# Mixed Precision Comparison in Reconfigurable Systems

Gary C.T. Chow, K.W. Kwok and Wayne Luk
*Department of Computing*
*Imperial College London*
*London, United Kingdom*
*Email: {cchow, kkwok, wl}@doc.ic.ac.uk*

Philip Leong
*School of Electrical and Information Engineering*
*University of Sydney*
*Sydney, Australia*
*Email: philip.leong@sydney.edu.au*

***Abstract*—Customisable data formats provide an opportunity for exploring trade-offs in accuracy and performance of reconfigurable systems. This paper introduces a novel methodology for mixed-precision comparison, which improves comparison performance by using reduced-precision data-paths while maintaining accuracy by using high-precision data-paths. Our methodology adopts reduced-precision data-paths for preliminary comparison, and high-precision data-paths when the accuracy for preliminary comparison is insufficient. We develop an analytical model for performance estimation of the proposed mixed-precision methodology. Optimisation based on integer linear programming is employed for determining the optimal precision and resource allocation for each of the data-paths. The effectiveness of our approach is evaluated using a common collision detection problem. Performance gains of 4 to 7.3 times are obtained over baseline fixed-precision designs for the same FPGAs. With the help of the proposed mixed-precision methodology, our FPGA designs are 15.4 to 16.7 times faster than software running on multi-core CPUs with the same technology.**

***Keywords*-mixed precision, reconfigurable system.**

## I. Introduction

FPGAs have made rapid advances. Exciting developments include devices comprising nearly a million LUTs, thousands of high performance embedded multipliers, on-chip memory blocks, and I/O transceivers which provide bandwidth exceeding one Tera bits per second. These capable and abundant resources, combined with the flexibility provided by the reconfigurable logic blocks and routing network, can realise customised data-paths which significantly outperform high-end CPUs. In fact it has been shown that the peak floating-point performance of FPGAs has already surpassed that of CPUs [1].

This paper introduces a novel methodology for utilising reduced-precision data formats in numerical function comparisons. A numerical function comparison is defined as the difference between two functions $f(\mathbf{x})$ and $g(\mathbf{y})$, where $f$ and $g$ are two numerical functions and $\mathbf{x}$ and $\mathbf{y}$ are their input vectors. Such comparisons exist in many high performance applications. In clustering algorithms such as K-means clustering and EM clustering [2], [3], function comparisons are used to assign data points to their nearest cluster. In collision detection algorithms, function comparisons are performed to check the overlappings between objects [4]. In numerical optimisation algorithms such as Nelder-Mead simplex method and simulated annealing method, function comparisons are necessary to guide the searches [5], [6]. Any methodology enhancing the performance of FPGAs' function comparison will significantly improve these applications.

The flexibility of FPGAs provides trade-offs between performance and accuracy in the implementation of function comparison data-paths. Reduced-precision data-paths usually consume less logic resource, require lower I/O bandwidth and have higher clock frequencies at expense of lower comparison accuracy. For a given area, more parallelism may be achieved by using reduced-precision data-paths instead of high-precision ones. When accurate comparisons are required, high-precision data-paths with lower performance are unavoidable. Our mixed-precision methodology has the advantages of reduced-precision data-paths without compromising the comparison accuracy. Reduced-precision data formats are used for preliminary comparisons, and high-precision data-paths are used to re-compute the comparison when the accuracy of preliminary comparison is found to be insufficient.

The major challenges of applying a mixed-precision methodology are to decide which precision to be used in the reduced-precision data-paths, and to allocate resources among the reduced-precision and reference-precision data-paths. We refer these problems as the optimal precision and optimal resource allocation problems respectively. Using reduced-precision will increase the degree of parallelism as well as improve their performance. However, it will also decrease their accuracies and a higher percentage of comparisons will require corresponding re-computations using the reference-precision data-paths. By adjusting the precision, we can redistribute workloads between the reduced-precision and reference-precision data-paths. In a reconfigurable system, we can also adjust the degree of parallelism of the reduced-precision and reference-precision data-paths to match their workloads. To this end, we develop an analytical model that is used to solve these problems.

The major contributions of this work are:

- A novel mixed-precision methodology for numerical function comparison based on analysis of error information in reduced-precision data format (Section III).
- An analytical model of mixed-precision function comparison systems for determining the optimal precision

and resource allocation for reduced-precision and high-precision data-paths (Section IV).

- A demonstration of the effectiveness of the proposed methodology by mapping a common collision detection algorithm to reconfigurable systems. A performance gain of 4 to 7.3 times is obtained over the baseline fixed precision designs on the same FPGAs (Section V).

## II. Related work

Previous work can be classified into two categories. They are bit-width optimisation techniques and mixed-precision linear algebra methods.

