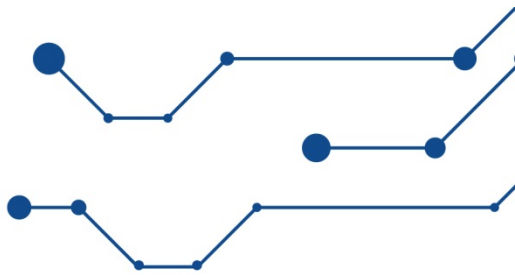
fortiss

RTSOPS, 2017

Estimating Worst-Case Bounds for Open CPS Runtimes with Genetic Algorithms

Oliver Horst, Uwe Baumgarten, Christian Prehofer

fortiss GmbH
An-Institut Technische Universität München



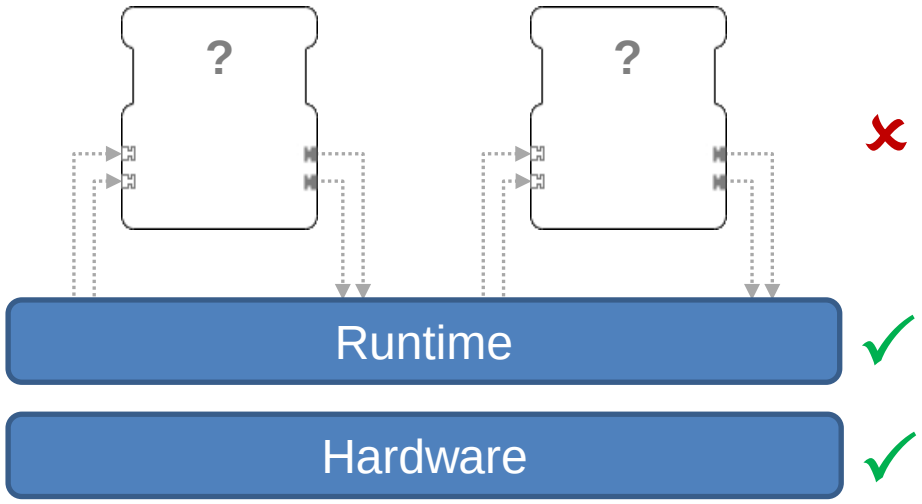
Content

1. Problem Statement
2. Related Work
3. Utilizing Genetic Algorithms to Estimate Worst-Case Bounds of Open CPS Runtime APIs
4. Research Challenge No.1: Defining the Level of Detail of the Hardware Model
5. Research Challenge No.2: Quantifying the Quality of the Determined Bounds
6. Conclusion

Characteristics of Open CPS

- Open CPS are CPS that can be extended at runtime with new functionality (so called Apps)
- The final deployment configuration is unknown until runtime
- Only the platform (i.e., runtime and hardware) is known, but everything on top is unknown

