An Automated Design Flow for Approximate Circuits based on Reduced Precision Redundancy

An Automated Design Flow for Approximate Circuits based on Reduced Precision Redundancy

Daniele Jahier Pagliari, Andrea Calimera, Enrico Macii and Massimo Poncino
Dipartimento di Automatica e Informatica
Politecnico di Torino, Turin, Italy
Email: {daniele.jahier, andrea.calimera, enrico.macii, massimo.poncino}@polito.it

***Abstract*—Reduced Precision Redundancy (RPR) is a popular Approximate Computing technique, in which a circuit operated in Voltage Over-Scaling (VOS) is paired to a reduced-bitwidth and faster replica so that VOS-induced timing errors are partially recovered by the replica, and their impact is mitigated.**
Previous works have provided various examples of effective implementations of RPR, which however suffer from three limitations: first, these circuits are designed using ad-hoc procedures, and no generalization is provided; second, error impact analysis is carried out statistically, thus neglecting issues like non-elementary data distribution and temporal correlation. Last, only dynamic power was considered in the optimization.
In this work we propose a new generalized approach to RPR that allows to overcome all these limitations, leveraging the capabilities of state-of-the-art synthesis and simulation tools. By sacrificing theoretical provability in favor of an empirical input-based analysis, we build a design tool able to automatically add RPR to a preexisting gate-level netlist.
Thanks to this method, we are able to confute some of the conclusions drawn in previous works, in particular those related to statistical assumptions on inputs; we show that a given inputs distribution may yield extremely different results depending on their temporal behavior.

I. Introduction

Approximate Computing (AC) is a design paradigm that exploits the error resiliency existing in many application domains to tackle energy consumption in digital systems [1]. The general idea is that, relaxing the constraints on maximum accuracy, it is possible either to reduce a system's complexity or to modify its operating conditions, to improve energy efficiency. Many AC techniques have been proposed in literature, operating at various abstraction levels [2], [3].

Several power-driven AC solutions leverage Voltage Over-Scaling (VOS), i.e., scaling of the supply voltage below the critical value $V_{dd,crit}$, as the underlying technology knob. A popular VOS-based approach is Algorithmic Noise-Tolerance (ANT), an architectural-level technique applicable to arithmetic and DSP circuits, that allows to operate in VOS conditions without reducing the original system throughput [5]. In ANT, the original circuit, called Main DSP (MDSP), is coupled with an Error Control (EC) block that limits the impact of *timing-errors* due to the lowered V_{dd}. The rationale is that, even if the MDSP operates in VOS conditions, only a small percentage of the total paths actually violates the timing constraints. ANT works nicely when the timing errors have large magnitude, as in most arithmetic circuits, where the critical path is associtated to the MSBs. In fact, the EC block can easily identify these events, and when they occur, provide a *good-enough* approximation of the correct result.

In this paper, we focus on one particular instance of the ANT technique, called Reduced Precision Redundancy (RPR) [6], in which the EC contains a reduced-bitwdith replica of the MDSP. RPR has been proved a quite effective embodiment of AC [6]. However, all literature on this subject focuses on single or small families of designs. In these works, the replica is designed using ad-hoc procedures, and no general and automated approaches have been proposed. Moreover, estimation of the output quality after the insertion of the EC block is obtained via statistical analysis. In order to do so, oversimplified assumptions are made on I/O data distribution, and important aspects such as input temporal correlation (i.e. the relation between values of subsequent inputs) are neglected. Finally, the optimization was done only considering dynamic power, and leakage was evaluated only a posteriori; however, the redundancy nature of RPR will obviously increase leakage power, which should be thus part of the cost function [6].

In this work, we propose a new generalized approach to RPR-ANT, that allows to overcome these limitations. By sacrificing theoretical provability in favor of an empirical input-based analysis, and by leveraging state-of-the-art EDA tools, we build a design tool able to automatically add RPR to a preexistent gate-level netlist. The specific contributions provided in this work include:

- We estimate all design metrics using accurate characterizations of real standard-cell libraries. This method provides more accurate estimates of savings and timing degradation with respect to the basic delay and power models of previous works. Moreover, it allows to consider all power components, including the leakage overhead introduced by the EC.
- Thanks to a simulation-based analysis performed on a set of *realistic* input stimuli, we obtain a more faithful estimation of the rate and impact of timing-errors. In fact, we are able to account for temporal correlation and non-trivial distributions, two aspects that cannot be treated by simple statistical models.
- We build the replica starting from an already synthesized gate-level netlist of the MDSP, leveraging standard EDA tools directives for its simplification. Our approach is thus *application-agnostic*, i.e. it does not require the knowledge of the functionality performed by a design. The method is

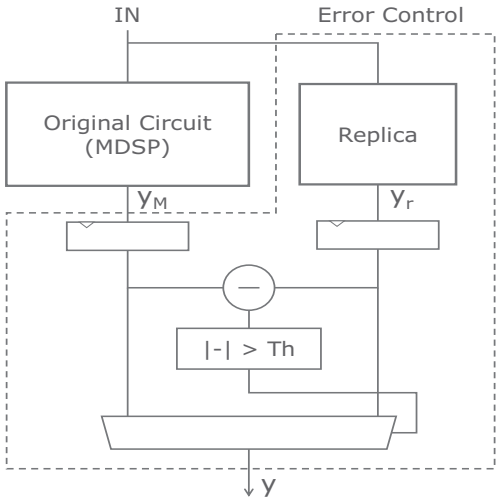


Fig. 1: Reduced Precision Redundancy ANT Block Diagram.

automatically able to discern whether the addition of RPR is suitable or not, and to synthesize the replica configuration that maximizes power savings under a user-defined quality constraint. The entire process is automated, so that no additional coding effort is needed by the user in order to implement RPR on top of an existing design.

