J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes,
Volume 4, Hors-Série 4, 8 (2005)
DOI : http://dx.doi.org/10.1051/bib-j3ea:2005858

Conception sûr de circuit numérique

J.L. Boulanger

Université de Technologie de Compiègne
Laboratoire HEUDIASYC, UMR CNRS 6599
F-60205 Compiègne CEDEX, France

Conception sûr de circuit numérique

Jean-Louis BOULANGER
Université de Technologie de Compiègne
Laboratoire HEUDIASYC, UMR CNRS 6599,
60205 Compiègne Cedex, France
Tel : 03 44 23 44 23, Fax : 03 44 23 44 77
E-mail : jean-louis.boulanger@utc.fr

Résumé

Initialement cette étude constituait un simple exercice de style ayant pour objectif d'évaluer la modélisation de circuits numériques simples à l'aide de la méthode B. Cette expérimentation était essentiellement motivée par la perception d'une analogie forte entre le domaine cible et les principes de la méthode B. Les circuits numériques sont de plus en plus complexes tant du point de vue de l'intégration que du point de vue des fonctions traitées. Actuellement, la principale activité de validation des circuits numériques consiste à réaliser une campagne de test. Il ne nous semble pas évident de faire front à cette complexité uniquement au travers d'activités de test. Le but de cet article est de présenter une approche méthodologique basée sur le couplage d'une conception $VHDL$ et sur une vérification par preuve formelle basée sur la méthode B.

Keywords : Circuits numériques, Modélisation, Méthode Formelle, Méthode B, Preuve, Raffinement, $VHDL$.

Introduction

Dans le cadre des systèmes dits sûrs de fonctionnement (centrale nucléaire, avionique, ferroviaire, ..), la principale problématique consiste à garantir un maximum de sécurité dans le fonctionnement de l'application. Cette sécurité est induite par les applications, mais aussi par les architectures matérielles. Il faut alors mettre en place des processus de développement, de vérification et de validation qui permettent de démontrer que certains objectifs sont atteints par le système réalisé. Afin d'atteindre ce but nous proposons de mettre en oeuvre un processus basé sur des méthodes formelles. Les méthodes formelles sont en plein essor notamment dans les applications critiques telles les centrales nucléaires, l'avionique ou le transport ferroviaire ([7] et [4]). L'apport des méthodes formelles est d'offrir un cadre mathématique au processus de développement, ce qui permet de disposer d'une méthode permettant la production de logiciels corrects par construction grâce à un processus de développement vérifiable par des techniques de validation telles que la preuve ou l'exploration de modèle (model-checking). Pour cela, il faut évidemment décrire de façon précise les propriétés que le système informatique doit avoir. Dans le cadre du projet B-HDL nous proposons donc de combiner le langage $VHDL$ (validation basée sur le test) et la méthode B (validation basée sur le test). Dans le cadre de cet article trés général, nous ne présenterons pas de propriété dite de sécurité, mais un processus permettant de développer des circuits sûr de fonctionnement.

1 *VHDL*

La conception de circuits met en œuvre plusieurs techniques telles que l'abstraction, la décomposition et la réutilisation. A titre d'exemple, la figure 1 présente une vue abstraite d'un circuit et sa décomposition.

