Εργαστήριο 1: Matrix Multiplication

Ομάδα 10

November 22, 2024

Abstract

Σκοπός της εργασίας είναι η υλοποίση ενός αλγορίθμου πολλαπλασιασμού πινάκων σε στο Vitis HLS. Συγκεκριμένα θα χρησιμοποιηθούν HLS directives για μείωση του execution time.

1 Ερώτημα 1

Μπορείτε να βρείτε τον κώδικα του project στο Github.

2 Ερώτημα 2

Θεωρούμε Πίνακες $A[n \times m]$ και $B[m \times p]$ όπου

- $m = 2^{lm}$ όπου lm ένας ακέραιος τέτοιος ώστε $1 \leq lm \leq 7$
- $n = 2^{ln}$ όπου lm ένας ακέραιος τέτοιος ώστε $1 \leq ln \leq 7$
- $p = 2^{lp}$ όπου lm ένας ακέραιος τέτοιος ώστε $1 \leq lp \leq 7$

Εκτέλεση C synthesis για $lm = ln = lp = 6$:

Estimated clock period	6.456ns
Worst case latency	24591(cycles) — 0.246ms
Number of DSP48E used	32
Number of BRAMS used	66
Number of FFs used	653
Number of LUTs used	4968

3 Ερώτημα 3

Εκτέλεση C/RTL cosimulation για $lm = ln = lp = 6$:

Total Execution Time	246055ns
Min latency	24585
Avg latency	24585
Max latency	24585

4 Ερώτημα 4

4.1 Παρατηρήσεις για lm σταθερό και μεταβλητές ln, lp

Οι συνολικές πράξεις (πολλαπλασιασμοί και προσθέσεις) είναι αναλογικές με το: $n \times p \times m$
Για παράδειγμα με σταθερή την εσωτερική διάσταση $m = 6$ και αυξάνοντας το $ln = 7$

μειώνοντας το $lp = 5$ έχουμε θεωρητικά ίδια αλγοριθμική πολυπλοκότητα ωστόσο παρατηρούμε ότι η απόδοση έχει μειωθεί.
Μερικά παραδείγματα

lm	ln	lp	Total Execution Time	latency (cycles)
6	6	6	158325 ns	12324
6	7	5	228535 ns	22802
6	5	7	169955 ns	16944

Παρατηρούμε επίσης ότι οι διαστάσεις n και p επηρεάζουν διαφορετικά το χρόνο στις δύο περιπτώσεις και ο λόγος είναι τα διαφορετικά memory access patterns.

Επιπλέον παρατηρούμε διαφορές στο utilization των δομικών μονάδων.

4.2 Βέλτιστη λύση

Αρχικά χρησιμοποιούμε το HLS INTERFACE directive με διαφορετικά memory banks για τα A και B

```
#pragma HLS INTERFACE m_axi bundle=gmem0 port = in1 depth = (n * m)
#pragma HLS INTERFACE m_axi bundle=gmem port = in2 depth = (m * p)
#pragma HLS INTERFACE m_axi bundle=gmem0 port = out_r depth = (n * p)
```

Χρησιμοποιούμε cyclic partition για τον A για row-wise access

```
#pragma HLS ARRAY_PARTITION variable = A dim = 1 cyclic factor = 16
```

και block partition για τον B για column-wise access

```
#pragma HLS ARRAY_PARTITION variable = B dim = 1 block factor = 16
```

Επιπλέον χρησιμοποιήσαμε το HLS TRIPCOUNT για να δώσουμε έξτρα πληροφορία για τον αριθμό των επαναλήψεων στον compiler.

4.3 Παρατηρήσεις

Δοκιμάσαμε αρκετές παραλλαγές για loop unroll με διαφορετικά factors για τα readA και readB χωρίς να δούμε κάποια βελτίωση στην απόδοση.
Δοκιμάσαμε unroll στο outer loop και επίσης δεν είδαμε βελτίωση.
Τέλος φαίνεται ότι υπάρχει αυτόματο pipelining και η δοκιμή μας με αυτό επίσης δεν επηρεάσε τις αποδόσεις.

4.4 Αποτελέσματα για $lm = ln = lp = 6$

Estimated clock perio	7.3ns
Worst case latency	12324(cycles) — 0.123ms
Number of DSP48E used	32
Number of BRAMS used	0 — (URAM 1)
Number of FFs used	3626
Number of LUTs used	9643
Total Execution Time	158325 ns
Min latency	15781
Avg latency	15781
Max latency	15781

4.5 Επιτάχυνση

Πετύχαμε $246055 \div 158325 \simeq 1.554\times$ επιτάχυνση.

5 Authors

Βογιατζής Χαρίσιος ΑΕΜ:9192
Τσινός Αναστάσιος ΑΕΜ:10223