Extended RISC-V hardware architecture for future digital communication systems

Mael Tourres, Bertrand Le Gal, Jeremie Crenne, Philippe Coussy, Cyrille Chavet

Extended RISC-V hardware architecture for future digital communication systems

Maël TOURRES[1,2], Cyrille CHAVET[1], Bertrand LE GAL[2], Jérémie CRENNE[2] and Philippe COUSSY[1]
1 - Lab-STICC laboratory (UMR 6285), Université Bretagne Sud, France
2 - IMS laboratory (UMR 5218), Bordeaux-INP, Université de Bordeaux, France

2021 IEEE 4th 5G World Forum (5GWF) | 978-1-6654-4308-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/5GWF52925.2021.00046

***Abstract*—The fast deployment of IoT (Internet-of-Things) devices for a few years has been impressive and the progressive deployment of 5G will accelerate things even further. Indeed, this standard opens the door to a new generation of standards aiming at a convergence of networks and communication protocols (Wi-fi, LTE, 4G, etc.). These results in the need for flexible implementations of different families of codes as for instance, Turbo, LPDC and Polar codes.**

In this context, the work presented in this article proposes to design such flexible ASIP (application-specific instruction set processor) in an IoT context. The approach discussed is supported by experimental results obtained on the basis of a RISC-V architecture to which specific instruction sets have been added. Results demonstrate a reduction of the required processing clock cycles up to 47.7%, 29.8%, 16.5% and 9.7% for Polar, LDPC, NB-LDPC and Turbo (LTE) codes, respectively.

***Index Terms*—IoT devices, Forward Error Correction codes, 4G, 5G, ASIP, RISC-V.**

I. Introduction

In our widely connected world, the advent of smartphones, tablets and IoT devices over the past years has resulted in an explosive growth of data traffic over the networks never reached in previous generations. Incoming 5G communication standard [1] associated with new system paradigms such as for instance Cognitive Radio, foresees to be the foundation of a deep revolution with new communication scenarios, communications inter-operability and new applications. These new applications will emerge to take advantage of wireless connectivity (e.g. IoT, wireless broadband, Digital/Smart Factory, Machine-to-Machine…). This trend will be characterized by a convergence of communication networks, merging several physical layers and different communication standards (e.g. 4G, Wi-Fi, LTE, RFID tags…). As an example, the new 5G standard gathers several different radio signals (LTE-A, Wi-Fi/WISE…) in a flexible network. Such heterogeneous networks have to deal with several different channel codes, different throughput targets and/or different energy consumption constraints. In a nutshell, this world will rely on multiple heterogeneous radio networks, allowing dynamic adaptation in frequency, codes and/or throughputs. These evolutions are leading to switch from one generation of standard to the next, at an increasingly rapid pace. This will result in technological obsolescence issues for the devices unable to adapt themselves to their new environment. A solution for the designers is to propose flexible architectures that could handle these constraints through re-configurable devices. Moreover, since most of these devices will be embedded systems, the related constraints (i.e. reduced cost, area and power consumption) must also be considered. In this context, the channel decoding, i.e., Error Correction Code (ECC), component is a critical part of the systems, since it impacts the quality of services, power consumption and efficiency of radio resources usage. In order to master the costs, either in terms of development cost or power consumption, an adaptive architecture that could deal with some (or all) of these ECC families is extremely interesting. This architecture must also be extensible (various throughputs/latency targets) and flexible, i.e. supporting several types of error correction codes (Turbo codes, LDPC, Polar codes, …) that could be changed dynamically (i.e. on-the-fly). This paper explores, through different ECC use cases, an innovative approach to design efficient, low-power, agile and dynamically re-configurable architectures in order to reduce hardware cost and support multiple standards with reduced memory footprint for IoT devices. To achieve the ability of flexibility and high-throughput, while taking account of low-energy consumption constraints, we explore the design of an ASIP based solution on the RISC-V Instruction Set Architecture (ISA) [2]. At the end, this work proposes a flexible parallel processor architecture able to efficiently support multiple digital communication standards (e.g. 3G to 5G, and beyond). The second section is dedicated to a presentation of the state of the art. Then, in the third section the most representative error correction codes are introduced. Their associated decoding algorithms are profiled in order to extract a subset of the most computer intensive instruction patterns for each code. In the fourth section, these subsets are transcribed in new ISA instructions to enrich different RISC-V processors. In the fifth section, the results of our experiments on different RISC-V cores are presented. Finally, we conclude this paper and we draw some perspectives for this work.

