A WCET Estimation Method for a Real-time Schedulability Analysis

Jaehong Park, [a] Yongsuk Suh, [a] Insoo Koo,[a] Sehyung Jung,[b]
a Div. of I&C and HFE, KAERI, 150 Dukjin-dong, Yuseong-gu, Daejon, Korea, 305-353, middle75@kaeri.re.kr
b Control Tech. Research Inst., SEC Ltd.,974-1 Goyeon-ri Woongchon-myon, Ulju-gun, Ulsan, Korea, 689-871

1. Introduction

In the hard real-time digital instrumentation and control (I&C) systems of nuclear power plants, a real-time schedulability analysis is required to check if a task can completely execute a required mission within its deadline. The worst-case execution time (WCET) is one of the timing parameters in the schedulability analysis. The WCET is defined as the maximal possible execution time of a program before using the program in a system. Although an exact prediction of WCET is impossible because of the non-deterministic characteristic of hardware, it must be safe (i.e., no underestimation of the execution time) and tight (i.e., as little overestimation as possible). Underestimation of the execution time causes a fatal error of a system. The WCET estimation should be applied to only one task and assume that preemption or interrupt does not occur during its execution.

This paper presents the safe and tight WCET estimation method applied to the schedulability analysis for SMART-P MMIS.

2. Methods and Results

The WCET estimation method consists of a high-level analysis, low-level analysis, and a path search. This method is procedurally performed to calculate a WCET.

2.1 High-level analysis

The high-level analysis is to draw a control flow graph (CFG) of a program and decompose a program with a consideration of an infeasible path, loop bound, and a function call. Analyzed information for the structure of a program is described with a specification language which consists of three parts such as the name of the scope where the fact is defined, context specifier, and the constraint expression [1] as shown in Figure 1. Each node of the CFG is a sequence of statements without a jump statement. Figure 2 shows that the program in Figure 2(a) is divided into the nodes of a CFG in Figure 2(b). The CFG can be attached with a flow fact as shown in Figure 2(b). The flow fact for the program will impact on the search for the longest execution path in the program.

2.2 Low-level analysis

Fact	→	*Scope* : *ContextSpec* : *Constraint*
ContextSpec	→	<>
	|	[]
	|	< *RangeList* >
	|	[*RangeList*]
RangeList	→	*Range*
	|	*Range* , *RangeList*
Range	→	*Integer* .. *Integer*
	|	*Integer*
Constraint	→	*Expression Relop Expression*
Expression	→	*CountVariable*
	|	*Integer*
	|	(*Expression Op Expression*)
Relop	→	≤ | = | ≥
Op	→	+ | − | * | /

Figure 1. Flow Facts Specification

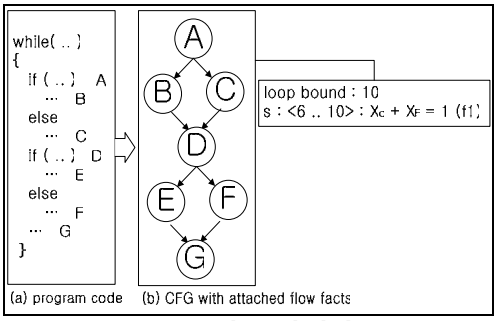

Figure 2. CFG with attached Flow Facts

To execute a low-level analysis, it is assumed that a processor is manufactured with a reduced instruction set computer (RISC) architecture that executes one instruction per clock cycle. The low-level analysis is to analyze the timing behavior of a target processor like cache and pipeline. Cache analysis is to predict the cache hit or cache miss during a run-time. Pipeline analysis is to analyze how many cycles are needed by each pipeline stage for instruction execution using the cache analysis results [2, 4].

The low-level analysis is performed using a simulator provided by a chip manufacturer. The simulator must have a cycle-accurate model of the CPU [3]. If we assumed to compute the execution cycle of each node (C and D) and two successive nodes in a CFG using the simulator as shown in Figure 3(a), the timing effect for the pipeline overlap between the two successive nodes can be calculated as shown in Figure 3(b):

