ERROR MODELLING OF DUAL FIXED-POINT ARITHMETIC AND ITS APPLICATION IN FIELD PROGRAMMABLE LOGIC

Chun Te Ewe, Peter Y. K. Cheung and George A. Constantinides

Department of Electrical & Electronic Engineering,
Imperial College London, Exhibition Road,
London SW7 2BT, United Kingdom.
{ct.ewe, p.cheung, g.constantinides}@imperial.ac.uk

ABSTRACT

Dual FiXed-point (DFX) is a new data representation which is an efficient compromise between fixed-point and floating-point representations. DFX has an implementation complexity similar to that of a fixed-point system with the improved dynamic range capability of a floating-point system. Automating the process of DFX scaling optimisation requires the knowledge of its truncation/rounding noise properties. This paper presents truncation and rounding error models for DFX arithmetic as traditional error models do not apply to DFX. The models were tested on a 159-tap FIR filter and the benefits of using DFX over floating-point are demonstrated with implementations on a Xilinx Virtex II Pro.

1. INTRODUCTION

FPGAs have long been an attractive alternative to digital signal processors in DSP applications provided that floating-point is not required. In recent years, the size of FPGA devices has increased to such an extend that floating-point implementations have become possible[1, 2]. Since floating-point designs are considerably larger and slower than their fixed-point counterparts, their use is only justified where very large dynamic range is required.

In [3] the authors introduced a new representation known as *Dual FiXed-point* (DFX). It showed that for the same chip area, DFX designs outperform floating-point due to their reduced complexity while having a similar dynamic range. To obtain an efficient DFX implementation while satisfying the computational accuracy constraints imposed by the designer, design automation tools that automatically determine the optimum parameters in a DFX design needs to be developed. One prerequisite to the development of such tools is an accurate truncation error model for DFX arithmetic modules. Unfortunately, traditional techniques such as the additive roundoff error model for fixed-point[4] and the relative roundoff error model for floating-point[5] cannot be utilised because DFX is a scaled number presentation and is not normalised.

This paper reports our latest work on deriving an accurate error model for DFX arithmetic modules. The original contributions of this paper are: 1) detail implementations of DFX arithmetic modules with rounding are examined and the sources of truncation errors identified; 2) analytical models for errors introduced by DFX arithmetic modules are developed; 3) these errors models are verified against simulation results and are shown to be accurate; 4) the models are applied to a 159-tap FIR filter and the size, speed and noise performance of the DFX implements are compared against those using floating point.

The paper is organised as follows. Section 2 presents the background and definition of DFX. The basic arithmetic modules using DFX and the sources of truncation errors are described in Section 3. Section 4 presents the error models of each individual arithmetic module. A case study showing the benefits of DFX on a FIR filter and a test of the error models are given in Section 5. Section 6 concludes the paper and suggestions for future work are presented.

2. DUAL FIXED-POINT: BACKGROUND AND DEFINITION

Fig. 1. DFX Format

An n-bit Dual FiXed-point (DFX) number consists of an exponent bit E, and $n-1$ bits of a signed significand X as shown in Figure 1. The exponent selects between two scalings for the significand X, giving two possible ranges for the number. The lower number range is referred to as $Num0$ while the higher number range is referred to as $Num1$.

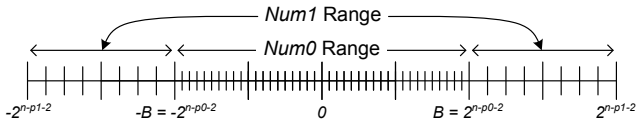

Fig. 2. $Num0$ and $Num1$ range in a DFX Number

To achieve two different scalings, $Num0$ has p_0 fractional bits and $Num1$ has p_1. The DFX format notation used is written as $n\_p_0\_p_1$.

