J. Michael, J. Pfeiffer, A. Wortmann (Hrsg.): Software Engineering 2022 Workshops,
Digital Library, Gesellschaft für Informatik 2022 113

Static Analysis Methodologies for WCET Calculating with Asynchronous IO

Georg Seifert[1]

Abstract: The estimation of the upper bound of the WCET is one of the hardest challenges in the analysis of safety critical real time applications. Since a long time, the static WCET estimation of single core CPU-focused systems without shared resources has been investigated and can now be regarded as solved. The WCET analysis with shared resources is not feasible with current practice due to the lack of information about the internal timing, especially the IO system. The rise of system functionality and the growth of interfaces with high bandwidth in MCUs has resulted in a situation where a CPU-only processing of the IO, or a degraded usage of DMACs, is no more feasible. Therefore, dedicated hardware components, like DMAC, have to be considered and the disadvantages of conflict-afflicted transfers must become part of the analysis. To resolve the problems with interference afflicted MCU internal data transfers, an approach is presented which describes the influence parameters on the WCET and expresses these in a simplified timing model of the MCU. Afterwards the information is used to extrapolate the increase of the execution time caused by a given type of traffic to estimate the WCET with asynchronous IO accesses.

Keywords: worst case execution time analysis; static estimation; IO subsystem; hard real time system; safety critical systems

1 Introduction and Motivation

In various industries, computing systems are required to exhibit real-time behavior. If, in addition to this requirement, safety-critical aspects are also demanded, as in the case of flight control computers in aviation, the violation of the execution time can cause fatal damage for humans or the environment. For example, the aircraft-specific standard for software development DO-178C [**do178c**] remarks that, in particular, the selection of hardware has an unpredictable influence on Worst Case Execution Time (WCET).

To satisfy these requirements for predictability, single-core systems are currently used, where internal accesses to different areas are handled in a conflict-free manner, to fulfill the prerequisites of the current static WCET calculation methods. The results of the static run-time analysis are based on representative models of the CPU and its associated caching-behavior. These models additionally do not cover other components of the Microcontroller Unit (MCU), such as DMA-Controller (DMAC), multiple processors or internal communication networks and the potential interference. These limitations of static WCET analysis lead to a restriction in the use of current hardware.

[1] Technische Hochschule Ingolstadt, Esplanade 10, 85049 Ingolstadt, Deutschland, Georg.Seifert@thi.de

Nevertheless, due to the increase of the required functionalities of the systems and the growth of the bandwidth of the embedded interfaces, a situation arises, where the CPU-driven processing approach of the Input/Output (IO) is no longer possible. Despite the requirements for more performance, regulatory authorities such as the European EASA [**easa**] are currently prefer to prevent the use of DMAC, but also multi-core systems, in safety-critical applications of the highest safety assurance levels (DAL-A/B), since there is no reliable proof of the functional safety of the hardware, expect of the ideas of CAST-32A, according to the current state of the art.

To solve this restriction, the currently most promising approach is the use of additional concepts, such as Direct Memory Access (DMA), which offload the IO processing from the CPU. In the past, the use and the associated certification of DMACs have been successfully performed in limited manner like in synchronous mode or with restricted time limits. In order to operate a DMAC independently and asynchronously to the CPU in the future, the current analysis methods for evaluating WCET must be extended to include these features. One potential methodology is presented here, which examines the hardware and software in detail. The derived information provides a novel approach to compute the WCET with asynchronous IO, which enables the usage of DMAC in safety-critical real-time systems.

To cope with this challenge, this paper describes a new approach dealing with asynchronous IO and shared resources: Sect. 2 shortly summarizes the current state of WCET estimation with the focus of shared resources. In Sect. 3 the problem of WCET estimation and shared resources in safety critical hard real time applications is described. Sect. 4 presents a methodology how to identify and quantify the parameters affecting the WCET with shared resources and to perform the recalculation of the WCET. This method is applied in Sect. 5 to a set of WCET benchmark tools and a comparison between different methods of estimating asynchronous DMA. Finally, Sect. 6 summarizes the technique and the results, and gives a short outlook on further work.

