An analytical approach for sizing of heterogeneous multiprocessor flexible platforms for iterative demapping and channel decoding

Vianney Lapôtre, Guy Gogniat, Jean-Philippe Diguet
Université de bretagne Sud, CNRS Lab-STICC UMR 6285
Lorient, France
Email: firstname.lastname@univ-ubs.fr

Salim Haddad, Amer Baghdadi
Institut Telecom, Telecom Bretagne, CNRS Lab-STICC UMR 6285
Brest, France
Email: firstname.lastname@telecom-bretagne.eu

***Abstract*—Flexible baseband receivers gain the interest of many research efforts to enable the design of future multi-modes multi-standards terminals. A main challenge in this domain is to provide this flexibility with minimum overhead in terms of area, speed, and energy. In this regard, heterogeneous multiprocessor platforms are emerging as a promising implementation solution. However, the heterogeneity of such platforms makes it complex to find the required number of processors supporting a specific configuration (i.e. requirements level).**

This paper investigates, in this context, the significant optimization potential both *at design-time* and *at run-time* regarding the selection of the most appropriate hardware configuration of a multiprocessor platform for iterative demapping and channel decoding. A formal representation of the architectural solution space which allows designers to find the minimum hardware configuration is proposed. The proposed approach is illustrated through a flexible multi-ASIP hardware platform for iterative demapping and channel decoding.

***Keywords*-Multiprocessor, ASIP; Self-adaptation; Wireless multi-standards receiver; Platform sizing; Run-time; Design-time;**

I. Introduction

Last years have seen considerable evolutions of wireless communication standards in the domain of cellular telephone networks, local/wide wireless area networks, and Digital Video Broadcasting (DVB). Besides the increasing requirements in terms of throughput and robustness against destructive channel effects, the convergence of services in single smart terminal becomes a crucial and challenging feature. As an example, the fourth generation (4G) of cellular wireless standards aims at providing mobile broadband solution to laptop computer wireless modems, smartphones, and other mobile devices. Diverse features such as ultra-broadband Internet access, IP telephony, gaming services, and streamed multimedia will be provided.

In order to enable such advanced services at the algorithmic level, new state of the art data processing techniques have been developed and adopted in the emerging wireless communication standards. At the architecture level, many efforts are being conducted towards the design of flexible high throughput hardware platforms which can be configured to the required configuration. The overall flexibility of the radio platform can be achieved through the flexibility of individual components at transmitter side (encoder, interleaver, mapper, etc.) and at receiver side (demapper, deinterleaver, decoder, etc.). In this context, heterogeneous multiprocessors platforms [1], [2], [3] have been widely adopted. These platforms usually integrate different tiles that provide high performances and high flexibility to respect services requirements. ASIP based tiles have been adopted to provide flexible and powerful solutions. For example, in [1], an 10.8 Mbps ASIP core is used for turbo-decoding. However, the high throughput requirement of emerging services imposes the efficient exploitation of different parallelism levels. Several recent works propose multiprocessor approaches to build these tiles [4], [5], [6], [7]. In this work we investigate the sizing of a heterogeneous multiprocessor flexible platform for iterative demapping and channel decoding and propose a novel approach for efficient *design-time* and *run-time* sizing. We illustrate how for a given level of requirement several architecture alternatives with different number of processors exist. A formal representation of the architectural solution space is proposed. This formulation enables the designer to find the most efficient hardware configuration. Based on this formal representation, the architecture can be chosen both at *design-time* and at *run-time* according to an optimization objective which could be, for example, minimizing the number of processors, reducing the active area on the chip, reducing the clock frequency, etc.

The proposed approach is illustrated through a flexible multi-ASIP hardware platform for iterative demapping and channel decoding. This platform integrates two different types of ASIPs (Application-Specific Instruction-set Processor): one for demapping, called DemASIP, and the second for turbo decoding, called DecASIP. This paper presents the following contributions:

- A formal representation of the architectural solution space is proposed.
- A method to apply this formal representation at *design-time* and at *run-time* is defined.
- A use case that demonstrates the interest of the proposed method to reduce the chip area at *design-time* and the active area at *run-time* is presented and evaluated.

