Compression de Données de Test : Réduction du Nombre de Broches et Gain en Temps de Test

Julien Dalmasso, Marie-Lise Flottes, Bruno Rouzeyre

Compression de données de test : Réduction du nombre de broches et gain en temps de test.

Julien DALMASSO, Marie-Lise FLOTTES, Bruno ROUZEYRE
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
61 rue Ada, 34932 Montpellier CEDEX 5, France
{*dalmasso, flottes,* rouzeyre}@lirmm.fr

Résumé

La réduction des coûts de test des circuits intégrés est devenu un axe majeur da la recherche en microélectronique. De nombreuses techniques permettent de réduire ces coûts en réduisant le temps de test et/ou le nombre de broches nécessaires de l'équipement de test. Mais, pour pouvoir effectuer la compression, les méthodes proposées requièrent la présence de bits non spécifiés (X) dans les séquences de test. D'autre part, la longueur de ces séquences est, de manière significative, supérieure à celle de séquences complètement spécifiées. A l'inverse des techniques de compression, les méthodes basées sur la sérialisation des données de test peuvent utiliser les séquences complètement spécifiées, par ailleurs beaucoup plus courtes. Cet article présente tout d'abord une nouvelle méthode de compression horizontale des données de test, puis propose une réponse à la question: y a-t-il réellement un gain en terme de temps de test et de volume des données de test à utiliser la compression plutôt qu'une simple sérialisation des données de test?

1. Introduction

Lors de la conception et la fabrication de circuits intégrés, on a besoin de vérifier le bon fonctionnement du circuit. Et, au vu de l'évolution des circuits ces dernières années, les opérations de test sont devenues de plus en plus longues et coûtent donc de plus en plus cher en temps de test. Une technique de conception en vue du test très répandue est d'organiser les bascules d'un circuit en un ou plusieurs registres à décalage (ou chaînes de scan) lors du test d'un circuit. L'utilisation de plusieurs chaînes de scan plutôt qu'une seule permet de réduire le temps de test puisqu'il est inversement proportionnel au nombre de chaînes de scan. Cependant, augmenter ce nombre pose un autre problème: la transmission des données depuis le testeur. Soit on rajoute des broches au testeur pour atteindre le nombre de chaînes, ce qui a un coût non négligeable, soit on sérialise les données ce qui peut entraîner un temps de test prohibitif.

Dans ce contexte, la compression des données de test est devenue une technique communément utilisée pour relier les canaux de test disponibles sur le testeur aux chaînes de scan présentes sur le circuit sous test (CUT). Ces méthodes de compression [7-17] (dites de compression horizontale) permettent de réduire le temps de test et le nombre de broches de test nécessaires sur le testeur.

On supposera dans la suite que le circuit possède N chaînes de scan et que M broches de test sont disponibles sur le testeur, avec M < N. Le cas du test de systèmes sur puces (SoC) sera principalement étudié puisque, souvent constitué d'IP ("intellectual property") dont on ne connaît pas la description structurelle. Dans ce contexte, aucun outil ATPG (générateur de séquence de test) ou simulation de fautes ne peuvent être utilisés.

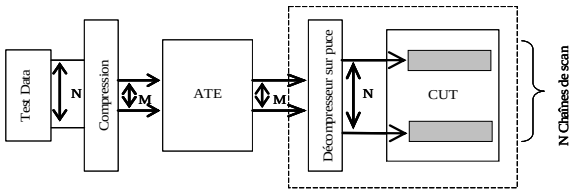

Figure 1: Schéma général de compression-décompression

Dans la suite, on appellera une tranche les valeurs d'un vecteur de test chargées dans les bascules de même rang des chaînes de scan. Chaque vecteur est composé de L tranches pour des chaînes de scan constituées de L bascules.

Une première solution pour relier les M canaux testeurs aux N chaînes de scan est de placer un décompresseur M bits vers N bits intégré sur silicium entre le testeur et le circuit sous test [2,3,4,5,9] (fig.1). Dans [2,3], un réseau de portes Xor est utilisé pour retrouver la séquence originale. Dans [5], un switch reconfigurable permet d'alimenter les N chaînes de scan avec les M canaux testeurs. Dans [9], le décompresseur est constitué d'un "ring generator" et d'un "phase shifter". Une reconfiguration statique et dynamique en ligne des chaînes de scan est proposée dans [4].

