NISC-Based Soft-Input–Soft-Output Demapper

Mostafa Rizk, *Student Member, IEEE*, Amer Baghdadi, *Member, IEEE*, Michel Jézéquel, *Senior Member, IEEE*, Yasser Mohanna, and Youssef Atat

***Abstract*—Applications in wireless digital communication field are becoming increasingly complex and diverse. Circuits and systems adopted in this application domain must not only consider performance and implementation constraints but also the requirement of flexibility. The combination of flexibility and the ever increasing performance requirements demands design approach that provides better ways of controlling and managing hardware resources. An application-specific instruction-set processor (ASIP) design approach is a key trend in designing flexible architectures. The ASIP concept implies dynamic scheduling of a set of instructions that generally leads to an overhead related to instruction decoding. The no-instruction-set-computer (NISC) concept has been introduced to reduce this overhead through the adoption of static scheduling. In this brief, the NISC approach is explored through a case-study design of universal demapper for multiple wireless standards. The proposed design has common main architectural choices as a state-of-the-art ASIP for comparison purpose. The obtained results illustrate a significant improvement in execution time and implementation area while using identical computational resources and supporting same flexibility parameters.**

***Index Terms*—Demapper, flexibility, iterative processing, multi-standard wireless system, no-instruction-set-computer (NISC).**

I. Introduction

CURRENTLY, flexibility is a major design requirement of embedded systems and circuits. Hardware architectures are supposed to accommodate multitude system configurations as well as their corresponding algorithmic variants. Because of the rapid evolution of related standards, modern wireless digital communication systems are highly concerned about the flexibility feature. However, the emergent flexibility need should not come at the cost of performance and implementation requirements. Application-specific processors are increasingly adopted to implement definite blocks of wireless system since they provide a good solution in designing flexible architectures that can fulfill nowadays requirements in terms of low error-rate performance and high throughput, and satisfy the tight constraints on implementation area and power consumption.

The application-specific instruction-set processor (ASIP) concept offers a tradeoff in terms of the flexibility of general-purpose processors and the efficiency of application-specific integrated circuit (ASIC) by customizing the functionality and the data path structure through a custom instruction set [1]. This tradeoff can be tuned in a language-based ASIP design approach, when the degree of flexibility is limited, to reach an implementation efficiency comparable to parameterized architectures using classical register-transfer level (RTL) design approach [2]. However, when hardware is dedicated for a specific application, processes of specifying and describing instructions at the designing phase and decoding them at runtime form an overhead in terms of productivity, execution performance, and implementation costs. The no-instruction-set-computer (NISC) concept [3] adopts static scheduling of operations instead of dynamic scheduling (i.e., decoding instructions at runtime to determine which operations to execute) to simplify the ASIP approach. Design productivity is increased by obviating the task of finding and designing a custom instruction set. The design quality is better ensured by reducing design complexity to match exactly the requirements of the desired application.

In a previous work, presented in [4], the NISC concept has been explored to realize flexible turbo equalizer. The comparison with a similar ASIP implementation, which uses identical computational resources and supports the same flexibility parameters, illustrates significant improvement in throughput with reduced implementation costs. However, additional memory locations are required to implement the control memory with respect to the ASIP program memory. In fact, this is directly related to the considered application and devised architecture choices. the NISC concept is evaluated in this brief through a different application design and architecture choices. This brief presents a novel NISC-based universal demapper. The proposed architecture is thoroughly described. In addition, the design is compared with a state-of-the-art ASIP-based equivalent implementation in terms of performance, throughput, and area of implementation. Recent emergent wireless communication standards, support various modes and configurations related to the characteristics of the target constellation such as modulation type and mapping style. Different order constellations have been employed starting from binary phase-shift keying (BPSK) up to 256-ary quadrature amplitude modulation (QAM).

