OCaml sur circuit FPGA

Type d'article : démonstration d'outil

Jocelyn Sérot[1] and Emmanuel Chailloux[2]

[1] *Institut Pascal, UMR 6602 UCA/CNRS/SIGMA*
jocelyn.serot@uca.fr
[2] *Sorbonne Université, CNRS, LIP6, F-75005 Paris, France*
emmanuel.chailloux@lip6.fr

Résumé

On propose une démonstration d'O2B (OCaml On Board), un prototype d'infrastructure logicielle permettant l'exécution de programmes OCaml sur des circuits reprogrammables de type FPGA de la famille Intel. La démonstration s'effectuera avec la chaîne de développement Quartus Lite disponible gratuitement sur le site d'Intel [1]et une carte DE10-Lite (embarquant un FPGA Intel Max 10) de Terasic. L'ensemble des outils utilisés et les exemples utilisés sont téléchargeables à l'adresse https://github.com/jserot/O2B.

1 Introduction

Largement utilisés depuis un vingtaine d'années par la communauté des concepteurs de circuits numériques, les architectures de type FPGA (Field Programmable Gate Array) font désormais l'objet d'un intérêt croissant de la part des programmeurs au sens large. En offrant la possibilité de « tailler sur mesure » le matériel aux besoins du logiciel, ce type d'architecture offre en effet un moyen inédit de contourner les limitations des architectures de processeurs classiques. La programmation de ce type de circuit, dans l'état de l'art, passe toutefois par l'usage de langages dédiés à la description de matériel comme VHDL ou Verilog, à faible niveau d'abstraction et difficiles à utiliser par des programmeurs non familiarisés avec les architectures matérielles [4]. Un certain nombre de travaux ont cherché à utiliser des langages fonctionnels pour pallier cette difficulté. Deux voies ont été explorées. La première consiste à transformer directement le programme en une représentation au niveau *transfert de registres* (RTL), description intermédiaire sous la forme de portes logiques et de registres de mémorisation, classiquement utilisée en amont des outils de synthèse physique sur les circuits. C'est l'approche suivie dans les projets Lava [3], Cλash [2] (pour le langage Haskell) et HardCaml [2] (pour le langage OCaml). La seconde consiste à exécuter une machine virtuelle, interprétant le *bytecode* résultant de la compilation du programme, sur un ou plusieurs processeurs *softcore* implémenté sur le FPGA. Cette machine virtuelle peut alors être décrite classiquement en C. C'est l'approche suivie par le projet Hume [1] et l'outil dont on fera la démonstration.

2 Description

L'infrastructure logicielle mise en place est décrite sur la figure 1. Les programmes OCaml sont d'abord compilés en *bytecode* par le compilateur `ocamlc` puis le *bytecode* produit est transformé en un tableau C par l'outil `bc2c` de l'environnement OMicroB [5, 6] qui définit en C une machine virtuelle OCaml et son *runtime*. L'exécutable généré embarque alors la machine virtuelle OCaml et le *bytecode* du programme. Il est associé aux fonctions du *Board Support*

1. https://fpgasoftware.intel.com
2. https://github.com/janestreet/hardcaml

Package donnant accès aux ressources matérielles de la carte cible et compilé en un code binaire pour le processeur *softcore* instancié sur le FPGA. L'architecture de ce processeur, ainsi que le nombre et la nature des périphériques associés, est définie séparément. Dans notre cas, on utilise l'outil QSys de la chaîne Quartus. Cette étape génère un ensemble de fichiers VHDL qui constituent la description de la plateforme matérielle. C'est cette description, une fois synthétisée, toujours avec les outils de la chaîne Quartus, qui est utilisée pour configurer le FPGA (et en particulier instancier le processeur *softcore*, un Nios2 dans notre cas).