The value of a DFX number, D, is given by

$$D = \begin{cases} X \cdot 2^{-p_0} & \text{if } E = 0 \\ X \cdot 2^{-p_1} & \text{if } E = 1 \end{cases} \quad (1)$$

A *boundary value*, B, is needed to decide the best scaling to use and hence the value of E. E is determined as follows,

$$E = \begin{cases} 0 & \text{if } -B \leq D < B \\ 1 & \text{if } D < -B \text{ or } D \geq B \end{cases} \quad (2)$$

In order to simplify the design of the arithmetic units, the boundary value is defined as the next incremental value after the maximum positive number of $Num0$, i.e. $B = 2^{n-p_0-2}$ (-2 because of the exponent and sign bits). The range and precision of $Num0$ and $Num1$ are illustrated in Figure 2.

3. DFX ARITHMETIC MODULES

The basic arithmetic modules for DFX have been designed in VHDL. Unlike the modules introduced in [3], these modules here perform rounding instead of truncation.

3.1. DFX Adder

The DFX Adder module (Figure 3) adds together two DFX numbers. Similar to a floating-point adder, DFX inputs may need aligning before addition. But unlike floating-point, the number of bits shifted is known *a priori*. This means only multiplexers are needed to perform the necessary scaling instead of barrel shifters. As a result, the DFX Adder is both smaller and faster than an equivalent floating-point adder. Note that "$\gg$" and "$\ll$" are right and left shift operators respectively which require only wire routing, and "mod 2^{n-1}" simply extracts the least significant $(n-1)$ bits.

Whenever the input exponents are different, i.e. one input is $Num0$ and the other $Num1$, the $Num0$ input will be shifted up to the $Num1$ range. When both exponents are the same, there will be no shifting. $gBits$ either retains the truncated bits as the result of shifting or carry all zero bits. The retained truncated bits is used to reduce the Adder's output truncation error (Section 4.4).

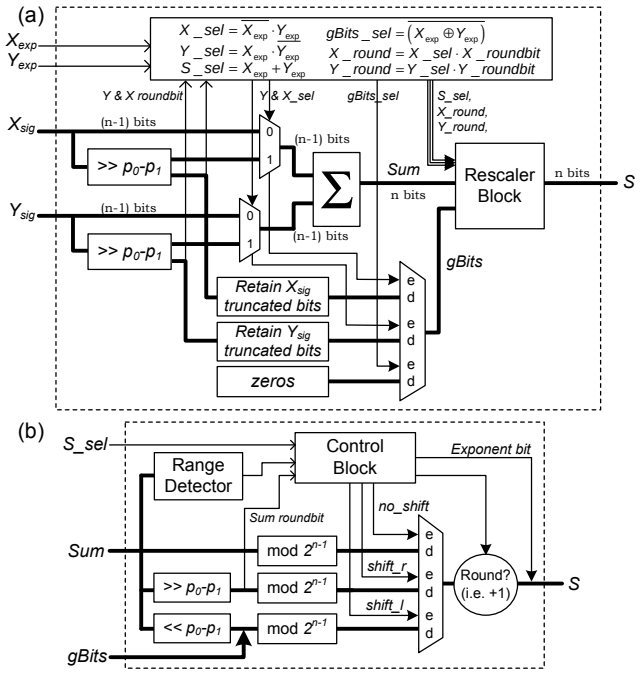

Fig. 3. DFX (a)Adder Module and its (b)Rescaler Block with rounding

The result of the adder, Sum, is fed into the Rescaler Block (Figure 3(b)). It first detects the range of Sum with the Range Detector. No shifting is needed if the adder's result remains in the same range. If the result changes from a $Num0$ to a $Num1$, Sum has to be shifted $(p_0 - p_1)$ bits to the right and sign extended. The result will however be shifted $(p_0 - p_1)$ bits to the left and concatenated with $gBits$ if the result changes from a $Num1$ to a $Num0$ number.

Rounding is done after all the scaling is complete. Two factors determine the rounding decision. Firstly, if either of the inputs are shifted right and the Sum is not shifted, the rounding decision depends on the either of the input's rounding bit. Secondly, if the Sum is shifted right, the rounding decision depends on the Sum's rounding bit.

