Seminar

Analyse von Programmausführungszeiten
Prof. Dr. Reinhard von Hanxleden

Thema
WCET im Context von
Hardware/Software Codesign
20.01.2003
Marcel Hamann

Overview

- The tool Polis
- A Compilation-based Software Estimation Scheme
- Efficient Software PerformanceEstimation Methods
- Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator
- Final remarks

The tool Polis

- Developed by the University of Berkeley, California
- Is a hardware/software codesign environment for control dominated embedded systems
- Based on a formal model of computation called codesign finite state machine (CFSM)
- Needs a formal language (e.g. Esterel) to model the CFSMs
- CFSMs are partitioned manually into HW and SW

Polis needs a formal language to work upon, as there is no language linked directly to it. Polis descibes a formal behavioral model based on a network of communicating entities (the CFSMs) You can map each CFSM to either HW or SW, choose a processor type and cache architecture.

The tool Polis

- Software parts are translated into s-graphs
- The s-graph is a directional acyclic graph which represents the control flow
- The generated code is a direct result of the s-graph

The CFSMs that are declared as SW are transformed into software graphs (s-graphs), this is a lower level of abstraction than the CFSM.The s-graph is a directional acyclic graph and can be manipulated to optimize the trade-off between performance and code size of the final implementation in the target system. It is possible to generate code on several different levels, e.g. From the s-graphs or on the higher level of abstraction, from the CFSMs.

A compilation-based software estimation scheme

- A Compilation-based Software Estimation Scheme by Lajolo/Lazarescu/Sangianno-Vincentelli published in CODES `99 Rome Italy [Lajolo]
- Main focus is software
- Idea: use the GCC to generate assembler level code and use instruction level timing analysis

This method is on a very low level of abstraction. The Polis tool is used, but all the optimization and evaluation is done directly on the produced code (in this case C-code) and little use is made of the possibilities of Polis. No considerations were made about hardware, as Hardware CFSMs take only one clock cycle to execute a tansition.

A compilation-based software estimation scheme

- Design the system with Polis and Esterel
- Repeatedly optimize the (software) C programm and compile it until you have an optimized RTL code
- Do an instruction level timing analysis and build a labeled CFG afterwards (the C simulation model)
- Cosimulate the HW and SW parts

The C-code needed is generated by Esterel. As C is (sometimes) rather close to assembler code, it is possible to estimate it`s (WC)ET rather accurate and so time tags, representing the execution time, can be added to pieces of code and a estimation of the execution time of the whole program can be obtained with the help of a labeled CFG. The CFG has been introduced in some of the prior sessions e.g. By Hendrik Janz.

A compilation-based software estimation scheme

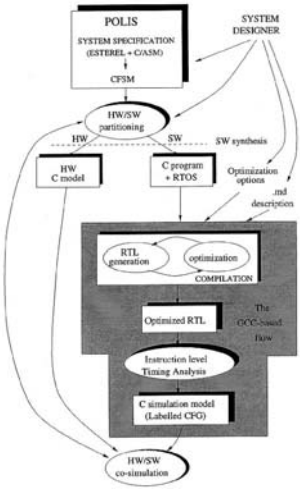

Figure 1: The simulation flow

A compilation-based software estimation scheme

- First results:
- Optimazation has a major effect on the performance of the system
- GCC-based estimation has an error of less than 4%

A WCET analysis on this level of abstraction is not new and this method is only a refinement of already existing (and also rather accurate) methods. The main problem is, that the whole idea is based on the GCC and not on every RT-system GCC generated code is used. So if your system will not use GCC generated code, this method can not be used accurately as other compiler might do different optimizations of the code.

Efficient Software Performance Estimation Methods For Hardware/Software Codesign

- Efficient Software Performance Estimation Methods For Hardware/Software Codesign by Suzuki and Sangiovanni-Vincentelli
- Main focus: Software and code-size
- Idea 1: analyze the s-graph
- Idea 2: analyze the CFSM

This paper is on a more abstract level than the first one, it tries to use the graphs produced by Polis to calculate a WCET on this level, without knowing the exact code.

Efficient Software Performance Estimation Methods For Hardware/Software Codesign

- Estimating the s-graph:
- Use precalculated cost parameters to add a wheight to each node and edge in the graph
- Use DFS to find both the maximum amd the minimum cost path
- Sum up the costs to get the estimated code size

As the s-graph is a directed acyclic graph, DFS can be used to calculate a longest (or shortest) path, if wheights are added to the nodes and/or edges.The wheights are a result of estimated execution times for the value/function of the node or edge. After an s-graph is constructed, it is optimized according to criteria that reflect running time or code size and DFS is used to estimate the WCET.

