DESIGN, AUTOMATION & TEST IN EUROPE

27 – 31 March, 2017 · STCC · Lausanne · Switzerland

The European Event for Electronic System Design & Test

Multi-armed Bandits for Efficient Lifetime Estimation in MPSoC Design

Calvin Ma, Aditya Mahajan, and Brett H. Meyer

Department of Electrical and Computer Engineering

McGill University

RELIABLE SILICON SYSTEMS LAB

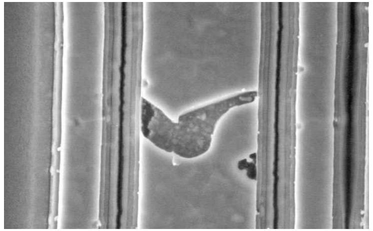
Design Differentiation in DSE

- **Design space exploration (DSE) is often used for MPSoCs**
- **Design spaces are large (on the orders of billions of alternatives)**
- **Design evaluation can be complex (requiring multiple metrics)**
- ***Exhaustive search is usually intractable***
- **Goals of DSE:**
  1. **Differentiate poor solutions from good ones**
  2. **Identify the Pareto-optimal set**
  3. **Do so quickly and efficiently**

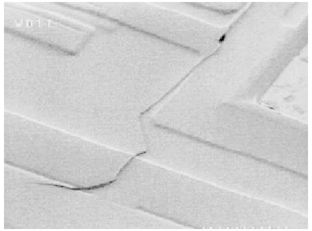
System Lifetime Optimization for MPSoC

- **Semiconductor scaling has reduced integrated circuit lifetime**

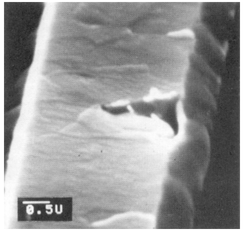
Electromigration



Thermal Cycling



Stress migration



[Source: JEDEC]

- **Many strategies have been developed to address failure:**
  - **Redundancy (at different granularities) or slack allocation**
  - **Thermal management and task migration**
- **System-level optimization seeks to maximize *mean time to failure* under other constraints (e.g., *performance*, *power*, *cost*)**

Evaluating System Lifetime

- **Failure mechanisms are modeled mathematically**
  - **Historically, with the exponential distribution: *easy to work with***
  - **Recently, with log-normal and Weibull distributions: *more accurate***
- ***There is no straightforward closed-form solution for systems of log-normal and Weibull distributions***
- **Therefore, Monte Carlo Simulation (MCS)!**
  - **Use failure distributions to generate a random system instance (*sample*)**
  - **Determine when that instance fails through simulation**
  - **Capture statistics, and repeat!**

Multi-armed Bandits for Smarter Estimation

- **Monte Carlo Simulation is *needlessly* computationally expensive**
  - **Samples are distributed evenly to *estimate lifetime***
  - ***Poor* designs are sampled as much as *good* designs**
- ***Multi-armed Bandits (MAB) are smarter***
  - **Samples are incrementally distributed in order to *differentiate* systems**
  - ***E.g.*, to find the *best*, the *best k*, etc.**
- ***Hypothesis: MAB can achieve DSE goals with fewer evaluations than MCS by differentiating systems, not estimating lifetime***

Outline

- **Multi-armed Bandits**
  - **Successive Accept Reject**
  - **Gap-based Exploration with Variance**
- **Lifetime Differentiation Experiments and Results**
- **Conclusions and Future Work**

Multi-armed Bandits Algorithms

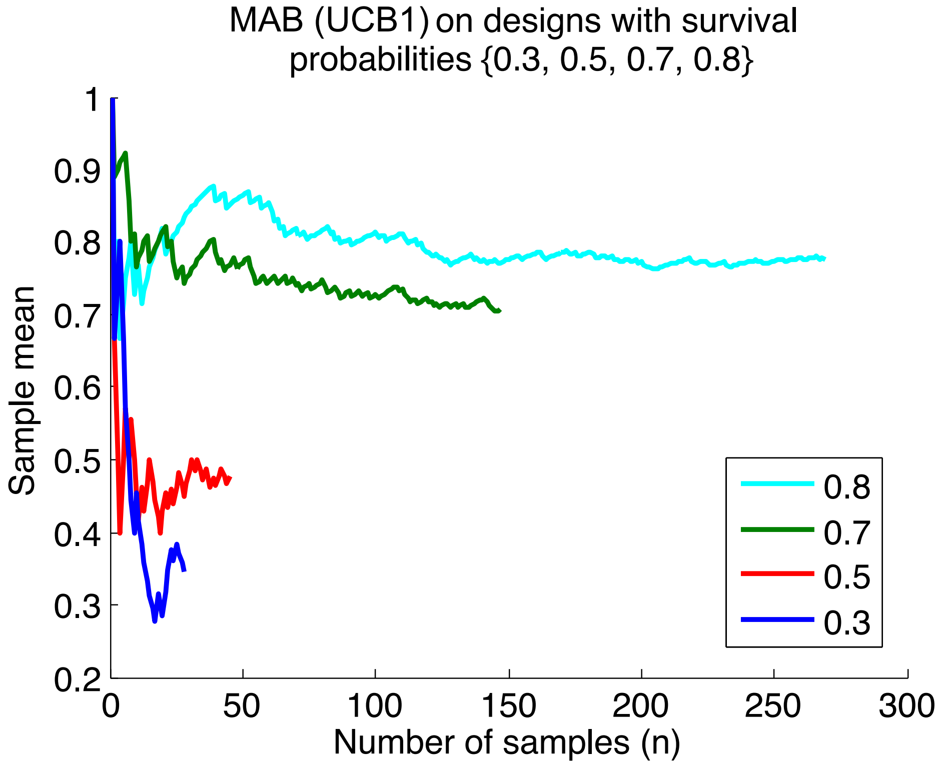
- Which slot machine is the best?
- *Monte Carlo Simulation* is systematic
  - Try every slot machine equally
  - In the end, compare average payout
- *Multi-armed Bandits algorithms* gamble intelligently
  - Try every slot machine, but stay away from bad ones
  - Do so by managing expected payout from next trial

