Zero-Latency Datapath Error Correction Framework for Over-Clocking DSP Applications on FPGAs

Rui Policarpo Duarte, Christos-Savvas Bouganis
Department of Electrical and Electronic Engineering
Imperial College London
SW7 2AZ London, U.K.
Email: r.duarte09@imperial.ac.uk, christos-savvas.bouganis@imperial.ac.uk

***Abstract*—Errors in the datapath of digital systems usually come with a cost that can be very expensive, either as a consequence of uncertain functionality, or extra resources required to implement mitigation mechanisms, and extra latency to recover from errors. In this work we propose and demonstrate a novel framework which allows to recover from timing errors on a DSP application under extreme over-clocking without adding extra latency into the circuit. Demonstration of the proposed framework on a real-life image processing problem shows an improvement, on average, of 20 dB over typical implementations for doubling of the operating clock frequency.**

I. Introduction

The continuous increase in demand for real-time Digital Signal Processing (DSP) applications that require implementations of arithmetic operators with very high throughput and low-latency, has led engineers to over-clock their designs [1], taking advantage of the gap in performance between the predicted maximum frequency provided by the vendors' models in the synthesis tools and the actual capabilities of the device, at risk of producing errors that can "break" the application.

Field-Programmable Gate Arrays (FPGAs) are often sought to implement such systems because of their characteristics: specialised embedded blocks with high-performance, customised datapath, reconfigurability, parallelism, reduced size and power consumption. Even when the performance improvement offered by this technology is not enough, and the algorithm to be implemented doesn't cope with deep pipelined implementations of the arithmetic units, an alternative avenue to increase performance is to operate in the error-prone regime. Thus, in order to have practical results, error resilience techniques have to be in place. One of them, known as Razor [2], allows to recover from an error condition in the datapath, at the expense of time redundancy. Another one, Reduced-Precision Redundancy (RPR), as been proposed in [3] as a mean to provide an approximate result to contain the error whenever the result from the original system is beyond a user-defined threshold.

The objective of the proposed method is to provide the arithmetic units with resilience from variation errors, due to infringement of timing constraints on their critical-paths, when the design is under variation of its operating conditions, without introducing latency penalty. Results show benefits of the proposed method against typical implementations, while performing almost at twice the clock frequency reported by the synthesis tool. Furthermore, the proposed framework is applied in a linear projection application, which is an algorithm widely used in many DSP applications with near real-time requirements. The proposed RPR architecture doesn't make a distinction between different arithmetic architectures, therefore it can be implemented in technologies alternative to FPGAs.

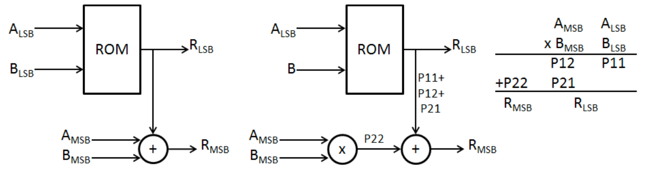

Fig. 1. ROM-based fast adder (left) and multiplier (right).

II. Background

In the quest to further accelerate the performance of computations in reconfigurable devices, designers place great efforts in reducing the length of the carry chain in the main arithmetic operators such as adders and multipliers. Taking advantage of the embedded memories in modern FPGAs, it is possible to tradeoff memory resources for performance. On this account, Fig. 1 presents architectures for fixed-point adder and multiplier. The main limitation of such approach is that such an approach doesn't scale well with the wordlength of the operators. E.g. a 8-bit unsigned multiplier with 4 bits assigned to both MSB and LSB produces 14 bits at the output of the ROM. Thus, it requires a ROM size of $14 * 2^{(4+8)} = 57344$ bits, corresponding to 11.4% of the M9K blocks available on a EP3C16 FPGA from Altera.

RPR is based on a truncated version of the unit considered, and is computed in parallel with the original unit. In case the absolute difference between the outputs of the original and the reduced units is above a threshold specified by the user, then the output of the reduced unit is placed at the output of the RPR unit instead of the value from the original unit. Since there's always a result present at the output of the RPR unit, it is advantageous to use in architectures that don't tolerate stalls in their datapath. Contributions on RPR, from [3], [4], follow the architecture depicted in Fig. 2.

The main advantage of this method is the fact that it allows to keep the throughput of the unit constant, but it requires the insertion of, at least, an extra clock cycle in the datapath. This can be problematic, or even prohibitive, in circumstances where the algorithm doesn't allow further pipelining due to the cost in extra resources, or penalty in the quality of the results.

