Multi-Objective Intrinsic Evolution of Embedded Systems (MOVES)

Paul Kaufmann, Marco Platzner

Computer Engineering Group

University of Paderborn

{paul.kaufmann, platzner}@uni-paderborn.de

Outline

- motivation/vision
- Δ to last status meeting
- Reconfiguration schemes for EHW classifier
- publications, collaborations

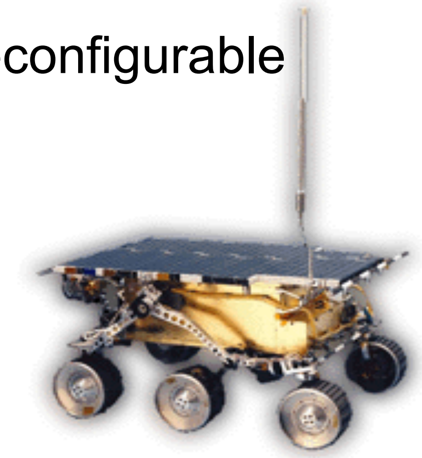
Motivation / Vision

- investigate simulated evolution as a mechanism to achieve self-adaptation and –optimization for autonomous embedded systems

- an embedded system should be capable of adapting to …
  - the environment
  - changes in resources

- adaptability achieved by combining intrinsic evolution with reconfigurable hardware (evolvable hardware, EHW)

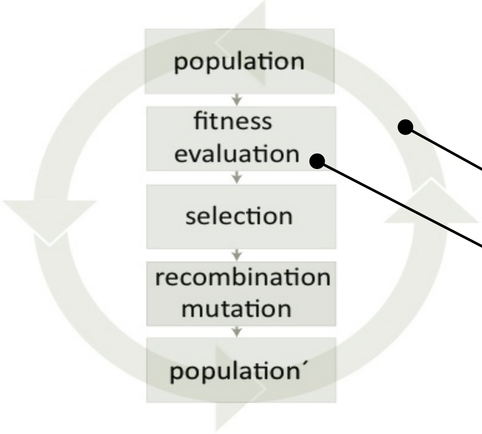
- working areas
  1. models and algorithms
  2. system architectures
  3. case studies, evaluation

