Sujet de stage (niveau Master, 2020-2021)
Extension "virgule fixe" du compilateur Faust

Tanguy Risset[1], Florent de Dinechin[1], Yann Orlarey[2], Stephane Letz[2], and Romain Michon[2]

[1]Citi/Socrate, Insa-Lyon,
Inria `https://team.inria.fr/socrate/team-members/`
[2]Grame, Lyon `https://faust.grame.fr/`

Faust [OLF09] (Functional Audio Stream, `https://faust.grame.fr`) est un langage de programmation spécifiquement conçu pour les applications de traitement du signal en temps réel pour l'audio. Le compilateur Faust permet de développer rapidement des logiciels efficaces et fiables pour les principales plates-formes audios (plugins VST et AU, Max/MSP, PureData, applications macOS, iOS, Android, Web, Linux, etc.). Faust a été développé par le GRAME, notamment par Yann Orlarey et connaît depuis quelques années un grand succès autant dans la communauté de recherche en informatique musicale, qu'au niveau des développeurs professionnels (Moforte, Analog Devices) ou du grand public (ex : GeekBagatelles - `http://www.grame.fr/prod/geek-bagatelles`).

Le projet Syfala [RMO+20] (`https://faust.grame.fr/syfala/`) est une collaboration entre le Citi et le Grame pour mettre en place une chaîne de compilation `faust2FPGA` qui permettrait de faire des traitement audio avec une très faible latence (quelques échantillons, moins de $50\mu s$). Ce projet comporte plusieurs problèmes intéressants à traiter comme par exemple l'utilisation efficace des outils industriel de synthèse de haut niveau (HLS).

Un des problèmes à résoudre pour avoir une telle chaîne de compilation est de permettre à Faust de générer une arithmétique en virgule fixe plutôt qu'en virgule flottante. Autant la virgule flottante est un bon choix pour une cible de type CPU qui possède une unité de virgule flottante, autant pour un FPGA il est beaucoup plus efficace d'avoir des opérations en virgule fixe, d'autant que l'on peu *tailler* la taille (en nombre de bits, on dit *largeur*) des entiers manipulés.

La passage en virgule fixe est cependant un problème qui est loin d'être simple à partir du moment ou l'on a de la latitude sur la largeur des entiers. Une plus petite largeur prendra moins de place sur le FPGA mais perdra en précision. Le problème doit donc être étudié de manière assez profonde et l'équipe Socrate du Citi a déjà abordé ce genre de problème pour les filtres récursifs notamment [VIDH19].

L'objectif de ce stage est donc de rajouter une option de compilation au compilateur Faust, lui permettant de générer des opérations en virgule fixe plutôt qu'en virgule flottante. Cette extension est essentiellement ciblée vers la génération de code pour FPGA (mais elle pourra se montrer utile pour les micro-controleurs n'ayant pas d'unité flottante aussi), elle sera donc validé dans la chaîne de compilation prototype Syfala utilisant les outils `Vivado` de `Xilinx`. Après avoir étudié les différentes bibliothèques pouvant être utilisées pour la virgule fixe, on mettra en place une première génération "simple" ou tous les entiers auront la même largeur, et on évaluera le gain (en place) et les pertes (en précision) des architectures FPGA générées.

Ensuite, on s'attaquera au problème difficile de trouver les meilleures largeurs pour chaque variable utilisée dans le programme (le fameux paradigme *computing just right* [VIDH19]). L'idée est que ce processus soit le plus transparent possible à l'utilisateur de Faust. De telle méthodologies existent partiellement mais le contexte particulier du traitement audio pourra sans doute simplifier le problème général.

On pourra enfin reprendre ce travail en essayant d'utiliser les techniques de l'état de l'art pour l'implémentation optimisée au niveau bit de filtres [VIDH19] et de sommation [BdDI+13]. On s'intéressera alors à l'utilisation du logiciel FloPoCo [dDP11] (`http://flopoco.gforge.inria.fr/`), développé par Florent de Dinechin, pour avoir plus de souplesse sur la manière dont les opérateurs virgule fixe du programme généré par le compilateur Faust seront implémentés.

L'idée du stage est de valider chaque étape sur une plate-forme FPGA avec des exemples réels, il y a donc un travail non négligeable de prise en main de la chaîne `faust2FPGA` et de son utilisation sur une ou plusieurs cartes FPGA. On appliquera ces optimisations notamment aux filtres multimodaux sur FPGA. Les filtres multimodaux permettent la reconstruction d'une ambiance sonore par la juxtaposition de nombreux petits filtres simples (biquad). Leur utilisation massive produit un niveau type de filtre, dit "modal" beaucoup plus paramétrable que les filtres classiques. Leur utilisation sur FPGA permettrait par exemple de faire du contrôle actif d'acoustique de salles.

Si cela est nécessaire, le stage pourra se dérouler en télétravail. Il donnera lieu à gratification.

Déroulement du stage Le candidat sera basé à Lyon, dans l'équipe Socrate (CITI, INSA), qui travaille sur FloPoCo, l'arithmétique pour FPGA, et les systèmes très basse consommation, et se fera en collaboration forte avec l'équipe des chercheurs de GRAME responsable du développement de Faust.

Les compétences principales demandées sont une formation de base en compilation et des notions de traitement du signal pour l'audio en particulier. Un goût pour la programmation embarquée et la programmation de FPGA sera apprécié. La maîtrise du langage C est indispensables.

Références

[BdDI+13] Nicolas Brunie, Florent de Dinechin, Matei Istoan, Guillaume Sergent, Kinga Illyes, and Bogdan Popa. Arithmetic core generation using bit heaps. In *Field-Programmable Logic and Applications*, September 2013.

[dDP11] Florent de Dinechin and Bogdan Pasca. Designing custom arithmetic data paths with flopoco. *IEEE Design & Test of Computers*, 28 :18–27, 04 2011.

[OLF09] Yann Orlarey, Stéphane Letz, and Dominique Fober. *New Computational Paradigms for Computer Music*, chapter "Faust : an Efficient Functional Approach to DSP Programming". Delatour, Paris, France, 2009.

[RMO+20] Tanguy Risset, Romain Michon, Yann Orlarey, Stéphane Letz, Gero Müller, Adeyemi Gbadamosi, Luc Forget, and Florent de Dinechin. Faust2fpga for ultra-low audio latency : Preliminary work in the syfala project. In *International Faust Conference (IFC)*, 2020.

[VIDH19] A. Volkova, M. Istoan, F. De Dinechin, and T. Hilaire. Towards hardware iir filters computing just right : Direct form i case study. *IEEE Transactions on Computers*, 68(4) :597–608, April 2019.