Σχεδίαση Συστημάτων Υλικού-Λογισμικού (1η Εργασία)

Ομάδα 10: Καραμητόπουλος Παναγιώτης ΑΕΜ 9743 karamitopp@ece.auth.gr

Σαριδάκη Μαρία-Ραφαηλία ΑΕΜ 9633 saridakm@ece.auth.gr

Ερώτημα 1: Υλοποιήθηκε σε κώδικα C, ο hardware accelerator MATRIX_MUL ο οποίος υπολογίζει το γινόμενο δύο πινάκων (A και B), και ένα testbench το οποίο υπολογίζει το γινόμενο των πινάκων (οι οποίοι αρχικοποιούνται με ψευδοτυχαίες τιμές στο εύρος 0-255) χρησιμοποιώντας τόσο μια SW μαζί με τη HW λύση ούτως ώστε να επαληθευτεί η σωστή εκτέλεση του HW.

Ερώτημα 2:

Κάνοντας c synthesis με default settings για Im=In=Ip=6 προκύπτουν:

Estimated clock period: 3.691ns

Worst case latency: 5.326 ms

Number of DSP48E used: 1

Number of BRAMs used: 0

Number of FFs used: 95

Number of LUTs used: 165

Ερώτημα 3:

Τρέχοντας C/RTL co-simulation η σχεδίαση περνάει το test επιτυχώς. Για Im=In=Ip=6 προκύπτουν:

Total Execution Time: `5326275 ns`

Min latency: `532609 (κύκλοι) = 5.326` ms

Avg. latency: `532609 (κύκλοι) = 5.326` ms

Max latency: `532609 (κύκλοι) = 5.326` ms

Ερώτημα 4:

i) Χρησιμοποιώντας PIPELINE, ARRAY_PARTITION και BRAMs για In = 3, Im = 6, Ip = 3 η καθυστέρηση είναι 1094 κύκλοι, οι κύκλοι που γίνεται ο πολλαπλασιασμός είναι: $2^3 \cdot 2^3 + 2 = 8 \cdot 8 + 2 = 64 + 2 = 66$ ενώ οι υπόλοιποι $1028 = 514 + 514 = (512 + 2) + (512 + 2) = 2 \cdot (2^3 \cdot 2^6 + 2)$ χρησιμοποιούνται για την αντιγραφή των δεδομένων από την DRAM στις BRAMs.

Χρησιμοποιώντας PIPELINE, ARRAY_PARTITION και BRAMs για In = 5, Im = 6, Ip = 5 η καθυστέρηση είναι 5126 κύκλοι, οι κύκλοι που γίνεται ο πολλαπλασιασμός είναι: $2^5 \cdot 2^5 + 2 = 32 \cdot 32 + 2 = 1026$ ενώ οι υπόλοιποι χρησιμοποιούνται για την αντιγραφή των δεδομένων από την DRAM στις BRAMs.

Χρησιμοποιώντας PIPELINE, ARRAY_PARTITION και BRAMs για In = 3, Im = 6, Ip = 2 η καθυστέρηση είναι 806 κύκλοι, οι κύκλοι που γίνεται ο πολλαπλασιασμός είναι: $2^3 \cdot 2^2 + 2 = 8 \cdot 4 + 2 = 34$ ενώ οι υπόλοιποι $772 = 2^3 \cdot 2^6 + 2 + 2^6 \cdot 2^2 + 2$ χρησιμοποιούνται για την αντιγραφή των δεδομένων από την DRAM στις BRAMs.

ii)

Estimated clock period: 7.721 ns

Number of DSP48E used: 32

Number of BRAMs used: 64

Number of FFs used: 703 (αντί cycle, αν γίνει block 680)

Number of LUTs used: 3383 (αντί cycle partition, αν γίνει block 3373 όλα τα υπόλοιπα παραμένουν ίδια)

