**Έλεγχος Ορθής Λειτουργίας Ψηφιακών Συστημάτων**

***Αναφορά Εργασίας***

Σπυροπούλου Θεώνη Κατερίνα (Α.Μ. 212)

Παπουτσής Γεώργιος (Α.Μ. 193)

***Θέμα:*** Yλοποίηση τεχνικών συμπίεσης δεδομένων δοκιμής προσχεδιασμένων πυρήνων (IP cores) και αξιολόγηση τους.

***Βιβλιογραφία:*** “Test Data Compression for IP Embedded Cores Using Selective Encoding of Scan Slices”, Zhanglei Wang, Krishnendu Chakrabarty, Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708***.***

**ΕΙΣΑΓΩΓΗ**

Η παρούσα εργασία αποτελείται από δύο σκέλη. Αφενός την υλοποίηση αλγορίθμων συμπίεσης σε γλώσσα C και την αξιολόγησή τους ως προς την αποδοτικότητα της συμπίεσης (compression ratio) αλλά και τον χρόνο εκτέλεσης του αλγορίθμου (runtime) για την εξαγωγή του συμπιεσμένου συνόλου δοκιμής, και αφετέρου την υλοποίηση του μηχανισμού ανάκτησης (decompressor) του συνόλου δοκιμής που ενσωματώνεται στον προς έλεγχο κύκλωμα, σε γλώσσα VHDL και την αξιολόγηση του ως προς το μέγεθος της περιοχής κάλυψης (hardware area overhead) σε ισοδύναμες πύλες (equivalent gates).

Η υλοποίηση κάθε σκέλους της εργασίας έγινε βάσει της δοθείσας βιβλιογραφίας και οι λεπτομέρειές της, όπως και τα αποτελέσματα της αξιολόγησης, παρατίθονται στη συνέχεια της αναφοράς.

**ΜΕΡΟΣ Α: ΑΛΓΟΡΙΘΜΟΣ ΣΥΜΠΙΕΣΗΣ**

Αρχικά θα πρέπει να αναφέρουμε ορισμένες παραδοχές που έγιναν σχετικά με τη μορφή εισαγωγής των διανυσμάτων δοκιμής στις scan αλυσίδες. Έστω διάνυσμα δοκιμής, μεγέθους 31 bits, . Θεωρώντας, για παράδειγμα, ότι υπάρχουν 5 scan αλυσίδες των 7 scan cells, η μορφή του διανύσματος δοκιμής μέσα σ’αυτές ακολουθεί τη μορφή του παρακάτω δισδιάστατου πίνακα, όπου κάθε γραμμή αντιπροσωπεύει μια scan αλυσίδα. Αν ο πίνακας είναι μεγαλύτερος από το διάνυσμα, γίνεται padding με αδιάφορους όρους.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | X | X | X |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Κάθε στήλη του πίνακα αντιπροσωπεύει ένα ***scan slice*** υποψήφιο προς κωδικοποίηση. Η κωδικοποίηση ξεκινά από την τελευταία στήλη του πίνακα, καθώς τα scan slices εισάγονται παράλληλα στις scan αλυσίδες και ολισθαίνουν κατά μήκος τους.

Η προτεινόμενη μέθοδος συμπίεσης του συνόλου δοκιμής βασίζεται στην κωδικοποίηση συγκεκριμένων bits ενός scan slice και ακολουθεί τους εξής κανόνες:

1) *Single-bit mode:* Όταν ένα scan slice περιέχει μόνο ένα 0 (1) και όλα τα υπόλοιπα bits είναι 1 (0), μόνο εκείνο το bit κωδικοποιείται και εφ’εξής ονομάζεται ***target symbol***.

2) *Group-copy mode:* Όταν ένα scan slice περιέχει περισσότερα 0 (1) από 1 (0), κωδικοποιούνται μόνο τα 1 (0).

3) Οι αδιάφοροι όροι αντιστοιχίζονται στο συμπληρωματικό target symbol.

4) Το μέγεθος κάθε κωδικοποιημένου scan slice είναι , όπου Ν ο αριθμός των scan αλυσίδων. Κάθε κωδικοποιημένο scan slice αποτελείται από bits ***data code*** συν 2 bits ***control code***. Για την περίπτωση του group-copy mode, το κάθε slice χωρίζεται σε ομάδες των k-bits και κωδικοποιούνται μόνο οι ομάδες που περιέχουν target symbols.

