A Mixed Precision Monte Carlo Methodology for Reconfigurable Accelerator Systems

Gary Chow[1], Anson Tse[1], Qiwei Jin[1],
Wayne Luk[1], Philip Leong[2], David Thomas[1]

Imperial College London[1]
University of Sydney[2]

Contributions

1. mixed precision algorithm: Monte Carlo simulation
   - low precision datapaths: efficient but introduce errors
   - high precision datapaths: correct errors in low precision
   - desired accuracy: appropriate precisions + iterations
2. model: optimise accelerator performance
   - solve by convex programming
3. comparison: with **quad-core** i7, C2070 GPU
   - up to 163 x speed, 170 x energy efficiency of i7
   - up to 4.6 x speed, 5.5 x energy efficiency of C2070

Monte Carlo simulation

- MC simulations used in many applications
  - eg financial, scientific, engineering …
- solve equations without analytical solutions
- usually intuitive and easy to formulate
- usually time consuming if accurate results required
- involve repeated random sampling

MC simulation: overview

- input parameters
  - sampling function $f(x_1, x_2 \ldots)$
  - range(s) for inputs $(x_1, x_2 \ldots)$
- process
  - generate random inputs within given ranges
  - find mean value of f
- MC simulation: sample mean value

Two types of errors in MC simulation

- sampling error (ϵ_S): $\epsilon_S \sim N(0,\sigma_f^2/N)$
  - follow a normal distribution (approximately)
  - σ_f : standard deviation of the sampling function
  - N: number of samples
  - no absolute bounds, only confident interval
- finite precision error (ϵ_{fin})
  - due to inexact arithmetic
  - increase as precision L of data-paths decrease
  - might be bounded
  - assume negligible for large world length (eg *double*)

1. Mixed precision method: motivation

- General Purpose Processors (GPPs)
  - double precision
- FPGA
  - reduced precision computation
- idea: correct FPGA errors using GPP
  - auxiliary sampling $f_a(\mathbf{x})$ *covering* finite precision error

$$f_a(\mathbf{x}) = f_H(\mathbf{x}) - f_L(\mathbf{x})$$

  - $f_H(\mathbf{x}) = f(\mathbf{x})$ evaluated in high precision
  - $f_L(\mathbf{x}) = f(\mathbf{x})$ evaluated in low precision

How to correct finite precision error?

- given magnitude and sign of $< f_a >$ (its mean value)
  - offset finite precision error in $<f_L>$
- strategy
  - many samples to find $<f_L>$
  - small number of samples for $<f_a>$

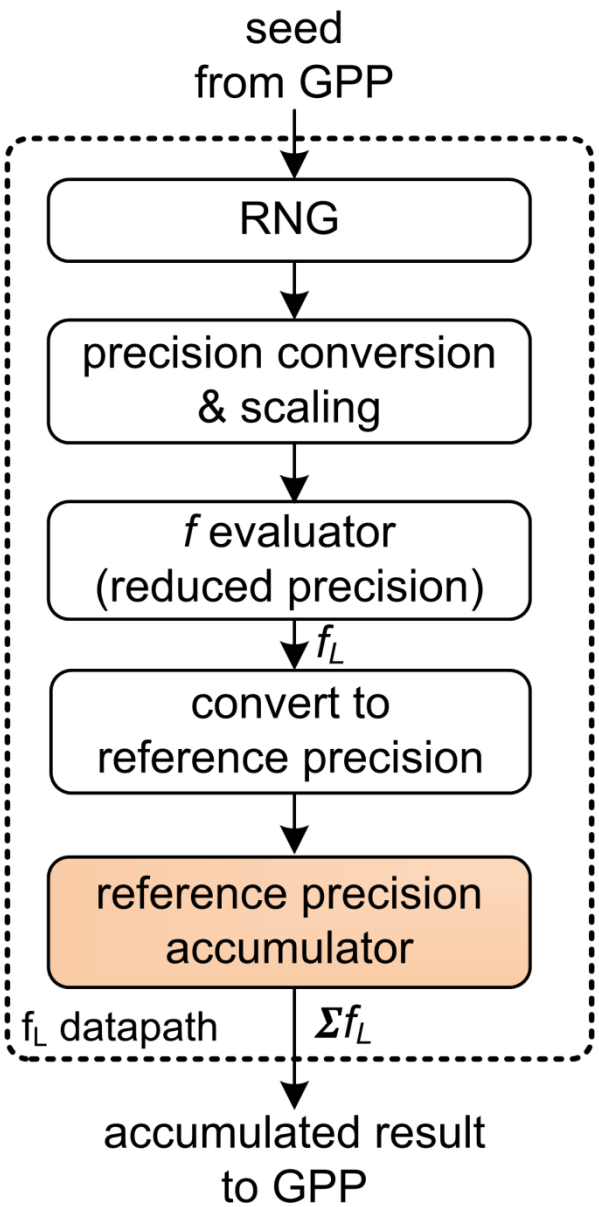
$$<f_H> \approx I_{mixed} = <f_L>_{N_L} + <f_a>_{N_a} \qquad \textit{where } N_L >> N_a$$

