Energy Efficient MIMO Processing: a Case Study of Opportunistic Run-Time Approximations

David Novo, Nazanin Farahpour and Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH–1015 Lausanne, Switzerland
{david.novobruna, nazanin.farahpour, paolo.ienne}@epfl.ch

Ubaid Ahmad and Francky Catthoor
Interuniversitair Micro-Electronica Centrum vzw (IMEC)
Kapeldreef, 75
BE–3001 Leuven, Belgium
{ubaid, catthoor}@imec.be

***Abstract*—Worst-case design is one of the keys to practical engineering: create solutions that can withstand the most adverse possible conditions. Yet, the ever-growing need for higher energy efficiency suggest a grim outlook for worst-case design in the future. In this paper, we propose opportunistic run-time approximations to enable a continuous adaptation of the processing precision (operator type and bitwidth) to the actual execution context without modifying the algorithm functionality. We show that by relaxing the processing precision whenever possible, a VLSI implementation of an advanced wireless receiver algorithm based on opportunistic run-time approximations can save about 40% of the energy consumed by an optimized static implementation. These energy savings are achieved at the expense of a slight increase in overall chip area.**

I. Introduction

The limitations of battery capacity, the slowing progress in CMOS scaling, and the ever-growing need for more computational power claim for new methodologies to design digital implementations of higher energy efficiency. Advanced digital signal processing algorithms are usually executed in static implementations despite typically operating under highly varying execution contexts (e.g., a wireless channel). In this paper, we introduce the concept of *opportunistic run-time approximations* in a fundamental departure from the notion of a static worst-case processing precision design point. Opportunistic run-time approximations move in a new practical direction by (1) generating a multiplicity of context-dependent relaxed specifications, (2) creating optimized implementations—by adapting the operation type, number and precision of each specification—, and (3) continuously monitoring the execution conditions. Thereby, the cheapest implementation that applies to the actual execution context can opportunistically be selected without compromising the quality experienced by the user. Opportunistic run-time approximations do not modify the algorithmic functionality but only its processing precision.

We use the VLSI implementation of an advanced *Mulitple Input, Multiple Output (MIMO)* wireless receiver to illustrate the application of opportunistic run-time approximations on a practical example. Our results show that the new approach is able to save about 40% of the energy consumed by a manually optimized static implementation designed to work under all conditions. Such an improvement in energy efficiency comes at the expense of a small overall area increase.

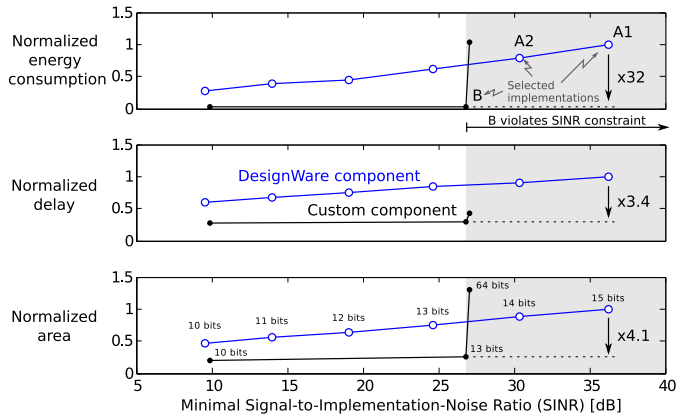
978-3-9815370-2-4/DATE14/©2014 EDAA

II. Motivational Example

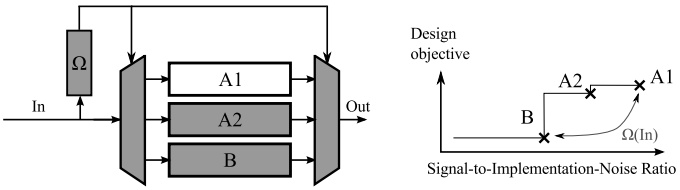
To illustrate the concept of opportunistic run-time approximations, let us consider the implementation of the reciprocal square-root operator, $b = 1/\sqrt{a}$, where a corresponds to the input signal and b to the output. Fig. 1(a) shows the energy consumption, delay and area of two different implementation approaches: the *DesignWare* component provided by Synopsys [13], and a *Custom* component that consists of a piece-wise linear approximation of the operator. The latter includes an initial scaling of the input range to $[\frac{1}{4}, 1)$, a linear approximation of the operation for that particular input range, and a final rescaling of the output. Fig. 1(a) also includes the minimal *Signal-to-Implementation-Noise Ratio (SINR)*, which represents the level of approximation of the operator compared to its infinite precision version—i.e., $SINR = 20 \cdot log_{10} \frac{b_{IP}}{|b_{IP} - b_{FP}|}$, where b_{IP} and b_{FP} correspond to the infinite (a very high precision in practice) and finite precision version of the output b.