Ο αλγόριθμος κωδικοποίησης είναι:

Μορφοποίηση διανυσμάτων δοκιμής σε scan sclices;

**for** κάθε slice

Καθορισμός αριθμού των 0 () και 1 () στο slice;

**if** () **then**

target-symbol := 1, control-code := 00;

**else**

target-symbol := 0, control-code := 01;

**for** κάθε ομάδα του slice

Υπολογισμός αριθμού target symbols;

**if** αριθμός target symbols > 2 **then**

κωδικοποίηση με group-copy mode

**else**

κωδικοποίηση με single-bit mode

**end**

Εξαγωγή κωδικοποιημένου slice;

**end**

Ακολουθούν ορισμένα παραδείγματα κωδικοποίησης:

A description...

*Εικόνα 1: Παραδείγματα κωδικοποίησης*

Ο κώδικας σε γλώσσα C για την παραγωγή του συμπιεσμένου συνόλου δοκιμής βρίσκεται στο αρχείο *encoder.c* και είναι επαρκώς σχολιασμένος.

Μπορούμε να επιτύχουμε περαιτέρω συμπίεση του συνόλου δοκιμής αν λάβουμε υπ’όψιν μας ότι συνεχείς επαναλήψεις ενός κωδικοποιημένου scan slice (pattern repeat) είναι πιθανές, επομένως μπορούμε να συμπεριλάβουμε στο τελικό σύνολο δοκιμής μόνο την πρώτη εμφάνισή του, καθώς και τον αριθμό των συνολικών εμφανίσεων που ακολουθούν. Για το λόγο αυτό δημιουργήσαμε το πρόγραμμα που βρίσκεται στο αρχείο *ATE\_Pattern\_Repeat.c,* καθώς και το πρόγραμμα που βρίσκεται στο αρχείο *reverseRepeat.c,* που αναιρεί τη διαδικασία pattern repeat, για την επαλήθευση της ορθότητας των αποτελεσμάτων. Η εικόνα που ακολουθεί δείχνει το αποτέλεσμα της διασικασίας pattern repeat.

A description...

*Εικόνα 2: Pattern repeat*

**ΜΕΡΟΣ Β: ΜΗΧΑΝΙΣΜΟΣ ΑΝΑΚΤΗΣΗΣ ΤΟΥ ΣΥΝΟΛΟΥ ΔΟΚΙΜΗΣ**

Το σύστημα αποκωδικοποίησης του συμπιεσμένου συνόλου δοκιμής για την εξαγωγή των αρχικών διανυσμάτων και ολίσθησή τους στις scan αλυσίδες έχει την ακόλουθη μορφή:

A description...

*Εικόνα 3: Decompressor*

Όπως βλέπουμε, ο αποκωδικοποιητής αποτελείται από τα εξής στοιχεία:

* ***FSM***: παραγωγή σημάτων ελέγχου.
* ***N-bit buffer***: αποθήκευση του αποσυμπιεσμένου διανύσματος δοκιμής, προκειμένου να ολισθήσει κατόπιν στις scan αλυσίδες.
* ***K-to-N decoder***: σε single-bit mode ενεργοποιεί μόνο ένα bit του buffer, σε group-copy mode ενεργοποιεί όλα τα bits της ομάδας που υποδεικνύεται από τον address register.
* ***K-bit address register***: στην περίπτωση του group-copy mode διευθυνσιοδοτεί τις συνεχόμενες ομάδες κωδικοποιημένων bits, περιέχοντας τη διεύθυνση του πρώτου bit της ομάδας.

Η FSM του συστήματος εκτελεί διάφορες λειτουργίες ανάλογα με το control-code του κωδικοποιημένου slice που δέχεται ως είσοδο το σύστημα, ακολουθώντας το επόμενο σχεδιάγραμμα:

A description...

*Εικόνα 4: Διάγραμμα καταστάσεων FSM*

|  |  |
| --- | --- |
| **Λειτουργία** | **Περιγραφή** |
| Ρ1 | Αν έχουμε control-code 00 (01), το target-symbol είναι 1 (0). Θέτουμε το bit που καθορίζεται από το data-code στην τιμή αυτή και όλα τα υπόλοιπα bits στη συμπληρωματική της. |
| Ρ2 | Ολίσθηση του τρέχοντος περιεχομένου του buffer στις scan αλυσίδες και επανεκτέλεση της Ρ1. |
| Ρ3 | Αποθήκευση του data-code στον address register (δείκτης προς το πρώτο bit μιας ομάδας). |
| Ρ4 | Αντιγραφή του data-code στην ομάδα bits του buffer που υποδεικνύεται από τον address register και αύξηση του περιεχομένου του κατά k (υποδεικνύει την επόμενη ομάδα). |

*Πίνακας 1: Περιγραφή λειτουργιών FSM*

Στην παρακάτω εικόνα βλέπουμε τη λογική υλοποίησης κάθε bit του buffer, καθώς και τις συναρτήσεις απόδοσης τιμών στα χρησιμοποιούμενα σήματα.