2 State of the Art

The general challenge about WCET analysis and the individual problems have intensively been investigated. Regardless of the main research of WCET analysis' focus on mitigation and prevention of shared resources, a quantification of timing issues during resource conflicts on common systems, like commercial MCUs, is not largely considered.

Wilhelm et al. give a general overview of WCET analysis and problems in [**wcet**]. As primary root causes for the complexity of WCET analysis they mention the following aspects: data-dependent control flows, context dependency of execution times and timing anomalies. This work includes also possible approaches about static and measurement based analysis, as well as their pros and cons. Along the theoretical challenges, the practical usage within the industry is discussed by describing state of the art tools and how to integrate the analysis into the development process.

Using an asynchronous DMAC, Pellizzoni et al. have developed in [**pessimistisch**; **Betti2013**] possibilities to control and throttle the external data flow for predictable CPU accesses. They extend the PCI and the PCIe system with special access control devices, which manage the data access of the IO subsystem. The first approach in [**pessimistisch**] was to block all external bus accesses during time critical accesses of the CPU to shared memory region. They have mitigated the problem of buffer overflow by further development in [**Betti2013**] through a real time bridge and a peripheral scheduler. The concept of extending the bus is only applicable for physically accessible interconnection systems – an adoption on internal MCU interconnects is not possible due to limited hardware access.

Besides the methods dealing with shared resources caused by DMA accesses, other considerations are based on multi-core systems and the Integrated Modular Avionics (IMA) approach and ARINC 653. IMA defines a real time computing network system for airborne systems, where different modules and applications can run with different criticality levels. Kim et al. evaluate in [**Kim2014**] a concept to use a dedicated core as an individual partition for managing the IO in a conflict free manner. Inter-core communication and interference between the different cores are not part of this evaluation and have to be managed using an IMA compatible RTOS, which guarantees the partitioning constraints. To fulfill one of the constraints, that only one CPU is accessing the shared resources at the same time, different partition techniques have been developed. For more general application on multi core systems, [**measuringMulticore**] provides a measurement based approach to evaluate the impact of interferences of competing cores. The outcome of this investigation is, that the application to be tested takes longer and execution time becomes less predictable. A detailed information, where the interference occurs is not part of this solution and has to be investigated in detail especially for hosting mixed-criticality software.

3 Problem Description

In the aviation industry, especially in the avionic part for flight control systems, safety-critical and hard real-time capable systems have the requirement to the highest certification levels (DAL-A/B). Therefore, the software is implemented as periodic control loop integrated on a simple MCUs. The main challenge here is not the development and implementation of the application itself, but the subsequent verification of the entire software/hardware system. This also includes the verification of the temporal behavior and the calculation of the WCET. Since the verification of the entire system is currently limited by the choice of hardware[2], simple MCUs such as the Texas Instruments Hercules TMS570LC4357 (Hercules TMS570), are used for this implementation. This kind of safety-MCUs are equipped with ARM Cortex-R5 real-time processors in lock-step mode and other safety functions.

The limitations given by the certification authorities taken care on software side by primitive implementations. This is mainly performed by CPU-only data processing (programmed

[2] The EASA defines in its Certification Memorandum [**easa**] MCUs with more than a single CPU as “Highly Complex COTS Microcontroller”.

IO) to prevent simultaneous access to shared resources. In order to enable more efficient processing of the IO, sequential DMA mechanisms are used next to the CPU. However, to guarantee interference-free operations, the CPU waits during processing of the DMAC for the completion of the transfer. Parallel processing on the same resources of the data by the DMAC and the CPU is currently not possible.