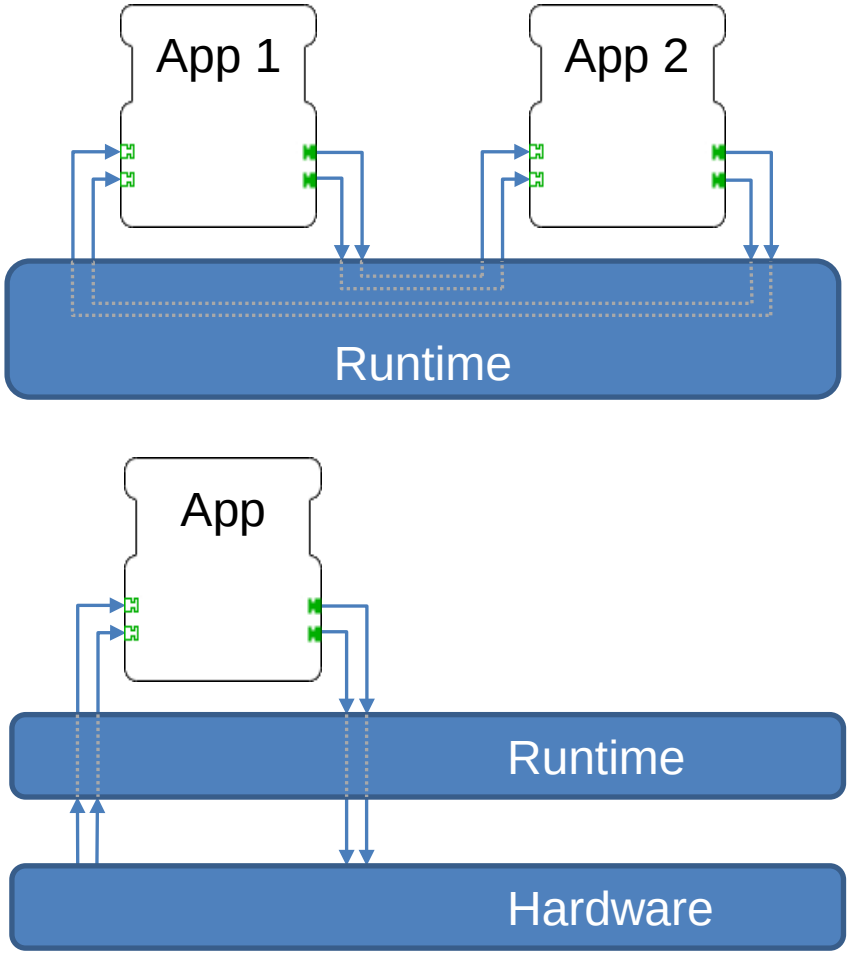
➡ Thus, it is important to determine the WCET of each App individually and independently



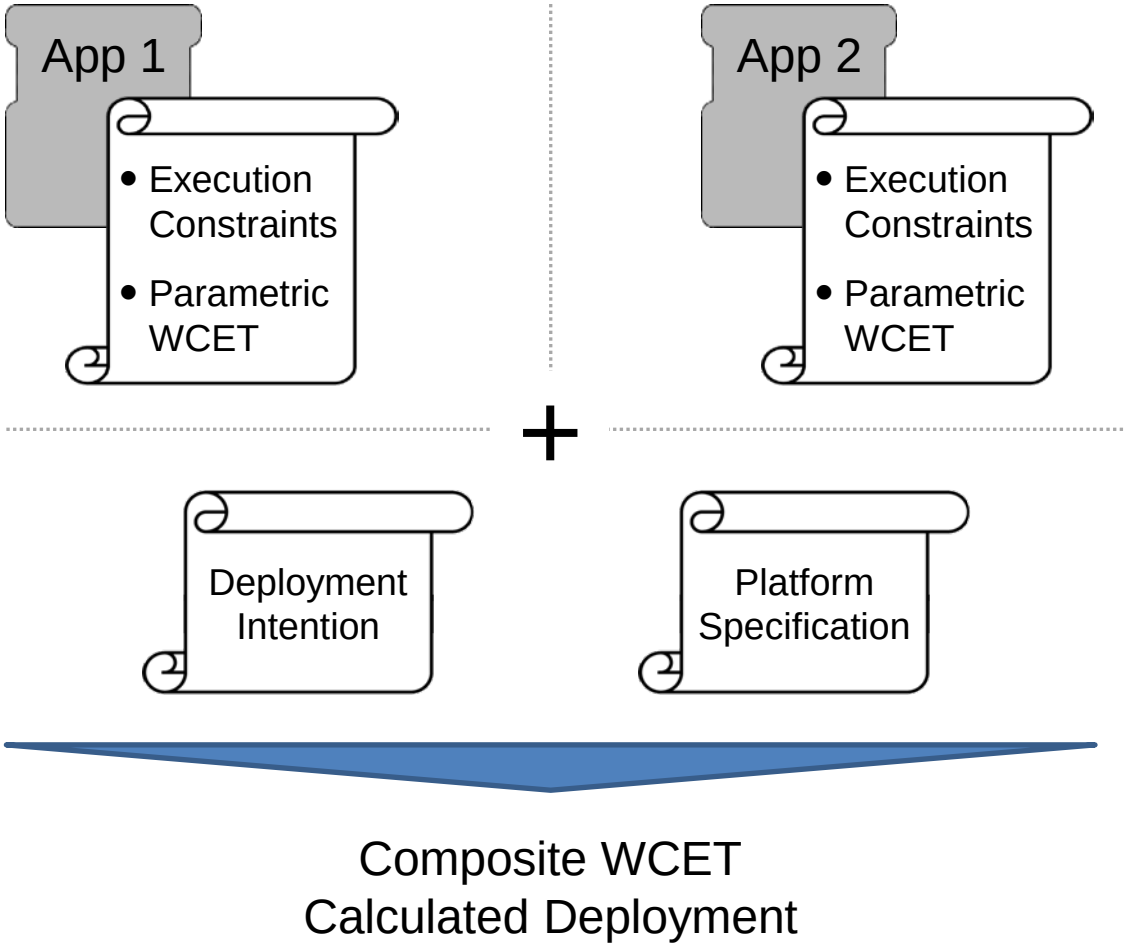
Characteristics of Open CPS Runtime APIs