Constellations with Gray mapping are adopted to achieve the lowest possible bit-error probability [5]. Furthermore, the independence between the in-phase (I) and the quadrature (Q) components of a symbol in Gray-mapped constellations can be exploited to reduce the computational complexity without suffering any performance loss. Accordingly, numerous simplifications have been proposed for specific constellations [6]. These simplifications can not be applied when incorporating signal space diversity (SSD) with rotated constellation introduced in DVB-T2 standard since the independence between I and Q components is broken.

On the other hand, an iterative receiver can significantly improve the performance compared with the noniterative receiver [7]. In iterative schemes, *a priori* information, which is generated by the decoder, is involved in demapping and imposes additional complexity.

Manuscript received March 1, 2015; revised April 23, 2015; accepted June 12, 2015. Date of publication July 13, 2015; date of current version October 30, 2015. This brief was recommended by Associate Editor G. Masera.

M. Rizk, A. Baghdadi, and M. Jézéquel are with the Department of Electronics, Telecom Bretagne, Le Centre national de la recherche scientifique (CNRS) unites mixtes de recherche (UMR) 6285 Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC), 29238 Brest, France (e-mail: mostafa.rizk@telecom-bretagne.eu; amer.baghdadi@telecom-bretagne.eu; michel.jezequel@telecom-bretagne.eu).

Y. Mohanna and Y. Atat are with the Faculty of Sciences, Lebanese University, Beirut, Lebanon (e-mail: yamoha@ul.edu.lb; youssef.atat@ul.edu.lb).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2015.2455991

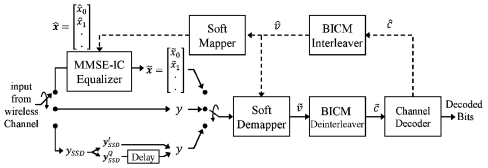

Fig. 1. Iterative receiver block diagram.

Most demapper implementations reported in the literature are of limited flexibility. In [8], the proposed soft-decision demapper architecture supports only four modulation schemes as specified in the DVB-S2 standard. Other demapper design has been presented in [9] for rotated QAM constellations targeting the DVB-T2 standard. To the best of our knowledge, only one universal demapper has been introduced in [2]. The demapper architecture is ASIP based and covers all flexibility requirements for recent wireless standards. Such wide flexibility to support different mapping styles, modulation schemes, SSD with rotated constellation, iterative and noniterative processing schemes becomes crucial in the current trend toward the convergence of wireless communication services [10] and the requirement of multistandard terminals. Furthermore, demonstrating the ability of designing highly flexible, yet efficient, demapping architectures can trigger the proposal of new modulation schemes and parameters that better suit the application and environment conditions. Such new schemes, associated with efficient flexible implementations, can then constitute potential candidates for adoption in next-generation communication systems.

II. System Model and Algorithm

Fig. 1 shows the block diagram of iterative receiver. Depending on the transmitter configuration and propagation conditions, the input from the wireless channel can be either directly delivered to the demapper or passed through a channel equalizer. To reduce the computational complexity, the demapper works in logarithmic domain and generates probabilities $\tilde{v}$ on received sequence in the form of log-likelihood ratios (LLRs), where v represents the binary mapping of the transmitted sequence. These LLRs construct, after deinterleaving, the input $\tilde{c}$ to channel the decoder. Through the feedback loop, the *a posteriori* information output from the decoder is interleaved and then fed back to the demapper and the equalizer. In this brief, the channel fading has a Rayleigh distribution with additive white Gaussian noise (AWGN). To compute the LLRs, the following expression is used [11]:

$$L\left(\tilde{v}_t^i\right) = \ln\left[\frac{\sum_{x\in\mathcal{X}_1^i}\left(e^{-\frac{1}{2\sigma^2}|y_t-\rho_t\cdot x|^2}\cdot\prod_{\substack{l=0\\ l\neq i}}^{m-1}P\left(\hat{v}_t^l\right)\right)}{\sum_{x\in\mathcal{X}_0^i}\left(e^{-\frac{1}{2\sigma^2}|y_t-\rho_t\cdot x|^2}\cdot\prod_{\substack{l=0\\ l\neq i}}^{m-1}P\left(\hat{v}_t^l\right)\right)}\right] \quad (1)$$