Total Execution Time: 123135 ns

Min latency: 12295 (κύκλοι) = 0.123 ms = 123 us

Avg. latency: 12295 (κύκλοι) = 0.123 ms = 123 us

Max latency: 12295 (κύκλοι) = 0.123 ms = 123 us

Στην πρώτη περίπτωση οι εντολές στις λούπες for εκτελούνται σειριακά, γεγονός που δικαιολογεί «σχετικά» μεγάλο execution time. Ειδικότερα αυτήν η υλοποίηση τρέχει σε περίπου **64X64X64X2=524288** κύκλους. Για να βελτιωθεί το execution time χρησιμοποιήθηκε PIPELINE και ARRAY_PARTITION. Με την χρήση του pipeline, οι δύο λούπες for έγιναν flatten ενώ η 3η έγινε unroll, με αυτόν τον τρόπο μειώθηκαν οι κύκλοι που χρειάζονται για την εκτέλεση τους, ποιο συγκεκριμένα απαιτούνται πλέον **64X64+3 = 4099** κύκλοι. Επιπλέον γίνεται partition στους πίνακες (λόγω των δύο θυρών, γίνεται partition με factor $= 32 = \frac{2^6}{2}$), για τον λόγο ότι για το pipelining θα πρέπει να γίνονται πολλαπλές και ταυτόχρονες προσβάσεις στην μνήμη.

Αν και οι βελτιστοποιήσεις σε επίπεδο προσομοίωσης δουλεύουν ορθά, στην πράξη θα υπήρχε πρόβλημα αφού δεν μπορεί να γίνει ταυτόχρονα ανάγνωση σε δύο θέσεις μνήμης της DRAM. Για το λόγο αυτό, αποθηκεύονται οι πίνακες στις BRAMs, στην οποία σε αντίθεση με την DRAM μπορεί να γίνει partition. Επιπλέον, για να μειωθούν οι κύκλοι ρολογιού, χρησιμοποιήθηκε pipelining ούτως ώστε σε κάθε κύκλο να μπορούμε να μεταβούμε από την DRAM στην BRAM. Τα παραπάνω αποτελέσματα είναι αυτά που προκύπτουν από τις βελτιστοποιήσεις PIPELINE και ARRAY_PARTITION συμπεριλαμβάνοντας και την αντιγραφή των δεδομένων από την DRAM στις BRAMs.

iii) Η επιτάχυνση της βέλτιστης υλοποίησης (123135 ns) συγκριτικά με την αρχική σχεδίαση σε HW (5326275 ns) είναι: X 43.25

$$\boldsymbol{speedup} = \frac{old\ execution\ time}{new\ execution\ time} = \frac{5326275}{123135} = 43.25$$

Παράρτημα Α

pragmas

`#pragma HLS loop_tripcount min=<int> max=<int> avg=<int>`
Προορίζεται μόνο για ανάλυση, δεν επηρεάζει τα αποτελέσματα της σύνθεσης και καθορίζει τον συνολικό αριθμό των επαναλήψεων που εκτελούνται από μια loop. Το εργαλείο βρίσκει τη συνολική καθυστέρηση κάθε loop, η οποία είναι ο αριθμός των κύκλων ρολογιού για την εκτέλεση όλων των επαναλήψεων της loop. Επομένως, η καθυστέρηση του βρόχου είναι συνάρτηση του αριθμού των επαναλήψεων της loop ή του tripcount. Το tripcount μπορεί να έχει μια σταθερή τιμή, μπορεί να εξαρτάται από την τιμή των μεταβλητών που χρησιμοποιούνται στην λούπα (πχ x < y), ή να εξαρτάται από τις εντολές ελέγχου που χρησιμοποιούνται εντός της λούπας. Τα *min*, *max*, *avg* καθορίζουν maximum/minimum/average αριθμό των επαναλήψεων της λούπας.