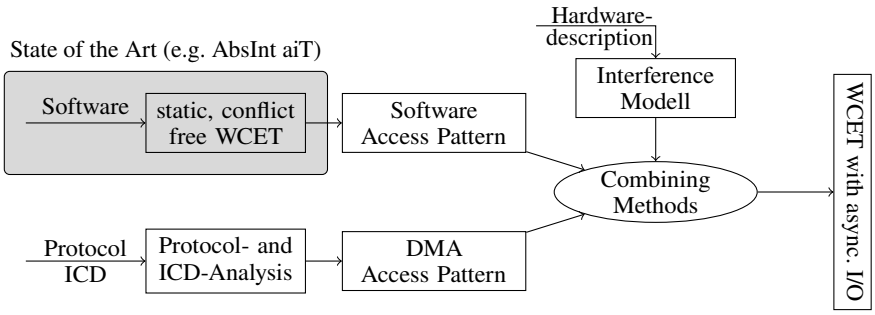


Fig. 1: Schematic Approach for Calculating the WCET with Asynchronous IO

The limitation of the used hardware and the software concepts without resource conflicts is not only due to the certification process of the regulatory authorities, but also due to the current WCET analysis tools (Fig. 1, gray box). These tools currently only model the processor itself and processor-related hardware, such as the cache. External components such as the internal interconnection network, based on the various crossbar switches and bus systems, and its resulting interference from competing transmissions are not considered. This is mainly caused by missing information about the internal structure of the systems and the incomplete documented behavior of the timing. In many cases, the manufacturers keep this detailed information confidential as their expertise.

To improve this situation, the possible timing effects for the different internal device groups (processors, DMAC and interconnection networks), need to be characterized by building an internal interference model, see Fig. 1. In addition, the DMAC access patterns must be considered in an abstract way to analyze the WCET with conflicting resources.

4 Methodology

For the calculation of WCET with asynchronous IO (1) the MCU internal interferences, (2) the access pattern of the CPU and (3) the access pattern of the DMA have to be evaluated.

Based on the information of the different analysis and the resulting models of the potential interference, the maximum number of possible collisions and the associated delays can be calculated in a final step.

4.1 MCU Internal Interferences

The evaluation of the hardware and the extraction of internal timing features, as well as the resulting interferences, have to be figured out by a detailed reference manual analysis and a follow-up reverse engineering of the hardware for completion and verification of the published information. The main information can be obtained within the sections of the interconnection systems, describing the crossbar switches and the internal bus systems.

Simple modern MCUs, like the Hercules TMS570, implement a set of interconnection systems, which are cascaded by connecting different crossbar switches, bus systems and related master (CPU, DMA, etc.) and slave interfaces (memory, timer, devices, etc.). In addition, the address ranges of the individual slaves and bus areas can be taken from the reference manuals. These allow the definition of the individual shared regions (i.e. shared resources) that cause interferences by simultaneous access.

With the help of the knowledge about the structure and the used communication interface, the access timing to the different segments of the MCU can be calculated. For this purpose, the individual configuration of the hardware, like the clock tree, transfer and burst sizes and potential further information such as chaining of accesses are consulted.

After the detailed analysis of the manuals, a completion and validation of the information have been performed. Therefore, for simple hardware systems a reverse engineering approach is chosen. A good trade-off between cost and effort are the usage of micro benchmarks, which access individual paths inside the MCU nearly without side effects, see [**arcs18**].

Micro benchmarks are performed to investigate the timing and potential interferences on bus segments, which base mainly on sequentially repeated load and store introductions. As only a couple of lines of machine code are used and after an initial instruction cache load, the information is present in the CPU cache, no additional memory transfers are performed by the application. Thereby, only interferences on the shared peripheral interface can occur and the potential variances there can be recorded.

In [**arcs18**] different methods for evaluating individual parts of the interconnection system are described. For a wide range of external devices simple micro benchmarks, which are toggling a given pin, can be performed. These are combined with a competing DMA transfer copying data between different locations of the MCU. These external pins are provided by devices, like GPIO but also by many other interfaces, which provide a software controlled state change for single pins. These pins can be captured by an external measurement setup based on an oscilloscope or logic analyzer, which can monitor the timing of the pin status changes and possible deviations.