- most evaluations of f will be in low precision

Summary: errors for mixed precision

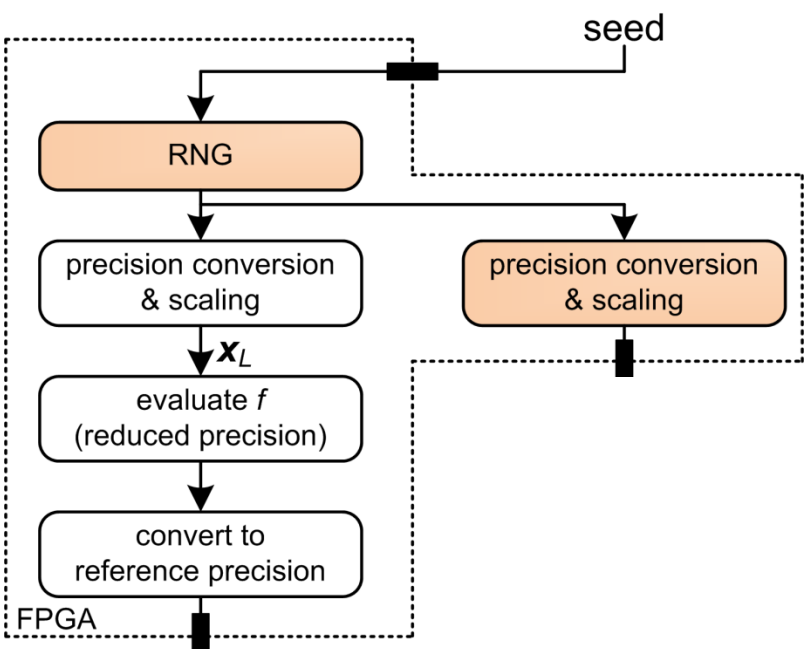
- given $\epsilon_S(I_{mixed}) \sim N(0,\sigma_{fL}^2/N_L + \sigma_{fa}^2/N_a)$
  - σ_{fL},σ_{fa}: *standard deviation of low prec./auxiliary sampling*
  - N_L,N_a *: number of samples*
- achieve any accuracy: adjust $\boldsymbol{N_L}$ (large), $\boldsymbol{N_a}$ (small)
- no finite precision error
  - $\langle f_H \rangle_\infty$ = expected mean of I_{mixed}
- sampling errors
  - two samplings, so two sampling errors
  - both approximately normally distributed, sum also

