Evaluation of Variable Bit-Width Units in a RISC-V Processor for Approximate Computing

Geneviève Ndour, Tiago Trevisan Jost, Anca Molnos, Yves Durand, Arnaud Tisserand

Evaluation of Variable Bit-Width Units in a RISC-V Processor for Approximate Computing

Geneviève NDOUR
CEA, LETI, Université Grenoble Alpes, Grenoble, France
CNRS, INRIA, IRISA, Université Rennes 1, France
genevieve.ndour@cea.fr

Tiago TREVISAN JOST, Anca MOLNOS, Yves DURAND
CEA, LETI, Université Grenoble Alpes, Grenoble, France
{tiago.trevisanjost,anca.molnos,yves.durand}@cea.fr

Arnaud TISSERAND
CNRS, Lab-STICC UMR 6285, Université Bretagne Sud, Lorient, France
arnaud.tisserand@univ-ubs.fr

ABSTRACT

Among various power reduction methods, variable bit-width arithmetic units have been proposed in approximate computing literature. In this paper, we add a variable bit-width memory unit in a RISC-V processor. Integrating both computation and memory units with variable bit-width leads to a power reduction: from 7% to 29% for Sobel filter application and from 13% to 24% for an application that computes the position of a robotic arm (forwardk2j). We also propose a global energy model for a RISC-V processor with variable bit-width units (for computation and memory). This model allows us to evaluate the impact of various parameters in both the software application (e.g., the amount of instructions that can be executed with a reduced bit-width) and the hardware architecture (e.g., impact of potential reduction for each unit).

KEYWORDS

Approximate computing, variable bit-width, embedded systems, energy optimization, systems on-chip

1 INTRODUCTION

Approximate computing provides several techniques to investigate various trade-offs between application accuracy and energy or performance [5]. For instance, several approximate units have been proposed. Due to constraints in terms of energy and/or silicon area in embedded systems, integer approximate units have been implemented [2]. However such integer approximate units have been mostly evaluated separately, i.e., not in a complete processor with a running application. The stand-alone evaluation of approximate units is not sufficient for an estimation of global energy reduction of a given application.

We aim to evaluate the potential interest of variable bit-width integer units. In this type of unit, only a configurable number of most significant bits (MSBs) is active. The number of active MSBs is configurable at runtime. We target general purpose embedded systems workloads. These variable bit-width units are integrated into a RISC-V processor to evaluate their impact on applications in terms of accuracy and energy reduction.

Previous work [6] proposes a RISC-V processor with variable bit-width arithmetic operators that are the most common in the literature, e.g., approximate additions [2], approximate multiplications [3]. The results indicate moderate energy gains due to the fact that only the computation units are approximated; all the other parts are accurate.

Our first contribution extends the previous work with bit-width configuration in the data memory, i.e., load/store only the active MSBs from/in the data memory. We evaluate the energy reduction with configurable bit-width in both computation and data memory units.

Our second contribution consists of a generic energy model that includes both software and architecture parameters. It aims to allow software and hardware designers to investigate the effects of potential optimizations performed on software and/or on units in approximate computing. The software designer can have, an early-stage insight into the limits of energy reduction reachable for a given application. The proposed energy model allows a hardware designer to find the most important architectural parameters that could be optimized for energy efficiency and to have an idea on the possible energy reduction in the optimization process.

The paper is organized as follows. Section 2 discusses the related work. Section 3 presents the extended RISC-V and the energy model per instruction. Section 4 presents the results of accuracy vs energy trade-off study. Section 5 describes our global energy model. Section 6 concludes the paper.

2 RELATED WORK

Several strategies have been proposed in approximate computing for the reduction of computation costs (e.g., energy) [5]. For example, approximate units have been implemented [4],[19],[3],[16], [2], [1]. The energy consumption of several instructions have been evaluated in several architectures. For example, the energy consumption of individual instructions have been estimated for two different ARM processors: Cortex-A7 and Cortex-A15 in [15] and on a RISC processor in [13]. However all the above energy estimations are for accurate instructions.

To reduce the energy consumption of accurate instructions, approximate units have been proposed for low power computation and low power storage. For example [7] implements a 16-bit multiplier with supply voltage scaling method to reduce the bit-width of the operator; as a result, the energy consumption of the multiplier is reduced by up to 39%. [2] proposes an accuracy-configurable approximate (ACA) adder, which reduces energy consumption by up to 30%. However the proposed operators are evaluated separately and are not evaluated on an application executed inside a complete processor.