978-1-4799-5944-0/14/$31.00 ©2014 IEEE

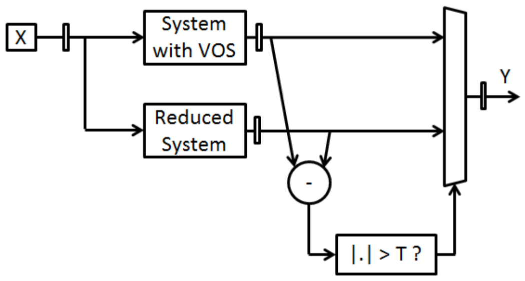

Fig. 2. Typical RPR architecture applied to a system.

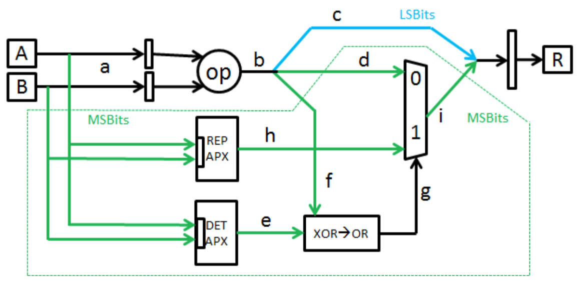

Fig. 3. Proposed RPR architecture applied to a generic combinatorial operator.

The research presented in this work aims to close this gap by proposing a novel framework that allows to add resilience to arithmetic units by limiting the errors at their output.

If the architecture presented in [3], [4] were to be modified to produce results within a single clock cycle, without registers at the output of the original and reduced units, the delay of the most critical-paths would be the delay of the original arithmetic unit plus the delays of the subtracter, comparator and multiplexer, forcing it to operate at a much lower clock frequency than the original unit.

The proposed framework intends to lift this limitation by addressing the role of the components contributing to the delay of the new critical-paths, hence reducing them to a minimum. In order to achieve this, a new architecture was derived and the possible design choices are automated through a framework that targets to minimise the over-clocking timing errors at the output of the device under various objective functions.

III. Reduced Precision Redundancy Framework

Whenever a data-path is clocked beyond the limit specified by the synthesis tool, it is likely that the arithmetic operators will produce wrong results in their outputs. The research here presented aims to close this gap by proposing a novel framework that allows to over-clock the arithmetic operators, within a single clock cycle, while limiting the magnitude of the errors at the output of the arithmetic operators.

The proposed framework relies on an operation similar to existing RPR schemes [3] as it creates a circuit that encapsulates the original unit, and replaces it in the system. Yet, it distances itself from previous works as it enjoys features that are specific for throughput and latency critical applications, and relies on a new RPR architecture without latency penalty.

The new framework borrows the idea of replacing a set of Most Significant Bits (MSBs) at the output of the arithmetic unit with an approximation, when an error is detected, but it proposes new methods to: a) detect timing errors and b) produce approximations of the correct results. Usually, RPR schemes target a set of MSBs of arithmetic operators, as they often hold the paths with the longest delay.

In terms of operation, the user specifies the design budget, in terms of FPGA resources, and the framework provides the solutions for possible implementations of RPR units. In general, this architecture scales to other wordlengths, and (sets of) arithmetic operators, while computing them within a clock cycle.

A. Architecture

In contemplation of the aforementioned constraints a new architecture has been devised. The two main novelties in this architecture are: a) the use of M9K Block RAMs (BRAMs), as Read-Only Memorys (ROMs), to hold the results for the approximation functions, instead of the truncated implementation of the operator; and b) a bit-wise comparison (*XOR→OR*) instead of a subtraction followed by a comparison with a user-defined threshold to detect the presence of errors.

Fig. 3 shows the proposed architecture applied to a generic combinatorial unit *op* (original unit). *A*, *B* and *R* are the inputs and output of the RPR unit, respectively. Identifier *a* refers to the input arguments and *b* to the result of the original operator. The remaining identifiers refer to the paths added by the RPR. The framework includes ROM blocks to provide approximations used in error detection (*DET APX*) and correction (*REP APX*). It also includes a combinatorial block, responsible for the indication of a mismatch in MSBs, named *XOR→OR*, and a multiplexer to select which value pass to the output. The output of the detection approximation (*DET APX*) is identified with *e*. This signal holds the MSBs to be used in the comparison with the MSBs at output of the original unit, signal *f*. The output of the replacement approximation (*REP APX*) is identified with *h*. This signal holds the MSBs to be used in the replacement, or correction, of the MSBs at output of the multiplexer (signal *i*).

