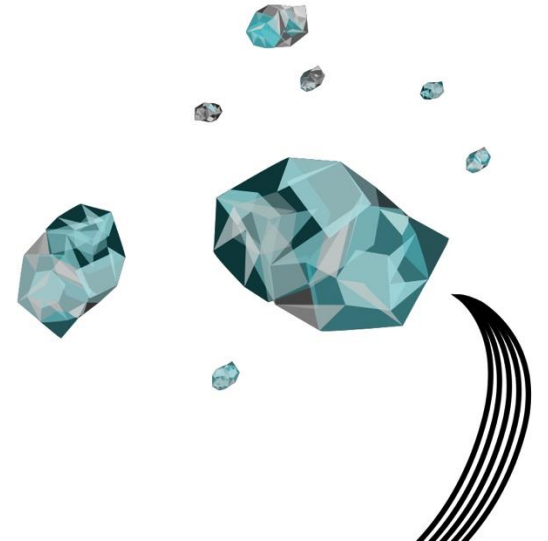
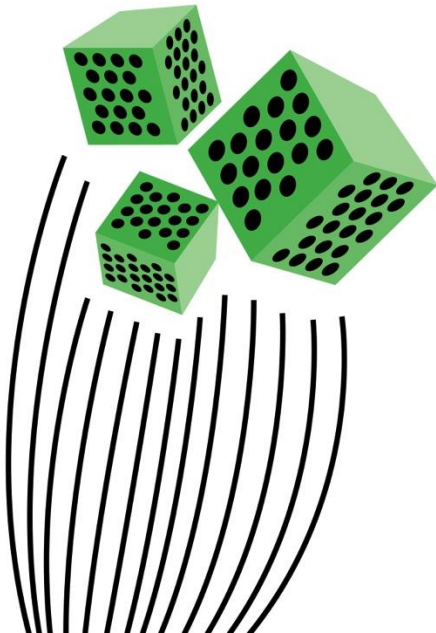


LOW VOLTAGE AND APPROXIMATE COMPUTING

A.B.J. Kokkeler

G.A. Gillani

L. Oudshoorn



OUTLINE

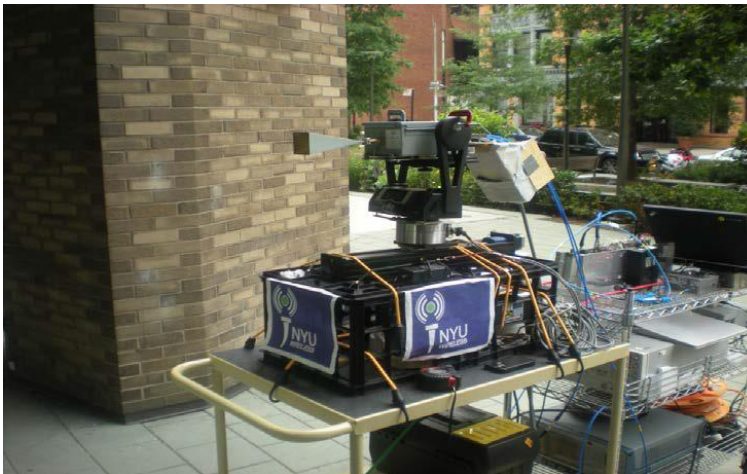
- Introduction
- Low Voltage Computing
 - Deterministic
 - Probabilistic
- Approximate Computing
 - Introduction
 - Case Study 1: Stefcad
 - Case Study 2: Digitally Assisted Beamforming
- Conclusion

INTRODUCTION

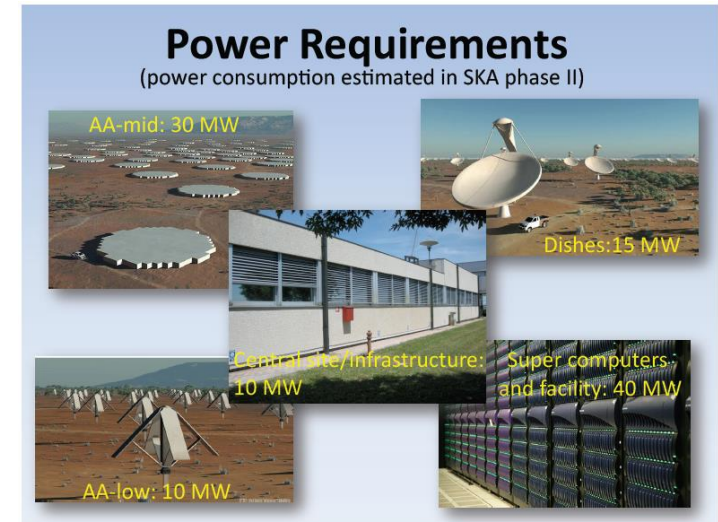
- Driving forces behind desire for energy efficient computing
 - More bandwidth
 - More processing

INTRODUCTION

- Driving forces
 - More bandwidth
 - More processing
 - **Restricted footprint and power budget**



mm wave RX hardware (Rappaport 2014)



105 MW (250.000 households in The Netherlands)

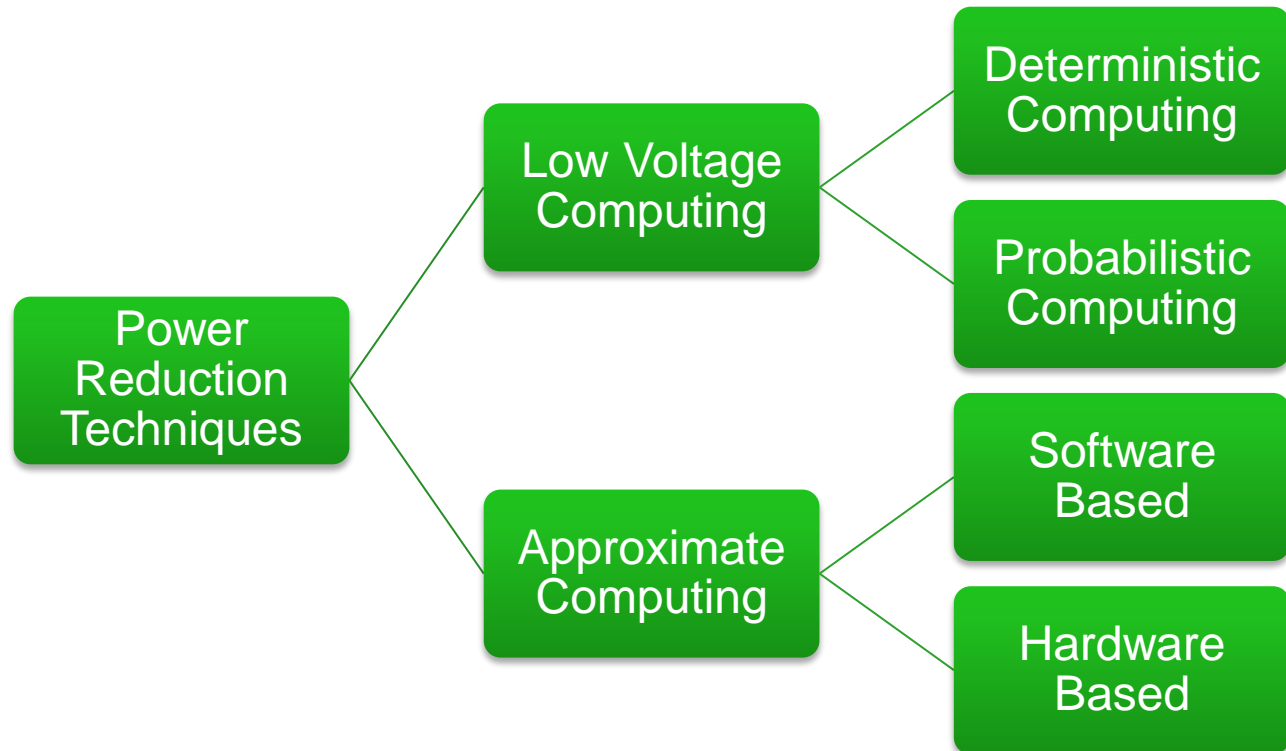
INTRODUCTION

- Driving forces
 - More bandwidth
 - More processing
 - Restricted footprint and power budget
- Context
 - ADC bottleneck
 - Moore's law
 - Providing more transistors per square mm
 - No frequency increase
 - No gain in power efficiency

INTRODUCTION

- Solutions
 - Heterogeneous architectures with specialized cores
 - Different cores exploit different techniques
 - Digital becomes analog (low voltage computing)
 - Signal processing tuned to precision that is actually required (approximate computing)

INTRODUCTION



OUTLINE

- Introduction
- Low Voltage Computing
 - Deterministic
 - Probabilistic
- Approximate Computing
 - Introduction
 - Case Study 1: Stefcad
 - Case Study 2: Digitally Assisted Beamforming
- Conclusion

