

CAO ÉLECTRONIQUE

# L'analyse formelle, outil précieux pour le débogage post-silicium

Le déploiement de la vérification formelle en débogage post-silicium autorise une analyse exhaustive du comportement du circuit. Des réponses concluantes sont obtenues à l'aide de cette technologie notamment pour connaître la source du comportement indésirable du composant. Cet article explique les bases de l'emploi de l'analyse formelle en laboratoire et détaille des exemples de la façon dont ces techniques ont été utilisées avec succès dans différents projets

La vérification des « super-circuits » intégrés actuels exige la mise en œuvre d'une combinaison de méthodologies de vérification diverses, associée à des outils les plus performants possibles. Dans cette palette, les technologies de vérification formelle ont toute leur place pendant les principales phases de la conception d'un circuit, de l'exploration architecturale jusqu'au débogage post-silicium. Et, pour cette dernière partie du flot, elles revêtent même un aspect crucial, comme l'illustrent les études de cas présentées dans cet article. Car cette approche apporte une valeur ajoutée considérable en laboratoire aux opérations post-silicium puisqu'elle permet non seulement de corriger un design mais aussi de vérifier l'exactitude de la correction.

Pour mieux comprendre les enjeux autour de cette question des bogues post-silicium, il faut prendre en compte le fait suivant : trouver des bogues lors de la phase de test du modèle est l'option la moins chère, car, par la suite, le coût d'un bogue augmente d'un facteur 10 lorsqu'il est détecté au cours des phases de test unitaire, à nouveau d'un facteur 10 lors



**JAMIL R. MAZZAWI (JASPER DESIGN AUTOMATION)**

Ingénieur d'applications chez Jasper Design Automation, Jami R. Mazzawi, est en charge du développement des méthodologies formelles au sein de la société. Titulaire d'un BSc en Ingénierie informatique de l'université d'Haifa (Israël) et d'un MBA de l'université de San Jose (USA), Jami R. Mazzawi a travaillé auparavant chez Siemens, Verisity, Rambus et Sun. Cet article a été écrit en collaboration avec Pearl Lee et Lawrence Loh de Jasper.

du test de système, et encore d'un nouveau facteur 10 lorsque le produit atteint le marché, ce qui constitue bien évidemment le pire des cas.

En conséquence, la phase de débogage post-silicium peut souvent être très stressante : que faire quand la puce ne fonctionne pas et que le temps est compté ? Car à ce niveau, ne pas respecter un délai peut aller jusqu'à, dans certains cas, mettre la société et ses employés, en péril. Alors pour se rassurer et être certains que la puce produite fonctionne parfaitement, on tente en laboratoire de reproduire les bogues constatés par le biais de simulations et d'émulations aléatoires dirigées. Mais, sou-

vent, ces approches traditionnelles s'épuisent et ne parviennent pas à déterminer la cause du bogue suffisamment rapidement. La vérification formelle s'avère ici d'un grand secours car dans ces situations elle révèle les causes profondes des bogues et valide des correctifs de manière exhaustive.

## Simulations aléatoires dirigées pour connaître la cause du bogue

Lorsque l'équipe de débogage post-silicium détecte un problème dans le circuit intégré à l'essai, il est d'abord de nature abstraite et peu claire : la puce se bloque, ne répond pas, saute des paquets de données, envoie des données erronées, etc. La première étape est

## 1.- QU'EST-CE QUE L'ANALYSE ET LA VÉRIFICATION FORMELLE ?

→ La vérification formelle est une technique purement mathématique utilisée pour s'assurer qu'un fichier de conception est conforme aux spécifications initiales. Les outils de preuve formelle utilisent trois principaux types d'entrées : le code RTL, les fichiers de contraintes liés aux limites imposées aux signaux d'entrées du circuit et les propriétés ou assertions qui sont des règles spécifiant comment le produit doit se comporter. A partir de ces trois classes de données, les outils de preuve formelle prouvent, au sens mathématique du terme, que la conception sera toujours conforme aux règles spécifiées. Ou bien

elle fournit un contre exemple qui est une trace, montrant qu'une séquence spécifique d'entrée va entraîner une violation des règles décrites par l'écriture des propriétés.