3.2. DFX-H Multiplier

A DFX-Half (DFX-H) Multiplier (Figure 4) takes one DFX input and a constant fixed-point multiplier. This is particularly useful in applications such as filtering where one of the operands is a constant. Unlike the DFX Adder, the inputs to the multiplier do not need aligning. However, the product P needs to be properly scaled and converted back to DFX.

Consider the multiplication of a DFX $n\_p_0\_p_1$ number with a fixed-point $n_m\_p_m$ (n_m is the word-length of the multiplier m and p_m is the fractional length). The product of the multiplication, $Prod$, would be in the format DFX

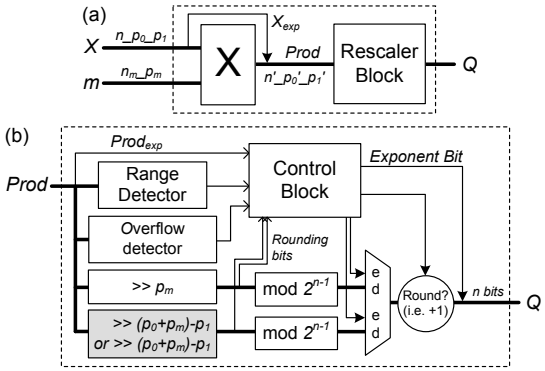

Fig. 4. (a)DFX-H Multiplier Module and its (b)Rescaler Block with rounding

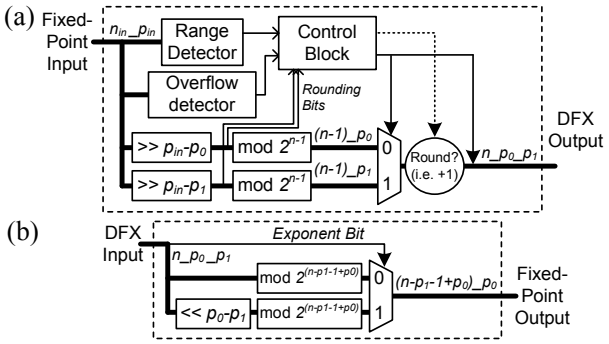

Fig. 5. (a) DFX Encoder with rounding and (b) DFX Decoder

Table 1. DFX-H Multiplier input and output combinations with their respective truncations and shifting

Case #	Input	Output	Truncation	Shift
1	$Num0$	$Num0$	$p_0+p_m \rightarrow p_0$	$\gg p_m$
2	$Num1$	$Num1$	$p_1+p_m \rightarrow p_1$	$\gg p_m$
3	$Num0$	$Num1$	$p_0+p_m \rightarrow p_1$	$\gg (p_0+p_m)-p_1$
4	$Num1$	$Num0$	$p_1+p_m \rightarrow p_0$	$\gg (p_1+p_m)-p_0$

$n'\_p'_0\_p'_1$, where $n' = n + n_m$, $p'_0 = p_0 + p_m$ and $p'_1 = p_1 + p_m$. It is then converted back to a DFX $n\_p_0\_p_1$ formatted number by the Rescaler Block (Figure 4(b)).

Table 5 shows all possible input and output combinations with their respective output truncations and output shifts required. The truncation $p_a \rightarrow p_b$ represents converting from a number with the binary point p_a to p_b. When the multiplier $|m| > 1$, Case 4 will never happen. On the other hand, Case 3 will never happen when $|m| < 1$.

The decision to round depends on the output scaling and it's rounding bit while the $Num0$ number does not overflow. Unlike ordinary truncation, rounding may cause an overflow due to the non-symmetrical nature of 2's complement representation. The overflow of the $Num0$ range is most vital of all as we need to guarantee there is no overflow within the whole number range.

3.3. DFX Encoder and Decoder

In order to utilize this number system, a method is needed to convert a number from a known type to DFX and vice-versa. Figure 5(a) and Figure 5(b) show the encoder and decoder modules that converts to and from 2's complement fixed-point representation and DFX representation. The decision to round depends on the rounding bits of the output scaling. Just like the DFX-H Multiplier, rounding will be done as long it does not cause an overflow.