We show that, when tested on the same families of designs treated with ad-hoc procedures in previous works, our approach reaches comparable savings. Moreover, we apply it to several architectures that were not considered previously, confirming its generality and flexibility. Globally, we are able to reduce *total power* up to 50%.

II. Background and Related Work

A comprehensive overview of AC techniques can be found in [1], while VOS and its applications to AC are discussed in [4]. Algorithmic Noise Tolerance (ANT) was first proposed by Hegde and Shanbhag in 2001 [5].

The general concept of ANT is shown in Figure 1: the original design (MDSP) is operated in VOS, which results in errors due to input-dependent timing violations. To reduce their impact, the MDSP is coupled with an Error Control (EC) block, that does not attempt to *correct* MDSP errors, but just to *mitigate* them, by providing a good-enough approximation of the correct result. Therefore, the EC block can be made simpler than the MDSP, and can be designed to be error-free (i.e. timing compliant) in VOS conditions. The EC block is composed of two main elements: an *estimator*, which generates the approximate version of the correct output, and a *decider* that produces the global output selecting between MDSP and estimator. In particular, the selection is performed based on the absolute value difference between the MDSP and estimator outputs. If this difference exceeds a given threshold, the estimator output is selected, assuming that a timing-error has occurred in the MDSP. Otherwise, the MDSP result is forwarded to the global output.

As explained in Section I, the rationale behind this scheme is that, for most arithmetic circuits, critical timing paths are related to the computation of the MSBs. Therefore, MDSP outputs generated in correspondence of a timing error will differ significantly from the approximations provided by the estimator.

Two main approaches to ANT are possible, which differ in the structure of the estimator [7]. In *Prediction-based* ANT, an interpolation filter is used to predict the next MDSP output as a linear combination of the recent output history. This scheme works better for narrow-band systems, i.e., with high correlation among subsequent outputs. Its main limitation is that all MDSP outputs, including erroneous ones, are used in the predictor filter. Therefore, prediction accuracy is affected by previous errors, and this may in turn influence the selection in the decider block.

In *Reduced-Precision Redundancy* (RPR) ANT, the estimator is a simplified replica of the original circuit. The most intuitive way to build the replica is by reducing the bit-width of its operands, removing some of the LSBs. With respect to the prediction-based approach, RPR has a larger overhead, but it can be applied to a wider range of systems, with different output bandwidths.

This work focuses on RPR-ANT, because of its greater flexibility. In the rest of this Section we present a recap of the theory behind this approach; a more exhaustive discussion can be found in [6].

With reference to Figure 1, the MDSP output at time n is:

$$y_M[n] = y_o[n] + \gamma[n] = (s[n] + \eta[n]) + \gamma[n] \qquad (1)$$

where y_o is the error-free output, and γ is the component due to timing violations, modeled as an additive noise. For some application domains (e.g. digital communications), y_o can be split into a signal component s and an additional noise component η (e.g. communication channel noise). The replica output is termed $y_r[n]$.

y_M and y_r are subtracted and compared against a threshold T_h. The decision MUX selects between its inputs as follows:

$$y[n] = \begin{cases} y_M[n], & \text{if } |y_M[n] - y_r[n]| \leq T_h \\ y_r[n], & \text{if } |y_M[n] - y_r[n]| > T_h \end{cases} \qquad (2)$$

A desirable property of this architecture is that $y = y_M$ in absence of timing errors. To ensure it, the threshold must be set as:

$$T_h = \max_{\text{all inputs}} |y_o[n] - y_r[n]| \qquad (3)$$

The RPR architecture must be designed so that the global output y satisfies a quality constraint, which is strongly *application-dependent*. A typical quality metric used in many DSP and communication applications is the Signal-to-Noise Ratio (SNR), defined as:

$$SNR_y = \frac{\sigma_s^2}{\sigma_{\gamma_r}^2 + \sigma_\eta^2} \qquad (4)$$

where σ_s^2 is the signal power, σ_η^2 is the channel noise power and $\sigma_{\gamma_r}^2$ is the residual noise power due to timing errors after the decider:

$$\sigma_{\gamma_r}^2 = E[|y_o[n] - y[n]|^2] \tag{5}$$

The quality constraint is satisfied by acting on the VOS supply voltage (V_{VOS}) and on the replica implementation. In particular, since the replica is a reduced width version of the MDSP, the possible implementations vary in accuracy and power/area overheads (both function of the number of removed bits B). In conclusion, the identification of the best RPR architecture can be formulated as an optimization problem with the following objective function:

$$\min_{V_{VOS},B}[P_{rpr}(V_{VOS}, B)] \tag{6}$$

Under the following constraints:

$$\begin{cases} Q(V_{VOS}, B) \geq Q_{target} \\ P_{rpr}(V_{VOS}, B) < P_{mdsp}(V_{nom}) \end{cases} \tag{7}$$

where Q is a generic quality measure (e.g. SNR), Q_{target} is the desired minimum quality constraint, and P_{mdsp} and P_{rpr} are the total power consumptions of the MDSP and of the RPR architecture (including the EC block overhead).

