Requirement-driven Model-based Development Methodology Applied to the Design of a Real-time MEG Data Processing Unit

Tao Chen[1,2], Sergey Suslov[1], Michael Schiek[1], Jürgen Dammers[2], N. Jon Shah[2,3,4], Stefan van Waasen[1,5]

[1]Central Institute of Electronic Systems (ZEA-2), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany
[2]Institute of Neuroscience and Medicine (INM-4), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany
[3]Department of Neurology, Faculty of Medicine, RWTH Aachen University, 52074 Aachen, Germany
[4]JARA–BRAIN–Translational Medicine, RWTH Aachen University, 52074 Aachen, Germany
[5]Communication Systems, Faculty of Engineering, University of Duisburg-Essen, 47057 Duisburg, Germany

Abstract — The paper describes a multidisciplinary work that uses a model-based systems engineering method for developing real-time magnetoencephalography (MEG) signal processing.

We introduce a requirement-driven, model-based development methodology (RDD & MBD) to provide a high-level environment and efficiently handle the complexity of computation and control systems. The proposed development methodology focuses on the use of System Modeling Language (SysML) to define high-level model-based design descriptions for later implementation in heterogeneous hardware/software systems. The proposed approach was applied to the implementation of a real-time artifact rejection unit in MEG signal processing and demonstrated high efficiency in designing complex high-performance embedded systems.

In MEG signal processing, biological artifacts in particular have a signal strength that overtop the signal of interest by orders of magnitude and must be removed from the measurement to achieve high-quality source reconstructions with minimal error contributions. However, many existing brain-computer interface (BCI) studies overlook real-time artifact removal because of the demanding computational process. In this work, an automated real-time artifact rejection method is introduced, which is based on the recently presented method "ocular and cardiac artifact rejection for real-time analysis in MEG" (OCARTA). The method has been implemented using the RDD & MBD approach and successfully verified on a Virtex-6 field programmable gate array (FPGA).

Keywords **- MBSE · SysML · real-time systems · MEG ·artifact rejection · Neurofeedback**

I Introduction

In recent years, semiconductor technology has advanced continually; the scale of heterogeneous systems has been increasing dramatically and very different disciplines are involved [1]. However, an interconnected System of Systems (SoS) as a whole is much greater than the sum of the individual subsystems [2]. The interconnectivity and data transfer among the constituent systems can be extremely complex and hard to control [3]. Systems engineering (SE) is an interdisciplinary approach for handling the complexity of large and intricate designs by providing a development environment at a much higher level of abstraction. Systems engineering methods utilize a large degree of abstraction to visualize a system and relevant parts, which helps a developer focus on the aims of a system rather than the implementation details. It enables the change from technical to strategic development decisions because the cost of principal mistakes increases with design complexity [4]. Systems engineering is becoming increasingly widely applied over the years, and numerous modeling tools and methods are proposed to address the increasing complexity of today's heterogeneous systems [3]. Model-based systems engineering (MBSE) has the advantages of low cost, time saving, and accuracy compared with traditional methodologies in embedded system design.

We proposed a requirement-driven, model-based development methodology (RDD & MBD) as a further development of comprehensive methodology for functional design in digital systems [4]. It has been elaborated in our institute to address the problems of heterogeneous computation and control system design using System Modeling Language (SysML) [5]. The methodology is exploited in the design and implementation of a real-time signal pre-processing system for magnetoencephalography (MEG). This work is part of the development of the MEG-2.0 Real-Time project (MEG-RT 2.0) at Jülich Research Center (Forschungszentrum Jülich GmbH, Germany) aiming at developing a MEG real-time signal processing device as an add-on to the 248-channel whole-head magnetometer MEG system (Magnes 3600WH) from 4D-Neuroimaging. Applying system modeling to our design allowed comprehensive and thorough evaluation of actual design requirements and forward planning of possible technical solutions to find an optimum solution in advance.

MEG is a non-invasive neuroimaging technique that investigates brain activities with high temporal resolution [6]. In modern MEG systems, an array of hundreds of low-temperature magnetometers is used to record components of the magnetic field produced by post-synaptic neuronal activities. The magnitude of the magnetic field components is in the range of a few tens to hundreds of femtotesla, while some noise and biological artifacts, such as ocular and cardiac activity, can be ten to hundred times larger [7]. These artifacts must be removed from the measured signals as they can lead to source reconstruction errors [7]. However, artifact rejection is computationally extremely demanding [8].

In MEG studies, real-time MEG data analysis has become a topic of high interest recently and requires automatized real-time artifact removal [8],[9],[10]. One of the first real-time MEG source analyses was demonstrated in [11], but only including cardiac artifact rejection and capable for real-time processing of a reduced number of channels. In other studies, real-time MEG analysis is limited to certain frequency bands only [12],[13],[14],[15].

The system modeling work of MEG signal pre-processing was recently published in [3]. In the present paper, we optimized the previous models and expanded the system engineering by implementing a System-on-Chip (SoC) capable of performing real-time artifact rejection.

In the present work, based on the performed system modeling and analysis, we planned the target design to be developed and implemented on a single workstation platform combining field programmable gate array (FPGA) and graphics processing unit (GPU) co-processing boards containing a software framework for real-time MEG data analysis.

The paper is structured as follows: Section II describes the systems engineering process of the target design together with the performance estimation. Implementation of MEG signal pre-processing is discussed in Section III. Section IV is devoted to results and discussion. Finally, the conclusion is given in section V.

II Model-Driven System Design

We have introduced a RDD & MBD for real-time computation systems based on vendor-neutral specifications. The methodology, as further development of a comprehensive methodology for functional design in digital systems presented in [5], has been elaborated in ZEA-2 to address the problems of heterogeneous computation and control system design using SysML [4] and practiced in this work. The development workflow is plotted in Fig. 2.1. It can be roughly divided into three steps: requirements formalization, functional modeling and structure allocation and partition.

Requirements formalization begins with elicitating the expectations and constraints of different stakeholders. The acquired expectations and constraints are analyzed to obtain a formalized set of stakeholder requirements, which conveys a preliminary description of the system and shows its boundaries from the view of stakeholders. Then, the stakeholder requirements are further decomposed into system requirements by adding abstract functional specifications, which are grouped into three categories, i.e. *Qualitative*, *Quantifiable*, and *Major operations*, as shown in Fig. 2.1. The qualitative category mainly includes immeasurable requirements such as safety, continuous observability, and ease of operation. The quantifiable category captures measurable parameters and constraints. With the refinement of system models, some qualitative requirements may become quantifiable. The major operations category depicts the functionality of the system without specifying implementation details. Finally, feature derivation and decomposition are performed on system requirements to produce technical requirements. The formalized system requirements are further refined with sub-categories and requirement diagrams by adding implementation specifics. Full or near full coverage can be achieved by grouping and generalizing the requirements, and their subsequent classification and categorization. Requirements formalization is important both for principle decision-making in the development process as well as for building a validation plan for feature checking, and in a certain amount for verification: an implementation at any abstraction level must match the requirements of corresponding levels.