Table 2. Comparison of 16-bit DFX arithmetic modules with equivalent floating-point implementations. Designs were mapped onto a Xilinx Virtex II (XC2V1000-6bg575) where multiplier blocks were used where appropriate.

Module	**Size (Latency)** / slices(ns)		
	DFX (Rounded)	DFX (Truncated)	Floating-Point
Adder	39(18.3)	36(15.3)	161(38.8)
Multiplier	24(19.1)	16(16.6)	50(29.0)
Encoder	14(15.1)	9(13.0)	n/a
Decoder	10(10.7)	10(10.7)	n/a

3.4. Arithmetic Module Comparisons

For completeness, here's a comparison of the DFX Modules (rounded and truncated) with equivalent floating-point implementations [6]. All modules have a 16-bit word-length. The DFX modules are of the format 16_19_14 and floating-point modules are of the format M7 E8 (7 mantissa bits and 8 exponent bits). The DFX modules with rounding is not very much larger than their truncated counterpart. This is because the addition logic for rounding is absorbed into the multiplexer stage before it.

4. ERROR ANALYSIS OF DFX ARITHMETIC MODULES

The noise for each DFX arithmetic module is modelled as an addition of an error source at the end of each module. These errors are highly dependent on the distribution and correlation of its inputs, thus ordinary static error analysis [4] is not possible. Provided that we know the probability distribution function of arithmetic module's inputs, we can estimate the output error. The distribution function is obtained by performing a single pass *profiling* simulation, which is explained later. This paper focuses on the noise added by each DFX arithmetic module, therefore all inputs to the modules are assumed to contain no errors.

4.1. Background

Errors are introduced into a system whenever truncation takes place. In the case of DFX modules, truncation occurs whenever there is a right shift in the data path. A two's complement signal with binary point p_a truncated to p_b will introduce an error with the mean and variance given by (3) which uses a discrete error distribution [7]. The equations are derived from the assumption that each of the combinations of the low-end truncated bits are equally likely, which holds true in practice if the signals have sufficient dynamic range over that bit-width. If rounding is performed instead of truncation, (3) still applies but the error mean becomes zero while the variance remains the same.

$$\begin{aligned} mean = \mu &= -\frac{1}{2}\left(2^{-p_b} - 2^{-p_a}\right) \\ variance = \sigma^2 &= \frac{1}{12}\left(2^{-2p_b} - 2^{-2p_a}\right) \end{aligned} \tag{3}$$

Since DFX has dual precision, more than one truncation/rounding error may occur within each arithmetic module. Let T be the set of all these possible truncation/rounding that may occur and let $i \in \mathrm{T}$. For every truncation/rounding i, there is a corresponding error mean, μ_i, error variance, σ_i^2 and probability of truncation occurring, P_i. From the profiling simulation, we can determine the probability of all the sources of truncations within each module. Therefore, the output error mean and error variance are given by (4). Again, if rounding is performed, the error mean will be zero and the variance is calculated with the zero error means.

$$\begin{aligned} \mu_{error} &= \sum_{i \in \mathrm{T}} P_i \mu_i \\ \sigma^2_{error} &= \sum_{i \in \mathrm{T}} P_i \left(\sigma_i^2 + \mu_i^2\right) - \mu^2_{error} \end{aligned} \tag{4}$$

4.2. Profiling Simulation

In the system context, the errors of each DFX arithmetic module are highly dependent on the correlation between the input signals. The purpose of the profiling simulation is to obtain the joint/probability distribution function of the inputs to each arithmetic module within the system. This is done by feeding a set of typical representative data into the system for a single pass simulation. While the simulation is running, information regarding the magnitude, sign and correlation between the inputs are gathered for each module. With this information, the probability distribution function (PDF) can be obtained for modules with a single input or the joint probability distribution function for dual input modules. The information gathered by this single pass simulation together with the following error models are sufficient to estimate the errors of any DFX format required.

