Σχεδίαση συστημάτων υλικού-λογισμικού
Εργαστήριο 1
Ναπολέων Παπουτσάκης 9170
Γιώργος – Εφραίμ Παππάς 9124

Ερώτημα 1

Για το εργαστήριο δημιουργήσαμε 3 αρχεία. file.cpp, tb.cpp, file.h. Στο file.h γίνονται οι αρχικοποιήσεις των διαστάσεων του πίνακα (lm,ln,lp). Το αρχείο file.cpp περιέχει την υλοποίηση της mult_hw που πρόκειται να τρέξει στο υλικό. Στο αρχείο tb.cpp έχουμε το testbench, με τη βοήθεια του οποίου επιβεβαιώνουμε τη σωστή λειτουργία του κώδικα. Αυτό γίνεται με τη δημιουργία της συνάρτησης mult_sw, η οποία έχει την ίδια λειτουργικότητα με τη mult_hw, αλλά τρέχει στη cpu του υπολογιστή μας. Έτσι με τη βοήθεια του testbench, παράγουμε με τη συνάρτηση mult_sw το αποτέλεσμα που θέλουμε και το συγκρίνουμε με αυτό που παράχθηκε απο τη mult_hw και αν τα δύο αυτά αποτελέσματα είναι ίδια, τότε θεωρούμε ότι η υλοποίηση της mult_hw είναι σωστή.

Τα αρχεία file.cpp, tb.cpp, file.h είναι ανεβασμένα στο link που ακολουθεί:

https://github.com/gpappasv/Hardware-Software-co-design-lab1

Ερώτημα 2

Estimated clock period: 3.691ns
Worst case latency: 33686017 cycles – 0.337 sec
Number of DSP48E used: 1
Number of BRAMs used: 0
Number of FFs used: 117
Number of LUTs used: 189

Ερώτημα 3

Total execution time: 336860355 ns
Min latency: 33686017
Avg. latency: 33686017
Max latency: 33686017

Ερώτημα 4

Τα directives που χρησιμοποιήθηκαν ήταν:

```
#pragma HLS ARRAY_PARTITION variable=BRAM_A complete dim=2
#pragma HLS ARRAY_PARTITION variable=BRAM_B complete dim=1
```

BRAM_A και BRAM_B είναι 2 πίνακες που δηλώθηκαν μέσα στη συνάρτηση mult_hw για να περάσουμε τα δεδομένα απο τους πίνακες που είχαμε στην είσοδο, σε bram ώστε με τη βοήθεια του ARRAY_PARTITION να αποφύγουμε το πρόβλημα των περιορισμένων port της dram και να πετύχουμε παραλληλία στους πολλαπλασιασμούς των στοιχείων που περιέχουν.

Σχεδίαση συστημάτων υλικού-λογισμικού
Εργαστήριο 1
Ναπολέων Παπουτσάκης 9170
Γιώργος – Εφραίμ Παππάς 9124

Στον BRAM_A κάναμε array_partition ως προς τις στήλες και στον BRAM_B ως προς τις γραμμές, καθώς ο πολλαπλασιασμός πινάκων γίνεται πολλαπλασιάζοντας τα στοιχεία της στήλης του A με τα στοιχεία της γραμμής B (θεωρούμε A[n][m] n->γραμμές, m->στήλες)

Επίσης δημιουργήθηκαν 2 συναρτήσεις copy1 και copy2 για να περάσουμε τα δεδομένα των πινάκων εισόδου (A,B που είναι σε DRAM) στους πίνακες BRAM_A, BRAM_B (στο screenshot φαίνεται η copy1). Χρησιμοποιήθηκαν, όπως φαίνεται τα εξής directives:

```
#pragma HLS UNROLL factor=2
#pragma HLS PIPELINE II=1
```

```
void copy1(uint8 BA[n][m],uint8 A1[n][m] ){
    //BRAM---------

    for(int i = 0; i < n; i++){
    #pragma HLS loop_tripcount min=nn max=nn
        for(int j = 0; j < m; j++){
        #pragma HLS loop_tripcount min=mm max=mm
        #pragma HLS UNROLL factor=2
        #pragma HLS PIPELINE II=1

            BA[i][j]=A1[i][j];

        }
    }//BRAM--------
```

Με το pipeline μειώθηκε ο χρόνος της λούπας που φαίνεται απο n*m*k (όπου k ο χρόνος για `BA[i][j]=A1[i][j]; )` σε n*m*1 + k

Με το unroll factor=2, εκμεταλλευόμαστε τα δύο Ports της dram.

Οι συναρτήσεις copy1 και copy2 καλούνται μέσα στη mult_hw, και με τη χρήση του `#pragma HLS DATAFLOW` καταφέρνουμε Pipeline σε επίπεδο συναρτήσεων και loops, οπότε οι συναρτήσεις copy1 και copy2 είναι pipelined.

Παράλληλα τα `#pragma HLS UNROLL factor=2` `#pragma HLS PIPELINE II=1` χρησιμοποιήθηκαν και στο σημείο των πολλαπλασιασμών.

Εδώ το pipeline κάνει και unroll στο loop k=(0 έως m), ενώ το unroll factor=2 αναφέρεται στη for από j=(0 έως p) για να εκμεταλλευτούμε πάλι τα 2 ports της dram (AB[i][j])

```
for(int i = 0; i < n; i++){
#pragma HLS loop_tripcount min=nn max=nn //min = nn ktl

    for(int j = 0; j < p; j++){
#pragma HLS loop_tripcount min=pp max=pp

        int result = 0;
#pragma HLS UNROLL factor=2
#pragma HLS PIPELINE II=1
        for(int k = 0; k < m; k++){
#pragma HLS loop_tripcount min=mm max=mm
        //BRAM_AB[i][j] += BRAM_A[i][k]*BRAM_B[k][j];
        result+= BRAM_A[i][k]*BRAM_B[k][j];
        }
        AB[i][j] = result;

    }
}
```

Βέλτιστη λύση για lm=ln=lp=8:
Estimated clock period: 8.563ns
Number of DSP48E used: 268
Number of BRAMs used: 512
Number of FFs used: 3060
Number of LUTs used: 22201
Total execution time: 655625 ns
Min latency: 65544
Avg. latency: 65544
Max latency: 65544

Mult_sw time elapsed: 1633722 us

A) speedup(hw) = 336860355/655625 = 513x
B) speedup(sw) = 1633722000/655625 = 2491x