where m is the number of bits per symbol, $i = 0, 1, \ldots, m-1$, $L(\tilde{v}_t^i)$ is the LLR of the ith bit of transmitted symbol at time t, $\mathcal{X}_0^i$ and $\mathcal{X}_1^i$ are the symbol sets of constellation for which symbols have their ith bit equals $b \in \{0, 1\}$, ρ_t is the channel fading coefficient and σ^2 is the AWGN variance, and $P(\hat{v}_t^l)$ is the probability of lth bit of symbol x computed through *a priori* information. To reduce the complexity, max-log approximation [12] is applied by using the following formulas:

$$\ln\frac{a}{b} = \ln(a) - \ln(b)$$

$$\ln(e^{\delta_1+\cdots+e^{\delta_n}}) \approx \max_{i\in\{1,\ldots,n\}}\delta_i \max(a) - \max(b)$$

$$= \min(-b) - \min(-a).$$

The expression in (1) becomes

$$L\left(\tilde{v}_t^i\right) \approx \min_{x\in\mathcal{X}_0^i}(D - Ap_i) - \min_{x\in\mathcal{X}_1^i}(D - Ap_i) \quad (2)$$

where

$$D = \frac{\left|y_t^I - \rho_t^I\cdot x^I\right|^2 + \left|y_t^Q - \rho_t^Q\cdot x^Q\right|^2}{2\sigma^2} \quad (3)$$

$$Ap_i = \sum_{\substack{l=0\\ l\neq i \text{ and } v^l=1}}^{m-1} L\left(\hat{v}_t^l\right) \quad (4)$$

where v^l is the lth bit of each received modulated symbol.

The simplified expression in (2) exhibits four main computation steps: 2-D Euclidean distance computation, *a priori* LLR summation, minimum operations referred by the $\min$ functions, and subtraction operation of minimum values. In the case of noniterative demodulation, no *a priori* information is provided to the demapper. The expression of LLRs in (2) becomes

$$L\left(\tilde{v}_t^i\right) \approx \min_{x\in\mathcal{X}_0^i}(D) - \min_{x\in\mathcal{X}_1^i}(D). \quad (5)$$

Moreover, for Gray-mapped constellations, I and Q components are independent from each other; hence, the Euclidean distance is calculated in one dimension. In case where m is even, further simplification can be applied. LLR computation expression in (5) can be transformed in this case into the following [6]:

$$L\left(\tilde{v}_t^i\right) \approx \min_{x\in\mathcal{X}(I)_0^i}(D^I) - \min_{x\in\mathcal{X}(I)_1^i}(D^I) \text{ for } i = 0, 1, \ldots, \frac{m}{2} - 1 \quad (6)$$

$$L\left(\tilde{v}_t^i\right) \approx \min_{x\in\mathcal{X}(Q)_0^j}(D^Q) - \min_{x\in\mathcal{X}(Q)_1^j}(D^Q) \text{ for } j = \frac{m}{2}, \ldots, m-1 \quad (7)$$

where

$$D^I = \frac{\left|y_t^I - \rho_t^I\cdot x^I\right|^2}{2\sigma^2} \qquad D^Q = \frac{\left|y_t^Q - \rho_t^Q\cdot x^Q\right|^2}{2\sigma^2}$$

and $\mathcal{X}(I)_b^i$ and $\mathcal{X}(Q)_b^j$ are the constellation point sets on I-axis and Q-axis with ith and jth bits of symbol x that have a value equals to b. Applying this simplification, $2^{m/2}$ 1-D Euclidean distances are computed instead of 2^m 2-D Euclidean distances for each LLR. For rotated constellations, a simplification has been proposed in [9] to reduce the number of Euclidean distance computations by dividing the constellation into four subregions. This subpartitioning technique reduces the number of candidate constellation points involved in computing 2-D Euclidean distances of QAM schemes from 2^m to $(2^{(m-2)/2}+1)^2$.