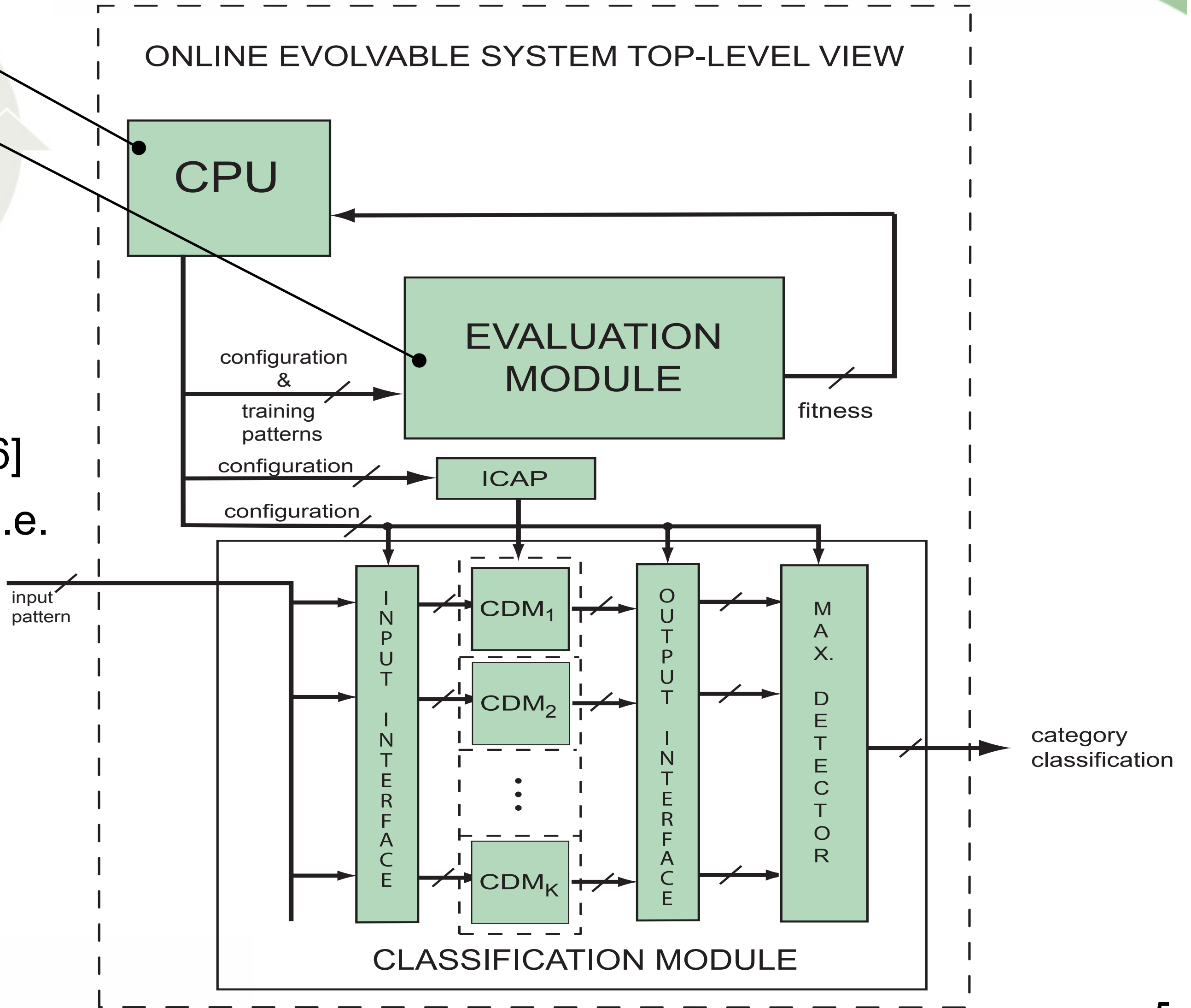


Δ to Last Status Meeting

- last status meeting
  - evolutionary algorithms
    - periodization of local and global search [Kaufmann et al., CEC '10]
  - evolvable hardware architecture
    - EHW classifier adaptation [Knieper et al., ICES '10]
  - application examples
    - prosthetic hand controllers [Kaufmann et al., EMBC '10]
- new work done
  - evolvable hardware architecture
    - <u>reconfiguration schemes for EHW classifiers</u> [Kaufmann et al., IJARAS '11, to app.]
  - application examples
    - lower-limb gait detection [Boschmann et al., ICBBT '11]
  - algorithms & applications [Miller (ed.), Cartesian Genetic Programming, Springer]

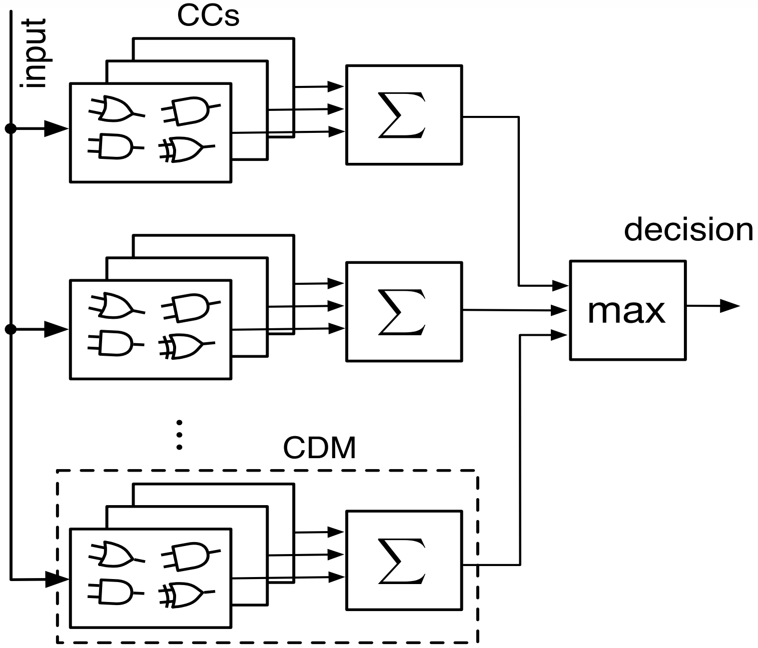
Functional Unit Row (FUR) Architecture





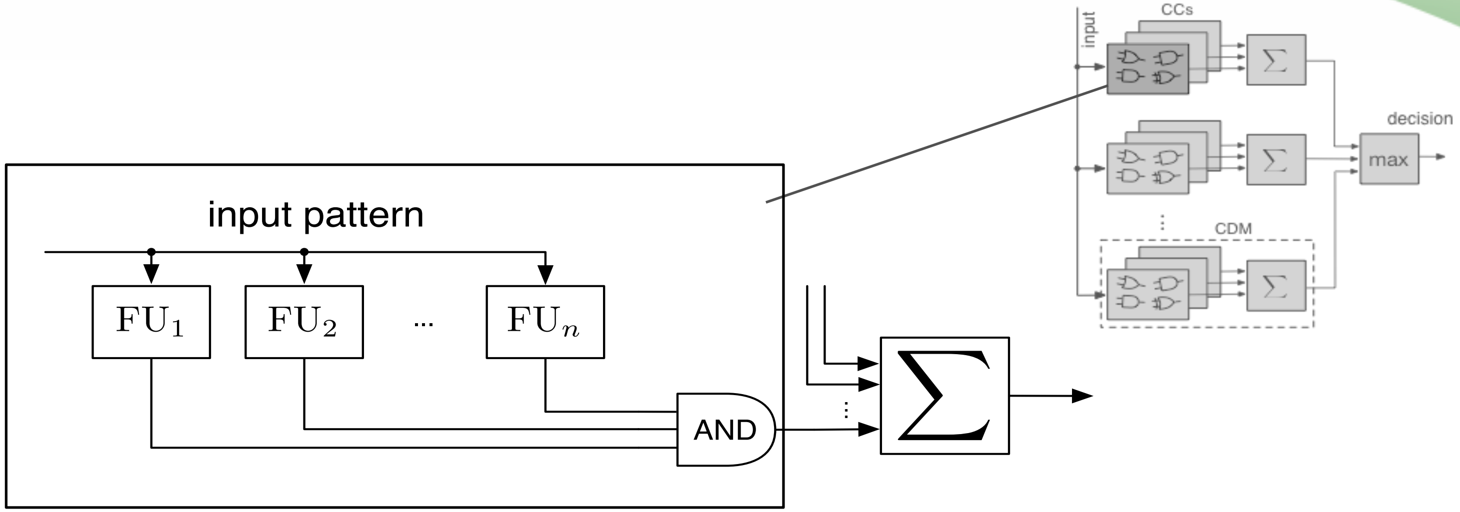
- by Glette & Torresen [Glette' 06]
- Virtual Reconfigurable Circuit, i.e. reconfiguration through registers / mux controls