Global energy evaluation solutions on applications have been proposed. For example [10] and [8] evaluate approximate instructions on applications such as: FFT, LU, Raytracer. The results indicate that the energy can be reduced by up to 50% in [10] and up to 35% in [8]. However the proposed solutions are only for floating-point applications and the estimated energy models are for arithmetic and memory units and do not include applications parameters.

In [6], the evaluation of variable bit-width integer units is performed on a Sobel filter in a RISC-V processor. However only the integer additions and multiplications units are approximated and that seems not sufficient for a high energy reduction.

To the best of our knowledge, we are the first to evaluate the energy reduction with both variable bit-width integer arithmetic and data memory units on applications and to propose a generic

energy model that combines architecture and software parameters to estimate the global energy reduction due to optimizations of applications source code and/or hardware units in approximate computing.

3 RISC-V PROCESSOR WITH VARIABLE BIT-WIDTH UNITS

Figure 1 presents the RISC-V processor [9] extended with variable bit-width units for both computation (*a.EXE*) and memory units (*a.LSU*). *a.EXE* performs computations on a number of active MSBs. *a.LSU* handles the load/store from/in the data memory of words composed of a variable number of active MSBs. One dedicated instruction is added to configure the bit-width for *a.EXE* and *a.LSU*.

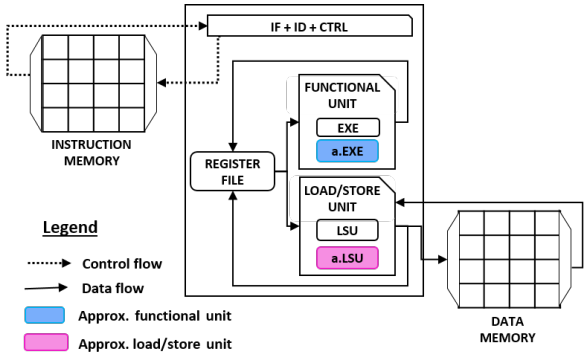


Figure 1: Extended RISC-V Architecture

3.1 RISC-V experimental environment

The aim of our experiments is to investigate the accuracy vs energy trade-off of a given application executed with variable bit-width integer operators. Most of the benchmarks proposed in the state of the art are implemented in floating-point. The evaluated applications are converted into fixed-point format before executing them in the extended RISC-V. To perform the experiments, we annotate the source code with pragmas that indicate the data format and delimits the parts of the fixed-point C source code to be executed with reduced bit-width. A compiler support based on LLVM is implemented to handle pragmas directives [14]. The RISC-V spike simulator [12] is augmented with the capabilities to count, for a given application, the number of instructions executed for each class. For each instruction class, this number is multiplied with the average energy value of the instruction class provided in Subsection 3.2. The sum of the energy values for each instruction class models the total energy consumed by the application.

3.2 Energy per instruction

The energy model per instruction is constructed using measurements on a in-house 28nm FD-SOI test-chip. This chip includes a processor core that follows the architecture described on Figure 1, connected to 256KB of memory, split into 64KB of instruction memory and 192KB of data memory. The energy consumption variations due to the input data are under 15%, and hence in what follows we utilize an average value. We split the RISC-V instructions into classes of similar energy consumption. The accurate instructions classes (and their average energy values related to the multiplication) are: arithmetic and logic instructions (`al`: 0.68), multiplications (`mul`: 1), branch instructions (`br`: 1.56), store (`st`: 0.78) and load (`ld`: 0.71). The memory is implemented in a low-voltage technology, which explains why the `ld` and `st` instructions consume less energy than a multiplication.

The energy values of variable bit-width multiplication and arithmetic and logic instructions are obtained by considering that the consumption of the *a.EXE* part varies with the bit-width as specified in [7], whereas the energy consumption of the other core parts are the same as in the accurate case. By looking at the data memory, we estimate that 40% of the accurate energy is not scalable, and the 60% varies linearly with the bit-width. Note that this percentage is dependent on the design of the memory. Setting it to a value representative for another memory implementation does not invalid our investigation method. The implementation of the approximate operators introduces an overhead energy. This overhead is caused by the extra elements needed to partition the operators into several threshold voltage domains, and it is taken into account in our model. Figure 2 presents the energy values of accurate and variable bit-width instructions classes.