The adoption of ROMs and bit-wise comparison (*XOR→OR*) leads to savings in terms of delay between the input port and the output of the multiplexer. Forasmuch as the gap in delay between the output of the approximation and the original result increases, it allows to push the clock frequency even further, as illustrated in Fig. 4. This figure shows the maximum clock frequencies for the original arithmetic unit (STD) and the RPR unit, with the proposed architecture, and their operating regimes: error-free/expected result (green/left), error-prone/approximated result (orange/middle) and error/uncertain result (red/right). It also shows the delays that contribute to the maximum clock frequencies of RPR units. Original units can operate at higher clock frequencies than the RPR unit but once they go beyond their limit, results at their output will be unpredictably incorrect. On the other hand, results at the output of the RPR unit will be incorrect up to a certain amount of error, defined by the datapath engineer. The clock frequency of the RPR unit will be limited by the redundant circuitry, inversely proportional to the delay of the elements in the approximation's critical-path.

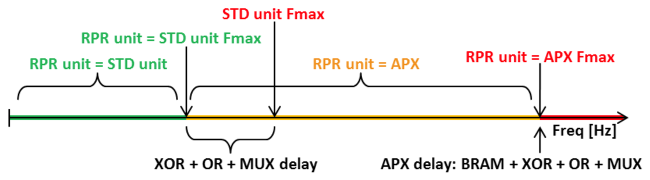

Fig. 4. Illustration of the maximum clock frequencies, and delays, for standard and proposed RPR units and their operating regimes: error-free/expected result (green), error-prone/approximated result (orange) and incorrect result (red).

A set of MSBs were chosen to be compared and replaced, instead of Most-Prone-to-Error-Bits[1], because usually they are the signals with the critical-path in the original unit. The Least Significant Bits (LSBs) are never corrected, hence they are the same as at the output of the original unit. MSBs are compared via a bit-wise XOR followed by an OR of all resulting bits (signal g). Whenever the MSBs match, the output of this block is 0, and 1 if at least one bit differs. In the first case, at the output of the multiplexer (signal i) will be the MSBs from the original unit (signal d). Otherwise, the multiplexer's output will be passed to the output of the replacement detection table (signal h). Moreover, this new architecture is focused on minimising timing errors on datapaths. Thus, it *shows* the approximation result at the output of the multiplexer *while* the MSBs at the output of the original unit *don't match* the MSBs from the detection approximation.

Given the aforementioned, the proposed framework not only allows to specify different word-lengths for the inputs and outputs of the approximation ROMs, but it also allows to use different approximation functions for detection and replacement. The approximation functions in the ROMs depend on the arithmetic operator used, and they are automatically created by the proposed framework to minimise the objective function.

On the grounds of possible implementations using the proposed architecture, a nomenclature is proposed to help identifying the word-lengths of key elements: original unit, detection and replacement ROMs. The syntax is as follows, being iWL and oWL the input and output word-lengths, and Ori, Det and Rep relate to Original, Detection and Replacement respectively:

Ori iWL : Ori oWL / Det iWL : Det oWL / Rep iWL : Rep oWL

Furthermore, to distinguish the different RPR schemes from each other, the following prefix is added to the above nomenclature, which is adopted throughout this work:

- LUT-SUB - previous RPR; approximation computed from a truncated arithmetic unit implemented using Look-Up Tables (LUTs)/Logic Elements (LEs); an error is detected from the absolute difference between the output of the original unit and the approximation;
- ROM-XOR - proposed RPR; detection approximation is retrieved from a ROM, and an error is detected from bitwise comparison with the MSBs from the original unit, in case of mismatch.

Additionally, it is possible to consider other combinations for the RPR architecture, if the design constraints impose them, i.e. ROM-SUB or LUT-XOR, but they aren't covered in this work.

B. Approximation Functions

At the heart of the proposed framework is the (offline) computation of the approximation functions that are stored in the ROMs. These approximation functions try to accurately represent the MSBs of the original unit to minimise the approximation errors. The strength over other implementations is the possibility to implement any approximation, instead of relying on a truncated version of the arithmetic unit computed in parallel.

The usage of a ROM to store the approximations concedes the opportunity to implement any function while the cost, in terms of hardware resources, is constant regardless of the function being implemented. The proposed framework is able to support different types of approximation functions from the truncated operators: truncated operation, linear approximation, or other approximations derived from different objective functions, e.g. mode values of the MSBs from the expected results and values that minimise error variance. Moreover, the adoption of FPGAs to implement such method offers the possibility to load different approximations in run-time, without downloading a new, or partial, bitstream into the device.