III. Automated RPR Design Methodology

A. Limitations of the basic RPR formulation

While theoretically solid, the formulation of RPR of Section II has a few downsides, that constrain its application to a small number of designs. The main issues are related to the evaluation of the decision threshold and of the quality function. In order to compute T_h with Equation 3, knowledge of the MDSP and replica outputs produced by the entire set of possible inputs is required. Computation of these *output images* is not easy, except for some trivial cases. Moreover, often applications use only a portion of the input domain of a hardware block. Therefore, the theoretical threshold might correspond to an input combination that never occurs during real operation.

Similar issues are also faced in the evaluation of Equation 5, which is required to measure the output quality in the case $Q \equiv SNR$. In fact, the expected value is computed over the *entire range* of error-free and RPR outputs. In addition to that, three more quantities are needed for the evaluation of the SNR: the *distribution* of output values, and the *probability* and *magnitude* of timing-errors. Output values distribution depends, in turn, on all inputs distributions; if those are not elementary (e.g. uniform) and if the MDSP function is not trivial, an analytical evaluation is unfeasible. Error probability and magnitude are required to compute the global RPR output y. In fact, we need to know how often the MDSP output in VOS differs from the error-free output y_o, and how many of those errors are detected by the EC block.

In previous works, the error probability was estimated as the percentage of *static timing paths* that violate the positive-slack constraint under VOS. This percentage was computed based on simple CMOS delay models [6]. However it is known that timing-errors are determined by *dynamic path activations*. Computing the error probability as the number of violating static paths implicitly assumes a uniform distribution of all *input pairs*, which is not realistic for the majority of designs. Similarly, also the error magnitude depends on which paths are activated by a given input sequence, and cannot be estimated correctly with a static analysis.

Notice that these issues are not specific to the SNR, since most standard quality metrics (e.g. PSNR, MED, MSE, etc.) involve at least one of the mentioned quantities.

B. Synthesis/Simulation-Oriented Perspective

Because of the limitations described in Section III-A, applications of RPR based on the theoretical formulation are limited to (1) a small class of designs, and (2) under the assumption of inputs uniformly distributed and completely uncorrelated in the time domain.

The objective of this work is to extend the RPR approach to a much wider family of designs. We propose a methodology and the relative tool to automatically check if RPR is feasible or not for a given MDSP design, and to identify the optimal configuration of parameters (V_{VOS}, B) according to the model of Equations 6 and 7. The tool is *application-agnostic* in that it does not require knowledge of the specific functionality performed by the MDSP.

To address the issues of the theoretical formulation, we need to change the perspective in favor of a more synthesis- and simulation-oriented view. The first improvement that we introduce over existing RPR optimization algorithms is a **more accurate evaluation of design metrics**. We operate on a *gate-level* model of the MDSP, which allows us to accurately estimate timing, power and area with characterized standard-cell libraries rather than simplified models. Our tool uses $N + 1$ *Liberty format* descriptions of the same gate library, corresponding to the nominal supply voltage V_{nom} and to a series of decreasing VOS operating points $[V_{VOS,1}...V_{VOS,N}]$.

The second novelty of our tool is that it **relies on user-provided input stimuli**. It is reasonable to assume that a designer wanting to add RPR on top of an existing hardware has a good knowledge of the inputs that will be applied to the circuit during normal operations. This information can be used to estimate all the quantities required by the RPR optimization (Equations 6 and 7) through simulations. With respect to the theoretical formulation, these estimations are not 100% accurate, being based on a finite-length sequence of inputs. However, this method allows to take into account temporal correlation of inputs and arbitrary data distributions.

Simulation outputs are used to check the effectiveness of RPR for a given MDSP, and to obtain the optimal parameters for the EC block. Then, the tool automatically **synthesizes the RPR architecture using state-of-the-art tools**. In particular, the replica is obtained from the MDSP via netlist manipulation directives.