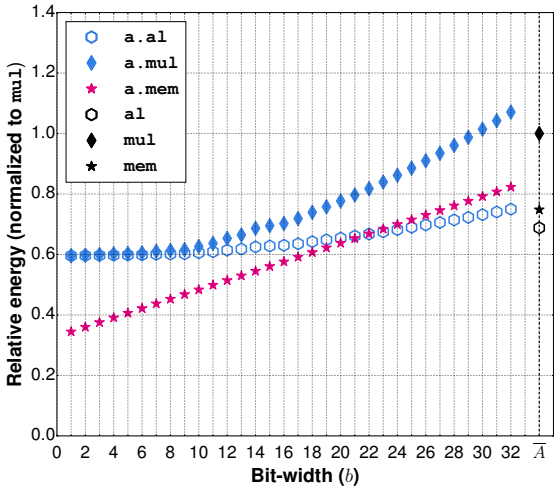


Figure 2: Relative energy values of variable bit-width and accurate instructions

4 IMPACT OF VARIABLE BIT-WIDTH UNITS ON BENCHMARK APPLICATIONS

The evaluation is performed on two applications cases studies from the Axbench suite [18]: Sobel filter and forwardk2j. Sobel filter is a kernel used in image processing for edge detection. To evaluate the accuracy in Sobel filter with reduced bit-width, we utilize the structural similarity metric (SSIM) [17] that is more correlated to human perception of image quality than other metrics such as the root mean square error (RMSE) or the peak signal to noise ratio (PSNR) [11]. The forwardk2j is a kernel used in robotics. It takes as input the angles of a 2-joint robotic arm and computes the position of its end-effector. To evaluate the errors induced by the reduced bit-width, we utilize the relative error metric. To compute the quality metric values of the above benchmarks, we consider as reference values, the outputs returned by the original floating-point program. Several optimizations have been applied on the applications source code to reduce the costs due to, for example, address computations, loop indexing. Figure 3 and Figure 4 present the instruction