LOW VOLTAGE COMPUTING

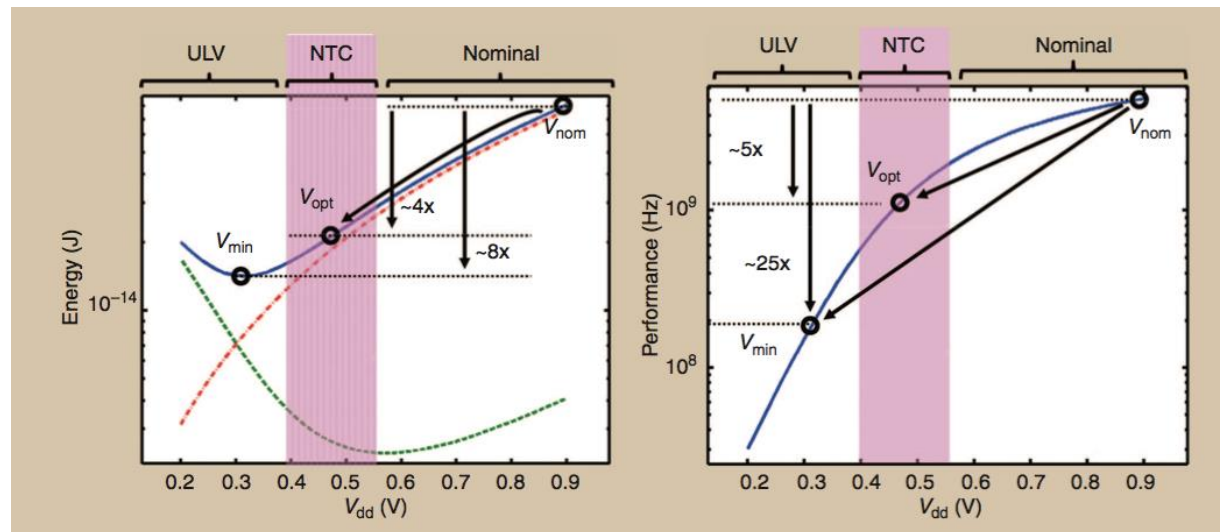
DETERMINISTIC

- Motivation:
 - $E_{dynamic} \propto V_{dd}^2$
 - Lowering voltage can increase energy efficiency even if additional hardware is required to maintain the performance
- Types:
 - Nominal / Super Threshold Voltage 0.55 to 1V: High - performance and Energy
 - Near Threshold Voltage (NTV) 0.4 to 0.55V: Mod - performance and Energy
 - Ultra Low Voltage (ULV) 0 to 0.4V: Low - performance and Energy
 - DVFS: Dynamic voltage and frequency scaling aims at run time adjustment
 - Heterogeneous Architectures: Multiple performance-energy tradeoffs because of multiple, different cores [2]

LOW VOLTAGE COMPUTING

DETERMINISTIC

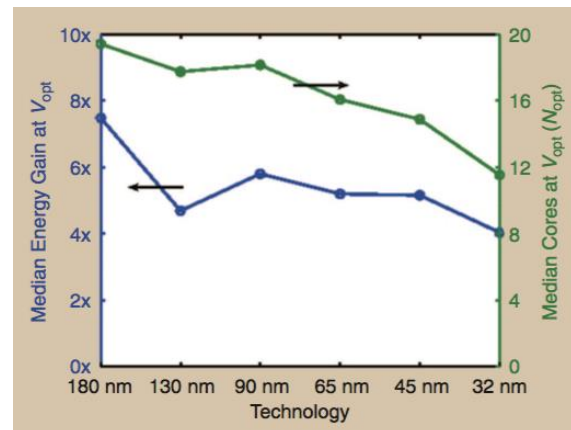
- Energy and Performance Trade-off for 32nm tech [3]



LOW VOLTAGE COMPUTING

DETERMINISTIC

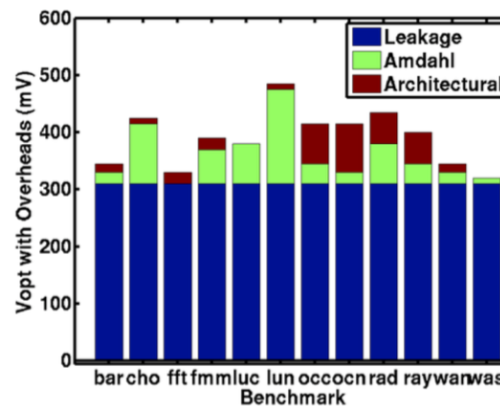
- Energy Benefits:
 - V_{opt} design showing number of cores parallelized across SPLASH-2 benchmarks has increased energy efficiency 4 times for 32nm tech [3]



LOW VOLTAGE COMPUTING

DETERMINISTIC

- Design Limiters [1]
 - Leakage or static power – Tech specific
 - Amdahl's overhead – App specific
 - Architectural overhead – App specific



LOW VOLTAGE COMPUTING

PROBABILISTIC

- Adaptive Voltage Overscaling:
 - Lowers voltage while leaving frequency unchanged
 - Simulations for texture decompression algorithm ftc1 show energy savings of 25% to 30% (1.2V to 1.8V operation) [4]
- Probabilistic Computing:
 - Reducing V_{dd} below safe region w.r.t. frequency of operation => probabilistic behavior of circuit
 - Safe region is also dependent on gate width and noise levels

LOW VOLTAGE COMPUTING

PROBABILISTIC

Probabilistic CMOS

- Operate PCMOS at various voltage levels below safe region
- This provides Energy- probability of correctness (E-p) tradeoff [6]

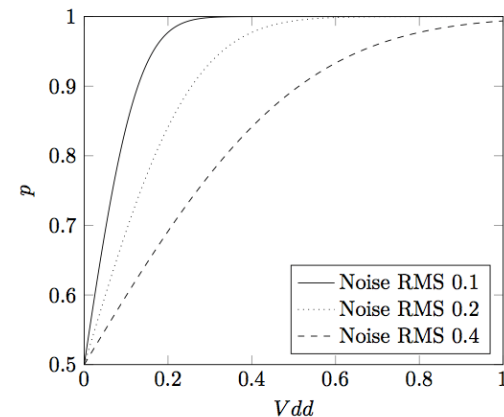
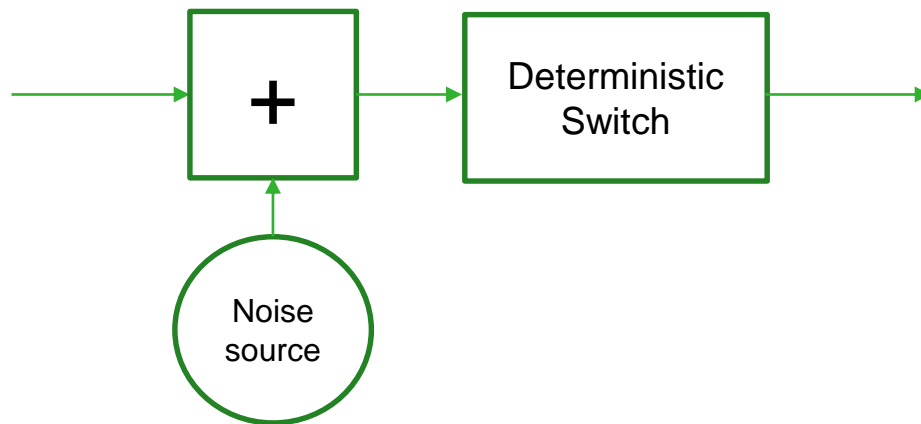


LOW VOLTAGE COMPUTING

PROBABILISTIC

Probabilistic CMOS

- Operate PCMOS at various voltage levels below safe region
- This provides Energy- probability of correctness (E-p) tradeoff [6]



LOW VOLTAGE COMPUTING

PROBABILISTIC

- Probabilistic switch, for instance an inverter, modeled with a noise coupled output as [7]

$$p = P\left(X \leq \frac{V_{dd}}{2}\right) = \frac{1}{2} + \frac{1}{2} \operatorname{erf}\left(\frac{V_{dd}}{2\sqrt{2}\sigma}\right)$$

Where σ is the RMS noise

LOW VOLTAGE COMPUTING

PROBABILISTIC

Error propagation in PCMOS

- It is also important to calculate the error propagation within the probabilistic system to find the total error, for example in ripple carry adder
- M. Lau [8] calculated the propagation error within a 4-bit ripple carry adder for each sum and carry output

$$P(s'_{i+1} = s_{i+1}) = \frac{1}{2} + \left(p_{i+1}^s - \frac{1}{2}\right) \times \left[\prod_{j=1}^i \left(p_j^c - \frac{1}{2}\right) + \sum_{k=1}^i \prod_{l=k}^i \left(p_l^c - \frac{1}{2}\right) \right]$$

Where p^c and p^s are the probabilities of correct outputs for carry and sum respectively

- Simulations show the impact of delay propagation in addition to error propagation as modeled above

LOW VOLTAGE COMPUTING

PROBABILISTIC

Impact of delay propagation at NTV

- Inverter simulations (umc65 library in Cadence IC) at various noise scales, gate widths and frequencies
- Approach: simulate 65nm technology with increased intrinsic noise due to channel resistance to represent the much smaller future transistors

