

SOFTWARE RECONFIGURABILITY - ALGORITHM LEVEL APPROACH

R. Hoshyar, S. Gultchev, K.Seo, R. Tafazoli

University of Surrey, UK

Abstract

Multi-mode operation and reconfiguration capability is one of the main requirements of future radio systems. Programmable hardware/software (HW/SW) technologies have enabling role in achieving this requirement. These technologies have their own constraints, and restrictions that must be appropriately accounted when mapping algorithms functionality onto them. Here an algorithmic level reconfigurability approach is proposed that provides HW/SW design flexibility in meeting the required functionalities and multi-mode capabilities within the HW/SW technologies constraints.

I. INTRODUCTION

The vision for future wireless system is a ubiquitous radio providing access at anytime, and anywhere. Different radio access systems of 2G, 3G, broadband short/long range, satellite, and broadcast systems will cooperate to provide seamless user-centric services in highly spectrum efficient way. Future heterogeneous wireless systems should support different requirements from different perspectives of the end user, application provider, service provider, network provider, and manufacturer [1]. Reconfigurable networks and terminals are the best candidates to accommodate the new requirements, and enable inter-working between different radio systems.

Multi-mode capability will be one of the basic features of Software Defined Radio systems ([2], and [3]). However, the user terminals and network equipment reconfiguration will require low complexity, power consumption, and within an acceptable duration of time are needed. Multi-mode reconfigurability imposes new requirements of flexibility to be considered in the design of the architecture, as well as the system algorithms in PHY and upper layers. Adopting suitable approaches in the introduction of algorithms reconfigurable to different modes with efficient complexity/performance trade off will have a crucial impact on the optimisation of processing resources. It will also impose some overhead costs on processing power, power consumption, latency, and software download volume. Effort should be made to minimize these costs while providing required reconfigurability, and future upgradability.

HW/SW design and implementation should comply with different multi-mode and reconfigurability requirements, and also achieve fine trade offs based on the state of the art technologies. In this regard reconfigurability analysis at algorithm level will be

useful. Commonalities and similarities among algorithms should be explored, and based on them appropriate HW/SW implementation solutions can be provided. Algorithm level reconfigurability analysis should provide structured implementation-independent information that can be easily used in HW/SW partitioning and final design decision. This paper addresses reconfigurability at algorithm level and proposes a classic method for provision and optimisation of reconfigurable algorithms.

II. RECONFIGURABLE PLANE AND ALGORITHMIC LEVEL MODELLING

Currently, the main attention, when investigating software radio implementation and architectures, has been on the development of a top level reconfigurable system with future provision of open programmable and reconfigurable radio platforms (i.e. based on SDR technology and defined using object oriented design methodologies as described in [4]).

However, the question of applying some kind of a methodical approach when designing such reconfigurable nodes stays unresolved. How to design and implement a reconfigurable system in the most efficient way, and with fewer resources that will use the least effort, less power consumption and processing computation etc?

Very few attempts are known for investigating an approach towards reconfigurable systems' design principles that will provide software design methodology (guidelines) for reconfigurable systems.

However, with ability and provision of reconfiguration software and combining various (software) implementations, from different sources, within one radio configuration requires the introduction of novel management and control mechanisms that will allow the enforcement of defined rules (mechanisms, principles) which regulate the configuration of any software radio platform and ensure the compliance to given radio standards. These rules have to cover compatibility of reconfigurable equipment, a set of laws when performing a reconfiguration and specific reconfiguration constraints for the terminal platform.

The investigation and development of such an approach will lead to creating a systematic understanding of reconfiguration control and management processes when performing the reconfiguration of the software radio equipment in the software module level. This ultimately involves an additional reconfiguration control and management plane that will support these processes

and will provide a coherent environment for those control and management entities.

An example of such a reconfiguration architecture including distributed reconfiguration management scheme that implements the shared responsibility within the distributed reconfiguration management architecture (i.e. the RMA) [5].

The support, provided by this reconfiguration plane, the RMA, and particularly by its Radio Module Controllers (RMCs) [6], allows the intrinsic incorporation of those rules when performing reconfiguration of the software modules.