Functional Unit Row Architecture (1)



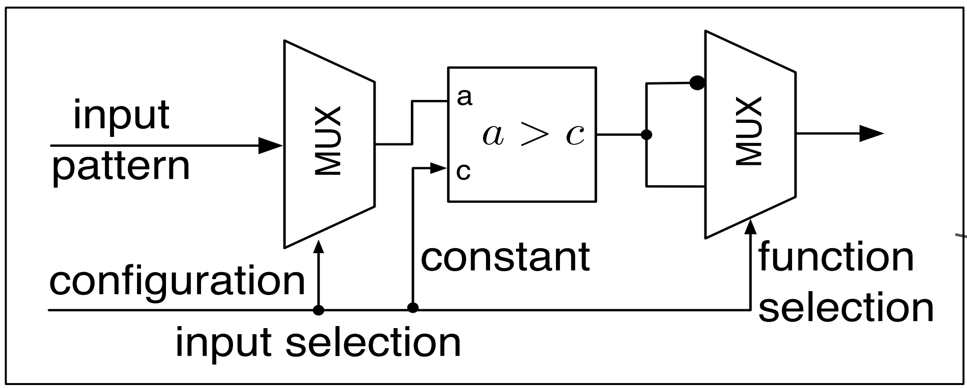
- FUR architecture comprises a Category Detection Module (CDM) for each category to be classified
- CDM contains a number of basic pattern matching elements (CC)
- category with most activated pattern matching elements defines classifier's decision

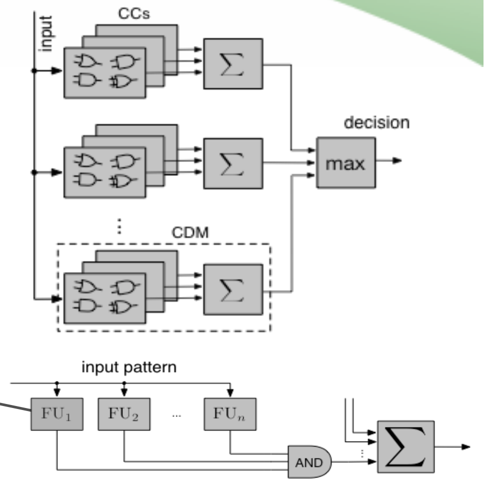
Functional Unit Row Architecture (2)



- Category Classifier (CC) decides, if the given input vector corresponds to its category
- Functional Unit (FU) implements a decision rule
- CC is a conjunction of a number of decision rules

Functional Unit Row Architecture (3)

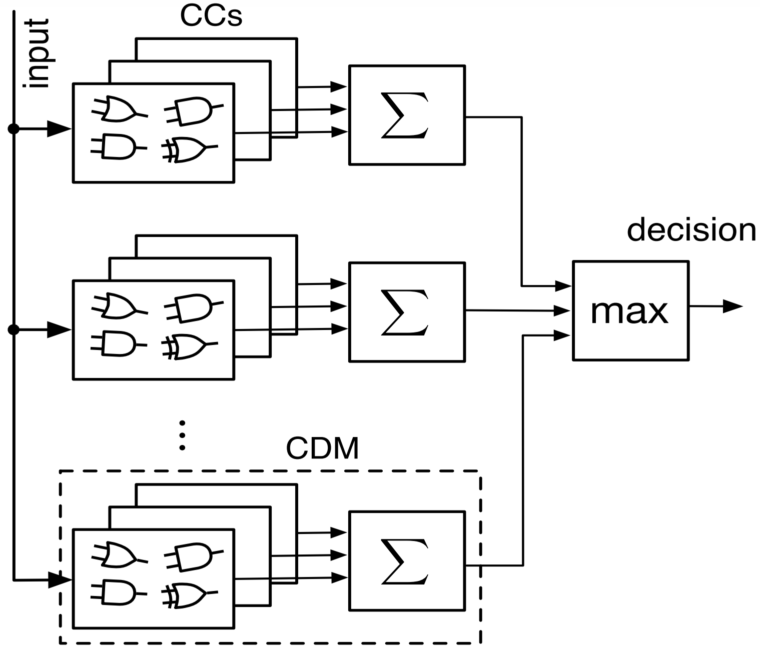




- Functional Unit (FU) compares a selected input value to a constant
  – similar to Decision Trees
- FU configuration is subject to evolutionary optimization
  – selection of the input value, reference constant, function selection

FUR's Fitness Definition

- n – number of categories and Category Detection Modules (CDM)
- V=(v,l) – labeled / classified input vectors