The linear approximation of a generic arithmetic operator ($\star$), with truncated operands, can have one or more unknown coefficients (i.e. k_A, k_B, l, m_A and m_B with $k_A, k_B, l, m_A, m_B \in \mathbb{Z}$).

$$A = \sum_{i=N-iWL}^{N-1} a_i.2^i, \quad B = \sum_{i=N-iWL}^{N-1} b_i.2^i \tag{1}$$
$$X = (A * k_A + m_A) \star (B * k_B + m_B) + l \tag{2}$$
$$E = a \star b - X \tag{3}$$

In above expressions iWL_A, iWL_B and oWL are the word-lengths of the inputs and output of the ROMs. These coefficients exist to overcome the distortions imposed by the approximation computed from truncated input values, such as negative bias. In this direction, the framework exhaustively searches the function's domain for the coefficients in the linear approximation function which results in the minimisation of the objective function, between the expected and the approximation MSBs.

The proposed architecture uses two approximations in parallel. The approximation for detection (*DET APX*) of errors is treated separately from the approximation to replace (*REP APX*) results detected as a mismatch. This is derived from the fact that information is missing from an approximation computed from truncated input arguments. Therefore, a bit-wise comparison between the expected result, from the original unit, and approximation unit will identify mismatches. This

[1]Most-Prone-to-Error bits are defined as the bits that are more likely to fail, and they may not be the MSBs. They are determined from characterisation of the arithmetic unit with a specific data set. Arithmetic units with different architecture, and input data with different distributions, are likely to exhibit different levels of error across their output bits [5], [6] .

is not desirable as it will identify correct results as incorrect and have their MSBs replaced with the values from the approximation. On this account, in detection it is desirable to use as many bits as possible to obtain the least number of false positive mismatches. On the other hand, if the original unit is producing incorrect MSBs it is likely that some of the following bits will also be incorrect. Hence it is advantageous to replace more MSBs than the ones considered in the detection block. Nevertheless, if the MSBs are to be replaced by an approximation, then the impact of the lack of information will correspond to the approximation error at the output of the RPR unit, similarly to the existing RPR schemes. To minimise this undesirable artifact, it is considered using another approximation function, to be stored in the replacement ROM, with contents different from the the detection ROM, either by using a different approximation function, different word-lengths, or both.

This is exemplified with a 4-bit multiplication with a ROM to hold the MSBs for two different approximations using truncated operators, and compare them against the expected result. The example considers a ROM with 2-bit per operand, and one output bit for the approximation MSB, with contents generated by (6).

$$apx(a, b, m, l) = (a + m) * (b + m) + l \tag{4}$$

Table I presents the results of 4-bit multiplications using the original and truncated operands, and its MSBs. It also presents the results and the MSBs for two approximations, using different approximation coefficients in (2). This example shows that even the MSB from the truncated approximation deviated from the expected result, allowing it to be replaced with another approximation would actually produce the expected, and correct, result.

The resources needed to store an approximation function in ROM depend exclusively on the word-length of the inputs and the output considered for the approximation, regardless of the function being used. Conversely, the framework determines which word-lengths are possible to use given a limited resource budget. The total number of memory bits is given by: $oWL(2^{(iWL_A+iWL_B)})$. Ideally it would be enticing to have all possible values in the ROM, instead of truncating the operand's word-length, but the resources needed would cause such implementation unfeasible. Moreover, in the particular case of using only one ROM, to hold both detection and replacement approximations, the result would be the same as using always the LSBs from the original unit and replace the MSBs with the approximation. This would simplify the design by eliminating the multiplexer and the extra ROM in the design, thus increasing the maximum clock frequency of the unit. Nevertheless, such scheme is not presented in this work even though the proposed framework could easily be adapted to generate the approximations and evaluate its error/area tradeoff.

	A	B	l	m	Result	MSB
Original	1001	1111	-	-	10000111	1
Apx. 1	1000	1100	0	0	01100000	0
Apx. 2	1000	1100	15	1	10000100	1

TABLE I. RESULTS OF 4-BIT MULTIPLICATIONS USING THE ORIGINAL AND TWO DIFFERENT APPROXIMATIONS FOR THE TRUNCATED OPERANDS.

Scope	Objective Function
Min. Hamm. dist.	$max\{i\} : \forall i \in oWL, ori(i) = apx(i)$
Min. error variance	$min\{var(\sum_{i=oWL} ori(i).2^i - \sum_{i=oWL} apx(i).2^i)\}$
Min. mean error	$min\{\sum_N (\sum_{i=oWL} ori(i).2^i - \frac{1}{N} \sum_{i=oWL} apx(i).2^i)\}$
iWL, oWL: input and output word-lengths; ori: result from the original unit; apx: result from the approximation function; N: number of values tested	

TABLE II. OBJECTIVE FUNCTIONS FOR ERRORS AT THE OUTPUT OF THE RPR UNIT.

