A Mixed Precision Monte Carlo Methodology for Reconfigurable Accelerator Systems

Gary C.T. Chow
Dept. of Computing
Imperial College London
SW7 2AZ, United Kingdom
cchow@doc.ic.ac.uk

Anson H.T. Tse
Dept. of Computing
Imperial College London
SW7 2AZ, United Kingdom
htt08@doc.ic.ac.uk

Qiwei Jin
Dept. of Computing
Imperial College London
SW7 2AZ, United Kingdom
qj04@doc.ic.ac.uk

Wayne Luk
Dept. of Computing
Imperial College London
SW7 2AZ, United Kingdom
wl@doc.ic.ac.uk

Philip H.W. Leong
School of Electrical and
Information Engineering
University of Sydney
Sydney, Australia
philip.leong@sydney.edu.au

David B. Thomas
Dept. of Electrical and
Electronic Engineering
Imperial College London
SW7 2AZ, United Kingdom
d.thomas1@imperial.ac.uk

ABSTRACT

This paper introduces a novel mixed precision methodology applicable to any Monte Carlo (MC) simulation. It involves the use of data-paths with reduced precision, and the resulting errors are corrected by auxiliary sampling. An analytical model is developed for a reconfigurable accelerator system with a field-programmable gate array (FPGA) and a general purpose processor (GPP). Optimisation based on mixed integer geometric programming is employed for determining the optimal reduced precision and optimal resource allocation among the MC data-paths and correction data-paths. Experiments show that the proposed mixed precision methodology requires up to 11 % additional evaluations while less than 4 % of all the evaluations are computed in the reference precision; the resulting designs are up to 7.1 times faster and 3.1 times more energy efficient than baseline double precision FPGA designs, and up to 163 times faster and 170 times more energy efficient than quad-core software designs optimised with the Intel compiler and Math Kernel Library. Our methodology also produces designs for pricing Asian options which are 4.6 times faster and 5.5 times more energy efficient than NVIDIA Tesla C2070 GPU implementations.

Categories and Subject Descriptors

C.0 [**Computer System Organization**]: System architecture; G.1.0 [**Numerical analysis**]: Multiple precision arithmetic

General Terms

Design, algorithms, performance

FPGA'12, February 22–24, 2012, Monterey, California, USA.

Keywords

Monte Carlo, mixed precision

1. INTRODUCTION

Monte Carlo (MC) simulations are a class of algorithms based on randomisation which are extensively used in many high performance computing applications in science, engineering and finance. High performance computing is often needed to solve these problems since they are computationally expensive. MC simulations are well suited to Field Programmable Gate Arrays (FPGAs), due to the parallel nature of MC algorithms and the availability of cost-effective random number generators for FPGAs. It has been shown that FPGA-based Monte Carlo applications can offer 1-2 orders of speedup over their software counterparts running on high-end CPUs [12, 21].

The ability to support customizable data-paths of different precisions is an important advantage of reconfigurable hardware. Reduced-precision data-paths usually have higher clock frequencies, consume fewer resources and offer a higher degree of parallelism for a given amount of resources compared with full precision data-paths. Although the use of reduced precision can lead to higher performance, it also affects the accuracy of the results. Most FPGA Monte Carlo designs exploit this trade-off and use data-paths that are sufficiently accurate to produce outputs within the required error tolerance [11, 17, 21]. However, when very accurate outputs are required, high precision data-paths with lower performance are unavoidable. This makes FPGAs less attractive in MC applications with high accuracy requirements.

This paper introduces a novel mixed precision methodology for accurate Monte Carlo simulations. The key difference between the proposed methodology and previous FPGA Monte Carlo designs lies in the way finite precision errors are handled. Instead of keeping the output error within a certain tolerance, the FPGA data-path is initially constructed with an aggressively reduced precision. This produces a result with finite precision error exceeding the error tolerance. An auxiliary sampling process using both a high precision reference and the reduced precision is then used to correct the error. The output accuracy of the proposed technique is not limited by the precision of the data-paths. The

proposed methodology can also exploit the synergy between different processors in a reconfigurable accelerator system. Reference precision computations required in the auxiliary sampling can be carried out by a general purpose processor (GPP) in a host PC, while reduced precision computations target customized data-paths on the FPGA. This allows different processors to work in precisions for which they are specialised, leading to higher overall performance.

The major contributions of this paper are:

- an error analysis that separates finite precision error and sampling error for reduced precision Monte Carlo simulations, and a novel mixed precision methodology to correct finite precision errors through auxiliary sampling (Section 2 and Section 3).
- techniques for partitioning workloads of different precisions for auxiliary sampling to a reconfigurable accelerator system consisting of FPGA(s) and GPP(s) (Section 4).
- an optimisation method based on an analytical model for the execution time of a Monte Carlo simulation on a reconfigurable accelerator system, and Mixed Integer Geometric Programming to find optimal precision for the FPGA's data-paths and optimal resource allocation (Section 5).
- evaluation of the proposed methodology using four case studies, with performance gains of 2.9 to 7.1 times speedup over FPGA only designs using double precision arithmetic. The mixed precision designs are also 44 to 163 times faster and 41 to 170 times more energy efficient compared with software design on a quad-core GPP (Section 6 and 7).

2. BACKGROUND

This section provides an error analysis for Monte Carlo simulations. The total error ϵ_{total} of a Monte Carlo simulation can be divided into two components. Sampling error ϵ_S is the error due to having a finite number of samplings and finite precision error ϵ_{fin} is due to non-exact arithmetic. It is assumed that when a sufficiently accurate precision, such as IEEE-754 double precision, is used, the finite precision error is negligible. We call this value the **reference precision**. We also review how finite precision error is handled in related work. Let us begin with sampling error. Monte Carlo methods are used to simulate random processes and estimate the distribution of the results. Consider a sequence of mutually independent, identically distributed random variables, X_i from a MC simulation. If, $S_N = \sum_{i=1}^{N} X_i$, and the expected value, I, exists, the Weak Law of Large Numbers states that if $p(x)$ is the probability of x, for $\epsilon > 0$, the approximation approaches the mean for large N [8],

$$\lim_{N\to\infty} p\left(|\frac{S_N}{N} - I| > \epsilon\right) = 0 \tag{1}$$

Moreover, if the variance σ^2 exists, the Central Limit Theorem states that for every fixed a,

$$\lim_{N\to\infty} p\left(\frac{S_N - NI}{\sigma\sqrt{N}} < a\right) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{a} e^{-z^2/2}dz \tag{2}$$

that is, the distribution of the standard error is normal.