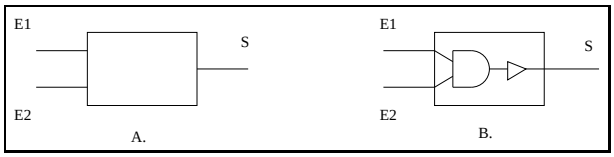

FIG. 1 – Le composant *NONET* en version graphique

Le langage $VHDL$[1] ([2]) permet de décrire rigoureusement le comportement d'un circuit numérique. $VHDL$ est un langage normalisé depuis 1987, créé spécifiquement pour la conception matérielle et la description de systèmes électroniques logiques et analogiques. Le langage $VHDL$ permet une description comportementale des fonctions d'un circuit, laissant le soin au synthétiseur de générer les structures électroniques qui réaliseront matériellement ces fonctions. Il a pour avantage de pouvoir être simulé en vue d'une validation fonctionnelle de la description, puis d'être synthétisé en vue de sa réalisation sous forme d'ASIC ou de composants programmables. Ce langage peut servir également à décrire des systèmes matériels à l'aide d'un haut niveau d'abstraction, il s'applique aux cartes de composants mais également à des systèmes entiers comprenant à la fois des parties matérielles et logicielles. Il permet enfin de décrire des algorithmes destinés à être implantés de manière physique. $VHDL$ propose à la fois des instructions d'un langage de programmation comme les boucles, mais également des termes descriptifs de bas-niveau tels que par exemple 'wait' (qui spécifie le retard d'un signal) en réponse aux besoins de conceptions de circuits digitaux complexes avec des contraintes temporelles fortes.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all
ENTITY NotAnd IS -- broches d'entrée et de sortie
 PORT (in1,in2: IN bit; out: OUT bit);
END;
ARCHITECTURE comportement OF NotAnd IS -- description de la fonction réalisée
 out <= not(in1 and in2);
END comportement;
```

FIG. 2 – Un exemple de vue comportementale d'un cirduit

Comme le montre la figure 2, tout circuit peut être divisé en blocs possédant des entrées, des sorties et réalisant une fonction bien définie. Chaque bloc peut alors être vu comme une boîte noire. C'est la première étape dans la rédaction du code. Chaque bloc est déclaré au travers d'une entité ($ENTITY$). A chaque bloc correspond la description de son comportement. C'est dans cette étape que les avantages de $VHDL$ sont les plus notables. L'association

[1] VHSIC Hardware Description Language

d'une fonction à un bloc se fait en définissant une architecture ($ARCHITECTURE$).

```
ARCHITECTURE structure OF NotAnd IS -- déclarations de modèles de composants
COMPONENT et
  PORT(in1,in2:IN bit; out:OUT bit);
END COMPONENT;
COMPONENT inv
  PORT(in:IN bit; out:OUT bit);
END COMPONENT;
SIGNAL f: std_logic; -- déclarations de signaux internes (fils)
BEGIN -- instanciation et câblage des composants (schéma)
  aa: And PORT MAP (x,y,f)
  nn: Not PORT MAP (f,s);
END structure;
```

FIG. 3 – Un exemple de vue structurelle d'un cirduit

L'approche structurelle permet de décrire le fonctionnement d'un circuit en fonction de ses composants physiques, voir la figure 3. La description du fonctionnement des composants physiques eux-mêmes doit alors exister dans une bibliothèque prédéfinie ou définie par le programmeur. Il faut se représenter les composants comme les portes du circuit de la figure 1 et les signaux comme les fils qui relient chacune des portes.

2 Méthode B

2.1 Principes

La méthode B, développée par Jean-Raymond ABRIAL[1], est une méthode basée sur les modèles au même titre que VDM et Z. Elle consiste essentiellement à modéliser un système par des entités ¡¡variables¿¿ sur lesquelles nous agissons par l'intermédiaire d'opérations permettant d'accéder ou de modifier les valeurs associées aux entités. La cohérence de l'évolution du système est assurée par la préservation de propriétés invariantes. La vérification de cette préservation étant assurée par la génération d'obligations de preuves reliant ces propriétés aux actions spécifiées dans les opérations. L'intéret majeur de la méthode B est sa prise en compte, dans un seul formalisme, du développement incrémental d'une spécification (mathématique), de ses raffinements successifs, jusqu'à l'implantation (informatique) correspondante. Les machines abstraites sont composées de trois parties : la partie déclarative, la partie de composition et la partie exécutive.

- La partie déclarative permet de décrire l'état d'une machine abstraite au travers de variables, de constantes, d'ensembles, et surtout de propriétés que doit toujours vérifier l'état de la machine. Cette partie est basée sur la théorie des ensembles et les prédicats du premier ordre. Nous pouvons dès lors parler de "modèle" à état.
- Les clauses de composition (SEES, INCLUDES, IMPORTS et EXTENDS) permettent de décrire les différents liens entre machines abstraites, chaque clause introduisant des règles de visibilité et d'accessibilité sur l'état et les opérations de la machine abstraite concernées.

– La partie exécutive, quand à elle, contient l'initialisation et les opérations de la machine abstraite, elle est basée sur le langage des substitutions généralisées (dit GSL).

2.2 De $VHDL$ à B

Le passage au modèle est assez simple. Les valeurs des ports *in* et *out* sont mémorisées par des variables globales qui ne sont accédées que par les opérations définies dans le modèle. Chaque port de sortie est associé à une opération de lecture et chaque port d'entrée est associé à une opération d'écriture. Les opérations seront utilisées par l'environnement du circuit pour placer ou lire une valeur (logique) sur un port. Par souci de lisibilité, il est commode de nommer les opérations associées à un port par homonymie (opération In pour le port *in*) avec un éventuel indice lorsque qu'il y a plusieurs ports analogues.

