TERM PROJECT REPORT

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Analysis of LLVMTA and Heptane for WCET Calculation of Dataflow Actors

Saltuk Çelik
Technische Universität Kaiserslautern, Department of Computer Science

To fully utilize parallelism of Dataflow Processing Networks, the worst-case execution time (WCET) of actors must be estimated. This paper provides analysis for two open-source WCET estimation tools from academia, LLVMTA and Heptane. A framework to run LLVMTA for dataflow actors is demonstrated. The framework successfully runs for simpler actors but some problems with more complex actors is observed.

1 Introduction

Dataflow process networks (DPN) are a model of computation where concurrent processes communicate through FIFO buffers [8]. The processes are defined as actor firings and actors implement functions that map input tokens to output tokens. CAL Actor Language (CAL) is a programming language that utilizes DPNs [4] and Dataflow Code Generator is a tool that transform DPNs specified in CAL to C++ code [1]. This tool is used for researching different optimization and mapping strategies. For efficient mapping of processes to the cores, worst-case execution time of actors should be estimated.

Worst Case Execution Time (WCET) is the time a task takes to execute on a specific hardware in a specific application context [9]. WCET of a program is an important criteria in real-time applications because it is used in performing schedulability analysis, ensuring meeting deadlines, and assessing the resource needs [5]. The airbag control software of a car is a good example of a system where knowing WCET is crucial, as deploying the airbags late in the event of a crash may have significant results.

Two main methods of estimating WCET are measurement analysis and static analysis. In measurement analysis, actual runtimes of the software on the particular machine are used. To make the estimation accurate, as much of the state space as possible must be explored. However, since most of the time it is not possible to cover everything, a safety margin is added to the maximum observed time [5]. RapiTime is a commercial WCET tool that makes use of measurement analysis for basic-block execution times on the real HW processor [10].

The other WCET estimation method is static analysis. For static analysis building of precise hardware model is needed. Given the code and hardware model for which the WCET is to be computed, the steps for analysis are as follows [6]

1. Obtain the Control Flow Graph (CFG) of the code and find possible paths
2. Determine the possible execution times of blocks, with consideration for the timing effects of microarchitectural features such as pipelining and caching

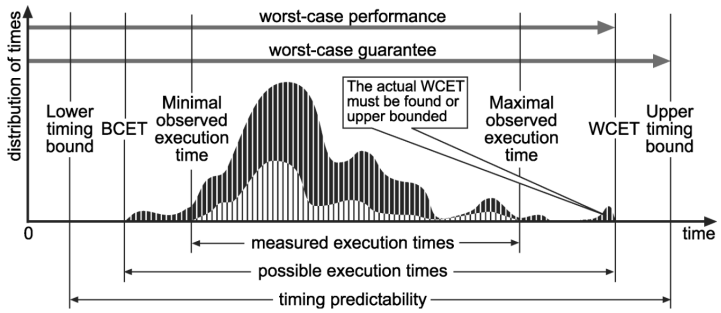

Figure 1: Typical WCET graph of a program [11]. Bold lines are theoretically possible execution times and light lines are observed execution times. WCET estimation must be larger than the longest possible time and as close to it as possible.

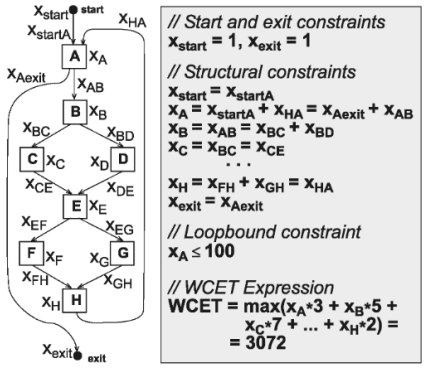

Figure 2: Implicit path enumeration technique (IPET) for WCET calculation [11] which is used in LLVMTA and Heptane.

3. Combine the information gathered from 1 and 2 to get the upper timing bounds. IPET (fig. 2) is a commonly used method at this step.