In practice, we must deal with finite N. If the sampling function f represents a mathematical expression defining the quantity being sampled, $\vec{x_i}$ is the input vector of length s from a uniform distribution [1] $[0,1)^s$, N is the number of sample points and $\langle f_H\rangle_N$ is the sampled mean value of the quantity, the conventional MC sampling process[2] can form an approximation to I,

$$I \approx \langle f_H\rangle_N = \frac{1}{N}\sum_{i=1}^{N} f_H(\vec{x_i}) \tag{3}$$

Thus a sampling error $\epsilon_S(\langle f_H\rangle_N) = I - \langle f_H\rangle_N$ with approximately normal distribution is introduced:

$$\epsilon_S(\langle f_H\rangle_N) \sim \mathcal{N}(0, \sigma^2_{f_H}/N) \tag{4}$$

Equation 4 shows that the bound of the sampling error can be constructed as a confidence interval. Given the same confidence level, the interval is proportional to the standard deviation of the sampling function, σ_{f_H}, and inversely proportional to the square root of the number of sample points, N. Hence quadrupling the number of sample points halves the confidence interval of the sampling error $\epsilon_S(\langle f_H\rangle_N)$. We assume there is no precision error associated with the sampling error. In FPGA designs, the sampling function f is usually evaluated using a low reduced precision, f_L, compared to the high reference precision, f_H. The reduced precision design is smaller and faster, at the expense of higher error. However, reduced precision increases the error. We call the difference between a reference precision computation and a reduced precision computation, $f_H(x) - f_L(x)$, the finite precision error.

Methods for dealing with finite precision error in FPGA-based MC simulations can be classified into two categories. In the first category, only standard precisions such as the IEEE single/double precision are used in sampling data-paths [10, 12]. Users are responsible for determining whether the finite precision error is acceptable, because the FPGA MC engines will follow the result of software exactly.

In the second category, error bounds of the finite precision error are constructed and the precision of the sampling data-path is adjusted such that the error bounds are smaller than the error tolerance. In [11], the maximum relative error of the sampling data-path is used to construct the error bound. The maximum relative error can be characterised using analytical methods such as interval [14] or affine arithmetic [7]. However, these approaches do not take into account that finite precision errors from different sample points might have different signs and would cancel out each other. Hence there is usually an over-estimation of finite precision error in Monte Carlo simulation.

In [17], test runs with a pre-defined number of sample points are used to empirically estimate the maximum percentage error due to finite precision effect empirically. The finite precision error of MC simulations using the same data-path and the same number of sample point are then assumed to share the same error bound. This assumption is not al-

[1]Some MC simulations require non-uniformly distributed $\vec{x}$ values, for example in many option pricing simulations normally distributed $\vec{x_i}$ are required.

[2]Throughout the paper, we use the subscript H and L to denote quantities evaluated with the reference precision arithmetic and the reduced precision arithmetic respectively. We use $\langle X\rangle$ to denote the sampled mean value of a random variable X and $\langle X\rangle_N$ to denote the sampled mean value of X calculated by N samples.

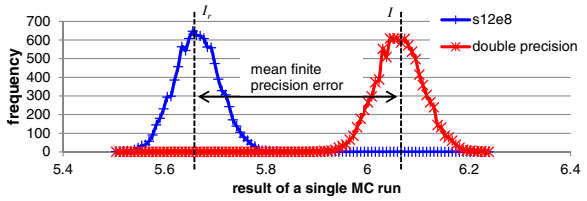

Figure 1: Distribution of 10k runs of a reduced precision and a double precision Monte Carlo.

ways valid and thus the empirical error bound can only be used as a reference rather than a rigorous bound.

In [18], a design is proposed with both high precision and reduced precision data-paths for computing cumulative distribution functions (CDFs). The two CDFs are compared using a Kolmogorov-Smirnov test, the distance score of which is then used to control the precision of the reduced precision data-path adaptively such that finite precision error is within the range of error tolerance.

In [4] a mixed-precision approach for comparison is presented. It is different from this work since it does not involve MC simulation.

The benefits for reduced precision designs are well-known. For instance, it has been shown [5] that appropriate word-length optimisation can improve the area of adaptive filters and polynomial evaluation circuits by up to 80%, power reduction of up to 98%, and speed of up to 36% over common alternative design strategies. This research shows that impressive gains can also be obtained by exploiting reduced precision for complex designs supporting Monte Carlo simulation.

3. MIXED PRECISION METHODOLOGY

Our novel mixed precision methodology is motivated by two ideas. First, we can correct the finite precision error when both its magnitude and sign are known. Second, in Monte Carlo simulations, we are only interested in the finite precision error in the final result but not the finite precision errors of individual sample points.

When a reduced precision data-path is used in a Monte Carlo simulation, the reduced precision expected value I_r is approximated by the following equation, where N_L is the number of sample points:

$$I_r \approx \langle f_L \rangle_{N_L} = \frac{1}{N_L} \sum_{i=1}^{N_L} f_L(\vec{x_i}) \tag{5}$$

Due to the effect of finite precision error, the reduced precision sample mean $\langle f_L \rangle_N$ cannot be used to approximate the expected value I directly as I might not equal to I_r. We define the difference of the two expected means as the mean finite precision error, $\mu_{\epsilon_{fin}}$, where

$$\mu_{\epsilon_{fin}} = I - I_r \tag{6}$$

Figure 1 shows the distributions of Monte Carlo simulations using a reduced precision (s12e8)[3] data-path and a double precision data-path of for pricing Asian options. In each MC simulation, $N = 32{,}768$ sample points are used and each of the reduced and double precision MC simulation is repeated for 10,000 times with different random seeds. As shown in the figure, the magnitude of the mean finite precision error $\mu_{\epsilon_{fin}}$ between the expected value of I and I_r is significant. When reduced precision data-paths are used for the Monte Carlo simulation without the correction by the auxiliary sampling, the true value of the simulation will lie within $\pm\mu_{\epsilon_{fin}}$ of the sampled value. Moreover, this uncertainty cannot be reduced by increasing the number of sample points and is a fundamental limit of conventional reduced precision MC simulations.