The *DesignWare* component consistently increases its energy consumption, delay and area when the input bitwidth grows from 10bit to 15bit. This tendency is similar to what can be observed in other elementary operators, such as adders or multipliers. Instead, the *Custom* component reaches a saturation point (at 13bit) from which wider bitwidths do not translate into a more accurate implementation. This condition corresponds to the maximum precision that can be achieved with such a linear approximation. If the target application requires higher precision (e.g., $SINR \geq 35dB$), design B of *Custom* (see top graph of Fig. 1(a)) cannot be used because it violates the specification; therefore, design $A1$ of *DesignWare* must be selected for implementation. Unfortunately, such a high precision comes at a cost: design A is $3.4\times$ slower and consumes $32\times$ more energy than design B.

However, the high precision requirement (worst case) of the application does not need to be met at all times in most complex real systems. Actually, such a high precision is required only in a small percentage of the possible deployment situations. Accordingly, we propose an adaptive approach that can track the variations in the required level of approximation and accordingly select the cheapest implementation that satisfies the actual precision requirement. Figure 1(b) sketches the proposed approach, where a monitor Ω processes the input and selects the cheapest implementation that applies to the given context. The possible implementations, $A1$, $A2$ and B, correspond to Pareto points in the space defined by design



(a) Reciprocal square root operator implemented with two approaches for different bitwidth configurations. Configurations A1, A2 and B are Pareto in the energy consumption-SINR design space.



(b) Opportunistic processing using the selected Pareto configurations.

Fig. 1. **Motivational example.** Opportunistic run-time approximations to reduce the processing precision whenever possible.

objective(s) and the level of approximation (i.e., SINR). This approach has the potential to reach efficiencies close to B while offering the user experience of implementation $A1$. Importantly, families of components and of design techniques that would not make sense in a worst-case context where the high-precision result alone is needed suddenly become of interest due to their extreme economy (i.e., design B).

III. Approximating Implementations

Before discussing opportunistic run-time approximations, this section describes the different types of approximations that can be considered while refining a real *Digital Signal Processing (DSP)* implementation. First, *algorithmic approximations* can be used to transform the original algorithm to a different one of lower precision. For example, an estimation algorithm can replace the ideal *2-norm* distance by a less complex *1-norm* distance at the expense of some loss in convergence optimality. Then, *Algebraic approximations* are a second family of transformations that can be applied once the algorithmic approximations have fixed the algorithm. These approximations replace the ideal operators with approximated ones. For example, consider the *Custom* implementation of the reciprocal square root used as a motivational example in Section II. Finally, *signal approximations* can be applied once the algebraic approximations have defined the operators. These approximations assign a finite number of bits to the inputs and outputs of each operator in the algorithm.

Accordingly, we define Implementation Noise (IN) as the difference on the output of the final approximated implementation with respect to the original ideal precision specification, the power of which is used to compute the SINR. The final IN is the aggregate of the three types of approximations

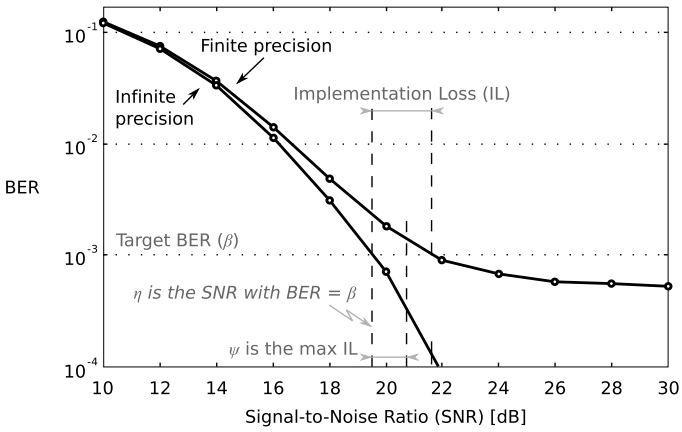


Fig. 2. **Finite precision wireless system.** The *Implementation Loss (IL)* is the difference in SNR between a finite precision implementation and its infinite precision spec at the target BER, e.g., $\beta = 10^{-3}$. Here the finite precision implementation is not within constraints as IL is larger than the maximum degradation specified by the designer, ψ.