A description...

*Εικόνα 5: Ένα bit του buffer*

Ο κώδικας σε γλώσσα VHDL για την υλοποίηση του μηχανισμού ανάκτησης των αρχικών διανυσμάτων δοκιμής βρίσκεται στο φάκελο «decompressor», όπου περιέχονται τα αρχεία .vhd για κάθε component του συστήματος, επαρκώς σχολιασμένα.

**ΜΕΡΟΣ Γ: ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΕΙΡΑΜΑΤΙΚΩΝ ΜΕΤΡΗΣΕΩΝ**

Στην ενότητα αυτή θα παρουσιάσουμε μετρήσεις που έγιναν για κάθε σκέλος της εργασίας, με βάση τα ακόλουθα κριτήρια:

* Για τον αλγόριθμο συμπίεσης εξετάσαμε την αποδοτικότητα της συμπίεσης () και τον χρόνο εκτέλεσης αλγορίθμου (***runtime***) για την εξαγωγή του συμπιεσμένου συνόλου δοκιμής.
* Για το μηχανισμό ανάκτησης εξετάσαμε το μέγεθος της περιοχής κάλυψης (***hardware area overhead***) σε ισοδύναμες πύλες ().

Για την αξιολόγηση του αλγορίθμου συμπίεσης ως προς την αποδοτικότητα της συμπίεσης, χρησιμοποιήσαμε τα σύνολα δοκιμής για τα συγκριτικά κυκλώματα ISCAS’89 και ΙWLS’05. Για την αξιολόγηση του χρόνου εκτέλεσης του αλγορίθμου χρησιμοποιήθηκε το πρόγραμμα *perf*. Τα αποτελέσματα των μετρήσεων φαίνονται στους πίνακες που ακολουθούν:

**ISCAS’89 stuck-at (MINTEST)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Test set*** | ***N*** | ***Compression ratio (%)*** | ***\*with ATE pattern repeat*** | ***Runtime*** |
| s13207f | 255 | **22,2** | **46,9** | 25,820,812 |
| 511 | -17,3 | **42** | 36,101,256 |
| 800 | -35,8 | **42,6** | 32,373,756 |
| s15850f | 255 | -25 | -1,3 | 4,247,785 |
| 511 | -26,9 | 0 | 4,253,213 |
| 800 | -40,8 | **6,6** | 8,821,747 |
| s35932f | 255 | -21,4 | -39,3 | 5,757,123 |
| 511 | -14,3 | -17,8 | 5,202,975 |
| 800 | -10,7 | -10,7 | 2,274,864 |
| s38417f | 255 | -52,5 | -74,7 | 32,842,421 |
| 511 | -46,3 | -58,6 | 34,083,534 |
| 800 | -41,3 | -46,9 | 37,579,208 |
| s38584f | 255 | -36,4 | -15,9 | 35,702,592 |
| 511 | -41,5 | -12,3 | 36,317,183 |
| 800 | -40,5 | -10,7 | 37,080,543 |

**ISCAS’89 LOC Transition Fault Testing (TetraMAX)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Test set*** | ***N*** | ***Compression ratio (%)*** | ***\*with ATE pattern repeat*** | ***Runtime***  ***(cycles)*** |
| s13207trans | 255 | -18,8 | **19,7** | 41,005,052 |
| 511 | -22,5 | **20,2** | 40,553,845 |
| 800 | -35,8 | **21,5** | 42,225,029 |
| s15850trans | 255 | -44,3 | -29,3 | 23,240,502 |
| 511 | -29,3 | -35,8 | 10,511,694 |
| 800 | -38,7 | -29,2 | 23,642,472 |
| s35932trans | 255 | **22,9** | **14,5** | 10,143,220 |
| 511 | **22,9** | **22,1** | 2,723,094 |
| 800 | **21,4** | **25,2** | 24,356,972 |
| s38417trans | 255 | **3,3** | **2,8** | 120,878,547 |
| 511 | **3,8** | **11,1** | 100,662,637 |
| 800 | **4,4** | **16,3** | 120,506,296 |
| s38584trans | 255 | -23,3 | -1,9 | 116,274,566 |
| 511 | -32,7 | -0,7 | 117,805,530 |
| 800 | -31,8 | -0,3 | 105,593,904 |

