PSO based optimization of worst-case execution time for ASIP application

Mood Venkanna¹ *, Rameshwar Rao², P. Chandra Sekhar³

¹ Dept of ECE UCE, Osmania University Hyderabad, India
² Former VC, JNTUH Dept of ECE UCE, Osmania University Hyderabad, India
³ Professor, Dept of ECE, UCE, Osmania University Hyderabad, India

*Corresponding author E-mail: venkatmood03@gmail.com

Abstract

Industrial requires hard real-time systems for safety and critical applications like automotive, Aeronautics, manufacturing control and train industries. Hard Real-Time Systems’ embedded controllers are with expectation of complete the tasks within a certain time bounds reliably including task scheduling. The estimation of upper bound limits corresponding to the execution times is often termed as the Worst-Case Execution Times (WCETs). It is an essential step in developing and validating the hard real-time systems. Particularly, the upper bounds need to satisfy these constraints related to the execution times. However, it is often not feasible many times to set upper bounds on execution times for programs. In present work, the problem of choosing reconfigurable Custom Instructions (CIs) is accomplished by optimizing the WCET corresponding to an application. This issue is designed using Particle Swarm Optimization (PSO) based program for a path analysis. The work emphasizes on the effectiveness of optimizing the WCET when applied to a reconfigurable processor. It evaluates a compound application of multimedia with a host of reconfigurable CIs corresponding to a number of hardware parameters.

Keywords: Hard Real-Time System; Embedded Controller; ASIP; Optimization; PSO; Space Exploration.

1. Introduction

In today’s complex applications including multimedia application involving multiple hot spots require different form of hardware accelerators in order to accomplish the intended performance. In this regard the Application Specific Instruction, set Processors (ASIPs) facilitate a developer to design specific pre-fabrication customizations so as to improve the desired degree of specialization to meet the actual application environments such as the computational hot spots. Nevertheless, the ASIP can be kept within its reasonable size by only a few subsets of hot spots.

It is essential to know the WCET of a program for the design and verify the real-time systems. An efficient WCET method of analysis need to consider the feasible program flow that involves the loop iterations, the function calls and the timing effects of the caches and pipelines. One of the essential parameter of WCET analysis is the calculation as the combination of the flow as well as the timing information of hardware to computes a corresponding program in WCET analysis. The precision of WCET estimate largely depends on the type of flow information. In conventional methods, a trade-off is made between the global calculations precision and the computational complexity. However, such traditional methods are although fast but unable to account for all such flow information.

The flow information are essentially acts as a set of flow facts, each of which gives the constraints with respect to the program flow that include the infeasible paths, loop bounds, execution dependencies, etc. The local nature of the flow facts often express information confined to smaller regions such as an if-statement or a single loop of the intended program. In some cases, sometimes these smaller regions may occupy the structures of the basic program.

The task of estimating a suitable upper bound for WCET remains complex in nature due to the performance enhancing features like Pipelining, Caches and Branch predictions made available in the modern day processor. The presence of micro architectural state creates latency in instruction execution based on the dependency level in the past. When micro-architectural component, are in a worst-case scenario like a cache miss wherein the micro-architectural state is unable to determinethe result statically in a safe WCET bound for the complete task.

We would like to present an efficient approach to select the reconfigurable CIs statically in view of optimize the WCET bound of a task. Further, the time analysis of the high-performance architectures has been made to advance the research. A major issue is to resolve the worst-case path volatility that selects the WCET and optimize the CIs. To accomplish this task the worst-case path latency is by inserting a CI and to generate an entirely new different path.
As shown in Figure 1, an integral component of WCET-optimizing CI selection is the WCET bound estimation. In this, the CI or the instruction set selection is another major step used in the extension problem of the instruction set [1]. The paper is sectioned as explained. In section [1-2] the concerned literature is presented. Section 1.3 explains the optimization technique using PSO. The experiment result discussion is presented on section 1.4. The conclusion part is described on section 1.5.