Bit-width optimisation techniques aim to improve performance by using minimum precision in a data-path given a required output accuaracy. These techniques are applicable to any application involving numerical computation. A common approach is to develop an accuracy model which relates output accuracy with the precisions of the data formats being used in the data-path. The performance and area of data-paths with different precisions both have to be modelled. By combining the two models, one can search for the design with a sufficient accuracy and a minimum area-delay product. Common accuracy modelling approaches include simulation approach [7], interval arithmetic [8], backward propagation analysis [9], affine arithmetic [10], [11], [12], [13], SAT-Modulo theory [14] and the polynomial algebraic approach [15]. The search for minimum area-delay product usually involves a non-convex integer programming problem with number of variables proportional to the complexity of the data-path, an exhaustive search is usually infeasible and thus the result is not guaranteed to be optimal. Although the polynomial algebraic approach reduced the problem to a convex integer programming problem [15]. However, an exhaustive search is still infeasible because the variables can only take integer values. Thus, bit-width optimisation of data-paths remains an open on-going research problem.

The major issue of bit-width optimisation techniques is the uncorrectability of the approach. A reduction of precision in any stage within a data-path will result in a loses in output accuracy. The introduced error is not correctable by performing additional computations. Thus high-precision data formats are required for accurate outputs. The benefit of employing bit-width optimisation is usually a 20 to 40 % reduction in area-delay product [9], [10], [11], [12], [15].

Mixed-precision linear algebra methods use reduced-precision data formats for computations. They are only applicable to iterative algorithms which can correct previous errors, such as iterative refinement [16], [17], iterative Jacobi solver [18] and conjugate gradient method (CG) [19]. Some work is reported in an attempt to utilise reduced-precision computations as much as possible, and the high-precision computations are only adopted to correct errors [16]. In [17], [19], high-precision computations are completely avoided and accurate results are achieved by running more iterations. Mixed precision linear algebra methods can usually achieve good performance due aggressive reduction in precision. For example Lopes et al. observed speedup of 14 to 36 times compared with a high-end CPU using a reduced-precision FPGA CG solver [19]. However, these methods are limited to linear iterative algorithms and are not applicable to other applications.

Our mixed precision methodology has similar advantages to both categories mentioned above. First, it is a generalised method and is applicable to any numerical function comparison. Second, re-computations are performed to correct the error introduced in previous reduced-precision computations. Thus we can use reduced-precision data formats without compromising the comparison accuracy.

## III. Mixed-precision methodology

In this section, we first define the accuracy of a numerical function comparison to illustrate the principle of our novel mixed precision methodology.

Referring to the distinction between precision and accuracy in [20], precision is the degree of correctness of each atomic computation and accuracy is the degree of correctness of the final computation result. In the context of numerical function comparison, precision is the correctness of each atomic computation in the data-path (i.e. cores) and comparison accuracy is the correctness of the comparison result.

A function comparison can be written as a subtraction of two functions (i.e. $D = f(\mathbf{x}) - g(\mathbf{y})$). Only the sign of the difference $D$ is of interest. When finite precison arithmetic is used for the computation of $D$, we might have a flipped sign compared with the true value of $D$. We use the following definitions in comparing the accuracy of two different function comparaison systems with different implementations.

**Definition 1.** *A correct comparison result is defined as the result computed without finite precision error.*

**Definition 2.** *Let $X_A$ and $X_B$ be the sets of inputs for which function comparison systems $A$ and $B$ give correct comparison results respectively. System $A$ is (at least) as accurate as system $B$ if $X_B \subset X_A$.*

Using the definitions, we can relate the comparison accuracy of a mixed precision system to the accuracy of another system with a well-defined precision, such as a system with IEEE double precision data format. [1]
A mixed precision comparison proceeds in the following steps:

1) Evaluate the two functions and compute their difference using a reduced-precision data format. (i.e. compute $D_L = f_L(\mathbf{x}) - g_L(\mathbf{y})$)

[1]Throughout the paper, we use the subscripts $L$, $H$ and $T$ to denote quantities computed with the reduced precision arithmetic, the reference-precision arithmetic and the true value of the quantities respectively.

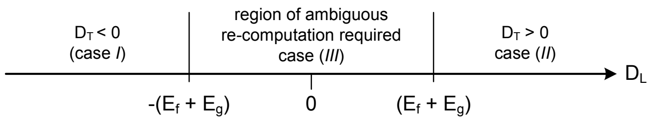


Figure 1. Three difference decision regions of reduced precision comparison.

2) Estimate the maximum and minimum values of the true value of the difference, i.e. $min(D_T)$ and $max(D_T)$, using $D_L$ and the pre-estimated error bounds $E_f$ and $E_g$. The minimum and maximum are computed as follows, where $E_f(\mathbf{x}, L)$ is the absolute error of $f_L$.

$$min(D_T) = D_L - E_f(\mathbf{x}, L) - E_g(\mathbf{y}, L) \quad (1)$$
$$max(D_T) = D_L + E_f(\mathbf{x}, L) + E_g(\mathbf{y}, L) \quad (2)$$

The absolute error $E_f(\mathbf{x}, L)$ is approximated based on the relative error $RE_f$:

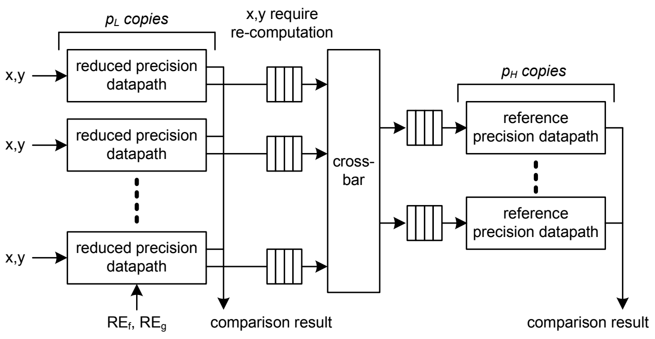
$$E_f = f_T \times RE_f(L) \approx f_L \times RE_f(L) \quad (3)$$

The relative error of a function is defined below, where $f_T(\mathbf{x})$ is the true value of $f$ and $\mathbf{x}$ is any valid input for $f$. Ways for acquiring these relative errors will be discussed in next section.

$$RE_f = max(|\frac{f_L(\mathbf{x}) - f_T(\mathbf{x})}{f_T(\mathbf{x})}|) \quad (4)$$

3) Return a comparison result or re-compute the comparison in reference-precision according to the minimum and maximum values. There are 3 cases:
case I: $min(D_T) > 0$, return $D_T > 0$
case II: $max(D_T) < 0$, return $D_T < 0$
case III: otherwise re-evaluate the two functions and compute their difference with the reference precision $H$ then return sign of $D_H$.

In case I and case II, the difference between functions $f$ and $g$ is large enough to distinguish the sign of $D$ even in the presence of errors introduced by reduced-precision computations. The reduced-precision comparison gives exactly the same comparison result as the reference precision data-paths in these cases. In case III, the difference is small compared with the uncertainty introduced. Hence, we re-evaluate the two functions and re-compute their difference in the reference-precision $H$ to ensure we achieve the same comparison result as $H$. Figure 1 shows the three decision regions of a mixed precision comparison. The mixed precision methodology gives exactly the same comparison as a comparison system with a reference precision $H$ in all the three cases, and thus has an equivalent comparison accuracy as precision $H$ according to our definitions.



$p_L$ copies
x,y require re-computation
x,y
reduced precision datapath
x,y
reduced precision datapath
x,y
reduced precision datapath
$RE_f$, $RE_g$
comparison result
cross-bar
$p_H$ copies
reference precision datapath
reference precision datapath
comparison result

Figure 2. System architecture of the mixed precision comparison.

## IV. Analytical model

In this section, we derive an analytical model for the performance of the mixed precision comparison systems being given a choice of reduced-precision. We define the input and output parameters of the model and show analytically how they are connected. We also show how the optimal precision and optimal resource allocation can be found by solving a set of integer linear programming problems.

Figure 2 shows the system architecture of the mixed precision comparison system for our model. All data (including the values of $\mathbf{x}$ and $\mathbf{y}$) are first fed into one of the reduced-precision data-paths. A comparison result is then returned immediately if the data does not require a re-computation. If re-computation is required, the input data are forwarded to one of the reference-precision data-paths through the communication infrastructure. This data-path includes FIFOs to maintain pipelining. A cross-bar keeps the workloads of the reference-precision data-paths balanced, by multiplexing each reference-precision data-path input to multiple reduced-precision data-path outputs. Although the reduced-precision data-paths adopt lower precision than the reference-precision data-paths, their input data have to be in reference-precision format and outputs are buffered. Data in reference-precision format are thus available if re-computation is required.

Regarding the mixed-precision comparison system in our model, the following assumptions are made:

- The communication overhead, in terms of time, between the reduced-precision data-paths and the reference-precision data-paths is negligible. This is mostly valid when both data-paths are located in the same FPGA.
- Each data-path adopts a homogeneous data format; all operators within a data-path have the same precision.
- The rate of re-computation is constant. This is valid for datasets from similar sources. When the rate of re-computation changes, either we restart the architectural exploration or we set rate of re-computation to the worst case (highest) in our initial exploration in order to cover all possible cases.

Table I
INPUT AND OUTPUT PARAMETERS OF OUR MODEL

| Input parameter | Definition |
|---|---|
| $H$ | the precision of reference-precision data-paths |
| $L$ | the precision of reduced-precision data-paths |
| $p_H$, $p_L$ | number of reduced-precision and high-precision data-paths |
| $C_{lut}(H)$, $C_{mul}(H)$ | area of reference-precision data-paths |
| $C_{lut}(L)$, $C_{mul}(L)$ | area of reduced-precision data-paths |
| $C_{lut}(com)$ | area of the communication infrastructure |
| $R_S$ | slack ratio |
| $T(L)$, $T(H)$ | throughput of reduced-precision and reference-precision data-path |
| $R_R$ | rate of re-computation, the percentage of reduced-precision comparison that require re-computations |
| $A_{lut}$, $A_{mul}$ | total amount of resource in the FPGA |
| **Output parameter** | **Definition** |
| $T(equ)$ | equivalent comparison throughput |

