2PSA: An Optimized and Flexible Power-Precision Scalable Adder

Roger Porto
ViTech/PPGC/UFPel, IFSul
Pelotas, Brazil
recporto@inf.ufpel.edu.br

Bruno Zatt
ViTech/PPGC/UFPel
Pelotas, Brazil
zatt@inf.ufpel.edu.br

Nuno Roma
INESC-ID, IST/ULisboa
Lisbon, Portugal
nuno.roma@inesc-id.pt

Luciano Agostini
ViTech/PPGC/UFPel
Pelotas, Brazil
agostini@inf.ufpel.edu.br

Marcelo Porto
ViTech/PPGC/UFPel
Pelotas, Brazil
porto@inf.ufpel.edu.br

***Abstract*— Adders are the core of all arithmetic circuits and the proposition of efficient adders, in distinct perspectives, are a constant in the last decades, with a myriad of solutions focusing on a wide variety of applications. The emergence of approximate computing encouraged the development of a new generation of dedicated imprecise adders intending to reduce delay, area, power and/or energy, but none of the proposed solutions is able to support run time definition of distinct power-precision operation points. This article presents the Power-Precision Scalable Adder (2PSA) which is a dynamically configurable power-precision imprecise adder, where the number of power-precision operation points can be configured at design time and each supported power-precision operation point can be changed in run time. The obtained experimental results showed that 2PSA is a fully flexible and efficient imprecise adder, supporting a high variety of power-SNR pairs, as well as a wide range of applications. Considering 8-bit adders, the power-SNR pairs vary from 2%-55dB to 60%-13.75dB, where eight operation points are allowed. Considering 64-bit adders, the power-SNR pairs ranges from 2%-325dB to 60%-16.65dB and 64 operation points are allowed. 2PSA also reached expressive power (from 18% to 73%) and area (from 54% to 73%) savings when compared with non-optimized solutions supporting the same operation points.**

Keywords—arithmetic operators, approximate computing, imprecise adders, low-power design, power efficiency.

I. INTRODUCTION

The unclear future of CMOS scaling associated with ever-increasing application requirements has been pushing computer architects to find alternatives to enable high performance and energy/power-efficient computation. In the current scenario, where inherently imprecision-tolerant applications - such as multimedia processing, neural networks, machine learning, etc. - became omnipresent, approximate computing [1][2] arises as an outstanding alternative where accuracy/precision represents an additional system optimization knob. In other words, approximate computing adds a novel dimension to the design space exploration aiming: (i) performance increase; (ii) energy/power efficiency; and (iii) power-precision scalability and controllability.

On the one hand, general-purpose architectures may benefit from approximate computing in certain classes of applications but must guarantee precise computation in case of hard-computing applications. On the other hand, ASIC implementations may use application-specific knowledge to define the optimal power-precision tradeoff at design time but still require support for distinct levels of approximation if adaptation to the dynamically changing system status (workload, battery level, etc.) and application/user settings is desired. In both cases, there is a dire need for architectural support for approximate computing with dynamic power-precision scalability features.

Efficient design of adder operators has always been of high interest for academia and industry due to its widespread use in all classes of applications. With the advent of approximate computing, a series of approximate adder operators has been proposed such as CCB (Carry Cut-Back Adders) [3], ETA (Error-Tolerant Adder) [4][5], ACA (Almost Correct Adders) [6], and LOA (Lower-Part-OR Adder) [7]. In general, they are based on splitting the adder into sub-adders to reduce the carry propagation chain or on removing/simplifying the carry generation circuitry of a subset of bits. These solutions require the accuracy/precision level to be statically defined at design time, not allowing any power-precision scalability. In turn, adders such as the GeAr (Generic Accuracy Configurable Adder) [8] enable a flexible selection among a wide set of power-precision operation points. However, the GeAr does not allow runtime reconfiguration of the power-precision operation point. Therefore, there is plenty of room for the development of approximate adders, capable of providing support to power-precision scalability through dynamic reconfiguration with reduced area/power/delay overhead.