Mapping to reconfigurable accelerator



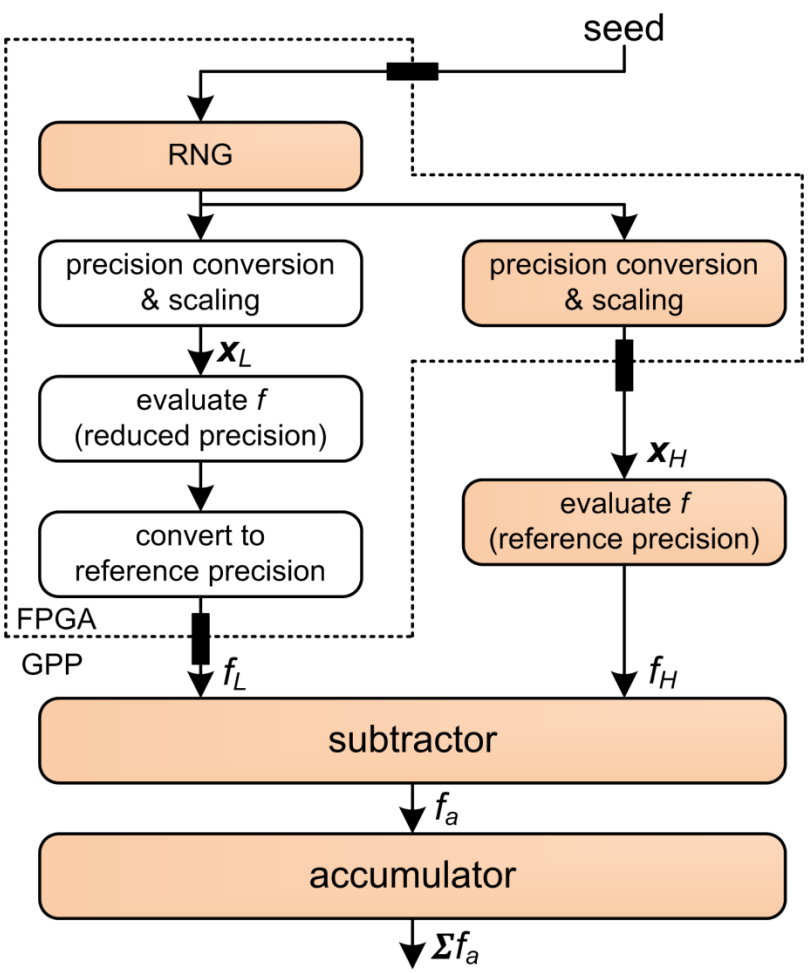
- $\langle f_L \rangle_{N_L}$: low precision computation
  - reduced precision
- entire data-path is in FPGA
- accumulator: high precision
  - reference precision (eg double)
  - prevent errors due to insufficient dynamic range (not modeled)

Mapping to reconfigurable accelerator



- $\langle f_a \rangle_{N_a}$
  - reduced + reference precision
- why FPGA?
  - random number generation: customisable parallelism
  - computational datapath: reduced precision

Mapping to reconfigurable accelerator



(c)

- FPGA to GPP
  - reduced precision result $f_L(\boldsymbol{x})$
  - random number $\boldsymbol{x}$
- GPPs compute
  - $f_H(\boldsymbol{x})$
  - $f_a(\boldsymbol{x}) = f_H(\boldsymbol{x}) - f_L(\boldsymbol{x})$

2. Model: Resource allocation

FPGA resource implement either:

1. reduced precision data-paths
2. "half" of the auxiliary sampling data-paths
   — reduced precision, without accumulator

$$\epsilon_S(I_{mixed}) \sim N(0, \sigma_{fL}^2/N_L + \sigma_{fa}^2/N_a)$$

- large σ_{fL}: assign resource to reduced precision
  – N_L can be large
  – error in the first term small
- large σ_{fa}: assign resource to auxiliary sampling
- find the optimal?

Our analytical model

- convex model
  - solvable by integer geometric programming
- constraints
  - area of FPGA data-paths
  - GPP performance and bandwidth
- different number of FPGAs and GPPs
  - project results to new systems
- solution for a particular reduced precision
  - does not tell which precision is optimal

Applying model for resource allocation

Step 1: collect information (for a particular precision)

1. cost, performance of FPGA data-paths + GPP evaluations
2. bandwidth of the system
3. σ_{fL} and σ_{fa}
4. output accuracy requirement