The involvement of a reconfiguration plane gives the first step towards more generic approach when performing a reconfiguration. The next step is to provide a more compact and generic representation when designing and installing those reconfigurable software modules. It also has to provide a lower level of optimisation, compatibility, portability and efficiency down to the simplest data type's implementation and design. This kind of a methodology can easily be achieved using an algorithmic level approach when modelling and reconfiguring software modules.

III. THREE PHASED RECONFIGURABILITY ANALYSIS

Functional grouping proposed in the IST project MuMoR (Multi Mode Radio) [7] provides a good starting point for reconfigurability analysis of the baseband algorithms. Baseband architectures of the different modes are compared to each other, and blocks of the different modes with similar functionality are grouped together. The approach of functional grouping is general and can be applied to upper layer algorithms as well. Modules in the same functional group can be compared to find commonality, and similarity among them. This will help in devising appropriate methods to combine the blocks, if applicable, resulting in one hybrid block with the functional capability of all the merged blocks, with similar processing resource demand as single blocks, and less programmability restrictions than the unmerged blocks together.

In this respect, the following method of "Partitioning, Classification and Parameterisation" will be presented. These three phases of "Partitioning", "Classification", and "Parameterisation" provide a general approach for developing complexity-optimised reconfigurable algorithms.

A. Partitioning

The 'Partitioning' of the algorithms will help to find common parts among them, and reduce some of the complexity factors by sharing the common parts. The reduced complexity factor has general meaning at this level and depending on HW/SW implementation it will be translated to reduced HW area and number of gates, smaller required HW reconfiguration area, reduced SW program size, or any other HW/SW implementation cost saving that can be achieved through sharing of the

identified common parts. The optimum level of partitioning can be adjusted based on the trade off between the saved complexity factor due to the common parts, and the overhead on data and program flow control required for sharing the common parts, and also the necessary reconfigurability control. However, to provide simplicity and generality, it is better to apply the same level of partitioning to all functional groups. After partitioning, component parts that are common, and usable in different algorithms can be identified. We call these general components Algorithm Level Primitives (ALP).

B. Classification

These ALPs can be classified, so that all ALPs belonging to the same class can be represented and implemented by a general parameterisable ALP. This implies that the reconfigurability is realisable through similar components that belong to the same class of ALPs.

C. Parameterisation

By parameterisation a general ALP component can be shared amongst the algorithms. The two phases of "classification" and "parameterisation" are closely related to each other, and classes of ALPs should be set in a way that an appropriate parameterisation could be developed for each class. Parameterisation may affect functionality of ALP, its interface, or both. Care should be made in the selection of parameterisable ALPs, as in HW interface change cannot be easily accommodated, while in SW it does not have any considerable restriction.

Three phases of "Partitioning", "Classification", and "Parameterisation" provide a general approach for developing complexity optimised reconfigurable algorithms. Figure 1 illustrates how two algorithms Alg1 and Alg2 can be realized through a number of parameterisable ALPs. Realization of these algorithms through the ALPs will take slightly more processing resources, as they require a parameterisation scheme (shown with grey line in the figure), and also the ALPs themselves will take more processing resource than the components precisely tailored for the specific requirements of the algorithm. However the ALP realization introduces flexibility into the algorithm that can be exploited for both intra- and inter- algorithm reconfigurability. On the other hand, partitioning into ALPs provides better possibility of comparing the two algorithms, and finding commonality and similarities between them.

In this example ALPs 1, 2, and 3 although require different parameter settings for the two algorithms, they can be shared efficiently between the two algorithms by introducing a reconfigurability control block, as shown in Figure 2. This block is responsible for providing necessary parameter settings, scheduling, and interconnection of the ALPs. Parameter p being fed to Alg12 may have more than two possible values, in this case besides reconfigurability of Alg12 to Alg1, and

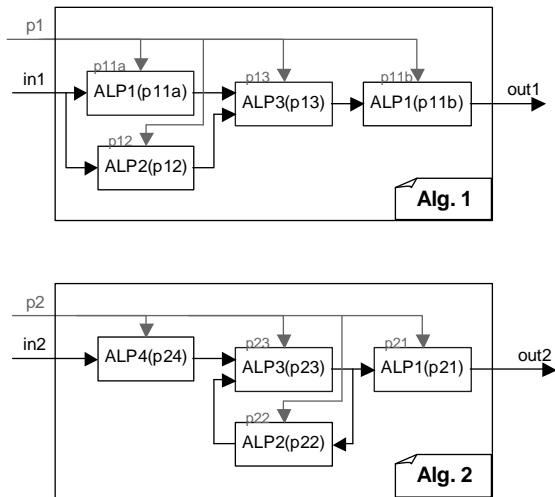


Figure 1. Partitioning, and realization of the two algorithms, Alg1, and Alg2 through parameterisable ALPs.