The rest of the paper is organized as follows. Section II provides an overview of relevant literature. Section III presents the system model and the configuration parameters. Section IV describes the proposed formal representation of the architectural solution space which allows the designer to size the platform depending on the system configuration. Section V evaluates the impact of a design-time sizing on the chip area and the impact of a run-time sizing on the active area for different receiver configurations. Finally, section VI provides a discussion on the proposed work and concludes the paper.

II. State of the Art

The high throughput requirement of emerging services imposes the efficient exploitation of different parallelism levels. In this context, multiprocessor architecture [4], [5], [6], [7] is a promising approach to reach high flexibility, high throughput and energy efficiency. In [4], an heterogeneous architecture for convolutional and turbo-decoding consisting of a dedicated 150Mbps IP block and a cluster of ASIPs is presented. The dedicated IP block is used when high throughput is required while ASIPs are used for lower throughput. Even if the authors superficially describe a multi-ASIP architecture in which the ASIPs are connected through a crossbar, the sizing of such an architecture is not addressed. The presented results are limited to two ASIPs that share a memory. In [5] and [6], the authors present a multi-ASIP platform for decoding in which the ASIPs are connected through a Network

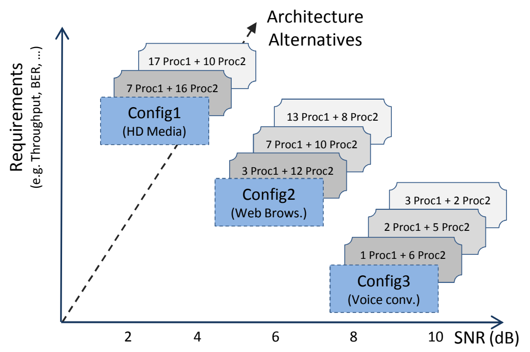

Figure 1. Usage scenario example of the considered heterogeneous multiprocessor platform integrates two different types of processors that perform demapping and decoding algorithms respectively (Proc1 and Proc2).

on Chip. The high flexibility of such architectures allows dynamic reconfiguration at run-time but the run-time sizing task is not addressed. In [5], different decoding tasks can be mapped at run time on different ASIPs but no methodology to define the number of processors necessary to perform a given configuration is presented. As in [7], recent works propose to combine several functionalities, like decoding and demapping, in a multi-ASIP heterogeneous platform. Unfortunately, the sizing of such platforms is not well addressed in the literature. We assume that the designer, based on his background and simulations, has to deal with the sizing of these complex heterogeneous platforms. However this approach could provide sub-optimal solutions and decrease the sizing flexibility at run-time since all the decisions are taken at design-time. In fact, flexible hardware multiprocessor platforms for iterative demapping and channel decoding are generally designed to support a set of communication standards which correspond to some specific application needs and usage scenarios. Each usage scenario corresponds to particular requirements for example in terms of throughput, latency, error rates, and/or others. Fig. 1 gives an example of such usage scenario which corresponds to a mobile terminal supporting different services (High Definition Multimedia, Web Browsing, Voice Conversation) at different channel conditions. Hence, *at design-time*, the platform must be dimensioned to support the highest requirements while, *at run-time*, the number of processors can be chosen depending of the current level of requirements. Furthermore, for a heterogeneous multiprocessor platform, and for a specific requirement level, several architectural configurations (i.e. with a different number of each type of processors) exist (Fig. 1). The alternatives exploration and the selection of the most appropriate one is a complex task. However, it represents an important optimization room which is not investigated in the literature. In this paper we address this point and propose a formal approach to find the optimal configuration.

III. System model and configuration

In order to illustrate the proposed approach for sizing of heterogeneous multiprocessor flexible platform for iterative demapping and channel decoding we consider in this paper the communication system model of Fig. 2. It consists of a multi-modes advanced wireless communication system integrating convolutional turbo code, Bit Interleaved Coded Modulation (BICM), various modulation schemes, and Signal Space Diversity (SSD). A brief presentation of the system model and the considered parameters is given in this section.

