Ερώτηση 1: Σας δίνεται ο ακόλουθος συνθέσιμος κώδικας (θεωρείστε ότι ο A, B, C είναι **32 bit** προσημασμένοι ακέραιοι, είναι αποθηκευμένοι στη BRAM και αρχικοποιημένοι).

```
for (int i = 0; i < dim1; i++){
        for (int j = 0; j < dim2; j++){
                for (int k = 0; k < dim3; k++){
                        C[i][j][k] = (A[i][j][k] * B[i][j][k]) << 2;
                        }
         }
}
```

Έστω ότι η πράξη «*C[i][j][k] = (A[i][j][k] * B[i][j][k]) << 2;*» χρειάζεται **t** κύκλους να εκτελεστεί στο Vivado HLS και θεωρείστε ότι έχετε όσα FPGA resources θέλετε και απαντήστε στα ακόλουθα ερωτήματα.

α) Ποιος είναι ο συνολικός χρόνος εκτέλεσης του παραπάνω κώδικα σε κύκλους ρολογιού;

β) Εφαρμόστε μόνο **unroll** optimization στο παραπάνω κώδικα και αναφέρετε σε ποια/ες γραμμή/ες θα μπει το directive. i) Ποιος είναι ο βέλτιστος συνολικός χρόνος που μπορείτε να πετύχετε σε συνάρτηση με το **t** και γιατί; (στην απάντηση σας περιγράψτε όλα τα optimizations που πρέπει να γίνουν καθώς και το partition στις BRAM). ii) Ποια είναι η επιτάχυνση (speed-up) σχετικά με την αρχική υλοποίηση;

γ) Αν η αρχική υλοποίηση χρειαζόταν ***d1*** DSPs, πόσα DSPs παραπάνω χρειάζεται η υλοποίηση σας εφαρμόζοντας **unroll** optimization και γιατί;

δ) Γράψτε το κώδικα που απαιτείται για τους πίνακες A και B (που θα είναι πριν το κώδικα που σας δίνεται) για να μεταφερθούν τα δεδομένα από τη DRAM στη BRAM με βέλτιστο τρόπο λαμβάνοντας υπόψη τους περιορισμούς της DRAM (δε μας ενδιαφέρει η αντίστροφη διαδικασία - από τη BRAM στη DRAM). Ποιος είναι ο χρόνος (σε κύκλους ρολογιού) που απαιτείται για τη μεταφορά των δεδομένων αυτών;

α) Εφόσον, έχουμε 3 nested loops και η πράξη απαιτεί t κύκλους τότε ο συνολικό χρόνος εκτέλεσης του παραπάνω κώδικα είναι:

dim1*dim2*dim3*t

β)

i) Αν μπορούμε στον παραπάνω κώδικα να εφαρμόσουμε μόνο unroll optimization τότε το βέλτιστο θα ήταν να τοποθετήσουμε το παρακάτω pragma ακριβώς μετά το 3^{ο} loop:

#pragma HLS unroll factor=dim3 ή σκέτο #pragma HLS unroll(θα κάνει fully unrolled το 3^{ο} loop)

Ο βέλτιστος συνολικός χρόνος που μπορούμε να επιτύχουμε είναι: dim1*dim2*t.

Φυσικά, για να συμβεί αυτό θα πρέπει να έχουμε κάνει και σωστό partionioning των BRAMs, έτσι ώστε να μπορούμε να διαβάσουμε παράλληλα dim3 στοιχεία. Αυτό θα γίνει κάνοντας χρήση των παρακάτω pragmas:

#pragma HLS ARRAY_PARTITION variable=A_in cyclic factor=dim3/2 dim=3

#pragma HLS ARRAY_PARTITION variable=B_in cyclic factor=dim3/2 dim=3

Κάνουμε partition τις BRAMs με factor ίσο με dim3/2 διότι έχουμε dual-port DRAM.

ii) Η επιτάχυνση που θα έχουμε σε σχέση με την αρχική υλοποίηση είναι dim3.

γ) Αν η αρχική υλοποίηση χρειαζόταν d1 DSPs τότε εφαρμόζοντας unroll optimization θα χρειαστεί d1*dim3 DSPs, διότι θα πρέπει να γίνει επεξεργασία dim3 στοιχείων παράλληλα.

δ)

```
int23_t A[dim1][dim2][dim3];

int23_t B[dim1][dim2][dim3];

#pragma HLS ARRAY_PARTITION variable=A cyclic factor=dim3/2 dim=3

#pragma HLS ARRAY_PARTITION variable=B cyclic factor=dim3/2 dim=3

for (int i=0; i<dim1; i++){

#pragma HLS loop_tripcount min=dim1 max=dim1

for (int j=0; j<dim2; j++){

#pragma HLS loop_tripcount min=dim2 max=dim2

for (int k=0; k<dim3; k++){

#pragma HLS loop_tripcount min=dim2 max=dim2

#pragma HLS PIPELINE II=1

A[i][j][k]= A_DRAM[i][j][k];

B[i][j][k]= B_DRAM[i][j][k];

} } }
```

Εφόσον, έχουμε τοποθετήσει το pipeline pragma ο χρόνος σε κύκλους ρολογιού θα είναι: dim1*dim2*dim3+ Iniatiation Interval μεταφοράς δεδομένων από την μνήμη DRAM. Στο εργαστήριο νομίζω μας είναι πει ότι στο design μας είναι περίπου 3 κύκλοι.