In this work, we propose the Power-Precision Scalable Adder (2PSA), featuring a design time definition of *n* power-precision operation points and a run time operation points selection. The 2PSA showed to be a flexible and efficient

978-1-7281-9625-1/20/$31.00 ©2020 IEEE

adder, which supports a diversity of power-precision operation points, allowing a dynamically tune of accuracy and power dissipation, according to the application and/or user requirements. Considering 8-bit adders, the power-SNR pairs vary from 2%-55dB to 60%-13.75dB, where eight operation points are allowed. Considering 64-bit adders, the power-SNR pairs ranges from 2%-325dB to 60%-16.65dB and 64 operation points are allowed. 2PSA also reached expressive power (from 18% to 73%) and area (from 54% to 73%) savings when compared with non-optimized solutions supporting the same operation points.

II. Related Works

Approximate computing using imprecise adders is an efficient solution to reduce delay, area, power and/or energy in arithmetic circuits at a cost of a limited level of imprecision. Many imprecise adders were proposed intending to explore the trade-off between the circuit features (power, energy, area, and delay) and the precision of the reached results.

The Accuracy-Configurable Adder (ACA) [6] segments the addition in three overlapped sub-adders, distributing the imprecision and reducing the carry propagation. The Carry Cut-Back Adder (CCB) [3] also segments the addition in sub-adders, by cutting the carry propagation of the less significant bits (LSB) considering the carry propagate of the most significant bits (MSB). The carry propagation is reduced through manifold propagate signals and multiplexers. The Error-Tolerant Adder (ETA) [3] splits the addition in two sub-adders (without overlapping) and the bits are evaluated from MSB to LSB: (i) when both input bits are "1", all LSB outputs starting at this position are set to "1"; and (ii) when at least one input bit is "0", no imprecision is applied, and the next input bits are evaluated. The Lower-Part-OR Adder (LOA) [7] splits the addition into two non-overlapped sub-adders. The MSB sub-adder does not use any imprecision technique and it is a conventional full adder. The LSB sub-adder is an imprecise adder, where the carry propagation is eliminated and a simply bitwise OR is applied to the inputs, instead a XOR. An extra AND is used in the most significant bits of this LSB adder to generate the carry-in for the MSB sub-adder. The number of precise and imprecise bits can be defined according to the application requirements. Although distinct levels of precision are possible at design time, none of these related works support multiple operation points in the same implementation.

The Generic Accuracy Configurable Adder (GeAr) [8] is a design-time fully configurable imprecise adder, including the number of sub-adders, the number of precision bits, the number of sum bits and the bit width. However, its flexibility comes at the cost of increased power and delay when compared to related solutions. As the original paper reports results for FPGA devices, we have implemented GeAr using ASIC 45nm and observed 41.6% increase in power-delay product when compared to LOA. Moreover, the GeAr does not allow runtime reconfiguration of the power-precision operation point.

Aiming a better comprehension of the solutions available in the current literature, we have performed an extensive analysis of the discussed approximate adders in terms of area, power, and delay, considering 8-bit adders implemented at Nangate 45nm ASIC technology [9].

Table I presents the normalized results for six configurations of these adders: (i) ACA-II using the 4-bit overlapped sub-adders; (ii) CCB using 2-bit sub-adders and one bit for the cut-back; (iii) ETA-I using three precise and five imprecise bits; (iv) ETA-IV using three sub-adders (3-bit, 3-bit, 2-bit), two bits in the first carry generation and three bits in the second carry generation; (v) GeAr using two 5-bit overlapped sub-adders; and (vi) LOA using three precise and five imprecise bits. The considered reference for normalization was a ripple-carry adder. The results in Table I considered power dissipation, delay, silicon area, and power-delay product (PDP). One can notice from these results that the LOA outperforms the related solutions in all dimensions. Therefore, it is worth considering LOA as the basis for the development power-precision scalable adders.

III. Power-Precision Scalable Adder

The herein presented Power-Precision Scalable Adder (2PSA) is a dynamically configurable power-precision imprecise adder. Its operation points can be dynamically defined according to the system requirements or external definitions, like the battery status, the level of user attention for a specific application, or any user configuration intended to save battery, among others. The proposed 2PSA can also be statically configured at design time to support different number of operation points (*NOP*), according to the application target. The maximum level of operation points is the operator bit-width. For example, for a 4- bit 2PSA the allowed operation points are *P*, *I1*, *I2* and *I3*, where *P* is the precise operation point and *In* are the imprecise operation points with *n* imprecise bits. The minimum level of operation points allowed is one, when the 2PSA will support only the precise operation point.