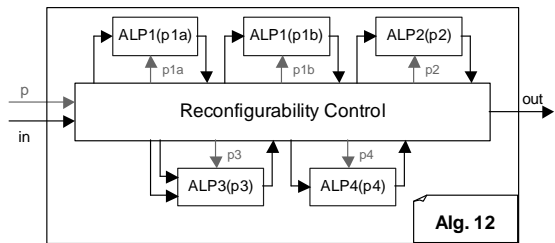


Figure 2. Reconfigurable algorithm Alg12 providing functionality of both Alg1 and Alg2 algorithms. This algorithm has been realized through parameterisable ALPs, and introducing a reconfigurability control block.

Alg2, to some extent it is also parameterisable, i.e. it will have the two features of reconfigurability, and parameterisability.

In the implementation level the interconnection of the ALPs will translate to wiring and connecting of logic blocks in HW, and data flow control in SW. In hardware realization of Alg12 depending on reconfiguration to Alg1, or Alg2 some of the ALPs can be switched off. Also one ALP1 can be used and shared for the two ALP1 functionalities within Alg1 through SW assisted scheduling. As another option Alg1 can be FPGA configured through four of its ALPs, and for reconfiguration to Alg2 only one of the ALPs needs to be reconfigured to ALP4 with extra or less number of gates depending on the size of the two ALPs. In software realization only four ALPs, i.e. 1, 2, 3, and 4, need to be stored in the program memory. Main program body of the Alg12 will include reconfiguration control part for appropriate data and program flow control, and some of the ALPs will be called once or more inside this algorithm. All of the possible HW or SW realizations discussed above are related to the HW/SW implementation phase. The structured information provided through ALP description of Alg12 will help to adopt appropriate HW/SW solution. Also, a

frequent ALP resolved among many algorithms suggests exploiting of a corresponding accelerator in its HW/SW realization, as considered in [8].

D. Multi-Dimensional Complexity Constraint Space

Complexity factors of HW/SW components, like processing speed, number of gates, interconnect capacity, power consumption, and I/O capability for HW, and similarly number of instructions per second, number of operations per second, power consumption, memory addressing, and so on, sketch a multidimensional complexity constraint space. The complexity demands of the required functionalities have to fit within this space. The granularity achieved by partitioning of algorithms into ALPs, and the explored commons, and similarities will provide implementation design flexibility in fitting multi-mode and reconfigurable functionalities within complexity constraint space. As an example, Figure 3 illustrates how the multi-mode Alg3, consisting of Alg3a and Alg3b, has been accommodated by the provided granularity and sharing of some ALPs. This also shows the trade off between the two complexity factors $c1$, and $c2$ to fit the demands within the constrain space. The simple switching between Alg3a, and Alg3b will result in considerable increase in complexity factor $c1$ that is not affordable. This figure has similarity in meeting implementation constraints with figures 7, and 8 of reference [9].

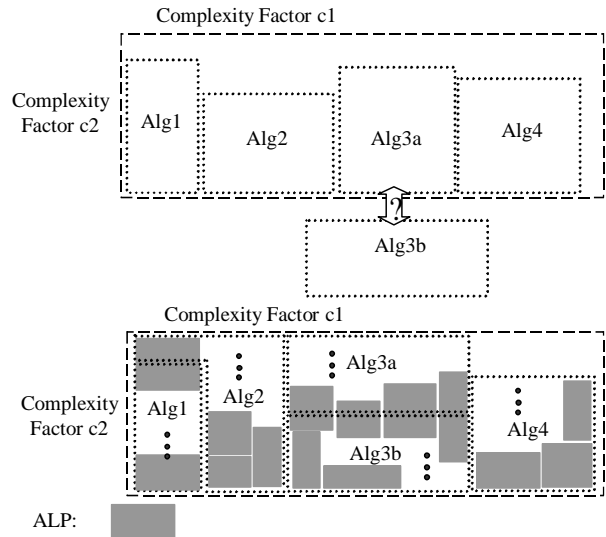


Figure 3. Sharing ALPs to fit the reconfigurable system within the multi-dimensional space of complexity factors' constraints.