To find both the magnitude and the signs of the mean finite precision error $\mu_{\epsilon_{fin}}$, we define an auxiliary sampling function $f_a(\vec{x})$:

$$f_a(\vec{x}) = f_H(\vec{x}) - f_L(\vec{x}) = \epsilon_{fin}(\vec{x}) \tag{7}$$

where ϵ_{fin} is the finite precision error for each $\vec{x}$. With a sufficient large sample size N_a, we can approximate the mean finite precision error $\mu_{\epsilon_{fin}}$:

$$\mu_{\epsilon_{fin}} \approx \langle f_a \rangle_{N_a} = \frac{1}{N_a} \sum_{i=1}^{N_a} f_a(\vec{x_i}) \tag{8}$$

The sampling error of this auxiliary sampling $\epsilon_S(\langle f_a \rangle_{N_a}) = \mu_{\epsilon_{fin}} - \langle f_a \rangle_{N_a}$ is approximately normal distributed:

$$\epsilon_S(\langle f_a \rangle_{N_a}) \sim \mathcal{N}(0, \sigma^2_{f_a}/N_a) \tag{9}$$

Finally, we can approximate the true mean I by two sets of sampling:

$$I_{mixed} = \langle f_L \rangle_{N_L} + \langle f_a \rangle_{N_a} \tag{10}$$

$$\begin{aligned} E(I_{mixed}) &= E(\langle f_L \rangle_{N_L}) + E(\langle f_a \rangle_{N_a}) \\ &= I_r + (I - I_r) = I \end{aligned} \tag{11}$$

As shown in Equation 11, the expected value of the auxiliary sampling is $I - I_r$. Hence the expected mean of the mixed precision approximation I_{mixed} is exactly the same as the expected mean I computed in the reference precision. Equation 10 can thus be viewed as the reduced precision sample mean plus a correction for the mean finite precision error.

Since two samplings are used in the proposed mixed precision methodology, there are two sampling errors in the result and they can be found using Equation 13 and 14. As both sampling errors are approximately normally distributed, their sum is also approximately normally distributed and has a variance equal to the sum of their individual variances as shown in Equation 15 if uncorrelated random numbers are used. By using the proposed mixed precision methodology, we effectively replace the finite precision error of reduced precision data-paths by the sampling error of the auxiliary sampling. A confidence interval can also be constructed using the combined variance.

$$\epsilon_S(I_{mixed}) = \epsilon_S(\langle f_L \rangle_{N_L}) + \epsilon_S(\langle f_a \rangle_{N_a}) \tag{12}$$

$$\epsilon_S(\langle f_L \rangle_{N_L}) \sim \mathcal{N}(0, \sigma^2_{f_L}/N_L) \tag{13}$$

$$\epsilon_S(\langle f_a \rangle_{N_a}) \sim \mathcal{N}(0, \sigma^2_{f_a}/N_a) \tag{14}$$

$$\epsilon_S(I_{mixed}) \sim \mathcal{N}(0, \sigma^2_{f_L}/N_L + \sigma^2_{f_a}/N_a) \tag{15}$$

Although the proposed mixed precision methodology is analysed mathematically, we also show its desired effect through experiments. Using Equation 15, we find that a mixed precision MC run using a precision of s12e8 with $N_a = 1078$ and

[3]In this paper, we use the notation $sAeB$ to denote a floating point representation, where A is the number of significand bits and B is the number of exponent bits.

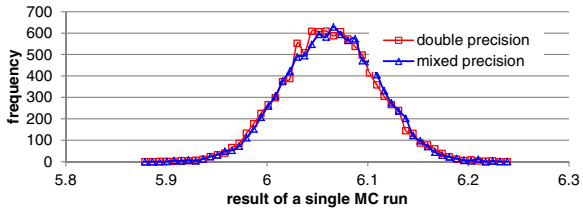

Figure 2: Distribution of 10k runs of a mixed precision and a double precision Mont Carlo.

$N_L = 33{,}773$ should yield the same error as a double precision sampling with $N = 32{,}768$. We repeat both the mixed precision and the double precision MC 10,000 times using different random seeds, and their distributions are shown in Fig. 2. Note that both distributions have roughly the same variance and mean. The result agrees with our mathematical model and no finite precision error exists between the double precision Monte Carlo and our mixed precision Monte Carlo runs.

The proposed mixed precision methodology provides several advantages over previous FPGA designs.

1. The final result is adjusted with an approximated mean finite precision error $\mu_{\epsilon_{fin}}$. This is a novel approach which enables us to obtain a more accurate result from the reduced precision result instead of passively finding the error bound.
2. Since there are only sampling errors in the output, we can achieve very accurate result by increasing the number of sample points N_L and N_a. The output accuracy is no longer limited by the reduced precision.
3. The methodology is applicable to any Monte Carlo simulation because no accuracy analysis is required for the relative error and the methodology is totally independent of the function f.

Although the proposed mixed precision methodology enables us to aggressively exploit reduced precision data-paths while maintaining the accuracy of the final result using auxiliary sampling, each auxiliary sampling still requires a costly evaluation of the sampling function f at the reference precision.

The effectiveness of the proposed technique depends heavily on how resources are allocated among the reduced precision hardware and auxiliary sampling hardware. To find the optimal resource allocation, we should consider a number of factors such as the cost of evaluating f_L and f_H, the area available on the FPGA, the bandwidth between the FPGA and GPP, and the reduced precision values being used.

In the next section, we propose different schemes for partitioning workloads. An analytical model is developed in Section 5 based on the partitioning schemes which enables us to find the optimal resource allocation and optimal reduced precision using mixed integer geometric programming.

4. WORKLOAD PARTITIONING

General-purpose processors (GPPs) are optimised for standard precisions such as IEEE-754 single/double precision. GPPs can also employ reduced precision via multiple precision software libraries such as MPFR [9]. Multiple standard precision instructions are required to complete a reduced precision computation even if the reduced precision format

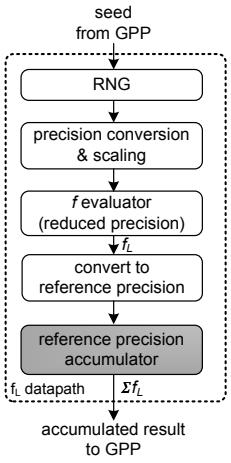

Figure 3: Reduced precision sampling data-path.

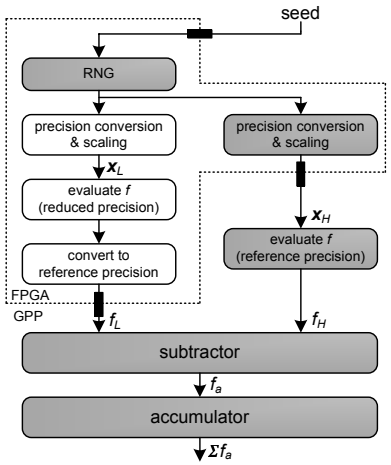

Figure 4: Workload partitioning of the auxiliary sampling. States in reference precision are shaded.

has a smaller wordlength. Hence, it is usually not cost effective to use GPPs for reduced precision computations. On the other hand, FPGA data-paths are customizable. Lower precision are usually preferred over higher precision ones because they usually have higher clock frequency, consume less resources and allow higher degrees of parallelism given the same amount of resources. It is thus better to perform reduced precision computations on the FPGA and leave reference precision computations to the GPP.