described earlier. Ideally, the opportunistic run-time approximations entail a separate IN budgeting for each of the Pareto configurations, e.g., for $A1$, $A2$ and B in the example of Fig. 1(b). However, algorithmic approximations are already discussed by Min Li et al. [7], and this paper focus in the complementary algebraic and signal approximations. Still, the we do make use of the algorithmic transformations to design the monitor discussed in Section V and the general framework to handle approximations proposed in this paper also covers algorithmic transformations.

A. Approximated Wireless Systems

The tolerable amount of IN is domain dependent. In wireless communication systems, the application performance is typically characterized by a *Bit Error Rate (BER)* curve. As shown in Fig. 2, the BER decreases monotonically with the Signal-to-Noise Ratio (SNR): the cleaner the channel, the more reliable the wireless link. After applying the approximated transformations described earlier, wireless systems experience a shift in the BER curve. The curve of a finite precision system will always need a higher SNR than its infinite precision version to reach the same BER. Such an increase in SNR is typically known as Implementation Loss (IL). Thus, the finite precision refinement of a wireless system is often defined as:

$$\arg\min C(\lambda) \text{ s.t. } \mathrm{IL} \leq \psi,\ \mathrm{BER} = \beta. \tag{1}$$

where C corresponds to the cost metric objective of the optimization (e.g., energy, area, etc.), λ is the vector of signal configurations (i.e., a signal configuration defines the operator approximation as well as the operator bitwidths), β is the maximum BER tolerated by the data transmission link[1], and ψ is the maximum IL. However, in practice BER can only be estimated after running a Monte Carlo simulation that takes a particular SNR as input. Accordingly, we propose remapping the original constraints to an equivalent set as follows:

$$\mathrm{IL} \leq \psi,\ \mathrm{BER} = \beta \ \Rightarrow\ \mathrm{BER} \leq \beta,\ \mathrm{SNR} = \eta + \mathrm{IL}, \tag{2}$$

[1]BER that guarantees the maximum packet error rate stated in the standard.

```
input:  λ[1..numSig], β;
output: λ[1..numSig];
Ber  = simBer(λ);  // Monte Carlo simulation
Cost = getCost(λ); // cost model function
while Ber ≤ β // degradation constraint
  for sig = 1 to numSig // try all signals
    λ(sig)       = λ(sig) − 1;  // prec. reduction
    thisDeltaBer  = simBer(λ) − Ber;
    thisDeltaCost = Cost − getCost(λ);
    thisMet(sig)  = Cost/thisDeltaCost +
        thisDeltaBer/Ber;
    λ(sig)       = λ(sig) + 1;
  end for
  // find signal minimizing metric Met
  thisSig    = find(thisMet == min(thisMet));
  λ(thisSig) = λ(thisSig) − 1;  // reduction step
  Ber        = thisBer(thisSig);
end while
λ(thisSig) = λ(thisSig) + 1;  // recover conf
```

Fig. 3. **Pseudocode of the finite precision optimization heuristic.** The sequential search [6] uses a complexity-distortion measure and iteratively chooses a signal to be reduced until finding the last configuration that satisfies the precision constraint.

where η corresponds to the SNR at which the infinite precision version delivers BER $= \beta$. Thereby, the optimization constraints can be checked with a single Monte Carlo simulation that only needs to evaluate whether BER $\leq \beta$.

B. Finite Precision Optimization

The optimization problem of Eq. 1 has been proven to be non-convex [3], and thus, a heuristic approach is required. In this paper, we use a slight adaptation of the *sequential search* heuristic [6], described in Fig. 3. This heuristic greedily reduces an initial overprecise configuration λ by decreasing at each iteration the precision of the signal that minimizes a complexity-distortion metric. Such metric combines the two effects of reducing the signal precision: the complexity reduction (e.g., area reduction) and the increase in distortion (e.g., increase in BER). The objective is to prioritize the precision reduction of the signals that have a high impact on the optimization metric and a low impact on the application performance metric. Ultimately, the heuristic selects the configuration that includes the most precision reduction iterations while still satisfying the BER degradation constraint. The precision reduction of a signal implies that the operator producing that signal changes. For instance, let us consider the motivational example illustrated in Fig. 1, where the reciprocal square root operator can take any of the three Pareto configurations, i.e., $[A1, A2, B]$. The optimization will attempt to replace configuration $A1$ with a cheaper implementation, pushing hard to include B in order reduce the complexity significantly.