Both LLVMTA and Heptane are static WCET estimation tools. In the next sections we give brief information about each, make a small comparison test, and then explain how LLVMTA was used for WCET calculation of dataflow actors. In the last section we explain how LLVMTA was used for WCET calculation of dataflow actors.

2 LLVMTA

LLVMTA is an academic WCET analysis tool based on (a patched version of) the LLVM compiler infrastructure [6]. It is developed in Saarland University. The main aim of LLVMTA is the state-of-the-art analysis frameworks, and it does not have a focus on the complexity of modeling many different real-world hardware architectures (like commercial tools do, e.g. aiT). The main feature of LLVMTA which the authors pay attention to is that it uses Abstract Execution Graphs

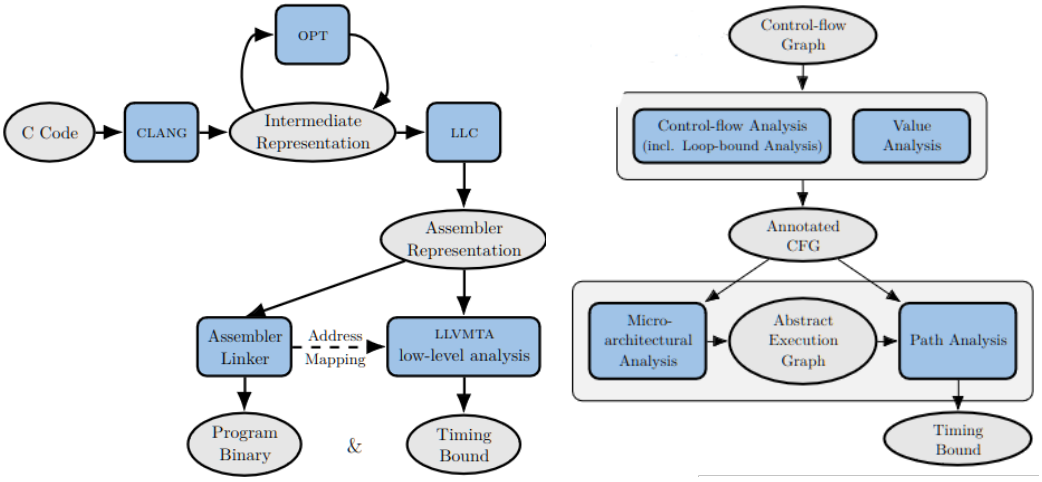

Figure 3: Overview of the LLVMTA workflow (left) and its place in the LLVM compilation flow (right) [6] The analysis runs on the final assembler code of LLVM, which is the closest representation to the machine code. This way LLVMTA does not bother with extracting the control flow of a program from the binary.

(AEG) (fig. 3). LLVMTA does cycle by cycle simulation on abstract microarchitectural states, which take into account features such as pipelining, forwarding, branch prediction, caches, etc. Instead of creating a single bound for each basic block, AEGs capture correlations between timing contributions of the basic blocks. The longest path in AEG is found by IPET with Integer Linear Program (ILP) solvers. LPsolve is the free option while IBM CPLEX and Gurobi Optimizer are the paid alternatives.

To use LLVMTA, a program written in C is provided. The architecture is specified from many possible options such as

- ARM and RISC-V ISA
- In-order, strictly in-order, and out-of-order execution pipelines
- Scratchpad memories, as well as caches with least-recently-used replacement policy and both write-through and write-back policy

LLVMTA automatically derives loop bounds using LLVM's scalar evolution analysis in many cases. When LLVMTA fails, the user should provide loop bounds manually in a CSV file. Likewise, the user has to provide annotations file for the external functions, which tells WCET bound and cache accesses of that function.