This information has to be provided for all used paths and word widths of the interconnection system in dependence of the configured clock tree and the corresponding DMAC configuration. Based on this information, in a follow-up step a detailed look-up model of the paths with possible interference to its address regions is build. This information [**arcs19**]

quantifies the disturbance on individual paths, which is used for the following WCET estimation.

4.2 Access Pattern of the CPU

In addition to the static knowledge of the internal communication paths, the temporal occurrence of accesses of the CPU to the different slaves of the MCU must be quantified in an abstract way. Here, results of the static, collision-free WCET-calculation are used, as provided for example by *aiT* from *AbsInt*, cf. Fig. 2(a).

Various intermediate steps of the abstract interpretation of the analysis are used. Important individual information here are: control flow of the basic blocks, worst case behavior of the individual basic blocks, value analysis of the memory accesses to the different segments, cache analysis

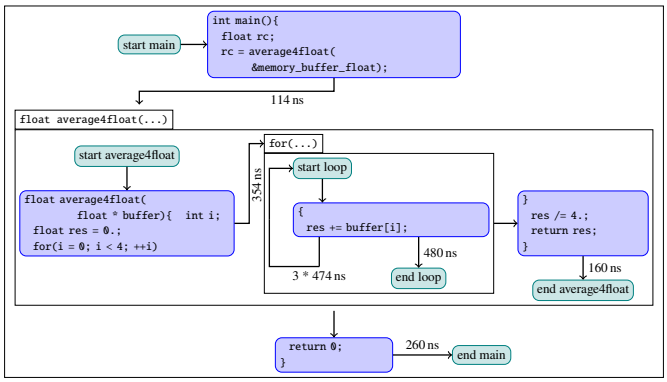
Based on the results in Fig. 2(a), only the temporal behavior, the accesses, the instruction cache loads and the control flow of the individual basic blocks are used in Fig. 2(b). The exact behavior of the application is no longer of interest here, since this is not relevant for the further analysis. Another representation, in which unneeded accesses (in this configuration, the conflict free accesses to the flash) were removed, is shown in Fig. 2(c). Here it is tried as far as possible to roll out loops and to represent branches in parallel, in order to make e.g. changes of the worst case paths in the control flow more visible. Such a change can occur if the collision-free run of an `if` branch takes longer than its counterpart, but the `else` branch is disturbed significantly longer by potential interference.

To reduce the analysis overhead, a grouping of basic blocks to super blocks can be performed, as a longer period can be considered and have been introduced by Chang et al. [**superblock**]. These super blocks are a well-known mechanism for code optimization in compiler construction. Such super blocks can only be entered by the first block, but left from any participating block. The benefit of the super blocks can mainly be used by loops (cf. Fig. 2(c), dashed), as here often the same sequence of basic blocks is repeated and no complex branch flows are present.

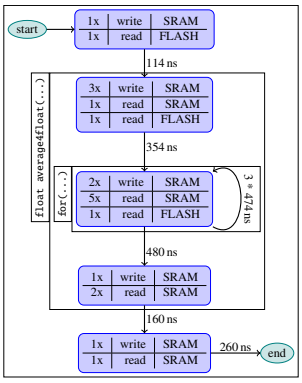
4.3 Access Pattern of the DMAC

In comparison to the access patterns of the functions executed by the CPU, existing analysis tools cannot be used for modeling the local and temporal DMAC accesses. To model a similar representation, different successive approaches can be implemented to obtain an upper bound of the accesses of the DMAC.

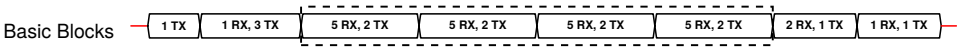
As a first approach here, the underlying protocol of the IO interfaces and the given DMAC configuration are used. Based on the physical transmission times of the interfaces and



(a) Control flow and WCET timing at code level.