The estimation of the complexity reduction caused by a particular precision reduction step is based on a model. First, we create a database with the estimated area of each of the operators for the different bitwidths. Then, every time that the complexity needs to be estimated, the operators composing the particular configuration λ are selected from the database and their areas are added up together. The purpose of the model is not to predict accurately the absolute complexity of the circuit but to guide relative decisions in the optimization process.

IV. The driver: a MIMO receiver

The *Hybrid Lattice Reduction (HLR)* algorithm, introduced by Ahmad et al. [1], is an effective way of improving the performance of modern wireless communication systems. Accordingly, this type of algorithm is expected to be implemented in future handhelds and be responsible for an important share of their energy consumption [2].

An $N_T \times N_R$ *Multiple Inputs, Multiple Outputs (MIMO)* communication system includes N_T antennas at the transmitter and N_R antennas at the receiver. Assuming ideal conditions, this system can multiply the throughput by the number of transmit antennas. However, in practice, the simultaneous transmissions interfere with each other and reduce the achievable throughput. The HLR algorithm can reduce these interferences by transforming the $N_T \times N_R$ channel matrix, H, experienced by each of the frequency subcarriers. Our HLR is instantiated as a 4×4 MIMO design for a 3GPP *Long Term Evolution (LTE)* system. Accordingly, every symbol includes 512 subcarriers, each with a 4×4 channel matrix that is generated based on the standardized channel model.

HLR is a good example of the type of algorithms that are expected to run in future energy limited platforms. HLR includes non-standard operations such as reciprocal or reciprocal square root and it exhibits a heavily input-dependent execution flow. It includes a variable iteration loop and three iterations over two conditionally executed *Basic Blocks (BB)*. Thus, depending on the input, the HLR executes 0 to 36 BBs. Harsh channels will trigger more computations, while good channels will require few or even no computations at all. Thus, if the type of channel can be identified before running the HLR algorithm, a more energy-efficient implementation can be optimized to process only the good channels. The next section discusses whether such a monitor is feasible.

V. Monitoring the deployment context

The monitor required by our opportunistic run-time approximations should be able to derive the execution context out of the analysis of some input data. The design of such a monitor involves detailed knowledge of the application as the monitor often needs to exploit complex correlations that are different for each application. Moreover, these correlations may not even be explicit in the application code. Thus, we must rely on our expertise of wireless communications to propose an effective monitor for the HLR algorithm. The question to ask is the following: *What external conditions can stress our algorithm?*

In the case of wireless applications, the channel is a natural external condition. A perfect MIMO channel matrix would be orthogonal, meaning that no mutual influence exists between the transmitted streams. For an orthogonal channel matrix, the simplest *Zero Forcing (ZF)* detector performs equal to the optimal and prohibitively computationally expensive *Maximum Likelihood (ML)* detector [15]. Thus, a metric able to measure the distance between the current channel matrix and an orthogonal one should be a good candidate to predict the HLR computational effort. Such a metric is called *Orthogonality Defect (OD)* [14] or δ, and is computed as follows:

$$\delta = \frac{\|r_1\|^2 \cdot \|r_2\|^2 \ldots \|r_M\|^2}{\det R^H R}, \tag{3}$$

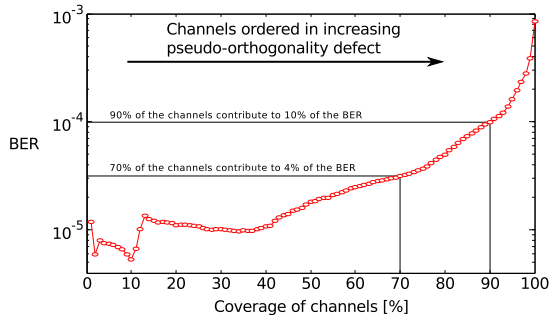


Fig. 4. **Correlation between the HLR monitor and the BER.** The pseudo-orthogonality defect can be used to identify the channels causing high BER.