Efficient Software Performance Estimation Methods For Hardware/Software Codesign

model name		estimated	measured	% difference
FRC	min	158	141	12.06
	max	469	496	-5.44
	size	654	690	-5.22
TIMER	min	223	191	16.75
	max	938	912	2.85
	size	1573	1436	9.54
ODOMETER	min	145	131	10.69
	max	361	363	-0.55
	size	454	457	-0.66
SPEEDOMETER	min	314	335	-6.27
	max	880	969	-9.18
	size	764	838	-8.83
BELT	min	119	111	7.21
	max	322	323	-0.31
	size	511	520	-1.73
FUEL	min	197	171	15.20
	max	533	586	-9.04
	size	637	647	-1.55
CROSS_DISPLAY	min	262	221	18.55
	max	16289	16979	-4.06
	size	32592	38618	-15.60

Figure 2: Experimental results from the s-graph level method

This are the results of the s-graph method. The results of the MDD traversion method have about the same minimum number of execution cycles as the s-graph method. Only the maximum number of execution cycles was larger than here.

Efficient Software Performance Estimation Methods For Hardware/Software Codesign

- Estimating the CFSM
- More problematic, as CFSM does not reflect the code structure as clearly as the s-graph
- Use a MDD (is a DAG) and transform it into an s-graph

The CFSM is not a directed acyclic Graph, so the former method can not be used that easily. But it is possible to transform the CFSM into a multivalued decision diagram (MDD). A MDD is a DAG, but the nodes are either terminal (level 0) or non-terminal nodes, with non-terminal nodes each corrosponding to submodels of a lower level. A MDDconstructed from a CFSM is similar to an s-graph in some ways. Most important is, that every path in an s-graph is also a path in the MDD. So the WCET estimation can be done by DFS on the MDD, but nothing can be said about code size.

Efficient Software Performance Estimation Methods For Hardware/Software Codesign

Results:

- The estimations of the s-graph are slightly better than those of the CFSM
- The s-graph is within 20% of the measured time, CFSM is 5% less accurate
- The s-graph method is useful for optimazation in software synthesis, CFSM is good enough to be used for min/max execution times

These methods are less accurate than the method by [Lajolo], but can be done much earlier in the design process, the CFSM-estimation can be done even before the prtition isnto HW/SW is performed.

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

- Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator by Liu, Lajolo and Sangiovanni-Vincentelli published 1998 in IEEE proceedings
- Idea: use an Instruction Set Simulator (ISS) with a fast event-based simulator
- Advantage: the delay of events can be measured instead of estimated

This paper focusses more on the integration of Polis with an ISS than on the exact estimation of the WCET. They use the Ptolemy environment developed by the University of Berkeley, California.

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

- Idea of using ISS is not new
- Achieving high accuracy and good performance for RTL-simulation is an unsolved problem
- Here a wrapper is used for interaction between ISS and simulator (Ptolemy)
- This delivers a uniform interface
- Every star communicates only with wrapper

Ptolemy uses many keywords from the Astrology, so the smallest components are called stars, larger ones e.g. galaxies. The wrapper used has to be programmed for the particular ISS used.

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

- Stars send all information to wrapper who forwards them
- Calculating results happens in two steps: ISS delivers time stamp, Ptolemy delivers values
- All computations base on the s-graph
- Computations are stored for later use

When the simulation is started, every star gather all the data relevant for him and delivers it as one package to the wrapper, which translates it into ISS syntax and sends it to the ISS, who returns a time tag for the operations done. The new values needed for the star are calculated by Ptolemy, not the ISS. The computations done by the ISS are stored in a hash table with a key attached to it. If the same computation (same key) ha to be done later, the value is just taken from the table, not calculated again.

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

Table 1: COMPARE module

State	Dcta	Dctb	Output	Estimated	Measured
0	-	-	-	57	51
1	0	0	0	77	63
1	0	1	0	77	59
1	1	0	0	77	49
1	1	1	1	106	56
1	1	1	2	106	58
2	-	0	0	67	49
2	-	1	1	98	56
2	-	1	2	98	58
3	0	-	0	41	66
3	1	-	1	100	61
3	1	-	2	100	63

Table 2 : Adder module

State	Dcta	A	B	Output	Estimated	Measured
0	-	-	0,1	0	120	114
1	0	-	-	1	45	50
1	1	0	0	1	887	392
1	1	0,1	0,1	1	887	755
1	1	2,30E+012	5,30E+008	1	887	790
1	1	0,12	0,99	1	887	855

COMPARE is a 2-input-2-output module which compares two integers A and B. If A>B it emits output O1 else it emits output O2. ADDER is a floating point adder with one input A one internal parameter B and an output SUM. DctA is the flag of detecting an input A.

Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator

Some results:

- Integration ISS-Polis was successful
- The measured WCET-values are not good
- Only a prototype

The results of the estimation were often totally useless (difference of up to 100% to measured time), so it`s hard to say at this moment if the method is really usefull.

Final Remarks

- Software execution time is rather accurate
- Communication overhead is a rather important problem
- More complex problems mean less accuracy in the measurements
- With more abstract methods accuracy is also most certainly lost

References

- Ptolemy home page: http://ptolemy.eecs.berkeley.edu
- Andrew S. Miner "Efficient State Space Generation of GSPNs using decision diagrams" Department of Computer Science, Iowa State University
- Marcello Lajolo etc. "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation" published CODES `99 Rome Italy
- Kei Suzuki etc. "Efficient Software Performance Estimation Methods for Hardware/Software Codesign" published ACM 1996
- Jie Liu etc. "Software Timing Analysis Using HW/SW Cosimulation and Instruction Set Simulator" published IEEE 1998