Since the sampling of $\langle f_L \rangle_{N_L}$ involves only reduced precision evaluations of f, we assume it is achieved by using reduced precision sampling data-paths on FPGA as shown in figure 3. A seed is fed into the random number generator from the GPP. The random numbers are converted into the reduced precision format and scaled to the sampling domain. Although only a small fraction of bits generated by the RNG are used in reduced precision sampling, we keep the bit-width of the RNG the same as that for reference precision sampling. The scaled random number is then evaluated by the reduced precision sampling function evaluator. The accumulation is performed in reference precision to avoid loss of accuracy due to insufficient dynamic range in the accumulator. Finally, the accumulated result is sent back to the GPP. Multiple reduced precision sampling data-paths

can be used with different seeds, and the averaging of the final results is done in the GPP.

Figure 4 shows the workload partitioning of the auxiliary sampling. It consists of 4 main stages: (1) random number generation, (2) evaluation of the sampling function f in reference and reduced precision, (3) computing the difference e between f_L and f_H in reference precision, and (4) accumulation of the difference. Auxiliary sampling is the process used to estimate the average finite precision error ($\mu_{\epsilon_{fin}}$) between the reduced and the reference precision data-paths under the same set of random inputs. We implement the random number generator using the FPGA and sent results back to the GPP. This method utilises highly efficient RNG generation on FPGAs which are an order of magnitudes faster than GPP based RNGs [16]. The trade-off for this partitioning method is increased bandwidth. For each sample point of the auxiliary sampling, we need to transfer s reference precision random numbers and one reference precision evaluation result from the FPGA to the GPP where s is the dimension of the sampling function.

Problem parameters	
σ_{tol}	output error tolerance, in terms of standard deviation of the output
s	dimension of the sampling function
σ_{f_L}	standard deviation of reduced precision sampling
σ_{f_a}	standard deviation of auxiliary sampling
L	the number of significand bits being used in the reduced precision data-paths
Resource allocation parameters	
$p_L \in \mathbb{Z}$	number of reduced precision sampling data-paths
$p_{aux} \in \mathbb{R}$	**effective** number of auxiliary sampling data-paths
FPGA parameters (for each FPGA)	
A_{total}	total available area
A_{com}	cost of communication infrastructure
R_S	slack ratio
$freq$	clock frequency
c	number of clock cycles to compute a sample point
A_{red}	cost of a reduced precision sampling data-paths as shown in figure 3
A_{aux}	cost of auxiliary sampling data-path
GPP parameters	
T_{aux}	time required to compute a sample point
System parameters	
N_{core}	number of cores in each GPP
N_{gpp}	number of GPPs in the system
N_{fpga}	number of FPGAs in the system
BW_{gpp}	bandwidth between the GPP and I/O the hub (in terms of number of reference precision data / sec)
BW_{fpga}	bandwidth between each FPGA and I/O the hub
Output	
t	time required for the system to get output with specific error tolerance

Table 1: Parameters in our analytical model.

5. MIXED PRECISION OPTIMISATION

In this section, we develop analytical models for determining the required execution time of the proposed mixed precision method on a reconfigurable accelerator system. Figure 5 shows the system architecture for the reconfigurable system in our analytical model. The GPP is connected to an I/O hub (i.e. North Bridge) through a high bandwidth

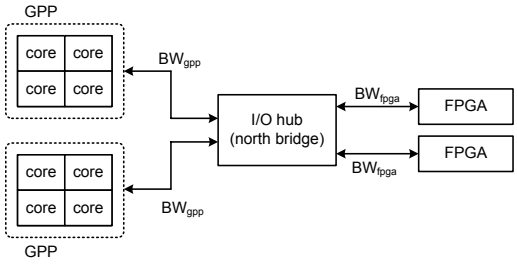

Figure 5: System architecture of the reconfigurable accelerator system in our analytical model.

communication channel such as the Intel QPI or the AMD HyperTransport link. The FPGAs are connected to the I/O hub through another bus, usually PCI express. Thus communication between the GPP and the FPGA has to pass through the two kinds of communication link.

Table 1 shows the parameters in our analytical model. It should be noted that all FPGA cost related parameters such as A_{total} and A_{red} should be applied to every kind of FPGA resource that is involved. For example, there will be 4 different A_{total} parameters for FPGA's look up table (LUT), registers, embedded DSP blocks and block memory respectively. Some other assumptions are made in the model. First, we assume a fixed amount of FPGA resources is used for the communication infrastructure between the FPGA and the I/O hub. Second, we assume the entire FPGA is running at a single clock frequency. Finally, we assume that a certain percentage of the FPGA's resource (the slack ratio) is left intentionally unused to avoid over-congestion in placement and routing.

Since the aggregated throughput of the auxiliary sampling on GPP does not always match the throughput of an auxiliary sampling data-path on the FPGA, we assume the effective number of auxiliary sampling data-paths can take fractional values. For example, $p_{aux} = 0.75$ means there is one auxiliary sampling data-path on the FPGA but only 75% of its outputs are computed by the GPP. The remaining 25% of the outputs are discarded.

Let TH_{red} and TH_{aux} be the aggregated throughput of reduced precision sampling and auxiliary sampling of the entire system. Using Equation 15, the required execution time for the system to produce an output with error equal to σ_{tol} can be found by Equation 16:

$$\sigma_{tol}^2 = \frac{\sigma_{f_L}^2}{t \times TH_{red}} + \frac{\sigma_{f_a}^2}{t \times TH_{aux}} \implies$$
$$t = \frac{\sigma_{f_L}^2}{\sigma_{tol}^2 \times TH_{red}} + \frac{\sigma_{f_a}^2}{\sigma_{tol}^2 \times TH_{aux}} \quad (16)$$

The aggregated throughput of the reduced precision sampling and the auxiliary sampling of all FPGAs can be modelled as:

$$TH_{red} = N_{fpga} \times p_L \times freq/c$$
$$TH_{aux} = N_{fpga} \times p_{aux} \times freq/c \quad (17)$$

The execution time for the mixed precision methodology is:

$$t(p_L, p_{aux}) = \frac{c}{\sigma_{tol}^2 \times N_{fpga} \times freq} \times \left(\frac{\sigma_{f_L}^2}{p_L} + \frac{\sigma_{f_a}^2}{p_{aux}} \right) \quad (18)$$

The following constraint should be applied to ensure the architecture described by the resource allocation parameters

can fit within the FPGA. We round p_{aux} to the next larger integer. The constraint (19) is transformed into two new constraints (20-21) using a new integer variable p_{aux_i} to avoid the ceiling function:

$$p_L \times A_{red} + \lceil p_{aux} \rceil \times A_{aux} \leq A_{total} \times (1 - R_S) - A_{com} \quad (19)$$

$$p_L \times A_{red} + p_{aux\_i} \times A_{aux} \leq A_{total} \times (1 - R_S) - A_{com} \quad (20)$$

$$p_{aux\_i}^{-1} \times p_{aux} \leq 1 \quad (21)$$

The number of auxiliary samplings that each GPP can perform is N_{core}/T_{aux} and the aggregated throughput of all GPPs is $N_{gpp} \times N_{cores}/T_{aux}$. Hence the effective number of auxiliary sampling data-paths on each FPGA is constrained by the following equation:

$$N_{fpga} \times p_{aux} \times freq/c \leq N_{gpp} \times N_{core}/T_{aux} \quad (22)$$