```
MACHINE  B_NotAnd_0
SEES     IEEE.B_STD_LOGIC_1164_0 ,B_STD_ENTRIE
DEFINITIONS
 Compute_(xx,yy,zz) ==
 (zz = bool_to_bit(bool(not((IS_X_VALUE(xx)=TRUE)&(IS_X_VALUE(yy)=TRUE)))))
VARIABLES       in1,in2,out
INVARIANT       in1:BIT & in2:BIT & out:BIT & Compute_(in1, in2, out)
INITIALISATION
in1, in2,out :( in1:BIT & in2:BIT & out:BIT & Compute_(in1, in2, out))
OPERATIONS
In_1 ( val ) =
PRE val:BIT THEN in1,out:(in1:BIT & in1=val & out:BIT & Compute_(in1,in2,out))
END;
In_2 ( val ) =
PRE val:BIT THEN in2,out:(in2:BIT & in2=val & out:BIT & Compute_(in1,in2,out))
END;
val <-- Out =  val := out;
END
```

FIG. 4 – Version B de la vue comportementale

Le modéle B, ici proposé au niveau le plus abstrait, est intrinsèquement non-déterministe. La forme purement logique de la fonction à réaliser (`Compute_`). permet de ne pas spécifier la chronologie des changement de valeurs pour les ports. Nous spécifions uniquement que, suite aux différentes invocations d'opérations, certaines valeurs seront modifiées avec la contrainte permanente du respect de l'expression logique `Compute_` De même, l'initialisation des valeurs présentes sur les ports n'est pas (encore) précisée hormis les types attendus (ici des variables booléennes). La figure 5 présente une traduction de la vue structurelle en B. Il s'agit d'une implantation. Le processus de preuve permet de démontrer qu'il s'agit bien d'un raffinement de la vue comportementale. Outre sa relative simplicité sur la plan théorique, la force de la méthode B réside également dans la disponibilité d'ateliers logiciels qui prennent en compte chaque phase du développement B.

