Σχεδίαση Συστημάτων Υλικού - Λογισμικού

Εργαστήριο 1ο

Ονοματεπώνυμο	AEM	E-mail
Στασινός Αλκιβιάδης	**9214**	**astasinos@ece.auth.gr**
Κοσεογλου Σωκράτης	**8837**	**sokrkose@ece.auth.gr**

Ερώτημα 1.

Δημιουργήθηκαν 3 αρχεία. Ένα για το design **matrixmul.cpp** , ένα για το testbench **matrixmul_tb.cpp** και ένα για τα απαραίτητα **defines/includes**. Τα δεδομένα των πινάκων εισόδου Α κ Β ορίστηκαν ως **8-bit unsigned** με τη χρήση της βιβλιοθήκης **ap_int.h** και του τύπου **ap_uint<8>** , ενώ για τον πίνακα εξόδου τα δεδομένα ήταν **unsigned integer.**

Όπως ζητείται οι πίνακες Α και Β αρχικοποιούνται στο testbench με τυχαία δεδομένα στο range 0 – 255. Στο design ορίζεται η συνάρτηση **hw_matrix_mul** που προορίζεται να τρέξει στο υλικό.

Στο testbench υλοποιείται επίσης μια συνάρτηση με την ίδια λειτουργικότητα με την **hw_matrix_mul** του υλικού με όνομα **sw_matrix_mul**, η οποία θα τρέξει στη CPU του υπολογιστή. Η καλή λειτουργία του κώδικα επαληθεύεται στο testbench με τη σύγκριση όλων των αποτελεσμάτων των δύο συναρτήσεων για ίδιους πίνακες εισόδου. Σε περίπτωση λάθους εκτυπώνεται μήνυμα λάθους “Bad Result” μαζί με το σημείο του λάθους. Αν όλα τα αποτελέσματα είναι σωστά και συμπίπτουν τότε εν τέλει εκτυπώνεται το μήνυμα “Test Passed”.

Github: https://github.com/astasinos/University-Projects/tree/master/Hardware%20Software%20Co-Design/First%20Lab

Ερώτημα 2.

Με τα **default settings** και με **lm,ln,lp = 8** έχουμε τα παρακάτω αποτελέσματα. Το μέγιστο latency είναι λογικό νούμερο αφού **m = n = p = 256** και μέσα από το Vivado HLS βλέπουμε ότι το τρίτο εμφωλευμένο loop χρειάζεται **2** κύκλους για να εκτελεστεί. Άρα **256*256*256*2 = 33554432** περίπου ίσο δηλαδή με τη **worst case latency**.

Estimated Clock Period	**3.691 ns**
Worst Case Latency	**33686017 cycles**
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 cycles**
Avg. latency	**33686017 cycles**
Max latency	**33686017 cycles**

Ερώτημα 4.

Αρχικά τοποθετήθηκε στο τρίτο εμφωλευμένο **loop** το directive **#pragma HLS PIPELINE II=1** και **#pragma UNROLL factor=2** ώστε σε ένα κύκλο να ολοκληρώνονται ο πολλαπλασιασμός μιας γραμμής και δύο στηλών και το αποτέλεσμα να αποθηκεύεται στον τελικό πίνακα εξόδου. Ωστόσο αν για παράδειγμα έχουμε **lm ln lp = 8** άρα και **m = n = p = 256** τότε θα πρέπει σε κάθε κύκλο να κάνουμε **256*2 = 512** αναγνώσεις και από τη μνήμη (πίνακας BRAM_in1) . Κάτι τέτοιο δεν είναι εφικτό αν δε γίνει **partitioning** στους πίνακες. Οι αρχικοί μας πίνακες **inArray1** και **inArray2** ως arguments της top function είναι αποθηκευμένοι στη μνήμη **DRAM** η οποία δεν επιδέχεται partitioning και μπορούμε να κάνουμε 2 accesses per cycle εφόσον είναι dual port. Για το λόγο αυτό δημιουργήθηκαν locally δύο νέοι πίνακες οι οποίοι αυτόματα βρίσκονται στη μνήμη **BRAM** και έγιναν **partition.** Ο **BRAM_in1** έγινε **complete partition** στη διάσταση των γραμμών (dim=1) ενώ ο **BRAM_in2 complete partition** στη διάσταση των στηλών (dim = 2). Εφόσον και η **BRAM** μνήμη είναι dual port με το partition αυτό μπορούμε να κάνουμε σε ένα κύκλο **2*n** και **2*m** αναγνώσεις από τους **BRAM_in1** και **BRAM_in2** αντίστοιχα. Το unroll factor δεν αυξήθηκε περαιτέρω καθώς τα αποτελέσματα πρέπει να γράφονται στον **outArray** ο όποιος είναι αποθηκευμένος στη **DRAM**.

```
    rows : for(int i = 0; i < n; i++){
    #pragma HLS loop_tripcount min=n_iter max=n_iter

    cols:   for(int j = 0; j < p; j++) {
        #pragma HLS loop_tripcount min=p_iter max=p_iter
        #pragma HLS UNROLL factor=2
        #pragma HLS PIPELINE II=1

            result = 0;
    product : for(int k = 0; k < m; k++){
                #pragma HLS loop_tripcount min=m_iter max=m_iter

                result += BRAM_in1[i][k] * BRAM_in2[k][j];
}

            outArray[i][j] = result;
        }
```