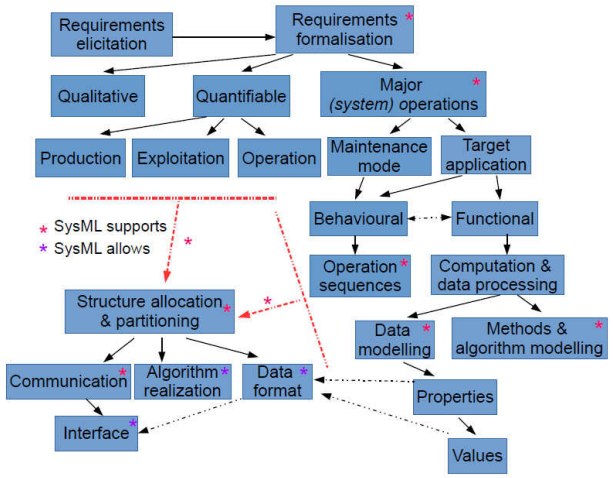

Figure 2.1. Model-based system engineering for real-time systems (MBSE-RT) workflow ([5])

In the second step of the workflow, **system modeling** is performed based on the specified system requirements and technical requirements. The corresponding system dynamic (behavior) and block structure are modeled in parallel, including *Data modeling*, i.e., data structure, data dependency, data exchange flows, and *Methods and algorithm modeling*.

In the final step (**structure allocation and partition**), quantifiable requirements are imposed in the form of mathematical constraints to the relevant structural model elements to properly allocate them to relevant computation structures with predictable implementation parameters, such as performance, resource budget, and power consumption. Thus, the functions are partitioned and allocated to the corresponding computation hardware. This process is iterative, which indicates that the procedure may be repeated for each sub-system performing refinement of the system on different abstraction levels [4],[5].

2.1 Requirements formalization

The most important part of system modeling is defining requirements, because requirements are the foundation of system

design, and the aim of system design is to satisfy the requirements [16].

2.1.1 State the problem

MEG enables studying the spatio-temporal dynamics of the electrophysiological interaction in the human brain non-invasively. With MEG, the brain in action can be investigated at very high temporal and relatively high spatial resolution depending on several factors (e.g., the strength and depth of the active sources) [17].

To push MEG further towards a clinical diagnosis tool (e.g., pre-surgery planning, early diagnosis, localization of epileptogenic zones, rehabilitation of stroke), results must be available during or soon after the investigation. The dynamics of the brain are extremely rich and it is possible to observe major effects by applying stimuli with respect to the state of the neuronal dynamic system [18], which is termed neuronal feedback. The ability to control in real-time provides new insights into the brain function of both normal and impaired brain processes. As indicated in [19], a time delay of >200 ms between brain activity, e.g., movement intention and the feedback output by the device, is clearly noticeable. In real-time computing, processing steps, including data acquisition, filtering, and artifact rejection, are designed in a pipelined structure. The latency is caused by the step with the largest delays. Therefore, signal processing systems with an overall constant delay of less than 200 ms are considered as real-time processing for neuroprosthetic control [19]. Brain–computer interfaces (BCI) utilizing neurofeedback stimulation hold great potential for neuroscience and therapy in neurology, but require adequate real-time analysis of ongoing brain responses. Therefore, the MEG-RT 2.0 project was proposed to address advanced MEG signal processing, including current source reconstruction, which will enable real-time neurofeedback applications.

2.1.2 Stakeholder requirements definition

It is important to identify the environment with which the system interacts. Use case diagrams describe the high-level functionality of the system and model interactions between users and the system. The use case diagram shown in Fig. 2.2 was created to define the boundaries of MEG-RT 2.0 and the operation contexts to which it is subjected. MEG-RT 2.0 may interact directly or indirectly with 4 actors, as shown in Fig. 2.2, an *Operator* (a stick figure sign) who operates the system, an *MEG Database* (a rectangle with the keyword «actor»), an *MEG Acquisition System* (DAS), and a *Subject* (Patient) who interacts indirectly with the system through the *MEG Acquisition System* (DAS). MEG operators and subjects are identified as main stakeholders. Other indirectly related stakeholders include system developers, hospital staffs, etc.

To improve the feasibility of MEG as a clinical diagnostic tool, the major expectations of stakeholders are to process MEG signals in real-time, thus enabling, for example, neurofeedback applications. Therefore, the MEG-RT 2.0 design, as an add-on to an existing data acquisition system (*DAS*), shall be capable of *Perform RT MEG Preprocessing*, *Perform Offline MEG Study* and *3D Source Localization*, as shown in Fig. 2.2. Additionally, is further extended with functions including record online data, MEG measurements filtering, real-time and automated artifact rejection, etc.

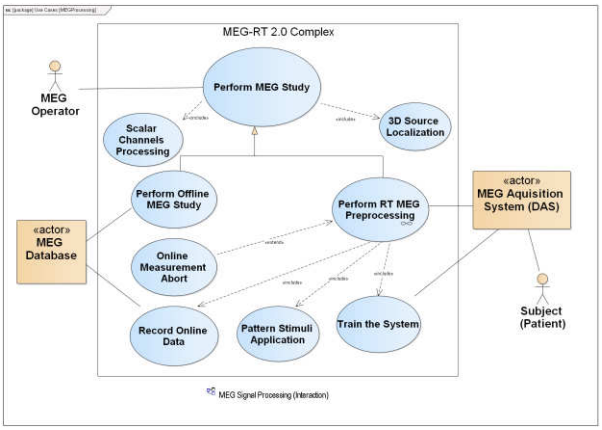

Figure 2.2. Use case diagram of MEG signal pre-processing

We attempt to identify all MEG equipment stakeholders and collect as many requirements as possible. The expressed stakeholder needs and expectations are far from formalness and maturity. They need to be refined into stakeholder requirements. Tab. 2.1 gives an idea of the aspects for requirements elicitation. The given approach to requirement elicitation is based on distinguishing phases of a product life cycle and categorizing the questions that can be stated in each phase to computing and controlling systems. Fig. 2.3 shows the overall structure of decomposing the formalized stakeholder requirements into system requirements and then further into technical ones by adding more details. Consequently, the decomposition process enables requirement traceability through different layers.

Table 2.1: Life-cycle requirement taxonomy ([5])

Life-cycle stage	Requirement
Concept	Business goals, operational concept
Design	Development cost, development time, system configurability, system modularity, 3rd-party compatibility, legislation
Manufacture	Manufacture cost, manufacture time, production volume, legislation
Deployment	Ecosystem, deployment cost, deployment time, construction
Operating	Functional, reliability, exploitational
Disposal	Disposal cost, dismounting time, component reuse, component recycle

2.1.3 System requirements specification

System requirements, in the conceptual sense, form the elementary basis in system development and determine the capability, function, or performance the system has to achieve [1],[20]. The system requirements for MEG signal processing need to be captured and traced in the system model. The step described in section 2.1.2 contains a set of stakeholder requirements that result in more specific system ones. First, the more refined system requirements are unstructured and put in a folder named *Unqualified*. Then, we formalize the system requirements in three categories: *Functional*, *Qualitative*, and *Quantifiable*. Fig. 2.3 shows the top-level package structure that accommodates these requirement categories. Functional requirements represent specific capabilities and major operations in the design of MEG-RT 2.0, including maintenance mode and major mode. We focus on major mode, including data decomposition and artifacts rejection. Qualitative requirements refer to unmeasurable stakeholder needs, including operator interface in Python, real-time operation, etc. Quantifiable requirements address properties, which can be measured and quantified, including the number of channels, the sampling rate, etc.

2.1.4 Technical requirements specification

The system requirements are elaborated further by the breakdown into technical requirements with implementation details.