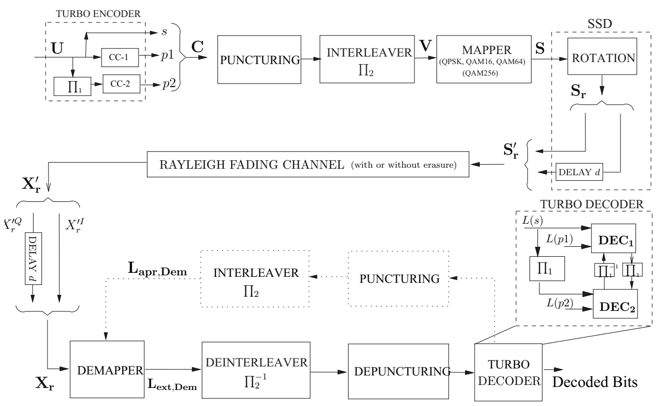

Figure 2. System model with TBICM-ID-SSD.

A. System model

Fig. 2 presents a simplified structure of the transmitter, the channel, and the receiver. On the transmitter side, information bits U which are called systematic bits are regrouped into symbols u_i consisting of k bits, and encoded with a q-binary turbo encoder. It consists of a parallel concatenation of two identical convolutional codes (PCCC). The output codeword C is then punctured to reach a desired coding rate R_c. In order to gain resilience against error bursts, resulting sequence is interleaved using an S-random interleaver Π_2. Punctured and interleaved bits denoted by v_i are then Gray mapped to complex channel symbols s_q chosen from a 2^M-ary constellation X, where M is the number of bits per modulated symbol. Applying the SSD consists of a rotation of the constellation followed by a signal space component interleaving.

At the receiver side (which is the topic of this paper), the corresponding operations to the transmitter ones are applied in reverse order. However, in order to meet the increasing requirements in terms of reduced error rates, the iterative processing is considered at two levels. The first level is at the channel decoding by adopting a turbo decoding process. The second level is between the channel decoder and the soft demapper. In fact, besides extrinsic information exchange inside the channel turbo decoder, additional extrinsic information is feedback as a priori information used by the demapper to improve the symbol to bit conversion. Thus, the receiver model, denoted as TBICM-ID-SSD, implements iterative demapping with turbo decoding.

B. Receiver configuration

A flexible software model of the whole system of the Fig.2 (transmitter, channel, and receiver) was developed. This model supports many parameters corresponding to the constellation type and modulation order, interleaving laws, turbo code type, code rate, and frame size.

Furthermore, the receiver can be configured to execute iterative or non iterative demodulation. For the case of iterative demodulation, state of the art implementations apply one turbo code iteration for each demapping iteration [8]. Thus, the number of demapping iterations (it_{dem}) is equal to the number of turbo decoding iterations (it_{dec}) in this case. This number of iterations constitutes another flexible parameter of the system model. On the other hand, for non iterative demodulation it_{dem} will be equal to 1.

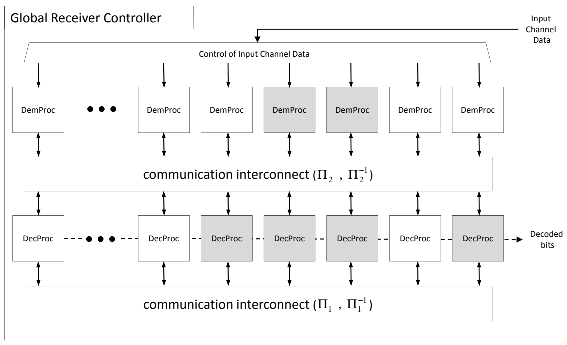
Global Receiver Controller
Input Channel Data
Control of Input Channel Data
DemProc
communication interconnect (Π_2 , Π_2^{-1})
DecProc
Decoded bits
communication interconnect (Π_1 , Π_1^{-1})

Figure 3. Generic architecture of the heterogeneous multiprocessor receiver. In this configuration, 2 DemProcs and 4 DecProcs are not used.

The configuration of these flexible parameters is generally constrained by the available communication standard, the channel condition, and the target system requirements in terms of throughput, latency, and error rate performance. The determination of their values should also take into consideration the complexity issue in order to advise the most efficient configuration (as many solutions generally exist). This task is out of the scope of this paper. However, in order to define the suitable system configurations of the usage scenario that will be considered in Section V, communication system experts were inquired and extensive simulations were conducted.

Based on the system model, the next section proposes a generic architecture model and a formal method for an efficient sizing of such platforms.

IV. Platform sizing

Multi-standards and multi-modes platforms have to be able to self-adapt when application requirements and environment evolve at run-time. A configuration is defined by the communication parameters which are chosen in accordance with the application requirements and the environment in which the communication is established. In this section we propose formal expressions which allow designers to optimize the receiver architecture by computing the required number of processors depending on each configuration. This point is essential as it enables designers to formally explore potential architectures that will meet performance constraints.