III. Architecture Design

The variety of specifications in multiple wireless standards imposes designing hardware architecture of high flexibility to enable the computation of LLR values for different system

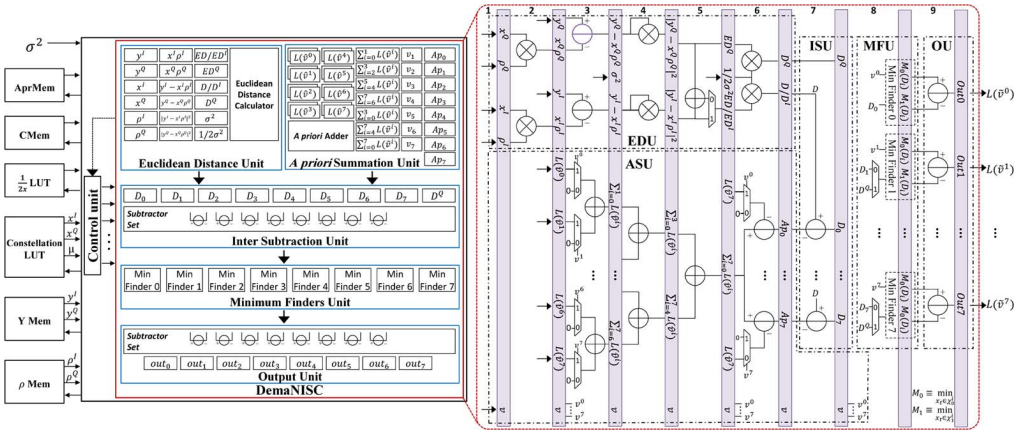

Fig. 2. Block diagram of the proposed NISC-based architecture detailing the pipeline structure of *DemaNISC* module.

configurations. The resources of the demapper flexibility are related to the characteristics of target constellation and the iterative demodulation concept. In this section, the architecture design proposed for the universal demapping is described. It is capable to generate soft-output probabilities in the logarithmic domain for various modulation schemes starting from BPSK up to 256-QAM with and without iterative demodulation using either the generic form or simplifications relative to the constellation rotation and the Gray mapping.

A. Architecture Choices

Toward achieving flexible demapper design, the following architecture choices are considered:

1) Modulation Order: Constellation information is stored in a lookup table (LUT) whose depth varies according to m. I and Q components of symbols and their corresponding binary mapping μ are rewritten for each target constellation. In addition, sufficient hardware operators, which are required to find minimum values and perform their corresponding subtractions required in (2), (5), (6), and (7), are allocated considering the largest target constellation (256-QAM). For lower order modulation schemes, unused resources are not activated. Sharing hardware resources among these operations decreases the throughput particularly for high-order modulation schemes. Furthermore, this architecture choice allows a fair comparison with the ASIP-based design presented in [2].

2) Mapping Style: The same hardware operators are utilized in computing Euclidean distances for Gray or non-Gray mapping styles. In fact, the computation of one 2-D distance is equivalent to that of two separate 1-D distances.

3) Iterative Demapping: Operators involved in *a priori* LLR summation are instantiated to accommodate all target constellations.

4) Quantization and Fixed-Point Arithmetic: To reduce the implementation complexity, fixed-point arithmetic is used, and computational values are quantized. Targeting a fair comparison with an ASIP-based design [2], identical quantization has been adopted for all computational parameters. With the aid of long simulations and analysis, bit widths are carefully selected to ensure least performance degradation. Using a reference software model, a degradation below 0.05 dB is measured at 10^{-3} frame-error rate over a fast fading Rayleigh channel.

5) Pipelining: Temporal parallelism using pipelining is applied to minimize the length of critical path and to enhance the computation performance and the efficiency of utilized hardware resources.