3 Heptane

Heptane is an open-source WCET estimation tool developed in University of Rennes [7]. Heptane does not do AEG but does contextual analysis, which means that the analysis of a function is

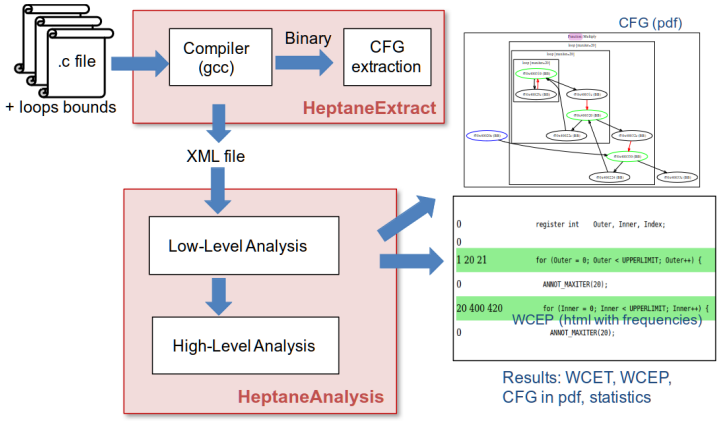

Figure 4: Workflow of Heptane [7]

performed for every call path. Similar to LLVMTA, Heptane has limited hardware support. It utilizes LPsolve and IBM CPLEX as ILP solvers.

Heptane is divided into two parts. HeptaneExtract generates CFG from source-code files written in C or assembly language, and stores the results into an XML file. HeptaneAnalysis applies analyses on this file to produce the WCET estimation.

To use Heptane, a program written in C and an XML configuration file describing the architecture is provided. The hardware specificiations can be

- MIPS, RISCV, and ARM ISA
- Only in-order execution pipeline
- Different cache hierarchies with various replacement policies and only write-through policy

Heptane does not include any analysis of maximum numbers of loop iterations, the source code has to be augmented by the user with annotations. There is no system for handling external functions either.

4 Running both tools on the same source file

To demonstrate the tools' functionality and differences, both of them were run for the WCET estimation of the same code, with small changes (fig. 5). For LLVMTA, the while loop was removed in compilation and WCET was estimated as 52 initially. After modifiying some compiler flags inside the main script, the loop was kept from being removed away.

Heptane required the loop bounds to be given inside the source file, meanwhile LLVMTA failed to extract the loop bounds automatically and a loop annotations CSV file was provided separately.

Even though both ISAs are chosen as ARM, microarchitectures of the two runs are not configured equally so it is pointless to comment on the difference in WCET estimations of the tools. But considering the capabilites of LLVMTA and the logarithmic difference in runtimes,

```
1
2  int main() {
3      int i = 0;
4      while ( i < 50){
5
6          ++i;
7      }
8      return 0;
9  }
```

```
1  #include <annot.h>
2  int main() {
3      int i = 0;
4      while ( i < 50){
5          ANNOT_MAXITER(50);
6          ++i;
7      }
8      return 0;
9  }
```

Figure 5: Source codes for LLVMTA (left) and Heptane (right)