The top-level system model for MEG-2.0 RT was developed in the present study. As the approach steers down to the modeling and design of lower levels, this study only concentrates on the design of real-time artifact rejection using OCARTA (ocular and cardiac artifact rejection for real-time analysis in MEG) in the MEG-RT 2.0 workflow. This is because previous studies either neglected or simplified artifact rejection [9], [21].

This section describes the breakdown of the system requirements into a cohesive set of technical requirements [22]. All essential requirements for MEG signal processing are clearly described so that they can be satisfied and traced by the system design. Fig. 2.4 depicts requirement decomposition and the initial derivation of requirements in MEG signal pre-processing. The stakeholder requirement *MEGPreprocessing* is separated into more detailed system requirements. The technical requirements of data processing methods are derived from an analysis of each functional requirement of the system level. In addition to relationships among requirements, traceability from system requirements to behavioral and structural elements is also established.

2.2 System structure modeling

System static structure models the blocks of which the system is composed. Structural modeling includes four steps: *Model the Problem Domain*, *Define Blocks*, *Allocate Requirements to Blocks* and *Define Ports* [16].

2.2.1 Domain modeling

Domain modeling describes real-world (problem domain) entities and the relationships between them, which models the system in the context of its environment. Fig. 2.5 shows a domain model for MEG signal processing. The operation domain model defines a set of abstractions based on real-world elements external to the system, including environment noise, clinical diagnosis and neurofeedback [16].

The problem domain model is intended to depict the "system" in which *MEG-RT 2.0* operates. Fig. 2.5 shows the systems that interact together with *MEG-RT 2.0* [16]. The *OperationDomain* is defined as a system containing other subsystems, including the humans (*Operator*, *Subject*, etc.) who directly or indirectly interact with the system, external systems (*DAS*, *Filter*, etc.) that communicate with the system, and environmental elements (*EnvironmentNoise*, etc.) that could impact the system. The operational domain model efficiently provides an overview of the system and how it operates. Details of the *OperationDomain* system are elaborated in the *OperationDomain* BDD shown in Fig. 2.5.

2.2.2 System architecture of MEG signal pre-processing

The SysML block definition diagram (BDD) is used to display various kinds of elements (blocks, actors, value types, interfaces, etc.) and their structural relationships (associations, generalizations, dependencies) about the MEG pre-processing unit [23]. As an example, Fig. 2.6 shows the child block definition diagram of *MEGPreprocessing* block in Fig. 2.5 in order to highlight the details. The composite association for the *MEG-RT 2.0* block is shown: the *MEGSignalPreprocessing* block is composed of four main sub-blocks: *InitialTraining*, *DataDecomposition*, *ArtifactIdentification*, and *DataCleaning*. Additionally, value properties (e.g. *meg_raw: MEGArray*) are listed in the second compartment of a block, which hold parameter values in MEG signal processing. The flow ports are listed in the third compartment of the *MEGSignalPreprocessing* block. The flow ports model the data that flow in and out of the *MEGSignalPreprocessing* block, which are detailed in the corresponding IBD. In the third compartment of part properties are operations of the relevant block, which represent behaviors the blocks perform.

In addition to relationships among requirements, traceability from data requirements to *Value Type* blocks is also established.

2.2.3 System internal structure

The internal block diagram (IBD) is a complementary view to the BDD. Fig. 2.7 shows the IBD of *MEGSignalPreprocessing* block in Fig. 2.6. It specifies the internal structure of *MEGSignalPreprocessing* with respect to how its parts are interconnected [22],[23]. It conveys the part properties of the *MEGSignalPreprocessing* block and the related connectors. Moreover, it is allowed to specify multiple levels of nested parts in a single view by showing part symbols within part symbols, e.g., the *it:InitialTraining* part property [22],[20].

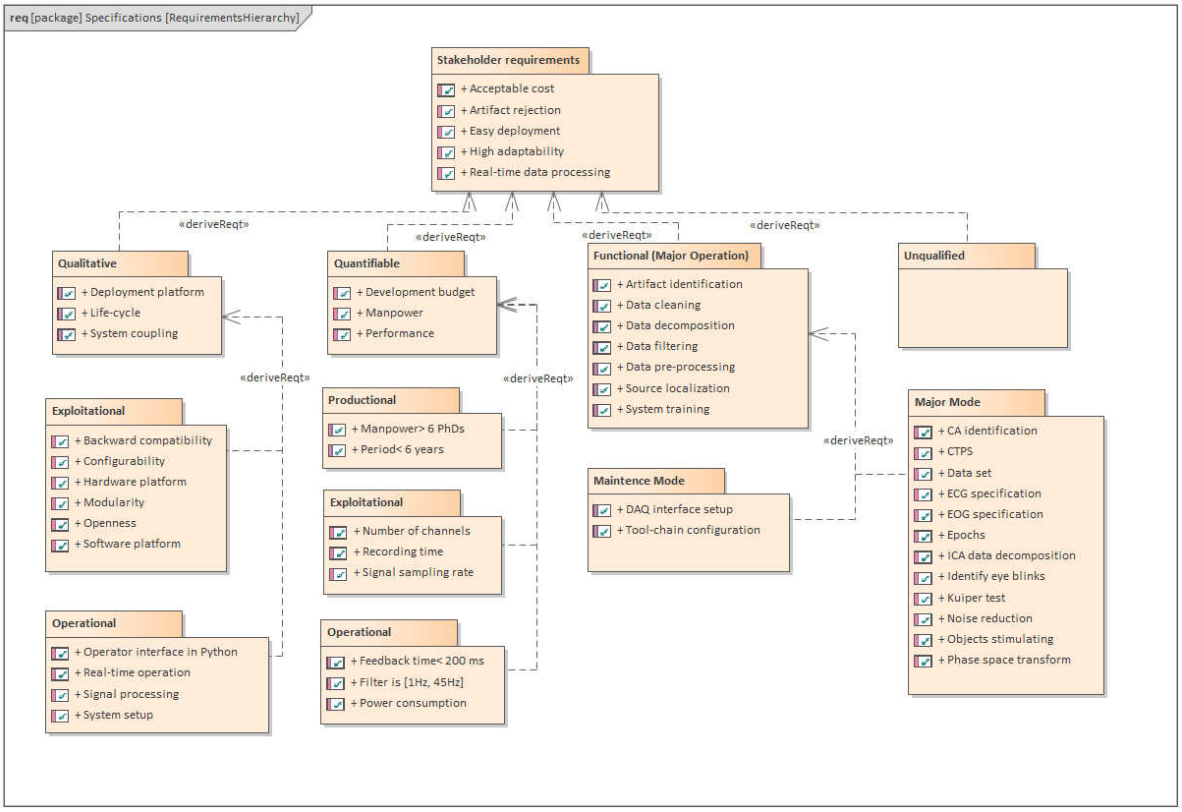

Figure 2.3. Requirement hierarchy of MEG-RT 2.0 (excerpt)

