν `short` τύπου στοιχεία και γεμίζουν τυχαία (στα πλαίσια του `testbench C++`) με την συνάρτηση `rand()` και με τα το `%` ορίζουμε το όριο του ανώτερου τυχαίου αριθμού.

# Ανάλυση κώδικα (βασική συνάρτηση)

- Η συνάρτηση είναι τύπου void.
- Δέχεται σαν εισόδου 4 πίνακες, τους 3 που θέλουμε να πολλαπλασιάσουμε με σωστή σειρά και ένα μηδενικό που θα έχει τις διαστάσεις του αποτελέσματος.
- Ο υπολογισμός γίνεται με δύο τριπλές for loops, όπου στην πρώτη αποθηκεύουμε το αποτέλεσμα σε ένα τοπικό buffer που δημιουργούμε, και στη δεύτερη κάνουμε τον τελικό υπολογισμό.

# HLS optimizations

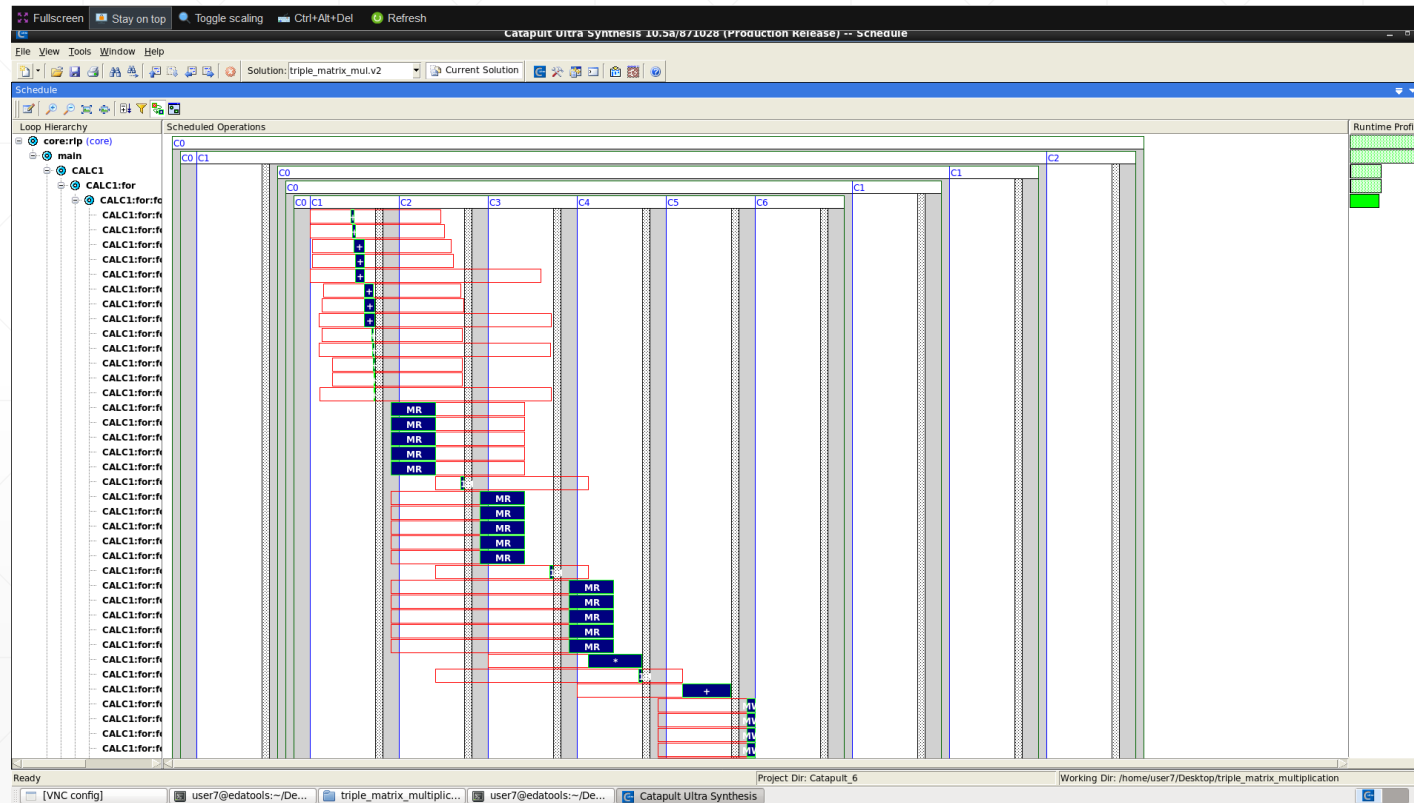
Τα HLS optimizations που δοκιμάστηκαν στο Catapult είναι:

- Buffering στον ενδιάμεσο πίνακα του πρώτου πολλαπλασιασμού.
- Loop Unrolling στο ένα από τα δύο loops , αλλά και στα δύο loops (ομάδες for loops).
- Pipeline στα blocks των for loops.

# HLS optimizations

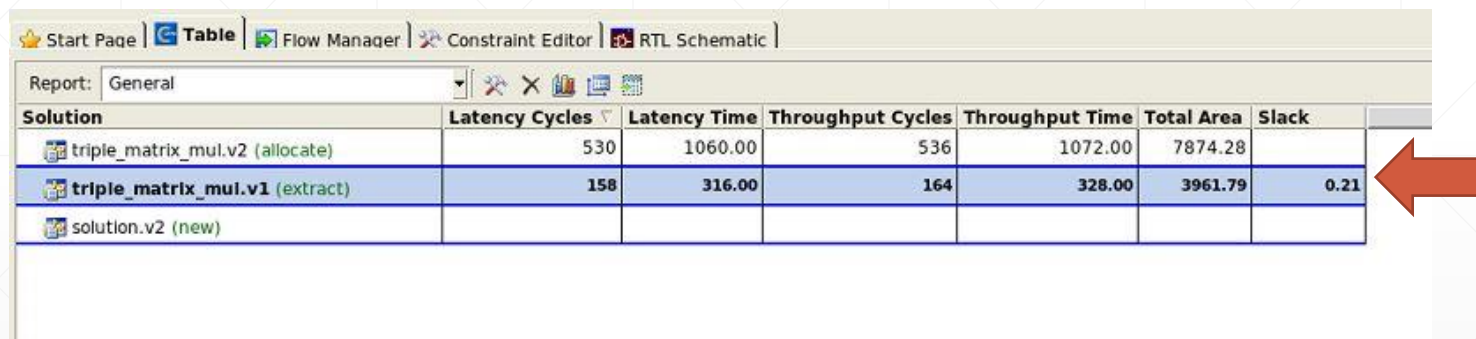
- Το clock ορίζεται στα 500MHz.
- Η μνήμη είναι μια nandgate 45nm dualport
- Η αρχική σχεδίαση δεν έχει καθόλου buffer ή οπότε το scheduling γίνεται πολύ περίπλοκο με πολλές προσπελάσεις της μνήμης, επίσης είναι πανομοιότυπο αφού και ο ενδιάμεσος πίνακας του  $A*B$  είναι μέρος της μνήμης.
- Τον ενδιάμεσο πίνακα εδώ τον βάλαμε σαν είσοδο στη συνάρτηση για να τον καταλάβει το εργαλείο σαν μέρος της μνήμης.