Simple MAB Example

- **Assume Bernoulli-distributed systems with different *p***
- **UCB1 *plays* (samples) the *arm* (system) that maximizes**

$$\bar{x}_i + \sqrt{\frac{2 \ln n}{n_i}}$$

- **Explore, but favor better arms**
- **Eventually, the *best* system is always played**



MAB for Lifetime Differentiation

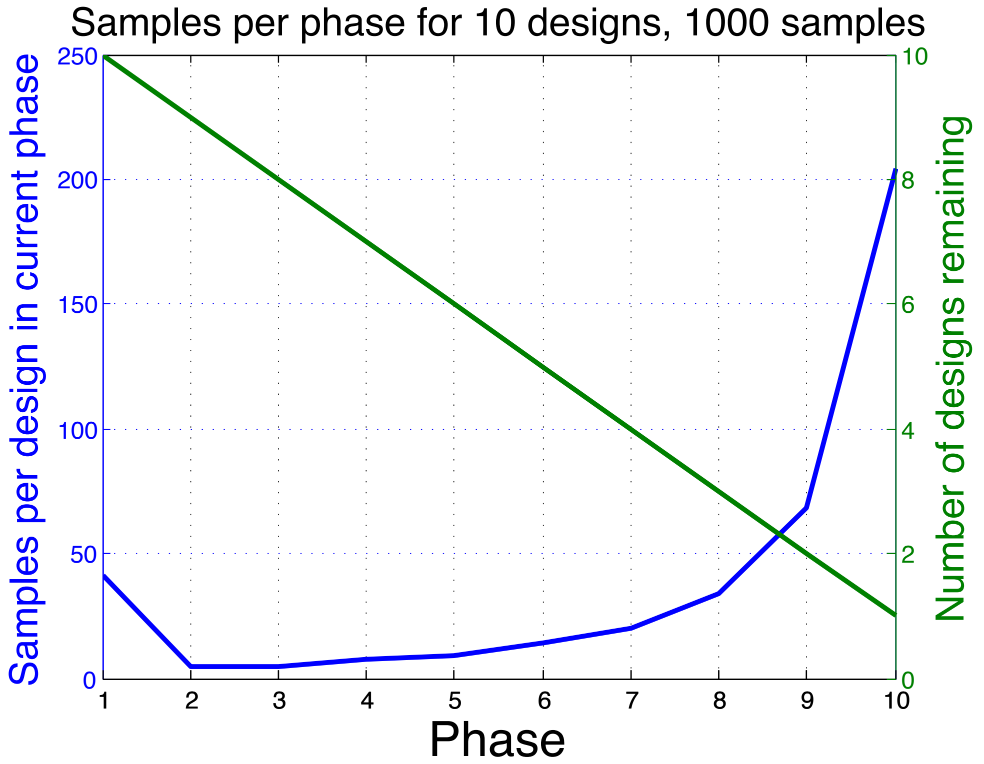
- **Conventional MAB formulations assume that**
  - **The *player* never stops playing**
  - **The *reward* is incrementally obtained after each arm pull**
  - **A single best *arm* is identified**
- **For DSE, we relax these assumptions**
  - **Assume a fixed sample budget used to explore designs**
  - **The reward is associated with the final choice**
  - **Find the best *m* arms**
- **Two MAB algorithms can be applied in this context**

Successive Accept Reject (SAR)