D'autres méthodes utilisent des décompresseurs séquentiels [6,7,8]. Dans [6], une architecture "circular scan" est proposée: les données alimentant les chaînes de scan sont soir les réponses au vecteur précédent, soit fournies par le testeur. Dans [7,8] des méthodes basées sur du "scan virtuel" sont présentées: ces méthodes utilisent des données générées à la fois par des éléments de test intégré et par des outils externes.

Finalement, des méthodes utilisant des dictionnaires intégrés sur silicium sont aussi utilisées pour compresser les vecteurs de test. Le testeur transmet l'adresse du vecteur désiré au décodeur qui alimente les chaînes de scan.

Toutes ces méthodes présentement un ou plusieurs des défauts suivants:

- La structure des chaînes de scan dépend du choix de la technique de décompression.
- La structure de décompression dépend des vecteurs de test.
- Les Techniques nécessitent des séquences de test spécifiquement créées.
- Elles sont basées sur de la simulation de fautes ou sur l'utilisation d'ATPG, ce qui rend ces méthodes incompatibles avec l'utilisation d'IP.

Dans ce qui suit, nous présentons une technique affranchie de ces défauts. Dans les sections 2 et 3, une nouvelle méthode de compression est présentée.

Dans les sections 4 et 5, les comparaisons théoriques et expérimentales en termes de temps de test et de volume de données sont présentées et discutées.

2. Compression – décompression.

Le principe général d'une architecture de compression-décompression horizontale pour un circuit avec N chaînes de scan (Fig.1) est d'envoyer les tranches de N bits avec seulement M canaux testeurs au décompresseur sur puce qui va régénérer exactement les tranches originales, les données compressées étant stockées dans la mémoire du testeur.

Nous proposons ici une amélioration de la méthode présentée dans [12]. L'idée principale est d'utiliser un décompresseur séquentiel constitué d'un additionneur et d'un registre à décalage (Fig.2). Le principe est d'envoyer la différence arithmétique (M bits) entre deux tranches plutôt que les tranches de N bits elles-mêmes.

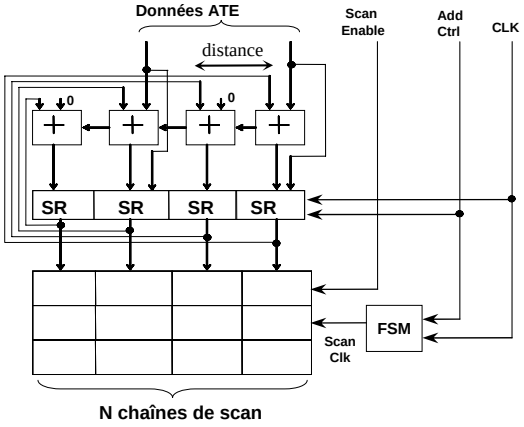

Figure 2: Architecture du décompresseur, N=4 et M=2

Soit S_i une tranche de la séquence de test et S_{i+1} la tranche suivante. La tranche compressée est notée Sci. Pour N chaînes de scan nous avons:

$S_i = a_{N-1}\ldots\ldots\ldots a_0$

$S_{i+1} = b_{N-1}\ldots\ldots\ldots b_0$

Pour M canaux testeur: $Sc_i = \mathbf{C}_{M-1}\ldots..\mathbf{C}_0$ (avec les $\mathbf{C}_i$ positionnés sur les canaux de données). Le décompresseur réalise donc l'opération:

$$\begin{array}{rl} & a_{N-1}\ldots\ldots\ldots\ldots\ldots\ldots\ldots a_0 \quad (S_i) \\ + & 0^{\cdots}0c_{M-1}0^{\cdots}0c_{M-2}0^{\cdots\cdots}0c_1 0^{\cdots}0c_0 \quad (Sc_i) \\ \hline & b_{N-1}\ldots\ldots\ldots\ldots\ldots\ldots\ldots b_0 \quad (S_{i+1}) \end{array}$$

Dans l'exemple de la figure 3, la séquence de test T est compressée en T' qui contient les Sci (N=8 et M=2). T* représente la séquence de test qui sera concrètement appliquée au circuit après décompression. L'obtention de ces séquences par compression est expliquée dans la section suivante.