La principale caractéristique des techniques formelles, par rapport aux approches traditionnelles à base de simulation, est l'exhaustivité. Ce qui signifie qu'aucun scénario particulier de mise en faute du circuit (les « corner case ») n'est oublié. Alors qu'avec les techniques de simulation aléatoire dirigée, l'équipe de vérification espère que le moteur de simulation va bien détecter tous les « corner case ». A l'inverse, un moteur

formel va montrer que, par rapport aux propriétés qu'il examine, la conception est exempte de bogues, quels que soient les scénarios d'entrées, donc y compris les « corner case ».

Les outils modernes de vérification à base de preuve formelle sont désormais capables de traiter des portions de circuits plus grandes que par le passé (ils étaient limités à l'analyse de blocs car ils sont très gourmands en ressources de calcul) et se sont dotés de capacités avancées de débogage. Ils sont traditionnellement mis en œuvre pendant les phases de conception présilicium, au niveau du code RTL.

donc de comprendre ce qu'il se passe dans la puce.

Aujourd'hui, de nombreux circuits disposent de capacités d'extraction de trace intégrées, par exemple des contrôles du gel des fonctionnalités de la puce lorsque certains événements sont identifiés. Ils sont aussi dotés d'analyseurs logiques intégrés qui permettent à un groupe défini de signaux d'être multiplexés sur les broches de sortie du circuit. L'ingénieur de vérification a, en outre, la possibilité d'enregistrer la valeur de certains signaux, un nombre N de cycles avant un événement de gel des fonctionnalités, au sein d'une mémoire interne de la puce ou encore d'utiliser des chaînes de « balayage », pour scanner des états en virgule flottante. Compte tenu de tous ces éléments, l'équipe de débogage post-silicium peut alors extraire une trace d'erreur et capturer un nombre limité de signaux pour un certain nombre de cycles avant et après qu'un problème ait été détecté.

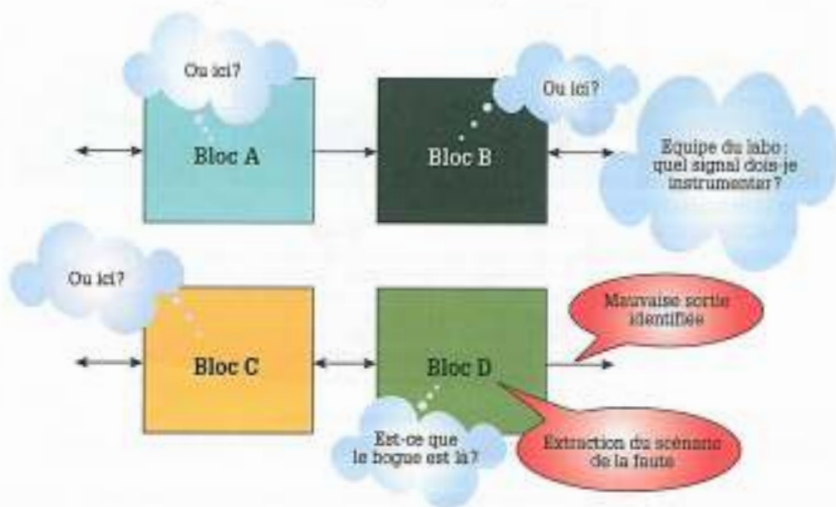
L'étape suivante consiste alors à isoler et à déterminer la cause du bogue post-silicium. A ce stade, il est établi que la puce présente un comportement interdit ou illicite comme la trace le montre, tout en gardant à l'esprit que cette dernière ne représente que les derniers N cycles d'exécution, ce qui fait que l'on ne sait pas comment cet état a été atteint. En laboratoire, les utilisateurs se trouvent, in fine, face à un dilemme : les derniers cycles du scénario d'erreur sont observés, mais comment déterminer la cause du problème ? Comment savoir dans quel bloc logique le bogue se situe-t-il ? (figure 1)

Traditionnellement, au sein d'une équipe de conception, on fait alors appel aux spécialistes des méthodes de stimulation aléatoire dirigée pour isoler le bogue et découvrir comment cet état a été atteint. Mais, dans la pratique, trouver la cause d'un bogue par le biais d'une telle technique de simulation est, dans de nombreux cas, très difficile. Car, s'il a fallu, par exemple, quatre heures de temps réel de trafic aléatoire pour détecter le bogue, combien de temps faudra-t-il pour le reproduire, sachant que le temps de simulation est mille fois plus lent ? 4000h de simulation ? Cette opération peut-elle être réalisée en une semaine ? Et pendant ce temps, la direction demande des explications sur la cause du bogue tous les deux jours !