- Reduced amount of APIs with focused functionality to lower the risk of attacks
- Interactions between apps and to the hardware are managed and coordinated by the runtime

➡ The API guarantees apps a compositionality with respect to functional and non-functional (e.g., timing) properties, according to [2,3]



Compositional WCET Analysis



- The WCET of each app can be calculated depending on the WCET of the API calls of the runtime
- During deployment the composite WCET can be calculated [1]

➡ Allows to make claims on real-time properties for deployed apps

Related Work

Determining WCETs for API Calls in Open Platforms

Static Analysis

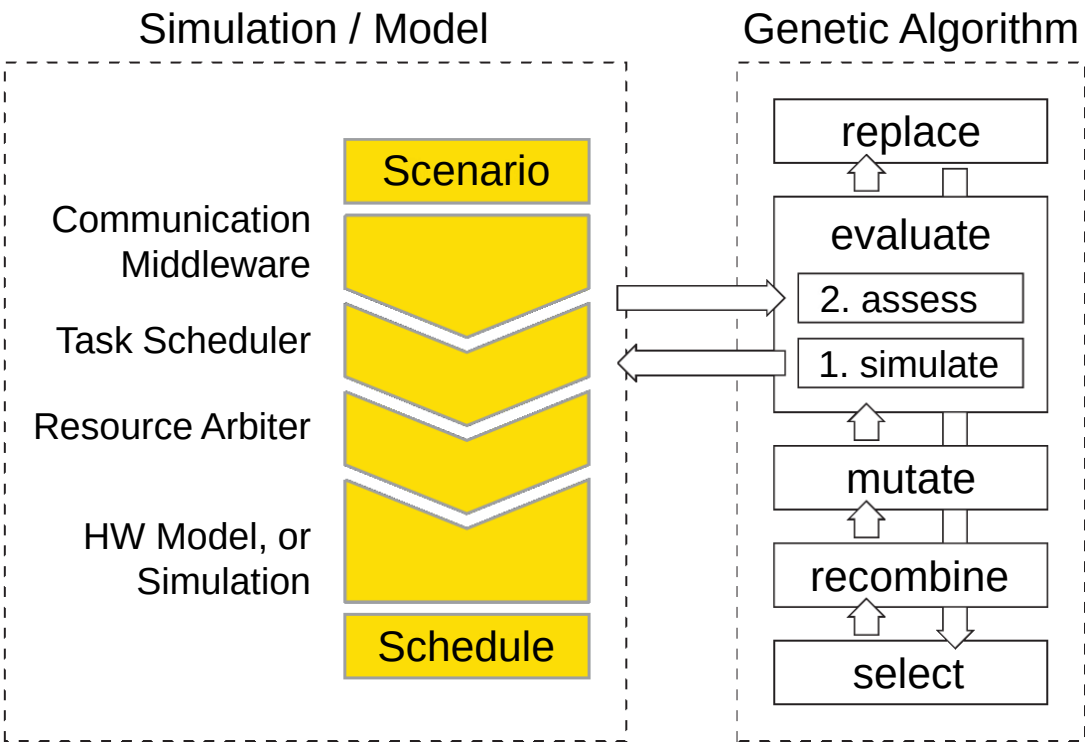
- Faces some limitations
  - Restrictions in reconstructing the CFG
  - Complexity of the underlying hardware, such as, multi-core interference, caches and pipelines
- Requires the final, complete binary
- Determines safe upper bounds for the WCET