(b) Simplified representation



(c) Abstract representation based on access Pattern

Fig. 2: Individual steps for calculation of the access pattern

the minimum size of a transmission, the transmission time of a message can be derived. For each copying of data via DMA, a potential conflict might occur on the access of the IO-interface and by the access of the data buffer in SRAM. If more complex interfaces are used, such as CAN or Ethernet, which allow a flexible data size, the minimum possible or, if known, the minimum used physical transmission size must be used. This guarantees, that the minimal distance of the disturbance is taken into consideration, which results in the maximum number of interferences within a given time.

As as a second approach, information from high level protocols and interface descriptions can optimize the analysis. In avionic systems, interfaces are usually described in an Interface

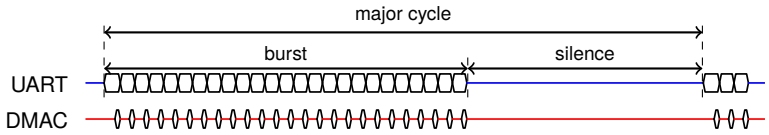


Fig. 3: Minimal Distance based on an ICD

Control Document (ICD). This is a detailed description document, which represents the timing, the size and various types of information about incoming and outgoing messages on the involved interfaces. Fig. 3 shows an ICD analysis of a major message cycle consisting of 25 individual transmissions sent within a given burst time followed by a bus silence. This reduces the transmissions and possible interferences within a given major cycle compared to the first analysis.

On an abstracted basis, the DMAC behaves just as an uncached part of the CPU, since it performs comparable memory accesses within a given time span. Based on the provided results, a comparable abstract model – as previously described for the CPU access pattern – can be generated.

4.4 Calculation of Possible Collisions and the corresponding WCET

By considering the individual abstract representations of the CPU- and DMAC-models, the potential collisions within the time period can be calculated. The upper limit is specified by combining the information of the execution time of the functions of the CPU and the maximum possible collisions initiated by the different DMA transfers. The first step is to determine which components will be combined together. In most cases, the CPU and the connected basic blocks (or the super blocks for a optimized analysis) are used as the main component. The interfaces are assumed to interfere this component. Subsequently, each basic block is considered independently and the delay caused by the interferences is calculated.

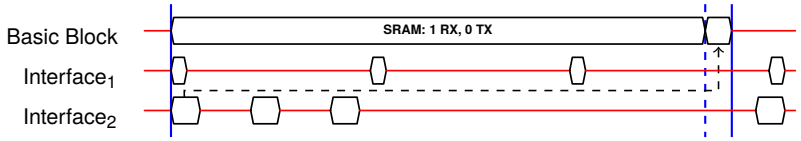


Fig. 4: Calculation of the maximum interference caused by asynchronous IO

For this purpose, the WCET of the individual basic block is taken and the maximum possible interferences of this period per interface are selected, cf. Fig. 4. Based on the potential interferences, the actual worse case collisions that can occur within this basic block are calculated. Since each memory access caused by the CPU can only be disturbed by one disturbance (or a given number of sequential disturbances, depending on the scheduling of the interconnection network), the worst case collisions can be estimated.

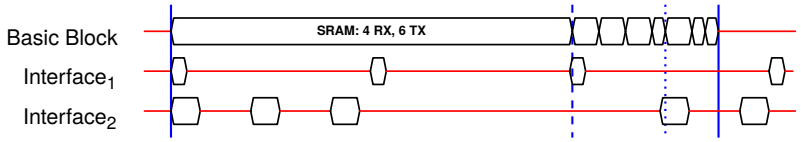


Fig. 5: Changes of the possible interference due to the iterative analysis

If several accesses are initiated by a basic block, the analysis process must be repeated iteratively. It should be considered that the worst case runtime of the basic block is increased by each disturbance and a new interval for possible disturbances must be considered. In Fig. 5, indicated by the dotted line, after the fourth iteration, another potential interference is performed by "Interface$_2$", which must be included in the calculation in the subsequent iterations.