### La vérification formelle à la rescousse

L'un des principaux avantages de l'utilisation de technologies éprouvées de vérification formelle en débogage post-silicium tient dans leur capacité à trouver les bogues rapidement. Et, habituellement, trouver des « contre-exemples » par le biais de cette approche est beaucoup plus rapide que d'obtenir des preuves complètes sur la même propriété. Un outil de

Figure 1.- La traque d'un bogue sur un SoC

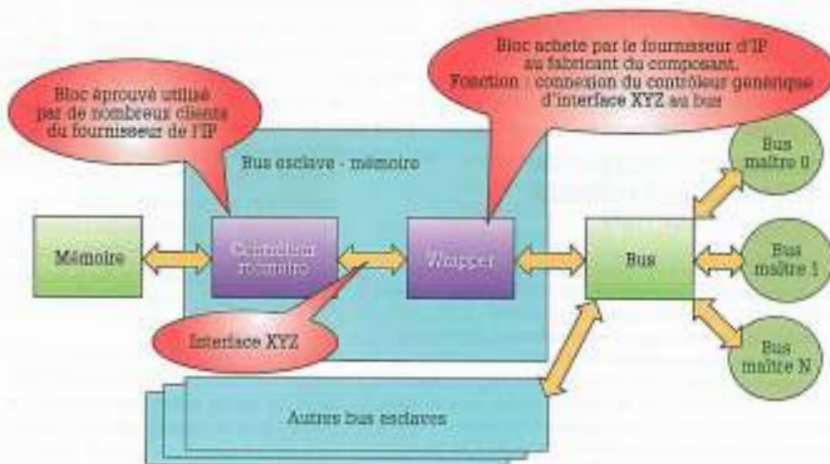


Le scénario de la faute constatée a été identifié, mais la difficulté consiste à savoir où se trouve le bogue.

vérification formelle évolué permet donc à l'utilisateur de geler un état spécifique dans un cycle spécifique, puis de continuer l'analyse à partir de ce point particulier, rendant ainsi inutile l'étude globale de la conception. La manière de trouver le bogue par le biais de la vérification formelle implique, dans la pratique, un processus similaire à celui utilisé dans un flot de vérification formelle au niveau RTL. Il existe cependant quelques différences significatives. Tout d'abord, la recherche ne

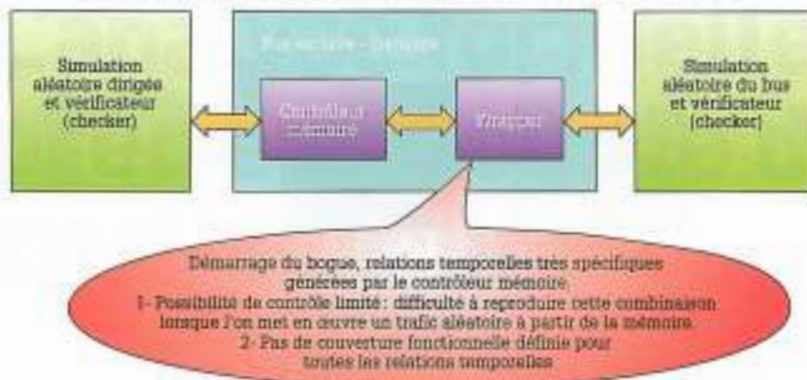
s'applique qu'à un bogue ou à un scénario spécifique. Ensuite, on ne cherche pas la preuve complète ou une couverture exhaustive. Autre différence, il suffit de trouver le scénario qui mène au comportement illégal. Enfin, cette technique permet de simplifier le processus et de ne pas recourir à des « excès » de contraintes (par exemple, de ne pas permettre d'écrire des transactions parce que le bogue ne se produit qu'avec des opérations de lecture).

Figure 2.- Bloc diagramme d'un grand SoC



Au sein de ce système intégrant un contrôleur de mémoire, il y a blocage en raison de la violation d'un protocole de bus.

Figure 3.- Recherche de bogue par simulation traditionnelle



Ici, la source du bogue est difficile à trouver par le biais de la seule simulation.