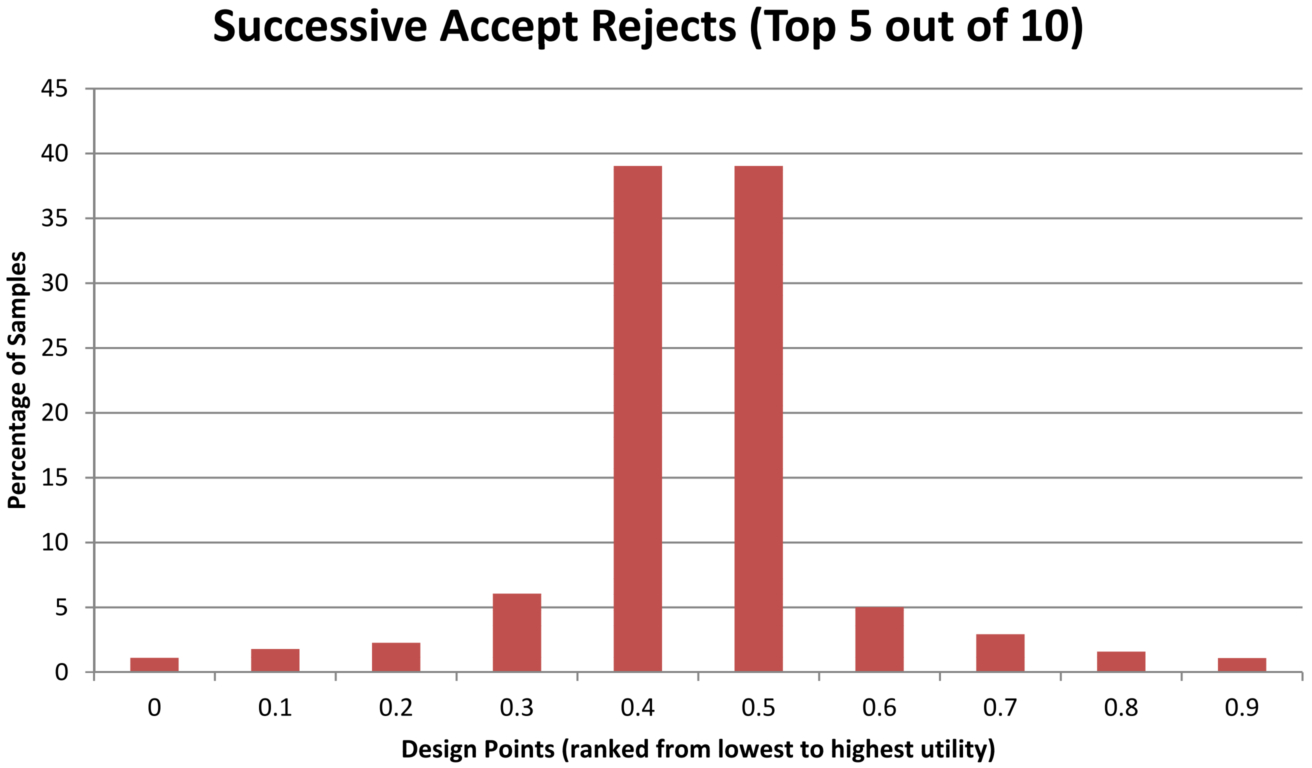
- SAR divides the sample budget into *n* phases to compare *n* arms
- Each phase, the allocated budget is divided across active arms
- After sampling, calculate the distance from boundary between the *m* good designs and *n* – *m* bad ones
  - Top *m* designs: $\Delta_i = \hat{\mu}_i - \hat{\mu}_{i_*}$
  - Bottom *n* – *m* designs: $\Delta_i = \hat{\mu}_{i^*} - \hat{\mu}_i$
- Remove from consideration the design with the biggest gap

Successive Accept Reject Example

- **Sample all designs initially**
- **Samples per design grows as designs are removed**
- **Many samples used to differentiate *m*th and *m+1*th designs**