Table II
DEPENDENCE OF DIFFERENT INPUT PARAMETERS.

| parameter | Dependence | | | | |
|---|---|---|---|---|---|
| | device | function | $H$ | $L$ | data |
| $C_{lut}(H)$, $C_{mul}(H)$ | ✔ | ✔ | ✔ | | |
| $C_{lut}(L)$, $C_{mul}(L)$ | ✔ | ✔ | ✔ | | |
| $C_{lut}(com)$ | ✔ | ✔ | ✔ | ✔ | |
| $R_S$ | ✔ | | | | |
| $T(H)$ | ✔ | ✔ | ✔ | | |
| $T(L)$ | ✔ | ✔ | | ✔ | |
| $R_R$ | | ✔ | ✔ | ✔ | ✔ |
| $A_{lut}$, $A_{mul}$ | ✔ | | | | |

Table I shows the input and output parameters of our model. We define *slack ratio* $R_S$ as the percentage of an FPGA's resources that should remain un-used to ensure that the maximum clock frequencies of the data-paths are not degraded due to routing or placement congestion. The value of $H$, $L$, $p_H$ and $p_L$ can be varied, while other input parameters are constants depending on different aspects of the system. The dependence of those constants is shown in Table II. The aggregated throughput of reduced-precision and reference-precision comparison relies on both the throughput of a single comparison data-path $T(L)$ and $T(H)$, and the parallelism of each kind of data-path ($p_L$ and $p_H$). The aggregated throughput $T_A$ can be expressed as:

$$T_A(L) = p_L \times T(L) \quad (5)$$
$$T_A(H) = p_H \times T(H) \quad (6)$$

The number of comparison ($N_{RR}$) that requires re-computation with reference-precision data-path per second is equal to the product of rate of re-computation and the aggregated throughput of reduced-precision data-paths.

$$N_{RR} = T_A(L) \times R_R \quad (7)$$

**Data-path ratio constraint.** When $N_{RR}$ is less than the aggregated throughput of reference-precision data-paths $T_A(H)$, the work load of the reference-precision data-paths is less than their aggregated throughput. Thus the reference-precision data-paths are sometimes stalled to wait for data from the reduced-precision data-paths. Likewise, when $N_{RR}$ is larger than $T_A(H)$, the reduced-precision data-paths are stalled sometimes to wait for the reference-precision data-paths. The equivalent comparison throughput (i.e. number of new inputs that the reduced-precision data-paths can accept per second) is:

$$T(equ) = max(T_A(L), T_A(H)/R_R) \quad (8)$$

The best performance occurs while both sets of data-path are working all the time (i.e. $N_{RR} = T_A(H)$). It implies that there is an optimal ratio for the parallelism of the reduced-precision to reference-precision data-path ($p_L$ / $p_H$) given $H$ and $L$. Figure 3 shows an example of an equivalent throughput curve with different reduced-precision to reference-precision data-paths ratios. Equation 9 shows that the optimal ratio depends on the ratio of the throughputs of the data-paths and the rate of re-computation only. The equation also verifies that more reference-precision data-paths are needed if more comparisons require re-computation.

$$N_{RR} = T_A(L) \times R_R = T_A(H) \Rightarrow \frac{p_L}{p_H} = \frac{T(H)}{T(L)R_R} \quad (9)$$

The optimal ratio stated in equation 9 may not be feasible because $p_L$ and $p_H$ can only take integer values. In such cases, either the reduced-precision data-paths or the reference-precision data-paths are stalled sometimes. We always pick $p_L$ and $p_H$ such that the reference-precision data-paths are stalled rather than the reduced-precision ones to ensure no overflow, which implies the following constraint on the number of data-paths,

$$\frac{p_L}{p_H} \leq \frac{T(H)}{T(L)R_R} \quad (10)$$

**Resource constraint.** As FPGA resources are finite, we include a constraint to ensure that the data-paths can fit in an FPGA. Equation 11 shows the general resource constraint in our model which should be applied to every resource in the data-paths (i.e. lut, mul).

$$p_L \times C(L) + p_H \times C(H) + C(com) \leq A \times (1 - R_S) \quad (11)$$

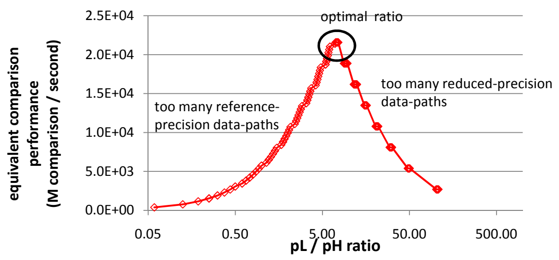


Figure 3. An example of an equivalent comparison throughput verse $p_L/p_H$ for a particualr choice of $L$ and $H$.

