DATE: 2018/1/9

Department of Electrical and Electronic Information Engineering	ID	M143231	Supervisor	Shuichi Ichikawa Naoki Fujieda
Name	Joji Sakamoto			

Abstract

Title	Implementation and Evaluation of High Level Synthesis of Processors with Special Instructions

Protection of intellectual property such as software is an important issue. Hardware Implementation of functions is a method to protect it. Meanwhile, soft-processors, written in HDL (Hardware Description Language), are widely used in embedded systems.

Skalicky et al. proposed a method to generate soft-processors from processor simulators using HLS (High Level Synthesis). They reduced the resource usage of soft-processor by excluding unused instructions in the target application. However, they did not use practical benchmarks for evaluation, nor compared the proposed method with lightweight soft-processors of the same instruction set.

This study firstly re-examines Skalicky's method. Then, I propose a method of implementing special instructions into soft-processors which are generated using HLS.

I compared the resource usage of Skalicky's soft-processor with that of Plasma, where Plasma is an open source MIPS compatible soft-processor. In the case of a single benchmark, blowfish presented the smallest resource usage. The number of slices was 40.4% smaller than Plasma. Meanwhile, aes presented the largest resource usage. The number of slices was 21.2% larger than Plasma. Only aes presented more resource usage than Plasma. In the case of multiple benchmarks, I evaluated with the combination in the smallest number of instructions. In case of 4 benchmarks, 452 slices were used, which is smaller than Plasma. Skalicky's method is regarded effective with four or less applications.

In this study, special instructions are implemented from functions of C language. I moved some of the functions of applications to the source code of soft-processor. In the case of call-by-value, the arguments are passed according to MIPS specification. In the case of call-by-reference, the following three methods are examined: (1) passing main memory's address, (2) using SPM (Scratch Pad Memory), and (3) using temporary variables. I implemented and evaluated special instructions in adpcm, aes, and motion. In adpcm, I implemented 4 functions as special instructions, where all functions have no arguments passed by reference. By implementing special instructions, the number of slices increased by 20.4% and the number of execution cycles decreased by 31.7%. The product of the number of slices and the number of execution cycles (AT-product) is 17.8% better (smaller) when implementing special instructions. In aes, I implmented 3 special instructions, where 2 functions have the arguments passed by reference. As a result, method (3) was better than other methods in AT-product. However, it was 67.3% worse than the processor without special instructions. In motion, I implemented one special instruction, which has the arguments passed by reference. Method (1) and method (2) couldn't be implemented because type conversion failed. By implementing special instruction, it was 78.0% better than the processor without special instructions in AT-product.