B. NISC Architecture

Fig. 2 presents the structure of the proposed architecture and shows the input/output connections. The inputs to the demapper architecture are the LLRs stored in *AprMem*, variance σ^2, *CWs* saved in *CMem*, constellation information arranged in *Constellation LUT*, and received symbols and fading factors reserved in *YMem* and *ρMem*, respectively. In addition, the LUT $(1/2x)$*LUT* provides the $1/2x$ inverse values required in the inversion operations.

The designed architecture is basically composed of a simple control unit and the module that performs the demapping functionality. Here, this module is referred to as *DemaNISC*.

1) Control Unit: It is mainly responsible for loading *CWs* from *CMem* into the components of *DemaNISC* module. To accomplish this functionality, the unit handles the address of *CMem* memory and constructs links to distribute the control-signal bits of *CWs* to appropriate components. In addition, the control unit manages the flow of input data coming from *YMem*, *ρMem*, and *Constellation LUT*. These basic tasks reveal the simple hardware structure required to implement the control unit.

2) DemaNISC Module: It is considered the main core of the demapper architecture. From a hierarchical scope, it can be viewed as a concatenation of five units.

a) EDU: This unit includes all hardware resources that incorporate in computing the Euclidean distance expressed in (3). It is supplied by I and Q components of received symbol y^I and y^Q, constellation symbol x^I and x^Q, and fading factor ρ^I and ρ^Q in addition to the noise variance σ^2. At each computation, Euclidean distance unit (EDU) can deliver two 1-D distances or one 2-D distance. EDU contains 18 registers, 6 real multipliers, 1 real adder, 2 real subtractors, and 1 2-to-1 multiplexer. The operators of the Euclidean distance calculator spread over five pipeline stages (stages 2–6, Fig. 2). In the second, third, and fourth pipeline stages, I and Q components of y, x, and ρ are exploited to compute two 1-D Euclidean distances. At the 5th pipeline stage, the two calculated distances may be added into one 2-D distance ED satisfying the implementation requirements of (2) and (5). In the case of Gray mapping style with no *a priori* information, the two 1-D distances are transferred to the next stage (ED^I and ED^Q). At this stage (fifth), the inverse value of σ^2, being provided at the $1/2x$ LUT index in the third pipeline stage, is retrieved. I and Q components of 1-D Euclidean distance (D^I and D^Q) or 2-D Euclidean distance D are ready at the end of the sixth pipeline stage (see Fig. 2).

b) ASU: In the case of turbo demodulation, the hardware resources embedded in this unit are responsible for generating

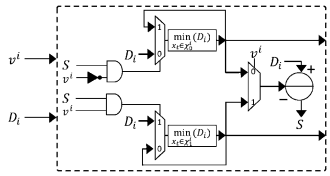

Fig. 3. Architecture of the minimum finder block.

the summation of input *a priori* LLRs as described in (4). The inputs to a priori LLR summation unit (ASU) are LLR values saved in *AprMem* memory and vector v representing the binary mapping μ of symbols from the *Constellation LUT*. ASU contains 38 registers, 7 real adders, 8 real subtractors, and 16 2-to-1 multiplexers. These components spread over five pipeline stages (stages 3–7, Fig. 2). The summation process is managed by the bit values of v, which represents the binary mapping of constellation symbol x under consideration and propagates along pipeline stages. In the case of turbo demodulation, input LLRs $L(\hat{v}_t^i)$, which are loaded in the second pipeline stage, are summed cumulatively in the third, fourth, and fifth stages. At this computational level, a copy of input *a priori* information is needed to subtract the LLR corresponding to bit v^i as expressed in (4). The final *a priori* information summations Ap_i corresponding to all bits are delivered at the end of the 6th pipeline stage.