**Algorithm 1** Optimisation process for optimal reduced-precision and resource allocation

```
1: $T_{max} \leftarrow 0$
2: for $L$ = $L_{min}$ to $L_{max}$ do
3:     locate optimal $p_L$ and $p_H$
4:     $T(equ) \leftarrow p_L \times T(L)$
5:     if $T(equ) > T_{max}$ then
6:         $T_{max} \leftarrow T(equ)$
7:     end if
8: end for
```

**Equivalent comparison throughput.** The equivalent comparison throughput $T(equ)$ is the number of new inputs that the system can accept per second. If the constraint in equation 10 is satisified, the allowed number of new inputs per second is only restricted by the aggregated throughput of the reduced-precision data-path.

$$T(equ) = p_L \times T(L) \tag{12}$$

There are four main parameters in our model. $H$ is the reference-precision and is determined by the comparison accuracy requirement. $p_H$ and $p_L$ determine the FPGA resource allocation between the two kinds of data-path. $L$ is the precision adopted by the reduced-precision data-paths. Algorithm 1 is used to obtain the optimal precision $L$ and the optimal resource allocation.

$L_{min}$ and $L_{max}$ are the minimum and maximum precision for the reduced-precision data-paths in the design space. In step 3 of Algorithm 1, we apply an integer linear programming (ILP) using equation 12 as the objective function and equation 10 and 11 as constraints. The optimal reduced-precision is simply the one that gives the highest comparison throughput among all the possible precisions. In the following section, we show how Algorithm 1 is used in optimising precision and resource allocation for a given application.

## V. Collision detection algorithm

We consider a collision detection algorithm of pairwise bounding sphere in 3-dimensional space as our example. It is commonly used in the broad-phase of collision detection algorithm being used in gaming and Physics simulations [4]. This section shows how it is mapped to a reconfigurable system using the proposed mixed-precision methodology.

The input to the collision detection problem is a set of spheres in 3-dimensional space with different centres and radii. Each sphere is represented by a quadruple ($c_x$, $c_y$, $c_z$, $r$) and the output of the algorithm is the set of overlapping sphere pairs. As shown in Algorithm 2, there exist ($\sim N^2/2$) function comparisons in a pairwise collision detection algorithm for $N$ spheres.

**Defining the functions for comparison.** The first step in mapping the collision detection algorithm is to identify the two functions for comparison. We choose $f = \sum_{k=x,y,z}(c_{ki} - c_{kj})^2$ and $g = (r_i + r_j)^2$. It is interesting to note that operations can be transferred from one function to another without altering the original comparison. For instance we can remove the square operation in $g$ and insert a square root operation in $f$ instead.

**Algorithm 2** 3D spherical volume based collision detection algorithm

```
Require: $\mathbf{c_i}$, coordinates of the centers of the spheres
  $r_i$, radius of the spheres
  for $i$ = 1 to $N-1$ do
    for $j$ = i+1 to $N$ do
      $D = ||\mathbf{c_i} - \mathbf{c_j}|| - (r_i + r_j)$
      if $D < 0$ then
        sphere $i$ collides with sphere $j$
      end if
    end for
  end for
```

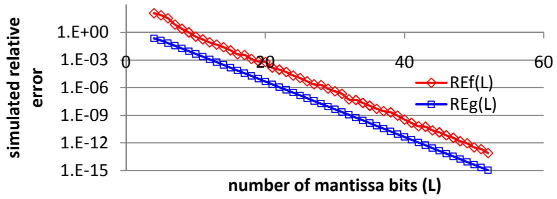


Figure 4. Simulated maximum relative error of function $f$ and $g$ for floating-point data formats with different number of mantissa bit.

**Error profiling of the functions.** The second step of the mapping process is to determine the relative error of the two functions in different precisions. Any accuracy modelling approaches mentioned in section II can be used for this purpose. A Monte-Carlo based method is employed for simplicity. We randomly generate a million test vectors and estimate their relative errors in different precision using equation 4. We use the MPFR multiple precision floating-point library to simulate the function values for different numbers of mantissa bits [21]. An $L$ mantissa-bit floating-point representation is used to compute the value of $f_L$ while a 500 mantissa-bit floating-point format is used to compute the true value ($f_T$) and other operations in equation 4. The number of exponent-bit is adjusted automatically by the library to prevent overflow in both representations. The IEEE round to the nearest rounding mode is chosen in the simulation to match the rounding mode of our hardware implementation based on Xilinx LogicCore library [22]. Figure 4 shows the simulated relative errors of the two functions in the collision detection algorithm. When the precision of the data-path increases, the relative error decreases.

**Profiling the rate of re-computation.** The third step of the mapping processing is to profile the rate of re-computation using the relative errors ($RE_f$ and $RE_g$) obtained. We generate random collision benchmarks with different characteristics using the common method mentioned in [23]; the

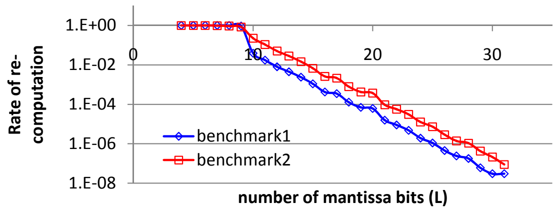


Figure 5. Rate of re-computation of an example benchmark of different choice of reduced-precision $L$.