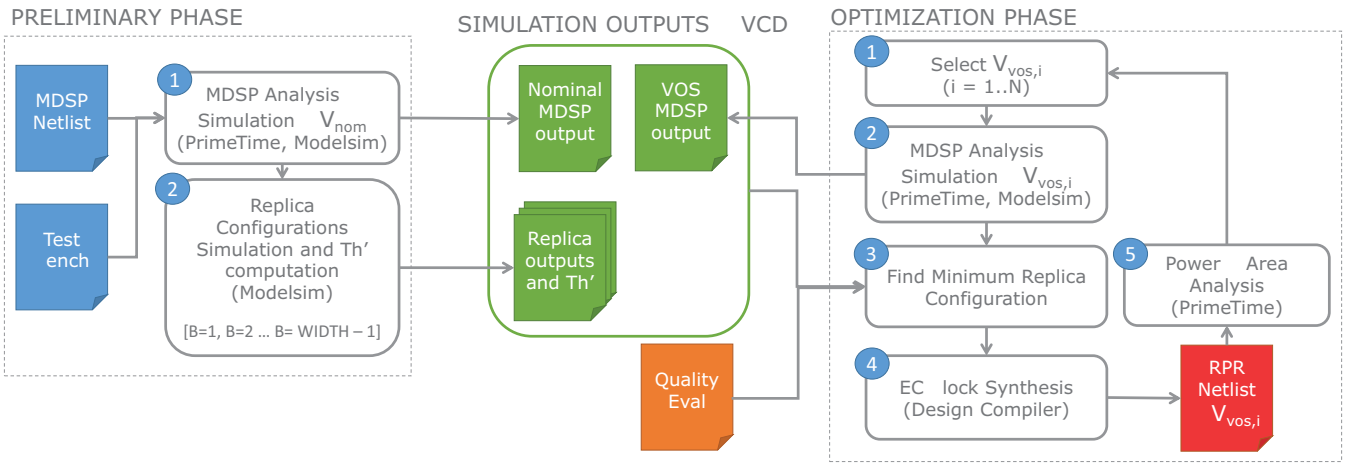


Fig. 2: RPR Automation Tool Flow.

C. Automation Flow

Figure 2 shows the flow of the RPR automation tool. In its description, we will refer to the concept of *input and output buses* to denote a coherent set of interface wires, not necessarily correspondent exactly to a HDL port. Within a bus, wires represent the digits of a binary fixed-point number. The tool receives a list of MDSP input and output buses, each mapped to the corresponding HDL interface signals, via simple *pragmas* inserted by the designer in the netlist.

As mentioned, our optimization relies on the information about input data. This information is provided to the tool in the form of a Verilog or VHDL *testbench* for the MDSP.

It is also reasonable to assume that the designer knows the most appropriate quality metric Q (and the relative desired quality Q_{des}) to use for each MDSP output bus. Therefore, the tool can accept a custom *quality evaluation function*, currently provided in the form of a Perl module. For each output bus, this function receives two arrays of binary words, representing the sequences of error-free and erroneous outputs, and must return the corresponding quality. Standard quality functions (e.g., $Q \equiv SNR$) can be used in place of custom ones.

In summary, the inputs that a designer should provide to the tool are:

- The gate-level netlist of the MDSP, annotated with bus pragmas.
- A testbench generating realistic input stimuli.
- Optionally, a custom quality evaluation Perl module.

The execution of the tool is split in two phases, grouped by dashed rectangles in Figure 2. In the **preliminary phase**, the MDSP netlist is simulated at the nominal voltage V_{nom} with Mentor Modelsim (① in the figure), in order to gather the set of *golden outputs* needed to measure quality during the optimization. Then, using Synopsys PrimeTime, the power and area of the MDSP at V_{nom} are annotated to be used as reference when comparing against RPR. Power analysis is performed using *switching activity* information previously collected in VCD files during the Modelsim simulation.

Another set of simulations is then run in order to determine the accuracy of different replica configurations (②). In doing this, we exploit two facts: (1) in a fixed-point datapath circuit, removing some LSBs from an input bus is functionally equivalent to setting them to logic 0; (2) in the final design, the replica will be timing-compliant, i.e. error-free.

Thanks to (1) we can obtain the outputs produced by a replica in which some LSBs ave been removed by forcing those bits to zero in the MDSP simulation. This does not require the availability of the actual replica netlist, and can be done directly on the MDSP. Thanks to (2) we can run *functional* simulations in this phase, without the need of accurate timing information. This allows one to execute a single set of replica simulations for all V_{VOS}, and also speeds them up.

By progressively zeroing out bits in the simulations in increasing bit-weight order, multiple sets of replica outputs with decreasing accuracies are obtained. Although precise power and area information are not yet available, it is safe to assume that at a given $V_{VOS,i}$, a larger number of removed bits corresponds to a smaller overhead.

When golden outputs and replica outputs for all configurations have been gathered, the decision threshold T_h' for each bus, and for each version of the replica is computed as:

$$T_h' = \max_{\text{all input stimuli}} |y_o[n] - y_r[n]| \leq T_h \qquad (8)$$

This expression is in accordance to Equation 3, where in this case y_o represents simulation golden outputs, and y_r represents replica outputs. Notice that the empirical threshold T_h' will be generally smaller than the theoretical one. Therefore, for some combination of inputs not present in the testbench, the decider may select the replica output even though the MDSP is correct. However, if the test stimuli are representative of real usage, this occurrence will be rare.

In the **optimization phase**, the tool iterates over the N standard-cell library characterizations in VOS (①). For each V_{VOS}, the MDSP is analyzed with PrimeTime and the timing

degradation due to the lowered V_{dd} is annotated in a Standard Delay Format (SDF) file. The SDF is then loaded in Modelsim, which runs a *timing-accurate* simulation of the MDSP in VOS conditions (②). This simulation may produce erroneous outputs due to the activation of violating paths; since a realistic input sequence is used, the error rate and magnitude are estimated accurately, also accounting for temporal correlation. With erroneous MDSP outputs available, the tool computes the minimum replica configuration that satisfies the quality constraints (③). To do so, it iterates over replicas in ascending order of removed bits. The outputs of a hypothetical RPR architecture are generated merging erroneous MDSP outputs and replica outputs taken from simulations, according to Equation 2 and to T_h'. The obtained sequence is processed by the quality evaluation module, where it is compared with the error-free version of the MDSP outputs. If the quality constraints are satisfied for all output buses, the evaluation is repeated with a smaller replica, until a configuration that violates at least one constraint is found.

