Application-Driven Synthesis of Energy-Efficient Reconfigurable-Precision Operators

Application-Driven Synthesis of Energy-Efficient Reconfigurable-Precision Operators

Daniele Jahier Pagliari
Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, Italy
Email: daniele.jahier@polito.it

Massimo Poncino
Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, Italy
Email: massimo.poncino@polito.it

***Abstract*—The increasing performance demands in emerging Internet of Things applications clash with the low energy budgets of end-nodes. Therefore, hardware operators able to reconfigure their computational precision at runtime are increasingly employed in these devices, to obtain good-enough results at minimal energy costs. Among the many methods proposed to implement such operators, Dynamic Voltage and Accuracy Scaling (DVAS) is particularly promising, due to its broad applicability and low overheads. However, a straight-forward application of DVAS conflicts with the optimizations performed by classic EDA algorithms, and does not yield the expected results.**

In this paper, we propose a novel synthesis algorithm for reconfigurable-precision circuits, that allows to integrate DVAS in a standard implementation flow. Moreover, we show how this algorithm can exploit information about the application, namely on the frequency of usage of each precision, to further reduce the total energy consumption. Applying our method to the popular LeNet neural network for digit recognition, we are able to reduce the energy due to Multiply-And-Accumulate (MAC) operations by 25%, compared to a straight-forward application of DVAS.

I. Introduction and Background

Combining energy efficiency with high performance is a critical problem in the design of digital devices for mobile and Internet of Things (IoT) applications. Emerging applications require end-nodes able to perform increasingly complex tasks (recognition, classification, etc.), over long time spans (months, years) within a single battery cycle. Fortunately, many of these applications are *error tolerant*, i.e. can tolerate a relaxation of the computational quality without a significant impact on the final results. This is the case for tasks that have humans as final users, e.g. multimedia, or that are based on statistical/aggregative algorithms, e.g. machine learning [1]. For example, several works have shown that both inference and training of deep neural networks can be performed with low-precision fixed-point operations, whilst maintaining the network performance very close to the nominal value (obtained with single/double-precision floating-point) [2]–[4].

Error tolerance allows trading off computational quality with other metrics, mainly energy efficiency, and is the enabler for the so-called *quality-scalable* design paradigm. In recent years, this paradigm has been applied at all levels of abstraction, from single devices to software [5], [6].

A fundamental element of a quality-scalable computational platform is the availability of hardware data-path operators (e.g. adders, multipliers, etc.) able to dynamically reconfigure their precision at runtime [7]–[11]. Early works on the design of these operators propose several architectural modifications to support multiple "quality-modes". One solution consists in implementing a simplified but approximate version of the operator, and then selectively activating some *error recovery* circuitry when high quality computations are required [7], [8], [10]. An alternative is *dynamic segmentation*, in which the operator is split in sub-blocks, and signals from one block to the next (e.g. the carry in an adder) can be selectively replaced by simpler but inaccurate *prediction* circuits [9], [10].

One drawback of architectural techniques for quality-scalable circuits is the lack of automation and generality. Indeed, most of these methods are only applicable to one particular adder or multiplier architecture, and require the designer to perform *manual* modifications on the specification. Moreover, several works have shown that these designs have worse energy versus quality tradeoffs compared to classic operators working at reduced bit-width [11], [12].

A promising approach that takes these considerations into account is Dynamic Voltage and Accuracy Scaling (DVAS) [11]. In DVAS, multiple quality-modes are obtained simply *disabling* (i.e. zeroing) some of the input Least Significant Bits (LSBs). This provides a dynamic power saving, thanks to the reduction in circuit activity. Moreover, decreasing the bit-width can also reduce the operator delay, since in theory the critical paths in a data-path circuit go from the input LSBs to the output MSBs. This additional *slack* can be leveraged for supply voltage scaling, for further dynamic and leakage savings. DVAFS [4] extends this concept further, combining it with frequency scaling and subword-parallel operation.

A straight-forward application of DVAS, however, conflicts with the optimizations performed in a standard EDA flow for ASICs [13]. In fact, EDA tools optimize critical paths in the circuit for performance, whereas less critical paths are exploited for area and power reduction [14]. To this end, gates in non-critical paths are mapped to standard cells with small widths and high threshold voltages, hence slower but with smaller dynamic and static power consumption. This causes an increase in the delays of short paths, which become comparable to critical ones. Eventually, after physical design, the circuit is likely to have a very skewed distribution of timing slacks, known as the *wall-of-slack* (WOS) [7]. An example of WOS is shown in Figure 1.