Successive Accept Reject Example



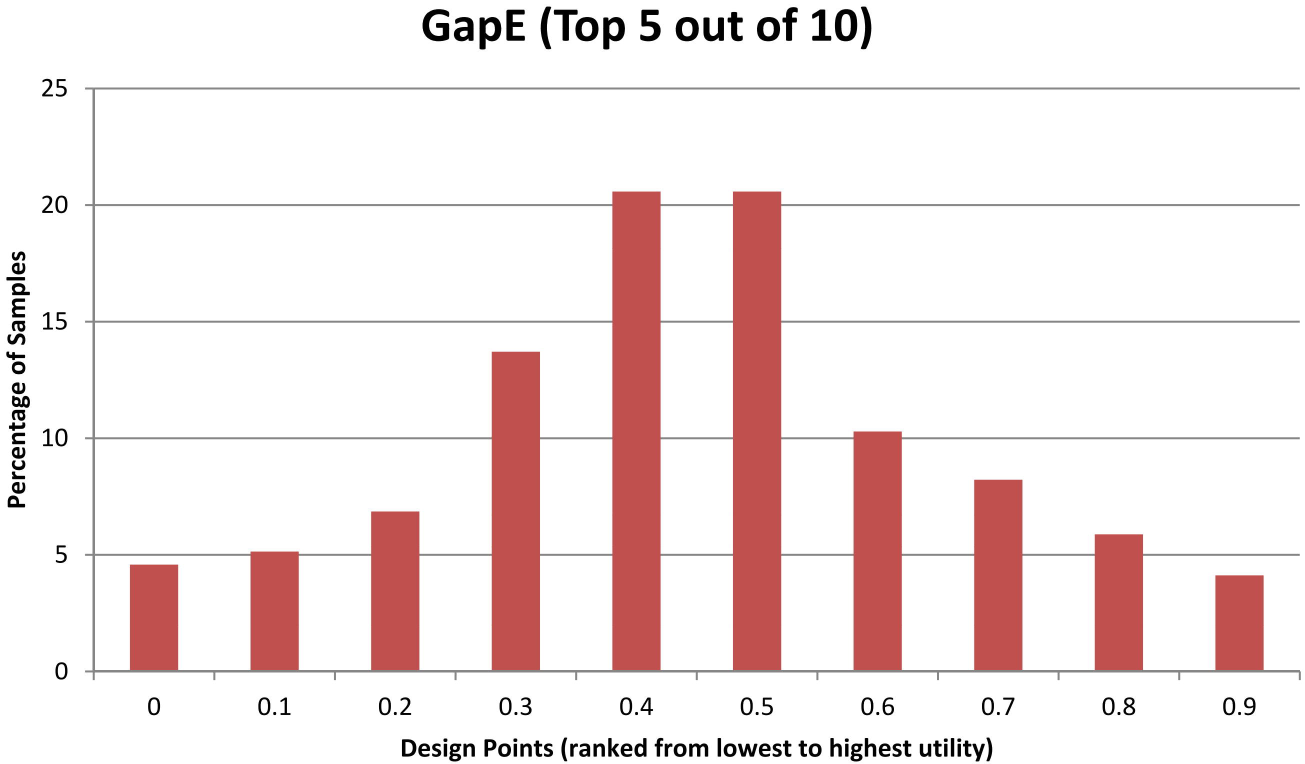
Gap-based Exploration with Variance (GapE-V)

- GapE-V never removes a design from consideration
- Instead, pick the design that minimizes the empirical gap with the boundary, plus an exploration factor

$$I_t = -\Delta_i + \sqrt{\frac{2a\hat{\sigma}_i}{T_i}} + \frac{7ab}{3(T_i - 1)}$$

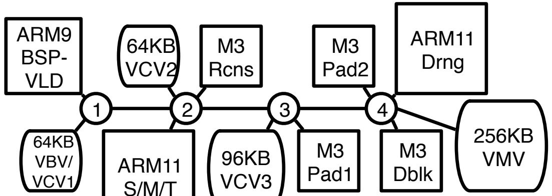
- Effort is focused near the boundary
- High variance, or a limited number of samples, increase likelihood a design is sampled

GapE-V Example

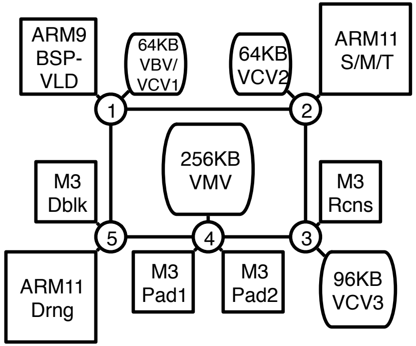


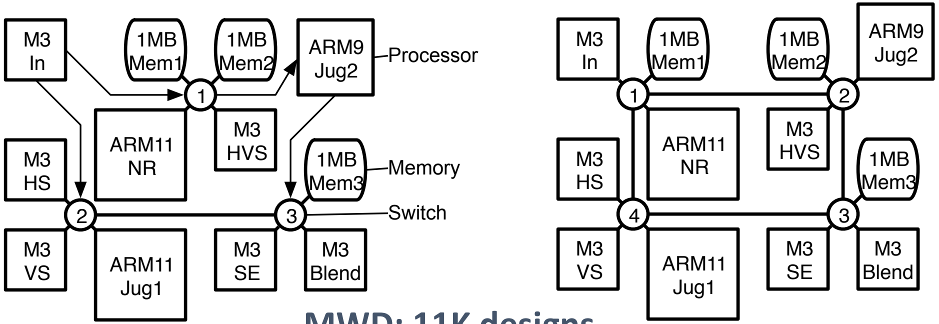
Experimental Setup

- NoC-based MPSoC lifetime optimization with *slack* allocation
  - *Slack* is spare compute and storage capacity
  - Add slack to components s.t. remapping mitigates one or more failures
- Two applications, two architectures each
- Component library of processors, SRAMs