c) ISU: This unit collects *a priori* LLR summation values produced by ASU and subtracts them in parallel from the value of 2-D Euclidean distance calculated by EDU. To perform this functionality, intersubtraction unit (ISU) includes a subtractor set, which is made of eight real subtractors considering the highest modulation order with $m = 8$. ISU also contains nine registers to store the subtraction results ($D_i = D - Ap_i$) in addition to the value of the Q-component of the 1-D Euclidean distance D^Q. Subtractors of ISU are placed in the seventh pipeline stage and are capable of producing up to eight D_i values (D_0 to D_7). If turbo demodulation is omitted, the computed values of Euclidean distance are transferred to the next pipeline stage (eighth) with no modifications. At the end of the seventh stage, three types of data are possible to be achieved:

- one 2-D distance minus the *a priori* information as expressed in (2);
- one 2-D distance only (noniterative demodulation case) as expressed in (5);
- two 1-D distances (Gray mapping with noniterative demodulation case) as expressed in (6) and (7).

d) MFU: This unit integrates eight minimum finder blocks whose architecture is presented in Fig. 3 and are established to realize minimum functions listed in (2), (5), (6), and (7) considering 256-QAM constellation. Each block is concerned to find minimum values associated to a bit location v^i along all constellation symbols. As shown in Fig. 3, the minimum finder structure contains two registers. The first register stores the updated minimum value that corresponds to symbol set $\mathcal{X}_0^i$; whereas the second register stores the minimum value corresponding to $\mathcal{X}_1^i$. For each new received symbol y, the two registers are initialized by loading the maximum numerical value. Each minimum finder benefits from new-updated D_i values in addition to v^i bits. Depending on v^i value ("0" or "1"), one of the two registers is chosen to be updated. The current value is replaced by input value D_i if the latter is smaller than the former. Otherwise, the register maintains its current value. A comparison is established by evaluating the sign S of the difference resulting from the subtraction operation of current value from input value D_i. At the input of each minimum finder block, a multiplexer is allocated to control input data flow to minimum finder block according to the required dimension of the Euclidean distance. Overall, minimum finder unit (MFU) is composed of 16 registers, 8 real subtractors, 31 2-to-1 multiplexers, 16 AND-gates, and 16 negators. All these components are placed at the eighth pipeline stage.

e) OU: This unit is in charge of delivering finally the LLR values corresponding to each bit $L(\tilde{v}_t^i)$ constellation symbol x. Inputs of the output unit (OU) are the minimum values available in the registers of MFU. Once minimum values of all constellation points are determined, the OU produces the difference between minimum pairs ($\min_{x_t \in \mathcal{X}_0^i}$ and $\min_{x_t \in \mathcal{X}_1^i}$) corresponding to each bit location v^i. After processing all constellation symbols, final resultant differences are loaded to their corresponding output registers (out_0 to out_7).

TABLE I
SYNTHESIS RESULTS

	Proposed design	*DemASIP* [2]
FPGA Synthesis Results (Xilinx Virtex5 xc5vlx330)		
Slice Registers	1,328 out of 207,360	1,918 out of 207,360
Slice LUTs	1,524 out of 207,360	3,201 out of 207,360
DSP48Es	6 out of 192	6 out of 192
Frequency	240 MHz	186 MHz
ASIC Synthesis Results (Synopsis Design Compiler)		
Technology	**65** nm **ST CMOS**	
Conditions	**nominal case** ($1V$; 25^oC)	
Area	0.048 mm^2	0.053 mm^2
Frequency	520 MHz	518 MHz

TABLE II
EXECUTION PERFORMANCE RESULTS

Modulation Type	Clock Cycles for 1 symbol		Throughput (Mega LLR/second) FPGA		ASIC	
	Proposed design	*DemASIP* [2]	Proposed design	*DemASIP* [2]	Proposed design	*DemASIP* [2]
Adopting Gray mapped constellations						
QPSK	3	4	160	93	347	259
16-QAM	5	6	192	124	416	345
64-QAM	9	10	160	111.6	347	310.8
256-QAM	17	18	112.94	82.67	244.7	230.2
Adopting DVB-T2 rotated constellations						
64-QAM	26	27	55.38	41.33	120	115.11
256-QAM	82	83	23.41	17.93	50.73	49.93