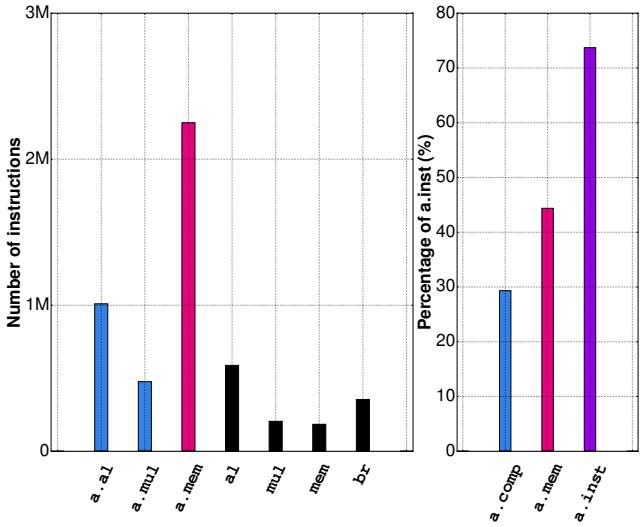


Figure 3: Sobel filter instructions breakdown

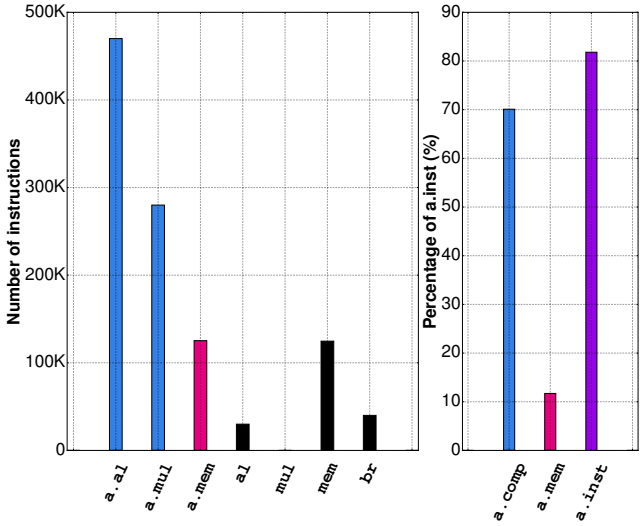


Figure 4: Forwardk2j instructions breakdown

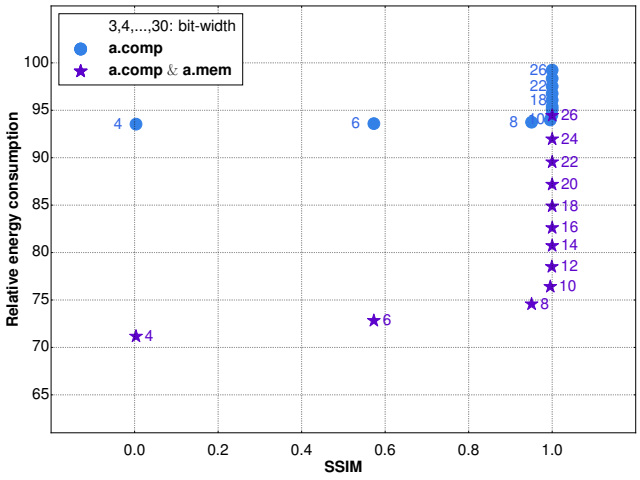


Figure 5: Sobel filter accuracy vs energy trade-off

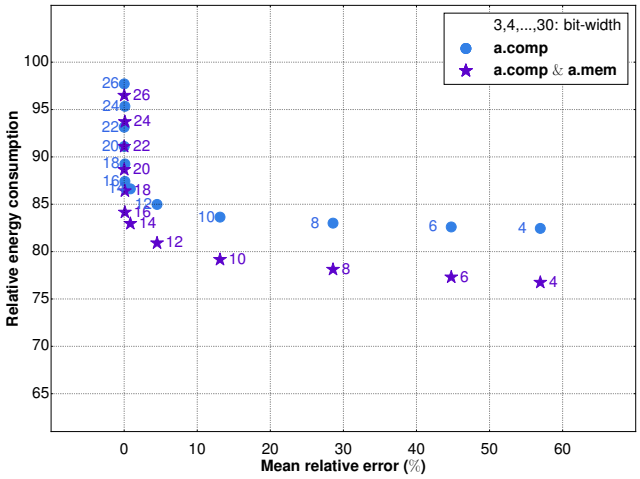


Figure 6: Forwardk2j accuracy vs energy trade-off

breakdown of the two evaluated applications. Sobel filter includes 73.7% of instructions that can be executed with reduced bit-widths, in which 44.3% of variable bit-width data memory instructions and 29.3% of variable bit-width computation instructions including arithmetic and logic instructions (`a.al`) and multiplication instructions (`a.mul`). Forwardk2j includes 81.8% of instructions that can be executed with reduced bit-widths, in which 70.1% of variable bit-width computation instructions and 11.7% of variable bit-width data memory instructions.

We compare the potential in terms of energy reduction on the two applications when executed with only variable bit-width computation units and when executed with both variable bit-width computation and data memory units. Figure 5 indicates that the energy reduction when executing Sobel filter with only variable bit-width for computation instructions (`a.comp`) is up to 7%. This low energy reduction is due to the fact that the application includes less computation instructions compared to other types of instructions, e.g., the branch instructions, the data memory access instructions. When the bit-width reduction is extended when accessing the data memory (`a.mem`), the energy can be reduced by up to 29% for bit-width equal to 4 bits.

Figure 6 indicates that the energy reduction when executing the forwardk2j application with only variable bit-width computation instructions (`a.comp`) is up to 13% for bit-width equal to 4 bits. The energy reduction with only `a.comp` is higher in forwardk2j than in Sobel filter because the fraction of variable bit-width computation instructions is higher in forwardk2j application than in Sobel filter. The integration of reduced bit-width memory access instructions (`a.mem`) improves the energy reduction. Figure 6 indicates that when the bit-width reduction is extended when accessing the data memory, the energy can be reduced by up to 24% for bit-width equal to 4 bits.

5 GLOBAL ENERGY MODEL

This section proposes a global energy model including both software parameters, e.g., the fraction of approximate instructions; and architecture parameters, e.g., the energy function of the bit-width, to evaluate the impact of variable bit-width units on the energy consumption of general applications. The proposed energy model allows software designers to have an overview of the global energy reduction, knowing the instruction breakdown and the energy consumption of each instruction class. Furthermore, in addition to RISC-V extensions with variable bit-width computation and data memory instructions, the computation and memory units themselves could be optimized by hardware designers to improve the energy reduction. The proposed global energy model allows to estimate the energy reduction on a given application when hardware units are optimized.

5.1 Notations

Table 1 defines the notations in our global energy model.