# HLS optimizations





# HLS optimizations



Start Page | Table | Flow Manager | Constraint Editor | RTL Schematic

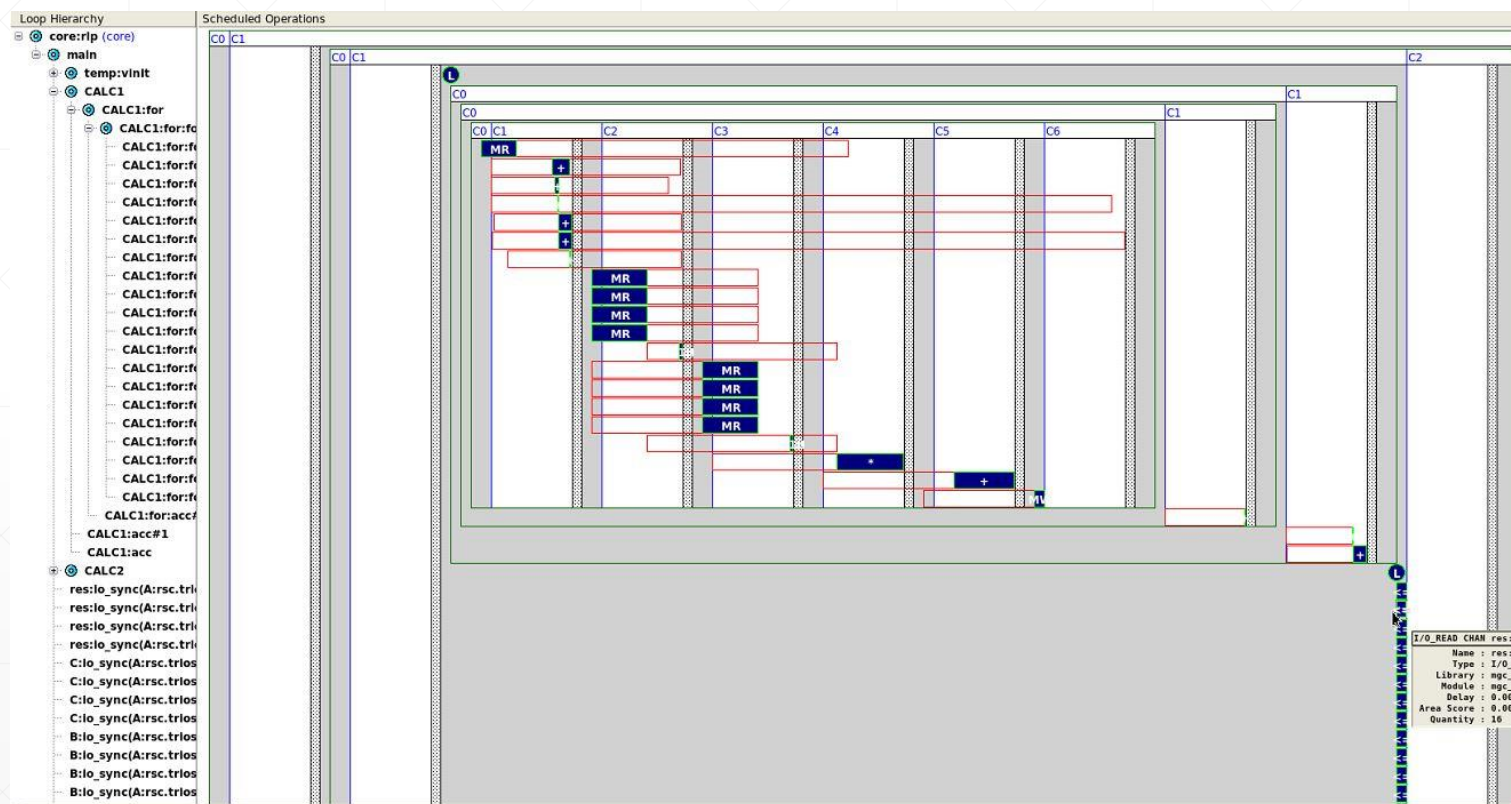
Report: General

Solution	Latency Cycles	Latency Time	Throughput Cycles	Throughput Time	Total Area	Slack
triple_matrix_mul.v2 (allocate)	530	1060.00	536	1072.00	7874.28	
<b>triple_matrix_mul.v1 (extract)</b>	<b>158</b>	<b>316.00</b>	<b>164</b>	<b>328.00</b>	<b>3961.79</b>	<b>0.21</b>
solution.v2 (new)						