When the minimum size replica configuration has been identified, the tool proceeds by generating its netlist (④). This task is performed with Synopsys Design Compiler (DC). In particular, we make use of the netlist simplification directives embedded in DC (e.g. `remove_net`). All input bits that were zeroed out during the replica simulation are removed from the netlist. Then, a re-synthesis is performed in order to propagate these simplifications, eliminating all cones of logic that have become useless. Re-synthesis also has a secondary purpose: DC will run all the necessary optimizations (e.g., gate resizing) in order to obtain a timing-compliant replica (at the chosen V_{VOS}), which is fundamental to correctly evaluate the power and area overheads of RPR. In fact, to meet timing, the replica may contain much larger gates with respect to the MDSP.

Once the replica is generated, the rest of the EC block must be synthesized. In particular, each output bus requires a proper *decider*. The decider has a fixed structure (subtractor, comparator and multiplexer), and the only variable parameters are the bitwidth and the threshold T_h'. Therefore, the tool uses a *generic* VHDL model to generate it.

The final step performed by the tool is the estimation of the total power and area costs of the RPR configuration in PrimeTime (⑤). Here, the tool also checks the actual timing compliance of the replica and decider, necessary for the correct behavior of the entire architecture. A configuration is considered feasible if the total power (including dynamic and leakage contributions) is less than the power of the MDSP at nominal voltage, and if the EC block meets timing. The loop is then repeated with the next VOS library characterization.

The tool also has a *full mode*, in which the complete tradeoff between accuracy and power savings is analyzed. In this mode, instead of just extracting the minimum replica size for each V_{VOS}, all configurations that meet the quality constraint are synthesized and analyzed. Clearly, with respect to the normal operation, this mode greatly increases the execution time. Finally, it is also possible to set a hard constraint on the maximum error rate allowed for the RPR architecture.

IV. Experimental Results

We validated the methodology proposed in Section III with experiments on a set of IPs publicly available on OpenCores [8]. We synthesized the original RTL models with Synopsys DC F-2011.09, targeting a commercial 45nm standard-cell library from STMicroelectronics. For all designs, we set the nominal (error-free) supply voltage to 1.1V, and we selected a synthesis frequency that produces a critical positive slack of less than 100ps in those conditions. We characterized the library in terms of timing and power in a set of VOS points going from 1.05V to 0.50V in steps of 0.05V. The flow depicted in Figure 2 has been integrated also with Synopsys PT F-2011.12 and Mentor Modelsim SE 6.4a.

We selected the following five designs for our experiments:

- Pipelined 16th-order lowpass FIR filter in direct form (12-bit in, 24-bit out) [9].
- Pipelined 4th-order lowpass IIR filter in direct form I, modeled after a butterworth analog filter (16-bit in, 32-bit out) [9].
- Decimation In Time Fast Fourier Transform (DIT-FFT) "butterfly" unit (16-bit in, 16-bit out) [9].
- Rotation Mode (RM) pipelined COordinate Rotation DIgital Computer (CORDIC) unit (16-bit in, 16-bit out) [10].
- Fixed-point 32-bit Square Root Unit (SRU) implementing the Goldschmidt recursive equation [11].

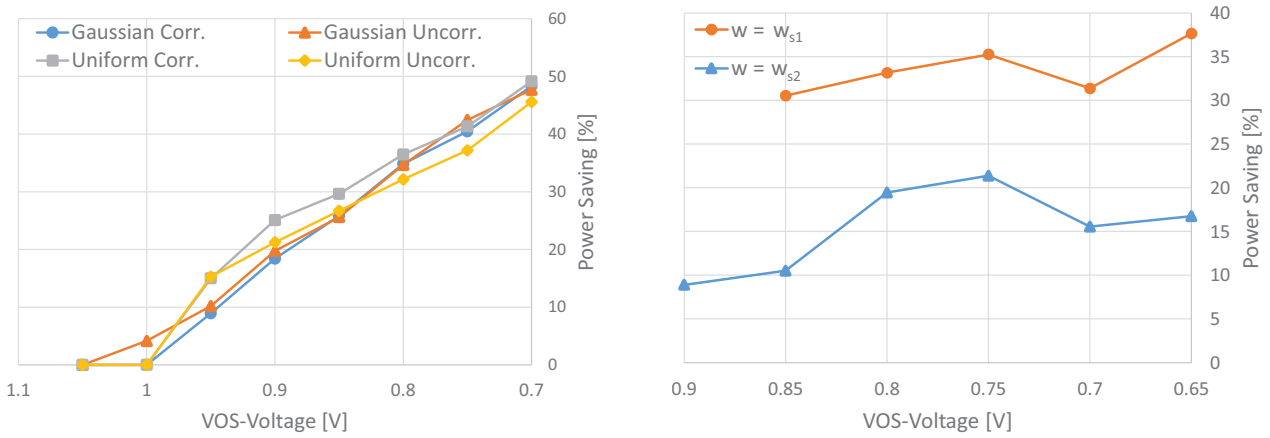
FIR and butterfly have been selected to compare our results with those of [6], while the others are used to show the extensibility of our algorithm, and to discuss the feasibility of RPR in relation to the characteristics of a given hardware.