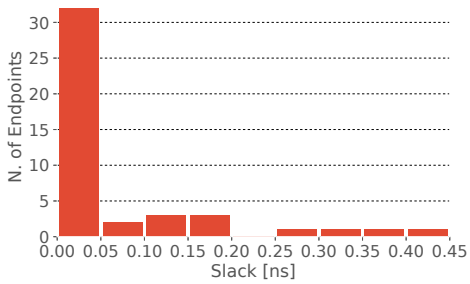

Fig. 1: Endpoint slack distribution for a 16x16-bit MAC with 44-bit accumulator, implemented in 28nm FDSOI technology, at $f_{clk} = 1.25$ GHz, $V_{DD} = 0.95$ V. The graph includes all paths that cross combinational logic.

Due to the WOS, the previously mentioned assumption on paths delays is no longer valid, and even at very low bit-widths, voltage scaling can seldom be applied in DVAS without incurring timing violations. The authors of [11] do mention this problem, and propose to solve it through modifications in the EDA flow which, however, they do not describe in the paper. One low-overhead solution is provided in [13], but this method depends on the availability of technology-specific knobs, namely the possibility dynamically tuning the threshold voltage via back-biasing.

In this paper, we propose the first algorithm for the synthesis of reconfigurable-precision circuits based on DVAS, that can be fully integrated with standard EDA tools. The algorithm identifies the optimal supply voltage for each target bit-width, and constrains the synthesis process accordingly, preventing the formation of the WOS. Throughout this process, information on the target application, obtained through off-line characterizations, can be leveraged to drive the optimization. Thus, our method is particularly suited for application-specific hardware accelerators. To show the effectiveness of our solution, we apply it to a Multiply-And-Accumulate (MAC) operator, which is the key data-path component in accelerators for artificial neural networks [4]. On this benchmark, we obtain a total energy saving $> 25\%$ compared to a standard implementation of DVAS.

II. Application-Driven Synthesis Algorithm

A. Multi-scenario optimization in DVAS-based circuits

All state-of-the-art commercial synthesis and place and route (P&R) tools support *multi-scenario* optimization [15]. In this procedure, the tool is instructed to simultaneously consider multiple *corners* (i.e. process, supply voltage and temperature or PVT points), each associated with a corresponding *operating mode* (i.e. a set of constraints), during optimization.

In a DVAS-based circuit, this functionality can be leveraged to ensure that the circuit does not have timing violations at the desired V_{DD}, for each considered bit-width mode. To this end, a separate scenario for each mode must be created, setting the desired V_{DD} as part of the corresponding PVT corner.

Then, *case analysis* features of the tool must be used, to mimic the effect of zeroing LSBs. In particular, constant logic-0s must be imposed on disabled inputs, so that the tool can propagate this constraint to the circuit internal signals. Paths that have zeros as inputs are marked as *false*, and the tool does ignore them during the optimization of the corresponding scenario. Notice that the tool will ignore false paths when checking timing compliance and power consumption in reduced-precision scenarios, but will not eliminate the corresponding logic, as the same paths are not false in the maximum-precision scenario.

Providing the tool with these scenarios will force it to *concurrently ensure timing compliance in all V_{DD} and bit-width combinations*. Thus, the design will be able to operate at reduced V_{DD} when low bit-widths are used, consuming less power, thanks to a more graceful-distribution of timing paths.

The drawback of this solution is an increase in area occupation and power consumption at *maximum precision*. In fact, to allow lower V_{DD} operation, gates belonging to true paths (i.e. not disabled) in low bit-width modes will be upsized and/or mapped to lower threshold voltage devices compared to a standard implementation of the same circuit. Fortunately, in many applications, maximum precision is only seldom required [2]–[4], hence this solution has the potential to be advantageous in terms of *total energy*.