MPEG-4: 140K designs





MWD: 11K designs

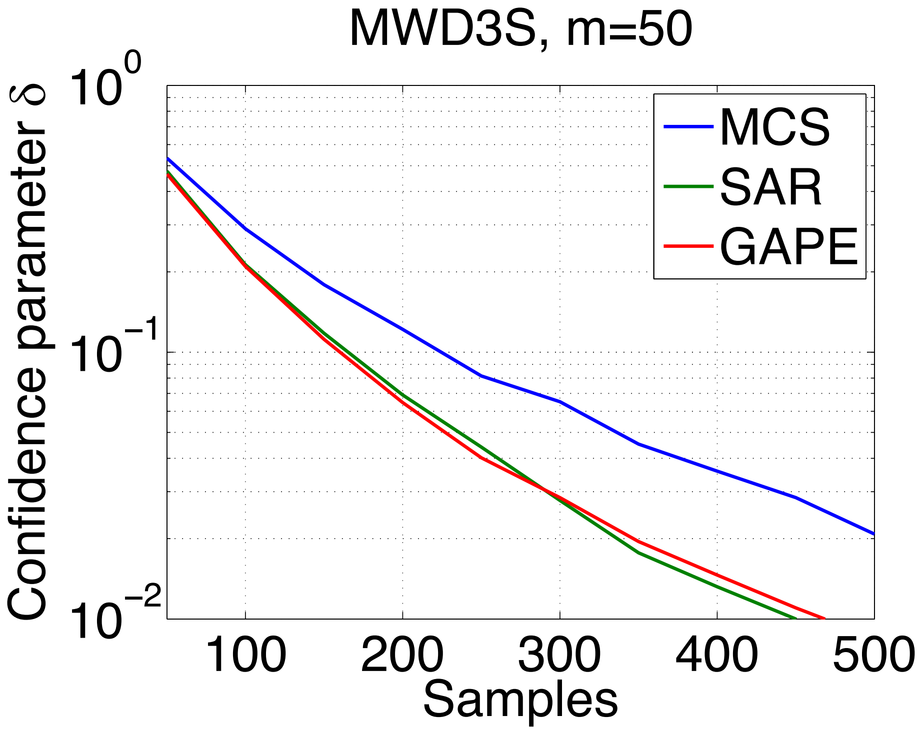
Evaluating the Chosen *m*

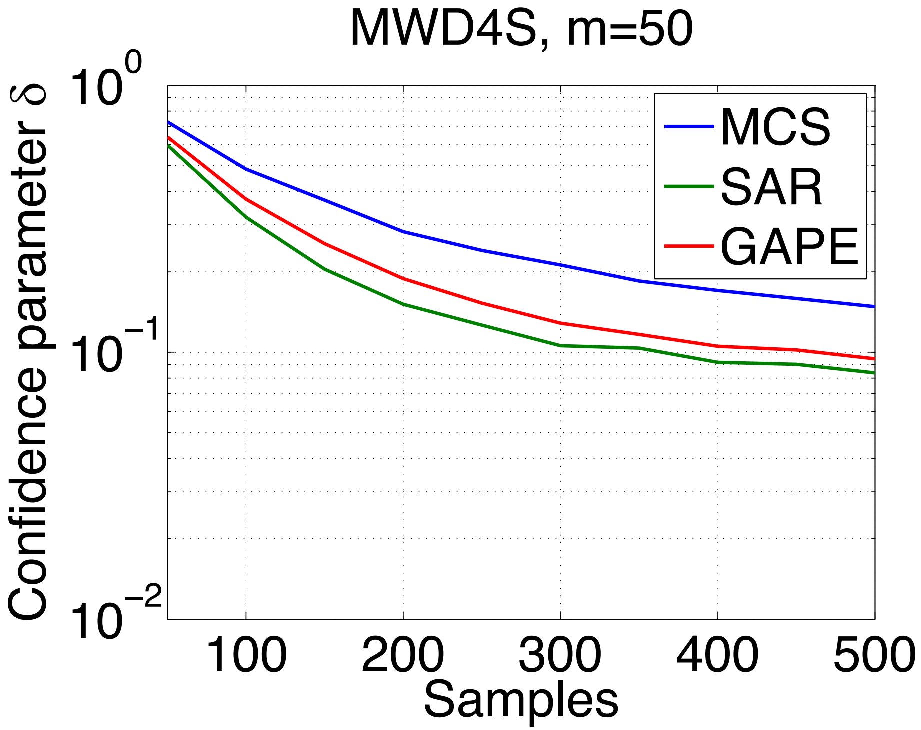
- **We compare SAR, GapE-V, and MCS**
  - **Optimal set determined with MCS using 1M samples per design**
- **How likely is it that an approach picks the wrong set?**
  - **Compare the aggregate MTTF using policy J and the optimal set**

$$Pr\left[\sum_{i=1}^{m} \mu_i^* - \mathbf{E}\mu_{J(i)} > \epsilon\right] \leq \delta$$