Τα δεδομένα μας όμως πρέπει αρχικά να αντιγραφούν από τη **DRAM** στη **BRAM**. Για το σκοπό αυτό δημιουργήθηκαν δύο συναρτήσεις **ARRAY_COPY1** , **ARRAY_COPY2** . Σε αυτές εφαρμόζεται **PIPELINE** και **UNROLL factor=2** στον πιο εσωτερικό βρόχο με σκοπό να αντιγράφουμε σε κάθε κύκλο **δύο** στοιχεία από τη **DRAM** στη **BRAM** το οποίο είναι και το μέγιστο.

```
void ARRAY_COPY1(ap_uint<8> inArray1[n][m],ap_uint<8> outArray1[n][m]){


    for(int i = 0; i < n; i++){
        #pragma HLS loop_tripcount min=n_iter max=n_iter
            for(int j = 0; j < m; j++){
                #pragma HLS loop_tripcount min=m_iter max=m_iter
                #pragma HLS UNROLL factor=2
                #pragma HLS PIPELINE II=1
                outArray1[i][j] = inArray1[i][j];
            }
        }



}
```

Τέλος εφαρμόστηκε **#pragma HLS DATAFLOW** ώστε οι δύο κλήσεις **ARRAY_COPY1** και **ARRAY_COPY2** να γίνουν παράλληλα εφόσον δεν υπάρχει κάποιο dependency μεταξύ τους.

Το συνολικό **latency** για **m = n = p = 256** έφτασε στα **65544 κύκλους**. Ο μισός χρόνος οφείλεται στην αντιγραφή από **DRAM → BRAM** , καθώς η αντιγραφή ενός πίνακα **n x m** μας κοστίζει **n * m / 2** . Άρα αφού οι αντιγραφές μας γίνονται παράλληλα κοστίζουν συνολικά **256 * 256 / 2 = 32,768** κύκλους.

Ό υπόλοιπος χρόνος δαπανάται στον ίδιο τον πολλαπλασιασμό, ο οποίος χρειάζεται **n*p / 2** , δηλαδή άλλους **32,768** κύκλους. Σύνολο **65,536** περίπου ίσο και με το αποτέλεσμα της προσομοίωσης.

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.563 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
65544	65544	0.655 ms	0.655 ms	32774	32774	dataflow

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
Loop_1_proc_U0	Loop_1_proc	32773	32773	0.328 ms	0.328 ms	32773	32773	none
ARRAY_COPY2_U0	ARRAY_COPY2	32770	32770	0.328 ms	0.328 ms	32770	32770	none
ARRAY_COPY1_U0	ARRAY_COPY1	32770	32770	0.328 ms	0.328 ms	32770	32770	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2072	-
FIFO	-	-	-	-	-
Instance	0	268	2542	15485	-
Memory	512	-	0	0	0
Multiplexer	-	-	-	4644	-
Register	-	-	518	-	-
Total	512	268	3060	22201	0
Available	4320	6840	2364480	1182240	960
Available SLR	1440	2280	788160	394080	320
Utilization (%)	11	3	~0	1	0
Utilization SLR (%)	35	11	~0	5	0

Cosimulation Report for 'matrix_mul_hw'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	65544	65544	65544	NA	NA	NA

Export the report(.html) using the Export Wizard

Estimated Clock Period	**8.563 ns**
Worst Case Latency	**65544 cycles**
Number of DSP48E used	**268 (3%)**
Number of BRAMs used	**512 (11%)**
Number of FFs used	**3060 (0%)**
Number of LUTs used	**22201 (1%)**
Total Execution Time	**655625 ns**
Min latency	**65544 cycles**
Avg. latency	**65544 cycles**
Max latency	**65544 cycles**

Ερώτημα 5. ΥπολογισμόςΕπιτάχυνσης

Για τη μέτρηση του χρόνου της συνάρτησης υλοποιημένης σε **Software** χρησιμοποιήθηκε η βιβλιοθήκη **<chrono>** .

Επιτάχυνση από αρχική υλοποίηση σε **Hardware → 336860355 ns / 655625 ns = 513.8**

Επιτάχυνση από αρχική υλοποίηση σε **Software → 2460420600 ns / 655625 ns = 3,752**