SW parameters	Description
N_{A_m}	Number of approx. data memory inst.
N_{A_c}	Number of approx. computation instructions
$N_A = N_{A_m} + N_{A_c}$	Number of approx. inst.
$N_{\bar{A}}$	Number of non-approx. inst.
$N = N_A + N_{\bar{A}}$	Total number of inst.
$f_{A_m} = \frac{N_{A_m}}{N_A}$	Fraction of approx. data memory instructions
$f_{A_c} = \frac{N_{A_c}}{N_A}$	Fraction of approx. computation instructions
$f_A = f_{A_c} + f_{A_m}$	Fraction of approx. inst.
$f_{\bar{A}} = 1 - f_A$	Fraction of non-approx. inst.
Archi. parameters	**Description**
o_c, o_m	Energy overhead for computation and data memory inst.
r	Ratio of non-scalable energy
e_{A_m}	Average energy of an approx. data memory inst.
e_{A_c}	Average energy of an approx. computation inst.
$e_{\bar{A}}$	Average energy of a non-approx. inst.
Global parameters	**Description**
b , B	Bit-width , maximum possible bit-width. The bit-width is configurable in $1 \leq b \leq B$
E_A	Energy consumed by all approx. inst. of an application
$E_{\bar{A}}$	Energy consumed by all non-approx. inst. of an application
E_T	Energy total of an application
α	Energy reduction of an application

Table 1: Software and architecture parameters

Let us consider a program as a set of approximate and non-approximate instructions. Approximate instructions are the approximate data memory instructions, i.e., approximate load and store (`a.ld`, `a.st`); and the approximate computation instructions (`a.al`, `a.mul`).

5.2 Energy reduction

Let $E_T(b)$ be the total energy consumed by an application executed with b bits (with $1 \leq b \leq B$).

$$E_T(b) = E_A(b) + E_{\bar{A}} \tag{1}$$

$$E_A(b) = N_{A_m} \times e_{A_m}(b) + N_{A_c} \times e_{A_c}(b) \tag{2}$$

$$E_{\bar{A}} = N_{\bar{A}} \times e_{\bar{A}} = (N - N_A) \times e_{\bar{A}} \tag{3}$$

When an application is executed with reduced bit-width (i.e., $b < B$), the energy consumption with accurate execution is reduced. The energy reduction α is estimated with Equation 4:

$$\alpha(b) = \left(1 - \frac{E_T(b)}{E_T(B)}\right) \tag{4}$$

$$\frac{E_T(b)}{E_T(B)} = \frac{N_{A_m} \times e_{A_m}(b) + N_{A_c} \times e_{A_c}(b) + (N - N_A) \times e_{\bar{A}}}{N_{A_m} \times e_{A_m}(B) + N_{A_c} \times e_{A_c}(B) + (N - N_A) \times e_{\bar{A}}} \tag{5}$$

Equation 5 expressed with fractions of approximate instructions gives:

$$\frac{E_T(b)}{E_T(B)} = \frac{f_{A_m} \times e_{A_m}(b) + f_{A_c} \times e_{A_c}(b) + \left(\frac{1}{f_A} - 1\right) \times e_{\bar{A}}}{f_{A_m} \times e_{A_m}(B) + f_{A_c} \times e_{A_c}(B) + \left(\frac{1}{f_A} - 1\right) \times e_{\bar{A}}} \tag{6}$$

Let e_{A_c} be the average energy consumed by a variable bit-width computation instruction, e_{A_m} the average energy value consumed by a variable bit-width data memory instruction.

At full-width ($b = B$), $e_{A_c}(B)$ and $e_{A_m}(B)$ are calculated with the following formulas:

$$e_{A_c}(B) = e_{\bar{A}_c} \times (1 + o_c) \tag{7}$$

$$e_{A_m}(B) = e_{\bar{A}_m} \times (1 + o_m) \tag{8}$$

where $o_c > 0$ and $o_m > 0$ are the energy overheads when implementing the variable bit-width computation and data memory instructions, respectively.

The estimation of $e_{A_c}(b)$ is described in Subsection 3.2. By looking at the data memory, we estimate the energy value $e_{A_m}(b)$ with the following formulas:

$$e_{A_m}(b) = e_{A_m}(B) \times (r + (1 - r) \times b/B), \tag{9}$$

where $r \in [0, 1]$ is non-scalable ratio from the energy consumed by a memory instruction.