A. Generic heterogeneous multiprocessor architecture model

Fig. 3 presents the generic architecture of a flexible multiprocessor hardware platform for iterative demapping and channel decoding. The aim of this platform is to provide a flexible and dynamic solution compared to existing ones [1], [2], [3] (generally based on hardware accelerators) where designer can tune the number of resources both at design-time and at run-time. As it will be presented, such an approach allows the system meeting performance constraints without loosing its flexibility. These features will be mandatory for future communication systems. In Fig. 3, DemProc and DecProc perform demapping and decoding algorithms respectively. These two processors are characterized by their area, maximum frequency, and their performance defined by the number of cycles to demap or decode one modulated or coded symbol respectively. The platform integrates a communication interconnect that allows extrinsic information exchanges (between DecProcs themselves and between DecProcs and DemProcs). In this paper, we assume that the communication interconnect is designed for the worst case configuration in which all processors exchange data at the same time and it is congestion and conflict free.

B. Formal representation of the architectural solution space

The generic architecture of Fig. 3 can be abstracted as two components: one demapper and one decoder. Each component uses several processors in parallel to perform the frame computation exploiting sub-bloc parallelism. These two components are serially connected. The time required to process one frame (T_{syst}) corresponds to the sum of the time required by the demapper (T_{dem}) and the time required by the decoder (T_{dec}) to execute all their iterations on the frame. It can be expressed as:

$$\begin{aligned} T_{syst} &= T_{dem} + T_{dec} \\ &= N_{dem}.T_{dem/symb} + N_{dec}.T_{dec/symb} \end{aligned} \tag{1}$$

where N_{dem} and N_{dec} represent, respectively, the number of modulated and coded symbols per frame. $T_{dem/symb}$ and $T_{dec/symb}$ represent the time required by the demapper and the time required by the decoder to execute all their iterations on one modulated and coded symbol respectively. Hence, the system throughput ($D_{syst} = N_{dec}/T_{syst}$) can be expressed as below.

$$D_{syst} = \frac{D_{dem}.D_{dec}.N_{dec}}{N_{dem}.D_{dec} + N_{dec}.D_{dem}} \tag{2}$$

where D_{dem} $(= 1/T_{dem/symb})$ and D_{dec} $(= 1/T_{dec/symb})$ are the demapper and the decoder throughputs (in modulated and coded symbols, respectively). In fact, considering the code rate R_c and the number of bits per symbol M, the relation between the number of coded symbols (N_{dec}) and the corresponding number of modulated symbols (N_{dem}) can be written as follows.

$$\begin{aligned} N_{dem} &= \frac{q}{M.R_c}.N_{dec} \\ &= \alpha.N_{dec} \end{aligned} \tag{3}$$

where q depends on the coding scheme ($q = 1$ for simple binary turbo code and $q = 2$ for double binary turbo code). Introducing this expression of N_{dem} into equation (2) gives the following system throughput expression.

$$D_{syst} = \frac{D_{dem}.D_{dec}}{D_{dem} + \alpha.D_{dec}} \tag{4}$$

The throughput of the system D_{syst} is generally imposed by the application requirement. On the other hand, the throughputs of the demapper and the decoder depend on the number of processors, the number of iterations, the number of clock cycles required to process on symbol, and the clock frequency. They can be expressed as follows.

$$D_{dem} = \frac{Nb_{demProc}.F_{dem}}{it_{dem}.cycles_{dem/symb}} \tag{5}$$

where $Nb_{demProc}$ is the number of demapping processors, it_{dem} is the number of demapping iterations, $cycles_{dem/symb}$ is the number of cycles necessary to demap one symbol, and F_{dem} is the clock frequency.

$$D_{dec} = \frac{Nb_{decProc}.F_{dec}}{2.it_{dec}.cycles_{dec/symb}} \tag{6}$$

where $Nb_{decProc}$ is the number of decoding processors, it_{dec} is the number of decoding iterations, $cycles_{dec/symb}$ is the number of cycles necessary to decode one symbol, and F_{dec} is the clock frequency.

n	$Nb_{demProc}$	$Nb_{decProc}$
0.25	40	44
0.75	56	21
1	64	18
1.25	72	16
1.75	88	14