For each circuit, we generated a testbench in VHDL that stimulates it with realistic inputs extracted from a suitable error-resilient application. Depending on the nature of inputs, we adopted two different quality metrics. For communication applications we measured the SNR as in Equation 4, while for arithmetic units we used the inverse of the Mean Squared Error (MSE):

$$Q_{mse} = \frac{n}{\sum_{i=0}^{n}(\overline{y}[n] - \overline{y_o}[n])^2} \tag{9}$$

where $\overline{y}$ and $\overline{y_o}$ are y and y_o normalized to the maximum value expressible on each output bus, in order to meaningfully compare errors on busses with different widths [1]. The discriminant for the usage of one quality metric or the other is the presence of channel noise. By combining the effects of timing-errors with a preexistent source of inaccuracy (noise), SNR is effective in showing how RPR does not affect the overall quality of processing in communcation systems, if the replica is sufficiently large. MSE, instead, is suitable to consider the impact of VOS-induced errors alone, for systems in which no information on input data noise is available.

[1]This metric is also sometimes referred to as Peak SNR (PSNR). We preferred a different notation to highlight that it does not account for external noise but only considers VOS-induced errors.



(a) Multiplier.

(b) FIR filter.

Fig. 3: RPR Input dependence, Power vs. V_{dd}.

For what concerns power savings and area overheads, we refer all of our measurements to the metrics of the MDSP in nominal conditions:

$$\begin{cases} P_{sav} = \left(1 - \frac{P_{rpr}(V_{VOS},B)}{P_{mdsp}(V_{nom})}\right) \cdot 100\% \\ A_{ovr} = \left(\frac{A_{rpr}(V_{VOS},B)}{A_{mdsp}(V_{nom})} - 1\right) \cdot 100\% \end{cases} \quad (10)$$

A. Analysis of Input Data Dependence

In order to show the importance of implementing RPR in an input-aware manner, we analyzed the impact of different input datasets on power savings for a pair of designs. As a first example, we applied RPR to a fixed-point signed array multiplier, with 16-bit inputs and 32-bit outputs. We used MSE as quality metric, and we set an arbitrary constraint $Q_{mse} \geq 60\ dB$ (i.e. $MSE \leq 10^{-6}$). We generated two sets of 5000 inputs: in the first, both operands of the multiplier are uniformly distributed in the range $[-2^{15} : 2^{15} - 1]$, while in the second, they follow a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 2^{10}$. Moreover, we repeated the two experiments after sorting the stimuli in ascending order, thus enforcing a strong temporal correlation among subsequent vectors. We set a maximum error rate of 0.2, which, for all inputs, led the tool to discard configurations with $V_{VOS} < 0.7V$.

Figure 3a shows the results produced by our tool on a power saving vs voltage plane. Power saving follows a similar monotonically ascending trend for all data sets. However, the maximum difference between the most and least favorable conditions is around 8% (at $V_{VOS} = 0.9V$), and the relation among input set and power saved is not trivial.

The reason for these results is twofold. On one hand, different inputs stimulate the internal paths of the multiplier differently. Because delay faults depend on *input patterns pairs*, both absolute values and temporal sequences are relevant. Since activation probability depends on input, also the percentage of violating paths at each V_{VOS} (and consequently the error rate) does. Thus, at a given voltage, the tool is often able to obtain the same quality using two *different replica bitwidths* for different stimuli. Clearly, a smaller replica will produce a smaller leakage overhead and a larger power saving. On the other hand, the sequence of inputs also directly affects the *switching activity* of the netlist gates, which in turn influences the dynamic power consumption of MDSP and replica.

As an example, at $V_{VOS} = 1V$ the correlated Gaussian inputs produce an error rate of $1.5 \cdot 10^{-3}$ while the error rate for uncorrelated data is only $0.4 \cdot 10^{-3}$. However, the total power consumption of the MDSP at this voltage is $0.6mW$ for the correlated case, and $1.7mW$ for the uncorrelated.

The impact of inputs on the effectiveness of RPR is even more evident for a more complex circuit, such as the 16th order FIR (Figure 3b). We fed the filter with two sets of full-swing sinusoid samples, respectively at $w_{s1} = 0.01\pi rad/s$ and $w_{s2} = 0.1\pi rad/s$, both affected by Additive White Gaussian Noise (AWGN) with a SNR of 25dB. The cutoff frequency of the filter is $w_c = w_{s2}$. In this case, the two sets of stimuli have the same distribution of values, only the rate of variation over time differs. We instructed the tool to keep the quality at $Q_{snr} \geq 25dB$ (i.e. the RPR configurations are accepted as long as the FIR output SNR is dominated by the channel noise power), and we set the maximum error rate to 0.2.

In Figure 3b, we only report results starting from $V_{VOS} = 0.9V$ because for higher voltages, even though the critical path slack is negative, no input pair generates a timing violation. When applying the input at w_{s1}, the circuit is still error-free also at $0.9V$.