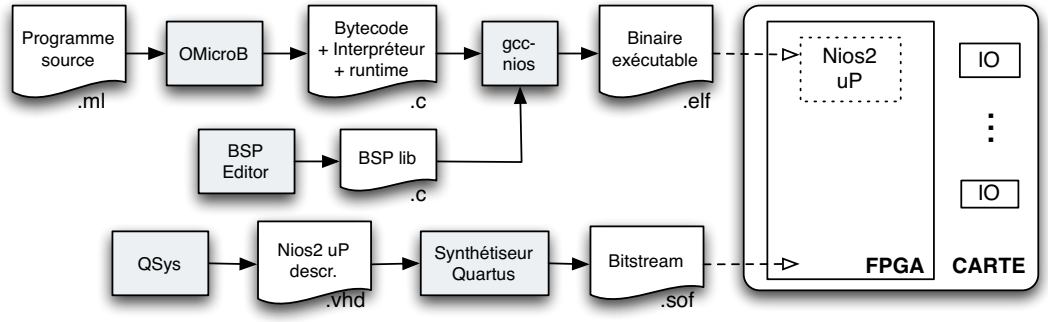

Figure 1 – Infrastructure logicielle de l'outil

Utilisée telle quelle, l'infrastructure décrite ci-dessus permet déjà d'augmenter sensiblement le niveau d'abstraction des programmes. Les opérations accédant aux dispositifs d'entrées-sorties peuvent en particulier être encapsulées au sein de modules pour une plateforme donnée. Plusieurs programmes mettant ces possibilités en évidence et s'exécutant sur une plateforme matérielle constituée d'une carte DE10-Lite – une cible d'entrée de gamme embarquant un FPGA de type Max10 (50000 blocs logiques, 2 Mo RAM interne, 128 Mo SDRAM externe) – sont disponibles sur le site d'O2B et seront présentés lors de la démo.

Il est aussi possible de définir des plateformes matérielles dédiées au sein desquelles certaines opérations du *bytecode* sont implémentées soit directement en C (via la mécanisme d'appel de fonctions externes d'OCaml) soit sous la forme de *custom instructions* du processeur Nios2, décrites au niveau RTL (en VHDL par exemple) et implémentées en matériel au niveau porte sur le FPGA. On montrera comment cette seconde approche, sur un simple programme de calcul du PGCD permet d'accélérer de manière considérable son exécution, en ramenant le nombre de cycles d'horloges requis par itération de l'algorithme de 4300 à 1.

3 Perspectives

Même s'il ne s'agit à ce stade que d'une simple démonstration de faisabilité, le travail illustré ici ouvre de nombreuses perspectives. On évoquera ainsi deux extensions possibles et riches d'applications. La première consiste à dériver automatiquement le code VHDL RTL des fonctions à accélérer à partir du programme OCaml (ce code est actuellement écrit à la main et intégré à la plateforme via l'outil QSys). La seconde consiste à instancier sur le FPGA non pas un seul processeurs *soft core* mais plusieurs, afin d'exécuter les programmes de manière parallèle.

Références

[1] Wim Vanderbauwhede Abdallah Al Zain et greg Michaelson. Hume to fpga. In P. Kitsos, editeur, *Proceedings of 10th International Symposium on Trends in Functional Programming*, University of Oklaoma, May 2010.

[2] Marco Egbertus Theodorus Gerards, C.P.R. Baaij, Jan Kuper, et Matthijs Kooijman. Higher-order abstraction in hardware descriptions with clash. In P. Kitsos, editeur, *Proceedings of the 14th EUROMICRO Conference on Digital System Design, DSD 2011*, pp. 495–502, United States, August 2011. IEEE Computer Society.

[3] Mary Sheeran Per Bjesse, Koen Claessen et Satnam Singh. Lava : Hardware Design in Haskell. In *Proceedings of the third ACM SIGPLAN International Conference on Functional Programming*, ICFP'98, pp. 174–184. ACM, 1998.

[4] Jocelyn Sérot. *La programmation des circuits FPGA et le langage VHDL. Une introduction pour les programmeurs et par l'exemple*. Ellipse, 2019.

[5] Benoit Vaugon Steven Varoumas, Basile Pesin et Emmanuel Chailloux. Programming microcontrollers through high-level abstractions. In *Proceedings of the 12th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages, VMIL@SPLASH 2020*, 2020.

[6] Steven Varoumas. *Modèles de programmation de haut niveau pour microcontrôleurs à faibles ressources. (High-level programming models for microcontrollers with scarce resources)*. PhD thesis, Sorbonne University, Paris, France, 2019.