**IWLS’05 LOC** **Transition Fault Testing (TetraMAX)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Test set*** | ***N*** | ***Compression ratio (%)*** | ***\*with ATE pattern repeat*** | ***Runtime*** |
| des.area.tmax.trans | 255 | -17,6 | -15,3 | 8,275,497 |
| 511 | -10,6 | **4,7** | 3,212,571 |
| 800 | -8,2 | **5,9** | 28,226,291 |
| des.perf.trans.tmax | 255 | -44,9 | -92,9 | 83,789,601 |
| 511 | -37 | -74,8 | 89,018,946 |
| 800 | -32 | -61,1 | 86,787,208 |
| ethernet.trans.tmax | 255 | **90,8** | **95,2** | 8,684,630,707 |
| 511 | **79,8** | **94,4** | 9,776,851,670 |
| 800 | **66** | **93,8** | 10,947,951,753 |
| mem\_ctrl.trans.tmax | 255 | -3,8 | **29,4** | 128,883,919 |
| 511 | -11,9 | **33,7** | 155,049,718 |
| 800 | -24,5 | **39** | 136,759,402 |
| pci\_bridge32.trans.tmax | 255 | **46,2** | **76,2** | 745,444,248 |
| 511 | **34,3** | **77,6** | 879,103,449 |
| 800 | **21,2** | **78,5** | 953,992,404 |
| usb\_funct.trans.tmax | 255 | **4,6** | **23,5** | 123,214,585 |
| 511 | -2,6 | **31,5** | 132,108,839 |
| 800 | -3,4 | **37,5** | 136,643,448 |
| vga.lcd.trans.tmax | 255 | **94,7** | **97,7** | 30,327,606,355 |
| 511 | **92,3** | **97,3** | 32,238,257,621 |
| 800 | **87,9** | **97,1** | 34,253,640,023 |

**IWLS’05 stuck-at (TetraMAX)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Test set*** | ***N*** | ***Compression ratio (%)*** | ***\*with ATE pattern repeat*** | ***Runtime*** |
| ethernet.auto\_cmpr | 255 | **68,5** | **83,6** | 1,195,312,783 |
| 511 | **43,9** | **81,3** | 1,406,180,827 |
| 800 | **29,3** | **80,4** | 1,548,412,239 |
| mem\_ctrl.auto\_cmpr | 255 | **37,4** | **60,2** | 82,134,011 |
| 511 | **23,3** | **60,4** | 113,263,935 |
| 800 | **9,3** | **59,8** | 105,001,950 |
| pci\_bridge32.auto\_cmpr | 255 | -10 | **25,2** | 102,941,161 |
| 511 | -24 | **24,9** | 97,492,294 |
| 800 | -21,9 | **25** | 121,540,570 |

Παρατηρούμε ότι η προτεινόμενη μέθοδος δεν μας δίνει πάντα την επιθυμητή συμπίεση, ειδικά στα μικρότερα σύνολα δοκιμής. Αυτό συμβαίνει λόγω της μορφής των δοθέντων συνόλων δοκιμής, για παράδειγμα αν σε ένα slice έχουμε ένα target symbol στο πρώτο γκρουπ κι άλλο ένα στο τελευταίο, ή όταν όλα τα γκρουπς περιέχουν target symbols, το κωδικοποιημένο αποτέλεσμα θα περιέχει εντέλει περισσότερη πληροφορία από το αρχικό. Ταυτόχρονα, όμως, παρατηρούμε ενίοτε και σημαντικά ποσοστά συμπίεσης με τη χρήση της μεθόδου pattern repeat, ειδικά στα μεγαλύτερα σύνολα δοκιμής.

Για την αξιολόγηση του μηχανισμού ανάκτησης σχετικά με το μέγεθος της περιοχής κάλυψης χρησιμοποιήσαμε το εργαλείο Design Compiler της Synopsys, εκτελώντας το script ***run\_decomp.tcl***, από τα αποτελέσματα του οποίου(*run\_decomp.log*) βλέπουμε ότι η σχεδίασή μας καταλαμβάνει 12838.82 e.g.