Dans un cas typique, l'équipe commence donc par ce qui est déjà connu. Il existe déjà une trace révélant le scénario illégal, et l'on sait que le problème survient lorsqu'une opération de lecture est suivie par une autre opération de lecture. Il faut donc que soit définie une propriété indiquant : - non (scénario illégal). Dans le processus décrit ici, il est inutile de contraindre la conception comme au cours d'une vérification formelle normale, ce qui simplifie le processus et permet de faire des raccourcis supplémentaires. Par exemple, il est possible de surcontraindre la conception et d'ajouter une contrainte du type : « aucune\_opération\_écrite\_permise » puisqu'on sait que le bogue s'est produit lors d'opérations de « lecture » et non pas d'« écriture ». Ensuite, il s'agit simplement de demander aux moteurs formels de prouver cette propriété, et ils remonteront jusqu'à la trace d'erreur à rebours, à partir du scénario illégal. Bien entendu, après avoir trouvé le bogue, le concepteur peut revenir en arrière, corriger son code RTL et exécuter la preuve formelle sur la même propriété pour empêcher que le

scénario illégal ne se reproduise. La possibilité de visualiser ce processus est une fonctionnalité intégrée au sein des outils de Jasper, autorisant ainsi la génération et la manipulation automatique des formes d'onde sans avoir recours à un banc d'essai.

### Catastrophe potentielle en cas de bogue d'IP

Pour illustrer la puissance du débogage post-silicium par analyse formelle, examinons quelques exemples d'équipes ayant surmonté des difficultés importantes via l'utilisation d'une technologie de vérification formelle de débogage.

Le premier exemple décrit le débogage d'un contrôleur de mémoire violant un protocole de bus (figure 2). Ce SoC doté d'un processeur et de périphériques multiples se comportait mal en condition opérationnelle, s'immobilisant sous certaines conditions. In fine, il a été rappelé par le fabricant. L'équipe de débogage post-silicium utilisait une simulation aléatoire dirigée. Elle a commencé avec le peu d'informations disponibles en laboratoire : la puce s'immobilisait et le problème était identifié comme provenant du contrôleur de mémoire alors qu'il effectuait une opération de lecture.

Les ingénieurs de vérification ont ainsi travaillé plus de trois mois jusqu'à ce qu'ils soient en mesure de déterminer la cause du bogue. Celui-ci a été identifié au sein du contrôleur de mémoire qui envoyait ses données vers le bloc de « classe enveloppante » selon un schéma très particulier qui activait un bogue

dans la « classe enveloppante » et causait une violation du protocole de bus (figure 3). Cet alignement temporel très précis des différents événements dans le système étant très difficile à atteindre par le biais de la simulation aléatoire, le bogue fut identifié comme un bogue de silicium, délicat à détecter dans les simulations post-silicium.

De toute évidence ici, la nécessité d'une période de trois mois de simulations pour déterminer la cause du bogue constitue un très gros problème pour le fabricant de la puce, ses clients et le fournisseur d'IP, d'autant plus que les produits contenant le circuit avaient déjà atteint les clients finaux, entraînant leur rappel.

Dans ce cas, il a été fait appel à la vérification formelle pour voir si cette technologie pouvait déterminer la cause du bogue plus rapidement que la simulation. Pour ce faire l'ingénieur en charge de la manipulation des outils formels a reçu exactement les mêmes informations que celles dont l'équipe de simulation disposait au départ. Et le bogue a été trouvé après seulement deux semaines et demi, et le temps gagné a été consacré à accélérer la conception et les protocoles concernés. En fait, une fois l'installation terminée et les propriétés écrites, le temps d'exécution pour trouver le contre exemple adéquat a été de moins d'une minute.