Step 2: compute minimal time using the convex model

– using parameters of this reduced precision

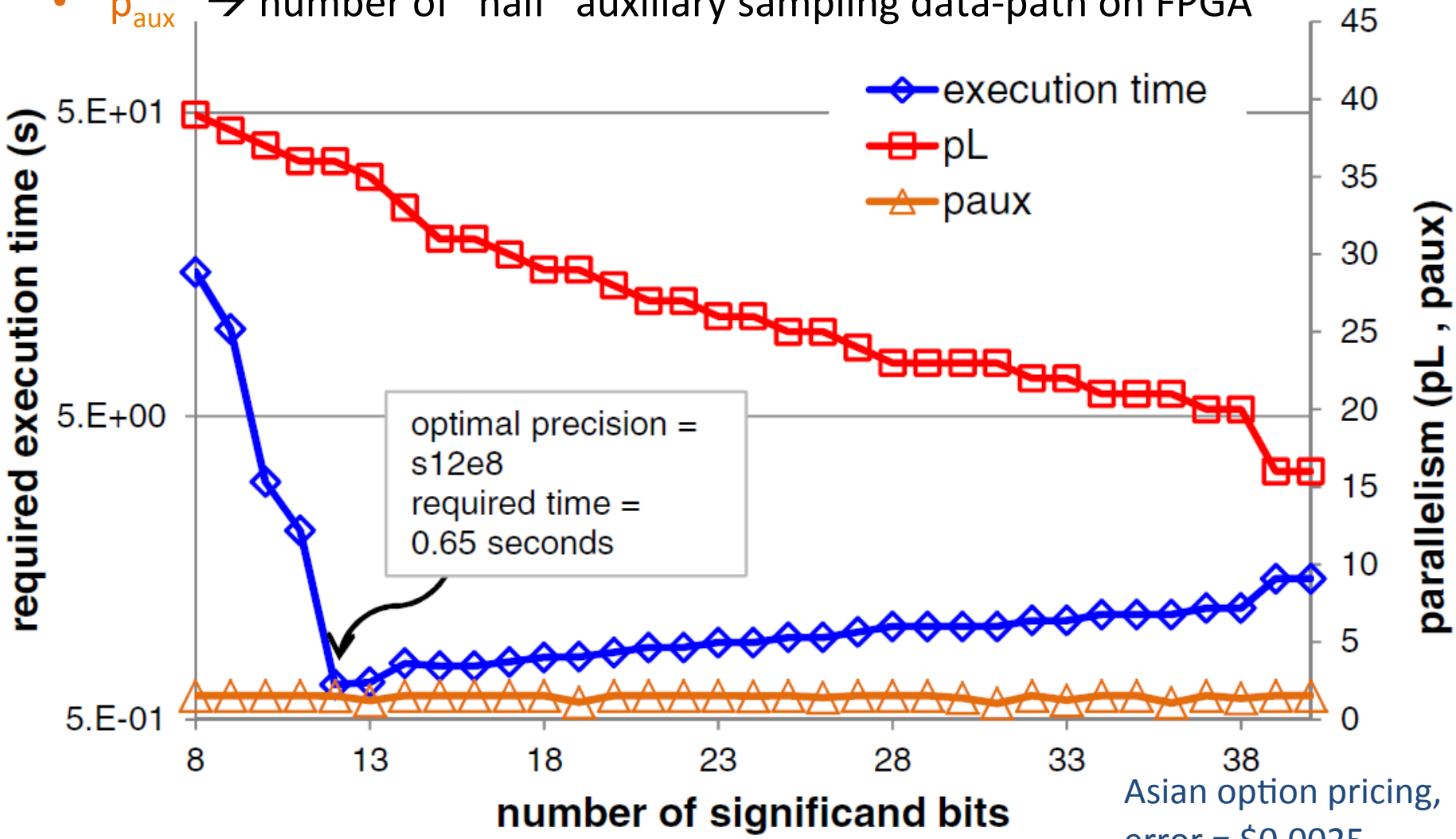
→ get optimal resource allocation for this precision