Remarquons que la première tranche de T' n'est pas compressée mais chargée en série dans le registre comme décrit ci-après. Bien sûr, selon les valeurs de S_i et S_{i+1}, il peut arriver qu'aucune tranche compressée Sc_i ne puisse être trouvée. Dans ce cas, le registre est initialisé avec la tranche S_{i+1}.

T:				**T'**:			
S_1:	x x 0 0	x 0 1 1		Sc_1:	1 1 0 0	1 0 1 1	
S_2:	x 1 0 x	1 x x 0		Sc_2:	1	1	
S_3:	1 x x 0	x x x 1		Sc_3:	1	1	
S_4:	x x 1 1	x 1 x x		Sc_4:	1	0	
S_5:	1 1 x x	x x 1 0		Sc_5:	0	1	
S_6:	x 0 x 0	1 x 1 x		Sc_6:	1	0	

T*		
S_1:	1 1 0 0	1 0 1 1
S_2:	1 1 0 1	1 1 0 0
S_3:	1 1 1 0	1 1 0 1
S_4:	1 1 1 1	1 1 0 1
S_5:	1 1 1 1	1 1 1 0
S_6:	0 0 0 0	1 1 1 0

Figure 3: exemple de séquence de test compressée

Comme dans le cas du chargement série, la tranche S_{i+1} est stockée dans le testeur en tant que $\lceil N/M \rceil$ mots de M bits qui sont envoyés en série, sans passer par le l'additionneur du décompresseur.

Deux façons d'envoyer une tranche au circuit sont donc nécessaires:

- si une tranche peut-être compressée, la Sc_i correspondante est chargée en un seul cycle grâce au décompresseur (*add mode*).
- sinon, la tranche est chargée en série en $\lceil N/M \rceil$ cycles (*shift mode*).

Le choix de répartir les bits c_i de manière homogène sur les entrées du décompresseur (donc sur les entrées de l'additionneur) est fait en fonction de la propriété suivante: la probabilité d'obtenir une bit bi de S_{i+1} à partir d'un bit ai de S_i est proportionnel à $1/2^d$ (d étant la distance entre b_i et c_i). Ce choix permet d'augmenter la probabilité de trouver une tranche compressée, et donc de réduire le nombre de tranches à charger en série.

3. Compression.

Comme pour toutes les méthodes de compression, les séquences doivent contenir des X pour pouvoir être compressées. Cependant, au moment d'être appliquées au circuit, ces séquences doivent être complètement spécifiées. Le nombre de tranches non compressées et, par conséquent, le temps et le volume dépendent de cette assignation des X.

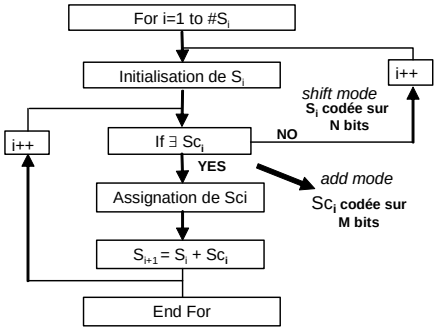

Figure 4: Algorithme de Compression

Dans un premier temps, la séquence est réorganisée de manière à ce que les colonnes contenant le plus grand nombre de bits spécifiés successifs soient alignées avec les broches d'entrée. Ces valeurs sont, en fait, les plus faciles à contrôler (d'après l'analyse de probabilité ci-dessus).

La première étape de l'algorithme (Fig.4) consiste à assigner les X de la première tranche pour l'initialiser. Le nombre de tranches compressibles dépend de cette initialisation. Cette opération est réalisée, pour chaque X, en parcourant en profondeur les tranches jusqu'à trouver une valeur spécifiée pour assigner ce X.