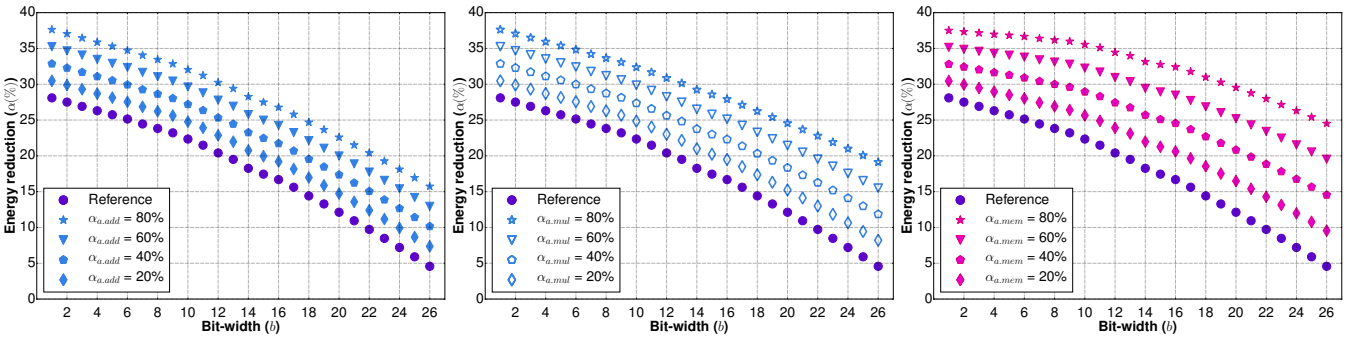
5.3 Case study: impact of hardware units and software parameters

The utilisations of the general model are twofold, as follows.

First, a hardware designer can use the model to have an overview on the impact of an optimized unit on the global energy reduction for a given application. For example, when the energy consumed by a unit, e.g., a multiplier, is reduced in half by optimizations, the proposed energy model allows estimating the global energy reduction on a given application on the complete processor.

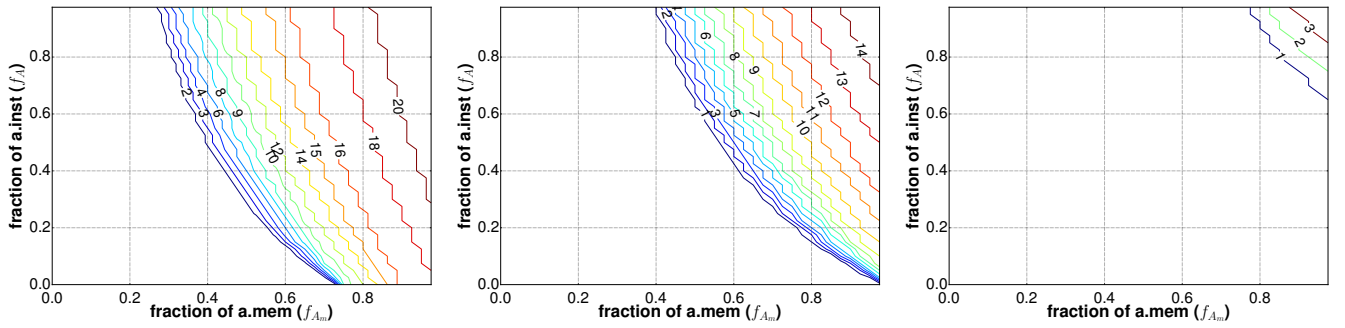
We estimate at various bit-widths the global energy reduction when one unit is optimized at a time. The global energy model in Equation 6 is instantiated with the energy values per instructions presented in Subsection 3.2 and the average fraction of approximate instructions (computation and data memory) of Sobel filter and forwardk2j, deduced from the instructions breakdown presented on Figures 3 and 4: $f_A = 0.77$, $f_{A_c} = 0.49$ and $f_{A_m} = 0.28$.

Figure 7 presents the potential in terms of energy reduction for the optimized hardware units for each bit-width. For example, Figures 7a, 7b, 7c indicate that for $b = 8$ bits, when the energy of an



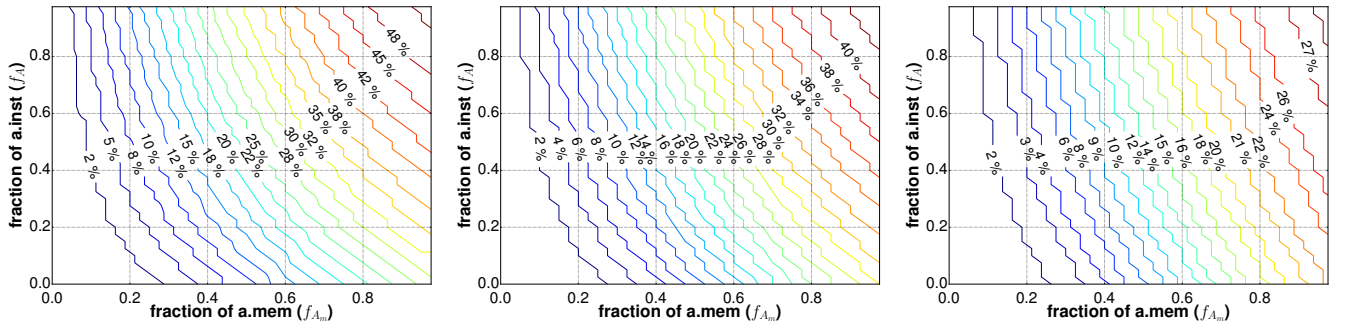
(a) Energy reduct. of optimized arith. and logic units **(b) Energy reduct. of optimized multiplication unit** **(c) Energy reduct. of optimized data memory unit**