Table I
ARCHITECTURE ALTERNATIVES IN FUNCTION OF N. EXAMPLE FOR: $D_{syst} = 200$ MBPS, QPSK, $R_c = 0.5$, $it_{dem} = it_{dec} = 8, cycles_{dem/symb} = 6, cycles_{dec/symb} = 1.75$ AND 0.75 FOR THE LAST ITERATION, $F_{dec} = F_{dem} = 300MHz$

It is worth noting that the linear increase in throughput with the number of decoding processors is limited due to the sub-bloc initialization issue [9]. This limitation, which depends on the target frame size and code rate, should be considered in the platform sizing. However, this issue is not encountered in the demapping sub-bloc parallelism.

In order to establish a relation between the demapping time and the decoding time, we define the ratio n as follows.

$$n.T_{dem} = T_{dec} \tag{7}$$

From this equation we can obtain a relation between the throughputs of the demapper and the decoder:

$$\begin{aligned} n.\frac{N_{dem}}{D_{dem}} &= \frac{N_{dec}}{D_{dec}} \\ D_{dem} &= D_{dec}.n.\frac{N_{dem}}{N_{dec}} \\ D_{dem} &= D_{dec}.n.\alpha \end{aligned} \tag{8}$$

We deduce from (8) and (4) the equations which link the throughput of the system with the throughputs of the demapper and the decoder:

$$D_{dec} = \frac{n+1}{n}.D_{syst} \tag{9}$$

$$D_{dem} = \alpha.(n+1).D_{syst} \tag{10}$$

Finally, from equations (5) and (6) we can express $Nb_{demProc}$ and $Nb_{decProc}$ as follows.

$$Nb_{demProc} = C_{dem}.D_{dem} \tag{11}$$

$$Nb_{decProc} = C_{dec}.D_{dec} \tag{12}$$

where $C_{dem} = \frac{it_{dem}.cycles_{dem/symb}}{F_{dem}}$ and $C_{dec} = \frac{2.it_{dec}.cycles_{dec/symb}}{F_{dec}}$ depend on the system configuration and the processor parameters.

Replacing D_{dem} and D_{dec} by their expressions from equations (10) and (9) allows to compute the number of processors necessary for a given configuration and a given n.

$$Nb_{demProc} = C_{dem}.\alpha.(n+1).D_{syst} \tag{13}$$

$$Nb_{decProc} = C_{dec}.\frac{n+1}{n}.D_{syst} \tag{14}$$

Table I illustrates, for a given configuration, how different values of n lead to different architecture alternatives, although all of them acheiving the target throughput and supporting the target system configuration. Depending on n we observe that the architecture alternative could be quite different. For example, when n= 0,25 the architecture consists of 40 processors for demapping and 44 processors for decoding while when n=1.25, 72 processors for demapping and 16 processors for decoding are necessary. It is essential, both at *design-time* and at *run-time*, to determine the value of n which optimizes the resources use. The optimization goal depends of designers priorities and could be for example the number of processors used for each possible configuration, the total area of the chip, the clock frequency for each type of processor, etc. In this paper we extend the previous equations in order to optimize the total area of the chip at *design-time*. The same optimization can be applied at *run-time* in order to reduce the active area for the configurations performed on the platform.

C. Area optimization

Heterogeneous processors have typically different areas and performances. One main optimization objective is to determine the number of DemProcs and DecProcs in order to minimize the receiver area for a given configuration. The total area of the receiver depends on n. It can be computed using the expression below.

$$A_n = A_{dem}.Nb_{demProc} + A_{dec}.N_{decProc} \tag{15}$$

where A_{dem} and A_{dec} are the area of one DemProc and one DecProc respectively. Therefore, by putting equations (11), (12) and (8) into equation (15), A_n can be expressed as a function of C_{dem} and C_{dec}.

$$A_n = (C_{dec}.A_{dec} + C_{dem}.A_{dem}.\alpha.n)D_{dec} \tag{16}$$

On the other hand, using equation (4), D_{dem} can be expressed as:

$$D_{dem} = \frac{\alpha.D_{syst}.D_{dec}}{D_{dec} - D_{syst}} \tag{17}$$

Moreover, D_{dec} can be expressed as a function of D_{syst} and n by putting equation (8) equals to equation (17).