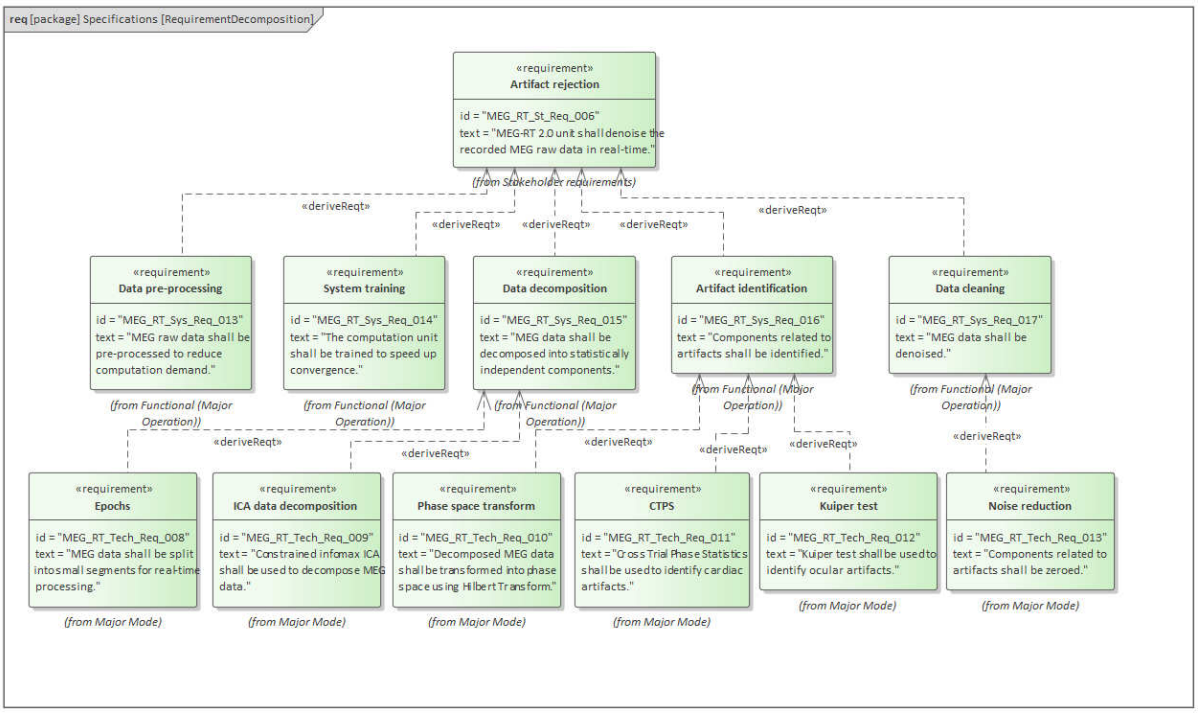

Figure 2.4. Requirement decomposition of MEG signal pre-processing (excerpt)

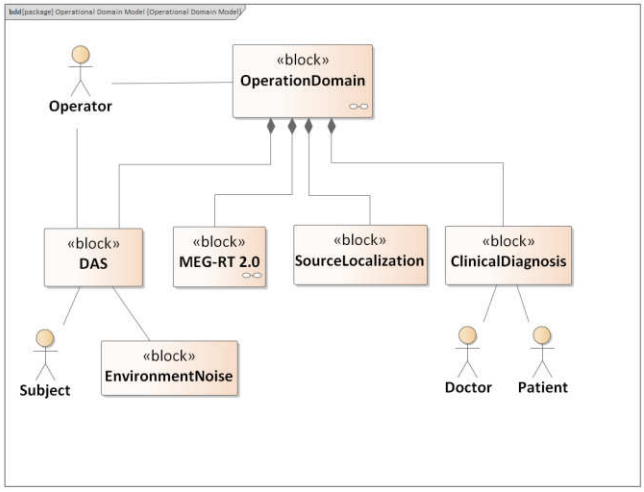

Figure 2.5. Domain model of MEG signal processing

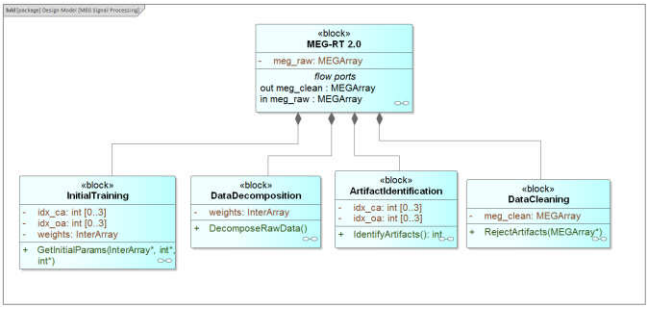

Figure 2.6. Top-level block definition diagram

2.3 System behavior modeling

Behavior models visualize, specify, construct, and document the dynamic behaviors of the system during the interactions with users and the environment [16]. The adopted SysML behavioral diagrams in this study include activity diagrams, state machine diagrams, and sequence diagrams. State machine diagrams describe event-driven behaviors that are not user-centric. Activity diagrams convey system behaviors in terms of the flow of inputs, outputs, and control. Sequence diagrams detail use cases and model interactions among parts of a block.

The MBSE toolkit used in this study supports the allocation of requirements to behavioral elements using simple drag-and-drop to create traceability from requirements to behaviors in the system model [16].

2.3.1 System behaviors of MEG signal pre-processing

Activity diagrams are adopted to specify the MEG data pre-processing workflow using a controlled sequence of actions that transform inputs (matter, energy, or data) to outputs [22],[23]. Fig. 2.8 shows the top-level activity diagram for MEG signal pre-processing. The initial node (a small, filled-in circle) *ActivityInitial* marks the starting point of *MEG Signal Preprocessing* activity. Because data decomposition and artifact identification are performed in parallel to the data cleaning, the pre-processing workflow has two data streams. One data stream is used to update the demixing matrix and the other is used to synchronously reject artifacts in real-time. Therefore, the control token on the initial node concurrently triggers the execution of *ReadData* activity and *DemixParams* action via an outgoing control flow (a line with an arrowhead) and a fork node. Furthermore, an object node models the input or output parameters of an action or activity. The notation for an object node is a small rectangle attached to an action or activity with the object node name and data type floating beside it, e.g., *ReadData* activity has one input parameter *Meg_raw* and one output parameter *Meg_raw* of data type *MEGArray*. When *ReadData* activity is executed, the output parameter *Meg_raw* is transmitted to *Filter* activity by an object flow (a solid line with an arrowhead). *InitialTraining* activity, *DecomposeData* activity, and *IdentifyArtifacts* activity are enabled in a similar manner. When the *IdentifyArtifacts* activity is executed, the output parameters flow to *dp: DemixParams* send signal action (a convex pentagon shaped like a signpost), a specialized action that asynchronously sends matter, energy, or data [23]. *DemixParams* accept event action (a rectangle with a triangular section missing on one side [22]) waits for the parameters of *dp: DemixParams* send signal action to arrive asynchronously. A send signal action and an accept event action work together to model communication between two concurrent work flows. Then the following activities are executed. Finally, the activity final node *ActivityFinal* indicates the termination of the entire activity.

2.3.2 System state transitions of MEG signal pre-processing

State machine diagrams describe system behaviors in terms of operating states, triggering events and system actions.

Fig. 2.9 shows the top state machine diagram for MEG signal pre-processing. It starts with an initial pseudostate, a small and filled-in circle named *GInitial*. Then a transition (a solid line with an arrowhead) leads to *Offline* state. The notation of a state is a round-cornered rectangle, which displays state name and other information. When the guard *Enable MEG-RT 2.0* evaluates to be true, it transits to the composite state *Online*. Again, the composite state starts in an initial pseudostate, then transits to *Initialization* state where the system performs self-check and initialization process. After that, the transition reaches *Training* state, where cost functions of artifacts and spatial template are calculated concurrently. Afterwards the transition goes to the composite state *Active* where MEG data decomposition and data cleaning are performed in parallel. If an error occurs to any of the above mentioned states, the state machine enters the state *Fail*. and then one outgoing transition from the join node goes through the following states and ends in a final state (a small, filled-in circle surrounded by a larger circle) until *GFinal*.