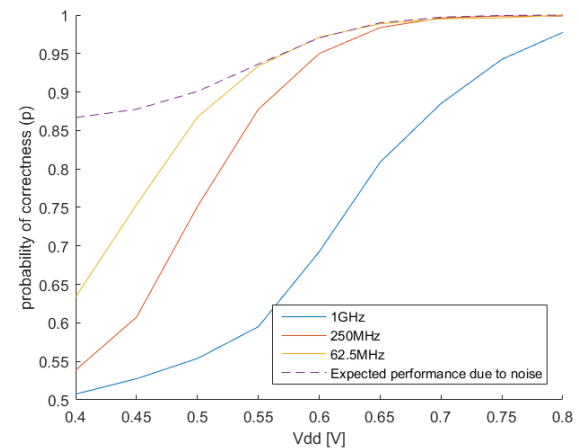
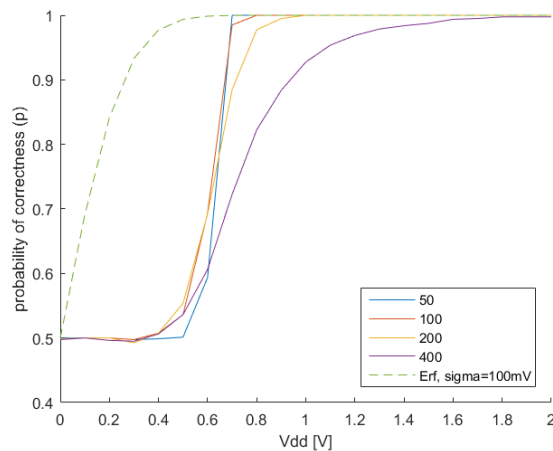
Parameters	Value
MOS type	Umc65ll N/P_12_llrvt
NMOS gate length	60nm
NMOS gate width	80nm
PMOS gate length	60nm
PMOS gate width	160nm
V _{dd}	Range: 0-2V, 10mV steps
C _{out}	10fF
Temperature	27°C
Noise amplification	50x/100x/200x/400x

LOW VOLTAGE COMPUTING

PROBABILISTIC

Impact of delay propagation

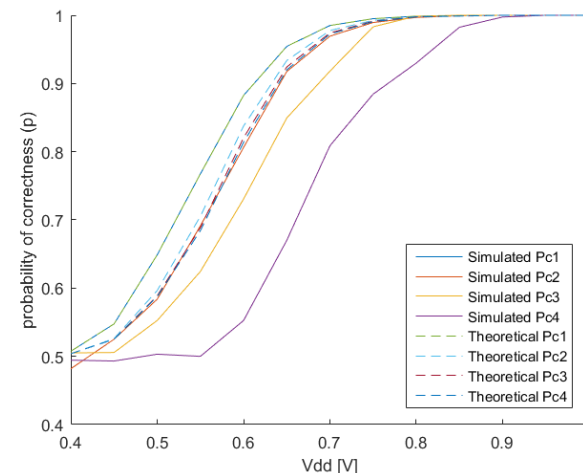
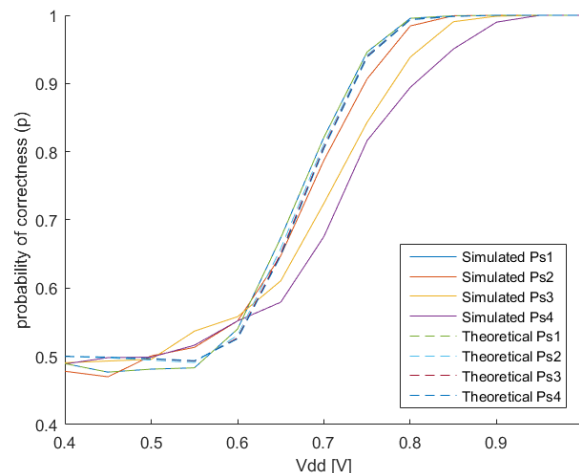
- Abrupt decrease in p can be observed at low voltages for various noise levels and frequencies of operation



LOW VOLTAGE COMPUTING

PROBABILISTIC

- To show delay propagation in PCMOS systems, simulation of 4-bit ripple carry adder in Cadence IC
- The theoretical curves are based on the assumption that the propagation of error is only due to probability of correctness metric
- However, the delayed correct outputs of stage i can make the probability of correctness worse for stage $i+1$



LOW VOLTAGE COMPUTING

PROBABILISTIC

Conclusions

- Simulations of an inverter and a 4-bit ripple carry adder in Cadence show importance of including delay in modeling PCMOS
- Impact of delay propagation in a digital system composed of probabilistic building blocks is investigated, which provides:
 - A clear insight of timing delay
 - Affecting the higher significant computational bits more than its lower significant counterparts
 - Hence contributing considerably to the total error

LOW VOLTAGE COMPUTING

PROBABILISTIC

References

- [1] Nathaniel Pinckney et al “Assessing the Performance Limits of Parallelized Near-Threshold Computing” DAC 2012
- [2] http://www.eetimes.com/document.asp?doc_id=1279167; (visited on 08-02-2016)
- [3] Nathaniel Pinckney et al “Low power near threshold design techniques to improve energy efficiency” IEEE SSC Magazine 2015
- [4] Philipp Klaus et al “Adaptive voltage over-scaling for resilient applications” 2011
- [5] Vinay K. Chippa et al “Analysis and Characterization of Inherent Application Resilience for Approximate Computing” DAC 2013
- [6] KV Palem. “Energy aware computing through randomized switching”. Technical Report GIT-CC-03-16, 2003.
- [7] Cheemalavagu, Suresh, et al. “A probabilistic CMOS switch and its realization by exploiting noise.” IFIP International Conference on VLSI. 2005.
- [8] Lau, Mark SK, et al. “Modeling of probabilistic ripple-carry adders.” Electronic Design, Test and Application, 2010. DELTA'10. Fifth IEEE International Symposium on. IEEE, 2010.

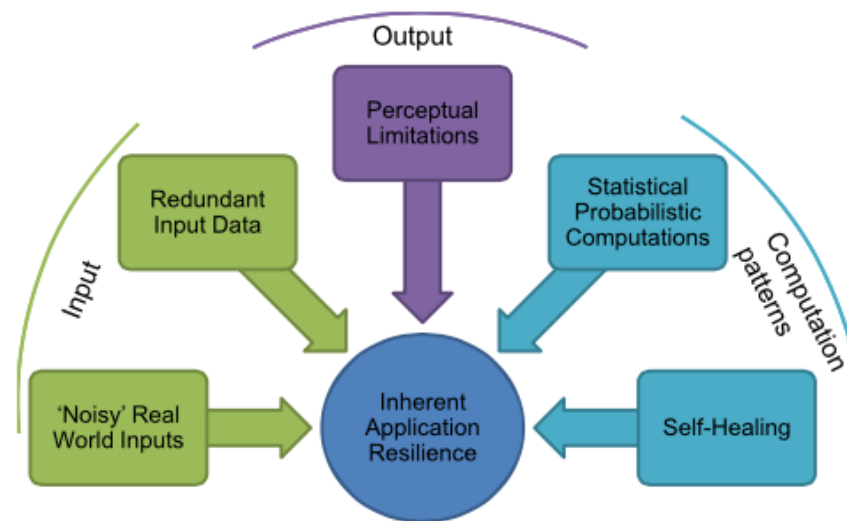
OUTLINE

- Introduction
- Low Voltage Computing
 - Deterministic
 - Probabilistic
- **Approximate Computing**
 - Introduction
 - Case Study 1: Stefcad
 - Case Study 2: Digitally Assisted Beamforming
- Conclusion

APPROXIMATE COMPUTING

INTRODUCTION

Why Resilience is present in some applications?



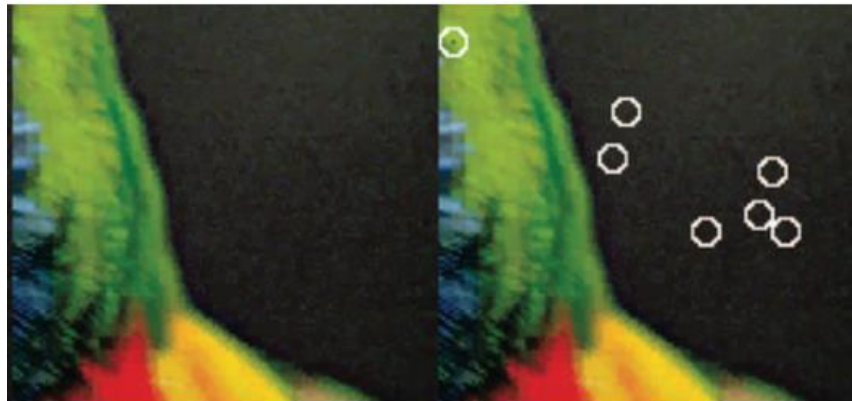
Vinay K. Chippa et al "Analysis and Characterization of Inherent Application Resilience for Approximate Computing" DAC 2013

APPROXIMATE COMPUTING

INTRODUCTION

Approximate computing resilient applications example

- Texture decompression: 25% to 30% energy savings for various images [4]



APPROXIMATE COMPUTING

INTRODUCTION

Approximate computing

- Inexact Computing or Best Effort Computing
- Compute with bare minimum accuracy to save costs
- Energy efficiency increases beyond V_{opt} operation
- Quality vs Cost trade-off
- For error resilient applications like multimedia digital signal processing, search engines and scientific computing