C. Objective Function Minimisation

In some cases, when it's not possible to faithfully generate the MSBs by the approximation ROMs, i.e. large input wordlength in the original operator and small input wordlength in ROM; it may be desirable to specify different objective functions for the generation of the approximations by the framework.

At present, the framework supports different objective functions, derived from different error metrics between the expected and the approximation results. Table II summarises these functions for 3 application scopes.

The search for the approximation coefficient values that minimise the objective function is as follows. A value is assigned to k, l and m, from an arbitrary set, and all possible values for the operands in the original adder, and in the approximation function, are tested to evaluate the error from the approximation function. The values returned for the approximation coefficients for both detection, and replacement, approximations are the ones that minimise the objective function.

IV. ROM-XOR RPR ARITHMETIC OPERATORS

The proposed framework has been introduced for generic arithmetic units. The specifics on how different basic arithmetic units are supported by the proposed framework are explained in this section. The proposed framework is demonstrated for the application of redundancy on arithmetic units usually used in DSP designs, namely adders, multipliers and multiplier-accumulators. Furthermore, the demonstration is conducted adopting the objective function that minimises the mismatches in the Hamming distance.

Although there are many possible synthesis implementation variants for each unit, the proposed framework is transparent to them. Moreover, the proposed framework can be extended to different arithmetic operators and scaled to different wordlengths.

A. Adder

The adder is one of the building blocks of arithmetic circuits. Even though it's a simple block, for large wordlengths, carry propagation becomes the bottleneck in terms of throughput. For this operator the linear approximation function considered is as follows:

$$apx(a, b, k) = a + b + k \tag{5}$$

iWL	oWL	ROM bits	k_{det}	k_{rep}	Errors
5,5	1	2048	8	0	0
5,5	2	4096	8	0	0
6,6	1	8192	4	0	0
6,6	2	16384	4	0	0
7,7	1	32768	2	0	0
7,7	2	65536	2	0	0

TABLE III. CORRECTNESS OF THE MSBS FOR DIFFERENT COEFFICIENTS OF APPROXIMATIONS FOR 8:9/IWL:OWL/IWL:OWL ROM-XOR RPR ADDITION, AND THE TOTAL NUMBER OF ROM BITS USED.

where a and b are the truncated input arguments and k is a bias to compensate for the truncation of the input operands.

The proposed framework optimises two approximation functions, with two approximation coefficients k_{det} and k_{rep}, simultaneously. To illustrate the application of the framework, a 8-bit unsigned adder is considered. Fig. 5 shows the output of the objective function in the values of the MSBs for a particular case of a 8:9/5:2/5:2 ROM-XOR RPR adder.

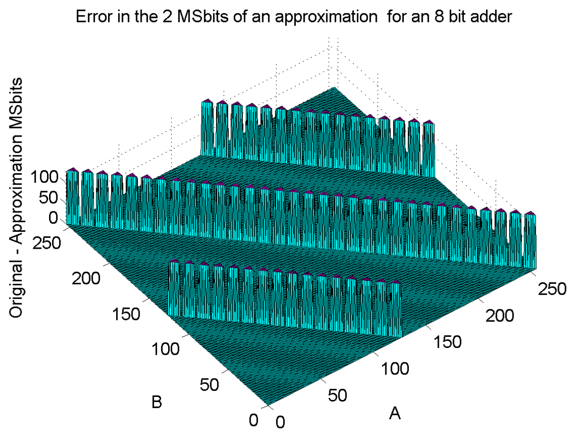

Fig. 5. Difference in the MSBs between the detection approximation and the expected result for an 8-bit adder (Hamming distance).

Table III shows the linear approximation coefficients to produce the same MSBs as the original unit for different ROM sizes. The output of the RPR 8-bit adder is the same as the 8-bit original adder, when operating in the error-free regime.

B. Multiplier

Like addition, multiplication is widely used and present nearly in all DSP systems. Synthesis of the circuit to implement this unit has more, and longer, paths than the adder, and often holds the critical-paths of DSP designs. The only difference in the architecture, when compared to the RPR adder, is the original arithmetic unit. For this arithmetic operation, a new approximation function is studied to minimise the objective function. The function chosen to approximate the multiplication is given in (6):

$$apx(a, b, m, l) = (a + m) * (b + m) + l \tag{6}$$

In this case a and b are the truncated inputs, and m and l are the approximation function coefficients.

Using two distinct approximations, for detection and correction, allows to set the correct MSBs whenever the detection detects them as wrong. Fig. 6 shows the discrepancy in the detection approximation of the MSBs for a particular case of a 8:16/5:2/5:2 ROM-XOR RPR multiplier. Table IV shows the linear approximation coefficients to produce the same MSBs as the original unit for different resource budgets.