`#pragma HLS array_partition variable=<name> type=<type> factor=<int> dim=<int>`
Διαμερίζει έναν πίνακα σε μικρότερους πίνακες ή μεμονωμένα στοιχεία, πετυχαίνοντας τα εξής:

1) Προκύπτει RTL με πολλαπλές μικρές μνήμες ή πολλαπλούς καταχωρητές αντί για μια μεγάλη μνήμη
2) Αυξάνει τον αριθμό θυρών για read/write
3) Βελτιώνει την απόδοση της σχεδίασης
4) Απαιτεί περισσότερους registers ή memory instances

variable: προσδιορίζει τον πίνακα που πρέπει να διαχωριστεί.

factor: Καθορίζει τον αριθμό των μικρότερων πινάκων που θα δημιουργηθούν (για block και cyclic partitioning, το factor είναι απαραίτητο).

dim: Καθορίζει ποια διάσταση θα διαχωριστεί, αν dim = 1 (2) διαχωρίζεται μόνο η πρώτη (δεύτερη) διάσταση του πίνακα, αν dim = 0 διαμερίζονται όλες οι διαστάσεις του πίνακα.

type:

`cyclic`: Ο πίνακας διαμερίζεται κυκλικά, κάθε ένα στοιχείο τοποθετείται σε κάθε νέο πίνακα πριν επιστρέψει στον πρώτο πίνακα για να επαναλάβει τον κύκλο μέχρι να διαμεριστεί πλήρως ο πίνακας. Για παράδειγμα, εάν χρησιμοποιείται factor=3:

Το στοιχείο 0 ανατίθεται στον πρώτο νέο πίνακα, το στοιχείο 1 ανατίθεται στον δεύτερο νέο πίνακα, το στοιχείο 2 ανατίθεται στον τρίτο νέο πίνακα, το στοιχείο 3 ανατίθεται και πάλι στον πρώτο νέο πίνακα.

`block`: Δημιουργεί μικρότερους πίνακες από διαδοχικά μπλοκ του αρχικού πίνακα. Ουσιαστικά χωρίζει τον πίνακα σε N ίσα μπλοκ, όπου N είναι ο ακέραιος αριθμός που του factor.

`complete`: Διασπά τον πίνακα σε μεμονωμένα στοιχεία. Για έναν μονοδιάστατο πίνακα, αντιστοιχεί στην ανάλυση μιας μνήμης σε μεμονωμένους καταχωρητές.

(αν γίνει `complete` *οι BRAMS που χρησιμοποιούνται είναι 0, αλλά χρησιμοποιούνται πολλά περισσότερα FFs και LUTs, αν χρησιμοποιηθεί* `complete` *ή* `block` *τα αποτελέσματα είναι παρόμοια, υπάρχουν 2 πολύ μικρές αλλαγές που αναφέρονται παραπάνω).*

`#pragma HLS unroll factor=<N> region skip_exit_check`
Οι λούπες στην C/C++ by default είναι τυλιγμένοι, οπότε κατά την σύνθεση δημιουργείται η λογική για μια επανάληψη του βρόχου και η σχεδίαση RTL εκτελεί αυτή τη λογική για κάθε επανάληψη του βρόχου διαδοχικά. Το πράγμα unroll ξετυλίγει τις λούπες δημιουργώντας πολλαπλά αντίγραφα του εσωτερικού της λούπας στη σχεδίαση RTL, γεγονός που επιτρέπει ορισμένες ή και όλες τις επαναλήψεις της λούπας να γίνουν παράλληλα, και να αυξηθεί η απόδοση.

Τοποθετείται στο εσωτερικό της λούπας.
`factor`: καθορίζει αν θα γίνει μερικό ή πλήρες «ξετύλιγμα». Το πλήρες ξετύλιγμα της λούπας δημιουργεί ένα αντίγραφο του εσωτερικού της λούπας στο RTL για κάθε επανάληψη του βρόχου, ώστε ολόκληρη η λούπα να μπορεί να εκτελεστεί ταυτόχρονα. Κατά το μερικό ξετύλιγμα της λούπας δημιουργούνται N αντίγραφα του εσωτερικού της λούπας για να μειωθούν ανάλογα οι επαναλήψεις του βρόχου.