One evaluated value of f and s random numbers must be sent every cycle to the GPP to complete the subtraction and accumulation for each auxiliary sampling, hence the bandwidth constraints are:

$$p_{aux} \times freq/c \times (s+1) \leq BW_{fpga} \quad (23)$$

$$N_{core}/T_{aux} \times (s+1) \leq BW_{gpp} \quad (24)$$

The optimal resource allocation among the reduced precision sampling and the auxiliary sampling can be found by applying the following optimisation:

$$\min_{p_L \in \mathbb{Z}, p_{aux} \in \mathbb{R}, p_{aux\_i} \in \mathbb{Z}} t(p_L, p_{aux}, p_{aux\_i})$$

$$s.t. \text{ constraints (20)-(24) are satisfied}$$

Since the objective function $t(p_L, p_{aux})$ and all the constraints are posynomial, the optimisation can be solved using mixed integer geometric programming (MIGP) [2]. The globally optimal p_L, p_{aux} values and the optimal precision can be found using enumeration from Algorithm 1, where L_{min} and L_{max} are the minimum and maximum choice of reduced precision in the system respectively.

Algorithm 1 Enumeration process for optimal reduced precision and optimal resource allocation.

```
1: t_global ← huge_value
2: for L = L_min → L_max do
3:    apply MIGP on t(p_L, p_aux) for the minimum execution
      time t_min in precision L
4:    if t_min < t_global then
5:       t_global = t_min
6:       p_aux(global) = p_aux, p_L(global) = p_L
7:    end if
8: end for
```

6. CASE STUDIES

6.1 Asian option pricing

The first case study for our mixed precision methodology is an arithmetic mean Asian call option pricing problem. An Asian call option is characterised by S_0, the current price of the underlying asset; K, the strike price; T, time to maturity and *steps*, the number of observation points to maturity. The arithmetic mean of the asset's current price and the prices at all the observed points computed. At maturity, if the mean is larger than the strike price, the option pays the owner the mean less the strike price. Otherwise, if the strike price is larger, the payoff of the option is zero. Unlike the geometric mean Asian option pricing problem, there is no closed form analytical solution for this problem and Monte Carlo simulation is a common way for pricing this option.

Algorithm 2 shows the sampling function for pricing Asian options using the Black-Scholes model. Since the intermediate variables $drift$ and $vsqrtdt$ are the same for every sample point, they are pre-computed to reduce computation workload. We do not compute the actual arithmetic mean in each sample point. Instead, the strike price is multiplied by $(steps + 1)$ and it is compared with the sum of the prices. This optimisation removes the division operation from the sampling function as it is expensive to implement. We use a fully pipelined design and it takes on average $(steps + 1)$ clock cycles for the FPGA to complete a single sampling of the Asian option problem.

Algorithm 2 Sampling function for the Asian call option pricing problem

Input: S_0 = current price of the underlying asset, K = strike price of the option, v = volatility of the underlying asset, r = interest rate, $steps$ = number of time step, δt = time period between two time steps
$W \sim \mathcal{N}(0,1)$ Gaussian random number
Output: p_steps = payoff of the option multiplied by $(steps + 1)$

```
1: drift ← (r − v²/2)δt, vsqrtdt ← v√δt
2: S_i ← S_0, S_sum ← S_0
3: for i = 1 → steps do
4:    S_i ← S_{i−1} × exp(drift + vsqrtdt × W)
5:    S_sum ← S_sum + S_i
6: end for
7: p_steps ← max(0, S_sum − K × (steps + 1))
```

6.2 The GARCH volatility model

Our second case study is for pricing of a fixed strike lookback call option under the GARCH model. This option pays the owner $max(S_{ceil} - K)$ at maturity, where K is the strike price and S_{ceil} is the maximum day closing price of the underlying asset within the lifetime of the option.

In the original Black-Scholes model, the volatility of an asset is assumed to be constant. However, this assumption may not be realistic. A solution is to employ a stochastic volatility model such as the generalised autoregressive conditional heteroskedasticity (GARCH) model proposed by Bollerslev [1]. We use the common GARCH(1,1) model where the volatility of the asset v_i at time step i can be modelled. Let α and β be pre-calibrated model constants, v_0 be the volatility at the start time and λ be a random number following a $\mathcal{N}(0,1)$ distribution.

$$\begin{aligned} v_i^2 &= v_0 + \alpha v_{i-1}^2 + \beta v_{i-1}^2 \lambda^2 \\ &= v_0 + v_{i-1}^2(\alpha + \beta\lambda^2) \end{aligned} \quad (25)$$

The implementation of lookback option pricing is similar to Asian option pricing, except that $drift$ and $vsqrtdt$ are updated every time step according to Equation 25. An additional random number source is also required.

6.3 Collateralized Mortgage Obligation

Our third case study concerns pricing Collateralized Mortgage Obligation (CMO) [15]. A CMO is a security which

generates cashflow to the owner from interest and prepayments from a pool of mortgages. The actual payoff of CMO depends on the classes of the CMO, commonly referred to as tranches, and a set of pre-specified rules [6]. We adopt the algorithm in [15] in our FPGA design. An arctan function is required in each random walk of the interest rate. This function is often not efficiently implemented in GPP maths libraries On the other hand, lookup table based function evaluation is efficient on FPGA, especially when the required precision is low. Algorithm 3 shows the sampling function of the CMO pricing problem. The variables can be found in Table 2.

u_k	discount factor for month k
m_k	cash flow for month k
i_k	interest rate for month k
w_k	fraction of remaining mortgages prepaying in month k
r_k	fraction of remaining mortgages at month k
c_k	(remaining annuity at month k) / c
c	monthly payment

Table 2: Variables in the sampling function of CMO pricing problem.