II. Related Works

The perspective of potentially connecting billions of heterogeneous devices is a challenge. To date, channel decoding is one of the most critical parts of a network since it impacts both the quality of service (QoS) and the radio resources utilization. For instance, ECC typically share an share an important hardware part with the receiver. Channel decoders can reduce the required transmission power. Energy efficiency of devices and networks are thus strongly correlated with decoders performances. To address the issue of high flexibility,

978-1-6654-4308-1/21/$31.00 ©2021 IEEE
DOI 10.1109/5GWF52925.2021.00046

high throughput and low energy consumption, competing codes families are being studied: 1) Turbo codes [3] are used in the 2G, 3G and 4G standards, 2) LDPC codes [4] are employed in Ethernet, Wi-fi and Wimax, Non-Binary LDPC (LDPC-NB) [5] applications are found in the CCSDS satellite communication standard [6] and both binary LDPC and Polar codes [7] are integrated into the 5G standard. However, development of such architecture is tedious. Algorithms share very few similarities so specialized knowledge is required to integrate multiple ECC in a single device. Usual approaches implement several dedicated hardware circuits in parallel. Indeed, many works focused on efficient hardware implementation of single standard decoding circuits [8]–[10]. A new generation of communications systems entirely designed using software-defined radio (SDR) blocks have emerged [11]–[14]. The implementation feasibility of such systems has recently been demonstrated but only on non-embedded devices, i.e., in base stations with multi-cores central processing units (CPUs) and/or graphical processing untis (GPUs). Previous works report efficient implementations in terms of throughput and latency on SDR target systems for Turbo codes [15], LDPC codes [16] and Polar codes [17], [18]. Although more flexible and scalable than application-specific integrated circuit (ASIC)-based designs, industrial systems have limitations. Energy consumption and thermal dissipation [19] of these architectures strongly exceed embedded IoT typical constraints. A number of academic [20]–[22] and industrial proposals [22] exist that make the design of optimized hardware architectures possible. Based on application-specific instruction set processors (ASIP), these implementations provide the flexibility needed for the ECC decoding process [23]. However, their benefits are limited as they are designed considering a worst-case design scenario: they are generally larger and consume more energy. Specific and costly development tools and methodologies [24] also have to be integrated as part of designs trade-offs. To design systems tailored to IoT constraints i.e. maximizing computational performances *vs.* minimizing area costs and energy, these systems have less processor cores and reduced frequency. Customizing the processor instruction set architecture (ISA) to match with targeted codes is *de facto* a relevant option. Diverse applications can benefit from this technique. Typical examples include video processing and artificial intelligence [25], [26]. As part of the effort, some works propose the design of processor architectures with an ECC-oriented ISA. Two main contributions have been reported:

1) In [22] and [21], authors have focused on the design of efficient programmable architectures for Turbo codes and LPDC codes, respectively. A number of application parameters allows to manage intra and inter-standards (frame size, code structure, speed).
2) In [27] and [28], authors present programmable architectures to support several and distinct ECC families. They exploit multi-standards applications scenarios. In [28], authors describe a low-cost architecture based on a butterfly network. However this proposal is only focused on memory and network optimization, the design of the processor is then out of the scope of this work.

III. ECC DECODING ALGORITHMS

Error-correcting codes are well-known for their high computational complexity and irregular memory accesses [29], [30]. The improvement of both efficiency and flexibility have been the subject of a continuous effort for decades. Research works have demonstrated that efficient hardware implementations and integration of ECC decoder are possible. For most ECC families, low complexity implementations are achieved with 8-bit fixed-point number representation and arithmetic. It is demonstrated in [31]–[37] that this fixed-point data format only slightly impacts on processing results. In this work, we consider four families of ECC:

- **Turbo codes** - Turbo codes [3] are found in 2G, 3G and 4G LTE standards and may be candidates for the future 6G [38]. Their decoding algorithm consists of tracking/routing two mesh networks which are connected to a data interleaver [15], [39]. Fixed data widths of 8 or 16 bits are used. Considered algorithm is a max-log-MAP [40]. Involved arithmetic operations are *additions*, *subtractions*, *multiplications*, *minimum searching/computations* and *maximum*.
- **LDPC codes** - Low Density Parity Check (LDPC) codes are commonly found in ground radio frequency (RF) standards (Wi-fi, Wimax and 5G). Their decoding algorithms can be summarized as a nest of loops where data and information are computed and shared across a network. This network is implemented using a bipartite graph topology [41], [42]. All computations manipulate 8-bit integer data. Considered algorithm is a Horizontal-Layered Min-Sum (HL, MS) [15], [16]. Involved arithmetic operations are *addition*, *saturated subtractions*, *minimum searching*, *masking* and *bit-wise operations/selections*.
- **Polar codes** - Polar codes are a recent family of ECC [7]. These codes have been integrated into the 5G standard to provide protection of control frames. Polar codes decoding are performed by using a recursive path over a binary tree [7], [17], [43]. Data accesses are mostly regular. However, the computing parallelism is reduced while the decoding tree is being traversed. Considered algorithm is a Successive-Cancellation (SC) [17]. Our version of the SC-List algorithm is not fully implemented yet. Involved logical and arithmetic operations in the decoding process are *additions*, *saturated subtractions*, *minimum searching* and *masking*.
- **Non-binary LDPC codes** - Instead of conventional binary coding, non-binary LDPC codes are a transposition of LDPC codes to the Galois Field i.e. $GF(n)$ with $n > 2$. This extension provides a significant improvement in the decoding process but leads to an increase in computation complexity. Their decoding performance motivated their usage into the CCSDS satellite communication standard [6]. Decoding algorithms are designed with nested loops. It brings regularity in memory accesses by taking

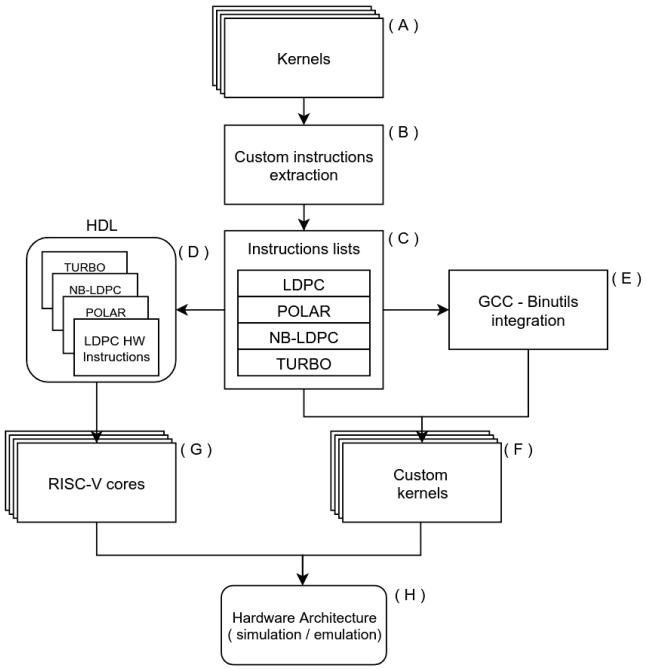

Fig. 1. Proposed workflow for identification, extraction and integration of specific instructions in processor cores. (HDL : Hardware Description Language)

advantage of GF representation. Considered algorithm is a Min-Sum (MS) [44]. Involved operations are *additions*, *saturated subtractions*, *minimum searching* and *XOR*.

Optimized software descriptions of the selected ECC decoding algorithms have been entirely designed in this work. Leveraging previous works [15], [17], [41], [45], we target a general-purpose processor integration. All algorithmic descriptions were subject to extensive analysis with hand-crafted optimizations to avoid complex operations and conditional branches. They have been validated through the simulation of BER/FER performances. Similarly to the hardware decoders found in the literature (with the exception of Turbo codes), they handle 8-bit arithmetic data representation and manipulation. From these optimized codes, programs executions and patterns detection have been performed to extract and translate the most suitable instructions. They were implemented in hardware as dedicated processor instructions afterward. The set of extracted patterns and resulting instructions are shown in Table I. In this table, the mnemonics of the proposed instructions and their algorithmic descriptions are listed.

IV. Detailed workflow for RISC-V cores

In the previous section, the main operations performed in the different ECC families have been introduced. With the observation of very similar instructions, our fine-grained code analysis allowed us to extract and highlight common points between the studied families. Consequently, the idea of sharing such common parts in dedicated instructions is confirmed.