The 2PSA imprecision explores the LOA concept [7] in a new dimension, using two non-overlapped sub-adders with dynamically configurable bit-widths. The adder was optimized to reduce the hardware overhead caused by the support of different operation points and it uses operand isolation [10] to increase the power efficiency. An n-bit 2PSA operator is defined with an interface containing two n-bit inputs,

TABLE I. Normalized Imprecise Adders Results

Adder	Power	Area	Delay	PDP
ACA-II (4-bit sub-adders)	1.01	0.96	0.87	0.88
CCB (2-bit sub-adders)	0.97	0.95	1.04	1.01
ETA-I (3 precise and 5 imprecise bits)	0.82	0.96	0.90	0.74
ETA-IV (3-, 3- and 2-bit sub-adders)	1.04	1.03	0.92	0.95
GeAr (5-bit sub-adders)	0.99	0.93	0.89	0.88
LOA (3 precise and 5 imprecise bits)	**0.81**	**0.90**	**0.77**	**0.62**

a power-precision control signal *PQControl* input used to dynamically define the operation point, an n-bit output and a carry-out output. Any bit-width *n* is supported by the 2PSA and the *PQControl* input bit-width depends on the supported operation points – defined as $\lceil \log_2(NOP) \rceil$.

Fig. 1 presents the block diagram of a 4-bit 2PSA, where the OI_n blocks are the operand isolation gates, IA_n blocks are the 1-bit imprecise adders and the PA_n blocks are the 1-bit precise adders. Considering the *PQControl* input values of Fig. 1, the IA_n and PA_n outputs are combined to generate the outputs for distinct precision levels. The same *PQControl* input controls the carry-in of PA_n blocks, which can be the carry out of the PA_{n-1} block or the carry out of the IA_{n-1} block. Finally, *PQControl* is used to control the OI_n blocks, defining the operand isolation of each 1-bit adder.

Fig. 2 presents the logic diagram of the IA_n, PA_n and OI_n blocks. The imprecise adder block is inspired into the LOA structure [7]. Thus, an OR gate is applied to the inputs, replacing the XOR operation used in precise adders. The carry propagation is eliminated. The precise adder block is a traditional 1-bit full adder, with XOR gates and carry propagation. Finally, the operand isolation block uses two ANDs to isolate (or not) the adder inputs, according to the description of the technique.

As presented in Fig. 1, the 2PSA implementation requires extra hardware when compared to a regular Ripple Carry Adder (RCA) [11] or to a regular LOA. This extra hardware is necessary to support the dynamic selection of the operation points. As a result, if the 2PSA is implemented to support a precise operation point, this operation point tends to dissipate more power than an isolated RCA with the same bit-width. However, the imprecise operation points will allow expressive power savings when selected.

The 2PSA allows the application (or the user) to define either a precise or a low-imprecision operation point to be used when the device is plugged into a power supply, delivering a better quality at the cost of increased power, or higher-imprecision operation points when the device is running out of battery, increasing the battery life.