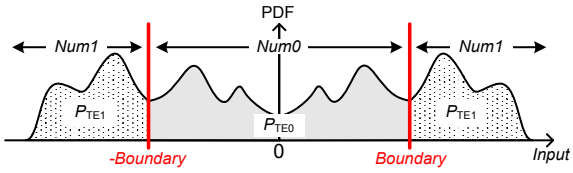

Fig. 6. Probability density function(PDF) of DFX Encoder Input signal

Table 3. DFX Encoder error means and variances where p_{in} is the binary point of the Input

Trunctation	P_{T}	μ_{T}	σ^2_{T}
T_{E0}	$P_{\mathrm{T}_{E0}}$	$-\frac{1}{2}(2^{-p_0} - 2^{-p_{in}})$	$\frac{1}{12}(2^{-2p_0} - 2^{-2p_{in}})$
T_{E1}	$P_{\mathrm{T}_{E1}}$	$-\frac{1}{2}(2^{-p_1} - 2^{-p_{in}})$	$\frac{1}{12}(2^{-2p_0} - 2^{-2p_{in}})$

4.3. Error: DFX Encoder Module

This module performs two forms of quantisation depending on its input. If the output is a $Num0$, the output would be truncated by T_{E0} and if the output is a $Num1$, the output would be truncated by T_{E1}. The quantisation means and variances of T_{E0} and T_{E1} are shown in Table 3.

From the profiling simulation, the PDF of the input can be obtained as shown in Figure 6. From the PDF, the probability $P_{\mathrm{T}_{E0}}$ is the integral of the PDF curve whereby the *Input* is a $Num0$. Likewise, the probability $P_{\mathrm{T}_{E1}}$ is the integral of the PDF curve whereby the *Input* is a $Num1$. Therefore using (4), the modelled error mean and the error variance is given by (5). If rounding is performed, the error mean is zero and the error variance is calculated similar to the ordinary truncation but with zero error mean.

$$\begin{aligned} \mu_{Enc} &= \mu_{\mathrm{T}_{E0}} P_{\mathrm{T}_{E0}} + \mu_{\mathrm{T}_{E1}} P_{\mathrm{T}_{E1}} \\ \sigma^2_{Enc} &= \left((\sigma^2_{\mathrm{T}_{E0}} + \mu^2_{\mathrm{T}_{E0}}) P_{\mathrm{T}_{E0}} + (\sigma^2_{\mathrm{T}_{E1}} + \mu^2_{\mathrm{T}_{E1}}) P_{\mathrm{T}_{E1}}\right) \\ &\quad - \mu^2_{Enc} \end{aligned} \tag{5}$$

4.4. Error: DFX Adder Module

The analysis of the error model for this module begins with analysing all possible input combinations. Table 4 depicts that truncation happens in only 2 out of 6 cases. Ideally, if the inputs were independent of each other, obtaining the probability distribution of the inputs individually would be sufficient. However, in practice, input signals have some degree of correlation between them. Instead, a joint probability distribution table (Figure 7) of the inputs is obtained via the profiling simulation. The table can be viewed as a graph with the x-axis for the input X and y-axis for the input Y. Boundaries marked on the table separates the regions where the inputs are a $Num0$ and a $Num1$. The shaded area denotes the area where the adder's result is truncated.

Table 4. DFX Adder Module input combinations and their respective output truncations

Case #	Input Combination	Addition Result	Truncation
1	$Num0+Num0$	$Num0$	None
2	$Num0+Num0$	$Num1$	Yes
3	$Num0+Num1$	$Num0$	None
4	$Num0+Num1$	$Num1$	Yes
5	$Num1+Num1$	$Num0$	None
6	$Num1+Num1$	$Num1$	None

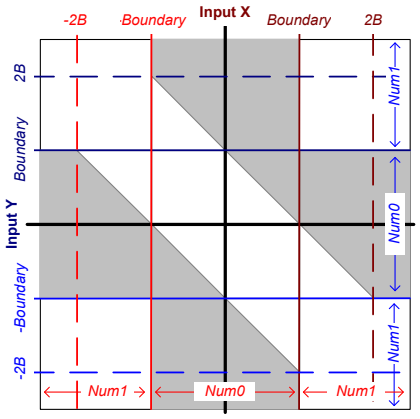

Fig. 7. DFX Adder input joint probability distribution table

Provided that DFX Adder's inputs and output have the same DFX format, the truncations within the DFX Adder will have the same error mean and error variance as given by (6). The probability of truncation occurring, P_{T_A} is the integral of the shaded area of the joint probability distribution table (Figure 7).