Analysis by Genetic Algorithms

- Used in three related areas
  - Search based WCET analysis
  - Evolutionary testing of binaries
  - Deriving stress test for task schedulers
- It has been shown that GA-based approaches can obtain WCETs in the range of 3-10% of predeter-mined safe upper bounds [4]
- Determines only a lower bound for the WCET

Estimating Worst-Case Bounds with GAs

The General Idea



- The GA tries to find an execution scenario that provokes the BCET or WCET of a particular API call
- Each scenario is evaluated for its timing by a model or a simulation to steer the evolution process

Scenario: Definition and Evaluation

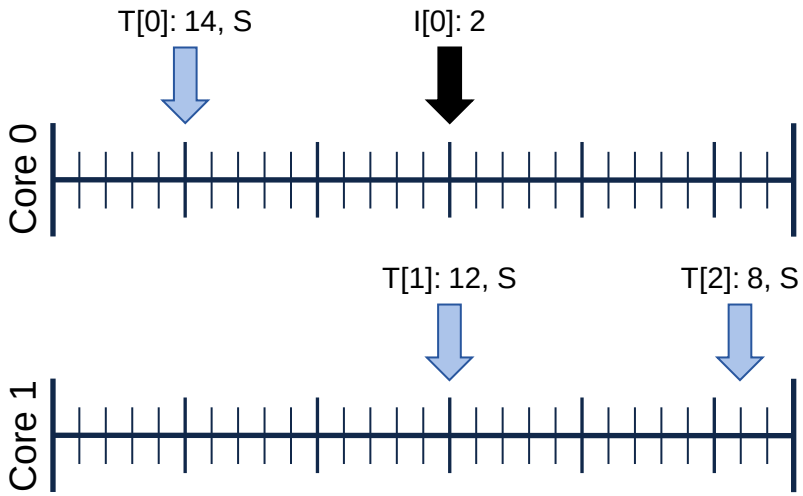
- A scenario models the interaction of tasks with the runtime, interfered by interrupts
- Each generated scenario consists of three sets
  - Task set
  - Interrupt set
  - Message set

  defined over a fixed time interval

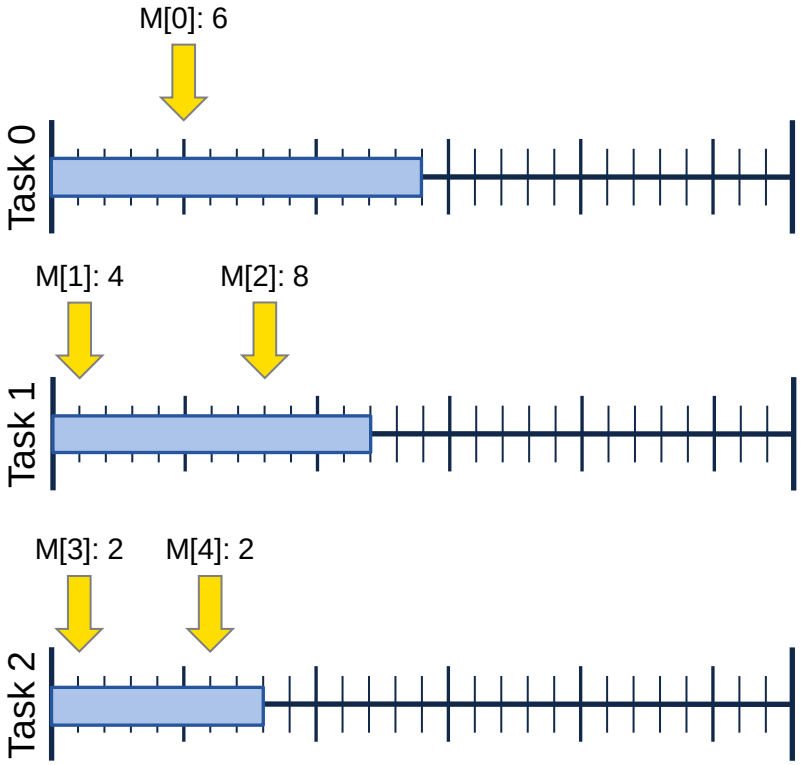
- Our model targets platforms with a finite impulse response
- Determining the WCET/BCET requires the evaluation of three full system cycles
- The platform specifies the length of one system cycle
- Programming errors, e.g., buffer overflows, and timing impacts of specific app instructions onto the runtime code are explicitly excluded from our considerations