Step 3 (enumeration): repeat step 1 and 2

– for global optimal reduced precision and resource allocation

Enumeration for optimal precision

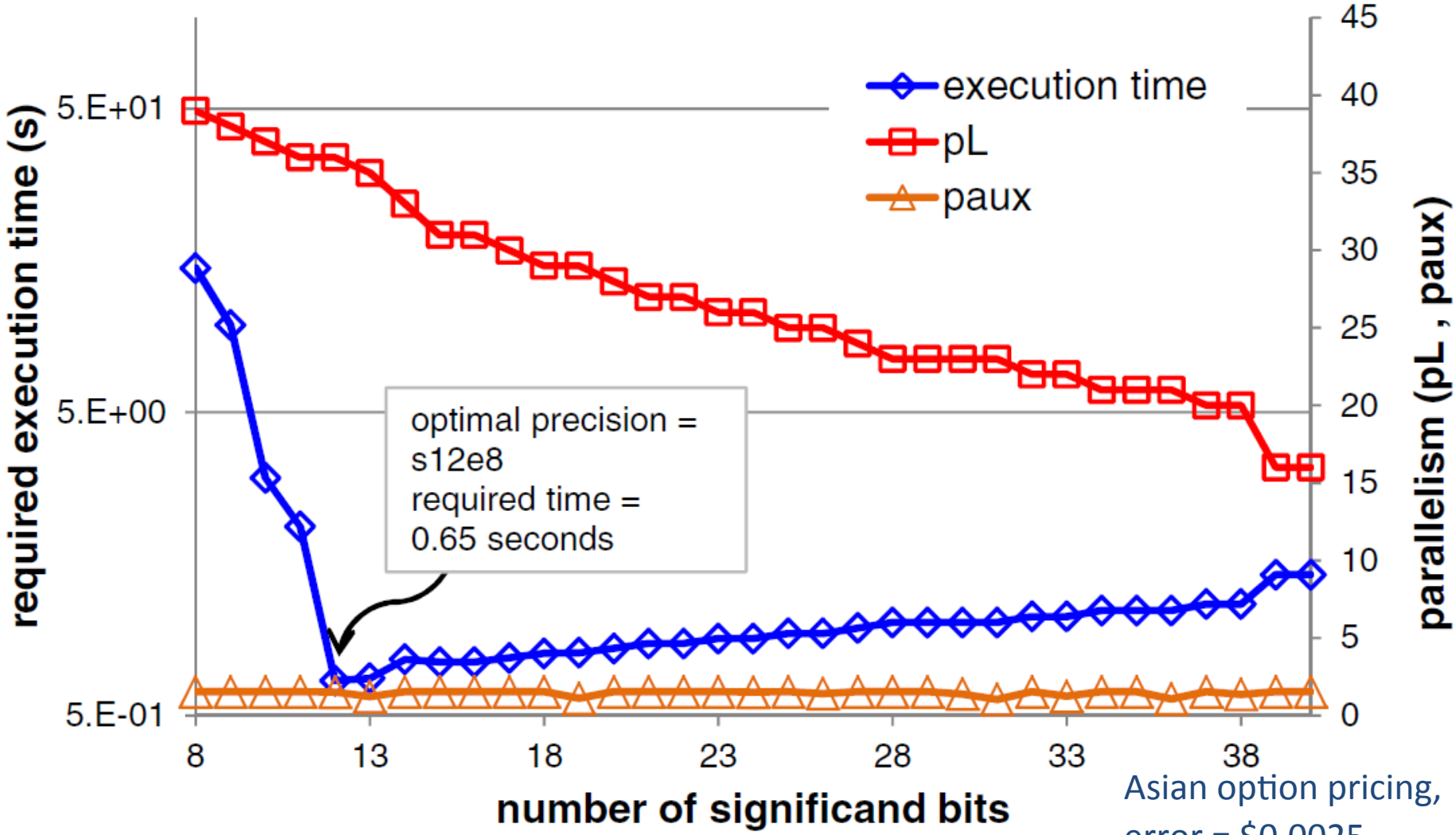
- solve IGPs for each reduced precision
- p_L → number of reduced precision data-path
- p_{aux} → number of "half" auxiliary sampling data-path on FPGA



Asian option pricing, error = $0.0025

Enumeration for optimal precision

- there exists optimal precision: minimum execution time
- output accuracy: same for different reduced precision



Asian option pricing, error = $0.0025

3. Evaluation: FPGA and GPP

- MaxWorkstation from Maxeler Technologies
  - Xilinx Virtex-6 SX475T FPGA
  - Intel i7-870 quad-core GPP host
  - MaxCompiler, map Java dialect into FPGA
- Intel Compiler + Intel Vector Statistic Library
- all software optimised as much as possible
- benchmarks
  - 3 financial: Asian options, GARCH, CMO
  - numerical integration

Three Setups

1. **SW**: software floating-point
   - double precision in four i7-870 GPP cores
2. **FP**: hardware floating-point
   - double precision in FPGA data-paths
3. **Mixed**: mixed precision, hardware + software
   - four i7-870 GPP cores: double precision
   - FPGA: reduced precision, from enumeration
   - resource allocation: from analytical model

Results: parallelism

	Asian Option		GARCH		CMO		Integration	
	FP	mixed	FP	mixed	FP	mixed	FP	mixed
# of double precision cores	5	-	5	-	5	-	5	-
# of reduced precision cores (for reduced precision sampling)	-	36	-	24	-	20	-	16
# of "half" auxiliary sampling cores (for correction with GPPs)	-	1.5	-	0.9	-	0.65	-	0.18