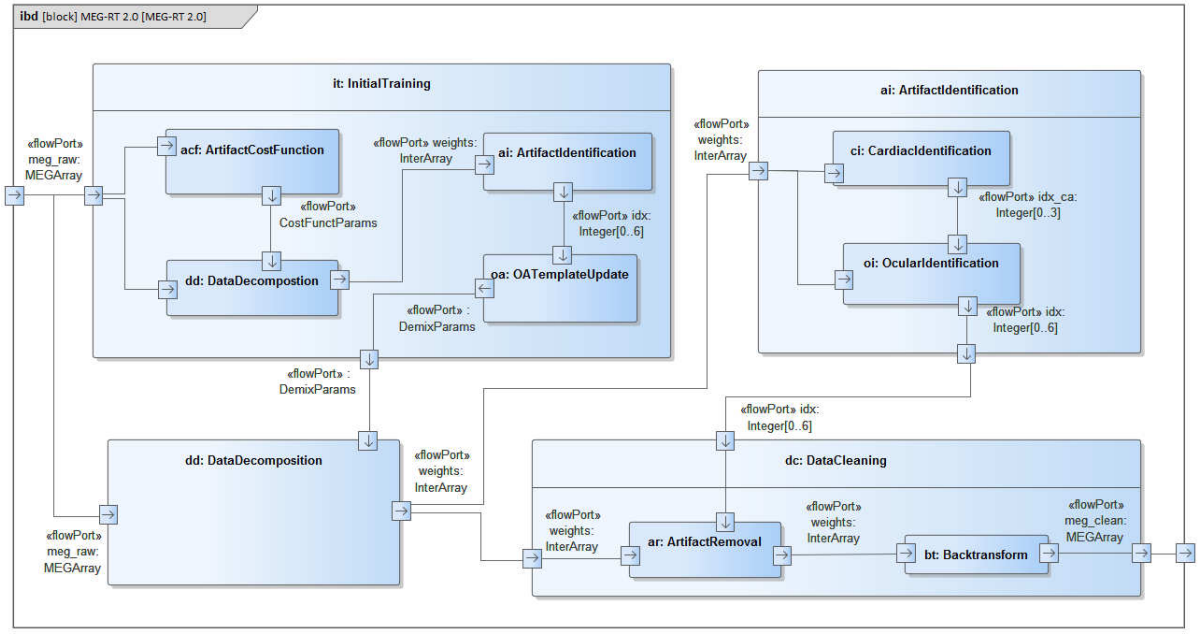

Figure 2.7. IBD of *MEGSignalPreprocessing*

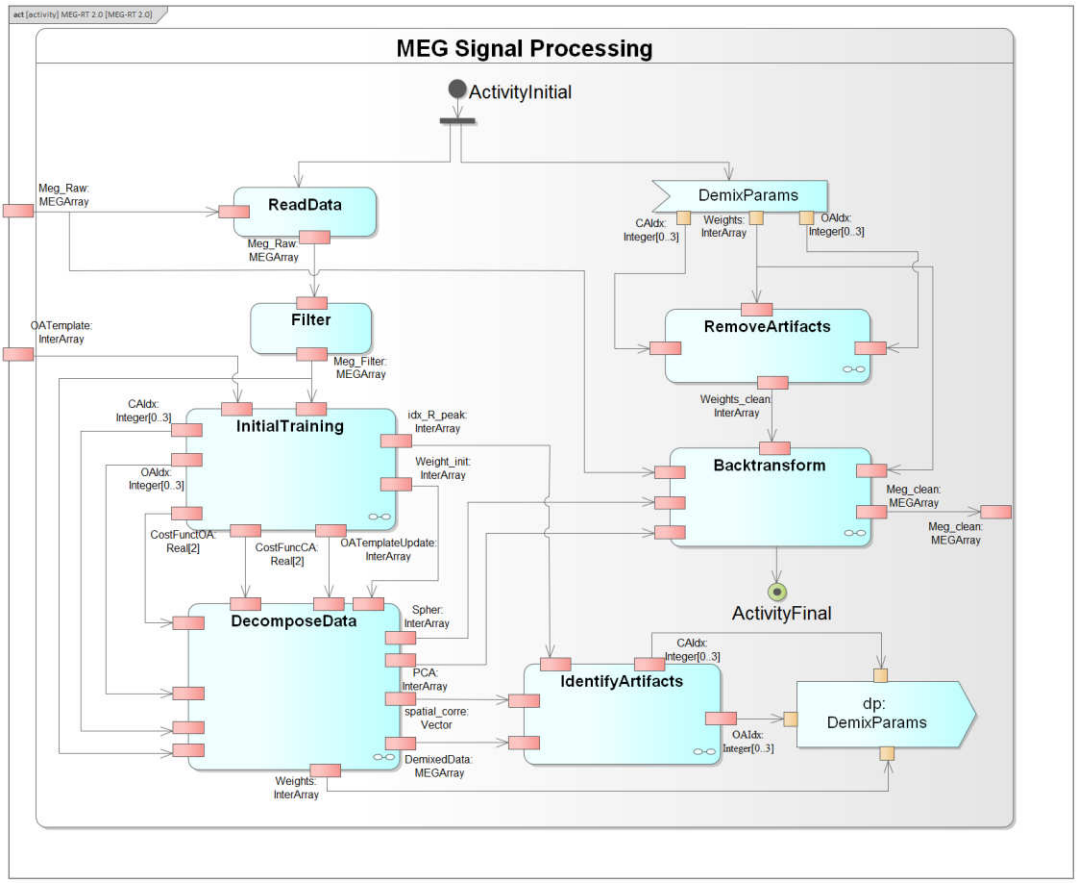

Figure 2.8. System activity diagram of MEG data pre-processing

2.3.3 System interactions of MEG signal pre-processing

Sequence diagrams depict the interaction between system components as a sequence of message exchanges [22]. As plotted in Fig. 2.10, the structural components of MEG pre-processing are represented by lifelines (a rectangle attached with a dashed line), *InitialTraining*, *DataDecomposition*, *ArtifactIdentification*, and *DataCleaning*. The dashed line represents the relevant lifetime of the structural component. Lifelines are participants involved in an interaction that interact with each other by exchanging messages. In *InitialTraining*, the first 12 s of the recording of each measurement are used as a training data set to produce initial parameters, including *CalculateCostFunction*, *DecomposetheData*, *IdentifyArtifacts*, and *UpdateArtifacts* [10]. In real-time MEG signal pre-processing, each measurement is split into segments (e.g., 12 s each) and the segments are processed separately [8],[10].