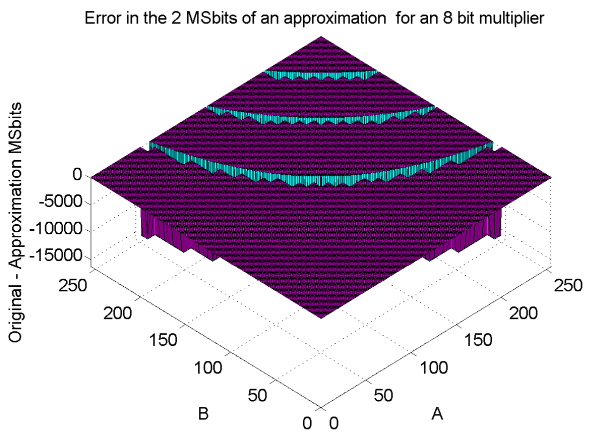

Fig. 6. Difference between a detection approximation and the expected MSbits at the output of the multiplier.

iWL	oWL	m_{det}	l_{det}	m_{rep}	l_{rep}	Errors
5,5	1	1	2015	0	0	0
5,5	2	3	1767	0	0	0
6,6	1	1	723	0	0	0
6,6	2	1	887	0	0	0
7,7	1	0	356	0	0	0
7,7	2	0	444	0	0	0

TABLE IV. CORRECTNESS OF THE MSBS FOR DIFFERENT APPROXIMATION COEFFICIENTS IN AN 8:16/IWL:OWL/IWL:OWL ROM-XOR RPR MULTIPLICATION.

C. Multiplier-Accumulator

Implementing a Multiply-Accumulator (MAC) using the proposed RPR architecture involves adding the RPR circuitry while having data computed within one clock cycle. This operator exhibits extra complexity, when compared to the previous operators, as the approximation functions have to have a new intermediate result to regard preservation of the circuit's functionality. Fig. 7 shows the block diagram of a folded multiply-accumulate unit with RPR. The RPR block is attached to the datapath in a similar pattern as the previous units, but uses a new approximation function, shown in (7):

$$apx(a, b, m, l, r) = (a + m) * (b + m) + r + l \tag{7}$$

In this approximation function, a, b, and r are the truncated input operands, and m and l the approximation coefficients which are determined by the framework to minimise the objective function.

Alternatively, the multiply-accumulate unit with RPR can be constructed at the expense of the independent arithmetic RPR units. In this case, each arithmetic operator (adder + multiplier) has their own RPR block. Moreover, when there's a significant gap in the delay of the critical-paths in the adder and in the multiplier, or when it's foreseen that one of these units will never experience timing violations in their paths, it

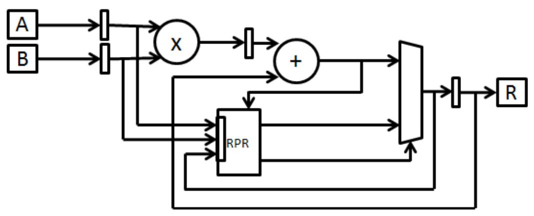

Fig. 7. Diagram of a rolled multiply-accumulate circuit with RPR.

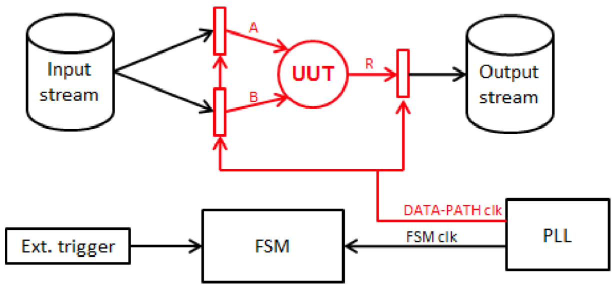

Fig. 8. Block diagram of the test circuit for RPR units under variation.

may be preferable to apply the RPR block only to the most prone to error unit, hence saving circuit resources.

V. Performance Evaluation

To evaluate the proposed framework ,its performance is compared against existing implementations of stand-alone arithmetic units, and on a DSP application, with and without the proposed RPR. It uses the default synthesis implementation for the arithmetic operators using the vendor tool.

The stimulus used by the evaluation is a pseudo-random vector with 20k samples for the stand-alone arithmetic units, and a set of 50x40 pixels grayscale images, from [7], for the linear projection applications. In the implementation, data is represented using 9-bit sign-magnitude representation.