B. Greedy synthesis algorithm

The methodology described in Section II-A assumes that the best V_{DD} value for each bit-width mode is known a priori. In practice, finding this value is not trivial. In this work, we propose to find an appropriate V_{DD} for each bit-width using the greedy incremental synthesis algorithm shown in Figure 2.

```
1:  procedure GREEDY INCREMENTAL SYNTHESIS
2:      V_start = current supply voltage, initialized to nominal V_DD
3:      s_0 = nominal scenario [V_start, b_max]
4:      S = list of scenarios, initialized to s_0
5:      C = nominal circuit netlist, synthesized in scenario s_0
6:      for all reduced bit-widths b (in decreasing order) do
7:          s_new = [V_start, b] (same V_DD as previous scenario)
8:          P_new = weighted power of C in scenarios S + s_new
9:          V_new = V_start
10:         do
11:             s_ref = s_new, P_ref = P_new, V_ref = V_new
12:             V_new = decrease V_new
13:             s_new = [V_new, b]
14:             C_new = incr. synthesis of C in scenarios S + s_new
15:             Ts = worst slack of C_new in scenarios S + s_new
16:             P_new = weighted power of C_new in scenarios S + s_new
17:         while V_new > V_min and Ts ≥ 0 and P_new < P_ref
18:         S = S + s_ref
19:         V_start = V_ref, C = C_new
20:     end for
21: end procedure
```

Fig. 2: Proposed greedy incremental synthesis algorithm.

This procedure takes as input a gate-level netlist of the target circuit, synthesized in nominal conditions, i.e. maximum V_{DD} (V_{start}) and bit-width (b_{max}). Synthesis constraints (clock

frequency, boundary conditions, etc.) are also received as inputs, as well as the set of reduced bit-width modes to consider during the process, and the available V_{DD}s. In the pseudo-code, the $+$ symbol used with scenarios (lines 8, 14, 15, 16) corresponds to the list concatenation operation.

The core of the algorithm spans the available set of supply voltages, starting from the nominal value and progressively decreasing it. While doing so, it adds one bit-width mode at a time to the considered set, in decreasing order.

For each new bit-width, the first step (line 8) is an assessment of the power consumption of the circuit when the new mode uses the *same* supply voltage as the previous one (i.e. the only difference between the modes is in the number of zeroed LSBs). This power estimation, as well as the one in line 16, considers *all bit-width modes* together, as detailed in Section II-C. Then, V_{DD} is decreased (line 12), and an *incremental* synthesis (line 14) is performed on the circuit, so that the tool can try to enforce timing compliance at a lower V_{DD} for the same bit-width. During the re-synthesis phase, the tool also accounts for previous bit-widths, using the multi-scenario functionality described in Section II-A. In some cases (i.e. when V_{DD} is too low), the tool can fail to resolve timing violations. Therefore, the resulting netlist undergoes a Static Timing Analysis (STA) in line 15, where the worst setup and hold slacks in *all* bit-width modes are evaluated. Then, the power of the modified netlist is evaluated in line 16.

The progressive scaling of V_{DD} is continued until it results in a *reduction of total power*, and the tool is able to avoid timing violations. The lowest V_{DD} that satisfies these two conditions is selected as final supply voltage for the considered bit-width, and the corresponding scenario is saved (line 18). Then, a new precision mode is added, and the procedure is repeated starting from the current V_{DD}, i.e. V_{start}.

The proposed algorithm requires to re-synthesize the operator several times. Thus, we perform this optimization at gate-level, rather than after P&R. Although the latter solution would yield more accurate results, it would also require a longer execution time. Moreover, as long as power estimations after synthesis and P&R are correlated [15] (although exact values might differ), it makes sense to *select* the appropriate V_{DD}s post-synthesis, and then *enforce* them during P&R. In this way, a single (multi-scenario) P&R is performed, using the previously selected V_{DD} for each bit-width.

C. Application-driven power weighting

A key feature of the algorithm in Figure 2 is the way in which the total power consumption of the operator is assessed. Specifically, as reported in the figure (lines 8 and 16), we consider a *weighted* contribution from each scenario, that is:

$$P_{tot} = \sum_i w_i P_i \tag{1}$$

where P_i is the power in each bit-width mode. P_i depends on the V_{DD} applied to the circuit in that mode, and is affected by the re-synthesis phase, because of the possible gates resizing/remapping performed by the tool when further scenarios are added.

Using $w_i = 1$ for all i, would give the same importance to the power consumption in all modes. However, our main concern is *energy* efficiency, which also depends on the usage frequency of each mode. Therefore, we propose to select weights w_i in a way that lets our algorithm minimize a "proxy" of the total energy consumption, leveraging information about the target application. In practice, this can be done assigning to w_i a value proportional to the *probability* that a given operation requires mode i, which can be obtain through an offline characterization of the application.