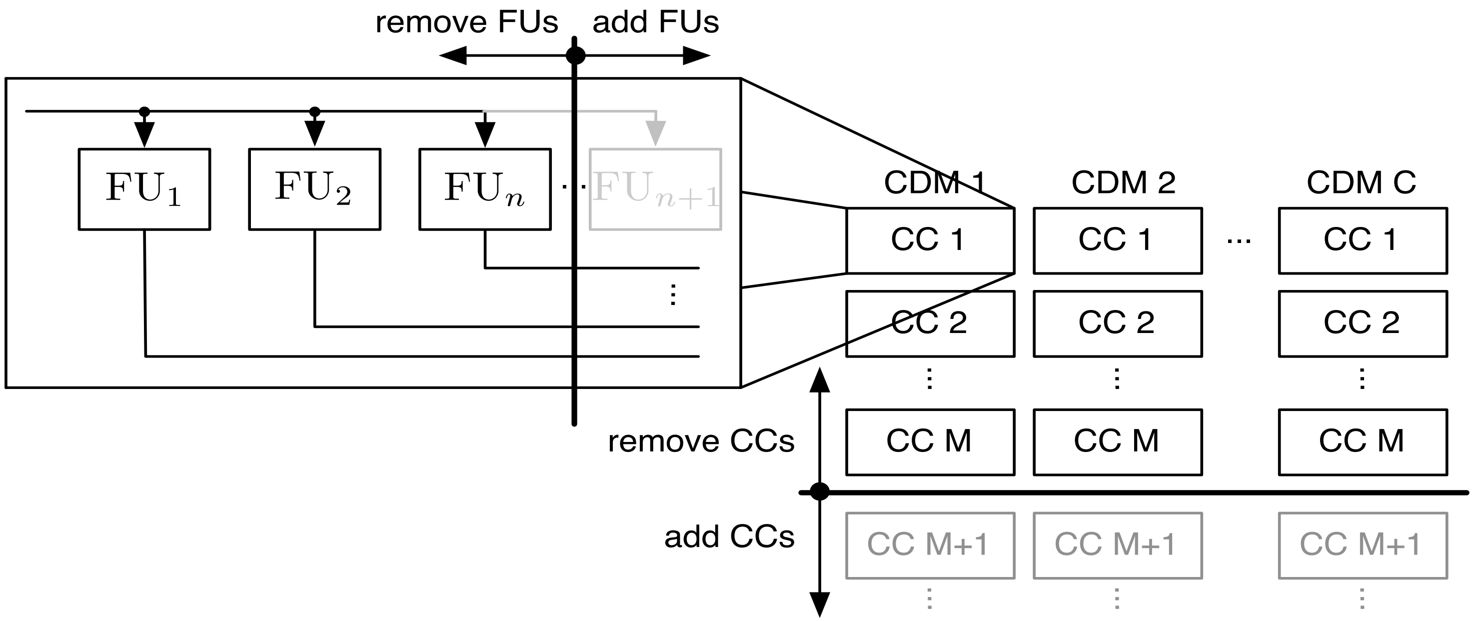
$$\text{accuracy(FUR)} = \frac{1}{|V|} \sum_{(v,l)\in V} \begin{cases} 1 & : \text{ if } (\max_{i=1}^{n} \mathrm{CDM}_i(v) = \mathrm{CDM}_k) \ \&\& \ (k = l) \\ 0 & : \text{ else.} \end{cases}$$

EHW Classifier Adaptation

- previous work:
  - FUR architecture applied to classification of electromyographic signals
    - [Glette, Kaufmann, Torresen, Platzner: ICES'08]
    - [Glette, Gruber, Kaufmann, Torresen, Sick, Platzner: AHS'08]
  - investigation of run-time reconfigurable FUR architectures
    - [Knieper, Kaufmann, Glette, Platzner, Torresen: ICES'10]
    - [Kaufmann, Glette, Platzner, Torresen: IJARAS'11]

- new work: improve classification behaviour during architectural reconfigurations
  - questions:
    - how large are the accuracy drops during architectural reconfigurations?
    - what kind of strategies can be used to reduce the impact of architectural reconfigurations?

Reconfigurable FUR Architecture



- reconfigurable FUR architecture shows two degrees of freedom
  - number of FUs in a CC
    - depends largely on the application
  - <u>number of CCs in a CDM</u>

Architectural Reconfiguration Strategies (2)

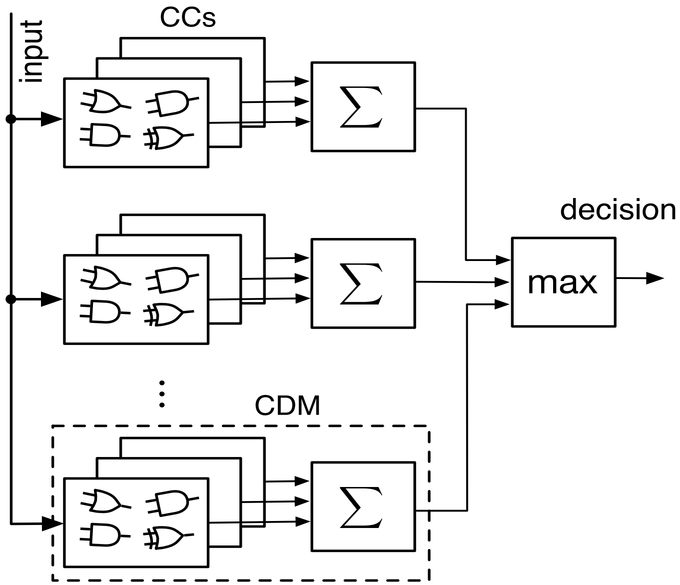
- introduce a penalty counter for every CC