Scenario: An Example

Task and Interrupt Set



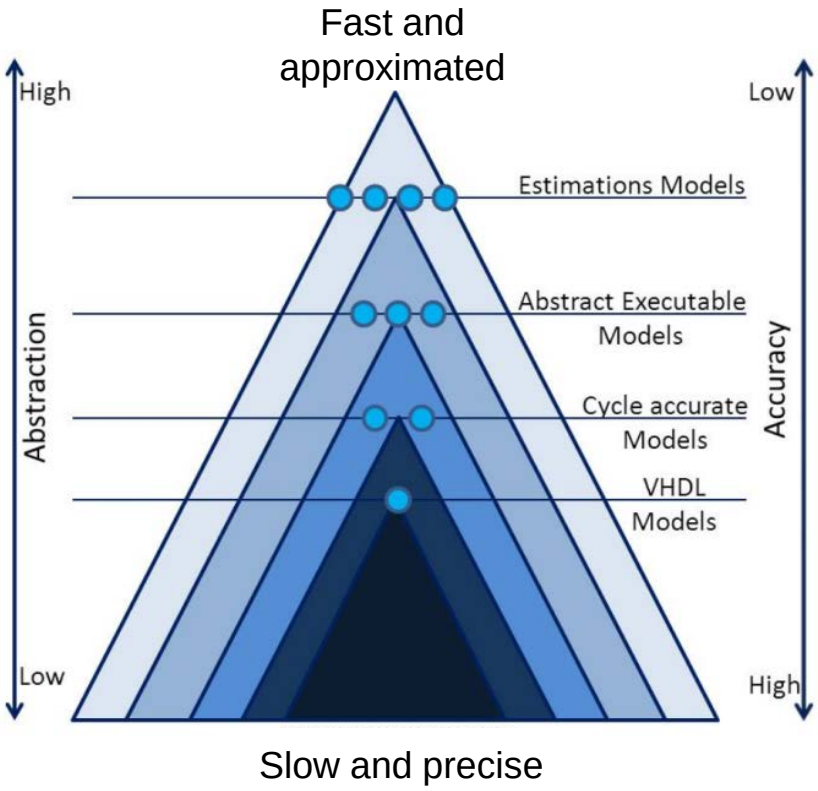
Message Set



Defining the Level of Detail of the HW Model

Research Challenge No.1

- Tension in the field of slow and precise or fast and approximated results
- Benchmarks indicate that 10^5 calls to the fitness function are needed per dimension for good results [5]
- We assume a complexity of ~10^3 dimensions
- We see two relevant options:
  - Abstract HW Model
  - Cycle Accurate Model



© Butko, Anastasiia, et al. "Accuracy evaluation of gem5 simulator system." Int. Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC). 2012.

Relevant Hardware Model Options

Abstract HW Model

- Ideally, models all hardware internals like bus and device arbitration schemes, memory delays, caches, etc.
- Needs to consider a lot of components and special cases

➡ How and at which level would you model the hardware?

➡ How close could the model reproduce the accurate timing?

Cycle Accurate Model

- Tests the whole software stack as is with the generated scenario as stimuli
- Can determine very precise and realistic timing

➡ What are possible simulation environments you would consider as suitable?

➡ Should something in-between the two options be considered?

Quantifying the Quality of the Determined Bounds

Research Challenge No.2

- Formally proving the convergence of metaheuristics, such as genetic algorithms, is a relatively young research area
- Proofs for simple versions of the algorithm and standard problems exists, but assume an infinite runtime
- Considerations for the convergence speed of metaheuristics are still not mature and very limited with respect to their application fields

➡ At the moment, it cannot be formally proven that after a certain number of generations a given quality goal is reached!