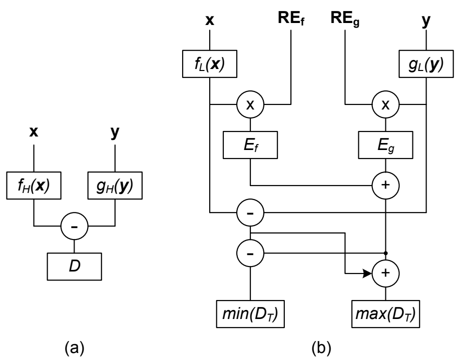


Figure 6. The reference-precision data-path (a) and the reduced-precision data-path (b).

details of the benchmark generation is explained in the next section. As shown in figure 5, the rate of re-computation drops quickly with increasing precision of the data-paths and become undetectable when there are more than 31 mantissa bits. We also find that the rate of re-computation is dataset dependent and it changes according to the characteristic of the dataset. Thus the benchmark for profiling $R_R$ should follow the characteristic of the dataset of actual operations. When prior knowledge about the dataset is not available or it changes constantly, we can either use a worst-case rate of re-computation (i.e. the maximum among all datasets) or use a run-time re-computation rate profiling method. The first method will lead to a non-optimal precision allocation, and the second method requires run-time reconfiguration.

**Extracting the characteristics of the data-paths and the communication infrastructure.** The next step of the mapping process is to extract the throughput and area of the data-paths and communication infrastructure. Figure 6 shows the block diagrams of the reference-precision and reduced-precision data-paths. As shown in the figure, the reduced-precision data-path requires 2 additional multipliers and 3 additional adders in order to compute the minimum and maximum values of the difference. The additional operators are the overhead of using reduced-precision data-paths and the overhead remains the same for any function $f$ and $g$.

We select two FPGAs from Xilinx to implement the mixed-precision comparison:

1) The Virtex-5 VLX 330T which is based on 65nm technology. The number of embedded multipliers on this FPGA is relatively limited and they are not used in our designs.
2) The Virtex-6 VLX760 which is based on 40nm technology. Both LUTs and embedded multipliers are used in the designs.

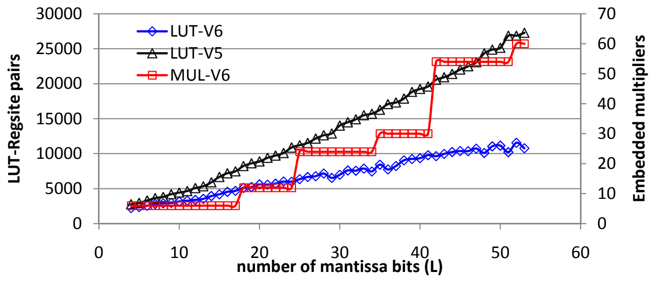


Figure 7. Cost of comparison data-paths with different precision.

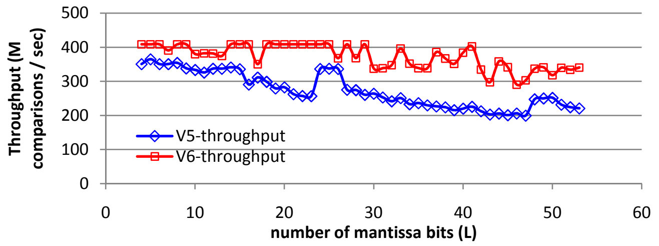


Figure 8. Throughput of comparison data-paths with different precision.

We use the Xilinx LogicCore floating-point library to implement our data-paths. For the reference-precision data-path, an s53e8 (i.e. 53 mantissa bits and 8 exponent bits) data format is used which mimic the number of mantissa bits of an IEEE double precision format. For the reduced-precision data-paths, we use $L$-bit mantissa floating-point format with 8-bit exponent, where $L$ range from 4 to 52. Figure 7 and figure 8 respectively show the cost and throughput of the reduced-precision data-paths after the place and route process. As shown in the figures, low-precision data-paths consume significantly less resources and have higher throughputs compared with the high-precision ones. We also place and route the reference-precision data-paths and the corresponding results are reported in Table III. On both FPGAs, the reference-precision data-paths with an s53e8 data format consume fewer LUTs and embedded multipliers than any reduced-precision data-path with more than 41 mantissa bits. This is because of the additional operators in the reduced-precision data-paths for the computation of maximum and minimum. We use "crossover precision" to denote the precision where reduced-precision data-paths become more expensive than the reference-precision data-path. In the process of locating optimal precision using Algorithm 1, we set $L_{max}$ to be the crossover precision since it will be more efficient to use the reference-precision data-path directly provided that $L$ is larger than that value.