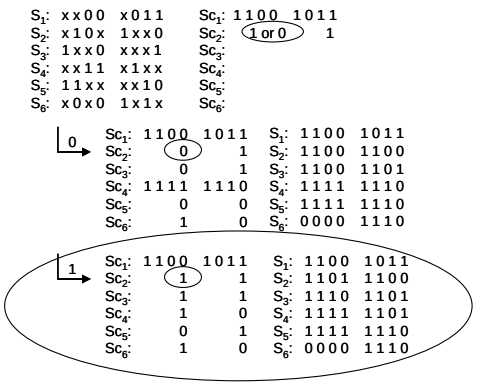

Figure 5: Algorithme de décision en profondeur

Ensuite, pour chaque tranche S_i de la séquence, l'algorithme de la figure 4 recherche un Sci tel que $S_{i+1} = S_i + Sc_i$. Comme S_{i+1} contient des X, plusieurs valeurs de Sci peuvent exister. L'assignation de S_{i+1} dépend donc de l'assignation de Sc_i qui, à son tour, a une influence sur les possibilités d'assignation de S_{i+1} et ainsi de suite. Le choix de l'assignation des Sc_i a donc une influence sur le nombre de tranches compressibles. Dans l'exemple de la figure 5, selon le choix de Sc_2, on a une ou deux tranches non compressées. Quand un Sc_i existe, un arbre de décision est parcouru pour trouver la meilleure assignation, le coût étant le nombre de tranches successibles compressibles.

4. Comparaison en temps et volume.

Dans cette section, les expressions théoriques du temps de test et du volume des données sont décrites. Ces équations sont utilisées pour comparer les résultats de la compression avec ceux de la sérialisation des données..

Pour la sérialisation, les équations 1 et 2 représentent les expressions du temps et du volume. P_1 représente le nombre de vecteurs. Dans l'équation 1, le premier terme correspond au chargement des vecteurs dans les chaînes de scan. Le second terme, lui, représente la récupération des réponses du circuit tandis que le troisième correspond aux cycles de capture de chaque vecteur. Dans l'équation 2, le volume total d'une séquence de test est le nombre final de tranches multiplié par leur taille M.

(1) $$T_1 = P_1 \cdot L \cdot \left(\left\lceil \frac{N}{M} \right\rceil + 1 \right) + L + P_1$$

(2) $$V_1 = M \cdot L \cdot \left\lceil \frac{N}{M} \right\rceil \cdot P_1$$

L'équation 3 donne l'expression du temps de test pour une séquence compressée avec notre méthode. N_{SL_C} est le nombre de tranches compressées, $N_{SL_{NC}}$ le nombre de tranches non compressées, et P_2 le nombre de vecteurs dans la séquence originale. Notons que chaque tranche compressée est chargée en un cycle dans les chaînes de scan. Une tranche non compressée, elle, a besoin de $\lceil N/M \rceil$+1 cycles pour être chargée puisqu'elle est stockée en tant que $\lceil N/M \rceil$ mots dans la séquence compressée, et est envoyée en série au circuit. L cycles sont nécessaires pour récupérer les réponses et P_2 cycles pour appliquer le contenu des chaînes de scan au circuit.

(3) $$T_2 = N_{Sl_C} + N_{Sl_{NC}} \cdot \left(\left\lceil \frac{N}{M} \right\rceil + 1 \right) + L + P_2$$

L'équation 4 donne le volume de données d'une séquence compressée. Le volume V_2 est le produit du nombre total de mots stockés dans la mémoire du testeur (compressés: N_{SL_C} et non compressés: $N_{SL_{NC}} * \lceil N/M \rceil$) par leur taille M.

(4) $$V_2 = M \cdot \left(N_{Sl_C} + N_{Sl_{NC}} \cdot \left\lceil \frac{N}{M} \right\rceil \right)$$

5. Résultats

Dans cette partie sont donnés les résultats expérimentaux sur les circuits de référence ISCAS'89. Trois types de séquences de test ont été utilisés pour comparer le chargement série à la compression. Pour cette dernière, nous avons d'abord utilisé une séquence, appelée type 1 dans la suite, contenant beaucoup de X (typiquement 96%) et donc beaucoup de vecteurs (sans option de compaction sans remplissage aléatoire) et un second type (type 2) avec moins de X et donc plus courte (option de compaction sans remplissage aléatoire). Pour effectuer une comparaison honnête, les séquences utilisées pour la sérialisation ne contiennent pas de X et sont les plus courtes (type 3: option de compaction et remplissage aléatoire). Dans ce dernier cas, la compaction est plus efficace grâce à la présence de X qui aide à la compaction dynamique des vecteurs. Les trois types de séquences parviennent au même taux de couverture de fautes. Elles ont été obtenues grâce à l'outil ATPG de Synopsis, TETRAMAX [1].