APPROXIMATE COMPUTING

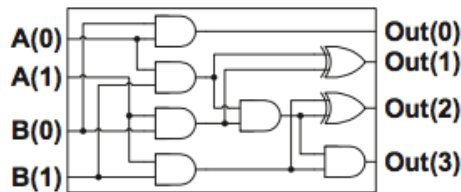
INTRODUCTION

- Techniques:
 - Data size or width reduction
 - Loop perforation
 - Approximate memoization
 - Logic simplification

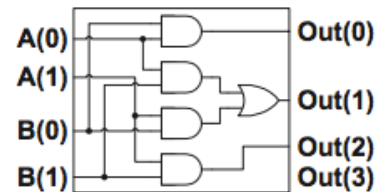
APPROXIMATE COMPUTING

INTRODUCTION

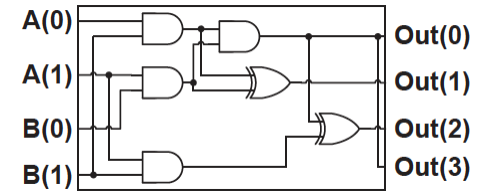
An example: Logic simplification



Accu-Mul



Ax-Mul₁



Ax-Mul₂

Design Type	Area	Latency	Power	Error Rate	Error Magnitude
Accu-Mul	6.88	0.1	543	0	0
Ax-Mul ₁	3.70	0.06	363	1/16	2
Ax-Mul ₂	4.94	0.1	262	3/16	1

Rehman, S, et al. (2016). Architectural-space exploration of approximate multipliers. Proceedings of the 35th ICCAD, ACM

- Will be discussed on Thursday

APPROXIMATE COMPUTING

INTRODUCTION

- Architectures:
 - Approximate architectures
 - Accuracy configurable architectures
 - Heterogeneous architectures

APPROXIMATE COMPUTING

INTRODUCTION

Error resilience analysis, why bother?

- Not every application is error-resilient
- Within a resilient application, not every kernel is error resilient
- Approximation (ax) techniques offer different error distributions
- Higher design space exploration requirements (ax alternatives)

- It is important to analyze applications in order to,
 - Address the above problems
 - Simulate the effects of approximations by applying corresponding models

APPROXIMATE COMPUTING

INTRODUCTION

Available tools for error resilience analysis

- Quality of Service (QoS) Profiling¹: Loop perforation
- iACT²: precision scaling, ax memoization, noisy hardware effects
- Automatic Sensitivity Analysis (ASAC)³: Perturbation of program data
- ARC Framework: Characterizes the statistical distribution of error-tolerance based on,
 - Error Mean (EM)
 - Error Predictability (EP)
 - Error Rate (ER)

1. Sasa Misailovic, et al. Quality of service profiling. ICSE 2010

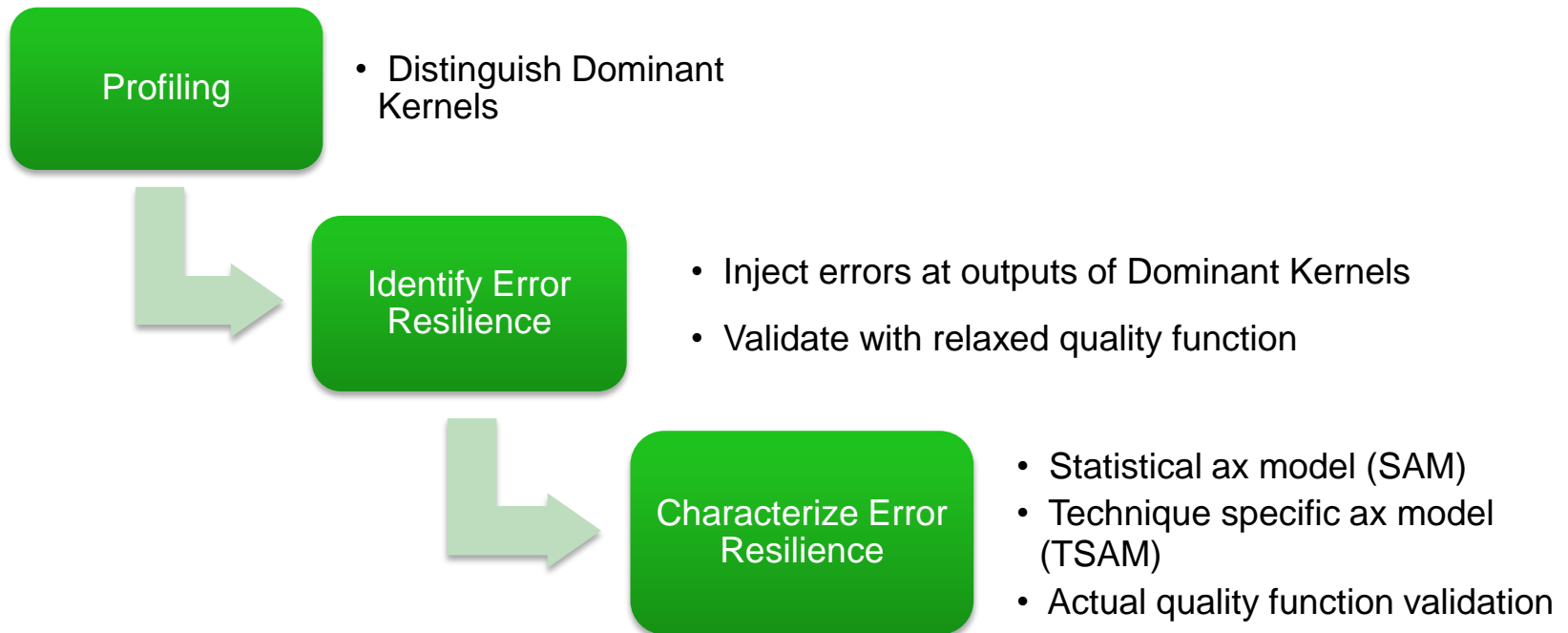
2. Asit Mishra, et al. iACT: A software hardware framework for understanding the scope of approximate computing. WACAS 2014

3. Pooja Roy, et al. Asac: Automatic sensitivity analysis for approximate computing. ACM SIGPLAN 2014

APPROXIMATE COMPUTING

INTRODUCTION

Error Resilience analysis methodology



Vinay K. Chippa et al “Analysis and Characterization of Inherent Application Resilience for Approximate Computing” DAC 2013

APPROXIMATE COMPUTING

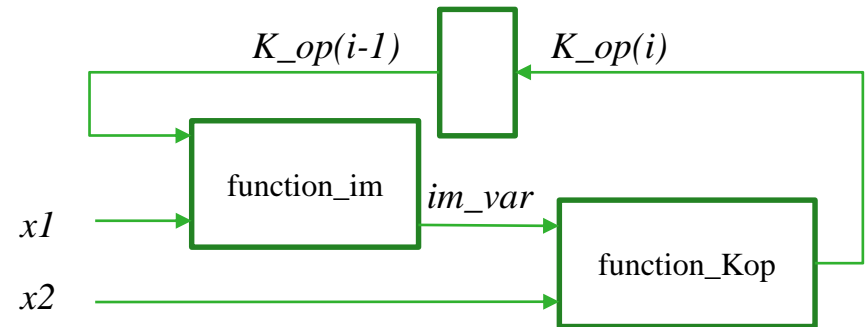
INTRODUCTION

Iterative workload

Input: x_1, x_2

Output: K_{op}

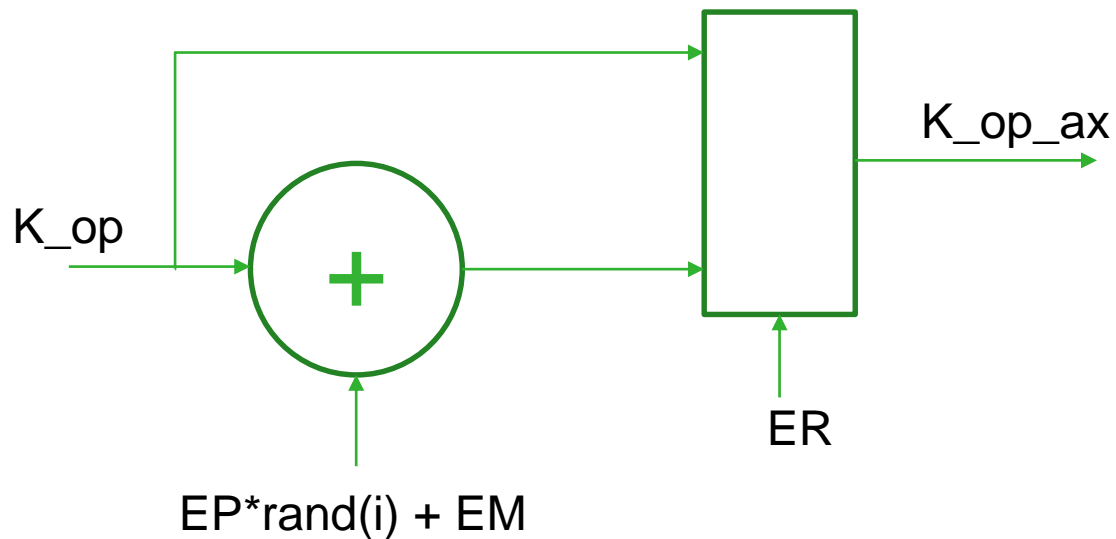
- 1: Initialize $K_{op}(0)$
- 2: **for** $i = 1, 2, \dots, N$ **do**
- 3: $im_var = \text{function_im}(x_1, K_{op}(i - 1));$
- 4: $K_{op}(i) = \text{function_Kop}(im_var, x_2);$
- 5: $convergence_met = \text{function_conv}(K_{op}(i), K_{op}(i - 1));$
- 6: **if** ($convergence_met \leq tol$) **then**
- 7: break; // convergence reached
- 8: **end if**
- 9: **end for**