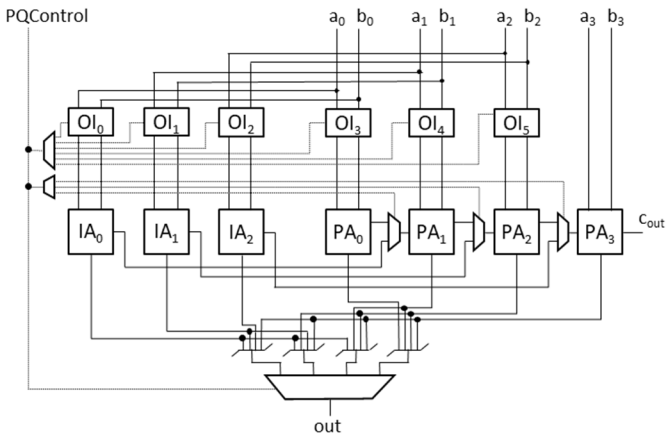

Fig. 1. Block diagram of a 4-bit 2PSA

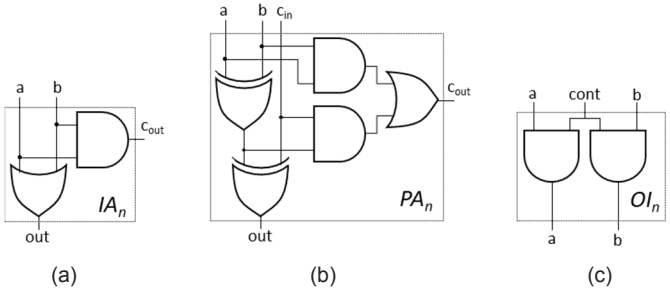

Fig. 2. 2PSA blocks (a) Ia_n, (b) PA_n and (c) OI_n.

IV. 2PSA Power-Precision Evaluation

The 2PSA quality evaluation was done by considering four adder bit-widths: 8-bit, 16-bit, 32-bit, and 64-bit. First, the 2PSA operators were described in C, by considering all allowed operation points (8, 16, 32, and 64 for 8-, 16-, 32-, and 64-bit adders, respectively, as previously discussed). Each operator was evaluated for 250,000 operations and the input samples were extracted from test video sequences [12] recommended for video encoding evaluation. The reached results were compared with a precise adder in terms of signal-to-noise ratio (SNR) – less data-dependent than PSNR or MSE, as defined in (1).

In (1), "$Average_n$" represents the average value of the evaluated precise output for each n-bit 2PSA and the MSE is the mean square error for this same 2PSA operator. The MSE was calculated according to (2). In (2), k is the number of evaluated additions (as referred above, k was set to 250,000). P_i is the precise result in position i and I_i is the imprecise result at the same position i.

$$SNR = 20\log_{10}\left(\frac{Average_n}{\sqrt{MSE_n}}\right) \quad (1)$$

$$MSE = \frac{1}{k}\sum_{i=0}^{k-1}(P_i - I_i)^2 \quad (2)$$

The power evaluations were performed by also considering the four bit-widths 2PSA operators. For such purpose, the operators were described in VHDL and synthesized for a 45nm 1.1V Nangate [9] standard cells library using Cadence Encounter RTL compiler tool. The operators were configured to support all available operation points for each evaluated bit-width. The power evaluations considered the same 250,000 operations previously referred and one evaluation was done for each operation point of each operator.

Fig. 3 presents the plots comparing the reached power savings versus the SNR response for each evaluated 2PSA operator. The power savings represent the percentage of power reduction of each imprecise operation point when compared to the precise operation point. In Fig. 3, the x-axes present the precision level, where *P* denotes precise operation point and I_n denotes imprecise operation point with *n* bits of imprecision, as previously discussed.