Table III
AREA AND THROUGHPUT OF THE REFERENCE-PRECISION DATA-PATHS.

| | LUT-reg pairs | multipliers | throughputs (MHz) |
|---|---|---|---|
| Virtex 5 | 17198 | 0 | 223 |
| Virtex 6 | 7350 | 40 | 285 |

We design a crossbar with a variable number of I/O ports for the distribution of re-computations among the reference-precision data-paths. To simplify our calculation, we use the area of the design with a maximum number of I/O ports as the communication infrastructure overhead ($C_{lut}(com)$). The overhead (i.e. FIFOs and the crossbar) is 11440 LUTs in the Virtex-6 design and 4790 LUTs in the Virtex-5 design. The Virtex-6 one consumes more area because it is connected to more data-paths.

**Optimal precision and optimal resource allocation.** The final step of the mapping process is to locate the optimal precision for reduced-precision data-paths and the optimal resource allocation using Algorithm 1. Figure 9 shows the maximum performance and the optimal number of reduced and reference-precision datapaths, for different precisions $L$ of an example benchmark. Each data point on the curve represents a mixed-precision system with optimal number of reduced-precision and reference-precision data-path. However, only the data point with highest performance has the optimal precision $L$. On the left of the optimal precision, the re-computation rates are so high that too many comparisons require re-computations by the reference-precision data-paths. Beyond the optimal precision to its right, re-computation rates are low but the area cost of reduced-precision data-path is too high.

## VI. PERFORMANCE EVALUATION

We generate random collision benchmarks with different characteristics using a common method described in [23]. 8192 spheres are defined with their centres randomly located inside a 3D cube with length equal to 100 units. Their radii are randomly assigned as 0 to $r$ units. We design 20 benchmarks with values of $r$ between 5 to 100 units. Different values of $r$ give datasets with different re-computation rate characteristics. We consider two scenarios for the application of our mixed-precision methodology to the benchmarks. In scenario $I$ we assume that the rate of re-computation of each benchmark is known a priori and a mixed-precision system with optimal precision and resource allocation is used. In scenario $II$, we randomly choose a benchmark but the re-computation rate is unknown to the mixed-precision system. The worst-case rate of re-computation (i.e. The highest one among all datasets) is used in this scenario. As a result, optimal performance is not guaranteed.

We compare the performance of the collision detection algorithm for different systems. The following alternative

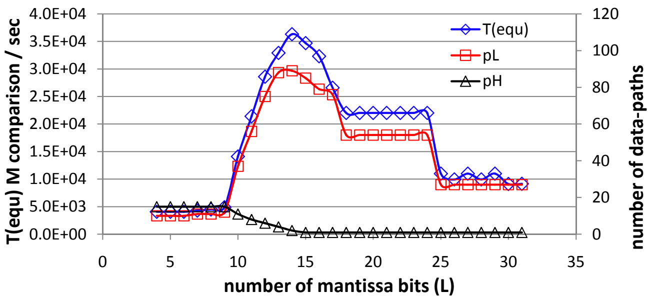


Figure 9. Equivalent comparison throughput $T(equ)$ and the corresponding optimal parallelism ($p_L$, $p_H$) for reduced-precision and reference-precision data-paths for different choice of precision of an example benchmark.

implementations are chosen for the comparison to our mixed precision methodology on FPGAs. IEEE double precision is used in the software implementations and s53e8 data format is used in the baseline FPGA implementation. We select CPUs with the same technology to ensure a fair comparison. The performance on a single core of the CPUs is multiplied by the number of cores as the aggregated throughput. It is important to note that all implementations being compared have the same comparison accuracy.

Table IV shows the equivalent comparison performance of different software and hardware designs of the collision detection algorithm. Although the clock frequencies of FPGA designs are an order of magnitude lower than that of a CPU with the same technology node, the FPGA comparison data-paths have similar performance to a single core on the CPUs. The baseline FPGA designs are 2.09 to 4.08 times faster than the software implementations due to their higher degree of parallelism (i.e. more cores). The degree of parallelism is significantly increased when mixed precision methodology is applied to the FPGA designs. Additional speedups of 4.09 to 8.61 times are gained when we have prior knowledge about the benchmarks and the speedups are reduced to 4.09 to 7.37 times when the knowledge is not available. The major advantage of our mixed precision methodology is that the additional performance gain does not come at a cost of reduced comparison accuracy. Using the mixed precision methodology, we can achieve 15.4 to 16.7 times speedup compared with multi-cores CPUs with the same technology at worst.

## VII. CONCLUSION

In this paper, we propose a novel mixed-precision methodology for numerical function comparison in reconfigurable systems. The methodology covers any application involving function comparison and exploits customisable reduced-precision data format which is only available in reconfigurable systems. A model is developed for locating the optimal precision and the optimal resource allocation. Experimental results for a common collision detection problem

Table IV
PEFORMANCE OF DIFFERENT FUNCTION COMPARISON SYSTEMS.

