TP3-4
Conception d'architecture autotestable

2009-2010

Mounir Benabdenbi
Frédéric Arzel

Objectif

Ce TP est une introduction aux techniques d'auto-test plus connues sous le nom de **BIST** (Built-In Self Test).

Déroulement

En plus des outils Alliance que vous avez déja manipulé (genlib, genpat, asimut, genstil, etc.), vous disposez d'outils supplémentaires à l'emplacement suivant ~testools/archi/$(MACHINE)/

Ce sont des outils en développement (tess, genmux) ou de petits scripts qui vous simplifieront le déroulement du TP (inout2out).

Exercice 1 : Autotest de la partie contrôle de l'AMD2901

On désire pour cet exercice réaliser au final, le circuit décrit par la figure 1.

Les fichiers nécessaires sont dans ~trncomun/TP-Test/2007_2008/TP3_4/fichiers/

Un petit conseil : utilisez l'encapsulation de l'amd2901 (amd.vst): toutes les entrées (hors alimentations !) et toutes les sorties sont regroupées en deux bit_vector, ce qui facilite leurs instanciations.

Le déroulement doit être le suivant :

1) Utlisation d'un LFSR

- Conception du générateur de vecteurs de test
  - Choisir le polynôme générateur du LFSR (Linear Feed-back Shift Register)
  - Décrire la net-list structurelle du LFSR (il est conseillé d'écrire un fichier C compilé avec genlib).

- Générer un multiplexeur qui prendra en entrée soit les entrées primaires du circuit à tester soit les sorties du LFSR (utiliser genmux).

- Assembler le LFSR, le multiplexeur et l'AMD pour constituer le circuit global au format vhd.

- Ecrire le fichier genstil qui permet de générer les vecteurs de test à appliquer au circuit (.spf et .stil). Ce fichier définira trois phases de test :
  - une phase correspondant à un cycle en mode fonctionnel
  - une phase d'initialisation du LFSR
  - une phase de test correspondant à la génération par le LFSR des vecteurs de test.

- Utiliser Tetramax pour effectuer la simulation de faute du circuit assemblé en utilisant le fichier généré par genstil. Quel est le taux de couverture obtenu ?

- Augmenter le nombre de vecteurs (allonger la simulation du LFSR) et/ou changer la valeur initiale jusqu'à obtenir un taux de couverture sur le circuit proche de 90 %.
  Pour générer les vecteurs d'entrée, pourquoi n'utilise-t'on pas par exemple un compteur en lieu et place du LFSR ?

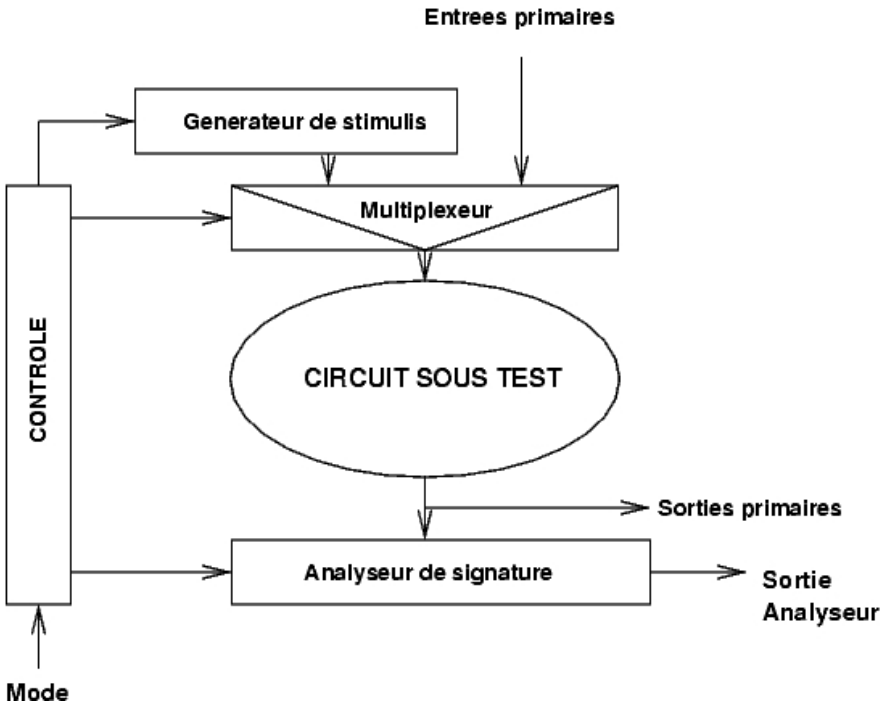

FIG. 1 Architecture traditionnelle d'un circuit BIST

2) Architecture autotestable

- Conception de l'analyseur de réponses
  - Utiliser l'outil ***tess***
  - Modifier le vbe pour que les entrées du misr soient forcées à 0 lorsqu'on se place en mode décalage. Pourquoi doit on faire cette modification ?
  - synthétiser le MISR en vst puis en vhd.

- Conception du contrôle
  - Prévoir les différents modes de fonctionnement du circuit replacé dans son contexte final.
  - Définir la logique qui va piloter le LFSR, le MISR et le multiplexeur.

- Assemblage des différents blocs: dans un premier temps, faites apparaitre les sorties primaires sur l'interface de l'assemblage.

- Réalisation d'une session de test
  - Ecrire un fichier de vecteurs qui effectuent l'ensemble du cycle de test: initialisation, fonctionnement en mode test, sortie de la signature. Les sorties primaires sont observées pendant le fonctionnement en mode test.
  - Soumettre ces vecteurs à Tetramax: quel est le taux de couverture obtenu ?
  - Quelles sont les fautes non détectées ? Qu'en déduisez vous ?

- Rôle du MISR
  - Retirer les sorties primaires de l'interface de l'assemblage (ce port devient un signal interne)
  - Recommencer l'analyse avec Tetramax (modifier le fichier genstil pour que la signature soit observée): quel est le taux de couverture obtenu ?
  Qu'en déduire sur le rôle du MISR ?

- Impact sur la fabrication
  - Calculer la surface après routage du circuit initial et du circuit auto-testable. Quel est le pourcentage de la surface ajoutée ? Expliquez.

Travail à rendre : un compte rendu dans lequel doivent apparaître les réponses aux différentes questions, les remarques et les commentaires. Précisez le chemin donnant accès à vos fichiers.