2. Related literature

The flow, low-level analysis and the calculation are the essential components for the generation of the WCET estimate. The flow analysis (FA) phase aims to represent a program’s dynamic behavior. The process involves the function to be called, number of loop iterations how many times loops iterate in case air-statements dependencies exist. All possible program executions need to be safe approximated as the FA is unaware of the execution path which with respect to the longest execution time. These information are extracted using either the integrated in the programming language (manual annotations) [2] or provided separately [3-5], or by automatic flow analysis techniques [6-10]. Conventionally, the FA is referred as high-level analysis as it is mostly accomplished using the source code although it may be possible to use the machine code or intermediate level.

Analysis of instruction caches, data caches, and branch predictors are configurable for implementation in the reconfigurable fabric sum approach is executed. In this piece of work, a dynamic reconfiguration based custom instructions has been targeted in which the area demands varies.

3. Optimization using particle swarm optimization (PSO)

In order to allow solutions in the reconfigurable fabric this work introduces an area constraint as the reconfigurable fabricsum area, which need to be implemented the corresponding CIs where, j denotes the selected implementation in which case the term $y_{kj}$ = 1. It needs to be either low or equal to the total fabric area $a_{20}$. Consider a program having N basic blocks. The objective function can be represented as

$$\text{max} \sum_{i=1}^{N} c \times i$$

Selecting an instruction set to optimize the WCET bound essentially means, we aim to minimize the WCET over all possible selections, that is, we aim to minimize the maximum execution time.

Steps for Optimization

The population based adaptive stochastic optimization method is PSO. The technique is based on collective swarm intelligence (SI). The steps of PSO algorithm are as follows:

Step 1: Start and define solution space

Step 2: Represent potential solution to particle

Step 3: Initialize the population with respect to the random position $x$ as well as the velocity vector

Step 4: Evaluate the fitness for each agent.

Step 5: If fitness ($x$) > fitness (gbest) then gbest=x

Step 6: If fitness ($x$) > fitness (pbest) then pbest=x

Step 7: Update position and velocity

Step 8: gbest is best solution of parameters

In this work, the use of PSO is based on [24], initialize the particle with position vector ($x$) and velocity vector ($v$) respectively. The updated position vector and velocity vector for particle ($i$) are given by:

$$x_{i}^{k+1} = x_{i}^{k} + v_{i}^{k+1} \Delta t$$

$$v_{i}^{k+1} = w v_{i}^{k} + c_{1} rand (p_{i}^{k} - x_{i}^{k}) + c_{2} rand (g_{i}^{k} - x_{i}^{k})$$

Where, $k$ represents present iteration, $i$ is particle index with $x_{i}^{k}$ and $v_{i}^{k}$ are present positions and velocities respectively. The updated velocity corresponds to the previous velocity weighted by inertia $w_{i}^{k}$ corresponding with the parameters $c_{1}$ and $c_{2}$, which are cognitive and social rates respectively. The cognitive and social
4. Result and discussion

The number of possible hardware configurations m per CI changes drastically. As an example, in this work, we encountered a minimum CI value of [1] with a corresponding maximum value of 80= M implementations. It includes the entire software implementation with respect to a single kernel. When the CI value has been 80 different implementations, different degrees of parallelism have been observed with latencies. The area and the reconfiguration delay have been same during synthesis of the reconfigurable fabric. In case of minimum-latency implementation of the fabric the proposed algorithm can provide a number of relevant implementations.

Fig.3: Optimal Results for H.264 Encoder and Different Values off CPU/Fabric, As Well as Reconfiguration Bandwidth.

The work evaluates the results of the heuristic with respect to an optimal search. To practically demonstrate our approach the runtimes have been studied.

5. Conclusion

This paper attempts a new method to compute the WCET of a program. The technique is a hybridization of the fast computation techniques like the tree and path-based algorithms, are less precise, and potentially slow global IPET. It aims to find the smallest possible program components that can be handled for precision. The method of calculation for every individual component is not fixed and is dependent on the flow information characteristics and the program structure. Due to smaller parts as compared to the overall program, the technique provides the desired precision and is fast while arbitrary boundaries are introduced similar to the tree and path-based approaches.

References