TABLE I
List of instruction mnemonic and associated implementation (underlined : not in RISCY's XpulpV2 ISA). Turbo not integrated in RISCY. (rD : destination register, rS : source registers)

	Mnemonic/instructions	Implementation
LDPC	ld.subsat(rD, rS1, rS2)	Res = rS1 – rS2 Res = (Res >127)? 127 : (Res <-127)? -127 : Res rD = Res
	ld.abs(rD, rS1)	rD = (rS1 >= 0)? rS1 : - rS1
	ld.max (rD, rS1, rS2)	rD = (rS1 >rS2) ? rS1 : rS2
	ld.min (rD, rS1, rS2)	rD = (rS1 >rS2) ? rS2 : rS1
	ld.nMess (rD, rS1, rS2)	rD = (rS1 == 1) ? rS2 : - rS2
Polar	pl.f (rD, rS1, rS2)	min1 = abs(rS1) min2 = abs(rS2) min1 = min(min1 , min2) sign = (rS1 <0) ^ (rS2 <0) rD = (sign == 0)? min1 : -min1
	pl.r (rD, rS1, rS2)	rD = (rS2) ? 0 : (rS1 <0) ? 1 : 0
	pl.addsat (rD, rS1, rS2)	Res = rS1 + rS2 Res = (Res >127)? 127 : (Res <-127)? -127 : Res rD = Res
	pl.subsat (rD, rS1, rS2)	Res = rS1 - rS2 Res = (Res >127)? 127 : (Res <-127)? -127 : Res rD = Res
LDPC-NB	ldn.add32u_sat64(rD, rS1, rS2)	res = rS1 + rS2 rD = (res >64) ? 64 : res
	ldn.min (rD, rS1, rS2)	same as ld.min
	ldn.add32s_sat64(rD, rS1, rS2)	res = rS1 + rS2 res = (sum >64) ? 64 : (sum <- 64) ? -64 : res rD = Res
	ldn.sub32s_sat64(rD, rS1, rS2)	res = rS1 - rS2 res = (sum >64) ? 64 : (sum <- 64) ? -64 : res rD = Res
Turbo Code	tb.subsat (rD, rS1 ,rS2)	same as pl.subsat
	tb.addsat (rD, rS1 ,rS2)	same as pl.addsat
	tb.max (rD, rS1, rS2)	same as ld.max
	tb.scale(rD, rS1, rS2)	sign = rs1 >0 A = abs(rs1) B = A >>2 C = A - B rD = (sign==0)? -C :C

A. Detailed design flow

In order to prototype and validate the set of selected instructions, the design approach presented in Fig. 1 is used. It should be noticed that each ECC kernel has been tailored to expose repetitive patterns of instructions, that can be extracted as one instruction (as previously described). First and foremost, lists of ECC-dependent instructions have been specified based on the following requirements (Figure 1-B,C):

- Instructions must fit within the standardized ISA's instruction specification (*i.e* RISC-V's R-type [2]), with operations having a maximum of two inputs and one output. This constraint ease providing an homogeneous support for a large variety of processors cores.
- The instructions are one cycle only, do not introduce any additional latency and avoid pipeline stalls.

The selected instructions will be synthesized in hardware to

be integrated in the different targeted cores, implementing the necessary arithmetic and logic operations in Register Transfer Level (RTL) as shown by D in Fig. 1. Then, a flexible and adaptable processor core is selected to allow us to enrich its ISA with our new RTL instructions. The RISC-V ISA family is among the most suited for architectural research, and benefits from a rich eco-system that helps time-consuming designs steps. Indeed, a lot of contributions helped to demonstrate the relative maturity of this novel ISA in the field of generic and optimized processor designs for IoT-based applications [46].

As previously explained, the proposed custom instructions are described at the RTL level. These hardware resources (used to describe the set of defined customs instructions) are implemented into RISC-V based processors (Figure 1-G). In parallel, to take advantage of these newly integrated instructions, the compilation workflow is updated accordingly (i.e. GCC 10.2[1] with modifications to *binutils* (Figure 1-E)). Finally, software ECC kernels are updated (Figure 1-F) to accommodate the customs set of instructions.

B. Targeted RISC-V cores

Regarding the architectural choice of RISC-V cores, many open-source cores are available. Each one offers different architectural features such as number of pipeline levels, ALU complexity, availability and size of memory caches... Moreover, some of them provide out-of-order capabilities or several parallelism degrees such as DLP (Data Level Parallelism), TLP (Thread Level Parallelism) and ILP (Instruction Level Parallelism). Although each core is theoretically extensible and customizable enough to accommodate custom dedicated instructions, their relative architectural complexity may hinder this operation. In this paper, four cores have been selected out of the multiple carefully evaluated RISC-V processors. This selection was based on their different architectural characteristics, general performances and relative size. These four cores are used to measure the gains resulting from our proposed approach. They all support the RV32-IMC instruction set (I-Integer, M-Multiplier&Divider, C-Compressed) and one to several pipeline stages. The RISCY core also benefits from a dedicated ISA extension namely *XpulpV2* dedicated to support signal processing features. To guarantee the homogeneity of the tests, the instruction set is restricted to the RV32I specification of the RISC-V ISA and the various cores are strictly set up (if possible) in their standard configuration.