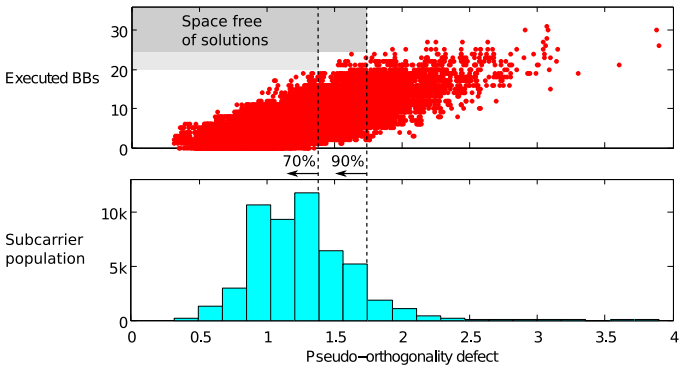


Fig. 5. **Correlation between the HLR monitor and the algorithmic complexity.** Subcarriers with a lower pseudo-orthogonality defect execute less BBs. Moreover, most of the subcarriers exhibit rather low POD, providing evidence that a worst-case implementation overprovisions in most of the deployment situations.

where R is one of the matrix inputs of the HLR algorithm and includes M vector columns r_i. R_H is the hermitian of R, det is the matrix determinant and $\|\cdot\|$ is the norm operator. The OD involves heavy computations; so we propose to reduce its complexity through the approximate algorithmic transformations described in Section III and derive a *Pseudo-Orthogonality Defect (POD)* or ρ, that captures the same principles but simplifies the implementation complexity:

$$\delta \sim \rho = \frac{|r_{11}| \cdot |r_{22}| \cdots |r_{44}|}{\prod_{j=1}^{M}(\sum_{i=1}^{j}(|\mathrm{Re}(r_{ij})| + |\mathrm{Im}(r_{ij})|))} = \frac{\rho_N}{\rho_D}, \quad (4)$$

where $|\cdot|$ is the absolute value operator and, Re and Im select the real and imaginary part of the operand, respectively.

To check the effectiveness of our approximated metric, we run an experiment and rank all the subcarriers of 51,200 channels according to POD. Fig. 4 shows that the proposed metric can successfully detect the good channels, which cause a small BER. E.g., when ordering all the channels according to the metric, the first 70% of the channels contribute to only 4% of the total BER.

Once we know that the POD is a good metric to differentiate good from harsh channels, we analyze whether the metric is also useful to predict the number of HLR computations. Fig. 5 demonstrates that the channels with lower POD execute fewer number of BBs. E.g., 70% of the channels will never execute more than 20 BBs. The execution of fewer operations also means that the implementation noise budget can be distributed over fewer operations, which enables a further approximation of the signals and operators. Accordingly, our monitor will include the computation of the numerator ρ_N and denominator ρ_D of Equation 4 and will trigger the execution of a relaxed implementation whenever $\rho_N < \rho_D \cdot \mathcal{T}$, where $\mathcal{T}$ corresponds to the threshold detailed in the next section.

VI. Exploiting Relaxation

After showing in the previous section that identifying relaxed execution contexts for the HLR is possible, this section proposes a method to safely design the relaxed implementations that will deliver ultimate energy efficiency. A critical design parameter is the definition of the threshold value $\mathcal{T}$. In this work, we sweep this parameter to study the effect on the global system. Moreover, we consider only two scenarios: a worst-case precision scenario, which is valid in any context, and a relaxed scenario, which is valid only for good channels. Accordingly, we select three different thresholds that basically determine that 90%, 70% or 50% of channels—with the smallest POD—are used to design the relaxed implementation. Each of these percentages reflects the coverage of the relaxed implementation and is represented with the variable x. B_x represents a BER simulation where only the x% best channels are used. Also, the finite precision configuration design resulting from the finite precision optimization of coverage x corresponds to λ_x.

Because any further loss of precision in the system will violate the original BER constraint, we need a new constraint in order to accommodate our relaxed implementation. This does not mean that the opportunistic run-time approximations incur a higher BER degradation than a worst-case implementation, but that the former needs to be built in a two-step approach. Thus, we define:

$$\beta' = \beta(1 + d/100), \quad (5)$$

where d is the BER percentage increase. Accordingly, if $B_{100}(\lambda_{100}) \leq \beta$, then the new constraint to be satisfied while optimizing the relaxed implementation is the following:

$$B_{100}(\lambda_{100}) + \frac{x}{100}(B_x(\lambda') - B_x(\lambda_{100})) \leq \beta', \quad (6)$$

where λ' is iteratively reduced in the optimization heuristic until violating the new constraint. The latter expresses that the bit errors produced by the relaxed channels when selecting configuration λ are now replaced by the errors produced by the alternative relaxed configuration λ'. Fig. 6 shows the result of optimizing the relaxed implementation for $x = [100, 90, 70, 50]$ and for $d = 5$, which increases the original BER constraint by 5%. Note that the starting point of all the optimizations is the same finite precision configuration, however, the BER largely varies depending on the channel coverage x. This shows that the sensitivity to the precision reduction steps also varies and the better channels can accommodate more precision reduction iterations before violating the new optimization constraint.