CC induction / replacement strategy	**increase FUR's size**	**decrease FUR's size**
randomly	initialize new CCs randomly	remove randomly selected CCs
low penalty selection scheme	duplicate CCs with lowest penalty counter	remove CCs with lowest penalty counter
high penalty selection scheme	duplicate CCs with highest penalty counter	remove CCs with highest penalty counter

- baseline method: induce randomly initialized, remove randomly selected CCs
  - requires no extension of the FUR architecture

Architectural Reconfiguration Strategies (1)

- penalizing false negative CCs?
  - CC should compute a "match" instead, it computes a "miss"
  - forces all CCs of a CDM to compute a "match" for a corresponding input vector
    - reduces classification rule diversity



→ penalize false positive CCs
  - for a false positive CC increase penalty counter by the number of false positive CCs in the same CDM

- $V=(v,l)_j$ – set of labeled / classified training data

$$\text{penalty}(\text{CC} \in \text{CDM}_k, (v,l) \in V, k \neq l, \text{CC}(v) = 1) = \sum_{\text{CC}_i \in \text{CDM}_k} \text{CC}_i(v)$$

Benchmarks

- UCI machine learning repository
  - Pima Indian Diabetes data set
    - 768 feature vectors, 8 values in a feature vector
    - 500 samples from negative-tested subjects
    - 268 samples from positive-tested subjects
  - Thyroid data set
    - 7200 feature vectors, 22 values in a feature vector
    - 6.666 samples from regular subjects
    - 166 samples from subjects with sub-normal function
    - 368 samples form subjects with hyper-normal function

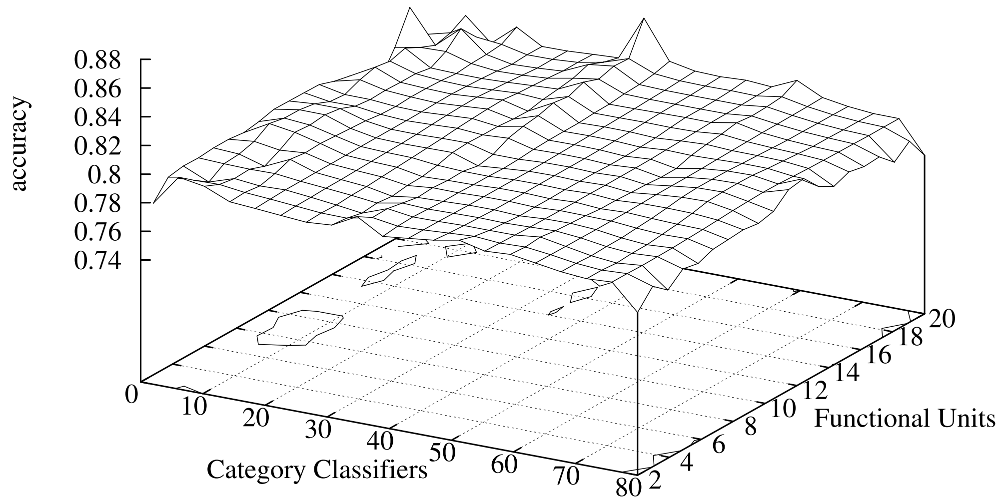
- benchmark selected because of the pronounced experiment results

Experiment configuration

- 1st experiment: investigate FUR architectures using grid search over the number of FUs and CCs
  - employ 12-fold cross validation, 100.000 generations
- 2nd experiment: use best configuration found for reconfigurable FUR
  - 4 FUs per CC, generations between changes in CCs: 50.000
  - number of CCs:
    - 2.1: gradual changes
      - 10→9→8→7→6→5→4→3→2→1
      - 1→2→3→4→5→6→7→8→9→10
    - 2.2: radical changes
      - 10→4→2
      - 2→5→10
- algorithm: 1+4 ES
  - three genes are mutated in each CC per generation
  - complete architecture is evolved in a single run