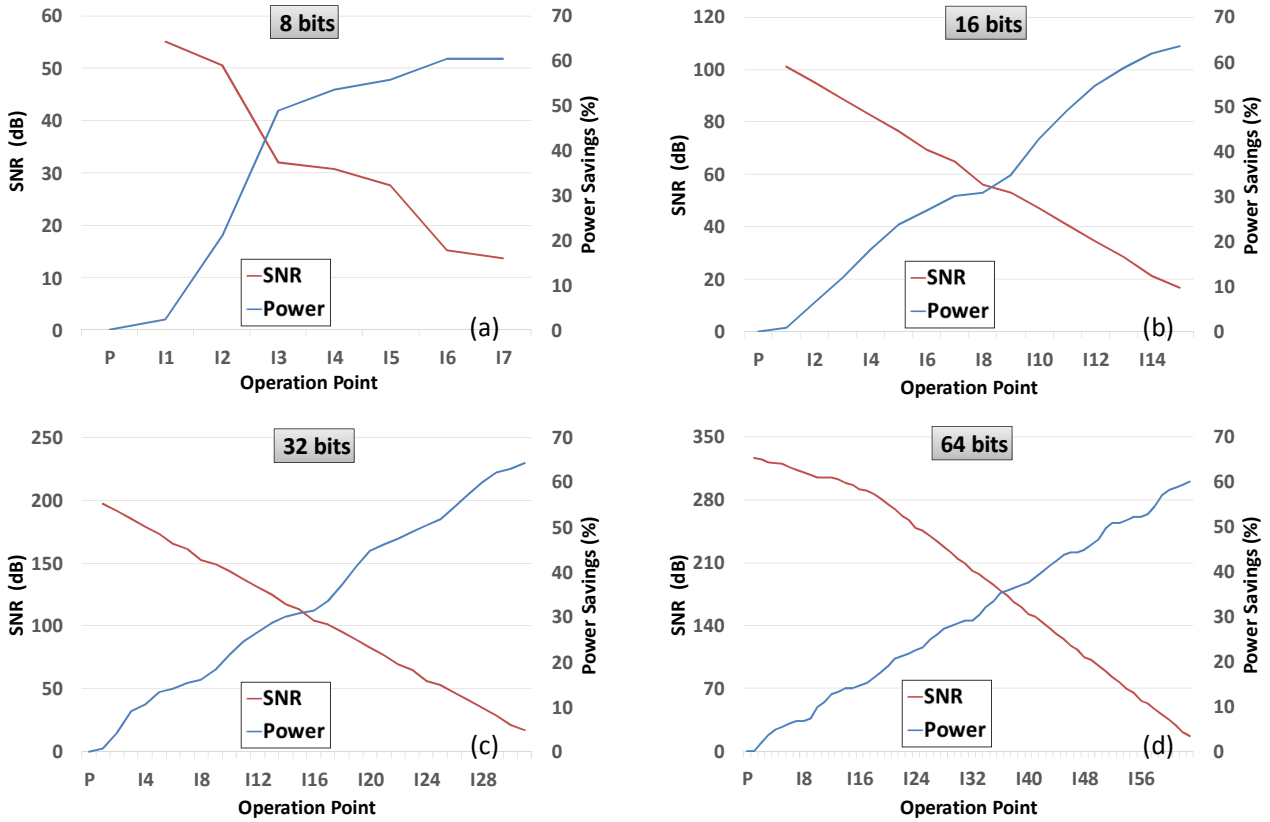

Fig. 3. Power-precision 2PSA evaluations (a) 8-bit adder, (b) 16-bit adder, (c) 32-bit adder and (d) 64-bit adder.

As expected, the higher the imprecision level, the higher the quality degradation and the power savings. Although the variation for the 8-bit operator is not very well behaved, larger operators (16-, 32-, and 64-bit adders) present a linear trend for both, SNR and power savings. Therefore, the ideal operation points depend on the application tolerance to imprecision or on application/user requirements. Taking the 32-bit adder for instance, if the application requires a 150dB SNR, the *I8* operation point may be employed leading to power savings of 16%. However, if 40dB is a tolerable quality, the *I27* operation point can be used resulting in power savings of 57% when compared to the precise operation. Note these numbers assume a 2PSA implementation supporting all possible operation points.

V. Results & Comparison: A Case Study

As previously stated, the proposed 2PSA provides total flexibility when it comes to the number of operation points and the definition of the operation points themselves. These parameters are defined at design time, whereas the selection among those operation points happens at run time. At the one hand, more operation points result in higher flexibility. At the other, the increase in the number of operation points lead to increased hardware overhead, as a result of more/larger control structures (multiplexers and OIs).

To provide a practical example, we present a case study implementation for four adder sizes (8, 16, 32 and 64 bits) with the number of operation points defined as $NOP=log_2(n)-1$, where n represents the adder bit width. One operation point is always the precise operation and the remaining are selected among all imprecise operation points. Although not formally defined, the imprecise operation points were empirically selected to provide a good observation of the power-precision scalability range, i.e., from low quality/power to high quality/power. Table II presents the operation points evaluated for each bit-width.