Algorithm 3 Sampling function for the CMO pricing problem

Input: c = monthly payment, K_0, K_1, K_2, K_3, K_4 = constants of the model, I_0 = initial interest rate, M = length of the mortgages, σ = standard deviation of interest rate
Pre-computed constants $c_k = \sum_{j=0}^{M-k}(1+I_0)^{-j}$, remaining annuity at month k) / c
$W \sim \mathcal{N}(0,1)$ Gaussian random number
Output:
PV = present value of the security

1: $sum \leftarrow 0$, $i_0 \leftarrow K_0 \times I_0$
2: $r_1 \leftarrow 1$, $u_1 \leftarrow (1+I_0)^{-1}$
3: **for** $k = 1 \rightarrow M$ **do**
4: $\quad i_k \leftarrow K_0 \times exp(\sigma W) \times i_{k-1}$
5: $\quad w_k \leftarrow K_1 + K_2 \times arctan(K_3 \times i_k + K_4)$
6: $\quad$ **if** $k \geq 2$ **then**
7: $\quad\quad r_k \leftarrow r_{k-1} \times (1 - w_k)$
8: $\quad\quad u_k \leftarrow u_{k-1} \times (1 + i_{k-1})^{-1}$
9: $\quad$ **end if**
10: $\quad m_k \leftarrow c \times r_k((1 - w_k) + w_k \times c_k)$
11: $\quad sum \leftarrow sum + u_k \times m_k$
12: **end for**
13: $PV \leftarrow sum$

6.4 Numerical integration

Our last case study is multi-dimensional integral evaluation using the Monte Carlo integration method. Multi-dimensional integrals arise in many areas such as engineering, biology, chemistry and physics modellings and they are not always solvable with analytical methods. Equation 26 shows a multi-dimensional integration where a_i and b_i are the lower and upper bounds of the integration domain of the i^{th} dimension. To evaluate the integral using Monte Carlo simulation, random input vectors are generated within the integration domain and the average value of the integration function f is sampled. The approximated value for the integral can then be found by multiplying the average with the hypercube V of the integration domain as shown in Equation 28. The Monte Carlo integration method is preferable over quadrature based integration methods for high dimensional integrals, because MC integrations always converge with a rate of $\mathcal{O}(1/\sqrt{N})$ and the complexity of MC integration does not increase exponentially as quadrature based numerical integration methods. We refer to [3] for an extensive introduction of Monte Carlo integration method.

$$I = \int_{a_1}^{b_1} dx_1 \int_{a_2}^{b_2} dx_2 \cdots \int_{a_n}^{b_n} dx_n f(x_1, x_2, \cdots x_n) \quad (26)$$

$$I \sim V \times \langle f \rangle \quad (27)$$

$$V = \prod_{i=1}^{n}(b_i - a_i) \quad (28)$$

We use Genz's "Discontinuous" multi-dimensional integral in this case study (29). This is a common test integral used in evaluation of different numerical integration methods. In our tests we use $n = 8$ as the dimension and an integration domain $[0, 1)^8$. Fully parallelised designs are used in our FPGA implementations and the data-paths can compute a single sample point per clock cycle, with constants c_i and w_i:

$$f_{dis} = \begin{cases} 0 & \text{if } x_0 > w_0 \text{ or } x_1 > w_1 \\ exp(\sum_{i=1}^{n}(c_i \times x_i)) & \text{otherwise} \end{cases} \quad (29)$$

7. EVALUATION

7.1 Reconfigurable accelerator system

We use the MaxWorkstation reconfigurable accelerator system from Maxeler Technologies for our evaluation. It has a MAX3424A card with a Xilinx Virtex-6 SX475T FPGA. The card is connected to an Intel i7-870 CPU through a PCI express link with a measured bandwidth of 2 GB/s. The Intel GPP has 4 physical cores.

	current	I	II	III
N_{gpp}	1	8	1	1
N_{fpga}	1	1	4	8
$BW_{gpp}(GB/s)$	2	2	4×2	8×2
$BW_{fpga}(GB/s)$	2	8×2	2	2

Table 3: Parameters of the current system and other hypothetical systems.

An important advantage of having an analytical model for our mixed precision methodology is that system designers can predict the performance of a hypothetical system based on parameters of the current system and the analytical model. Table 3 shows the parameters for our current system and three hypothetical systems. The hypothetical systems are constructed in such a way that the aggregated computational power of the FPGAs or the GPPs are 4 or 8 times higher than the current system, and the bandwidth is scaled proportionally.

The Intel Compiler (ICC) and the Intel Math Kernel Library are used in our software implementations. We use the SFMT random number generator and the Box-Muller transformation in the Intel Vector Statistical Library (VSL) for the random number generation. Every effort has been made to ensure the software implementations are optimised, and

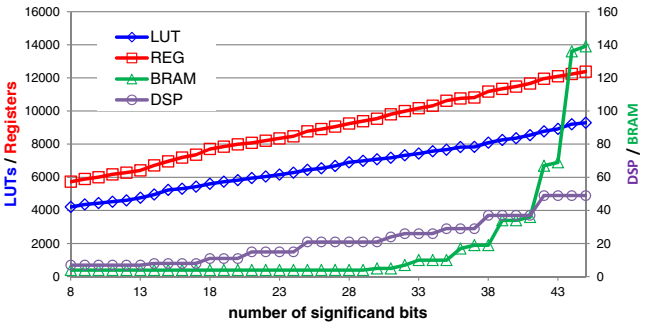

Figure 6: Cost of reduced precision sampling data-paths of the Asian option problem.

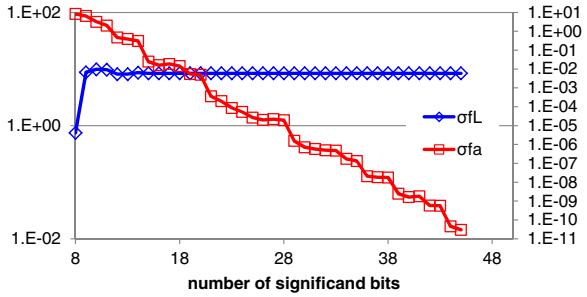

Figure 7: The standard deviations of the reduced precision sampling and the auxiliary sampling verses different precisions.

the comparisons are fair and accurate. For the FPGA implementations, we use the MaxCompiler as our development system. This adopts a streaming programming model similar to [13] and supports customisable data formats so that floating point can be exploited with different precisions. All the FPGA results reported in this paper are post place and route results.

The error tolerance σ_{tol} of the three financial case studies is set to 2.5e-3 such that 99.99% (4σ) of the time the error is less than a cent, given that the pricing is in dollars. For the numerical integration case study, the tolerance is set to 2.5e-4 since most scientific applications require high accuracy.

7.2 Applying optimisation

There are a few steps to apply Algorithm 1 in Section 5 in order to use the proposed mixed precision methodology optimally on a reconfigurable accelerator system. The first step is to find the system parameters such as N_{core}, N_{gpp}, N_{fpga}, BW_{gpp} and BW_{fpga}. These parameters can usually be found in the specification of the reconfigurable accelerator system.