The Pima Benchmark

- general FUR performance for the Pima benchmark



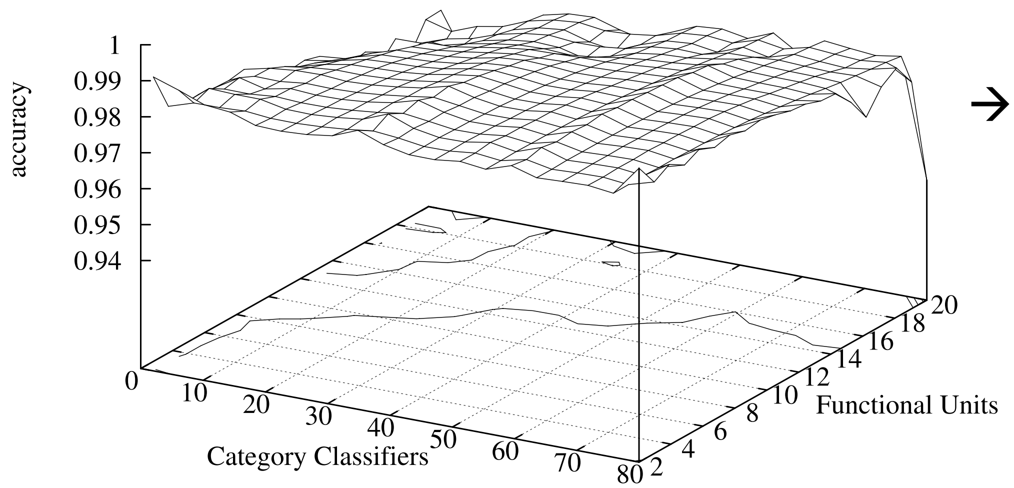
- comparison of test accuracies in %
FUR configuration: (40, 4)

Algorithm	Error Rate	± SD
→ **FUR**	**21.35**	
SVM*	22.79	4.84
LDA*	23.18	4.64
Shared Kernel Models	23.27	2.56
kNN*	23.56	3.07
GP with OS, |pop|=1.000	24.47	3.69
CART*	25.00	3.61
DT*	25.13	4.30
GP with OS, |pop|=100	25.13	4.95
MLP*	25.26	4.50
Enhanced GP	25.80 – 24.20	
Simple GP	26.30	
ANN	26.41 – 22.59	1.91 – 2.26
EP / kNN	27.10	
Enhanced GP (Eggermont et al.)	27.70 – 25.90	
GP	27.85 – 23.09	1.29 – 1.49
GA / kNN	29.60	
GP (de Falco et al.)	30.36 – 24.84	0.29 – 1.30
Bayes	33.40	

* own experiments

The Thyroid Benchmark

- general FUR performance for the Thyroid benchmark



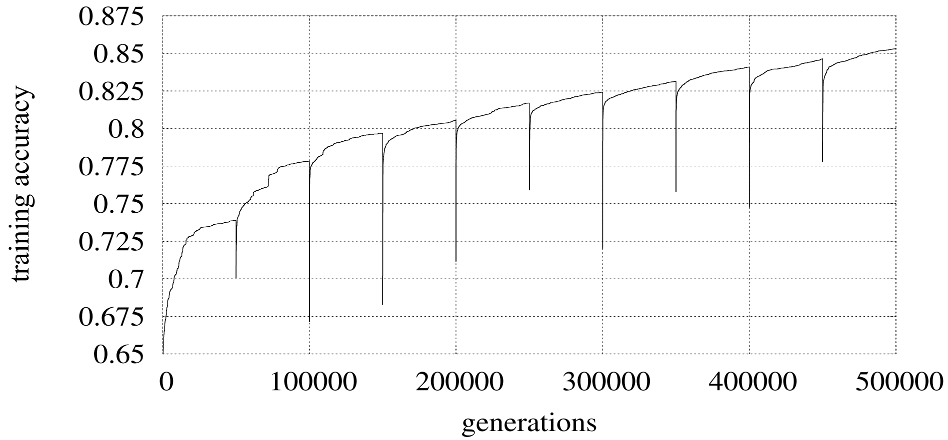
- comparison of test accuracies in %
FUR configuration: (40, 4)

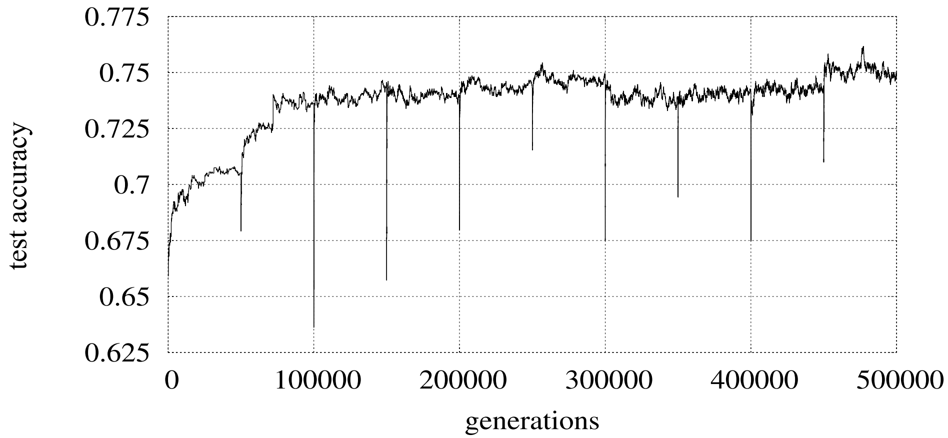
Algorithm	Error Rate	± Standard Deviation
DT*	0.29	0.18
CART*	0.42	0.27
CART	0.64	
PVM	0.67	
Logical Rules	0.70	
FUR	**1.03**	
GP with OS	1.24	
GP	1.44 – 0.89	
BP + local adapt. rates	1.50	
ANN	1.52	
BP + genetic opt.	1.60	
GP	1.60 – 0.73	
Quickprop	1.70	
RPROP	2.00	
GP (Gathercole et al.)	2.29 – 1.36	
SVM*	2.35	0.51
MLP*	2.38	0.62
ANN	2.38 – 1.81	
PGPC	2.74	
GP (Brameier et al.)	5.10 – 1.80	
kNN*	5.96	0.44