La Table 1 présente les résultats obtenus avec le circuit s5378 pour trois configurations de chaînes de scan. Les 214 bascules sont dans un premier temps organisées en 8 chaînes de scan de 27 bascules, puis en 16 chaînes de 14 et finalement en 32 chaînes de 7. Les résultats pour M variant de 5% à 50% de N sont présentés dans la Table 1. *%X* représente le pourcentage de X dans la séquence, *volume (bits)* le volume de données en bits stockées dans la mémoire testeur, *comp. Ratio* est le taux de compression atteint et *time (cycles)* est le nombre de cycles testeurs nécessaires à l'application de la séquence au circuit. Les deux dernières colonnes de la table représentent les gains en temps et volume atteints par notre compression avec les séquences de type 2 par rapport à la sérialisation des séquences de type 3.

Il faut d'abord noter que, dans tous les cas, la compression avec les séquences de type 1 n'est jamais meilleure que la sérialisation. Le bon taux de compression obtenu par notre méthode ne compense pas la différence sur le nombre de vecteurs dans les séquences. Les conditions sur le nombre de tranches à compresser pour obtenir un gain ne sont pas atteintes. Dans certains cas, par exemple dans la configuration à 8 chaînes de scan, même si toutes les tranches de la séquence de type 1 avaient été compressées, il aurait fallu 41K cycles pour les appliquer alors que 39K suffiraient à la sérialisation avec le type 3.

Par contre, si on utilise les séquences de type 2 avec la compression, les conditions sont remplies et nos résultats de compression sont nettement meilleurs que ceux obtenus par le chargement série des séquences de type 3. Les valeurs numériques de gains sont données dans les deux dernières colonnes de la Table 1. Donc, cette méthode de compression, si elle est utilisée sur des séquences de type 2, donne de meilleurs résultas que la sérialisation des séquences de type 3, même si ces

			type1: compression - 1531 patterns							type2: compression - 217 patterns							type3: serial loading - 160 p.			Gain	
L	N	M	% X	Nsl_{P2}	original volume (bits)	comp volume (bits)	comp ratio %	Nsl_c	Time (cycles)	% X	Nsl_{P2}	original volume (bits)	comp volume (bits)	comp ratio %	Nsl_c	Time (cycles)	Nsl_{P1}	Volume (bits)	Time (cycles)	Volume %	Time %
27	8	1	95,98	41337	330696	54896	83,4	39400	58391	82,57	5859	46872	13181	63,37	4813	14471	4320	34560	39067	61,87	62,96
		2				92334	72,08	39727	49335				17670	59,41	4867	10071		34560	21787	48,88	53,78
		4				168020	49,19	40669	44231				25472	44,63	5350	7121		34560	13147	26,3	45,84
14	16	1	96,12	21434	342944	43694	87,26	19950	46723	82,78	3038	48608	13988	56,41	2308	14949	2240	35840	38254	60,98	60,93
		2				61936	81,94	20072	33875				15792	55,2	2344	8821		35840	20334	55,94	56,62
		4				98936	71,15	20334	27379				19172	30,87	2453	5609		35840	11374	46,51	50,69
		8				179056	47,79	20486	24875				27328	19,39	2660	4025		35840	6894	23,75	41,62
7	32	2	96,12	10717	342944	53624	84,36	9644	29423	83,19	1519	48608	16928	47,91	1056	9151	1120	35840	19207	52,77	52,36
		3				63111	81,6	9685	23607				18177	47,15	1065	6737		36960	13607	50,82	50,49
		8				101864	70,3	10045	14943				21992	45,17	1109	3383		35840	5767	38,64	41,34
		16				180032	47,5	10182	13325				29904	33,14	1169	2443		35840	3527	16,57	30,74

Table 1: résultats pour le S5378, 214 bascules

séquences contiennent moins de X que les séquences de type 1.

D'après ces résultats, on peut déduire certaines règles sur l'influence du nombre de canaux testeurs. On peut remarquer sur la Table 1 (pour les trois configurations de chaînes de scan du s5378: N=8, 16 et 32) que les gains en temps et volume diminuent quand M augmente. En effet, la sérialisation est plus efficace, à mesure que M augmente, en terme de réduction de temps, sans augmenter le volume de données tandis qu'avec notre méthode, le volume augmente alors que le temps de test - qui dépend du nombre de tranches compressées – diminue. Donc le choix de M est le résultat d'un compromis entre la réduction du temps de test et une augmentation du volume de données.