As mentioned earlier, the processing of one segment is based on the prior calculations of the last segment, as the data segments overlap within a range. Therefore, based on the prior results obtained from *InitialTraining*, the design forms a 2-stage pipeline structure, with *DataDecomposition* and *ArtifactIdentification* being the first stage and *DataCleaning* the second stage. In addition, it is also permissible for a lifeline to send messages to itself, as the case *DataDecomposition* shows. A constrained independent component analysis (cICA) approach is used to decompose MEG data into statistically maximal independent components, including updating data with prior results (*UpdateRawData*), transforming the data with the cost function (*FitSigmoid*) and then computing the demixing matrix iteratively (*ICALearning*) [10]. SysML supports a mechanism termed combined fragments to model complex control logic of interactions. The control logic type is defined by a string (an interaction operator) that appears in a pentagon in the upper left corner of the combined fragment rectangle [23]. The many computation iterations of *DataDecomposition* are modeled with a *loop* operator *loop (0, 200)*, as shown in Fig. 2.10. The loop stops when the demixing matrix converges. When raw data are successfully decomposed into independent components, *ArtifactIdentification* identifies cardiac components based on CTPS (cross trial phase statistics) (*IdentifyCardiacArtifact (CAIdx)*) and ocular components based on a spatial and temporal correlation (*IdentifyOcularArtifact (OAIdx)*) [7],[10],[23]. Finally, *DataCleaning* rejects artifacts components (*RemoveArtifacts (Weights_clean)*) and back-transforms data to MEG space (*Backtransform(Meg_clean)*). A *par* operator is used to indicate that *DataDecomposition* and *ArtifactIdentification* are done in parallel to *DataCleaning*.

2.3.4 System consistency

"*A diagram of the model is never the model itself; it is merely one view of the model*" [23]. Different diagrams were created to show the system from different perspectives. The established MEG model is verified by model execution. The accuracy and integrity of the model is checked by visual examination of the model execution.

The consistency between the system behavioral diagrams is assured by the MBSE toolkit. Every action on an activity diagram must have at least one corrsponding messsage on one of the sequence diagrams. Each messsage on a sequence diagram must appear at least once on one of the activity diagrams. All actions and messages on the behavioral diagrams are realized by the definition of ports and associated interfaces on structural diagrams. The correct behavior of the structural diagrams in response to the inputs is verified by execution and visual check of the corresponding state-based behavior.

To assure that all the defined functional and non-functional requirements are covered, traceability links are established by allocating requirements to the model elements.

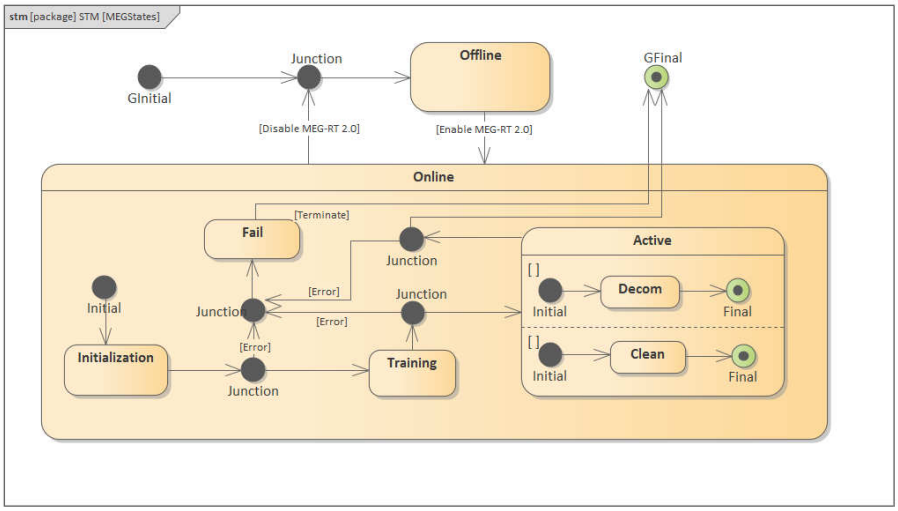

Figure 2.9. System state machine diagram of MEG data pre-processing

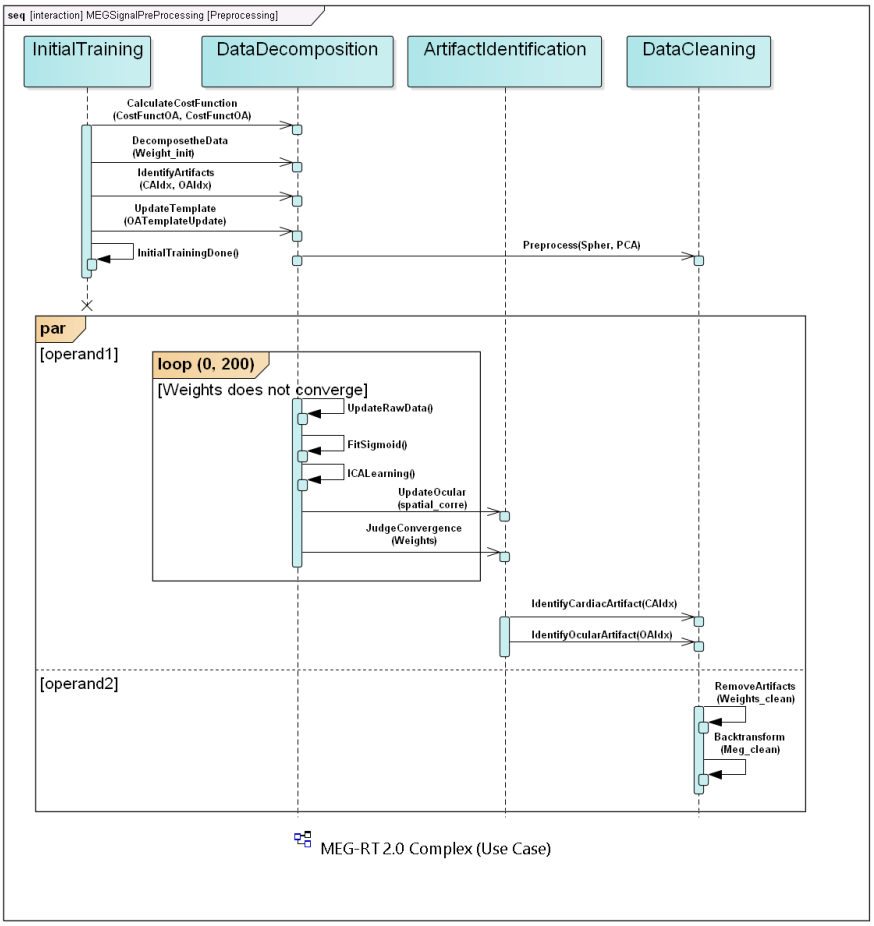

Figure 2.10. System sequence diagram of MEG data pre-processing

2.4 Structure allocation and partition

2.4.1 Constraints and parametrics definition

SysML parametric models include constraint blocks and parametric diagrams, which together impose mathematical relationships on the values of the design.

In Fig. 2.11, constraint blocks contain properties to specify the input and output parameters. Additionally, scripts are embedded in the constraint blocks to depict the executable components. However, Fig. 2.11 does not specify how the constraint blocks and expressions are connected to each other to accomplish a complex computation; this is done by a parametric diagram [23].

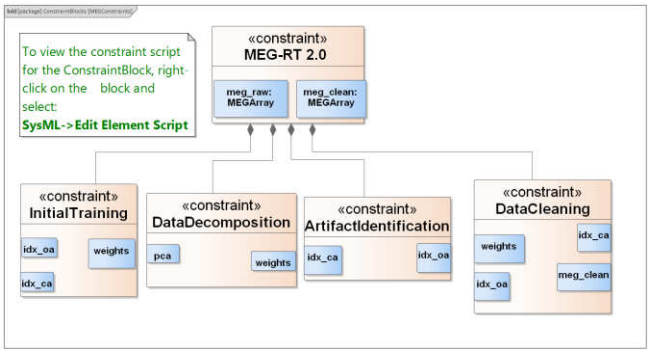

Figure 2.11. Constraint blocks of MEG signal pre-processing

2.4.2 Parametric diagrams and performance analysis

A parametric diagram is a restricted type of IBD that connects instances of primitive constraint blocks to transform a set of inputs into outputs [16] [23].