$$\mu_{\mathrm{T}_A} = -\frac{1}{2}\left(2^{-p_1} - 2^{-p_0}\right) \quad \sigma^2_{\mathrm{T}_A} = \frac{1}{12}\left(2^{-2p_1} - 2^{-2p_0}\right) \tag{6}$$

Therefore the DFX Adder's output error mean and error variance injected are given by (7). Again, if rounding is performed, the error mean is zero and the error variance are calculated with zero error mean.

$$\begin{aligned} \mu_{Adder} &= P_{\mathrm{T}_A}\mu_{\mathrm{T}_A} \\ \sigma^2_{Adder} &= P_{\mathrm{T}_A}\left(\sigma^2_{\mathrm{T}_A} + \mu^2_{\mathrm{T}_A}\right) - \mu^2_{Adder} \end{aligned} \tag{7}$$

4.5. Error: DFX-Half Multiplier (DFX-H Multiplier)

As mentioned earlier in Section 3.2, the DFX-H Multiplier multiplies a DFX number, X, with a fixed-point constant multiplier, m, with the format $n_m.p_m$. Table 5 shows all the possible truncation error means and variances.

Table 5. DFX-H Multiplier Module input combinations and their respective output truncation error means and variances

Case	Input	Output	P_{T}	μ_{T}	σ^2_{T}
1	$Num0$	$Num0$	$P_{\mathrm{T}_{M0\_0}}$	$-\frac{1}{2}(2^{-p_0}-2^{-(p_0+p_m)})$	$\frac{1}{12}(2^{-2p_0}-2^{-2(p_0+p_m)})$
2	$Num1$	$Num1$	$P_{\mathrm{T}_{M1\_1}}$	$-\frac{1}{2}(2^{-p_1}-2^{-(p_1+p_m)})$	$\frac{1}{12}(2^{-2p_1}-2^{-2(p_1+p_m)})$
3	$Num0$	$Num1$	$P_{\mathrm{T}_{M0\_1}}$	$-\frac{1}{2}(2^{-p_1}-2^{-(p_0+p_m)})$	$\frac{1}{12}(2^{-2p_1}-2^{-2(p_0+p_m)})$
4	$Num1$	$Num0$	$P_{\mathrm{T}_{M1\_0}}$	$-\frac{1}{2}(2^{-p_0}-2^{-(p_1+p_m)})$	$\frac{1}{12}(2^{-2p_0}-2^{-2(p_1+p_m)})$

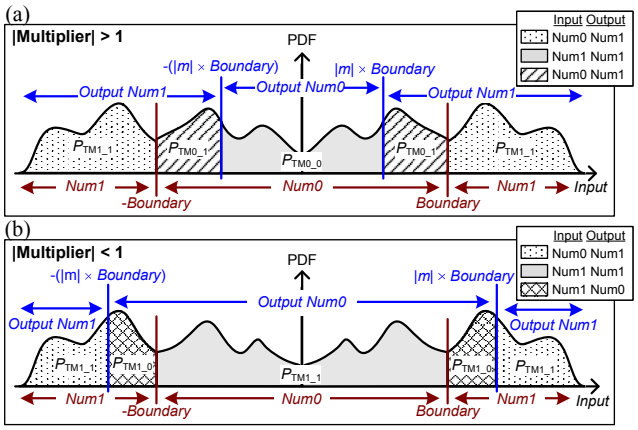

Fig. 8. PDF of the DFX-H Multiplier input and the probabilities of truncation when (a)$|m| > 1$ and (b)$|m| < 1$