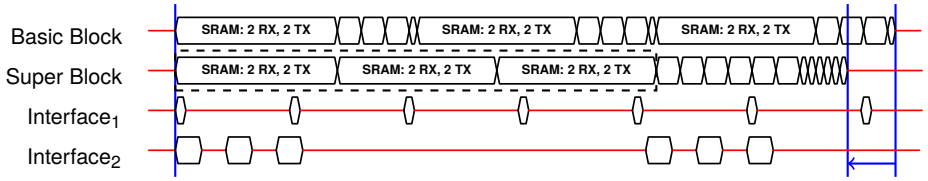


Fig. 6: Super Block with Shorter Collisions

One manageable approach for optimization of the overestimation is to use a sequence of related basic blocks, which are combined to a super block, cf. Fig. 6. For super blocks with simple control flow, like loops, the complete period of the unrolled loop can be analyzed. This has the advantage, that the worst case assumption is used for multiple blocks, which results in a closer overestimation to the actual, but not known, WCET.

The example in Fig. 6 illustrates the comparison of the super block optimization and the single basic block consideration. Both approaches induce the same number of potential collisions for these three loop blocks, but the estimated impact is different. When considering the basic blocks independently, each can be disturbed by one "Interface$_1$" and three "Interface$_2$" interruptions, which combines to three "Interface$_1$" and nine "Interface$_2$" interferences. In the super block constellation a longer period is considered and only six of the twelve potential collisions are disturbed by the long transfers of "Interface$_2$", whereas the other six accesses can only be interrupted by the short requests of "Interface$_1$".

In addition, it is noticeable from the examples (Fig. 4, 5 and 6) that an improvement over the pessimistic estimate does not always occur. Only if a blocks make more accesses to the memory than are initiated at the same time by the competing DMAC, a reduction of the calculated maximum interference is possible.

5 Realization

The test hardware used here is the MCU Hercules TMS570, a dual-core system running in lock step, which behaves like a single-core system on software side. The internal connection network in this MCU is based on two different crossbar switches and several bus systems that are connected in cascade. The devices and interfaces of the MCU are connected to these bus systems, where potential interference can be expected. Aside from the documentation provided for the hardware, not all parameters can be taken from the manuals, so some required parameters are evaluated and quantified using reverse engineering, as described in [**arcs18**]. For the applications used in the examples, the following delays can be observed for accesses to the SRAM: Both, the 8-bit (1x each UART DMA transfer) and the 32-bit (2x each CAN DMA transfer), memory transfers delay the CPU transfer to the SRAM by 4 CPU cycles in the worst case ($4 * 1/300\,\text{MHz} \approx 13.3\,\text{ns}$), because the built-in crossbar switch initiates a minimum bus transfer of 32 bit.

To obtain a typical representation of the IO-accesses in the laboratory setup, the ICD of the Open Inovation Research Demonstrator "SAGITTA" is used. This ICD describes the used interfaces (two UART and two CAN each), the expected data and their external bus-schedule. The two CAN interfaces generate access patterns where messages arrive at a minimum distance of 178 µs, and initiate two consecutive 32 bit DMA transfers each. For the two UART-interface, access patterns can be derived where each initiating a single 8 bit transfer within 1.38 ms. Since the test functions share only SRAM accesses with the DMAC, unneeded access paths are not considered for the further analysis.

As test-functions, a selection of single-core WCET-benchmarks [**benchmark**] (bsort100, cnt, matmult) are used. To calculate the access patterns, the functions are evaluated using the WCET-Analysis-tool *aiT* from *AbsInt* and subsequently abstracted into access patterns for follow-up asynchronous computation. The selection criteria of the benchmarks are mainly based on the complexity of memory accesses and not on the actual algorithm.

As a reference, the pessimistic approach, which is used currently for DMAC based approaches, is calculated. For this, a delay of 4 cycles (13.3 ns) is assumed for each data access and instruction cache miss. This assumption calculates a safe upper bound, but an overestimation must be expected. For the example functions, a Worst Case Slowdown Factor (WCSF) in the range of 30 % to 40 % can be calculated (Tab. 1), which is comparable to the results of the state of the art, like mentioned in publication [**pessimistisch**].