Figure 7: Energy reduction of optimized variable bit-width units



(a) Bit-widths for an energy reduction α = 20% **(b) Bit-widths for an energy reduction α = 30%** **(c) Bit-widths for an energy reduction α = 50%**

Figure 8: Bit-widths for a given value of energy reduction



(a) Energy reduction (α) for b = 4 bits **(b) Energy reduction (α) for b = 8 bits** **(c) Energy reduction (α) for b = 16 bits**

Figure 9: Energy reduction for a given bit-width

approximate instruction is reduced by up to 60%, the global energy reduction at application level is equal to 31% for computation units and 34% for memory units. For b = 16 bits the energy reduction decreases: 24% with adders, 25% with multipliers and 29% with memory units. This investigation suggests that the optimization of memory units has more potential in terms of energy reduction than the optimization of computation units in our extended RISC-V. In our experiments, despite the fact that f_{A_c} is almost the double of f_{A_m}, the energy reduction with memory units is higher than the energy reduction in computation units.

Second, depending on the software parameters, one can use the model to either search an acceptable bit-width for computation depending on the energy budget or to estimate the energy reduction for a given accuracy. Figures 8 and 9 provide to a software designer an insight on the energy reduction, knowing the fraction of variable bit-width instructions. In a given application, for a required energy

reduction, the figures determine the adequate bit-width for computations and memory storage. Inversely, the designer knowing the required bit-width for a given accuracy, can have an estimation of energy reduction based on the fraction of approximate instructions.

Figure 8 indicates, for a given application, the fraction of approximate instructions (f_{A_c} and f_{A_m}), based on the required energy reduction and the bit-width required for an acceptable quality of result. For example Figure 8a shows that if a software designer desires a reduction of energy by 20%, and the acceptable application error recommends 8 bits, the application would require at least 35% of approximate data memory instructions, i.e., $f_{A_m} = 0.35$.

The energy reduction depends both on the bit-width and the fraction of approximate instructions, as presented in Figures 8a, 8b and 8c. Figure 8a indicates that the energy can be reduced by up to 20% with bit-width from 1 to 20 bits with at least 25% of approximate memory instructions. Figure 8b indicates that the energy can be reduced by up to 30% with bit-width from 1 to 14 bits with at least 40% of approximate memory instructions. Figure 8c shows that the reduction of the total energy consumption by 50% can be never reached with the evaluated applications on the RISC-V architecture if the user requires high accuracy. Because computing with 1, 2 or 3 bits is rarely possible in terms of accuracy.

Figure 9 presents the energy reduction, depending on the bit-width, for fractions of variable bit-width instructions f_{A_c} and f_{A_m} in $[0, 1]$. For example, with $f_A = 0.8$ and $f_{A_m} = 0.83$, the energy can be reduced by: up to 48% for $b = 4$ bits, up to 40% for $b = 8$ bits, and up to 26% for $b = 16$ bits.

5.4 Analogy to Amdahl's law

The proposed energy model is similar to the Amdahl's law. In parallel computing, the Amdahl's law indicates that the speedup in an application execution depends both on the number of processors and on the percentage of the parallel part in the algorithm. Similarly, on approximate computing, the results indicate that when energy consumption is estimated at application level, the energy reduction does not depend only on the degree of approximation, but depend also on the percentage of approximate instructions of a given program as presented in Figure 9. For energy efficiency, the aim is to be on the right upper part of Figure 9.

6 CONCLUSION

This paper extends a RISC-V processor equipped with variable bit-width arithmetic units by adding a variable bit-width memory unit. All variable bit-width units are configurable at runtime with a number of active MSBs. The impact of both variable bit-width arithmetic and memory units on the output accuracy and on the energy consumption are evaluated on two applications: Sobel filter and forwardk2j.