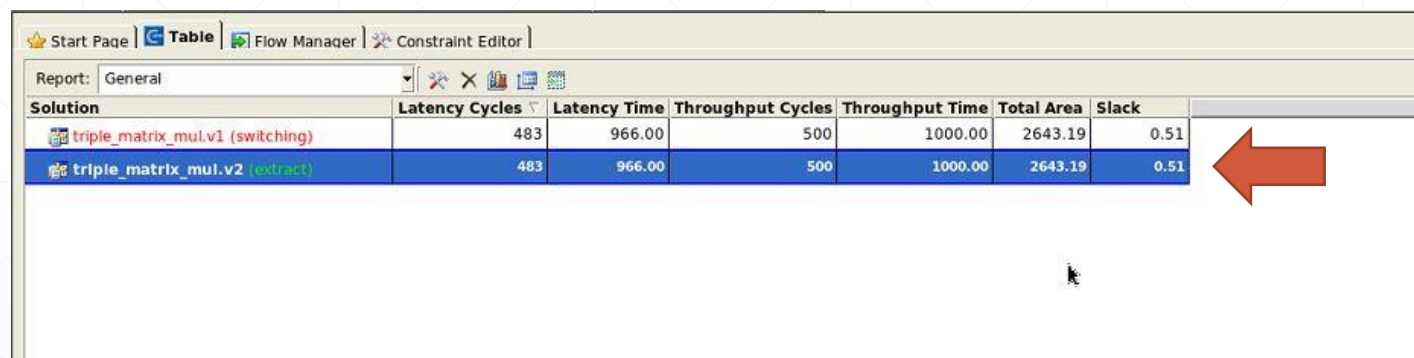
# HLS optimizations (buffering)

- Το 1<sup>ο</sup> optimization που κάνουμε είναι να αφαιρέσουμε τον ενδιάμεσο πίνακα από τη μνήμη και να το κάνουμε buffer, δηλαδή απλό τοπικό πίνακα στη συνάρτησή μας.
- Ο ενδιάμεσος πίνακας στο schedule θα είναι ένα register file 45nm

# HLS optimizations (buffering)



# HLS optimizations (buffering)



The screenshot shows the 'Table' tab in the Vivado report window. The report is titled 'General'. The table compares two solutions: 'triple\_matrix\_mul.v1 (switching)' and 'triple\_matrix\_mul.v2 (extract)'. The columns are: Solution, Latency Cycles, Latency Time, Throughput Cycles, Throughput Time, Total Area, and Slack. The 'Slack' column is highlighted with a red arrow, indicating the difference between the two solutions.

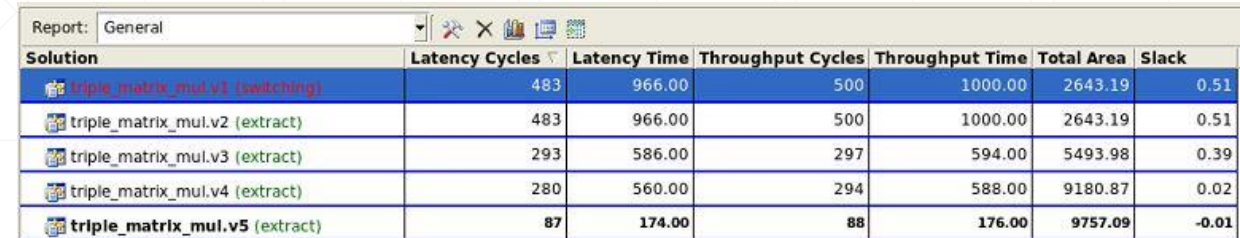
Solution	Latency Cycles	Latency Time	Throughput Cycles	Throughput Time	Total Area	Slack
triple_matrix_mul.v1 (switching)	483	966.00	500	1000.00	2643.19	0.51
triple_matrix_mul.v2 (extract)	483	966.00	500	1000.00	2643.19	0.51

# HLS optimizations (loop unrolling)

- Επειδή έχω δυο ομάδες από επαναλήψεις (blocks of loops) θα κάνουμε πρώτα loop unrolling στην 1<sup>η</sup> ομάδα και μετά στην άλλη.
- Εδώ το schedule που παράγει το εργαλείο είναι μεγάλο και δεν φαίνεται καλά σε screenshot όμως, ομοίως και στην δεύτερη περίπτωση και τα αποτελέσματα θα τα δούμε στις μετρικές τιμές και θα συγκρίνουμε τα νούμερα και για τις 3 περιπτώσεις.