IV. RESULTS AND COMPARISON

In this brief, the NISC design toolset has been used. The adopted design flow is thoroughly described in [13]. Table I summarizes the synthesis results, whereas Table II shows the number of clock cycles required to produce LLRs for different system configurations along with the achieved throughput. The results are in addition compared with those of *DemASIP* [2], which is an ASIP dedicated for demapping with customized data path and instruction set. Both processors support same flexibility parameters, use identical computational resources, and adopt identical quantization for all computational parameters. The comparison between the two designs shows a significant improvement in terms of execution time and implementation area. For fair comparison, logic synthesis of the proposed architecture HDL description has been conducted targeting the same device (Xilinx Virtex-5 LX330 FPGA) and using the same synthesis options and tools. Logic utilization is reduced by 30.8% for slice registers and 52.4% for slice LUTs compared with *DemASIP*. In fact, the implementation of resources responsible for instruction fetching and decoding increases hardware resource utilization. Whereas, in the NISC-based proposed demapper, the architecture is designed

to match exactly the requirements of the application. Moreover, *DemASIP* can operate at a maximum frequency of 186 MHz, whereas the proposed architecture can achieve a maximum operating frequency of 240 MHz. Hence, it is 1.29 times faster than *DemASIP*. This is due to the *DemASIP* critical path that includes several levels of combinational logic and which is related to fetch program address generator, whereas only one level of logic exists in the critical path of the proposed design. In addition, Table I shows a comparison between the ASIC implementation of *DemASIP* and that of the NISC-based architecture on dedicated chips. In fact, the HDL code generated by Synopsys (ex CoWare) for the ASIP-based design (which is available at our research group) and the HDL code generated by NISC toolset for NISC-based design are delivered as sources to the Design Compiler tool from Synopsys to achieve the ASIC implementation targeting the ST CMOS 65-nm technology. The comparison shows that the NISC-based design offers about 10% area reduction compared with the ASIP-based design. Note that the detailed results illustrate that the control unit occupies only 1.6% of total implementation area of proposed architecture, whereas the components dedicated for fetching and decoding instructions occupy 7% of *DemASIP* implementation area.

On the other hand, operations in ASIP-based design are limited to the available instruction set. Overlapping of operations is not allowed. The instructions are fetched and decoded at runtime, and their functionalities are limited to their structure. Whereas, the NISC-based proposed design enables a direct and mastered access to control signals of hardware resources. Different operations are combined and then scheduled statically. Merging of operations ensures less execution time compared with *DemASIP*. At runtime, operations are directly performed, and LLRs are generated with no additional overhead. Table II shows that the proposed architecture outperforms *DemASIP* in all system configurations. For all combinations of mapping styles, modulation types, and SSD, better throughput is always provided.

Concerning *CMem* requirements, its width is significantly optimized (from 91 bits to 18 bits) by specifying same control bits to all components that have the same executions in all steps, whereas its depth varies according to the number of the needed execution steps to compute all LLRs corresponding to one input symbol. The required memory size varies from 23 B for quadrature PSK (QPSK) with Gray-mapped constellation to 594 B for 256-QAM with non-Gray mapped constellation. Compared with the ASIP program memory, *CMem* requires less memory space to be implemented. In fact, the assembly code used for *DemASIP* includes in addition to PROCESS instruction, which is the core instruction of LLR generation, instructions dedicated to loading input data, exporting output LLRs, looping, and no-operation instructions [2]. The available assembly code for QPSK non-Gray constellation shows that these additional instructions forms 54% of the total number of listed instructions [2]. For this system configuration, the memory space required to implement the program memory of *DemASIP* is 60.3% more than that required to implement the control memory (*CMem*) of the NISC-based proposed demapper. Note that, for both processors, CWs and assembly code are produced and optimized by hand as in this case the hardware is highly dedicated to the application, and it is programmed by its designers and not by its users.