We can notice how in this case power savings do not increase monotonically with decreasing voltage. This phenomenon is typical for complex designs, especially in deep VOS conditions. Since the replica must be timing compliant, the tool optimizes its netlist until a positive slack is obtained, possibly modifying gate sizes, and thus increasing the leakage overhead of the RPR system. This contribution may be larger than the decrease in dynamic power with respect to the previous VOS point. When voltage is reduced further, if the error rate

Benchmark	Net Count	f_{clk} [MHz]	Optimal V_{VOS} [V]	Replica/MDSP Bits	Tot. Power Saving [%]	Area Overhead [%]
FIR Filter	3379	330	0.60	6/12	44.96	82.39
FFT Butterfly	3840	200	0.55	13/16	49.66	133.20
IIR Filter	3556	250	0.50	9/16	8.82	181.82
RM-CORDIC	4939	400	0.55	11/16	42.05	127.64
SRU	21267	125	0.75	17/32	47.91	143.32

TABLE I: Summary of Results for the Five Benchmarks.

does not increase significantly, allowing to maintain a similar replica size, total savings can start to increase again.

The main difference with respect to the multiplier example, however, is the influence of the input/output transfer function on quality. In fact, the attenuation due to the lowpass FIR is different at w_{s1} and w_{s2} and this causes a reduction in signal power for the higher frequency samples. Consequently, the output SNR is significantly different, even under the same conditions of channel noise and rate of VOS-induced errors. This dependency adds up with the influence of vectors sequences on activated paths, resulting in a power saving difference between the two inputs sets larger than 20%.

These two examples clearly highlight that the assumption of uniformly distributed inputs can lead to suboptimal RPR architectures (either overly optimistic or pessimistic depending on circuit). They also show that the sole distribution of values is not a sufficient indicator, and that accurate simulations can provide much more realistic results.

B. Flexibility of the Approach and Comparison with Ad-Hoc Solutions

For each of the five target benchmarks we selected an error-resilient application that uses it, and profiled it to generate the simulation stimuli required by the tool.

For the FIR filter and FFT-butterfly, we considered the same applications examined in [6], i.e. the lowpass filtering at the receiver of a QPSK communication link, and the modulation/demodulation in Orthogonal Frequency Division Multiplexing (OFDM) WLAN respectively. We also assumed the same channel SNR conditions ($21.5dB$ for the FIR and $55dB$ for the butterfly). As above, we set the quality constraints of the tool to the same SNR values, in order to enforce the negligibility of VOS-induced errors. To generate the signal component, we started from a MATLAB model of the IEEE 802.11g WiFi standard. We extracted the input data for the two circuits, converted it to the appropriate binary format and added AWGN. The produced sets of vectors were loaded in the VHDL testbench passed as input to the automation tool.

RM-CORDIC can be used in place of the classical butterfly unit in FFT accelerators [12]. Similarly, the lowpass butterworth IIR filter can replace the FIR filter in a QPSK receiver. Therefore, we decided to test these two new circuits for the same applications of the previous two, and under the same quality constraints. This shows how our tool can be used to determine empirically the most suitable architectures for an approximate realization of a given task.

Finally, the Square Root Unit (SRU) is used in many error-resilient applications, from FFT magnitude evaluation to SVD-based image compression. For our experiments, we targeted the Automatic Gain Control (AGC) task present in many DSP systems [9]. We generated the inputs to the SRU from a set of baseband audio samples. In this case, in absence of precise data on noise, we used the MSE quality metric, and we decided to accept a maximum MSE of 10^{-12}.

Table I summarizes the obtained results for all five benchmarks. The power saving values confirm the generality of the tool, since almost 50% reduction in *total* power can be obtained for 4 of the 5 benchmarks, with the notable exception of the IIR filter, which is however due to intrinsic features of the RPR approach.

We profiled the computational burden of the RPR automation tool, in order to assess its applicability in real-life scenarios. Rather than absolute values, we measured the ratio between the execution time of the tool and that of a normal design and verification flow (i.e. a single sequence of synthesis, timing annotation, timing simulation and power estimation). This metric is in first approximation independent from the workstation used in the process, as well as from circuit complexity and input stimuli length; it only depends on the number of VOS voltage points and on the number of feasible solutions found by the tool. We discovered that the execution time of the RPR automation tool for the five benchmark circuits is between $6x$ and $9x$ that of a standard flow. Notice that this overhead could easily be reduced selecting a smaller number of VOS points. Moreover, the reference time for the standard flow does not consider any power optimization, which could require several iterations to converge to an optimal implementation.

The RPR EC block introduces a significant area overhead of more than 100%, except for the FIR benchmark. This is partly caused by the gate resizing performed by DC during the replica optimization; however, it is also a consequence of our application-agnostic approach. In fact, since we create the replica from the MDSP netlist, using Design Compiler directives, we only allow *logic-level* optimizations to be performed on the reduced unit. DC cannot perform higher-level optimizations such as instantiations of different DesignWare macro-blocks. As an example of this issue, under the same synthesis constraints, an 8x8-bit multiplier generated removing inputs from a 16x16 netlist is typically 1.3 times larger than one generated directly from behavioral VHDL. However, we did not want to limit the applicability of our tool only to designs defined parametrically, in which replicas could be

generated setting some *generics* appropriately. Therefore, we accepted this additional overhead.