The aforementioned performance for each circuit is compared in terms of clock frequency, circuit resources and errors at the output. For precise measurements the operating conditions of the device were kept constant, at 20 degrees Celsius, through the usage of a cooling element on top of the FPGA device and 1200 mV from an external power supply. All tests were carried on a EP3C16F484C6 Cyclone III FPGA from Altera [8], on a DE0 board from Terasic [9].

To facilitate the implementation, and minimise the influence of external circuitry in the evaluation of the design, all designs were placed inside a test circuit. In addition, this test circuit promotes the reduction of placement and routing variation in the design, to which changes the delays of the most critical-paths, thus making the circuits under variation to produce different error patterns.

The circuit to test the proposed RPR arithmetic units under variation is presented in Fig. 8. The Finite State Machine (FSM) controls the test execution, and provides the unit under test (UUT) with a constant stream of data. The datapath of the circuit, which holds the unit under test, is highlighted in red. This part of the circuit is clocked at twice the clock frequency of the FSM.

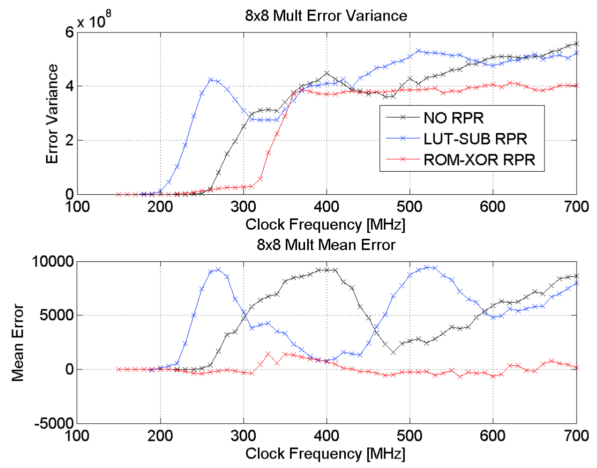

Fig. 9. Error variance and mean error at the output of an 8-bit bit unsigned LUT-based multipliers at different clock frequencies.

Multiplier	NO RPR		ROM-XOR	
	LEs	mW	LEs	mW
8-bit	101	1.23	206	8.43
16-bit	345	5.08	452	12.28
32-bit	1213	24.31	1321	31.51
48-bit	2605	54.26	2722	61.46

TABLE V. Resources and power taken by the different multiplier implementations.

A. Adder

Implementation of adders on FPGAs is facilitated by their architecture which offers dedicated carry propagation lines, thus avoiding routing through the LEs. Because of its simplicity and reduced number of LEs it is the fastest arithmetic operator possible to implement on an FPGA.

The proposed ROM-XOR RPR adds extra complexity to the adder circuit and its maximum clock frequency. Hence, unviable to compete against the clock frequency of a typical adder implementation. The maximum clock frequency results for a 16-bit adder without RPR is 411 MHz, whereas the 16-bit RPR adder can only be clocked up to 248 MHz. In terms of performance, it equalises for 42-bit wordlengths.

B. Multiplier

Three 8-bit multipliers were implemented for performance comparison: no RPR, LUT-SUB RPR and ROM-XOR RPR. Fig. 9 shows the variance and mean of the error between the expected value and the value read from the board for each multiplier implementation. The maximum throughput of the ROM-XOR-RPR scheme is close to the multiplier without any redundancy, and it performs better than any other design under extreme over-clocking, e.g. 300 MHz. Only above 340 MHz the ROM-XOR-RPR multiplier exhibits the same variance as the other multipliers. Notwithstanding, its mean error remains close to zero. Table V and **??** shows the maximum clock frequencies, the resources occupied and the power consumed by the different multiplier implementations.

Forasmuch as the new architecture for the ROM-XOR RPR as been proposed, there's a limitation in using the output registers in the embedded multipliers, as the new architecture firstly makes a decision in which approximation value to use and then registers it. This represents an increase in the delay of the critical-path, through the embedded multipliers, from 2.726 ns (using internal registers) to 4.4 ns (using external registers), for a Cyclone III FPGA from Altera.

C. Multiply-Accumulator

The synthesis of a MAC unit reveals that the delay of the critical-path of the 16-bit adder is greater, by more than twice the delay of the critical-path of the 8-bit LUT-based RPR multiplier. This gap in maximum clock frequency between the two RPR units, suggests that the version of the MAC unit using discrete adder and multiplier can be achieved with redundancy only in the multiplier. This imposes that the maximum clock frequency can't exceed the maximum clock frequency of the adder for error-free operation. Consequently, the previous RPR multiplier can be reutilised to implement the MAC unit with RPR.