Furthermore, the proposed demapper is compared with the dedicated architectures reported in [8] and [9], which use a classical RTL design approach. These architectures support soft-decision demapping; however, their flexibility is limited to the requirements of DVB-S2 and DVB-T2 standards, respectively. Targeting a fair comparison, our proposed design has been implemented on the same target devices used in [8] (Virtex II XC2-V6000) and [9] (Virtex II Pro XC2VP30). Compared with the architecture in [9], the proposed demapper requires almost 3.33 times less dedicated multipliers, 3.1 times less LUTs, but 2.2 times more registers. In terms of throughput, it outperforms the design in [9] for QPSK by 6.2 times, for 16-QAM by 3.1 times, and for 64-QAM by 1.2 times. Whereas for 256-QAM, the design in [9] shows better throughput (2.6 times) since it can concurrently compute nine Euclidean distances. In [8], the timing information about the hardware implementation is not available. The presented logic utilization summary reveals the need of 1.8 times more logic devices and 2.67 times more multipliers compared with the proposed NISC-based demapper.

V. Conclusion

In this brief, an NISC-based architecture of universal demapper for multiple wireless communication standards has been proposed. The flexibility of the demapper is not restricted to certain modulation types and/or mapping styles. Hardware design and implementation have been conducted using the NISC design approach. While using identical computational resources and supporting same flexibility parameters, the proposed NISC-based demapper architecture outperforms state-of-the-art ASIP-based architecture with reduced implementation costs. In addition, less memory space is required to implement control memory compared with the ASIP program memory.

References

[1] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*. New Delhi, India: Wiley, 2006.
[2] A. R. Jafri, "Architectures multi-ASIP pour turbo recepteur flexible," Ph.D. dissertation, Elect. Dept., Telecom Bretagne, Brest, France, 2011.
[3] D. Gajski, "NISC: The ultimate reconfigurable component," Center Embedded Comput. Syst. (CECS), Univ. California, Irvine, CA, USA, Tech. Rep., Oct. 2003.
[4] M. Rizk *et al.*, "Flexible and efficient architecture design for MIMO MMSE-IC linear turbo-equalization," in *Proc. IEEE ICCIT*, Jun. 2013, pp. 340–344.
[5] E. Agrell *et al.*, "On the optimality of the binary reflected Gray code," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3170–3182, Dec. 2004.
[6] E. Akay and E. Ayanoglu, "Low complexity decoding of bit-interleaved coded modulation for M-ary QAM," in *Proc. IEEE ICC*, Jun. 2004, vol. 2, pp. 901–905.
[7] X. Chen *et al.*, "VLSI implementation of a high-throughput iterative fixed-complexity sphere decoder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 5, pp. 272–276, May 2013.
[8] J. W. Park *et al.*, "Low complexity soft-decision demapper for high order modulation of DVB-S2 system," in *Proc. IEEE ISOCC*, Nov. 2008, vol. 2, pp. II 37–II 40.
[9] L. Meng, C. Abdel Nour, C. Jego, and C. Douillard, "Design of rotated QAM mapper/demapper for the DVB-T2 standard," in *Proc. IEEE Workshop SiPS*, Oct. 2009, pp. 18–23.
[10] A. Osseiran *et al.*, "Scenarios for 5G mobile and wireless communications: The vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.
[11] C. Abdel Nour, "Spectrally Efficient Coded Transmission for Wireless and Satellite Applications," Ph.D. dissertation, Dept. Elect., Telecom Bretagne, Brest, France, 2008.
[12] P. Robertson *et al.*, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *Eur. Trans. Telecom.*, vol. 8, no. 2, pp. 119–125, Mar./Apr. 1997.
[13] M. Rizk *et al.*, "Design and prototyping flow of NISC-based flexible MIMO turbo-equalizer," in *Proc. IEEE Int. Symp. RSP*, Oct. 2014, pp. 16–21.