$$\delta_{CD} = T_{CD} - T_C - T_D = 21 - 17 - 5 = -1$$

2.3 Path Search

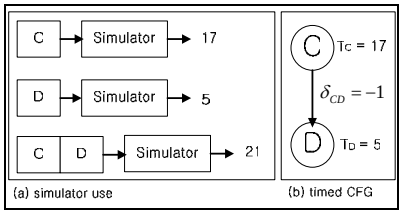

Figure 3. Timing Effects using a Simulator

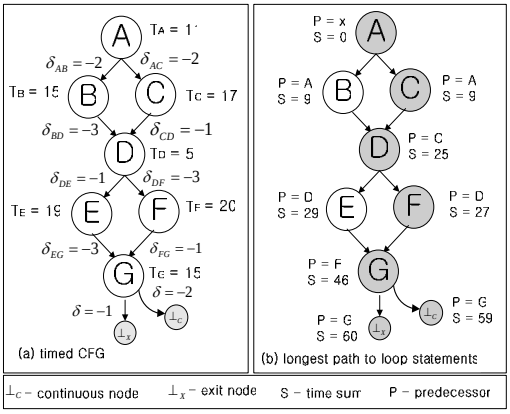

Figure 4. Longest Path Search

When we have assumed the cycle time of all the nodes and the timing effect between successive nodes as shown in Figure 4(a), the longest path in a loop can be searched by computing the time sum and the predecessor for each node of the CFG. The longest path of a loop statement is A-C-D-F-G as shown in Figure 4(b). When we have assumed a loop bound, the WCET of the loop statements becomes:

$time\_sum(\perp_C) * (loopbound - 1) + time\_sum(\perp_X)$.

If a path has a flow fact and/or a long timing effect, the path search should accommodate them [4, 5]. When we consider the flow fact of Figure 2(b) into Figure 4(b), the path should be divided into two virtual scopes as shown in Figures 5(a) and 5(b). In Figure 5(b), the original longest path with an infeasible path is removed and the new longest path (A-C-D-E-G') is searched by adding new nodes and edges. When we have assumed a long timing effect across the successive nodes (C-D-F) as shown in Figure 6(a), the new longest path is searched by adding the time of the timing effect to the last edge in the sequence as shown in Figure 6(b). By a summation of 5(b) and 6(b), the WCET for the loop-statement is (4 * 58) + 59 + (5 * 63) = 606 cycles.

3. Conclusion

The WCET estimation method for a program with a high-level and low-level analysis and a path search is presented in this paper. We assumed a RISC processor and that the chip's timing behavior is provided by the chip-maker.

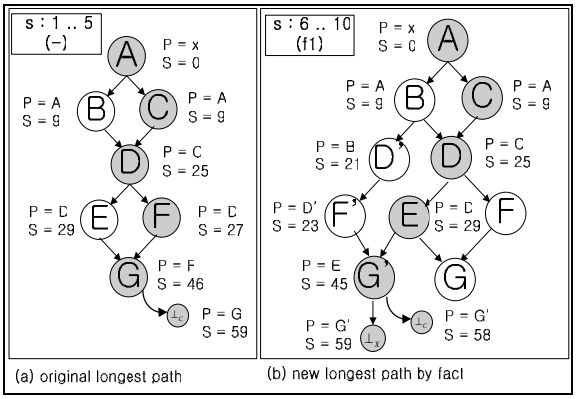

Figure 5. Longest Path Search by Fact

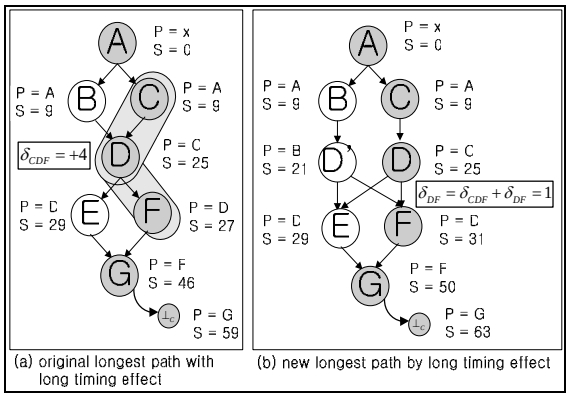

Figure 6. Longest Path Search by long timing effect

The WCET estimate will provide the most important timing parameter for the real-time schedulability analysis. For a future work, we plan to build a pipeline model for a specific processor and research the timing parameters like a blocking and interference for the real-time schedulability analysis.

REFERENCES

[1] Jakob Engblom, et al., "Modeling Complex Flows for Worst-Case Execution Time Analysis", 21th IEEE Real-Time Systems Symposium (RTSS'00), November 2000.

[2] C. Healy, et al., "Bounding pipeline and instruction cache performance", IEEE Transactions on Computers, 48(1), January 1999.

[3] J. Engblom and A. Ermedahl, "Pipeline timing analysis using a trace-driven simulator", 6th International Conference on Real-Time Computing Systems and Applications (RTCSA), IEEE Computer Society Press, December 1999.

[4] F. Stappert and P. Altenbernd, "Complete worst-case execution time analysis of straight-line hard real-time programs", Technical Report 27-97, C-LAB, Paderborn, 1997.

[5] Friedhelm Stappert, et al., "Efficient Longest Executable Path Search for Programs with Complex Flows and Pipeline Effects", 4th International Workshop on Compiler and Architecture Support for Embedded Systems (CASES 2001), ACM Press, November 2001.