For the novel analytical approach, only the possible collisions per basic block or super block are calculated. This results in a more realistic estimate of the upper bound compared to the previously mentioned pessimistic approach. The reduction of the overestimation always occurs as long as the blocks generate more accesses than can be induced at the same time by the DMAC. For the example function *cnt*, an additional new path analysis must also be performed, since alternative branches can be traversed. In this case, the analysis is simple,

since both paths induce an identical number and size of accesses, which means that no change in the WCET-path occurs.

6 Results and Outlook

The calculation of WCET with asynchronous IO by the novel approach allows to achieve a reduced upper bound of WCET compared to the pessimistic approach, cf. Tab. 1. Only a WCSF in the range of 11 % to 23 % occurs in the example functions using the basic block analysis. This allows an improvement compared to the pessimistic WCSF between 40 % for *cnt* and up to 67 % for *matmult*. The super block analysis improves the reduction compared to the pessimistic WCSF in a minimum of 1.5 % for *matmult*.

benchmark	conflict free	pessimistic		basic block		super block	
bsort100	17.08 ms	23.91 ms	40.0 %	20.30 ms	18.9 %	17.11 ms	0.2 %
cnt	9.38 ms	12.99 ms	38.6 %	11.53 ms	22.8 %	9.40 ms	0.3 %
matmult	6.31 ms	8.43 ms	33.7 %	7.00 ms	11.1 %	6.40 ms	1.5 %

Tab. 1: Results (rounded up) of the WCET with shared resources

This concept overcomes the restrictions of the current static WCET analysis and encourages the use of parallel accesses to memory areas by DMAC. For this purpose, the verified system is additionally analyzed for possible internal interference and its effect, and is combined with the access patterns of the CPU and DMAC. However, the initial additional work, especially in analyzing the hardware, only needs to be performed once for each MCU. Subsequently, this information can be used for any system using this MCU and is comparable to the initial effort for integrating new CPUs into WCET tools. The transfer of the software and the ICD into the access patterns must be performed individually for each project, but is comparable to the current static WCET analyses.

The increase of the WCET compared to conflict-free analyses must be expected with this approach, but it is lower than the pessimistic approaches. A worse result compared to the pessimistic approach cannot occur, as each access is added with a corresponding collision. This result represents the upper bound of the calculation and corresponds to the pessimistic evaluation. However, with the DMA operation, the CPU driven processing of the IO in software can be skipped. As the total WCET of the application is lower in many cases, the additional effort is justified. In addition, the certification overhead of the software may also be reduced because the CPU driven scheduling becomes simpler, since the processing of the IO is eliminated. Therefore, in a future research the complete system software has to be analyzed and the novel approach has to be compared to each other.

Since this approach is based on abstract representation of access patterns, the analysis method is not limited to MCU with a single CPU and one DMAC, but can also be used for multi core systems with dedicated IO-CPUs. By using dedicated IO-CPUs, the overestimation can be further reduced, since defined synchronization ranges can be specified on freely

programmable units and potential conflicts only occur in this predefined time slots. For general multi core systems a significant higher effort must investigated, as challenges like core changes, caching behaviors, etc. has to be taken into account. To further reduce the overestimation of the upper bound, these potential conflicts can be reduced by implementing techniques such as storing local data in scratchpad memory [**scratchpad**] or targeted cache prefetching [**prefetch**].

Acknowledgement

The author would like to thank the “Verbundkolleg Mobilität & Verkehr of the Bayerisches Wissenschaftsforum” (BayWISS) at the Technischen Hochschule Ingolstadt, Airbus Defence and Space and AbsInt Angewandte Informatik GmbH for supporting this work. Additionally, the author would like to express an additional thanks to the anonymous reviewers who provided useful input for the paper and further research.