IV. EXAMPLES ON RECONFIGURABILITY ANALYSIS ON UMTS FDD/TDD

UMTS 3G radio system is based on WCDMA radio access technique. This system can work on one of the two duplex modes of FDD and TDD. HSDPA (High Speed Downlink Packet Access) has been added to this system to support high data rates up to 10 Mbps in

downlink direction. UMTS mobile terminals should be able to work in the two FDD/TDD modes, whenever and wherever required, and also capable of HSDPA operation. The IST MuMoR project addresses multi-mode and reconfigurability capability of 3G terminals in both RF and baseband parts. In the following two examples of the application of the proposed three-phase reconfigurability analysis on UMTS will be presented.

IV.A. Reconfigurability Analysis of Spreading and Scrambling

In spreading/scrambling of the uplink transmitter there are some similarities in the way that the FDD and TDD signal is being processed. Figure 4 shows UMTS FDD and TDD spreading/scrambling processing, respectively in the uplink transmitter and how partitioning of parameterisable ALPs can be done for each mode.

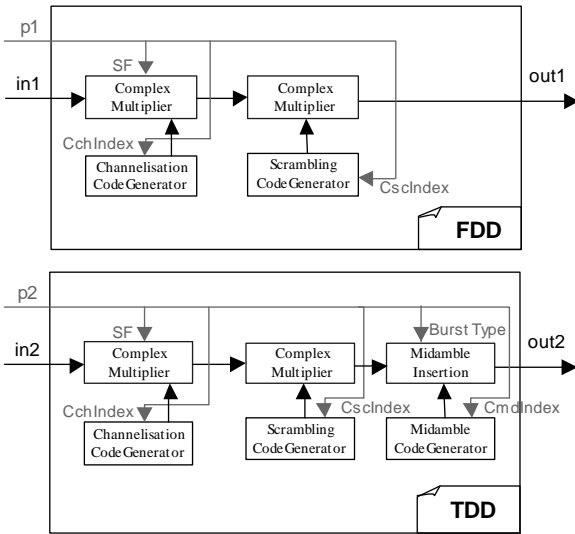


Figure 4. Partitioning, and realization of UMTS FDD and TDD in uplink transmitter spreading/scrambling processing through parameterisable ALPs.

Spreading is almost the same for both FDD and TDD apart from the differences of the channelisation code phase between them which are caused by the channelisation code specific multiplier for the TDD mode. So this functionality for the FDD and TDD can be easily merged into one with some reconfiguration control signals. Scrambling is also almost the same for both FDD and TDD except the different scrambling codes between them to be used. So this functionality for the FDD and TDD can be easily merged into one with some reconfiguration control signals. On the other hand, midamble insertion is unique functionality to TDD. So it should be separated for TDD mode only.

Figure 5 shows how this reconfigurability analysis can be realised by parameterisable ALPs in order to merge these two algorithms into one using reconfigurability control. From this merged implementation, we can get some reduced size of hardware rather than implementing them separately even though we'll have to pay some overhead for reconfigurability control.

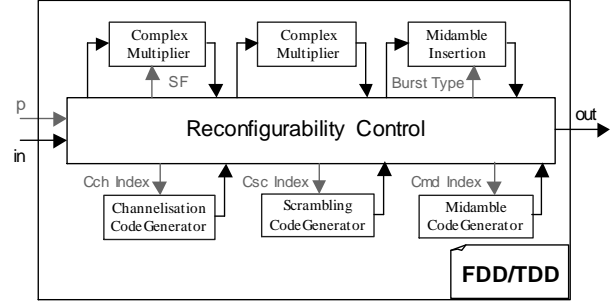


Figure 5. Reconfigurable FDD/TDD uplink transmitter spreading/scrambling providing the functionality of both FDD and TDD algorithms. This algorithm is realized through parameterisable ALPs, and introducing a reconfigurability control block.