- **PicoRV32** [47] - A low-complexity core mainly designed for simple applications. It is a very fast RISC-V core implementing a single *in-order* pipeline architecture.
- **IBEX** [46], [48] - Initially known as micro-RISCY, this RISC-V core is more balanced (performance vs complexity) architecture. It implements a 2-stage *in-order* pipeline architecture in its basic version (small).
- **SCR1** [49] - This last evaluated core presents good performances for a reasonably low hardware complexity and a controlled energy consumption. It is implemented with an architecture integrating 3-stages of *in-order* pipelines in its basic (*base*) configuration.
- **RISCY** [50] - A well-designed and more capable RISC-V core implementing a set of customs instructions targeting regular arithmetic optimizations and low power DSP (*XpulpV2*). It is a 4-stage single-issue *in order* pipeline architecture, developed as a part of the PULP project [51].

[1]GCC 7.2 for the RISCY core tool-chain

These four RISC-V compliant cores (have been selected to) allow a fair evaluation of our proposed ISA extension. Indeed, their architectural characteristics will enable the evaluation of benefits and costs in different application contexts. The obtained results are presented and analyzed in the next section.

V. Experimental Results

In this section, we evaluate the impacts of our proposal on the decoding speed/latency and throughput performances of the selected cores. These are measured via the execution of the ECC presented in section 3. Results are put in perspective with the hardware complexity increase (on FPGA target) induced by the ISA enrichment of the selected RISC-V cores. First, the initial software descriptions of the decoding algorithms have been executed on each core, to obtain a reference in terms of execution time. Then, the modified kernels (i.e. updated with calls to our custom instructions) have been executed to measure their impacts on the performances.

Timing/temporal performances -The results obtained during our experimentation are presented in a synthetic way in the Table II. This table presents for each core the number of clock cycles for the *base* codes (i.e. without the custom instructions) and the number of clock cycles gained with the *modified* codes (i.e. with the custom instructions). It can be noticed that the proposed approach allows us to almost systematically optimize the performances, however we also observe that these gains are not homogeneous over all the configurations. For example, as shown for the LDPC code, performance gains can range from 26% for the IBEX core, down to 3.2% on the RISCY core. This volatility is explained by different compilation flows (and software libraries) for each core. It should be noted that RISCY has a tool-chain entirely optimized for control applications. This explains why for complex codes (e.g. LDPC-NB) we see a noticeable degradation in the number of processing clock cycles with our Instruction Set Extension (ISE). Previously selected instructions were found in the *XpulpV2* Instruction Set Extension of the RISCY core (see Table II). Thus, to avoid duplication, some of our instructions were not integrated (*ld.subsat(), pl.f(),* ...).

Impact on hardware cost - In Table III, the impact of the instruction set extension on the hardware complexity of the cores is provided. Processors have been implemented on an FPGA device (XC7K325FFG900-2) by using Xilinx Vivado 2018.2. The results presented in this section have been obtained after the placement and routing steps. Original versions of the processors and the enriched versions have been implemented in order to measure the direct impact (of the new instructions), and also the induced impact (instruction

TABLE II
REQUIRED CLOCK CYCLES PER ECC DECODING KERNEL EXECUTION DEPENDING ON RISC-V CORE. (NUMBERS SPACED FOR CLARITY. MOD. IS SHORT FOR MODIFIED. MINUS SIGN REPRESENT THE NUMBER OF RETIRED CLOCK CYCLES)

		LDPC (int8)	POLAR (int8)	LDPC NB (uchar)	TURBO (int)
PicoRV	Base	219 543	1 331 700	3 480 560	2 568 651
	Mod. (Δ)	-30 924 (**-11.3%**)	-574 685 (**-47.7%**)	-351 359 (**-7.1%**)	-67 706 (**-2.6%**)
IBEX	Base	83 983	516 817	1 408 610	1 029 657
	Mod. (Δ)	-25 054 (**-29.8%**)	-213 570 (**-41.3%**)	-233 685 (**-16.5%**)	-100 122 (**-9.7%**)
SCR1	Base	76 629	418 572	1 312 194	937 707
	Mod. (Δ)	-19 071 (**-24.9%**)	-137 810 (**-32.9%**)	-145 112 (**-11%**)	-42 709 (**-4.5%**)
RISCY	Base	50 533	280 923	791 437	n.a.
	Mod. (Δ)	-2 635 (**-5.2%**)	-99 245 (**-35.3%**)	+139 358 (**+17.6%**)	n.a.