APPROXIMATE COMPUTING

INTRODUCTION

Model for SAM analysis for iterative workloads



APPROXIMATE COMPUTING

INTRODUCTION

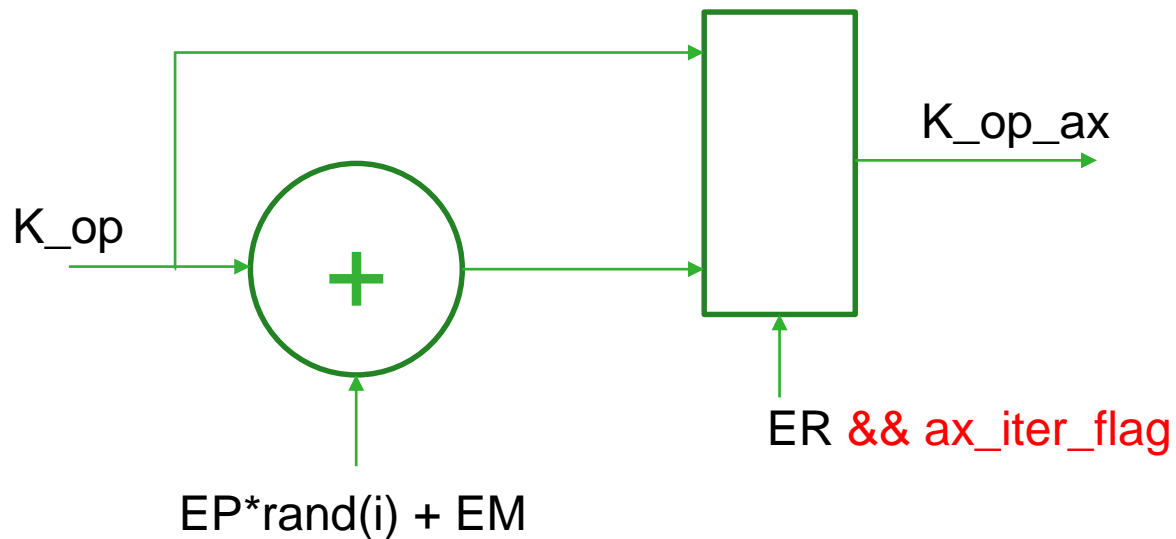
- Iterative workloads are potential candidates of approximate computing such as K-means¹, GLVQ¹ and Model Predictive Control²
- SAM analysis provides resilience profile based on error injection in every iteration
- What if the application is not error resilient for all iterations, but can utilize approximate computing for some iterations?
- Adaptive statistical profile is required that also quantifies the number of approximate iterations

1. Jiayuan Meng, et al. Best-effort parallel execution framework for recognition and mining applications. IPDPS 2009
2. Antonio et al. More flops or more precision? Accuracy parameterizable linear equation solvers for model predictive control. FCCM'09

APPROXIMATE COMPUTING

INTRODUCTION

Model for Adaptive-SAM analysis for iterative workloads



APPROXIMATE COMPUTING

INTRODUCTION

Advantages:

- Adaptive-SAM can apply statistical errors adaptively by selecting Number of Approximate Iterations (NAI)
- This provides statistical error resilience profile of an iterative workload by quantifying NAI in addition to EM, EP and ER
- The resultant profile can help to better exploit,
 - Accuracy-configurable architectures
 - Heterogeneous (having exact and ax cores) architectures

APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

Radio Astronomy Calibration Application*

- Radio astronomy studies celestial objects at radio frequencies
- Calibration algorithm (StEFCal) is a strict quality of service iterative method
- StEFCal estimates complex antenna gains \mathbf{g}_p for the P sensors in a radio telescope
- The algorithm computes \mathbf{g}_p vector based on visibility matrix (R) and the model covariance matrix (M)

$$g_p^{[i]} = \frac{R_{:,p}^H \cdot Z_{:,p}^{[i-1]}}{(Z_{:,p}^{[i-1]})^H \cdot Z_{:,p}^{[i-1]}} \quad Z = M \odot g^{[i-1]}$$

* Stefano Salvini et al. Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications. Astronomy & Astrophysics 571 (2014), A97

APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

StEFCal Algorithm

- The dominant kernels are element-wise product and dot product
- Quality criterion:

$$\text{Convergence} = \frac{\|g^{[i]} - g^{[i-1]}\|_F}{\|g^{[i]}\|_F} \leq 1.10^{-6}$$

$$\text{Diff_rel} = \frac{\|g_{ex}^{[i]} - g_{ax}^{[i]}\|_F}{\|g_{ex}^{[i]}\|_F} \leq 1.10^{-5}$$

- Quality Acceptance Range is Satisfied (QARS) when both of above satisfy

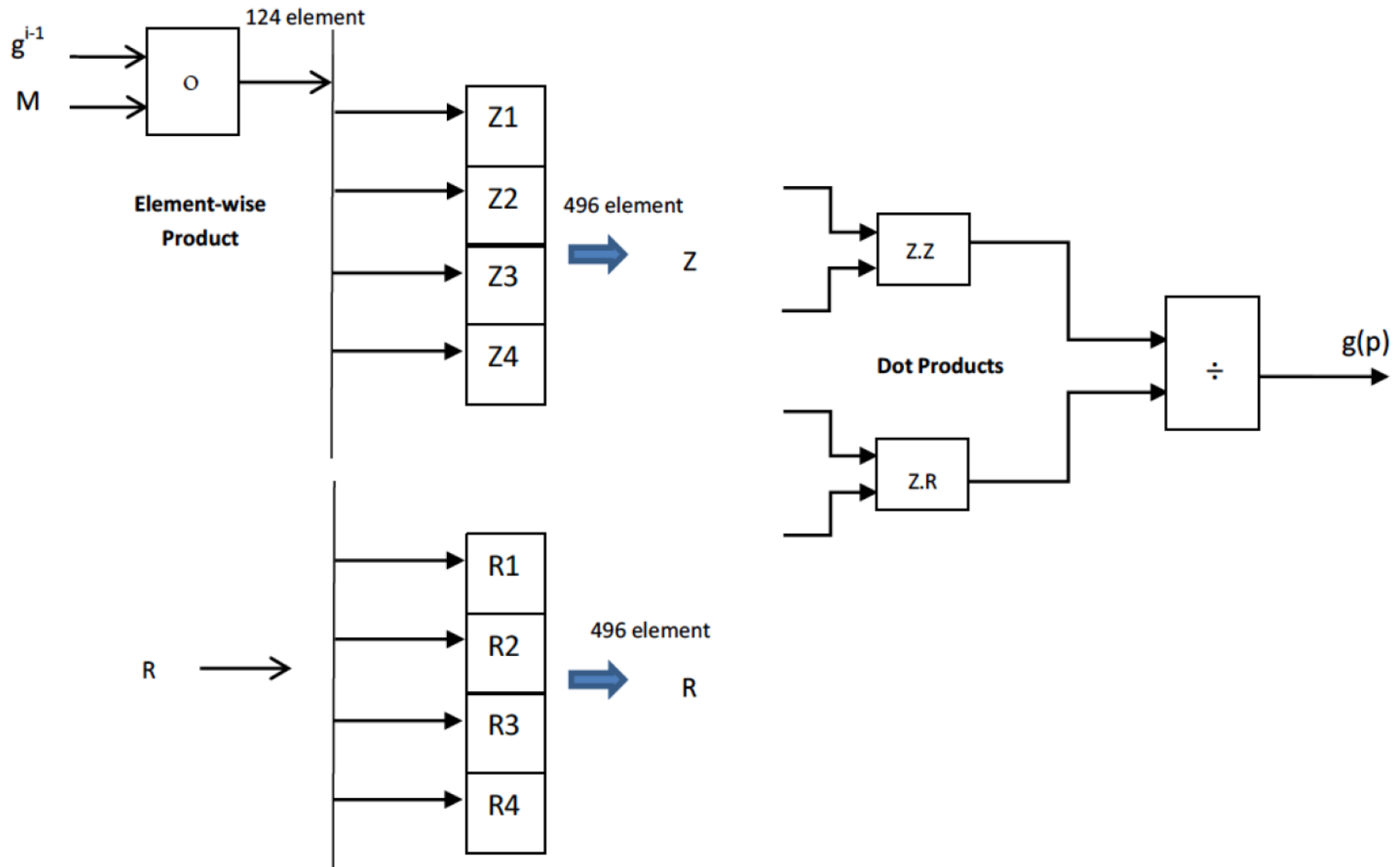


Figure 1.1: Algorithm flow for one antenna gain computation - StEFCal

APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

Algorithm Profiling

- Initialization: $g = \text{norm}(R)/\text{norm}(M)$ % 124x124x4 matrices
- for iter = 1:niter % Until solution converges
 - for j=1:n % Loop over antennas
 - for ch=1:nch % Loop over data samples
 - $z = \text{conj}(\text{gold}) \cdot b(:,j,\text{ch});$ % Element-wise Product
 - $at = a(:,j,\text{ch});$
 - end
 - $w = z' \cdot z;$ % Dot Product outer loop
 - $t = z' \cdot at;$ % Dot Product outer loop
 - $g(j) = t/w;$
 - end
 - end
- Convergence: $dg = \text{norm}(g - \text{gold}, 'fro') / \text{norm}(g, 'fro')$