IV.B. Reconfigurability Analysis on Soft Bit Demapper

Here as another example we apply the three-phase approach to the soft bit Demapper block to be used in UMTS FDD/TDD. QPSK and 16QAM modulations have been specified for both FDD, and TDD modes. TDD mode modulations are $\pi/4$ anti clock wise rotated and $1/\sqrt{2}$ scaled of FDD mode ones. 16QAM is specified for downlink HSDPA channel in both modes. Applying LLR computing at the output of the rake receiver following approximate equations will be resulted:

UMTS/FDD QPSK modulation:

$$\begin{aligned} llr(c^{(1)}; \bar{z}, QPSK) &= \text{Re}\{\bar{z}\}, \\ llr(c^{(2)}; \bar{z}, QPSK) &= \text{Im}\{\bar{z}\}. \end{aligned} \quad (1)$$

Where $\bar{z} = \sum_{l=1}^L (4E_s/N_0)_l \hat{\alpha}_l^* \bar{y}_l$ is a version of MRC output accounting for the different SNRs of rake fingers. $L_{Cl} = (4E_s/N_0)_l$ is the channel reliability of the l -th finger.

UMTS/FDD 16QAM modulation:

$$\begin{aligned} llr(c^{(1)}; \bar{z}, \hat{\rho}) &= LLR_4AM1(\text{Re}\{\bar{z}\}, \hat{\rho}), \\ llr(c^{(2)}; \bar{z}, \hat{\rho}) &= LLR_4AM1(\text{Im}\{\bar{z}\}, \hat{\rho}), \\ llr(c^{(3)}; \bar{z}, \hat{\rho}) &= LLR_4AM2(\text{Re}\{\bar{z}\}, \hat{\rho}), \\ llr(c^{(4)}; \bar{z}, \hat{\rho}) &= LLR_4AM2(\text{Im}\{\bar{z}\}, \hat{\rho}), \end{aligned} \quad (2)$$

where

$$\begin{aligned} LLR_4AM1(g, \hat{\rho}) &= \max_{\bar{x} \in A(1,0)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{\rho} \bar{x}|^2 \right) - \\ & \max_{\bar{x} \in A(1,1)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{\rho} \bar{x}|^2 \right) \\ LLR_4AM2(g, \hat{\rho}) &= \max_{\bar{x} \in A(2,0)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{\rho} \bar{x}|^2 \right) - \\ & \max_{\bar{x} \in A(2,1)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{\rho} \bar{x}|^2 \right) \end{aligned} \quad (3)$$

The output of the l -th finger is y_l , and $\hat{\alpha}_l$ is the corresponding estimated channel gain and $\hat{\rho} = \sum_{l=1}^L |\hat{\alpha}_l|^2$. Over-bar symbols are corresponding normalized values

with $\bar{x} = x/\sqrt{E_s}$, and $\bar{y}_l = y_l/\sqrt{E_s}$.

$\max_i^*(a_i) = \ln \sum_i \exp(a_i)$, and can be expressed in terms of normal max operation plus a correction term: $\max^*(a_1, a_2) = \max(a_1, a_2) + \ln(1 + e^{-|a_1 - a_2|})$. Subsets $A(j, c)$ are shown in Figure 6.

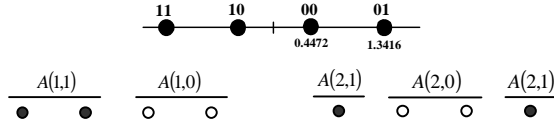


Figure 6. 4AM mapping constellation, and its corresponding subsets to be used for bits LLR computation of UMTS/FDD 16QAM soft de-mapping.

