DATE: 2019/1/8

Department of Electrical and Electronic Information Engineering		ID	M153206
Name	Ryodai Iwamoto		

Supervisor	Shuichi Ichikawa Naoki Fujieda

Abstract

Title	Evaluation of special instruction implementation methods by high level synthesis

Protection of intellectual properties such as procedures of software is an important issue. Hardware implementation of functions is a method to protect them. Meanwhile, soft-processors, written in HDL (Hardware Description Language) are widely used in embedded systems. Sakamoto (2018) proposed to implement some functions of software as special instructions of a soft processor generated from an instruction set simulator by high level synthesis. However, the number of applications were too small (3 applications) to show its practicability, and the criteria were not clearly declared to select the target functions to be special instructions.

In this research, Sakamoto's method is applied to eleven programs of CHStone benchmark to examine applicability of the method, and to present the difficulties and their solutions when applying the method. Also, as a criterion for selecting a function, its processing time and code size are considered. The generated soft processor is simulated and synthesized to confirm that a suitable circuit for each criterion can be generated.

One of the problems appeared in argument passing of 64-bit variables. In this case, the value of the variable is stored in two MIPS general purpose registers. Therefore, it was necessary to change the description accordingly to pass 64-bit arguments to the special instruction that corresponds to the target function. When casting from pointer of memory which is an array of 32-bit variable to 64-bit pointer type, high level synthesis failed. In this case, it must be accessed by two 32-bit pointers.

By converting functions of large processing time into special instructions, the overall execution time was reduced by 33.0% on average. It was observed that the execution time increases by implementing a special instruction, where a large array of arguments is transferred by hardware with SPM (scratch pad memory). When the special instruction is implemented for the functions with large code size, the code size was reduced by 9.5% on average. When converting functions that directly manipulate global variables into special instructions, the description must be changed so that the pointer to the global variable can be passed by reference to the function, which results in smaller reduction of code size. Since data handled only by functions that is converted to special instructions is transferred to the simulator as well as functions, the reduction of code size became larger. Finally, the impact on implementation by each selection criterion was evaluated. When a function with a large processing time is converted into a special instruction, the average execution time is reduced by 18.0% as compared with the case where a function with a large code size is converted to a special instruction, while the code size is increased by 6.0% on average. When comparing the AT product as an evaluation on the cost performance aspect, the average when converting a function with a large processing time into a special instruction was 36.6% smaller on average.