When $|m| > 1$, the output product will never be a $Num0$ if the input is a $Num1$ which means Case 4 will never occur. The probabilities for each case can be found from Figure 8(a) through integration. Therefore, using (4), the output truncation error mean and variance when $|m| > 1$ are given by (8).

$$\begin{aligned} \mu_{Mgt1} &= \mu_{\mathrm{T}_{M0\_0}}P_{\mathrm{T}_{M0\_0}} + \mu_{\mathrm{T}_{M0\_1}}P_{\mathrm{T}_{M0\_1}} + \mu_{\mathrm{T}_{M1\_1}}P_{\mathrm{T}_{M1\_1}} \\ \sigma^2_{Mgt1} &= \left(\sigma^2_{\mathrm{T}_{M0\_0}} + \mu^2_{\mathrm{T}_{M0\_0}}\right)P_{\mathrm{T}_{M0\_0}} + \left(\sigma^2_{\mathrm{T}_{M0\_1}} + \mu^2_{\mathrm{T}_{M0\_1}}\right)P_{\mathrm{T}_{M0\_1}} \\ &\quad + \left(\sigma^2_{\mathrm{T}_{M1\_1}} + \mu^2_{\mathrm{T}_{M1\_1}}\right)P_{\mathrm{T}_{M1\_1}} - \mu^2_{Mgt1} \end{aligned} \tag{8}$$

However when $|m| < 1$, the output product will never be a $Num1$ if the input is a $Num0$ which means Case 3 never happens. The probabilities for each case can be found from Figure 8(b) through integration. Once again, the output truncation error mean and variance when $|m| < 1$ are given by (9). As before, if rounding is performed, the error mean is zero and the error variance is calculated with zero error mean.

$$\begin{aligned} \mu_{Mlt1} &= \mu_{\mathrm{T}_{M0\_0}}P_{\mathrm{T}_{M0\_0}} + \mu_{\mathrm{T}_{M1\_0}}P_{\mathrm{T}_{M1\_0}} + \mu_{\mathrm{T}_{M1\_1}}P_{\mathrm{T}_{M1\_1}} \\ \sigma^2_{Mlt1} &= \left(\sigma^2_{\mathrm{T}_{M0\_0}} + \mu^2_{\mathrm{T}_{M0\_0}}\right)P_{\mathrm{T}_{M0\_0}} + \left(\sigma^2_{\mathrm{T}_{M1\_0}} + \mu^2_{\mathrm{T}_{M1\_0}}\right)P_{\mathrm{T}_{M1\_0}} \\ &\quad + \left(\sigma^2_{\mathrm{T}_{M1\_1}} + \mu^2_{\mathrm{T}_{M1\_1}}\right)P_{\mathrm{T}_{M1\_1}} - \mu^2_{Mlt1} \end{aligned} \tag{9}$$

Table 6. Evaluation of *truncation* error models with DFX format 16_19_12 and 16_17_9

16_19_12	Estimated Error		Actual Error		Difference	
	Mean	Var.	Mean	Var.	Mean	Var.
Encoder	-5.40e-05	5.81e-09	-5.39e-05	5.78e-09	0.20%	0.50%
Adder	-2.71e-04	8.09e-08	-2.77e-04	8.23e-08	-2.30%	-1.74%
Multiplier	-1.57e-04	3.40e-08	-1.55e-04	3.38e-08	1.35%	0.67%
16_17_9	Estimated Error		Actual Error		Difference	
	Mean	Var.	Mean	Var.	Mean	Var.
Encoder	-1.68e-04	1.85e-07	-1.68e-04	1.86e-07	0.17%	-0.68%
Adder	-1.06e-03	3.16e-06	-1.07e-03	3.18e-06	-0.97%	-0.48%
Multiplier	-5.56e-04	1.20e-06	-5.58e-04	1.21e-06	-0.40%	-1.20%

Table 7. Evaluation of *rounding* error models with DFX format 16_19_12 and 16_17_9

16_19_12	Model Var.	Actual Var.	Difference
Encoder	2.18e-09	2.17e-09	0.52%
Adder	1.36e-08	1.38e-08	-1.75%
Multiplier	8.85e-09	8.81e-09	0.50%
16_17_9	Model Var.	Actual Var.	Difference
Encoder	5.32e-08	5.33e-08	-0.27%
Adder	4.04e-07	4.08e-07	-0.86%
Multiplier	2.40e-07	2.42e-07	-0.78%

4.6. Error Model Evaluation

For verification of error models, audio samples were used as an input sample. The error is the difference between the estimated output and actual (double precision) output. Tables 6 and 7 shows that with two different DFX formats, the truncation and rounding error models are capable of providing error estimates that are within $\pm 3\%$ of the actual error.