Constraints are expressed as equations or inequalities whose parameters are bound to the properties of the design [22]. A constraint block is a special kind of block used to encapsulate constraint expressions [23]. Constraint blocks have two main

features: constraint parameters and a constraint expression to impose equations or inequalities. The BDD in Fig. 2.11 shows that the *MEGSignalPreprocessing* constraint block is composed of the four simpler blocks to build a more complex computation. In the constraint blocks defined by the modeling toolkit, a constraint parameter has the notation of a small rectangle floating inside the block. A constraint expression is defined by adding JavaScript, Jscript, or VBScript scripts to a block, which provides the underlying mathematical foundation for further analysis or simulation.

MEGSignalPreprocessing uses these computations mainly to decompose MEG raw data into independent components for subsequent identification and rejection of artifacts.

The mathematical relationships in SysML parametric models can specify the physical properties (e.g., equations, inequalities, or parameter bindings) or the non-functional properties (e.g., performance, speed, power dissipation, or reliability) of a system. Fig. 2.12 shows a parametric diagram for the constraint block *DataDecomposition* from Fig. 2.11, where the *DataDecomposition* is depicted as the frame of the parametric diagram. The constraint properties in the diagram are usages of the constraint blocks nested within *DataDecomposition*. The parameters of the constraint properties are bound to each other and to the parameters of *DataDecomposition*. Together, they show the computation flow of the data decomposition process in MEG signal pre-processing. Specifically, Fig. 2.12 defines precisely how the sub-modules of MEG data decomposition are related. These relationships model the engineering physics and behaviors of the data propagation between the ports of the components within the ICA subsystem. Parametric constraints represent equations addressing the inputs and outputs. Constants are given in the form of value properties.

Based on the established models and analysis, the design is first partitioned to a certain technological basis to best suit system requirements and constraints. The development of MEG-RT 2.0 is based on a single workstation platform combining FPGA and GPU co-processing boards, including a software framework for real-time MEG data analysis. The project covers developments for data pre-processing steps utilizing a Xilinx Virtex-6 FPGA board as well as the integration of hardware modules connecting the software framework of the workstation. The GPU is connected to the FPGA for exchanging intermediate data. Further 3D reconstruction and multi-modal visualization (i.e., MEG and magnetic resonance imaging [MRI] data) takes place in the GPU. As the work focuses on real-time artifact rejection, performance analysis is carried out on the MEG signal pre-processing unit.

Given the parametric analysis and the deduced data-stream models, the MEG signal pre-processing can be quantitatively estimated for performance. In the depicted SE models, the design performance depends on the amount of data exchanged through the modules.

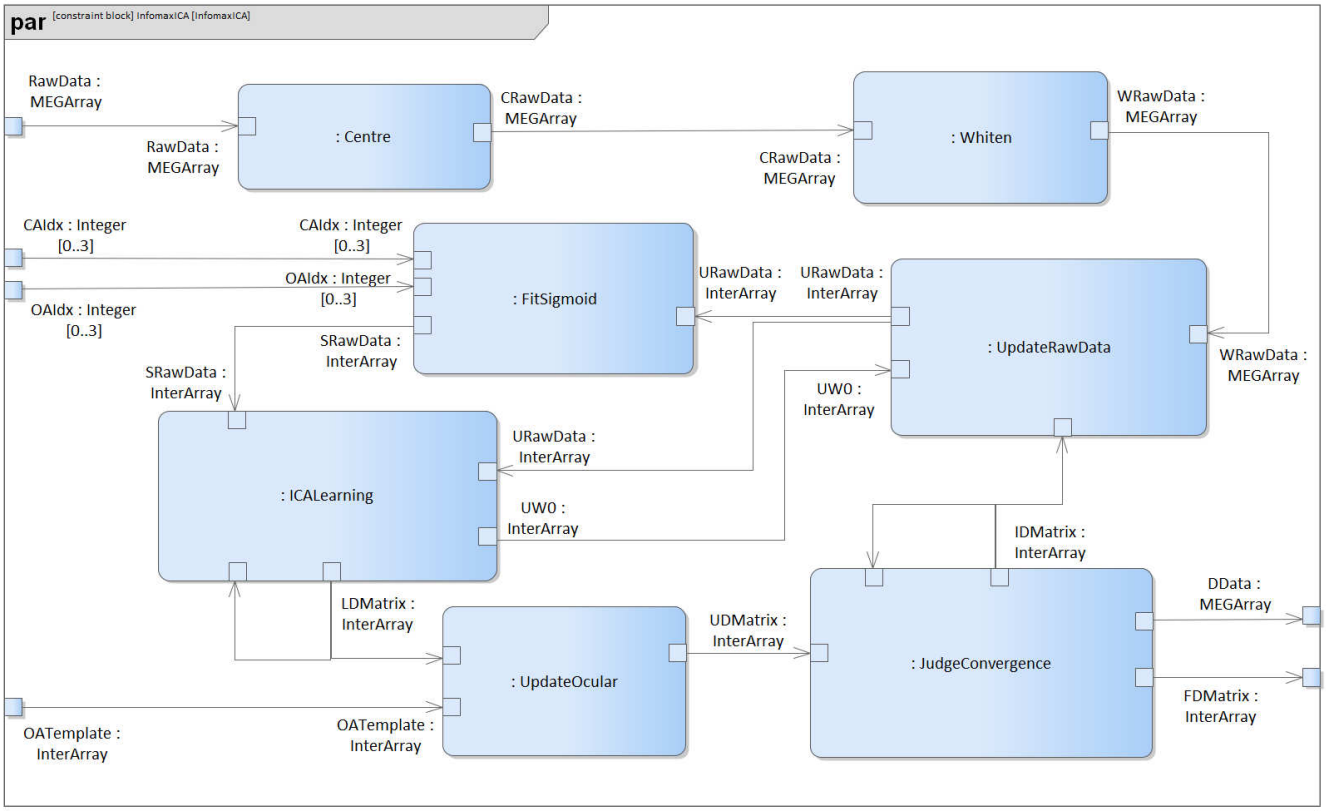

Figure 2.12. Parametric diagram of *DataDecomposition*

Thus, the performance is assessed by calculating the data traffic from the top module to lower level modules.

The present study uses a 16-bit fixed-point numeric. One bit is used for the sign, one bit for the integer part, and 14 bits for the fractional part. Even though the double precision floating point is of higher precision, fixed-point number representation is more practical for hardware designs. The disadvantages of realizing floating-point units on FPGA are high resource and high clock frequency demands. Although previous studies have implemented floating-point computation architectures on FPGAs, very few practical applications exist [24]. On the other hand, modern FPGAs are equipped with numerous hardwired fixed-point digital signal processors (DSPs) suitable for fixed-point arithmetic realization, which potentially improves performance and power efficiency. Convergence has been reported in [24],[25],[26],[27] for the FPGA implementations of ICA and deep neural network training using 16-bit fixed-point computation units. The rough precision of the 16-bit fixed-point numeric can reach 0.000030518. Before the computation, normalization is performed on the MEG data to avoid overflow.