Non-formal Quantification Options

Comparison against known WCET bounds

- Determine the WCET for known and representative deployment configurations with the help of static-analysis tools
- Compare the results of the GA with the obtained results

Empirical study on mathematically similar functions

- Study the convergence quality of the GA with mathematical functions that calculate the timing for a given scenario and rebuild the structure of the real problem as close as possible

➡ Which of the two variants do you consider as more suitable?

➡ How many samples would you consider as sufficient to prove the convergence quality of the presented approach?

Conclusion

- Characteristics of open CPS and their APIs
- Approach to estimate worst-case bounds of open CPS runtime API calls with genetic algorithms, based on the generation of execution scenarios, which consists of three sets:
  - Task set
  - Interrupt set
  - Message set

➡ **Two open research problems:**

1. Defining the level of detail of the hardware model
2. Quantifying the quality of the determined bounds

Contact // Oliver Horst

fortiss GmbH
An-Institut Technische Universität München
Guerickestraße 25 · 80805 München · Germany

tel +49 89 3603522 15 **fax** +49 89 3603522 50

oliver.horst@fortiss.org
www.fortiss.org

Bibliography

[1] Leveque, T., Borde, E., Marref, A. and Carlson, J., Hierarchical Composition of Parametric WCET in a Component Based Approach, Proceedings of the 4th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2011, pp. 261-268

[2] Baldovin, A., Mezzetti, E. and Vardanega, T., Towards a Time-Composable Operating System, Keller, H. B., Plödereder, E., Dencker, P. & Klenk, H. (ed.), Reliable Software Technologies -- Ada-Europe 2013: 18th Ada-Europe Int.l Conf. on Reliable Software Technologies, Berlin, Germany, June 10-14, 2013. Proceedings, Springer, 2013, pp. 143-160

[3] Hahn, S., Reineke, J. and Wilhelm, R., Towards Compositionality in Execution Time Analysis: Definition and Challenges, ACM SIGBED Review, Association for Computing Machinery (ACM), 2015, Vol. 12(1)

[4] Wegener, Joachim, and Frank Mueller. "A comparison of static analysis and evolutionary testing for the verification of timing constraints." Real-Time Systems 21.3 (2001): 241-268.

[5] http://coco.gforge.inria.fr/

Mathematical Characterization of the Problem (1/2)

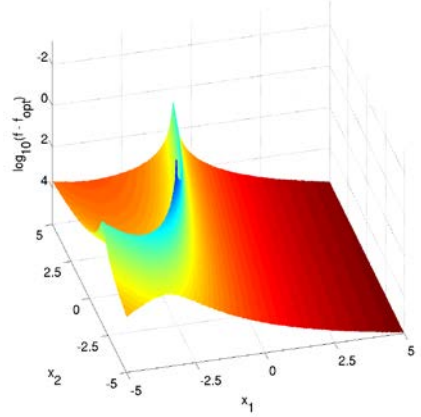
To assess the suitability of specific genetic algorithms for tackling our problem, a mathematical test function is needed. In the mathematical sense, we cope with a discrete function, which is costly to evaluate and is defined by

- multiple dimensions (e.g., > 1.000), with
- many local maxima,
- multiple global maxima,
- plateaus, valleys, and
- cross linked dimensions.

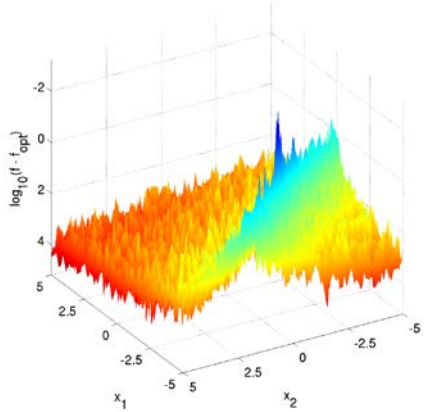
The Black-Box-Optimization-Benchmarking (BBOB) workshop developed a number of benchmarking functions for genetic algorithms. Six of them match our problem statement and could be used to evaluate the suitability of genetic algorithms.