$$D_{dec} = \frac{D_{syst}(n+1)}{n} \tag{18}$$

Finally, A_n can be expressed as a function of n by putting the equation of D_{dec} above into equation (16).

$$A_n = \frac{a.n^2 + b.n + c}{n} \tag{19}$$

where
$$a = C_{dem}.A_{dem}.D_{syst}.\alpha$$
$$c = C_{dec}.A_{dec}.D_{syst}$$
$$b = a + c$$

The derivative function of the equation 19 is then computed. Only one extremum (n_{ext}) is found.

$$n_{ext} = \sqrt{\frac{c}{a}} = \sqrt{\frac{2.it_{dec}.cycles_{dec}.F_{dem}.A_{dec}}{it_{dem}.cycles_{dem}.F_{dec}.A_{dem}.\alpha}} \tag{20}$$

The second derivative function is also computed at n_{ext}. It shows a positive value corresponding to the minimum area (A_{next}) of the receiver. Finally, A_{next} can be expressed as:

$$A_{next} = a + c + 2\sqrt{a.c} \tag{21}$$

For a given configuration, n_{ext} is determined with equation (20). With the obtained value of n_{ext}, the number of DemProc and DecProc which minimizes the area can be calculated using equations (13) and (14). The number of processors is then rounded up to guarantee the throughput constraint. Note that, due to the sub-bloc initialization issue [9], the number of DecProc is limited by the maximum number of frame sub-blocs that can be extracted from the entire frame. If the number of processors determined is upper that this limit, their number is saturated in accordance to the maximum level of available parallelism and the corresponding number of DemProc is computed with respect to the throughput requirement.

Based on the set of equations above it is now possible to analyze how the system can be tuned both at design-time and at run-time to meet performance requirements for a given configuration.

D. Design-time sizing

Platform sizing at design-time allows designers to determine the hardware configuration which minimizes the total area of the chip. This objective has been considered as it strongly impacts the cost of the chip. The design space of potential configurations is too large to allow designers to efficiently explore all possible architecture alternatives. So first, the designer needs to list the critical configurations that will be executed on the platform. These configurations will require the largest number of operations to demap and decode a frame. Then, equations (20), (13) and (14) are successively used to determine for each critical configuration the architecture alternative which minimizes the total area. Finally, The number of processors implemented on the chip is the maximum number of DemProcs and DecProcs among the different hardware configurations.

E. Run-time sizing

Platform sizing at run-time allows to determine the hardware configuration which optimizes the resources usage. Several objectives can be addressed using equations previously described. When a configuration has to be executed on the platform, the architecture alternative which optimizes the objective can be determined using the proposed equations. Configuration parameters are used to determine n_{ext} which optimizes the objective. Then, n_{ext} is used in equations (13) and (14) to compute the required number of DemProcs and DecProcs. If the number of processors required to perform the configuration is lower than the platform capacity, which is defined at design-time, unused cores can be for example switched-off as they will not be use during the execution of the current configuration.

In this section, we have proposed formal equations to explore the alternative architectures of a heterogeneous multiprocessor receiver. These equations have been applied on a particular optimization objective in order to optimize the total area used for a given configuration. We have also explained how to consider such a solution both at design-time and at run-time. The next section presents the results of this method on a typical heterogeneous multi-ASIP receiver.

V. Case study and results

A. Multi-ASIP platform

In order to apply and evaluate the proposed approach we consider the heterogeneous multi-ASIP receiver platform presented in [7]. This platform integrates two types of ASIPs which perform the main functions of iterative demodulation and turbo decoding. The first, called DemASIP [10], is dedicated to the Max-Log-MAP Demapping Algorithm. This ASIP can be used for multiple modulation schemes adopted at the transmitter side. The DemASIP provides support for BPSK to 256-QAM constellation for any mapping style with or without SSD. Depending on the modulation scheme, the time to demap one symbol evolves from 6 to 258 clock cycles. The second called DecASIP [6], performs the Max-Log-MAP Decoding Algorithm. It supports convolutional turbo codes up to eight-state double binary turbo codes or sixteen-state simple binary codes. It is able to decode a symbol in 1.75 clock cycles in turbo-demodulation scheme and in 0.75 clock cycle when only turbo-decoding is applied.