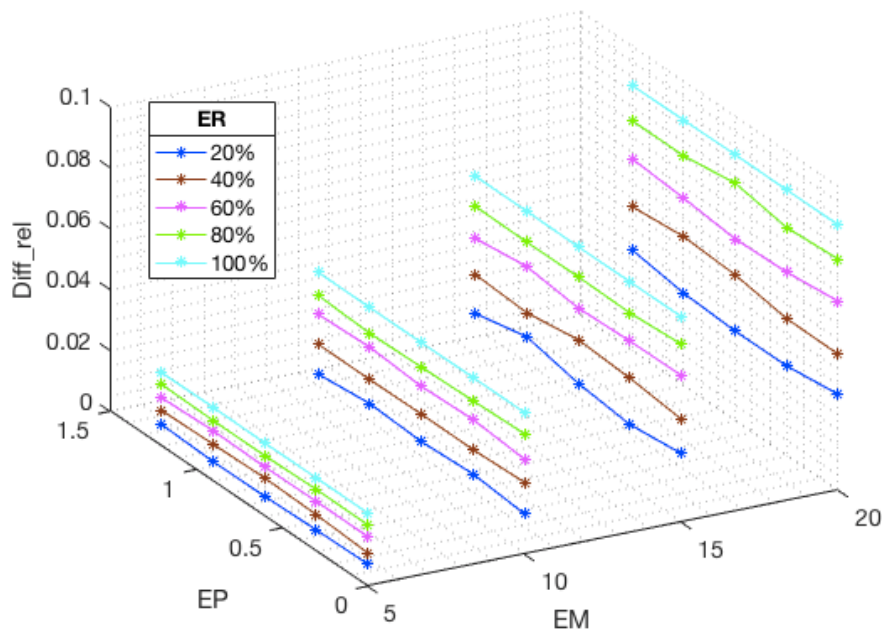


MFLOP		P = 124, ch = 4, Niter = 40
14.7 (27%)		
39.3 (72%)		

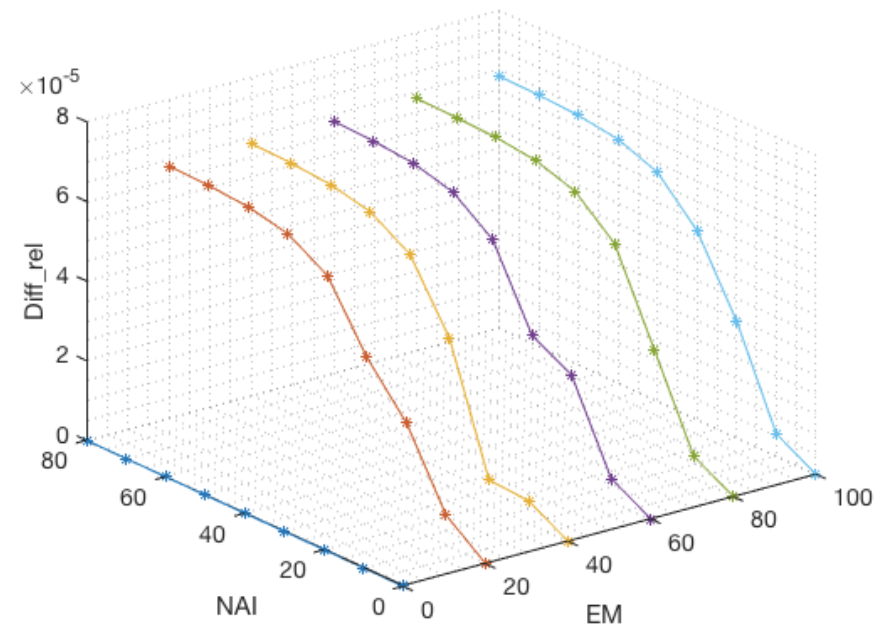
APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

Simulation Results



SAM analysis: Convergence reached
for min EP and max ER



Adaptive-SAM analysis for min EP and
max ER

APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

Simulation Results


- To quantify statistical error resilience – QARS is achieved for:

Approximation Model	EM (%)	EP	ER (%)	NAI (%)
SAM	0.002	$2 \cdot 10^{-4}$	100	100
Adaptive-SAM	12	0.2	100	23

APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL

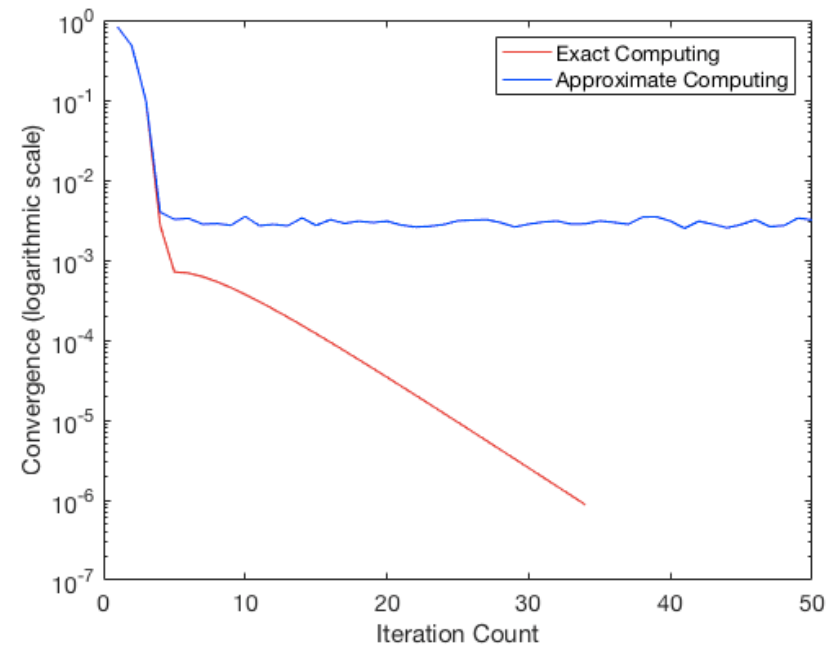
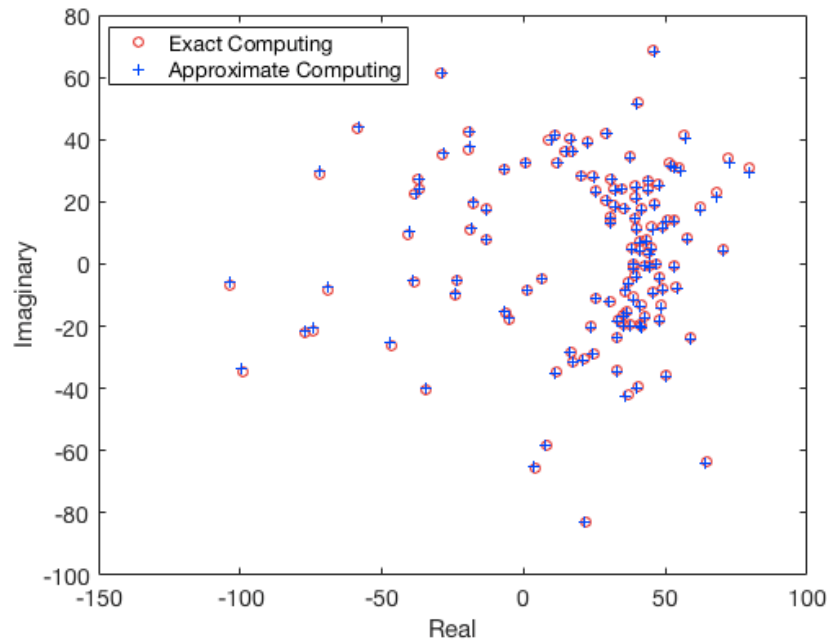
Significance of quality function reconsideration

- Apply approximation models  Validate using quality function
- In iterative workloads, convergence metric is generally utilized to indicate that no further precision can be achieved by computing more iterations
- However, in the error resilience analysis process, it can not be guaranteed that the acceptable solution is achieved when converged
- Perhaps, the solution is precise but not accurate-enough
- In such cases, additional quality metric is to be defined (accuracy based)