decoders, multiplexing in the ALU,...). The results obtained are reported in the Table III. These results only display the hardware overhead measured for each instruction set and core. As with the previous table, we also show the percentage increase *vs.* the cost of the architecture without our custom instructions. First, we observe a difference in the cost of integrating the instructions into the processor cores. Indeed, the data presented in Table III demonstrate that this cost strongly depends on the considered RISC-V core and its associated design flow. Thus, the integration of instructions specific to LDPC codes increases the cost of the IBEX (small) architecture by 120 ($+4,9\%$) LUTs (Look-Up Table) , while on a PicoRV32 core it increases by more than 250 ($+25\%$) LUTs. These disparities coming from the seemingly same RTL descriptions (the PicoRV32 core is described in Verilog, others cores in System-Verilog) show a strong adhesion between the considered instruction set and the target core, without a clear trend. This is linked to the fact that each core is generated on different synthesis chains, making a fair analysis of the results complex. However, we can suppose that probably during the logical optimization stage, the synthesizer likely merges or duplicates some resources according to design constraints that still need to be explored. The second observation is related to the hardware cost of the extension of the cores. Apart from the experiments carried out on the PicoRV32 core, and for which the overhead is 20% to 25%, the overhead of the instructions remains limited. When compared to the improvements in execution times, it demonstrates the interest of the proposed approach and selected instructions sets selected. The impact of these instructions strongly depends on the experience of the designer to optimize the quality of integration on the targeted core.

VI. CONCLUSION

Digital embedded systems rely upon reliable, flexible, scalable and energy-efficient communications. Software solutions are currently being studied, as they avoid the pitfalls in terms of flexibility and scalability of ASIC/FPGA solutions. However, their overall high power consumption incurs a severe drawback to match the energy requirements of low energy footprint of embedded systems. In this article, the followed approach firstly aims at evaluating the performances (in terms of throughput and latency) of the main families of modern ECC codes targeting well-known embedded RISC-V processor cores. In order to improve the adequacy between these custom-tuned algorithms and the embedded processor architectures, an extension of their standard instruction set with carefully designed custom instructions is proposed. Achieving a fully functional architectural extension of the selected cores, we demonstrated an almost systematic performance increase for all considered algorithms. Indeed, with the addition of a small number of scalar instructions, the decoding performances of ECC algorithms are significantly improved when focusing on RISC-V low-power class cores. Results demonstrate a reduction of the required processing clock cycles up to 47.7%, 29.8%, 16.5% and 9.7% for Polar, LDPC, NB-LDPC and Turbo (LTE) codes, respectively. ECC's decoding algorithms are highly sensitive to concurrent calculations and exhibit dramatic speed increase when exploited on SIMD-based solutions. To obtain further optimization, the next step is to explore the integration of parallelization mechanisms through the use of SIMD instructions (such as in related works focusing on high-end multi-core or many-core targets). Finally, an (in-depth) evaluation and optimization of energy consumption will also be taken into account.

TABLE III
ADDITIONAL HARDWARE COSTS RESULTING FROM INTEGRATION OF DEDICATED INSTRUCTIONS IN LUT(LOOK-UP TABLE).

	BASE	LDPC	LDPC-NB	POLAR	TURBO
PicoRV32 (1 pipe)	1010	+250 (+25%)	+203 (+20%)	+122 (+12.8)	+295 (+23.5%)
IBEX (2 pipes)	2446	+120 (+4.9%)	+255 (+10.4%)	+138 (+5.6%)	+248 (+10.1%)
SCR1 (3 pipes)	3984	+193 (+4.8%)	+275 (+6.9%)	+140 (+3.5%)	+312 (+7.8%)
RISCY (4 pipes)	11156	+165 (+1.5%)	+300 (+2.7%)	+189 (+1.7%)	n.a.

REFERENCES

[1] S. Baek, D. Kim, M. Tesanovic, and A. Agiwal, "3GPP new radio release 16: Evolution of 5G for industrial internet of things," *IEEE Communications Magazine*, vol. 59, no. 1, pp. 41–47, 2021.