Notice that the algorithm of Figure 2 is greedy, in that it stops decreasing V_{DD} at the first minimum of Equation (1), although this might not be the global best. For example, if the nominal V_{DD} and bit-width for a circuit are 0.9V and 16-bit respectively, using 0.8V for the 8-bit mode might cause a slight increase in the total weighted power, due to the impact of gates resizing on the consumption at 16-bit. However, decreasing the 8-bit supply voltage further, e.g. to 0.7V, might reduce the power consumption at that precision enough to improve the value of Equation (1). The impact of this occurrence depends on the values of weights w_i, on the topology of the considered circuit, and on synthesis constraints (especially f_{clk}).

The advantage of the greedy approach is that it requires a smaller number of re-synthesis. Alternatively, an exhaustive search could also be performed, considering all possible supply voltages (among those smaller than the ones used at higher bit-widths). This would guarantee optimality at the cost of longer execution time, and it would still be feasible if the number of bit-width modes and V_{DD}s is not too large. In practice, in our test case, the situation described above never verifies, and the greedy and exhaustive solutions coincide.

III. Testcase Results

We demonstrate the effectiveness of the methodology described in Section II on a 16x16-bit Multiply-And-Accumulate (MAC) operator, with a 44-bit accumulator. We described the architecture of the MAC in VHDL, and synthesized it targeting a 28nm FDSOI technology library from STMicroelectronics. The clock frequency and nominal supply voltage were set to $f_{clk} = 1.25GHz$ and $V_{DD} = 0.95V$ respectively. In the following experiments, we considered supply voltages from $0.95V$ to $0.60V$ in steps of $0.05V$. For logic synthesis (both the full run to get the nominal circuit, and the incremental re-synthesis of our algorithm) we used Synopsys Design Compiler L-2016.03. P&R was executed with Cadence Innovus 16.1, while STA and power analyses were performed in Synopsys PrimeTime L-2016.06. The algorithm of Figure 2 was implemented in Python 3.5, and internally makes use of both Design Compiler and PrimeTime.

In the following, we assumed that the MAC operator is part of an accelerator for the inference (i.e. classification) phase of the well-known LeNet-5 neural network, for handwritten digit recognition [16]. We selected this benchmark due to

Bit-Width	Number of MACs
16	$0.01 \cdot 10^6$
8	$1.6 \cdot 10^6$
4	$0.3 \cdot 10^6$

TABLE I: Number of MAC operations performed at each bit-width during the classification of one digit in LeNet-5 [16]

its relevance, and the availability of previous bit-width characterizations. However, notice that our method is applicable to any other domain for which the number of operations performed at different bit-widths can be estimated through offline characterizations.

MAC operations constitute the bulk of the computations performed during neural networks inference. Many researchers have shown that these operations can be done in fixed-point, with relatively low bit-widths, without a significant impact on the classification accuracy compared to floating point baselines [2]–[4]. In particular, the authors of [4] have shown that better accuracy is obtained if a different bit-width is used for each *layer* of the network. In our experiments, we considered the same bit-widths derived in [4] for the two convolutional layers of LeNet-5. Moreover, we assumed that 16-bits are used for Fully Connected (FC) layers, in accordance to [3]. With these assumptions, the number of MAC operations performed at each bit-width for the classification of one digit using LeNet-5 is reported in Table I.

We considered three different DVAS-based MAC operators, each supporting 4-bit, 8-bit and 16-bit modes. A first *Classic* version, used as baseline for comparison, was obtained without synthesis optimizations (i.e. applying DVAS to a circuit implemented with a standard flow). Additionally, we applied the algorithm of Figure 2 twice, first setting all weights w_i to 1 (*Uniform* version), and then setting weights proportional to the usage probability of each bit-width mode, i.e. $w_{16} = 0.01$, $w_8 = 1.6$, $w_4 = 0.3$ (*Weighted* version).

A. Power, area and energy comparison

Table II shows the supply voltages used in reduced bit-width modes by the three circuits, as well as the area overheads with respect to the Classic version. Figure 3 shows the post-P&R total power consumption (including leakage and dynamic contributions) obtained by the different MACs at each bit-width. In the Classic MAC, scaling the V_{DD} to $0.90V$ causes

MAC Version	8-bit V_{DD} [V]	4-bit V_{DD} [V]	Area Ovr. [%]
Classic	0.95	0.95	-
Uniform	0.90	0.90	9
Weighted	0.75	0.70	16

TABLE II: Reduced bit-width V_{DD} and area overhead (normalized to the Classic version) of the three MAC versions.