APPROXIMATE COMPUTING

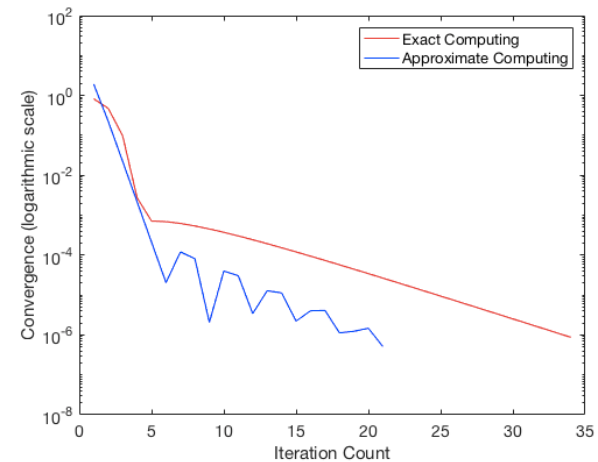
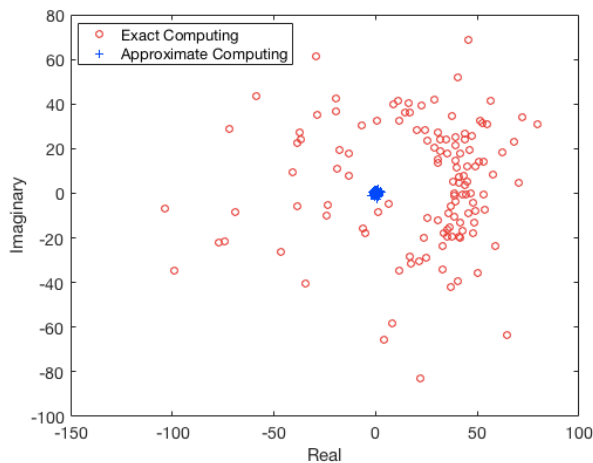
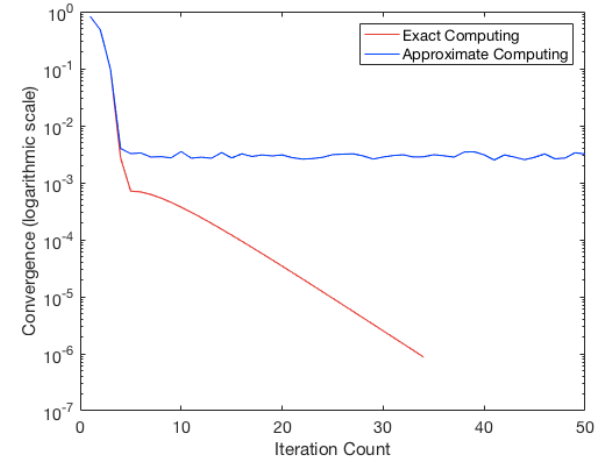
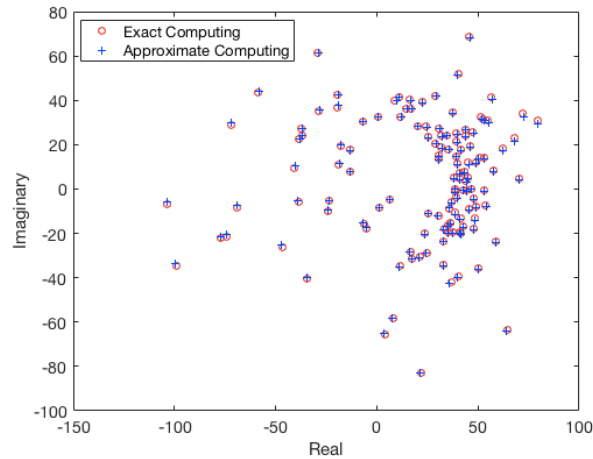
CASE STUDY 1: STEFCAL

Significance of quality function reconsideration;
radio astronomy calibration simulation results



APPROXIMATE COMPUTING

CASE STUDY 1: STEFCAL



APPROXIMATE COMPUTING

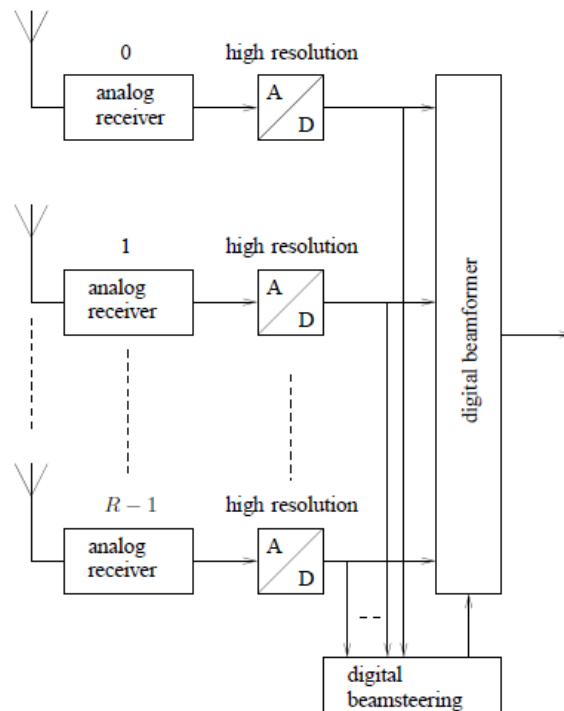
CASE STUDY 1: STEFCAL

Conclusions

- Statistical error analysis helps to reduce design space for ax computing
- Adaptive-SAM has shown improvements in the error resilience analysis of iterative workloads by
 - Quantifying the number of resilient iterations in addition to statistical parameters
 - Better exploiting accuracy-configurable and heterogeneous architectures
- Quality function should be reconsidered in the error resilience analysis process
 - Precision based (convergence) criterion might not be necessarily sufficient
 - Additional quality (accuracy based) metric may be required

APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING



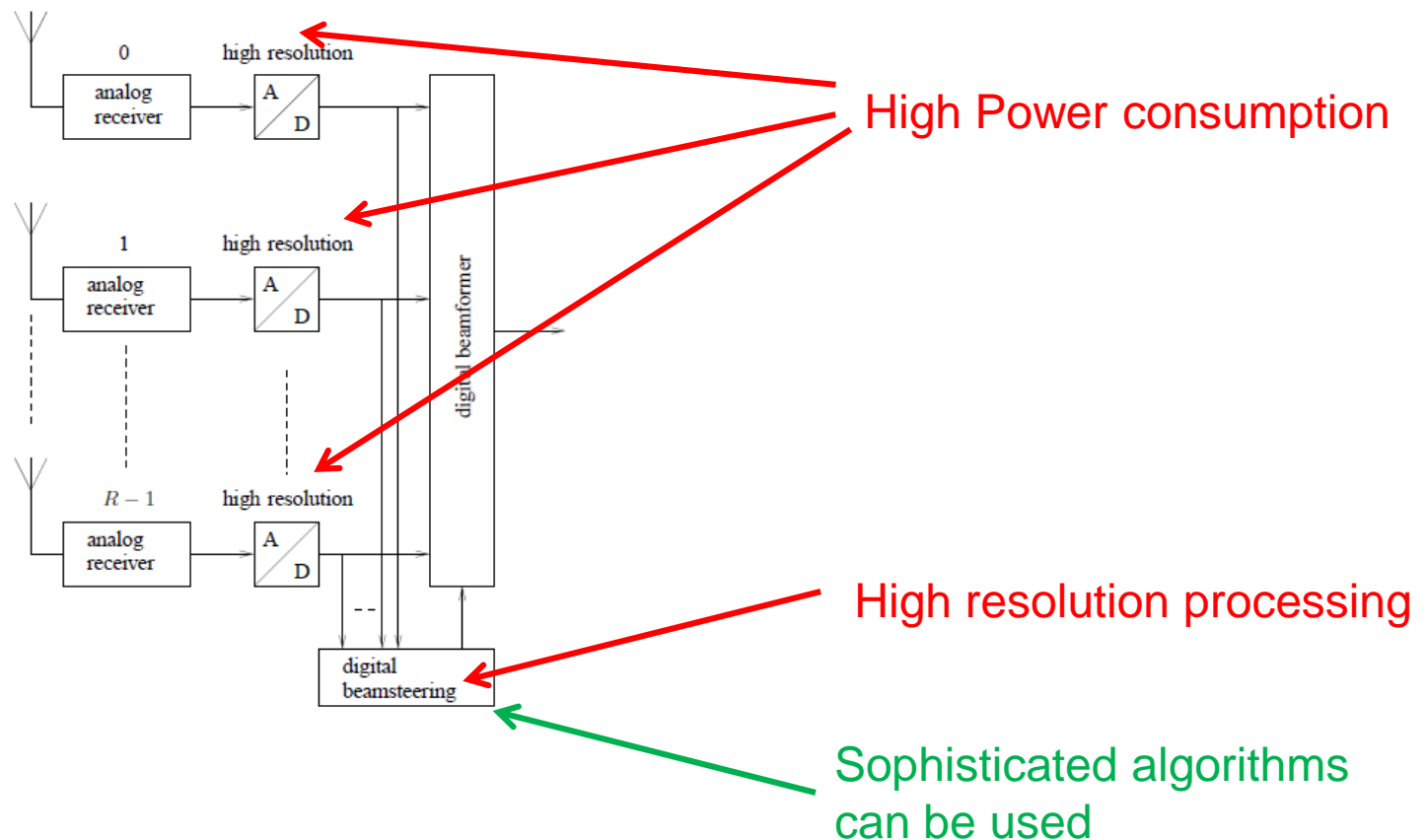
Beamforming:

- Weighted summation of signals from multiple antennas to
 - Enhance wanted signals
 - Suppress unwanted signals
- Elements
 - multiple receivers
 - digital beamformer: Weighted Addition
 - digital beamsteering: determine weights, based on received signals

APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

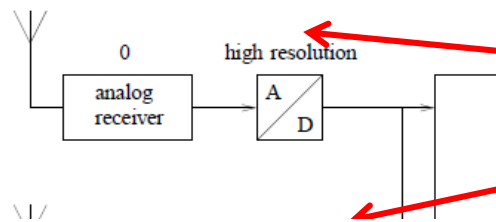
Full feature digital beamforming



APPROXIMATE COMPUTING

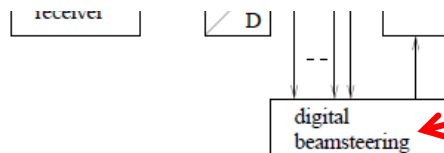
CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

Full feature digital beamforming



High Power consumption

Can we reduce this
and still use this



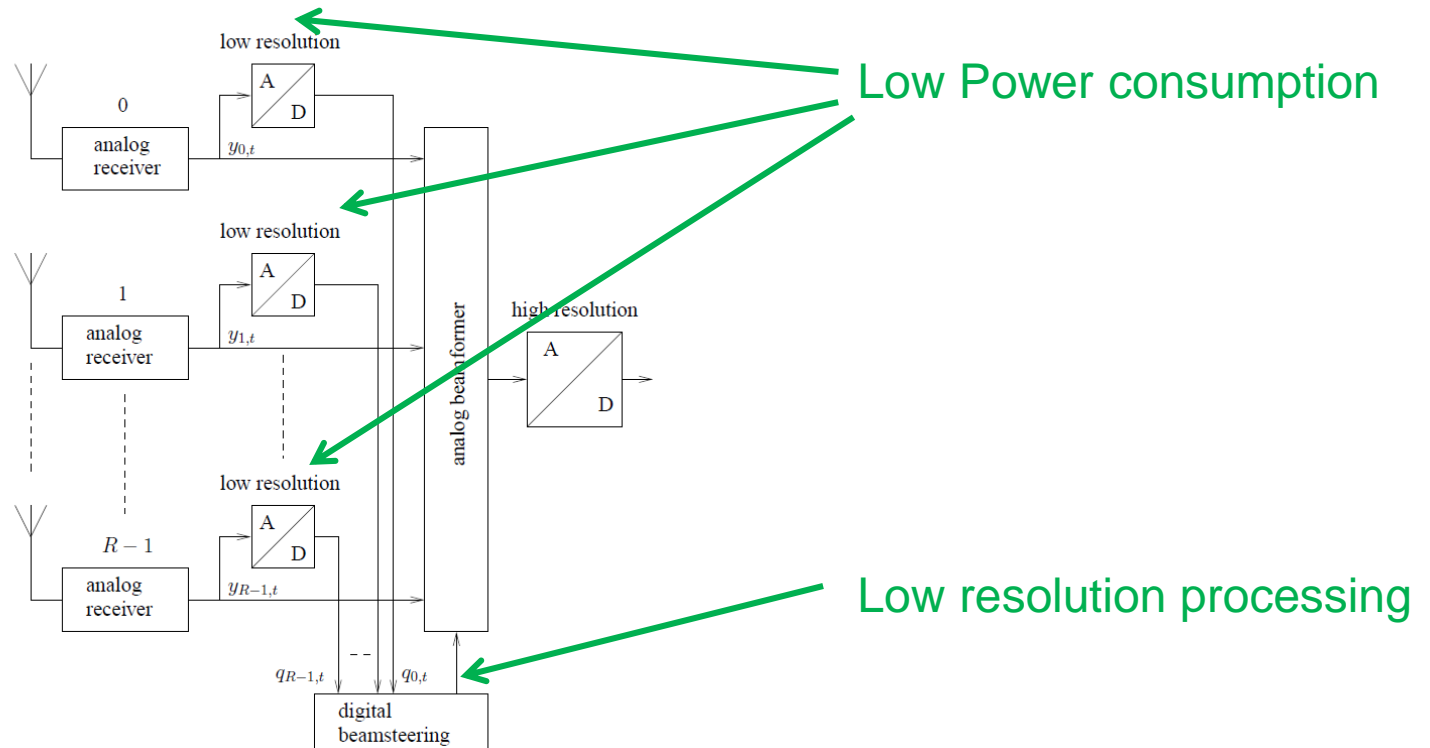
High resolution processing

Sophisticated algorithms
can be used

APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

Digitally assisted analog beamforming

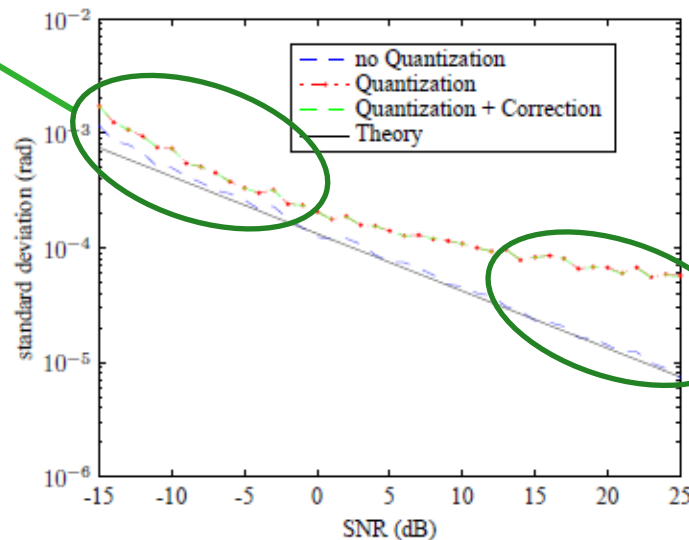


APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

Effects of 1 bit quantization on root music angle estimate

Std. of angle estimate follows theory up to ~0 dB SNR



Quantization noise becomes correlated between different antennas beyond ~0 dB SNR

Figure 2: Standard deviation of the angle estimate with $R = 32$, $T = 1024$, $\theta = 0.1\pi$ radian (18°)

APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

Effects of Van Vleck correction in case of 1 bit quantization

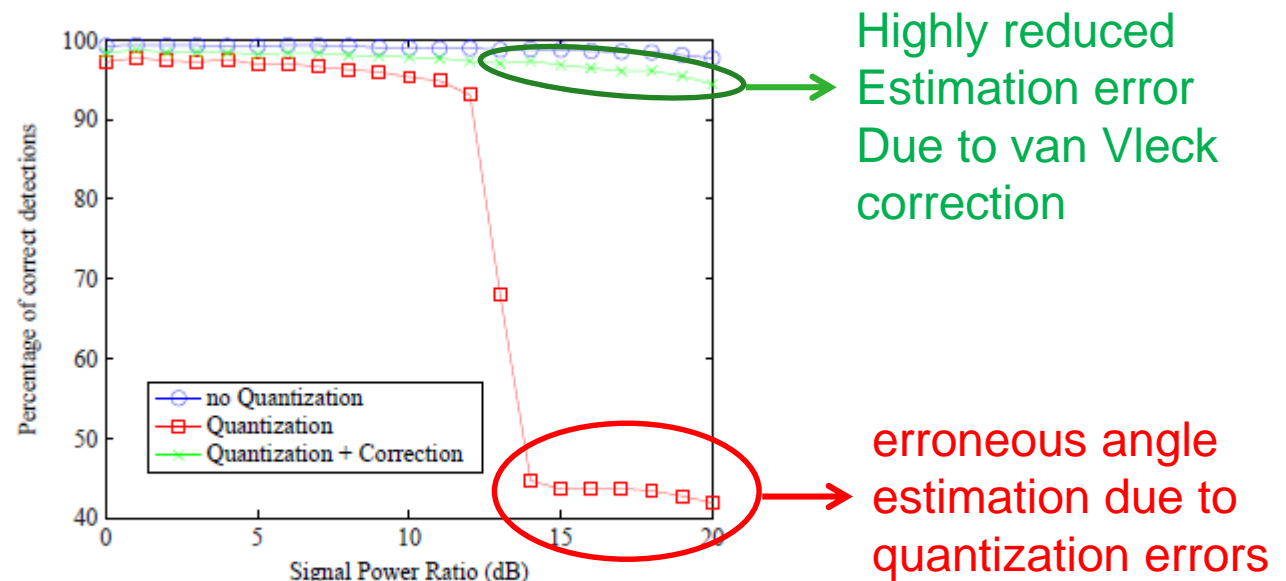


Figure 3: Percentage of correct detection of 2 sources of different strength with $R = 32$, $T = 1024$, $\text{SNR} = 20 \text{ dB}$, $|\text{error}| < 1^\circ$

APPROXIMATE COMPUTING

CASE STUDY 2: DIGITALLY ASSISTED BEAMFORMING

Conclusion

- Digital beamsteering can be combined with analog beamforming
- Results in efficient (low cost, low power, low overhead) solutions
- Key is the use of coarse quantization
- Effects of coarse quantization can be mitigated by digital postprocessing
- Shown by angle estimation using Root Music

THANKS FOR YOUR ATTENTION!