[2] RISC-V technical specification, *https://riscv.org/technical/specifications*, April 2021.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[4] R. Gallager, "Low-density parity-check codes," 1962.

[5] C. Poulliat, M. Fossorier, and D. Declercq, "Design of non binary LDPC codes using their binary image: algebraic properties," in *Proceedings of the IEEE International Symposium on Information Theory (ISTC)*, Seattle, WA, USA, July 2006, pp. 93–97.

[6] Consultative Committee for Space Data Systems (CCSDS), *CCSDS 231.1-O-1 - ORANGE BOOK - SHORT BLOCK LENGTH LDPC CODES FOR TC SYNCHRONIZATION AND CHANNEL CODING*, Washington, DC, USA, April 2015.

[7] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, July 2009.

[8] S. Weithoffer, C. A. Nour, N. Wehn, C. Douillard, and C. Berrou, "25 years of turbo codes: From Mb/s to beyond 100 Gb/s," in *Proceedings of the IEEE International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, Hong Kong, China, December 2018, pp. 1–6.

[9] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A survey of FPGA-based LDPC decoders," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1098–1122, December 2016.

[10] P. Giard, G. Sarkis, A. Balatsoukas-Stimming, Y. Fan, C.-y. Tsui, A. Burg, C. Thibeault, and W. J. Gross, "Hardware decoders for polar codes: An overview," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, Montreal, QC, Canada, August 2016, pp. 149–152.

[11] D. Wubben, P. Rost, J. S. Bartelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy, and G. Fettweis, "Benefits and impact of cloud computing on 5G signal processing: Flexible centralization through Cloud-RAN," *IEEE Signal Processing Magazine*, vol. 31, no. 6, November 2014.

[12] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for mobile networks - a technology overview," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[13] Intel corp., *INTEL 5G Vision (Network, Cloud Client)*, February 2017.

[14] Fujitsu Inc., "The benefits of Cloud-RAN architecture in mobile network expansion," Fujitsu Network Communications Inc., Tech. Rep., 2015.

[15] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo-decoders on multi-core architectures," *Annals of Telecommunications, Springer*, vol. 75, pp. 27–42, February 2020.

[16] ——, "Low-latency software LDPC decoders," in *Proceedings of the IEEE International Workshop on Signal Processing Systems (SIPS)*, Lorient, France, October 2017.

[17] B. Le Gal, C. Leroux, and C. Jego, "Multi-Gb/s software decoding of polar codes," *IEEE Transactions on Signal Processing (TSP)*, vol. 63, no. 2, pp. 349–359, January 2015.

[18] M. Léonardon, A. Cassagne, C. Leroux, C. Jégo, L.-P. Hamelin, and Y. Savaria, "Fast and flexible software polar list decoders," *Journal of Signal Processing Systems*, vol. 91, no. 8, pp. 937–952, August 2019. [Online]. Available: https://doi.org/10.1007/s11265-018-1430-3

[19] V. Pignoly, B. Le Gal, C. Jégo, B. Gadat, and L. Barthe, "Fair comparison of hardware and software LDPC decoder implementations for SDR space links," in *Proceedings of the IEEE International Conference on Electronics Circuits and Systems Conference (ICECS)*, November 2020.

[20] T. V. Aa, M. Palkovic, M. Hartmann, P. Raghavan, A. Dejonghe, and L. Van der Perre, "A multi-threaded coarse-grained array processor for wireless baseband," in *Proceedings of the IEEE Symposium on Application Specific Processors (SASP)*, June 2011, pp. 102–107.

[21] M. Alles, T. Vogt, and N. Wehn, "Flexichap: A reconfigurable asip for convolutional, turbo, and ldpc code decoding," in *Proceedings of the International Symposium on Turbo Codes and Related Topics (ISTC)*, Lausanne, Switzerland, September 2008, pp. 84–89.

[22] P. Murugappa, A. Baghdadi, and M. Jézéquel, "Parameterized area-efficient multi-standard turbo decoder," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, March 2013, pp. 109–114.

[23] C. Chavet and P. Coussy, *Advanced Hardware Design for Error Correcting Codes*. Springer, 2014.

[24] M. Léonardon, C. Leroux, P. Jääskeläinen, C. Jégo, and Y. Savaria, "Transport triggered polar decoders," in *Proceedings of the IEEE International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, vol. 2018, Hong Kong, China, December 2018, pp. 1–5.