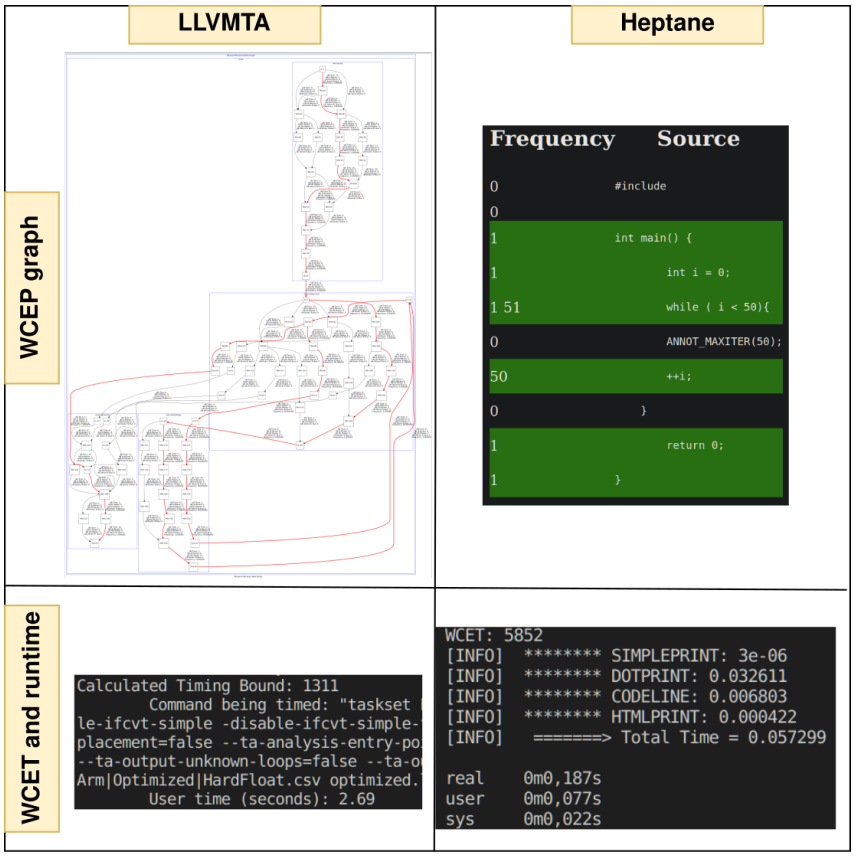

Figure 6: Comparison of worst case execution path and time outputs, and runtimes of both tools on given source files in fig. 5. LLVMTA found WCET as 1311 cycles and took 2.69 seconds to compute. Heptane found WCET as 5852 cycles and took 0.077 seconds to compute.

the best decision falls on what is prioritized, performance or accuracy. LLVMTA was chosen for WCET estimation of dataflow actors because it was easier to be modified to analyze C++ files.

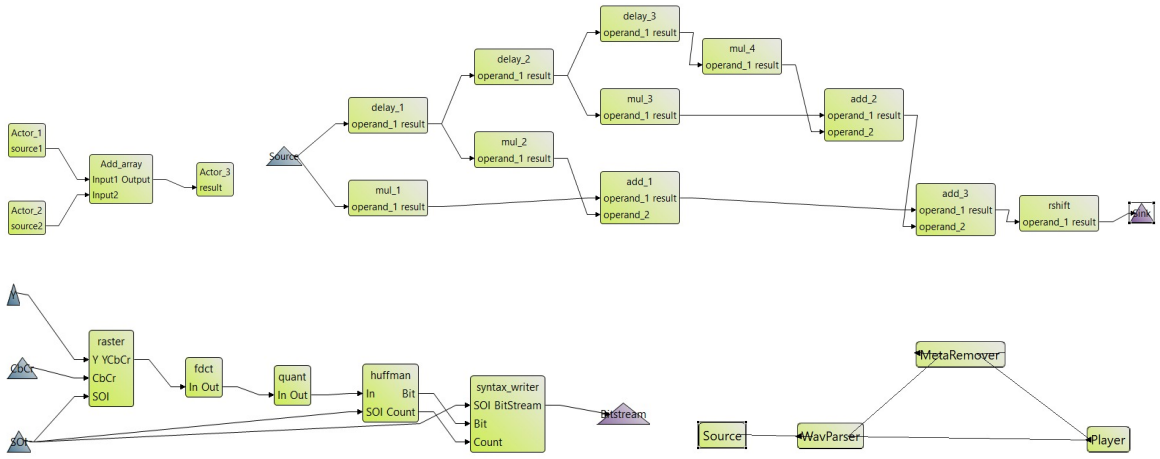

Figure 7: Example networks. AddArray, FIR, JPEGencoder, WavePlayer in reading order

5 WCET calculation of dataflow actors

We utilize LLVMTA to compute WCET estimation of actors genereated from Dataflow Code Generator. The modifications on LLVMTA to run for C++ files and other scripts can be found in [2]. The workflow to obtain each actors' WCET are

1. Run Dataflow Code Generator on CAL and XDF source files to create the C++ code. At this step each actor has its own header file and there are some additional files.
2. Put each actor file into a seperate folder with necessary header files.
3. Modify actor files so that LLVMTA can run on their main functions.
4. Run LLVMTA on each folder to obtain unknown loop bounds csv file.
5. Fill in unknown loop bounds as 1.
6. Run LLVMTA on each folder again to obtain WCET.