VII. Results

In this section, different VLSI implementations of the HLR algorithm are compared. First, we detail our experimental setup, then present the area and energy consumption results.

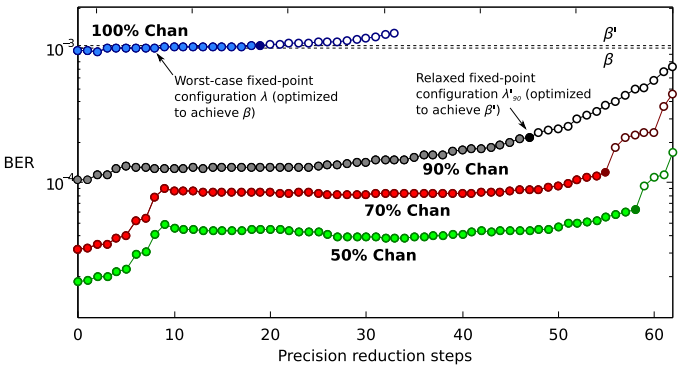


Fig. 6. **Finite precision refinement of the relaxed implementation.** The relaxed implementation can increase up to 5% the initial BER constraint β. All the fixed-point configurations satisfying such a constraint are marked with a full circle. For the selected configuration, the full circle is darkened.

TABLE I. POST-LAYOUT RESULTS.

Name	Cov. [%]	Max BER	Area [μm^2]	DynPow [mW]		Leakage [mW]
WC	100	1.00E-3 (β)	415K	97.47		6.03
RefWC	100	1.05E-3 (β')	389K (94%)	93.09	(96%)	5.41
Rel90	90	1.05E-3 (β')	312K (75%)	52.82	(54%)	3.78
Rel70	70	1.05E-3 (β')	239K (58%)	29.50	(30%)	2.80
Rel50	50	1.05E-3 (β')	225K (54%)	28.09	(29%)	2.51
Monitor	-	-	5K (01%)	1.73	(02%)	0.17

A. Area and Power Estimation Flow

The RTL-level description of the different designs was developed in Verilog and synthesized for a 65-nm STMicroelectronics standard cell technology with *Synopsys Design Compiler*. The synthesized netlist was then placed and routed with *Cadence SoC Encounter*. The area and delay were estimated from the resulting layout. Then the post-layout netlist (gate-level netlist annotated with extracted parasitics) was simulated with *Mentor Graphics Modelsim* to record in *Value Change Dump (VCD)* format the switching activity of the design while executing relevant inputs extracted from our Matlab model. Finally, the post-layout netlist and the switching activity traces were used to estimate the power consumption with *Synopsys PrimeTime*.

B. Opportunistic Run-Time Approximations on HLR

First, the *static worst-case (WC)* design is synthesized targeting a delay of zero. Obviously, such synthesis will fail at satisfying the timing constraint. However, the delay achieved by the tool before giving up is a good indication of the minimum circuit delay. Such minimum delay is then relaxed 10% in a second synthesis run, which yields a rather fast circuit but with moderate area and power consumption. Table I indicates the area and power consumption for the reference WC circuit, which was optimized to achieve a BER of β.

As discussed in Section VI, we do enable extra degradation to accommodate our relaxed implementations. Accordingly, the processing precision of the WC implementation is further relaxed to achieve the new BER degradation β', resulting in the configuration λ'_{100}, which is synthesized for the same delay as the WC design leading to the *RefWC* implementation. This represents a fair static reference for our opportunistic run-time approximations.