* own experiments

Increasing Resources: Pima Benchmark (1)

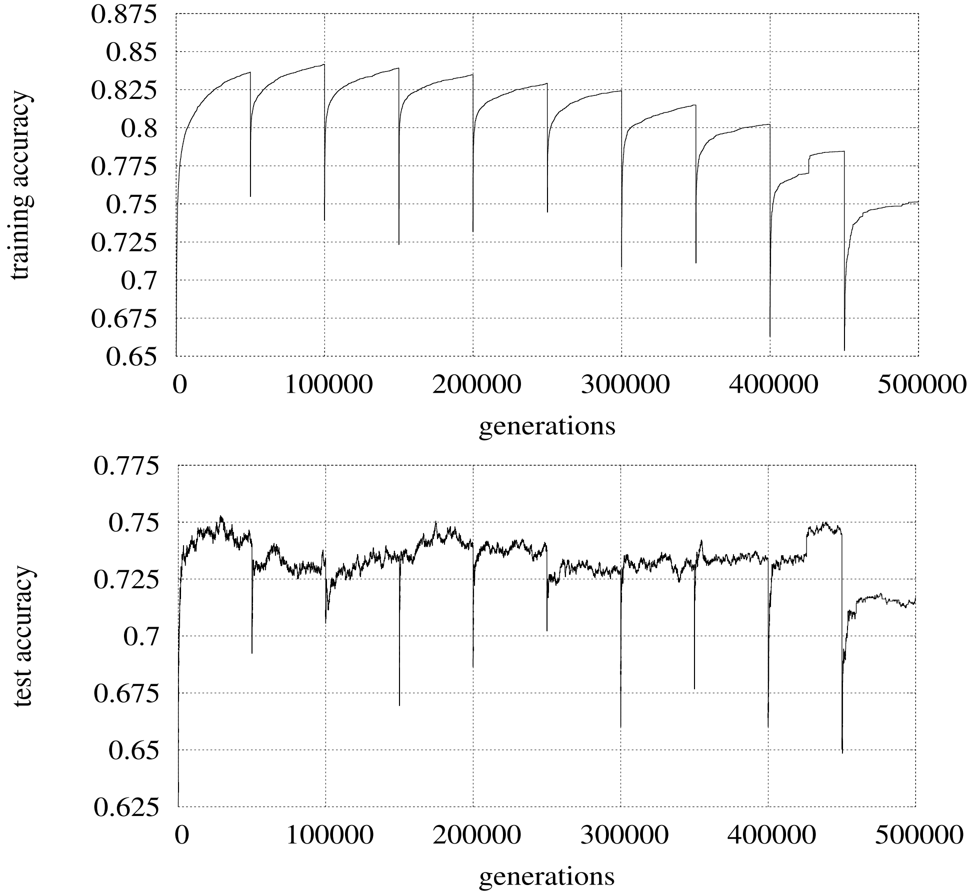
- 4 FUs per CC, number of CCs: 1→2→3→4→5→6→7→8→9→10
- add randomly initialized CCs
- test accuracy reaches high regions for small FUR configurations





Decreasing Resources: Pima Benchmark (2)

- 4 FUs per CC, number of CCs: 10→9→8→7→6→5→4→3→2→1
- remove randomly selected CCs



Fluctuating Resources: Gradual Changes

- averaged accuracy drops in % over 96 algorithm runs

		$10 \rightarrow 9 \rightarrow \cdots \rightarrow 1$		$1 \rightarrow 2 \rightarrow \cdots \rightarrow 10$	
		training	test	training	test
Pima	random	10.87	5.70	8.33	5.70
	low penalty	13.57	7.90	7.18	**5.15**
	high penalty	9.39	**4.23**	8.90	6.11
Thyroid	random	23.94	23.77	15.91	**15.74**
	low penalty	40.87	40.73	16.13	16.03
	high penalty	12.21	**12.00**	20.60	20.53

- reconfiguration effects are rather small for the Pima benchmark
- Thryoid benchmark: replicating “low penalty” CC is slightly worse than inducing random CCs

Fluctuating Resources: Radical Changes

- averaged accuracy drops in % over 32 algorithm runs

		$10 \to 4$		$4 \to 2$		$2 \to 5$		$5 \to 10$	
		training	test	training	test	training	test	training	test
Pima	random	21.90	13.76	17.75	9.91	13.46	11.86	18.27	15.42
	low penalty	21.59	10.93	19.94	11.52	8.98	**6.34**	16.52	**9.32**
	high penalty	16.65	**10.30**	10.57	**5.61**	14.66	11.91	21.71	16.35
Thyroid	random	60.00	59.37	45.96	45.55	30.29	30.27	44.13	44.00
	low penalty	54.50	54.28	54.75	54.72	30.14	29.88	34.90	**34.93**
	high penalty	34.27	**33.91**	35.71	**35.89**	18.65	**18.30**	71.84	72.16

- outlier: increasing size for small FUR configurations and the Thyroid benchmark

Results

- FUR architecture provides good accuracies for many benchmarks, even when using few CCs
  - confirmed additionally by sonar and road sign benchmarks