Modulation mapping used for UMTS/TDD is $\pi/4$ anti-clock wise rotation, and $1/\sqrt{2}$ scaled of the mapping specified for UMTS/FDD. Therefore, by appropriate rotation and scaling the same de-mapping will be applicable to both. After rotation and scaling for TDD mode only, soft bit values will be computed based on equation (1) for QPSK, and equations (2), and (3) for 16QAM mapping. Now we apply the three phases of partitioning, classification, and parameterisation. QPSK de-mapping is very simple, and none of the three phases are applicable to it. Partitioning level of 16QAM de-mapping can be kept at the level of LLR_4AM1, and LLR_4AM2 functions, or further be proceeded to \max^* , and squaring operators level. Considering that \max^* operator is the main ingredient in Soft Input Soft Output (SISO) algorithm of iterative decoding, it can be shared between 16QAM de-mapping and SISO algorithm. However for re-configurability of de-mapper alone, partitioning down to LLR_4AM1, and LLR_4AM2 functions will suffice. Further inspecting equation (3), it is observed that both LLR_4AM1, and LLR_4AM2 functions can be classified in the same class, and can be expressed by the following parameterisable LLR_4AM function:

$$LLR_4AM(g, \hat{p}; A(j,0), A(j,1)) = \max_{\bar{x} \in A(j,0)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{p}\bar{x}|^2 \right), \quad (4)$$

$$\max_{\bar{x} \in A(j,1)}^* \left(-(E_s/N_0)_{MRC} |g - \hat{p}\bar{x}|^2 \right)$$

where subsets $A(j,0)$, and $A(j,1)$ are its parameters. One LLR_4AM function can be time shared and used four times in soft bit computation of equation (2). In this example, because of the simplicity of QPSK de-mapping and minor difference between FDD and TDD mode, the proposed three-phase approach is not necessary, for none of the inter and intra mode re-configurability. Nevertheless this approach enables us to exploit the similarity, and commonality within a single algorithm, and achieve a trade off between volume and speed of processing by time-sharing of the common and similar parts.

V. CONCLUSION

Algorithmic level analysis for multi-mode reconfigurability has been presented. Reconfigurability analysis at algorithm level is required to explore high-level concepts on commonalities and similarities. In this regard, this paper proposes a reconfigurability analysis based on three phases of “Partitioning”, “Classification”, and “Partitioning”. It provides HW/SW design phase a structured implementation-independent view on how to achieve multi-mode flexibility requirements within the implementation constraints. The granularity achieved by the proposed method, creates HW/SW design flexibility in making desired trade offs between different complexity factors, and fitting the required features, and functionalities within multi-dimensional complexity constraint space.

VI. ACKNOWLEDGMENTS

The work presented in this paper is based on the IST project MUMOR, [7], whose funding and support, is gratefully acknowledged. For more detailed technical information, please refer to [7].

VII. REFERENCES

- [1] N.J. Drew, M.M. Dillinger, “Evolution Toward Reconfigurable User Equipment,” *IEEE Commun. Mag.*, Vol. 39, No. 2, pp. 158-164, Feb 2001.
- [2] J. Mitola, “The Software Radio Architecture,” *IEEE Commun. Mag.*, Vol. 33, No. 5, pp. 26–38, May 1995.
- [3] W. Tuttlebee, *Software Defined Radio, Enabling Technologies*, J. Wiley & Sons, Oct. 2000.
- [4] Mitola J, “Software Radio Architecture—Object Oriented Approaches to Wireless Systems Engineering”, Wiley-Interscience, ISBN 0-471-38492-5, 2000.
- [5] Gultchev S, Moessner K, Tafazolli R, “Controlling Reconfiguration”, IEE 3G2002, London, UK, 8-10 May 2002.
- [6] Gultchev S, Moessner K, Tafazolli R, “Management and Control of Reconfiguration Procedures in Software Radio Terminals”, WSR2002, Karlsruhe, Germany, 20-21 March 2002.
- [7] <http://www.mumor.org>,
- [8] J. Brakensiek, B. Oelkrug, M. Bucker, D. Uffmann, A. Droge, M. Darianian, M. Otte, “Software Radio Approach for Re-Configurable Multi-Standard Radios,” The 13th IEEE Intern. Symp. PIMRC, Vol. 1, pp. 110-114, 2002.
- [9] J. Mitola, III, “Software Radio Architecture: A Mathematical Perspective,” *IEEE J. Select. Areas Commun.*, Vol.: 17, No. 4, pp. 514-538, Apr 1999.