The second step is to collect application specific FPGA and GPP parameters. In the MaxCompiler system, we describe the precision of the entire sampling data-path using a global variable and scripts are used to automatically generate data-paths with varying number of significand bits. Figure 6 shows the place and routed result of reduced precision data-paths of the Asian option problem. It is clearly shown that all the resource requirement increase with precision. Moreover, due to the function approximator in the exponential function, the block memory usage increases exponentially with the precision. The figure shows the A_{red} parameters of different precisions used in our model. Other FPGA parameters such as the A_{aux} parameters can be found using a similar method. The cost of communication infrastructure A_{com} is assumed to be constant. We also estimate the GPP parameter T_{aux} by writing a software benchmark program, which implements the data-flow in Figure 4c for certain iterations, and the average time required for an iteration is used as T_{aux}.

The next step is the estimation of the standard deviations for reduced precision sampling, σ_{f_L}, and for auxiliary sampling, σ_{f_a}. An FPGA bit-stream with auxiliary sampling data-paths of different precisions is loaded and the results of the sampling function evaluations in different reduced precisions are sent back to the host PC. Using these results and the reference precision sampling function evaluations result from the GPP, the two standard deviations can be estimated using the two-pass algorithm on the host PC [20]. Figure 7 shows how the two standard deviations change with different precisions in the Asian option problem, using the parameters ($S_0 = K = 100, T = 1, v = 0.2, r = 0.05, steps = 360$).

It is interesting to note that the standard deviation of the auxiliary sampling σ_{f_a} decreases exponentially with increasing precision. However the standard deviation of the reduced precision sampling σ_{f_L} is low when the precision is low and increases rapidly to reach a constant maximum value with further increases in precision. The same pattern is also observed in the standard deviations of other case studies. A possible explanation is that when the reduced precision is low, different values are compressed to the same numerical representation and hence the standard deviation is reduced. The standard deviation grows with reduced precision because there are more possible representations, and will finally converge to a value where the value is the same as the standard deviation of the reference precision sampling (i.e. σ_{f_H}). The observed exponential reduction of σ_{f_a} could be explained by the fact that finite precision error decreases exponentially with the number of significand bits in floating point formats.

Using parameters collected in the previous steps, we can apply geometric programming to find the optimal precision and resource allocation. A major assumption of this flow is that the two standard deviations do not change with input parameters (e.g. strike price of an option). If this assumption does not hold, we can profile the common σ_{f_L} and σ_{f_a} combinations and generate an optimal bit-stream for each of these combinations. When the input parameters change, we profile the two standard deviations again, run the geometric programming solver and load the bit-stream closest to the optimal configuration.

It is important to note that the choice of error tolerance σ_{tol} affects the execution time of our mixed precision methodology as shown in Equation 16. However, the optimal reduced precision and resource allocation do not change with the error tolerance. Hence there is no need to rerun the geometric programming.

7.3 Performance: parallelism versus precision

Figure 8 shows the execution time and the degree of parallelism of the Asian option pricing problem for different reduced precision in the current system as evaluated by our analytical model. The optimal reduced precision in this benchmark is s12e8. The performance curve and the optimal point can be explained by considering Figures 6 and 7.

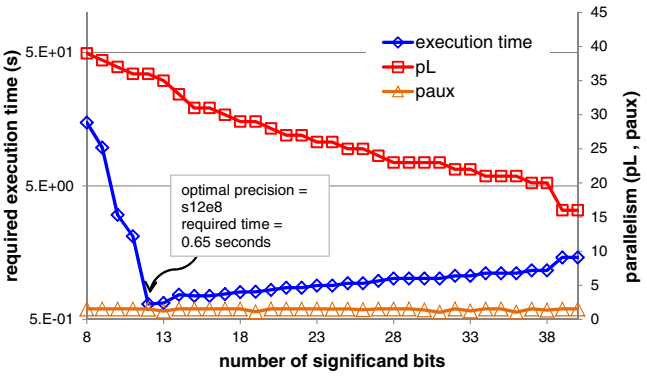

Figure 8: Results of Asian option pricing versus different number of significand bits.

If the reduced precision is lower than the optimal one, the auxiliary sampling error σ_{f_a} is high, and more computations must be included. This will take a longer time even though parallelism is increased due to smaller data-paths. If the reduced precision is higher than the optimal one, the decrease of the auxiliary sampling error is marginal and cannot offset the disadvantage of reduced parallelism.

	current	I	II	III
execution time (s)	0.65	0.65	0.19	0.09
optimal precision	s12e8	s12e8	s15e8	s16e8
p_L/p_{aux}	23.9	13	73.9	147.9

Table 4: Execution time, optimal reduced precision and the p_L/p_{aux} ratio of the same Asian option pricing under different system parameters.

We also investigate the relationship between the optimal reduced precision and system parameters. Table 4 shows that when the aggregated FPGA computational power is increased (II and III), the optimal reduced precision will increase because the system can perform more reduced precision sampling (i.e. higher p_L/p_{aux} ratio), and thus afford a higher σ_{f_L}. Investing in higher aggregated GPP computational power (I) seems to have a marginal effect in this benchmark as the sampling error in reduced precision already dominates the sampling error in the auxiliary sampling.

7.4 Comparison: GPP/FPGA double precision

Table 5 shows comparisons of the 4 MC case studies running on a GPP only system with double precision arithmetic, an FPGA only system with double precision arithmetic, and a reconfigurable accelerator system with our proposed mixed precision methodology. All designs are run for a specific time so that the 3 systems have the same accuracy. As shown in the table, the mixed precision methodology requires 5-11 % additional sample function evaluations but only 1-4 % of total evaluations are computed in reference precision. This clearly shows the trade-offs between number of computations and the contribution of each computation in increasing the accuracy of the final result. Using the mixed precision methodology, we achieve 2.9 to 7.1 times speedup over the double precision FPGA designs and 44 to 163 times speed up over the **quad-core** GPP designs.

We also compare the energy efficiency of the three settings. The average power consumption is measured using a remote power measuring socket from Oslon® electronics with an measuring interval of 1 second. As shown in Table 5, although the mixed precision designs using both the FPGA and the GPPs have the highest power consumption compared with the GPP only or the FPGA only settings, they consume the least total energy to achieve the required accuracy because the execution times are significantly reduced thanks to our technique for workload partitioning. Our mixed precision methodology achieves 1.4 to 3.1 times energy saving compared with the FPGA only designs with double precision, and 41 to 170 times energy saving compared with GPP only designs, while meeting the same output accuracy requirement.

7.5 Comparison: GPU

We also compare our mixed precision methodology on a reconfigurable accelerator system with a graphics processing unit (GPU). Table 6 compares the execution time and power consumption of our mixed precision methodology with an NVIDIA Tesla C2070 GPU for pricing Asian options. The GPU has 448 cores running at 1.15 GHz. Both the Tesla 2070 GPU and the Virtex-6 FPGA are fabricated using 40nm technology. We use the same GPU Asian option pricing design as described in our previous work [19]. Since random number generation and pricing calculation take place on the GPU, communication is not a bottleneck as only the final accumulated result in double-precision need to be sent from the GPU to the GPP. Using our mixed precision methodology, an Virtex-6 ST475X FPGA and an i7-870 GPP are able to out-perform the GPU by 4.6 times. We also achieved 5.5 times energy saving compared with the GPU.