# HLS optimizations (loop unrolling)

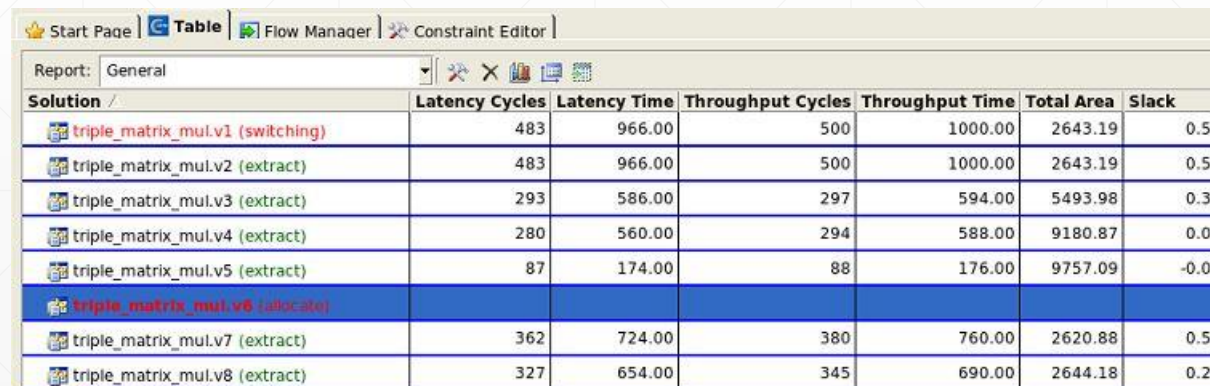
- Στο v2 έχουμε την απλή έκδοση μόνο με το buffering.
- Στο v3 και v4 έχουμε τα μεμονωμένα unrolling και τη 1<sup>η</sup> και 2<sup>η</sup> ομάδα for loops αντίστοιχα.
- Στο v5 έχουμε και τις 2 ομάδες με loop unrolling.
- Παρατηρούμε ότι όσο κάνουμε loop unrolling πέφτουν οι κύκλοι.
- Δεν παρατηρείται μεγάλη διαφορά στα loop unroll και η διαφορά είναι η αποφυγή της προσπέλασης της μνήμης.



Solution	Latency Cycles	Latency Time	Throughput Cycles	Throughput Time	Total Area	Slack
triple_matrix_mul.v1 (switching)	483	966.00	500	1000.00	2643.19	0.51
triple_matrix_mul.v2 (extract)	483	966.00	500	1000.00	2643.19	0.51
triple_matrix_mul.v3 (extract)	293	586.00	297	594.00	5493.98	0.39
triple_matrix_mul.v4 (extract)	280	560.00	294	588.00	9180.87	0.02
triple_matrix_mul.v5 (extract)	87	174.00	88	176.00	9757.09	-0.01