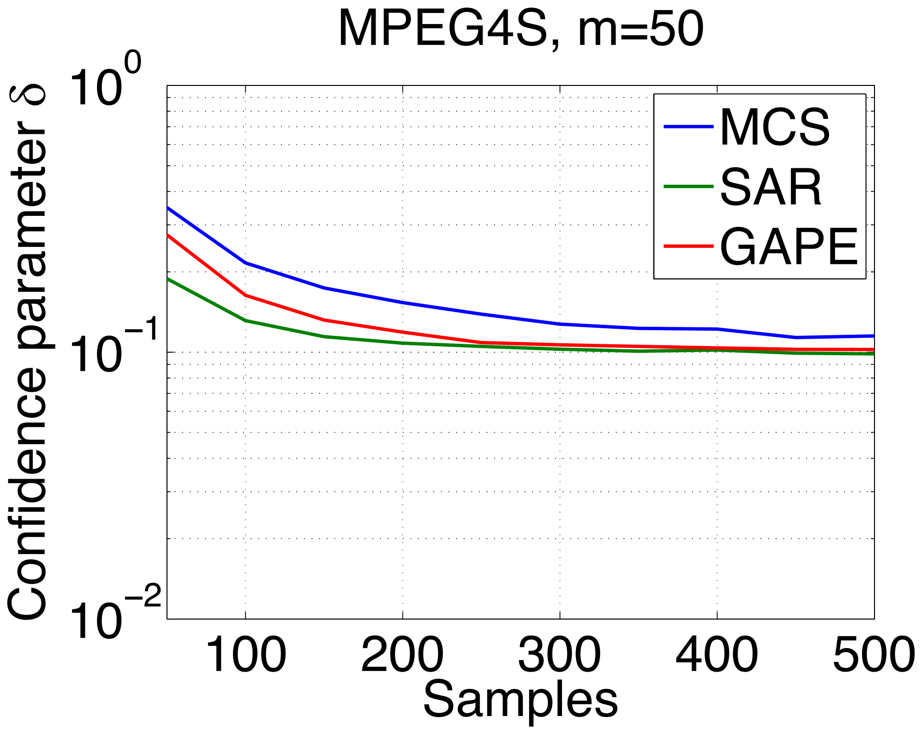
- δ **is the probability of *identification error*, the chance a subset of *m* differs significantly from the optimal set**