[25] S. Payvar, M. Khan, R. Stahl, D. Mueller-Gritschneder, and J. Boutellier, "Neural network-based vehicle image classification for IoT devices," in *Proceedings of the IEEE International Workshop on Signal Processing Systems (SiPS)*, Nanjing, China, October 2019, pp. 148–153.

[26] R. Porter, S. Morgan, and M. Biglari-Abhari, "Extending a soft-core RISC-V processor to accelerate CNN inference," in *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)*, December 2019, pp. 694–697.

[27] S. U. Reehman, C. Chavet, P. Coussy, and A. Sani, "In-place memory mapping approach for optimized parallel hardware interleaver architectures," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, March 2015.

[28] A. Sani, P. Coussy, and C. Chavet, "A dynamically reconfigurable ECC decoder architecture," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, March 2016, pp. 1437–1440.

[29] C. Chavet, F. Lozachmeur, T. Barguil, A. S. Hussein, and P. Coussy, "Solving memory access conflicts in LTE-4G standard," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 1518–1522.

[30] H. Harb and C. Chavet, "Fully parallel circular-shift rotation network for communication standards," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3412–3416, May 2020.

[31] H. Michel and N. Wehn, "Turbo-decoder quantization for UMTS," *IEEE Communications Letters*, vol. 5, no. 2, pp. 55–57, February 2001.

[32] A. Singh, E. Boutillon, and G. Masera, "Bit-width optimization of extrinsic information in turbo decoder," in *Proceedings of the International Symposium on Turbo Codes and Related Topics (ISTC)*, Lausanne, Switzerland, September 2008, pp. 134–138.

[33] T. T. Nguyen Ly, "Efficient Hardware Implementations of LDPC Decoders, through Exploiting Impreciseness in Message-Passing Decoding Algorithms," Ph.D. dissertation, Université de Cergy Pontoise, 2017.

[34] M. Meidlinger, A. Balatsoukas-Stimming, A. Burg, and G. Matz, "Quantized message passing for LDPC codes," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, November 2015, pp. 1606–1610.

[35] Z. Shi and K. Niu, "On uniform quantization for successive cancellation decoder of polar codes," in *Proceedings of the IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Washington, DC, USA, September 2014, pp. 545–549.

[36] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, October 2015.

[37] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, May 2010.

[38] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 1833–1844, May 2018.

[39] A. Cassagne, T. Tonnellier, C. Leroux, B. Le Gal, O. Aumage, and D. Barthou, "Beyond Gbps turbo decoder on multi-core CPUs," in *Proceedings of the International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, September 2016, pp. 136–140.

[40] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo decoders on multi-core architectures," *Annals of Telecommunications, Springer*, Sepember 2019.

[41] B. Le Gal and C. Jego, "High-throughput LDPC decoder on low-power embedded processors," *IEEE Communication Letters*, vol. 19, no. 11, pp. 1861–1864, November 2015.

[42] J. Andrade, G. Falcao, V. Silva, and L. Sousa, "A survey on programmable LDPC decoders," *IEEE Access*, vol. 4, July 2016.

[43] X. Zhou, Y. Shen, X. Tan, X. You, Z. Zhang, and C. Zhang, "An Adjustable Hybrid SC-BP Polar Decoder," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Chengdu, China, 2018, pp. 211–214.

[44] E. Boutillon, L. Conde-Canencia, and A. Al Ghouwayel, "Design of a GF(64)-LDPC Decoder Based on the EMS Algorithm," *IEEE Transactions on Circuits and Systems Part 1 Fundamental Theory and Applications*, vol. 60, no. 10, pp. 2644–2656, 2013.

[45] B. Le Gal and C. Jego, "High-throughput FFT-SPA decoder implementation for non-binary LDPC codes on x86 multicore processors," *Journal of Signal Processing Systems*, vol. 92, pp. 37–53, March 2020.

[46] D. Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *Proceedings of the International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Thessaloniki, Greece, September 2017, pp. 1–8.

[47] PicoRV32 RISC-V Core, *https://github.com/cliffordwolf/picorv32*, March 2021.

[48] Ibex RISC-V Core, *https://github.com/lowRISC/ibex*, March 2021.

[49] SCR1 RISC-V Core, *https://github.com/syntacore/scr1*, March 2021.

[50] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "Near-threshold RISC-V core with DSP extensions for scalable iot endpoint devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, February 2017.

[51] RISCY RISC-V Core, *https://github.com/pulp-platform/pulpissimo*, April 2021.