The case study experiments also considered operators described in VHDL and synthesized for a 45nm Nangate [9] using Cadence Encounter RTL compiler tool. The obtained results are shown in Table III, including maximum operating frequency, area, power, and quality. One can notice that smaller adders (e.g., 8 bits) leave less room for improvements, since the *I4* operation point provides a modest 19% power savings whereas posing expressive degradation (30.74dB). i.e., further increasing the imprecision would lead to extreme quality losses.

TABLE II. 2PSA Case Study: Operation Points

Bit Width / NOP	Operation Points				
8-bit / NOP = 2	P	I4	-	-	-
16-bit / NOP = 3	P	I6	I12	-	-
32-bit / NOP = 4	P	I8	I16	I24	-
64-bit / NOP = 5	P	I12	I24	I36	I48

TABLE III. 2PSA RESULTS

Bit Width	Freq. (MHz)	Area (gates)	Operation Point	Power (mW)	Power Savings	SNR (dB)
8-bit / NOP=2	900	202.7	P	0.027	-	∞
			I4	0.022	19%	30.74
16-bit / NOP=3	450	630.3	P	0.107	-	∞
			I6	0.068	36%	69.28
			I12	0.029	73%	34.59
32-bit / NOP=4	230	1259.7	P	0.118	-	∞
			I8	0.094	20%	152.46
			I16	0.066	44%	104.30
			I24	0.039	67%	56.13
64-bit / NOP=5	120	2513.7	P	0.136	-	∞
			I12	0.112	18%	305.04
			I24	0.093	32%	248.73
			I36	0.068	50%	179.11
			I48	0.048	65%	104.20

In turn, the 32-bit 2PSA provides power savings of 20% with 152.46dB response at the *I8* operation point; or power savings of 67% with 56dB response at the *I24* operation point. By comparing similar quality operation points – let us consider 8-bit adder at *I4* (30.74dB) and 16-bit adder at *I12* (34.59dB) - the power savings in the larger operator are higher. This can be better understood by observing the relation between active PAs and IAs. Whereas the 8-bit adder at *I4* uses 4 active PAs and 4 IAs (50% – 50%), the 16-bit adders at *I12* employs 4 active PAs and 12 IAs (25% – 75%). Since the IAs dissipate less power than the PAs, the more the active IAs, the higher the savings. Overall, in this specific case study, the 2PSA demonstrates to allow power-precision scalability (up to five operation points) ranging from precise computation to power savings of up to 73% with 34.59dB response for 16-bit adders or power savings of up to 65% with 104.20dB response for 64-bit adders.

The absence of any other power-precision scalable adders structures supporting multiple operation points forbids a direct comparison with previously published works. Nevertheless, in order to draw a proper comparison, we have designed adders with the same capabilities as the 2PSA case study presented above, by replicating adders using LOA style. For instance, to implement the same capabilities of the 16-bit 2PSA (NOP=3), we have employed one 16-bit RCA (precise adder), one 16-bit LOA adder with 6 imprecise bits (to behave as the 16-bit 2PSA at I6 mode), and one 16-bit LOA adder with 12 imprecise bits (to behave as the 16-bit 2PSA at I12 mode). A similar *PQControl* is used to select the operation mode. The output is selected using multiplexers and operand isolation is used at the input of unused operators for fairness of comparison.

Fig. 4 presents the comparison between the proposed 2PSA and the solution using replicated adders. It demonstrates that the area reductions allowed by the 2PSA ranges between 54% to 73%, whereas the power savings range from 18% to 73%. As expected, as higher is the number of operation points for a same bit-width, as higher are the power and area savings when comparing 2PSA with implementations using independent adder to support the same operation points. In turn, for each bit width, operation points with higher imprecision levels present higher power savings.