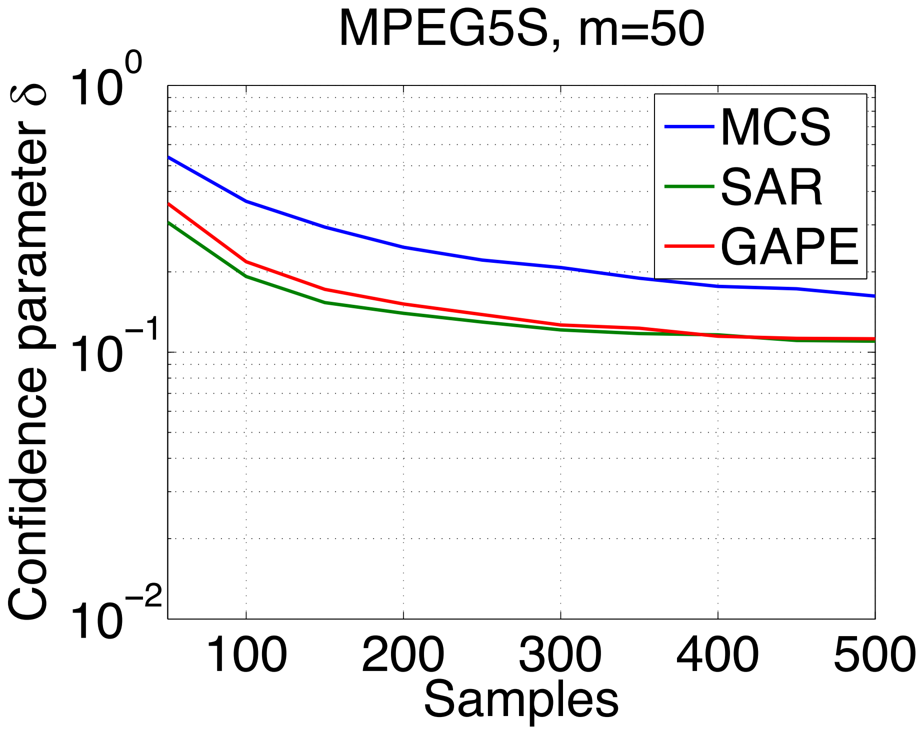
Picking the Top 50, MWD





Picking the Top 50, MPEG-4





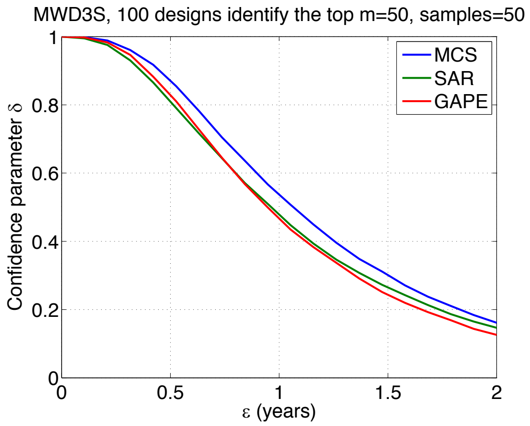
Comparison with MCS after 500 samples

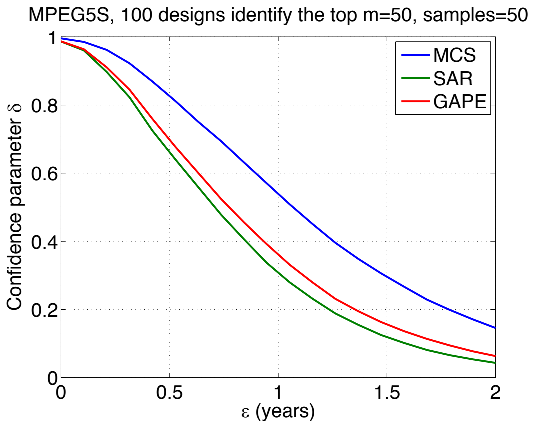
	m=20			m=30		
Benchmark	δ	SAR	GapE-V	δ	SAR	GapE-V
MWD3S	0.002	1.92x	1.72x	0.003	1.72x	1.71x
MWD4S	0.071	3.33x	2.13x	0.112	2.96x	2.07x
MPEG4S	0.120	3.57x	2.70x	0.101	3.52x	2.48x
MPEG5S	0.052	5.26x	3.57x	0.083	4.07x	3.05x

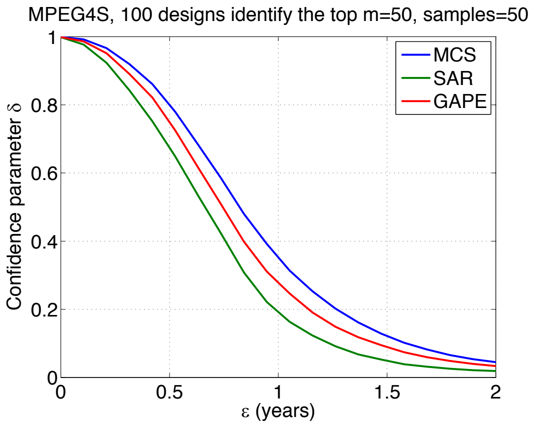
	m=40			m=50		
Benchmark	δ	SAR	GapE-V	δ	SAR	GapE-V
MWD3S	0.009	1.79x	1.67x	0.021	1.49x	1.45x
MWD4S	0.180	2.54x	2.01x	0.148	2.44x	1.92x
MPEG4S	0.202	3.60x	2.43x	0.115	3.33x	2.27x
MPEG5S	0.292	3.70x	3.07x	0.162	3.57x	2.86x

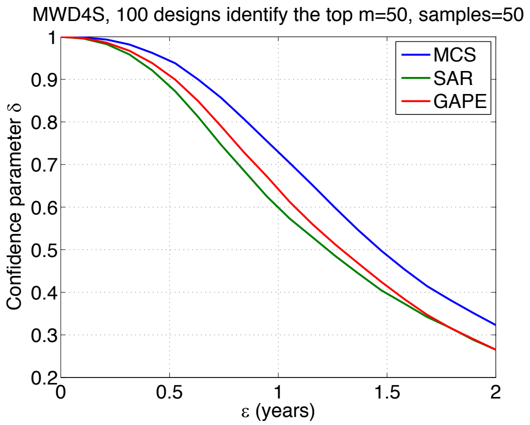
Does Error Tolerance Matter?

- **No; MAB wins!**









What About Complexity?