Conf.	Mod.	Throughput (Mbps)	Freq. (MHz)	it_{dem}	it_{dec}	R_c
1	QPSK	200	300	8	8	1/2
2	16-QAM	200	300	6	6	1/2
3	64-QAM	200	300	1	8	1/2
4	64-QAM	200	300	1	9	2/3

Table II
Use case configurations

Based on this platform and in order to demonstrate the benefits of the proposed approach, a representative use case has been developed.

B. Use Case Study

The considered use case targets at the output of the channel decoder a throughput of 200 Mbps with BER performance between 10^{-5} and 10^{-6} for an SNR range from 3dB to 13dB. It can correspond to future wireless HD Media service in mobility context (e.g. during a train trip).

In order to find the suitable system parameters of the flexible communication system model of Fig. 2 with respect to this use case, extensive simulations were conducted. Results of this step (out of the scope of this paper, cf. sub-section III-B) correspond to the configurations described in Table II. The frequency of each ASIP is 300 MHz. The other parameters evolve depending on the environment conditions: Conf. 1 corresponds to severe channel conditions (i.e. lowest SNR) whereas Conf. 4 corresponds to good channel conditions (i.e. highest SNR).

C. Results

The first step of platform sizing is performed at *design-time*. For this purpose, the method described in sub-section IV-D is used to determine the best hardware sizing for the critical configurations which will be performed on the platform. For the scenario previously explained, the critical configurations are Conf. 1 and Conf. 2. They require higher number of iterations than Conf. 3 and Conf. 4 (Table II). These configurations determine the maximum number of ASIPs which needs to be implemented on the chip. Table III shows the comparison between the number of processors and the total area of the chip using the proposed method and an approach where n_{ext} is not defined. In that last case, the designer may not be able to efficiently tune the ratio between the time to demap a frame and the time to decode a frame in order to minimize the total area of the chip. A manual exploration based on the designer experience can still be performed in order to test several ratio but such an approach is time consuming and there is no guarantee to find the best one. Thus a default value of $n = 1$ is generally used. $n = 1$ means that the time to demap a frame is equal to the time to decode a frame. The last row of Table III corresponds to the number of processors that have to be implemented to support the highest requirements. It is determined by selecting the maximum number of processors needed to perform critical configurations. For this case, results show that using our formal equations allows to save 9.6% of the total area by implementing 55 DemASIP and 23 DecASIP instead of 72 DemASIP and 18 DecASIP when a default value of n is used.

Conf.	proposed method				no exploration				Gain
	n	$Nb_{demASIP}$	$Nb_{decASIP}$	Area (in mm^2)	n	$Nb_{demASIP}$	$Nb_{decASIP}$	Area (in mm^2)	(in %)
1	0.63	53	23	8.75	1	64	18	9.1	3.8
2	0.51	55	19	8.35	1	72	13	9.15	8.7
Chip	-	55	23	8.95	-	72	18	9.9	9.6

Table III
DESIGN TIME: APPLICATION OF THE PROPOSED METHOD ON THE TWO CRITICAL CONFIGURATIONS OF THE CONSIDERED CASE STUDY. $A_{dec} = 0.15mm^2$ AND $A_{dem} = 0.1mm^2$ ($90nm$ CMOS).

Conf.	n	$Nb_{demASIP}$	$Nb_{decASIP}$	Active area (in mm^2)
1	0.63	53	23	8.75
2	0.51	55	19	8.35
3	0.91	29	9	4.25
4	1.1	24	9	3.75

Table IV
RUN TIME : APPLICATION OF THE PROPOSED METHOD.$A_{dec} = 0.15mm^2$ AND $A_{dem} = 0.1mm^2$ ($90nm$ CMOS).

Once platform sizing performed, various required configurations (Table II) can be selected at run-time. The number of ASIPs that the configuration mode requires can be calculated at run-time on a GRC processor (Global Receiver Controller, Fig. 3). As the complexity of the proposed approach is very low (constant time), it allows a very efficient analysis of the best configuration at run-time. This point will be mandatory to meet real-time constraints for future adaptive communication systems. Once a new configuration has been computed the whole platform can be reconfigured.The reconfiguration mechanism itself is out of the scope of this paper but the general schedule can be sketched. The ASIPs that will be used to perform the new configuration can be loaded with appropriate parameters and program whereas the rest of the ASIPs will be idle. Once the right configuration is available the computation starts. Depending on the context the unused ASIPs can be for example powered down to reduce the total power consumption. Table IV shows the number of ASIPs necessary to perform the different configurations using our approach to reduce the active area. Results demonstrate a significant reduction of the active area when configuration corresponding to low requirement are performed. For example, in the case of Conf. 4, only 3.75 mm^2 of the chip have to be activated while 8.75 mm^2 are necessary for the highest requirements corresponding to Conf. 1. Such an approach allows to build optimization strategies for example to tune power consumption or to minimize platform aging.