timing violations at both 8-bit and 4-bit precision, due to the WOS. In the Uniform version, conversely, thanks to the multi-scenario optimization, the synthesis tool enforced timing compliance at $0.90V$ for 4 and 8 bit modes. The V_{DD} was not reduced further because of the high power overheads

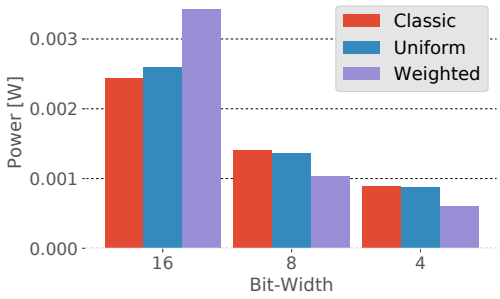

Fig. 3: Power consumption in different bit-width modes.

that this would cause at 16-bit. However, this intermediate solution does not take the statistics about application data into account. In particular, it ignores that maximum-precision is only required in about 0.005% of the MAC operations.

Conversely, in the Weighted case, our algorithm selects a much more aggressively scaled V_{DD} for both 8 and 4-bit, since these two modes are much more relevant than 16-bit in LeNet inference. This causes an area overhead of 16% compared to the Classic version, but allows significant power savings at reduced bit-widths (27% at 8-bit, and 31% at 4-bit).

The global advantage of the Weighted solution is shown in Figure 4, which reports the total energy consumption due to MAC operations for classifying one digit in LeNet-5, if the three MAC versions are used. Results are normalized to the Classic implementation. While the Uniform version

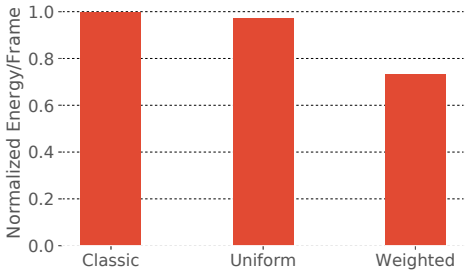

Fig. 4: Energy due to MAC operations for classifying one frame in LeNet-5.

only consumes $\approx$ 5% less energy compared to a standard design, the Weighted solution reduces consumption by $\approx 27\%$. Importantly, this result also accounts for the additional leakage energy of the Weighted version, caused by the larger area.

IV. Conclusions

We have proposed a new synthesis algorithm for data-path circuits that can operate in multiple precision modes, based on the principles of DVAS. This algorithm is fully automated, and leverages industrial EDA tools for synthesis and place and route. Moreover, we have shown that the effectiveness of our method improves if characterizations of the target application are available.

When tested on a MAC operator, used in the acceleration of neural networks inference, our solution reduces the total energy consumption of 27% compared to a standard DVAS implementation.

REFERENCES

[1] V. Chippa, S. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. of the 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–9.

[2] M. Courbariaux, Y. Bengio, and J. David, "Low precision arithmetic for deep learning," *CoRR*, vol. abs/1412.7024, 2014. [Online]. Available: http://arxiv.org/abs/1412.7024

[3] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *CoRR*, vol. abs/1502.02551, 2015. [Online]. Available: http://arxiv.org/abs/1502.02551

[4] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Dvafs: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2017, pp. 488–493.

[5] Q. Xu, N. S. Kim, and T. Mytkowicz, "Approximate computing: A survey," *IEEE Design and Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.

[6] M. Alioto, "Energy-quality scalable adaptive vlsi circuits and systems beyond approximate computing," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2017, pp. 127–132.

[7] A. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 820–825.

[8] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2014, pp. 95:1–95:4.

[9] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 48–54.

[10] D. Mohapatra, V. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2011, pp. 1–6.

[11] B. Moons and M. Verhelst, "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2015, pp. 237–242.

[12] B. Barrois, O. Sentieys, and D. Menard, "The hidden cost of functional approximation against careful data sizing - a case study," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2017, pp. 181–186.

[13] D. Jahier Pagliari, Y. Durand, D. Coriat, A. Molnos, E. Beigne, E. Macii, and M. Poncino, "A methodology for the design of dynamic accuracy operators by runtime back bias," in *Proc. of the Design, Automation and Test in Europe, Conference and Exhibition (DATE)*, 2017, pp. 1165–1170.

[14] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[15] *Design Compiler User Guide L-2016.03-SP5-2*, Synopsys, Inc., 2016.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.