The power savings for the FIR and butterfly are comparable to those obtained by Shim et al. in [6]. In particular, [6] achieves 60% power reduction for the filter (versus our 44.96%) and 44% (versus 49.6%) for the FFT component. Differences are partly related to different testing conditions (especially the technological node), partly to the different timing and power models adopted. For instance, the leakage power overhead is much more significant at 45nm than at 250nm. However, the main contributor to the discrepancies might be the fact that we take into account input sequences in our optimization, obtaining more realistic estimates of the real timing and power behavior. In summary, the comparison with an ad-hoc approach clearly shows the competitiveness of our methodology.

C. Impact of Architectural Features

The IIR benchmark deserves a separate analysis. We introduced this circuit in our experiments because it has a significant difference with respect to all others: the presence of a feedback network from the outputs [9]. The fact that previous outputs are used to produce new ones is clearly a drawback for the insertion of RPR. When a timing error is generated, it remains in the system for more than a single clock cycle, worsening the output quality. In other words, even if the rate of timing violations is low, the error rate becomes extremely large. Since the "memory" of the IIR is limited in time by the finite data precision, the error gradually reduces in magnitude and eventually becomes comparable to channel noise. This is the reason why our tool is still able to produce feasible solutions. Moreover, the presence of feedback also limits the simplifications that can be performed by DC when inputs are removed for replica generation. Thus, this benchmark also has the largest area overhead.

The IIR example highlights some of the characteristics that a circuit should possess to be an ideal candidate for RPR. However, results show that our tool is able to deal also with unfavorable input netlists, generating all feasible solutions. The assessment of the convenience of such solutions is clearly left to the designer.

D. Quality versus Power tradeoff

As a last result, we present in Figure 4 an example of the output produced by our tool when executed in *full mode*, for the FIR filter test case. To generate it, we set a very low quality constraint ($0\ dB$), and no error rate constraint. In practice, this forces the tool to produce the entire set of feasible configurations for each V_{OS} point in which the error rate is not 0. Input vectors and noise parameters are the same used in Section IV-B. Two trends can be identified in the graph: on the left side (correspondent to large replica implementations), the effects of channel noise dominate, and the quality saturates at the $21.5\ dB$ limit; as soon as the RPR effects start to become relevant due to the reduction in replica size, the tradeoff between power and quality becomes evident. Given a desired quality value, the optimal configuration is found as the rightmost point above it in the graph. Note that not all curves have the same number of points. This happens because the tool still eliminates those replica implementations that either produce a negative power saving, or cannot be synthesized with positive slack at the given V_{OS}.

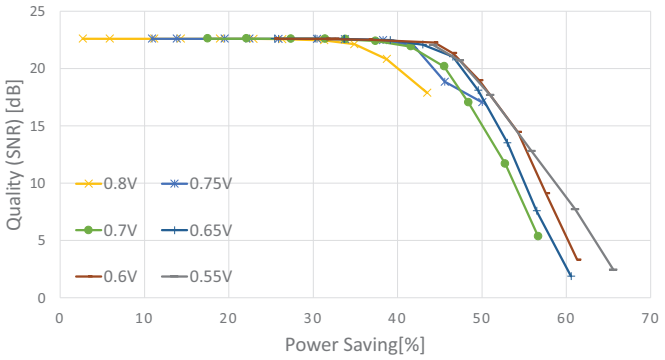


Fig. 4: Quality vs. Power tradeoff for the FIR Filter.

V. Conclusions

We have proposed a flexible framework for the implementation of RPR architectures starting from existing gate-level netlist of fixed-point arithmetic and DSP circuits. Our tool improves the analysis accuracy over previously proposed theoretical RPR formulations by using accurate circuit models and taking into account input dependence, whose importance was clearly shown in this work. Despite its generality, which allowed to apply RPR to circuits that had never been considered before, the tool is still able to reach comparable results with respect to ad-hoc approaches.

We have also confirmed that sequential loops represent a major obstacle against the adoption of RPR. A possible future work for the improvement of our tool is the revision of RPR in order to take this aspect into account.

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS'13*, pp. 1–6.

[2] V. Chippa et al. "Analysis and characterization of inherent application resilience for approximate computing," in *DAC-50*, 2013, pp. 1–9.

[3] K. Nepal et al. "Abacus: A technique for automated behavioral synthesis of approximate computing circuits," in *DATE'14*, pp. 361:1–361:6.

[4] G. Karakonstantis and K. Roy, "Voltage over-scaling: A cross-layer design perspective for energy efficient systems," in *ECCTD'11*, pp. 548–551.

[5] R. Hegde and N. Shanbhag, "Soft digital signal processing," *IEEE Trans. on VLSI*, vol. 9, no. 6, pp. 813–823, Dec 2001.

[6] B. Shim et al. "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. on VLSI*, vol. 12, no. 5, pp. 497–510, May 2004.

[7] B. Shim and N. Shanbhag, "Performance analysis of algorithmic noise-tolerance techniques," in *ISCAS '03*, pp. 113–116 vol.4.

[8] *opencores.org*

[9] A. V. Oppenheim et al. *Discrete-time Signal Processing (2Nd Ed.)*. Prentice-Hall, Inc., 1999.

[10] J. E. Volder, "The cordic trigonometric computing technique," *IRE Trans. on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, Sept 1959.

[11] I. Koren, *Computer Arithmetic Algorithms*, A. K. Peters, Ltd., 2001.

[12] P. Choudhary and A. Karmakar, "Cordic based implementation of fast fourier transform," in *ICCCT'11*, pp. 550–555.