`skip_exit_check`: προαιρετική λέξη-κλειδί που χρησιμοποιείται μόνο πρόκειται να γίνει μερικό unroll.

`#pragma HLS pipeline II=<int> off rewind style=<value>`
Κλασικό pipelining - κάθε <N> κύκλους ρολογιού (initiation interval) αρχίζει να εκτελείται μια νέα εντολή. Έτσι, μειώνεται ο χρόνος εκτέλεσης του προγράμματος, καθώς δε χρειάζεται να ολοκληρωθεί η εκτέλεση μιας εντολής, για να αρχίσει η επόμενη (στη γενική περιγραφή αγνοούμε τα data dependencies).

`II`: initiation interval – ο αριθμός των κύκλων ρολογιού που χρειάζονται για να αρχίσει να εκτελείται κάθε φορά η επόμενη εντολή

`off`: προαιρετική λέξη-κλειδί, απενεργοποιεί το pipeline για μια συγκεκριμένη λούπα ή συνάρτηση. Χρησιμοποιείται για την απενεργοποίηση του pipeline σε μια συγκεκριμένη λούπα όταν config_compile -pipeline_loops is used to globally pipeline loops.

`Rewind`: προαιρετική λέξη-κλειδί, ενεργοποιεί τη διοχέτευση (pipelining) συνεχόμενων loops, χωρίς παύσεις στο ενδιάμεσο. Χρήσιμο μόνο στην περίπτωση μοναδικής λούπας σε top-level function.

style=<stp | frp | flp>: καθορίζει το είδος του pipeline:

- ✓ stp: το default είδος. Στην περίπτωση που δεν υπάρχουν διαθέσιμα δεδομένα εισόδου, κάνει stall (stall pipeline).
- ✓ frp: free-running, flushable pipeline – το pipeline «τρέχει», ακόμα και όταν δεν υπάρχουν διαθέσιμα δεδομένα εισόδου. Βελτιώνει το χρονισμό του συστήματος, αλλά αυξάνει την καταναλισκόμενη ενέργεια.
- ✓ flp: flushable pipeline – συνήθως απαιτεί περισσότερο υλικό ή/και μεγαλύτερο initiation interval.

Παράρτημα Β (περιέχει κάποια από τα αποτελέσματα των δοκιμών που έγιναν με τα pragmas)

array partition without BRAMs, tripcount pragma

Κάνοντας c synthesis με default settings για lm=ln=lp=6 προκύπτουν:

Estimated clock period: 4.345 ns

Worst case latency: 5.326 ms

Number of DSP48E used: 1

Number of BRAMs used: 0

Number of FFs used: 107

Number of LUTs used: 461 (το ερώτημα 3 είναι ακριβώς ίδιο με πάνω)

Unroll, array partition, without BRAMs and tripcount pragma

Estimated clock period: 7.619 ns

Number of DSP48E used: 32

Number of BRAMs used: 0

Number of FFs used: 460

Number of LUTs used: 1981

Total Execution Time: 124355 ns

Min latency: 12417 (κύκλοι) = 0.124 ms = 124 ns

Avg. latency: 12417 (κύκλοι) = 0.124 ms = 124 ns

Max latency: 12417 (κύκλοι) = 0.124 ms = 124 ns

Pipeline, tripcount pragma, array partition without BRAMs

Estimated clock period: 7.721 ns

Number of DSP48E used: 32

Number of BRAMs used: 0

Number of FFs used: 604

Number of LUTs used: 2038

Total Execution Time: 41175 ns

Min latency: 4099 (κύκλοι) = 40.990 us

Avg. latency: 4099 (κύκλοι) = 40.990 us

Max latency: 4099 (κύκλοι) = 40.990 us