5. CASE STUDY

A 159-tap transposed direct form FIR filter for both DFX and floating-point were implemented using Xilinx Virtex II Pro XC2VP70-6ff1517. The DFX designs were made using the rounding modules shown in Section 3 while the floating-point designs were made using floating-point library by [6]. Table 8 shows the size, latency and *signal to noise*(SNR) performance of these filters with different word-lengths and the best scaling were used for each word-length. SNR is the ratio of the desired output power over the noise power. D1-3 are DFX designs and P1-3 are floating-point designs. For designs with the same word-length, DFX designs are about 5 times smaller and 3.1 times faster than an equivalent floating-point design. Furthermore, the SNR performance of floating-point designs are about 7dB less than DFX.

The SNR of the DFX designs were predicted using the error models mentioned in the previous section. Provided that all the cross-correlations between the individual errors injected can be accounted for, the predicted SNR are found to be within 3% of the actual SNR as shown in Table 8.

Table 8. FIR filter size, latency and SNR performance.

Filter Type	Design	Format	Size (Slices)	Clock (MHz)	SNR (dB)		
					Actual	Est.	Diff.
DFX	D1	14_17_12	4,100	88.4	41.0	40.2	-2.1%
	D2	16_19_14	4,613	90.4	53.0	52.2	-1.5%
	D3	18_21_16	5,085	80.6	64.9	64.2	-1.0%
Floating -Point	P1	14b M7 E6	21,186	27.4	33.1	n/a	
	P2	16b M9 E6	22,858	26.7	45.0		
	P3	18b M11 E6	25,102	27.5	57.1		

6. CONCLUSION

As a prerequisite to automating the design process of DSP systems using our new DFX data representation, an accurate and reliable error model of DFX arithmetic modules has been developed. A single profiling simulation run is all that is required to obtain the probability distribution tables necessary to perform error estimation. In a system context, the characteristics of the output error can be found provided that the cross correlations between the errors injected by each arithmetic module is known.

Future work will include the exploration of multiple word-length designs using DFX and the optimisation of DFX design for area, accuracy and speed.

7. REFERENCES

[1] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, "Analysis of high-performance floating-point arithmetic on FPGAs," in *Proceedings 2004 IEEE International Parallel and Distributed Processing Symposium*, April 2004, pp. 149–156.

[2] K. D. Underwood and K. S. Hemmert, "Closing the gap: CPU and FPGA trends in sustainable floating-point BLASperformance," in *Proc 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*, April 2004, pp. 219–228.

[3] C. T. Ewe, P. Y. K. Cheung, and G. A. Constantinides, "Dual FiXed-Point: An efficient alternative to floating-point computation," in *Field Programmable Logic and Application: 14th International Conference, FPL 2004*, Leuven, Belgium, August 2004, pp. 200–208.

[4] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital signal processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 957–976, 1972.

[5] J. Kontro, K. Kalliojrvi, and Y. Neuvo, "Floating-point arithmetic in signal processing," in *Proceedings 1992 IEEE International Symposium on Circuits and Systems*, San Diego, CA, USA, May 1992, pp. 1784–1791.

[6] J. Detrey and F. de Dinechin, "A VHDL library of parametrisable floating-point and LNS operators for FPGA," June 2005. [Online]. Available: http://perso.ens-lyon.fr/jeremie.detrey/FPLibrary/

[7] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Truncation noise in fixed-point SFGs," *Electronic Letters*, vol. 35, no. 23, pp. 2012–2014, 1999.