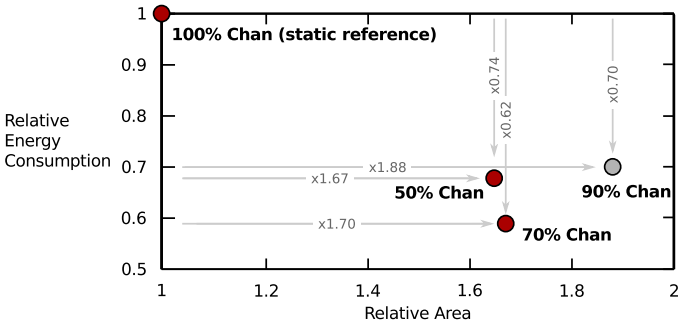


Fig. 7. **Area-Energy consumption Pareto space.** The implementations using opportunistic run-time approximations achieve a higher energy efficiency, however, not all the design points are Pareto optimal.

Intuitively, one should do better if this extra degradation budget is used to include a second implementation optimized only for a "gentle" subset of the channels. Three different instantiations of our opportunistic run-time approximations are explored: *Rel90*, *Rel70*, and *Rel50* targeting a 90%, 70% and 50% channel coverage, respectively. The resulting circuits, also synthesized for the same delay, consume significantly lower power than the *RefWC* implementation. Finally, the monitor described in Section V together with the mux and decoder shown in Fig. 1(b) are also implemented and the results show that their contribution to the overall area and energy is minimal.

C. Analysis and Discussion

To compare the different implementation alternatives we use the area and energy consumption estimations of Table I. The overall energy consumption, E of an run-time approximated implementation corresponds to the following:

$$E = x \cdot E_x + (1 - \frac{x}{100}) \cdot E_{WC} + E_M + L_x + L_{WC} + L_M, \quad (7)$$

where E_x, E_{WC} and E_M are the energy consumption in the dynamic power of the HLR implementation designed to cover the best $x\%$ channels, the *WC* implementation and the *Monitor*, respectively. The other components correspond to the energy consumed in leakage. The area, A, corresponds to the aggregate of the areas of the mentioned components. Accordingly, Fig. 7 plots in a two-dimensional Pareto space the area and energy consumption of the different implementations using opportunistic run-time approximations relative to the *RefWC* implementation. The results show that our adaptive implementations achieve higher energy efficiency than the one optimized to be valid in all channel conditions. However, the design targeting a relaxed mode coverage of 70% of the channels achieves the highest energy efficiency. This is because $Rel90$ has a more consuming relaxed implementation and $Rel50$ can trigger less often the relaxed implementation.

The implementations using opportunistic run-time approximations are significantly bigger than the reference implementation. However, one should consider that this number includes only datapath area. The latter is usually a big contributor to the overall energy consumption but only represents a small portion of the overall area of a wireless baseband implementation, which is usually dominated by memories. For example, the area or our reference WC design represents only a 2.5% of the overall area of an entire baseband chip [4].

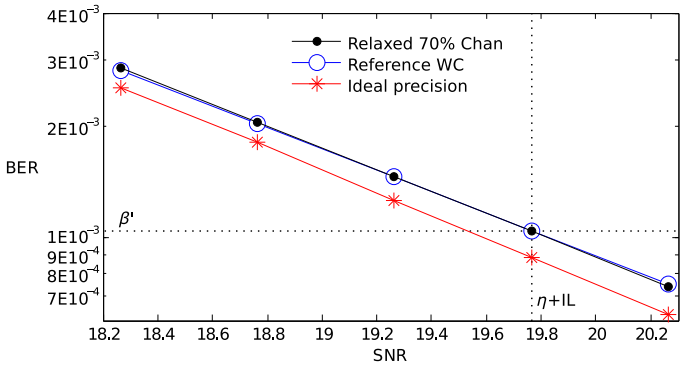


Fig. 8. **Identical performance.** BER simulation over 400 channels different from the training set. The simulation shows no meaningful performance difference between our adaptive approach and the worst-case design.

Finally, we select the most energy efficient implementation of our opportunistic run-time approximations and run a BER simulation with an extended set of 400 new symbol channels not used during the design phase, which included 100 symbol channels. Fig. 8 shows that the opportunistic run-time approximations deliver a BER which is indistinguishable from a traditional system, which means that from an application layer perspective both implementations are equivalent. However the one using opportunistic run-time approximations, uses the relaxed implementation 71.4% of the time, which leads to almost 40% savings in energy consumption.