| device | Intel Core 2 | Virtex-5 | Virtex-5 (mixed precision) | | Intel Xeon | Virtex-6 | Virtex-6 (mixed precision) | |
|---|---|---|---|---|---|---|---|---|
| | 6420 | (baseline) | scenario I | scenario II | E5420 | (baseline) | scenario I | scenario II |
| Technology node | 65nm | | | | 40nm | | | |
| # of core (reference-precision) | 2 | 9 | 1 | 1 | 8 | 17 | 1-3 | 3 |
| # of core (reduced-precision) | - | - | 24-32 | 24 | - | - | 87-108 | 87 |
| clock frequency | 2.13 GHz | 223 MHz | 199-337 MHz | | 2.53 GHz | 285 MHz | 285 - 408 MHz | |
| throughput (single core) (G comparison/sec) | 0.24 | 0.22 | not applicable | | 0.29 | 0.28 | not applicable | |
| throughput (aggregated) (G comparison/sec) | 0.49 | 2 | 8.17 - 10.6 | 8.17 | 2.3 | 4.8 | 35.5-41.3 | 35.5 |
| Normalised speedup[a] | 1x | 4.08x | 16.7 - 21.6x | 16.7x | 1x | 2.09x | 15.4 - 18x | 15.4x |
| mixed precision gain[b] | - | 1x | 4.09 - 5.29x | 4.09x | - | 1x | 7.37 - 8.61x | 7.37x |

[a] Normalised to the multi-core throughput of CPU within the same technology.
[b] Normalised to the multi-core throughput of reference-precision design on the same FPGA.

show that the methodology can provide additional performance gain of 4 to 7.3 times over the baseline reference-precision implementations on the same FPGAs.

Future work includes applying the methodology to other hardware architectures having large performance difference between different precision formats such as GPU, and comparing the performance, area and energy efficiency of different devices. The integration of our mixed-precision method with other techniques such as data clustering is also one of our future research directions.

**Acknowledgement.** The support of the Croucher Foundation, UK EPSRC, Alpha Data and Xilinx is gratefully acknowledged. The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement number 248976 and 257906.

## REFERENCES

[1] K. Underwood, "FPGAs vs. CPUs: trends in peak floating-point performance," in *Proc. FPGA*, 2004, pp. 171–180.

[2] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[3] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Computational Statistics and Data Analysis*, vol. 14, no. 3, pp. 315–332, October 1992.

[4] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe, "Collision detection: A survey," in *IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 4046–4051.

[5] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, January 1965.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[7] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Trans. CAD*, vol. 20, no. 8, pp. 921–930, 2001.

[8] C. F. Fang, R. A. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *IEEE/ACM International Conference on Computer-Aided Design*, 2003, pp. 275–282.

[9] A. A. Gaffar, O. Mencer, W. Luk, and P. Y. K. Cheung, "Unifying bit-width optimisation for fixed-point and floating-point designs," in *FCCM*, 2004, pp. 79–88.

[10] D.-U. Lee, A. A. Gaffar, O. Mencer, and W. Luk, "Minibit: bit-width optimization via affine arithmetic," in *DAC*, 2005, pp. 837–840.

[11] D.-U. Lee, A. A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. on CAD*, vol. 25, no. 10, pp. 1990–2000, 2006.

[12] W. G. Osborne, R. C. C. Cheung, J. G. F. Coutinho, W. Luk, and O. Mencer, "Automatic accuracy-guaranteed bit-width optimization for fixed and floating-point systems," in *Proc. FPL*, 2007, pp. 617–620.

[13] W. Osborne, J. Coutinho, R. Cheung, W. Luk, and O. Mencer, "Instrumented multi-stage word-length optimization," in *Proc. ICFPT*, 2007, pp. 89–96.

[14] A. Kinsman and N. Nicolici, "Finite precision bit-width allocation using SAT-Modulo theory," in *Proc. DATE*, 2009, pp. 1106–1111.

[15] D. Boland and G. Constantinides, "Automated precision analysis: A polynomial algebraic approach," in *Proc. FCCM*, 2010, pp. 157–164.

[16] R. Strzodka and D. Goddeke, "Pipelined mixed precision algorithms on FPGAs for fast and accurate PDE solvers from low precision components," in *Proc. FCCM*, 2006, pp. 259–270.

[17] J. Sun, G. D. Peterson, and O. O. Storaasli, "High-performance mixed-precision linear solver for FPGAs," *IEEE Trans. Computer*, vol. 57, pp. 1614–1623, December 2008.

[18] G. Morris and V. Prasanna, "An FPGA-based floating-point Jacobi iterative solver," in *Internation Symposium on Parallel Architectures, Algorithms and Networks*, 2005.

[19] A. Roldao-Lopes, A. Shahzad, G. A. Constantinides, and E. C. Kerrigan, "More Flops or more precision? Accuracy parameterizable linear equation solvers for model predictive control," in *Proc. FCCM*, 2009, pp. 209–216.

[20] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM: Society for Industrial and Applied Mathematics, 2002.

[21] L. Fousse et al, "MPFR: A multiple-precision binary floating-point library with correct rounding," *ACM Trans. Math. Softw.*, vol. 33, June 2007.

[22] Xilinx inc., "DSS335: Floating point operator v5.0," 2009.

[23] M. Woulfe and M. Manzke, "A framework for benchmarking interactive collision detection," in *Proc. 25th Spring Conference on Computer Graphics*, 2009.