We apply these steps to 4 different networks specified in CAL and XDF files taken from Open RVC-CAL Applications repository [3]. The networks are in figure 7 and WCET results are in table 1.

While the WCET of simpler actors are computed with no issues, the more complex actors may cause problems. The errors observed due to LLVMTA have all occurred in LoopBoundInfoPass phase of LLVM. It should be noted that some of these problematic runs finished successfully after some trials. The errors observed due to Dataflow Code Generator were some functions that were used but not declared in the generated code.

AddArray			
Actor2	Actor3	Add	Actor1
2620	2339	8049	2620

FIR			
Mulc	Delay	Add	Rshiftc
5480	5491	8035	5480

JPEGencoder				
SyntaxWriter	Quantizer	FDCT	HuffmanEncoder	RasterToMB
4864	-	-	4046	-

WavePlayer			
Player	MetaRemover	WavParser	Source
x	11395	17119	x

Table 1: WCET calculation of the actors of example networks. "-" means could not be computed due to an error from LLVMTA, "x" an error from Dataflow Code Generator

6 Conclusion and Future Work

Both tools Heptane and LLVMTA shown to be good research tools to explore WCET of programs on generic hardware configurations, with LLVMTA being more up-to-date and detailed, and Heptane being much faster. Both tools can be used for WCET estimation of dataflow actors once the future plan of generating C code from Dataflow Code Generator is realized.

At the moment the given workflow can be used to estimate WCET of actors of simple networks. Possible ways to improve and resolve the problems could be choosing correct cross compiler libraries for the target machine, debugging the LoopBoundInfoPass phase of LLVM, and fixing the errors with Dataflow Code Generator code.

References

[1] *Dataflow Code Generator*. https://github.com/Florian233/Dataflow_Code_Generator.

[2] *LLVMTA*. https://github.com/SaltTuk/llvmta.git.

[3] *Open RVC-CAL Applications*. https://github.com/orcc/orc-apps.

[4] Johan Eker & Jorn Janneck (2003): *CAL language report*. Technical Report, Tech. Rep. ERL Technical Memo UCB/ERL.

[5] Björn Franke: *Embedded Systems Lecture 11: Worst-Case Execution Time*. https://www.inf.ed.ac.uk/teaching/courses/es/PDFs/lecture_11.pdf. [Online; accessed 09-April-2024].

[6] Sebastian Hahn, Michael Jacobs, Nils Hölscher, Kuan-Hsun Chen, Jian-Jia Chen & Jan Reineke (2022): *LLVMTA: an LLVM-based WCET analysis tool*. In: *20th International Workshop on Worst-Case Execution Time Analysis (WCET 2022)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[7] Damien Hardy, Benjamin Rouxel & Isabelle Puaut (2017): *The heptane static worst-case execution time estimation tool*. In: *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*, Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

[8] E.A. Lee & T.M. Parks (1995): *Dataflow process networks*. *Proceedings of the IEEE* 83(5), pp. 773–801, doi:10.1109/5.381846.

[9] P. Puschner, R. Kirner & B. Huber: *Worst-Case Execution-Time Analysis – WCET Analysis*. https://ti.tuwien.ac.at/cps/teaching/courses/real-time-systems/slides/rts06_wcet_analysis-1.pdf. [Online; accessed 09-April-2024].

[10] Hardik Shah, Andrew Coombes, Andreas Raabe, Kai Huang & Alois Knoll (2014): *Measurement based wcet analysis for multi-core architectures*. In: *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*, pp. 257–266.

[11] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra et al. (2008): *The worst-case execution-time problem—overview of methods and survey of tools*. *ACM Transactions on Embedded Computing Systems (TECS)* 7(3), pp. 1–53.