Mathematic Characterization of the Problem (2/2)

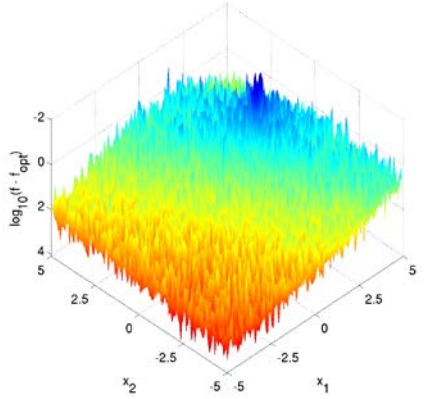
A BBOB Function Selection



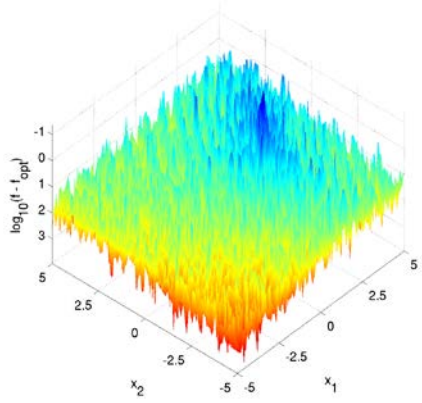
f105: Rosenbrock with moderate uniform noise



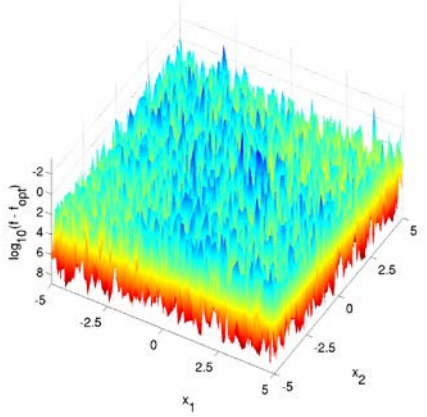
f113: Step ellipsoid with gaussian noise



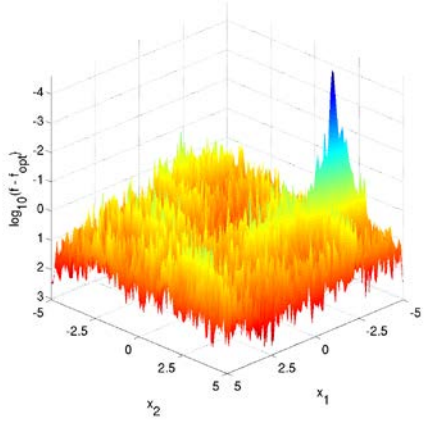
f119: Different Powers with gaussian noise



f122: Schaffer's F7 with gaussian noise



f126: Composite Griewank-Rosenbrock with uniform noise



f128: Gallagher's Gaussian Peaks 101-me with gaussian noise

Relevant Hardware Model Properties

At the Example of Shared Resources on Multicore Systems

Shared resource	Mechanism
System bus	Contention by multiple cores Contention by other device - IO, DMA, etc. Contention by coherency mechanism traffic
Bridges	Contention by other connected busses
Memory bus and controller	Concurrent access
Memory (DRAM)	Interleaved access by multiple cores causes address set-up delay Delay by memory refresh
Shared cache	Cache line eviction Contention due to concurrent access Coherency: Read delayed due to invalidated entry Coherency: Delay due to contention by coherency mechanism read requested by lower level cache Coherency: Contention by coherency mechanism on this level

Shared resource	Mechanism
Local cache	Coherency: Read delayed due to invalidated entry Coherency: Contention by coherency mechanism read
TLBs	Coherency overhead
Addressable devices	Overhead of locking mechanism accessing the memory I/O Device state altered by other thread/application Interrupt routing overhead Contention on the addressable device - e.g. DMA, Interrupt controller, etc. Synchronous access of other bus by the addressable device (e.g. DMA)
Pipeline stages	Contention by parallel hyperthreads
Logical units	Contention by parallel applications
	Other platform-specific effects, e.g. BIOS Handlers, Automated task migration, Cache stashing, etc.

© Kotaba, Ondrej, et al. "Multicore in real-time systems–temporal isolation challenges due to shared resources." Workshop on Industry-Driven Approaches for Cost-effective Certification of Safety-Critical, Mixed-Criticality Systems. 2014.