VIII. Related work

Opportunistic run-time approximations are a form of system-scenarios [5]. Such scenarios are used to reduce the system cost by exploiting information about what can happen at run-time to make better design decisions at design-time, and to exploit the time-varying behavior at run-time. However, opposite of this work, typical implementations of system-scenarios are bounded to maintain the bit-true I/O correctness. Other work extends classic system-scenarios proposing communication mode dependent fixed-point refinement for communication systems, which could then be exploited in both hardware [8] and software [10] implementations to achieve higher processing efficiency. However, communication modes are just a subset of the scenarios that can be exploited with opportunistic run-time approximations. Alternatively, Novo et al. [11] propose to select different software mappings depending on the communication mode and on the amplitude of the channel coefficient. However, they do not evaluate the effect in energy efficiency and, more importantly, their reference design includes overheads not present in static designs. Instead, we quantize the energy savings with respect to an optimized static VLSI implementation.

Other existing types of approximated design, such as probabilistic switching [12] or significance driven computation [9], can occasionally result in huge degradations, which can be problematic in some application domains. Instead, our approach provides hard guarantees on the maximum degradation by designing the monitor to ensure that the relaxed implementations are only triggered in harmless situations. Moreover, our technique works at the algorithmic level and can be built on top of any existing implementation flow, not requiring special logic or any tweaking of the standard design tools.

IX. Conclusions

This paper presents a case study of opportunistic run-time approximations. We propose an effective monitor that can successfully switch execution context at negligible cost. We also show that the worst-case conditions are rare in practice. Thus, by relaxing the processing precision whenever possible, we can save about 40% of the energy consumed by an optimized static implementation. These energy savings are achieved at the expense of a slight increase in overall chip area. Finally, we demonstrate that our opportunistic run-time approximations can meet the same functional specification—BER in our case—as the reference static implementation.

References

[1] U. Ahmad, M. Li, S. Pollin, A. Amin, L. Van der Perre, and R. Lauwereins, "Hybrid lattice reduction algorithm and its implementation on an SDR baseband processor for LTE," in *Proceedings of the 19th European Signal Processing Conference*, Barcelona, Spain, Sep. 2011, pp. 91–95.

[2] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes, and W. Xu, "On 3G LTE terminal implementation-standard, algorithms, complexities and challenges," in *Proceedings of the International Wireless Communications and Mobile Computing Conference*, 2008, pp. 970–975.

[3] G. Constantinides, P. Cheung, and W. Luk, *Synthesis and optimization of DSP algorithms*. Springer, 2004.

[4] V. Derudder, B. Bougard, A. Couvreur, A. Dewilde, S. Dupont, L. Folens, L. Hollevoet, F. Naessens, D. Novo, P. Raghavan *et al.*, "A 200Mbps+ 2.14 nJ/b digital baseband multi processor system-on-chip for SDRs," in *Symposium on VLSI Circuits*. IEEE, 2009, pp. 292–293.

[5] S. V. Gheorghita, M. Palkovic, J. Hamers, A. Vandecappelle, S. Mamagkakis, T. Basten, L. Eeckhout, H. Corporaal, F. Catthoor, F. Vandeputte *et al.*, "System-scenario-based design of dynamic embedded systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, p. 3, 2009.

[6] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 76–76, 2006.

[7] M. Li, D. Novo, B. Bougard, C. Desset, A. Dejonghe, L. Van Der Perre, and F. Catthoor, "Energy aware signal processing for software defined radio baseband implementation," *Journal of Signal Processing Systems*, vol. 63, no. 1, pp. 13–25, 2011.

[8] W. Ling and Y. Savaria, "Variable-precision multiplier for equalizer with adaptive modulation," in *Proceedings of the 47th Midwest Symposium on Circuits and Systems*, vol. 1, 2004, pp. I–553.

[9] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. ACM, 2009, pp. 195–200.

[10] D. Novo, B. Bougard, A. Lambrechts, L. Van der Perre, and F. Catthoor, "Scenario-based fixed-point data format refinement to enable energy-scalable software defined radios," in *Proceedings of the Design, Automation and Test in Europe*, 2008, pp. 722–727.

[11] D. Novo, M. Li, R. Fasthuber, P. Raghavan, and F. Catthoor, "Exploiting finite precision information to guide data-flow mapping," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 248–253.

[12] K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1123–1137, 2005.

[13] *DesignWare Library–Datapath and Building Block IP*, Synopsys, Inc., 2013, `www.synopsys.com/dw/buildingblock.php`.

[14] M. Taherzadeh, A. Mobasher, and A. K. Khandani, "Communication over MIMO broadcast channels using lattice-basis reduction," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4567–4582, 2007.

[15] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction," in *IEEE International Conference on Communications*, vol. 2, 2004, pp. 798–802.