Ces résultats sont liés à la manière dont les technologies formelles fonctionnent. Car dans ce cas, elles travaillent à rebours, de manière

### III.- POUR EN SAVOIR PLUS

Un outil adapté de vérification formelle, équipé de fonctions avancées, comprend les fonctions suivantes :

- Une grande capacité de calcul afin d'exécuter une vérification formelle pour des conceptions de centaines de milliers de portes réparties sur plusieurs blocs
- La capacité à prouver l'intégrité des transferts de données tout au long d'une conception, comme dans les outils Formal Scoreboard et Proof Accelerators de Jasper.
- La visualisation pour un débogage facile et la possibilité d'affiner facilement les scénarios pour correspondre aux données issues du laboratoire
- La manipulation d'horloge asynchrone

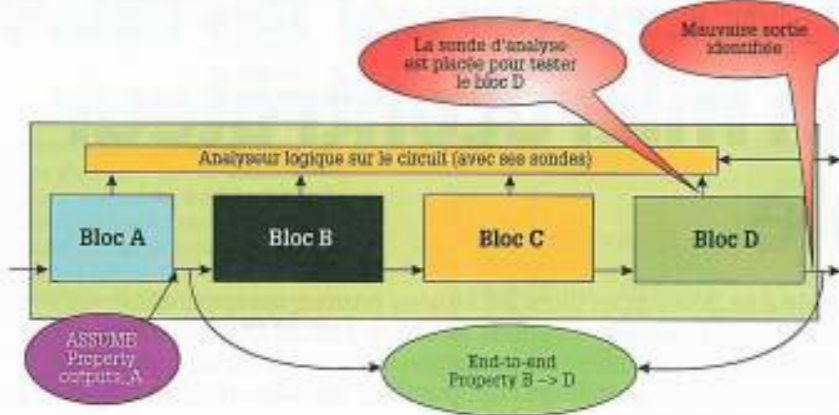
mathématique, à partir de la trace d'erreur, afin de déterminer immédiatement son origine. Alors que pour ce qui est de la simulation, les ingénieurs construisent un vérificateur afin de détecter le bogue au moment où il se produit, et utilisent la direction aléatoire pour faire plus de lectures dans l'espoir d'atteindre le bogue.

Par la suite, la vérification formelle a été réexécutée sur le code RTL corrigé et a permis de découvrir deux nouveaux bogues supplémentaires que la simulation post-silicium n'avait pas détectés, épargnant au fabricant de la puce une nouvelle mise à jour.

## II.- LES LANGAGES FORMELS POUR LA CONCEPTION

→ Les outils de preuve formelle utilisent des propriétés en entrées. Celles-ci peuvent être écrites dans trois langages différents : en System Verilog Assertion (SVA), un sous-ensemble du langage System Verilog, en Property Specification Language (PSL) ou en Open Vera Assertions (OVA), qui est une bibliothèque bâtie autour des langages SVA ou PSL. Les propriétés sont en général écrites sous un angle temporel, c'est-à-dire qu'elles décrivent des séquences d'événements sur un certain nombre de cycles d'horloge. Une fois qu'une propriété est définie elle peut être utilisée comme une assertion (au sens littéral du terme) qui doit être prouvée ou comme une supposition destinée notamment à contraindre les signaux d'entrée.

Figure 4.- Analyse formelle d'un ensemble de blocs



La rédaction d'une propriété, vérifiée par l'outil de preuve formelle, est appliquée ici de bout en bout sur les quatre blocs.

Le deuxième cas d'étude porte sur une équipe de vérification formelle qui travaillait en tandem avec des ingénieurs du laboratoire post-silicium. Une fois le bogue découvert en laboratoire, l'équipe de validation formelle a été appelée à la rescousse. La puce comprenait une série de quatre blocs logiques dans lesquels le bogue se produisait. Chaque bloc traite les données au niveau de ses entrées et les transmet au suivant. La puce avait, en outre, un analyseur logique intégré pouvant suivre un groupe de signaux d'observation au niveau de la sortie du circuit. Toutefois, pour l'équipe de laboratoire, parmi les signaux analysés celui qui révèle le problème constitue toujours une difficulté importante. Ici, puisque la trace d'erreur avait été détectée à la sortie des puces, ou aux sorties du bloc D, l'équipe de laboratoire a focalisé son attention sur ce dernier bloc.