	GPP only	GPU	FPGA only	FPGA + GPP
precision	double	double	double	mixed
execution time (s)	29	3	4.7	0.65
power (W)	183	236	85	192
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

Table 6: Comparison with GPP and GPU.

8. CONCLUSION

This paper proposes a novel mixed precision methodology for Monte Carlo simulation in reconfigurable accelerator systems. The technique is applicable to any Monte Carlo application and exploits the synergy between FPGA and GPP to produce results of the desired accuracy. An analytical model and optimisation method is developed for locating the optimal precision and optimal resource allocation. Experimental results on four realistic case studies show that auxiliary sampling would only require 5 % to 11 % additional evaluations, and less than 4 % of total evaluations are computed in the reference precision (Table 5). We demonstrate that reconfigurable accelerator system using our methodology can be up to 4.6 times faster than state-of-the-art GPU, 7.1 times faster than a baseline FPGA design using double precision, and 163 times faster than optimised software running on a quad-core GPP. It can also be up to 5.5 times more energy efficient than a GPU and 170 times more energy efficient than a quad-core software implementation.

	Asian option			GARCH			CMO			Numerical integration		
	SW	FP	Mixed	SW	FP	Mixed	SW	FP	Mixed	SW	FP	Mixed
clock freq. (GHz)	2.93	0.175	0.175[1]	2.93	0.175	0.175[1]	2.93	0.175	0.175[1]	2.93	0.175	0.16[1]
num. of cores[2]	4	5	36/1.5	4	5	24/0.9	4	5	20/0.65	4	5	16/0.18
num. of f_L evaluations (M)	0	0	12	0	0	321	0	0	7.2	0	0	2320
num. of f_H evaluations (M)	11.3	11.3	0.47	317	317	11.6	6.75	6.75	0.23	2230	2230	26.8
num. of total evaluations (M)	11.3	11.3	12.5	317	317	333	6.75	6.75	7.43	2230	2230	2347
additional evaluation (%)	-	-	10.6	-	-	4.8	-	-	10	-	-	5.2
evaluations in reference precision (%)	100	100	3.8	100	100	3.5	100	100	3.1	100	100	1.1
execution time (sec.)	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
power consumption (W) [3]	183	85	192	179	90	181	175	94	171	184	90	189
energy consumption (kJ)[4]	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

[1] Only the FPGA clock frequencies are reported and the 4 GPP cores are all running at 2.93 GHz.
[2] For the mixed precision design, all the 4 GPP cores are used and the number of reduced precision sampling and auxiliary sampling data-paths (pL/p_{aux}) are shown.
[3] The idle power consumption of the system is $80W$.
[4] Energy consumption = power consumption × execution time.
[5] The optimal precision of the 4 mixed precision designs is **s12e8**.

Table 5: Comparison of MC simulations using GPP only system (SW), double precision FPGA only system (FP) and mixed precision methodology using both GPP and FPGA (Mixed).

Future work includes applying the proposed methodology to other sampling methods such as the Quasi-Monte Carlo methods. Other directions of further research involve extending the proposed methodology to cover heterogeneous systems consisted of GPPs, FPGAs and GPUs, and automating the steps of the methodology.

Acknowledgments

The research leading to these results has received funding from the Croucher Foundation, Maxeler Technologies, Xilinx, the UK EPSRC, and the European Union Seventh Framework Programme under grant agreements number 257906 and 248976.

9. REFERENCES

[1] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[2] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8:67–127, 2007.

[3] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

[4] G. Chow, K. Kwok, W. Luk, and P. Leong. Mixed precision processing in reconfigurable systems. In *Proc. FCCM*, pages 17 –24, 2011.

[5] G. A. Constantinides. Word-length optimization for differentiable nonlinear systems. *ACM Trans. Des. Autom. Electron. Syst.*, 11:26–43, 2006.

[6] F. J. Fabozzi. *The Handbook of Mortgage-Backed Securities*. McGraw-Hill, 2005.

[7] C. Fang, T. Chen, and R. Rutenbar. Floating-point error analysis based on affine arithmetic. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–561–4, 2003.

[8] G. S. Fishman. *Monte Carlo Concepts, Algorithms, and Applications*. Springer, 1995.

[9] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33, 2007.

[10] M. Gokhale, J. Frigo, C. Ahrens, and R. Minnich. Monte Carlo radiative heat transfer simulation on a reconfigurable computer. In *Proc. FPL*, pages 95–104, 2004.

[11] A. Gothandaraman, G. D. Peterson, G. Warren, R. J. Hinde, and R. J. Harrison. FPGA acceleration of a quantum Monte Carlo application. *Parallel Computing*, 34(4-5):278 – 291, 2008.

[12] A. Kaganov, A. Lakhany, and P. Chow. FPGA acceleration of multifactor CDO pricing. *ACM Trans. Reconfigurable Technol. Syst.*, 4:20:1–20:17, 2011.

[13] O. Mencer. ASC: a stream compiler for computing with FPGAs. *IEEE Trans. CAD*, 25(9):1603 –1617, 2006.

[14] R. E. Moore. *Interval arithmetic and automatic error analysis in digital computing*. PhD thesis, Stanford University, 1963.

[15] S. H. Paskov. New methodologies for valuing derivatives. In M. Dempster and S. Pliska, editors, *Mathematics of derivative securities*, number 15 in Publications of the Newton Institute. Cambridge Univ. Press, 1997.

[16] D. B. Thomas, L. Howes, and W. Luk. A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. In *Proc. FPGA*, pages 63–72, 2009.

[17] X. Tian and K. Benkrid. Design and implementation of a high performance financial Monte-Carlo simulation engine on an FPGA supercomputer. In *Proc. ICFPT*, pages 81–88, 2008.

[18] X. Tian and C.-S. Bouganis. A run-time adaptive FPGA architecture for Monte Carlo simulations. In *Proc. FPL*, 2011.

[19] A. H. Tse, D. B. Thomas, K. H. Tsoi, and W. Luk. Efficient reconfigurable design for pricing Asian options. *SIGARCH Comput. Archit. News*, 38:14–20, 2011.

[20] E. W. Weisstein. Sample variance computation. `http://mathworld.wolfram.com/SampleVarianceComputation.html`, accessed Sept. 2011.

[21] G. Zhang et al. Reconfigurable acceleration for Monte Carlo based financial simulation. In *Proc. ICFPT*, pages 215 –222, 2005.