En résumé, pour un N donné, on peut dire que M doit être choisi suffisamment grand pour réduire le temps, mais pas trop pour être sur que le volume des données ne déborde pas de la mémoire du testeur, tout en conservant de meilleurs résultats que la sérialisation. Si on garde à l'esprit que le but principal de la compression horizontale est de réduire le nombre de canaux testeurs pour la transmission des données de test, un M petit avec notre technique est toujours meilleur qu'avec la sérialisation.

D'autres expérimentations ont été menées sur d'autres circuits ISCAS 89 avec plusieurs configurations de scan. Elles ont confirmé les remarques faites à propos du s5378.

Nous avons également comparé notre méthode avec [11]. Ces deux méthodes ont en commun tout d'abord d'utiliser un décompresseur indépendant de la séquence de test. Deuxièmement, chaque tranche est une combinaison des données venant du testeur et de la tranche précédente ou de la réponse du circuit. Par exemple, pour le circuit ISCAS'89, le s15850, avec N=32 et M=6, le gain en temps par rapport à un schéma classique (M entrées pour M chaînes de scan comme décrit dans [11]) est de 54,6% et reporté dans [11], tandis que notre méthode atteint un gain en temps de 69,8%. Et pour tous les résultats présentés dans [11], notre méthode donne de meilleurs résultats. Une remarque: bien que les séquences de test utilisées pour les deux méthodes soient différentes, elles sont similaires en termes de nombres de vecteurs et de pourcentage de X dans les séquences. De plus, à l'inverse des autres méthodes de compression, notre méthode est d'autant plus efficace en termes de gain de temps que le rapport M/N est petit.

7. Conclusion

Dans cet article, une méthode simple et efficace de réduction des broches testeurs nécessaires au test d'un circuit a été présentée. Il faut noter que cette méthode est indépendante des séquences de test et des netlists des circuits, que des vecteurs de test additionnels peuvent être ajoutés à tout moment durant la production sans remettre en cause la conception des chaînes de scan et du décompresseur, et également, qu'elle ne modifie en rien le taux de couverture de fautes d'une séquence de test donnée.

Notre méthode a été comparée à la technique classique de sérialisation des données de test. Avec le même nombre de canaux testeurs, notre technique est toujours plus efficace en termes de réduction du temps de test et du volume de données. Même avec des séquences plus longues contenant des X, notre technique atteint des gains de temps entre 25 et 50% et des gains en volume entre 15 et 60 %. De plus, le décompresseur de cette technique ne nécessite que très peu d'éléments additionnels sur puce.

8. References

[1] www.synopsys.com/products/test/tetramax_ds.html

[2] I. Bayraktaroglu, A. Orailoglu, “Test volume application time reduction through scan chain concealment”, DAC'01, pp: 151-155.

[3] K.J. Balakrishman, N.A. Touba, "Reconfigurable linear decompressor using symbolic Gaussian elimination", DATE'05, pp: 1130-1135.

[4] N. Sitchinava et al., "Changing the scan enable during shift", VTS'04, pp: 73-78.

[5] H. Tang, S.M. Reddy, I. Pomeranz, “On reducing test data volume and test application time for multiple scan chain designs”, ITC'03, pp: 1079-1088.

[6] B. Arslan, A. Orailoglu, "CircularScan: a scan architecture for test cost reduction", DATE'04, pp: 1290-1295.

[7] A. Jas, B. Pouya, N.A. Touba, "Virtual Scan Chains: a means for reducing scan length in cores", VTS'00, pp: 73-78.

[8] L-T Wang et al., "VirtualScan: a new compressed scan technology for test cost reduction", ITC'04.

[9] J. Rajski et al., "Embedded deterministic test for low cost manufacturing Test", ITC'02, pp: 916-922.

[10] L. Li, K. Chakrabarty, "Test data compression using dictionaries with selective entries and fixed-length indices", ACM TODAES, Vol. 8, No. 4, October 2003, pp: 470-490.

[11] A. Würtenberger, C.S. Tautermann, S. Hellebrand, "Data compression for multiple scan chains using dictionaries with corrections", ITC'04, pp: 926-935.

[12] M-L. Flottes, R. Poirier, B. Rouzeyre, "An arithmetic structure for test data horizontal compression", DATE'04, pp: 428-433.