- reconfigurable FUR architecture is well-suited for dealing with fluctuations in available resources
  - classification accuracy is sensitive to changes in the number of CCs, but recovers quickly
  - recovery process is faster, when using more resources
  - reconfiguration schemes can reduce accuracy drops significantly
    - replicate “low penalty” CCs
    - remove “high penalty” CCs
    - reconfiguration scheme important for larger (relative) resource changes

Outline

- motivation/vision
- Δ to last status meeting
- EHW classifier adaptation to fluctuating resources
- publications, collaborations

Publications

Book chapters:

- P. Kaufmann, and M. Platzner. *Cone- and Age-based Module acquisition for Cartesian Genetic Programming*. In J. Miller, editor, *Cartesian Genetic Programming*, Springer, 2011.
- P. Kaufmann, and M. Platzner. *Embedded Cartesian Programming for Evolvable Hardware Classifiers*. In J. Miller, editor, *Cartesian Genetic Programming*, Springer, 2011.
- P. Kaufmann, C. Plessl and M. Platzner. *Evolvable Caches*. In J. Miller, editor, *Cartesian Genetic Programming*, Springer, 2011.
- P. Kaufmann, and M. Platzner. Multi-objective Intrinsic Evolution of Embedded Systems. In C. Müller- Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, Springer, 2011

Journals:

- P. Kaufmann, K. Glette, M. Platzner, J. Torresen. *Compensating Resource Fluctuations by Means of Evolvable Hardware: The Run-Time Reconfigurable Functional Unit Row Classifier Architecture,* In *International Journal of Adaptive, Resilient, and Autonomic Systems (IJARAS)*, 2011 (to appear).
- P. Kaufmann, K. Glette, T. Gruber, M. Platzner, J. Torresen, B. Sick. *Classification of Electromyographic Signals: Comparing Evolvable Hardware to Conventional Classifiers*. In *IEEE Transactions on Evolutionary Computation*, 2011 (submitted).

Papers:

- A. Boschmann, P. Kaufmann, and M. Platzner. *Accurate Gait Phase Detection using Surface Electromyographic Signals and Support Vector Machines,* IEEE Intl. Conf. Bioinformatics and Biomedical Technology (ICBBT'11)
- P. Kaufmann, K. Englehart, and M. Platzner. *Fluctuating EMG Signals: Investigating Long-term Effects of Pattern Matching Algorithms,* 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'10).
- T. Knieper, P. Kaufmann, K. Glette, M. Platzner, J. Torresen. *Coping with Resource Fluctuations: The Run-time Reconfigurable Functional Unit Row Classifier Architecture*, 9th International Conference on Evolvable Systems (ICES'10), **Awarded best student paper**
- P. Kaufmann, T. Knieper and M. Platzner. *A Novel Hybrid Evolutionary Strategy and its Periodization with Multi-objective Genetic Optimizers,* IEEE Congress on Evolutionary Computation (CEC'10).
- P. Kaufmann, C. Plessl and M. Platzner. *EvoCaches: Application-specific Adaptation of Cache Mappings,* NASA/ESA Conference on Adaptive Hardware and Systems (AHS'09).

Publications

- A. Boschmann, P. Kaufmann, M. Platzner, and M. Winkler. *Towards Multi-movement Hand Prostheses: Combining Adaptive Classification with High Precision Sockets,* Technically Assisted Rehabilitation (TAR' 09).
- K. Glette, J. Torresen, P. Kaufmann, and M. Platzner. *A Comparison of Evolvable Hardware Architectures for Classification Tasks,* 8th International Conference on Evolvable Systems (ICES'08).
- T. Schumacher, R. Meiche, P. Kaufmann, E. Lübbers, C. Plessl, and M. Platzner. *A Hardware Accelerator for k-th Nearest Neighbor Thinning*, Engineering of Reconfigurable Systems and Algorithms (ERSA'08).
- T. Knieper, B. Defo, P. Kaufmann, and M. Platzner. *On Robust Evolution of Digital Hardware*, Biologically Inspired Collaborative Computing (BICC'08).
- P. Kaufmann and M. Platzner. *Advanced Techniques for the Creation and Propagation of Modules in Cartesian Genetic Programming,* Genetic and Evolutionary Computation Conference (GECCO'08).
- K. Glette, T. Gruber, P. Kaufmann, J. Torresen, B. Sick, and M. Platzner. *Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control,* NASA/ESA Conference on Adaptive Hardware and Systems (AHS'08), **Awarded best paper**
- P. Kaufmann and M. Platzner. *MOVES: A Modular Framework for Hardware Evolution,* NASA/ESA Conference on Adaptive Hardware and Systems (AHS'07), **Awarded best paper**
- P. Kaufmann and M. Platzner. *Toward Self-adaptive Embedded Systems: Multi-objective Hardware Evolution,* Architecture of Computing Systems (ARCS'07).
- P. Kaufmann and M. Platzner. *Multi-objective Intrinsic Hardware Evolution,* MAPLD'06 International Conference.

Collaborations

- reconfigurable architectures
  - Jim Torresen & Kyrre Glette, University of Oslo
- pattern matching algorithms
  - Bernhard Sick & Thiemo Gruber, University of Kassel
- EMG signal analysis
  - Kevin Englehart, University of New Brunswick

Thank you for your attention!