VI. FINAL DISCUSSION AND CONCLUSIONS

Heterogeneous multiprocessor platforms for iterative demapping and channel decoding provide high performance and high flexibility to perform several configurations. Moreover, they provide promising solutions to be integrated in future flexible baseband receivers. Unfortunately, the first degree of flexibility of a multiprocessor system (i.e. the number of processors used for a given configuration) is currently not taken into account. The platforms are generally statically sized at design-time to reach a given maximum requirement and used, at run-time, without changing the architecture configuration. In this context, the proposed work provides an efficient method for platform sizing which could be used both at design-time and run-time. Depending on the actual requirements, this method allows a dynamic sizing at run-time which optimizes the resources management of the platform.

In this paper we propose an approach for efficient sizing of heterogeneous flexible multiprocessor for iterative demapping and channel decoding. In fact, for a given communication requirement many architecture alternatives exist and selecting the right one at design-time and at run-time is an essential issue. The proposed approach defines the mathematical expressions which exhibit the number of heterogeneous cores and their features. It has been applied on a flexible multi-ASIP hardware platform for iterative demapping and channel decoding. Results analysis demonstrates a reduction of the chip area of 9.6% compared to an approach in which alternative architectures presented in this paper are not explored. Future work targets the model extension with more functionalities, like equalization, and the application of the proposed sizing approach on a dynamic reconfigurable platform and to build optimization strategy to dynamically adapt the configuration.

REFERENCES

[1] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn, "MAGALI: A Network-on-Chip based multi-core system-on-chip for MIMO 4G SDR," in *Proc. of IEEE International Conference on IC Design and Technology (ICICDT)*, 2010, pp. 74 –77.

[2] U. Ramacher, "Software-Defined Radio Prospects for Multistandard Mobile Phones," *Computer*, vol. 40, no. 10, pp. 62 –69, 2007.

[3] J. Declerck, P. Raghavan, F. Naessens, T.V. Aa, L. Hollevoet, A. Dejonghe, and L. Van der Perre, "SDR platform for 802.11n and 3-GPP LTE," in *Proc. of International Conference on Embedded Computer Systems (SAMOS)*, 2010, pp. 318 –323.

[4] C. Brehm, T. Ilnseher, and N. Wehn, "A scalable multi-ASIP architecture for standard compliant trellis decoding," in *International SoC Design Conference (ISOCC)*, 2011, pp. 349 –352.

[5] T. Vogt, C. Neeb, and N. Wehn, "A reconfigurable multi-processor platform for convolutional and turbo decoding," in *Proc. of International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2006, pp. 16–23.

[6] P. Murugappa, Al-Khayat R., A. Baghdadi, and M. Jézéquel, "A Flexible High Throughput Multi-ASIP Architecture for LDPC and Turbo Decoding," in *Proc. of Design, Automation and Test in Europe Conference & Exhibition (DATE)*, 2011.

[7] A. R. Jafri, A. Baghdadi, and M. Jezequel, "FPGA Prototype of Flexible Heterogeneous multi-ASIP NoC-based Unified Turbo Receiver," in *University Booth of the Design, Automation and Test in Europe Conference & Exhibition, DATE'11*, 2011.

[8] S. Haddad, A. Baghdadi, and M. Jézéquel, "Reducing the Number of Iterations in Iterative Demodulation with Turbo Decoding," in *Proc. of International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2011.

[9] O. Muller, A. Baghdadi, and M. Jézéquel, "Parallelism Efficiency in Convolutional Turbo Decoding," *EURASIP Journal on Advances in Signal Processing*, 2010.

[10] A. R. Jafri, A. Baghdadi, and M. Jezequel, "ASIP-Based Universal Demapper for Multiwireless Standards," *IEEE Embedded Systems Letters*, vol. 1, no. 1, pp. 9–13, 2009.