D. Linear Projection Designs

Linear projection, also known as Karhunen-Loeve transformation (KLT) [10], is usually used for compression of data. In the evaluation, the comparison is made between a 8:16/5:3/5:3 ROM-XOR RPR, a LUT-SUB RPR with an approximation computed from the 5 MSBs, and the implementation of the linear projection without any redundancy.

The architecture of the circuit to compute the linear projection is a folded MAC, for each projected dimension, which has been presented before. From all possible implementations possible, this was chosen because it is the one minimises the area footprint. In the circuit implementation all inputs are encoded using sign-magnitude with 9 bits. The output of the multiplier is 16 bits unsigned. The word-length of the output increases with the number of accumulation stages (log_2).

Fig. 10 shows the results for the 3 implementations at 270 MHz. The top row shows the expected result, without variation errors. The following rows correspond to the results for: NO RPR, LUT-SUB RPR and ROM-XOR RPR. On top of each image there's the PSNR from the reconstruction of the projected data, on the FPGA, into the original space in software. It's evident that the ROM-XOR RPR creates the linear projections which better recreate the original images and produce the least reconstruction Peak Signal-to-Noise Ratio (PSNR) for all images.

From the results above it's conclusive that a little penalty in the maximum performance for the error-free regime is later retributed when operating in the error-prone regime. The usage of a few extra LEs and BRAMs allows to achieve a clock frequency close to the maximum specified to the embedded multipliers (340 MHz).

VI. Conclusions

This contribution proposes a framework for a novel RPR architecture which is able to perform in systems with high-throughput and without latency penalty. Results show that the benefits of protecting the most error prone units in a DSP design allowed to operate it at almost twice the clock frequency with minimum errors, which overshadows the implementation cost. The presented RPR scheme represents a breakthrough as at present there's no comparable method available.

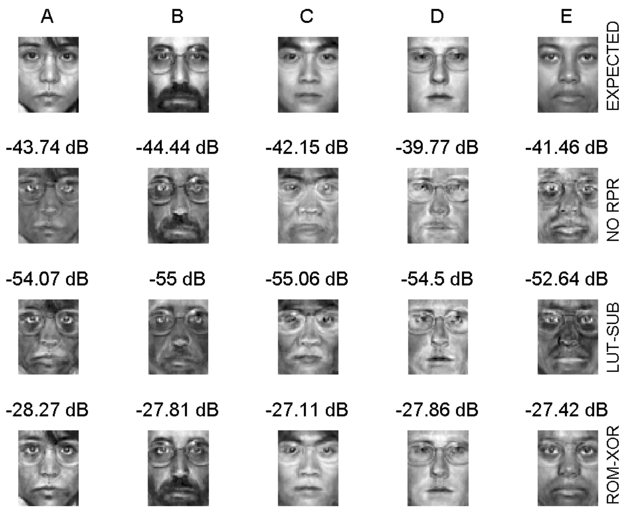

Fig. 10. Reconstructed faces (A-E) in the original space without timing errors (EXPECTED), and computed from the projections collected from implementations of different multipliers architectures (NO RPR, LUT-SUB RPR, ROM-XOR RPR) at 270 MHz. On top of each face there's the corresponding reconstruction error.

References

[1] R. P. Duarte and C.-S. Bouganis, "A unified framework for over-clocking linear projections on FPGAs under PVT variation," in *Applied Reconfigurable Computing (ARC), 2014 10th International Symposium on*, 2014, pp. 49–60.

[2] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec 2003, pp. 7–18.

[3] B. Shim and N. Shanbhag, "Reduced precision redundancy for low-power digital filtering," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, vol. 1, 2001, pp. 148–152 vol.1.

[4] B. Shim, S. Sridhara, and N. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 5, pp. 497 –510, may 2004.

[5] R. P. Duarte and C.-S. Bouganis, "Over-clocking of linear projection designs through device specific optimisations," in *21st Reconfigurable Architectures Workshop (RAW 2014)*, 2014, pp. 9–60.

[6] J. Wong and P. Cheung, "Timing measurement platform for arbitrary black-box circuits based on transition probability," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 12, pp. 2307–2320, Dec 2013.

[7] S. Ekvall, "CODID - CVAP object detection image data base," online, 2014. [Online]. Available: http://www.nada.kth.se/ ekvall/codid.html

[8] Altera. Cyclone III device handbook. Online. Altera. [Online]. Available: http://www.altera.co.uk/literature/hb/cyc3/cyclone3_handbook.pdf

[9] Terasic Technologies. (2009) Terasic DE0 board user manual v. 1.3. [Online]. Available: http://www.terasic.com.tw

[10] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 417–441, 1933.