- **Complexity is a function of *sampling* and *selection***
- ***Sampling* time *ND* x T_{sample} is fixed across approaches**
- **MCS performs no selection: all designs are sampled equally**
- **SAR (GapE-V) additional sorts the design list *D* (*ND*) times**

Algorithm	Run Time (Upper Bound)
MCS	$ND \times T_{sample}$
SAR	$ND \times T_{sample} + D \times T_{sort}(D)$
GapE-V	$ND \times T_{sample} + ND \times T_{sort}(D)$

MAB When Sampling is Expensive

Algorithm	Number of Designs			
	50	100	200	400
MCS	4.41s	8.52s	16.86s	34.54s
SAR	4.48s	10.41s	27.22s	95.26s
GapE	5.33s	11.46s	34.31s	108.64s

- **500 samples per design, Intel E5-2670, 96GB RAM averaged over 10 trials, or <1 ms per trial**
- **When sampling complexity is *low*, MAB loses as the population grows (*sorting dominates*)**

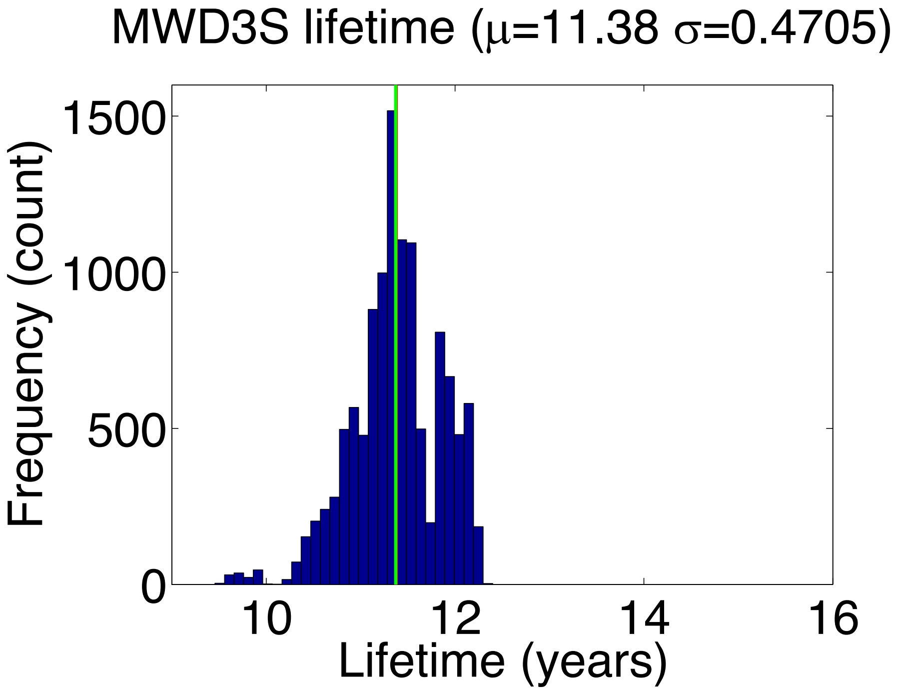
Conclusions and Future Work

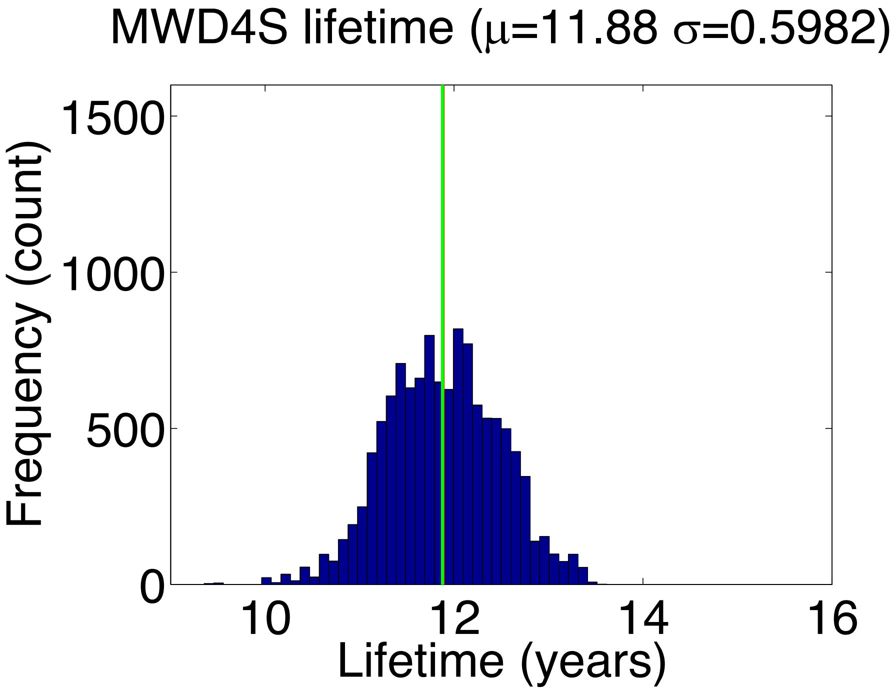
- **The objective of DSE is to *differentiate* designs**
- **MCS is *poorly suited* for this: why evaluate bad designs?**
- **MAB spends samples to *efficiently separate* metric estimates**
- **Estimating system lifetime, MAB uses 33-81% fewer samples**
- ***Next step*: apply in population-based design space exploration**

Thank you!

Questions?

Lifetime Distributions, MWD





Lifetime Distributions, MPEG-4