```
IMPLEMENTATION  B_NotAnd_n
REFINES         B_NotAnd_0
SEES            IEEE.B_STD_LOGIC_1164_0,B_STD_ENTRIE
IMPORTS         AA.B_And_0,NN.B_Not_0
INVARIANT       in1=AA.in1 & in2=AA.in2 & AA.out=NN.in & out=NN.out
INITIALISATION
VAR xx IN AA.In_1(OO); AA.In_2(OO) ; xx <-- AA.Out ; NN.In(xx) END
OPERATIONS
In_1 ( val ) = VAR xx IN AA.In_1(val); xx <-- AA.Out; NN.In(xx) END;
In_2 ( val ) = VAR xx IN AA.In_2(val); xx <-- AA.Out; NN.In(xx)END;
val <-- Out  = val <-- NN.Out;
END
```

FIG. 5 – Implémentation de la vue structurelle

3 Méthodologie

La faisabilité de la modélisation de circuit numérique en B a déjà été faite voir [8]. Dans le cadre de [6], nous avons présenté les bases de la méthodologie permettant de plonger un modèle $VHDL$ au sein de la méthode B. Ce plongement est réalisé au travers de la mise en place d'un ensemble de machines abstraites qui permet de disposer, dans le monde B, des concepts introduits au sein de la bibliothèque $IEEE\ STD\_LOGIC\_1164$. Nous nous limitons actuellement au traitement des circuits numériques synchronisés.

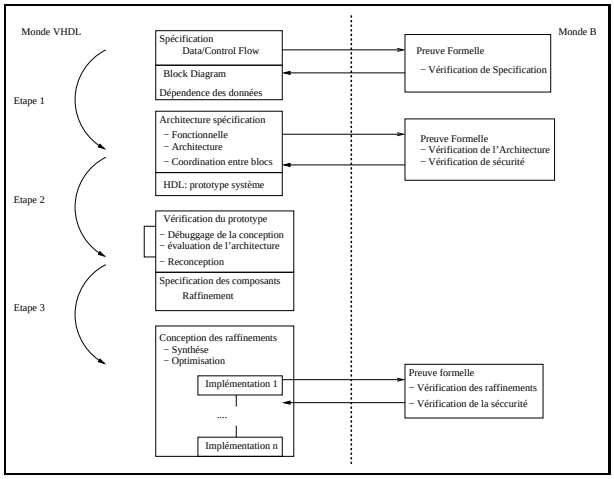

FIG. 6 – Processus

Nous proposons de mettre en œuvre un processus qui couple la conception $VHDL$ et la validation par preuve des modèles B générés. La figure 6 résume ce processus. Un premier outil intitulé $B - HDL$ basé sur ses principes a été présenté dans [3], cet outils est basé sur l'environnement $VHDL - GUI$. L'environnement $VHDL - GUI$ permet la conception graphique de circuit et la génération de code $VHDL$. Il est complété par un convertisseur VHDL/B. Un exemple de taille plus importante est présenté dans [5].

4 Conclusions

Dans le cadre de cet article, nous avons montré au travers d'un exemple comment il est possible de passer d'un modèle $VHDL$ à un modèle B. Ce passage permet d'introduire le concept de preuve formelle au sein du développement de circuits numériques. Ces preuves permettront de démontrer qu'un circuit numérique vérifie certaines exigences. Dans le cadre des circuits dit de sécurité, la difficulté réside dans la démonstration (actuellement basée sur la notion de test) qu'un certain nombre de propriétés (notamment des propriétés de fail-safing) sont garanties par la conception.Plusieurs actions de recherche complémentaires sont en cours, on trouve parmi elle une extension du langage B au calcul des durées. Cette extension devrait permettre de prendre en compte les circuits temporisés.

Références

[1] Jean-Raymond Abrial. *The B Book - Assigning Programs to Meanings*. Cambridge University Press, August 1996.

[2] R. Airiau, J.-M. Bergé, V. Olive, and J. Rouillard. *VHDL – Langage, modélisation, synthèse*. Collection Technique et scientifique des télécommunications. Presses Polytechniques et universitaires romandes, 1998.

[3] Ammar Aljer, Philippe Devienne, Sophie Tison, Jean-Louis Boulanger, and Georges Mariano. Bhdl : Circuit design in B. In Ricardo J. Machado Johan Lilius, Felice Balarin, editor, *ACSD, Third International Conference on Application of Concurrency to System Design*, pages 241 – 242, Guimaraes, Portugal, 18-20 June 2003. IEEE. The main goal of this project is to provide a method of correct design of digital circuit.

[4] Patrick Behm. Développement formel des logiciels sécuritaires de METEOR. In Henri Habrias, editor, *Proceedingsof 1st Conference on the B method*, Putting into Practice methods and tools for information system design, pages 3–10, 3 rue du Maréchal Joffre, BP 34103, 44041 Nantes Cedex 1, November 1996. IRIN Institut de recherche en informatique de Nantes.

[5] Jean-Louis Boulanger. BHDL - an example. In *SCI 2004 - The 8th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA*, volume IX, pages 150–155. International Institute of Informatics and Systemics, July 18-21 2004.

[6] Jean-Louis Boulanger, Georges Mariano, and Ammar Aljer. Conception sûre de circuits basée sur la notion de propriété. In *ICSSEA'2001 – 14th Int. Conf. on Software Systels Engineering and Their Applications*, volume 2, CNAM, Paris, France, December 2001.

[7] M. G. Hinchey and J. P. Bowen, editors. *Applications of Formal Methods*. Series in Computer Science. Prentice Hall International, 1995.

[8] Georges Mariano and Jean-Louis Boulanger. Modélisation formelle de circuits numériques par la méthode B. Rapport Technique 1999-25, INRETS-ESTAS, INRETS-ESTAS, 20 rue Elisee Reclus,59650 Villeneuve d'Ascq, December 1999.