### Un partenariat au laboratoire

L'équipe de validation formelle a d'abord écrit une propriété de bout en bout, des entrées de B jusqu'aux sorties de D (figure 4). La propriété était notamment rédigée sur la base de la trace illégale saisie en laboratoire, et seules les données entrantes causant le bogue étaient autorisées tandis que les autres étaient bloquées. En raison de la taille des trois blocs, la propriété ne convergait pas et a dû être découpée en propriétés plus petites. Ainsi, une propriété des entrées, à la sortie du bloc D, fut rédigée. Pour cette propriété, il était nécessaire d'ajouter des contraintes d'entrée afin de définir le comportement légal sur les entrées de D (figure 5). Cette

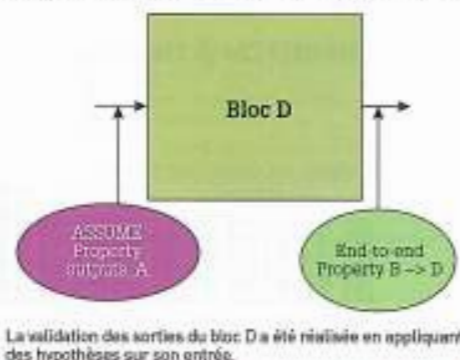
propriété a été écrite de telle manière que si les entrées de D agissaient en fonction des contraintes, le bloc D ne produirait jamais le comportement illégitime.

A partir de cette analyse, il a été établi que le bloc D était hors de cause, car exempt de bogue. Ensuite, l'équipe de débogage post-silicium a concentré son attention sur le bloc C, en réglant alors l'analyseur logique pour qu'il capte les signaux du bloc C.

En parallèle, l'équipe de validation formelle a travaillé à la vérification du bloc C : les suppositions quant aux entrées de D ont été converties en affirmations sur les sorties de C. Et de nouvelles contraintes ont été écrites pour permettre des entrées légales uniquement sur le bloc C.

Ces propriétés ont été prouvées en l'espace de quelques minutes. En utilisant la réponse exhaustive du bloc C de l'équipe de validation

Figure 5.- Ecriture de propriétés formelles



La validation des sorties du bloc D a été réalisée en appliquant des hypothèses sur son entrée.

formelle, les ingénieurs responsables du débogage post-silicium ont changé de cible, portant leur attention sur le bloc B devenu suspect. L'analyseur logique fut donc réglé pour capter les signaux de ce bloc. De cette façon, très vite, l'équipe de débogage post-silicium fut en mesure de trouver le bogue à partir de la trace extraite du bloc B. Ce cas constitue un excellent exemple de la manière dont les deux équipes peuvent travailler main dans la main, en utilisant chacune leurs points forts et en partageant l'information de l'autre équipe, afin d'isoler le bogue rapidement.

### L'analyse formelle trouve « ses » bogues

Un dernier scénario concret démontre la valeur de cette démarche pour une entreprise qui n'a pas inclus la validation formelle comme stratégie de vérification initiale. Dans cet exemple, la vérification consistait en des tests exhaustifs par le biais de la simulation et de l'émulation, sans validation formelle. Malgré tout, un bogue fut trouvé en laboratoire sur la puce. Des limites inhérentes à la simulation et l'émulation ont fait que ce bogue n'a pas été détecté car il s'agissait d'un cas d'exception dépendant du cycle.

Une modification du circuit a été réalisée (ECO, Engineering Change of Order), suivi de plusieurs semaines de simulation et d'émulation afin de confirmer que la correction était exacte. Par mesure de précaution, la validation formelle fut utilisée afin que le résultat soit doublement vérifié. En seulement quelques heures, il fut déterminé que le correctif était défectueux ! Sans même faire de banc de test, l'outil de validation formelle a trouvé un autre scénario susceptible d'entraîner le bogue et a établi que le correctif ne couvrait pas tous les cas.

Un deuxième correctif, examiné par validation formelle, a conduit à la découverte qu'à un niveau différent de la logique, le même bogue pourrait se produire, ce qui a permis d'obtenir un troisième correctif passant le test de la validation formelle sans aucun problème.

Dans cet exemple, il faut retenir que l'examen formel a épargné à l'entreprise de multiples mises à jour, car la simulation n'aurait pas pu détecter les bogues après avoir réexécuté les correctifs.

In fine, on peut affirmer qu'un bon outil de vérification formelle fournit un ROI (retour sur investissement) important en laboratoire, permettant d'économiser énormément de temps et d'argent. Et la solution adaptée consiste à intégrer l'analyse formelle comme partie prenante du processus de conception, afin d'extirper rapidement les bogues. Mais elle peut aussi être utilisée de concert avec des méthodologies post-silicium traditionnelles.