DATE: 2021/12/1

Department of Electrical and Electronic Information Engineering		ID	M183211
Name	Kazuki Iwahara		

Supervisor	Shuichi Ichikawa

Abstract

Title	Re-evaluation of a dedicated instruction implementation method using high-level synthesis

High-level synthesis (HLS) is a technique that converts a behavioral description in a high-level language such as C into an RTL description in a hardware description language. Skalicky et al. (2015) proposed a method to generate a soft processor by high-level synthesis of an instruction set simulator written in C language. Sakamoto et al. (2018) proposed a method to generate a soft processor, Spim-like_sp, based on Skalicky et al.'s method by creating a high-level synthesizable MIPS instruction set simulator, Spim-like, and implemented dedicated instructions for three CHStone benchmarks. In addition, Sakamoto et al. proposed to use three types of reference passing when reference passing is used in the functions to be dedicated instructions. Iwamoto et al. (2019) proposed to enhance the number of applications from 3 to 11, and to select functions implemented as dedicated instructions based on two selection criteria. Masanobu et al. (2020) clarified the defects in Iwamoto et al.'s method of implementing dedicated instructions, and conducted performance evaluation to show the effectiveness of using directives.

The purpose of this study is to re-evaluate the dedicated instruction implementation method with many implementation patterns by fixing the defects pointed out by Masanobu. In this study, Vivado HLS 2020.1 is used as the HLS tool, and Vivado 2020.1 is used as the logic synthesis tool.

First, in order to identify the implementation patterns that cause failures, we conducted a follow-up experiment to the C simulation conducted by Masanobu et al. As a result, out of 39 implementation patterns with different selection criteria for functions to be converted into dedicated instructions and different methods of reference passing, 23 patterns were found to be faulty. As a result of the correction effort, the self-check values of 14 out of 23 implementation patterns became normal. We evaluated the performance of Spim-like_sp, which was generated by 28 out of 30 implementation patterns whose self-check values were found to be normal and which were successfully synthesized.

In the performance evaluation, we evaluated the trade-off using the AT product. The evaluation results are shown as relative values between the evaluation results without dedicated instructions and those with dedicated instructions. In the implementation pattern where the function selection criterion is execution time priority, the average of the relative values of latency is 0.872, the average of the relative values of the number of SLICE is 1.2, and the average of the relative values of the AT product is 1.01, indicating that there is little impact on the performance of the soft processor. In the implementation pattern using "spm_h" as the reference pass, the average of the relative values of latency was 0.757, the average of the relative values of the number of SLICE was 1.55, and the average of the relative values of the AT product was 1.27, indicating that the impact on the performance of the soft processor was small.