# HLS optimizations (pipeline)

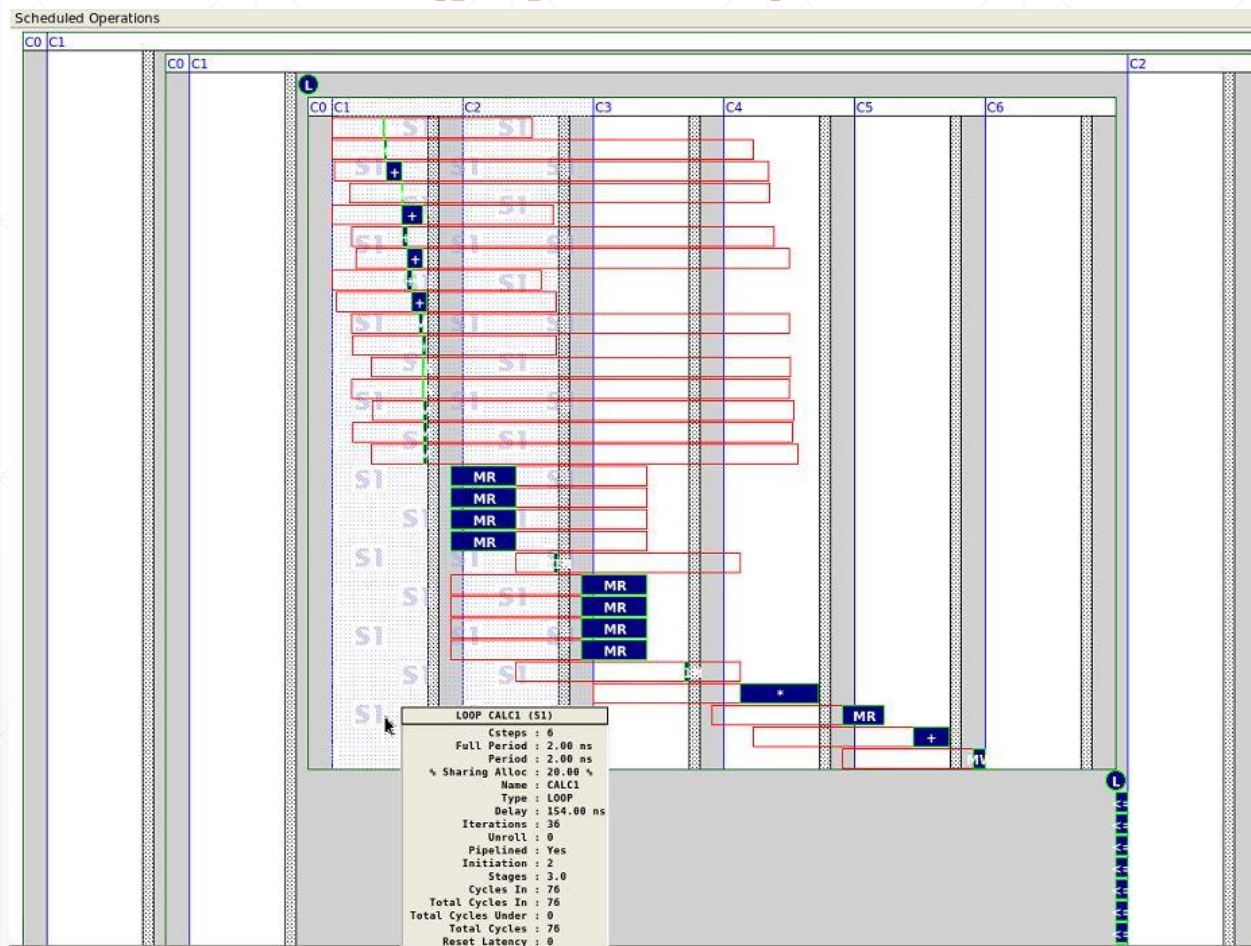
- Στο πρόβλημα του τριπλού πολλαπλασιασμού πινάκων το καλύτερο initiation interval που μπορεί να επιτευχθεί είναι  $II = 2$ .
- Δεν είναι δυνατή η επίτευξη  $II = 1$  διότι κάνουμε προσπάθεια σε 2 πίνακες που είναι αποθηκευμένοι στην ίδια μνήμη που έχει μία πόρτα ανάγνωσης.
- Στην προσπάθεια n8 είναι το pipeline με  $II = 2$ , και τα αποτελέσματα είναι βελτιωμένα από την n2 που δεν εφαρμόσαμε κανένα pipeline.



Solution /	Latency Cycles	Latency Time	Throughput Cycles	Throughput Time	Total Area	Slack
triple_matrix_mul.v1 (switching)	483	966.00	500	1000.00	2643.19	0.51
triple_matrix_mul.v2 (extract)	483	966.00	500	1000.00	2643.19	0.51
triple_matrix_mul.v3 (extract)	293	586.00	297	594.00	5493.98	0.39
triple_matrix_mul.v4 (extract)	280	560.00	294	588.00	9180.87	0.02
triple_matrix_mul.v5 (extract)	87	174.00	88	176.00	9757.09	-0.01
triple_matrix_mul.v6 (allocate)						
triple_matrix_mul.v7 (extract)	362	724.00	380	760.00	2620.88	0.54
triple_matrix_mul.v8 (extract)	327	654.00	345	690.00	2644.18	0.23



# HLS optimizations (pipeline)





# References

- Διαφάνειες μαθήματος από e-class:  
<https://eclass.duth.gr/modules/document/?course=TMA463>
- [https://en.wikipedia.org/wiki/Matrix\\_multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication) (εικόνα της διαφάνειας 3)