- parallelism of FPGA cores: much increased
  - optimal precision: 12-bit mantissa, 8-bit exponent
- non-integer "half" cores
  - processing power of GPP not multiple of FPGA core

Results: evaluation distribution

	Asian Option		GARCH		CMO		Integration	
	SW/FP	mixed	SW/FP	mixed	SW/FP	mixed	SW/FP	mixed
# reduced precision evaluation (M)	-	12	-	321	-	7.2	-	2320
# reference precision evaluation (M)	11.3	0.47	317	11.6	6.75	0.23	2230	26.8
total # evaluation	**11.3**	**12.5**	**317**	**333**	**6.75**	**7.43**	**2230**	**2347**
Additional evaluation (%)	-	10.6	-	4.8	-	10	-	5.5
% eval. In reference precision	100	**3.8**	100	**3.5**	100	**3.1**	100	**1.1**

- mixed precision: more function evaluations
  - 4.8 to 10.6% more double precision in software or FPGA
- 1.1 to 3.8 % of total evaluations in double precision
  - the rest: 12-bit mantissa, 8-bit exponent

Results: execution time

	Asian Option			GARCH			CMO			Integration		
	SW	FP	Mix	SW	FP	Mix	SW	FP	Mix	SW	FP	Mix
Execution time (s)	29	4.7	0.66	1560	131	26.6	117	2.8	0.72	95.8	2.6	0.9
Normalised speedup	1x	6.2x	44x	1x	12x	59x	1x	42x	163x	1x	37x	106x
Mixed precision gain	-	1x	**7.1x**	-	1x	4.9x	-	1x	3.9x	-	1x	**2.9x**

- mixed precision designs
  - up to 163x speed of quad-core GPP designs
  - 2.9 to 7.1x speed of double precision FPGA only design
- CMO speedup is huge
  - because of the inverse trigonometric function

Results: energy efficiency

	Asian Option			GARCH			CMO			Integration		
	SW	FP	Mix	SW	FP	Mix	SW	FP	Mix	SW	FP	Mix
Execution time (s)	29	4.7	0.66	1560	131	26.6	117	2.8	0.72	95.8	2.6	0.9
Power (W)	183	85	**192**	179	90	**181**	175	94	**171**	184	90	**189**
Total energy (kJ)	5.3	0.4	**0.13**	280	11.8	**4.8**	20.4	0.26	**0.12**	17.6	0.23	**0.17**
Normalised energy	41x	3.1x	1x	58x	2.5x	1x	170x	2.2x	1x	104x	1.4x	1x

- system power is **measured**
  - include hard drive, memory, …
- mixed precision: highest power consumption
  - but best energy efficiency, since they run much faster

GPU comparison: Tesla C2070

	GPP only	GPU	FP	Mixed
precision	double	double	double	mixed
execution time (s)	29	3	4.7	0.65
power (W)	183	236	85	192
total energy (kJ)	5.3	0.71	0.4	0.13
normalised speedup	1x	9.7x	6.2x	44.6x
normalised energy	40.7x	**5.5x**	3.1x	1x

- mixed precision (GPP + FPGA) vs GPU
  - 4.6 times faster (3/0.65)
  - 5.5 times more energy efficient

Current and future work

- FPGA only mixed precision: auxiliary sampling in
  - fine-grained + coarse-grained programmable fabric
  - on-chip microprocessors, eg ARM9
- mixed precision designs on GPU
  - sub-single, single, double precision
- mixed precision for heterogeneous systems
  - GPP, GPU, FPGA
- improve automation
  - optimise precision, resource allocation, code generation
- extend range of applications
  - eg quasi-Monte Carlo simulation, function comparison

Summary

- mixed precision Monte Carlo design
  - low precision datapaths: efficient but introduce errors
  - high precision datapaths: correct errors in low precision
  - desired accuracy: appropriate precisions + iterations
- analytical model: optimise resource allocation
- 4 large benchmarks
  - Asian options, GARCH, CMO, numerical integration
- comparison: with quad-core i7, C2070 GPU
  - up to 163 x speed, 170 x energy efficiency of i7
  - up to 4.6 x speed, 5.5 x energy efficiency of C2070