Moreover, we propose a generic energy model that combines hardware and software parameters to evaluate the reachable energy reduction due to reduced bit-width computation on applications. The aim is to perform an early evaluation of the impact, in terms of energy reduction, of optimizations performed by a software and/or hardware designer. A software designer, knowing the software parameters, i.e., the fraction of approximate instructions, can have an early estimation of the bit-width required to execute an application for a targeted energy reduction. The hardware designer can know in which parts of the architecture optimizations are required for more energy reduction on a given application.

To increase energy reduction in approximate computing at application level, the implementation of more approximate instructions other than the arithmetic and memory ones, e.g., instructions for loops indexing, branch control, with low impact on the output quality, could be a challenge in approximate computing.

ACKNOWLEDGEMENTS

This work was partially supported by the French Agence Nationale de la Recherche, under Grant Agreement ANR-15-CE25-0015, project ARTEFACT. Furthermore, we would like to thank our colleagues from the Design and Architecture Lab for providing us with the energy values.

REFERENCES

[1] Soheil Hashemi, R Bahar, and Sherief Reda. 2015. DRUM: A dynamic range unbiased multiplier for approximate applications. In *International Conference on Computer-Aided Design*. IEEE Press, 418–425.
[2] Andrew B Kahng and Seokhyeong Kang. 2012. Accuracy-configurable adder for approximate arithmetic designs. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. IEEE, 820–825.
[3] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. 2010. Low-power high-speed multiplier for error-tolerant application. In *International Conference of Electron Devices and Solid-State Circuits (EDSSC)*. IEEE, 1–4.
[4] Shih-Lien Lu. 2004. Speeding up processing with approximation circuits. *Computer* 37, 3 (2004), 67–73.
[5] Sparsh Mittal. 2016. A survey of techniques for approximate computing. *Computing Surveys (CSUR)* 48, 4 (2016), 62.
[6] Geneviève Ndour, Tiago Trevisan Jost, Yves Durand Anca Molnos, and Arnaud Tisserand. 2018. Evaluation of Approximate Operators - Case Study: Sobel Filter Application Executed On An Approximate RISC-V Platform. In *International Conference On Embedded Computer Systems: Architectures, Modeling And Simulation*. ACM.
[7] Daniele Jahier Pagliari, Yves Durand, David Coriat, Anca Molnos, Edith Beigne, Enrico Macii, and Massimo Poncino. 2017. A methodology for the design of dynamic accuracy operators by runtime back bias. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1165–1170.
[8] Jongse Park, Xin Zhang, Kangqi Ni, Hadi Esmaeilzadeh, and Mayur Naik. 2014. *Expax: A framework for automating approximate programming*. Technical Report. Georgia Institute of Technology.
[9] RISC-V 2018. (2018). https://riscv.org/.
[10] Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. 2011. EnerJ: Approximate data types for safe and general low-power computation. In *SIGPLAN Notices*. ACM, 164–174.
[11] R Sivakumar and D Nedumaran. 2010. Comparative study of speckle noise reduction of ultrasound b-scan images in matrix laboratory environment. *International Journal of Computer Applications* 10, 9 (2010), 46–50.
[12] Spike simulator 2018. (2018). https://github.com/riscv/riscv-isa-sim.
[13] Stefan Steinke, Markus Knauer, Lars Wehmeyer, and Peter Marwedel. 2001. An accurate and fine grain instruction-level energy model supporting software optimizations. In *Proc. of PATMOS*. Citeseer.
[14] Tiago Trevisan, Geneviève Ndour, Damien Couroussé, Christian Fabre, and Anca Molnos. 2018. ApproxRISC: An Approximate Computing Infrastructure For RISC-V. In *RISC-V Workshop, Barcelona, Spain*.
[15] Evangelos Vasilakis. 2015. An instruction level energy characterization of ARM processors. *Foundation of Research and Technology Hellas, Inst. of Computer Science, Tech. Rep. FORTH-ICS/TR-450* (2015).
[16] Ajay K Verma, Philip Brisk, and Paolo Ienne. 2008. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Proceedings of the conference on Design, automation and test in Europe*. ACM, 1250–1255.
[17] Zhou Wang, Alan C Bovik, and Hamid R Sheikh. 2005. Structural similarity based image quality assessment. *Digital Video image quality and perceptual coding* (2005), 225–241.
[18] Amir Yazdanbakhsh, Divya Mahajan, Pejman Lotfi-Kamran, and Hadi Esmaeilzadeh. 2016. *AxBench: A Benchmark Suite for Approximate Computing Across the System Stack*. Technical Report. Georgia Institute of Technology.
[19] Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. 2009. An enhanced low-power high-speed adder for error. (2009).