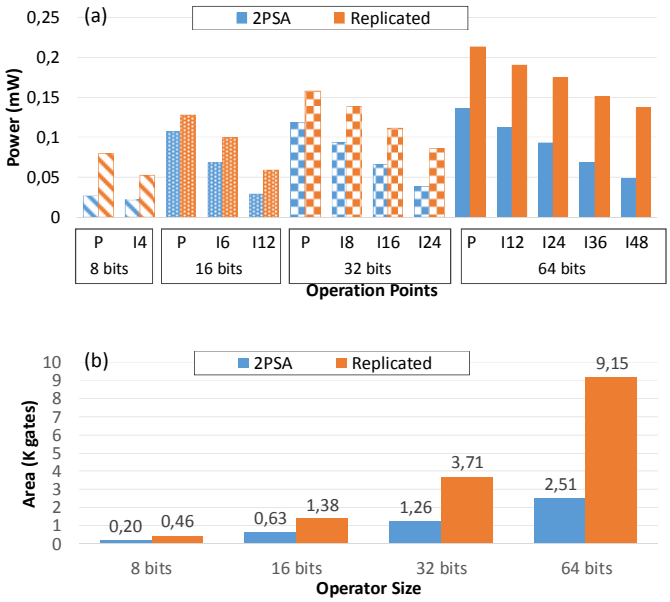

Fig. 4. (a) Power and (b) area comparison between 2PSA and the replicated adders architecture

VI. CONCLUSIONS

This article proposed the Power-Precision Scalable Adder (2PSA). The 2PSA can be configured at design time to support a variety of power-precision operation points, according with the runtime application or/and the user definition. The runtime selection of these operation points allows a dynamic selection of power-precision operation points according to the system status, like type of power supply or level of battery charge, or according to any other user configuration. The 2PSA uses operand isolation to increase the power savings, which can vary from 2% to 60% for 8-bit adders and from 2% to 60% for 64-bit adders, depending on the selected operation point. 2PSA was also optimized to reduce the hardware resources required to support the multiple levels of power-precision operation points and these optimizations in area also contribute to reduce the power and delay. When compared with other solutions able to support the same imprecision levels, but without optimizations, the 2PSA is able to reach savings from 54% to 73% in area and from 18% to 73% in power, depending on the adder bit-width and on the selected operation point.

REFERENCES

[1] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in *Proc. IEEE European Test Symposium (ETS)*, Avignon, France, 2013, pp. 1–6, DOI: 10.1109/ETS.2013.6569370.

[2] V. Chippa, S. Venkataramani, S. Chakradhar, K. Roy; and A. Raghunathan, "Approximate computing: an integrated hardware approach," in *Proc. Asilomar Conference On IEEE Signals, Systems And Computers (Asilomar)*, Pacific Grove, USA, 2013, pp. 111–117, DOI: 10.1109/ACSSC.2013.6810241.

[3] V. Camus, J. Schlachter, and C. Enz, "A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, USA, 2016, pp. 1–6, DOI: 10.1145/2897937.2897964.

[4] N. Zhu, W. Goh, W. Zhang, K. Yeo, and Z. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. on Very Large Scale Int. Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010, DOI: 10.1109/TVLSI.2009.2020591.

[5] N. Zhu, W. Goh, G. Wang, and K. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *Proc. IEEE Int. SOC Design Conf. (SOCDC)*, Seoul, South Korea, 2010, pp. 323–327, DOI: 10.1109/SOCDC.2010.5682905.

[6] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, USA, 2012, pp. 820–825, DOI: 10.1145/2228360.2228509.

[7] H. R. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Trans. on Circ. and Syst. I: Reg. Papers*, vol. 57, no. 4, pp. 850–862, Dec. 2010, DOI: 10.1109/TCSI.2009.2027626.

[8] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, USA, 2015, pp. 1–6, DOI: 10.1145/2744769.2744778.

[9] Nangate FreePDK45 Open Cell Library. Silvaco, [Online]. Available: https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library/index.html, Accessed on: May 8, 2019.

[10] N. Banerjee, A. Raychowdhury, S. Bhunia, H. Mahmoodi, and K. Roy, "Novel low-overhead operand isolation techniques for low-power datapath synthesis," in *Proc. Computer Design: VLSI in Computers and Processors (ICCD)*, San Jose, USA, 2005, pp. 1–6, DOI: 10.1109/ICCD.2005.80

[11] N. Weste and D. Harris, *CMOS VLSI Design, A Circuits and Systems Perspective*, 3rd ed., Englewood Cliffs, NJ, USA: Prentice Hall, 2004.

[12] F. Bossen, *Common test conditions and software reference configurations*, document JCTVC-L1100, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2013.