According to the data specification in Tab. 2.2, the volume of MEG data is exceedingly large for FPGA on-chip BRAM (block random access memory). Thus DDR3-SDRAM external memory is used to store MEG data and intermediate results of large volume. The Xilinx DDR3 IP core is utilized to instantiate and control the DDR3 RAM with the read/write width of 256 bits and clock frequency of 400 MHz.

The hardware resource model is based on counting of clocked memory (in bits) required for allocation of local data aggregates of the FPGA. Although it is not intended to yield a precise flip-flop budget for each processing unit of the FPGA implementation, it can predict each design module's order of complexity with a high degree of certainty. Tab. 2.2, Tab. 2.3, Tab. 2.4 and Tab. 2.5 presents the prognostic resource model computed for different modules of the MEG signal pre-processing system.

Tab. 2.2 shows the data traffic of different sub-modules in principal component analysis (PCA). It mirrors the application performances in different sub-modules depending on the data flow in the processing units. Tab. 3.1 also specifies the processing time (data in brackets estimate the processing time with an operating frequency of 100 MHz).

The PCA computation is separated into four steps. The data traffic bottleneck is the estimation of eigenvalues, where the calculation requires a number of iterations. The intermediate results of the *EigenAnalysis* and *PCMatrix* modules are small-volume and stored in the the block random access memory (BRAM) on FPGA for speeding up the computation by the short memory access time. It is important to note that Centre and COV modules involve processing very large dimension matrices. The input and output streams of the two modules have external SDDR3 as the memory bank, which adds to the processing time. Therefore, the upward and downward data streams of the two modules are split into three channels operating in parallel.

Table 2.2: Data traffic of PCA for performance estimation and total percentage of Virtex-6 flip-flop memory (in memory words)

Module	Centre	COV	EVD	PC	Percentage
Data traffic	3,027,336 (60.54 ms)	3,027,336 (375.36 ms)	61,504 (152.52 ms)	3,033,536 (75.68 ms)	41%

Tab. 2.3 specifies the traffic volume of different sub-modules in ICA and the application performances in different sub-modules are estimated based on the data flow in the processing units. The table 2.3 also calculates the processing time (data in brackets estimate the processing time with an operating frequency of 100 MHz).

The ICA computation is separated into three modules: data update, judge, and learning. The data traffic bottleneck is the learning module, where exponential computation and multiple matrix calculations are required. The intermediate results have small data volume and are stored in the block RAM on FPGA for speeding up the computation by the short memory access time. Prior knowledge from the previous data segment is incorporated during the computation. The measured signal mixtures are separated into underlying sources within one iteration, which greatly speeds up the decomposition.

Table 2.3: Data traffic of ICA for performance estimation and total percentage of Virtex-6 flip-flop memory (in memory words)

Module	Data update	Judge	Learning	Percentage
Data traffic	305,775 (8 ms)	625 (0.006 ms)	305,775 (158.68 ms)	68%

Tab. 2.4 shows the data traffic of different sub-modules in CTPS. Application performances in different sub-modules are presented based on the data flow in the processing units. Meanwhile, the processing time is analyzed (data in brackets specify the processing time under the operating frequency of 100 MHz).

The CTPS computation is divided into three modules: Hilbert transform, cumulative distribution function (CDF) estimation, and the Kuiper test [7]. The data traffic bottleneck is the Kuiper test, where exponential computation is involved. The intermediate results have small data volume and are stored in the block RAM on FPGA for speeding up the computation by the short memory access time.

Table 2.4: Data traffic of CTPS for performance estimation and total percentage of Virtex-6 flip-flop memory (in memory words)

Module	Hilbert transform	CDF estimation	Kuiper test	Percentage
Data traffic	509 (0.37 ms)	509 (0.15 ms)	509 (0.43 ms)	11%

As shown in Tab. 2.5, the data traffic density of temporal and spatial correlation design is much lower, as the computation is simple and only involves addition, vector–vector multiplication, and division. Thus, memory access occupies most of the computation time.

Table 2.5: Data traffic of temporal and spatial correlation for performance estimation and total percentage of Virtex-6 flip-flop memory (in memory words)

Module	Covariance	Variance	Division	Percentage
Data traffic	12206 (0.21 ms)	12206 (0.18 ms)	16 (0.08 ms)	4%

III Implementation of MEG Signal Pre-processing

Based on the behavioral models, HDL code can be generated automatically, which is often sufficient for direct implementation in hardware. The algorithms covered in artifact rejection were implemented by modifying the generated Verilog hardware description language (HDL) code. Each algorithm is described briefly, and then the implementation process is detailed.

3.1 Code generation

MEG signal pre-processing mainly involves matrix–matrix multiplications or matrix–vector multiplications, for which parallel computation is optimal [10]. For the purpose of acceleration, reconfigurable hardware solutions such as FPGA are currently valuable approaches for tackling the computation complexity.

Enterprise Architect well supports code generation and reverse engineering. We generate HDL code from MEG-RT 2.0 state machine diagrams, which describe system behaviors with respect to operating states, triggering events and system actions [16].

For generating code from behavioral models, the state machine diagrams are contained within a class diagram. Then, two triggers (reset and clock) are added and associated with the top-level state machine diagram. Meanwhile, the triggers are associated with the component's ports in the class diagram. After the preliminaries above, the desired HDL code is generated. The final step is to feed the generated code into an event-driven simulator and verify the design.

3.2 Implementation

Based on the behavioral models, HDL code can be generated automatically, which is often sufficient for direct implementation in hardware. The algorithms covered in artifact rejection were implemented by modifying the generated Verilog HDL code.

3.2.1 Implementation of PCA

PCA is one of the most commonly used techniques to decrease the dimensionality of a dataset while retaining most of the information by means of its variance [28], which reduces the computational complexity of ICA. PCA maps the original data into a new set of orthogonal basis vectors, where the components are stored in a decreasing order of variance [29]: The first axis (first principal component) accounts for the greatest data variance of the original data set while the last one the least data variance. In terms of MEG data processing, the first 25 principle components are chosen for data processing as they explain more than 95% of the data variance. Therefore, the design shall be adaptable to different MEG systems with different channel numbers.

PCA implementation is detailed here as it has been introduced in a previous publication [30].

3.2.2 Implementation of ICA

In this study, the natural-gradient version of Infomax ICA [31] is chosen for hardware implementation. It can be learned from Eq. 3.1 that not only is the convergence speed of $\boldsymbol{\Delta W}$ increased, but also that the computation of $[\boldsymbol{W^T}]^{-1}$ is avoided, which facilitates the FPGA realization.

Typically, ICA-based signal decomposition is performed on the whole data set at once. In our experiments, this would translate to about 170,000 time samples (see Section 4.1). To realize real-time MEG data processing, data are split into much smaller segments for analysis. For data decomposition, we use a sliding window with a window width of 12 s and 10 s overlap. Please refer to [8] for a detailed description on time window length selection and relevant effects.

$$\tilde{\boldsymbol{u}} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{W_0}, \text{ and}$$

$$\boldsymbol{W_{i+1}} = \boldsymbol{W_i} + \boldsymbol{\varepsilon}\left[\boldsymbol{BI} + \left(1 - 2g(\tilde{\boldsymbol{u}})\right)\tilde{\boldsymbol{u}}^{\boldsymbol{T}}\right]\boldsymbol{W_i} = \left[\boldsymbol{I} + \boldsymbol{\varepsilon BI} + \boldsymbol{\varepsilon}\left(1 - 2g(\tilde{\boldsymbol{u}})\right)\tilde{\boldsymbol{u}}^{\boldsymbol{T}}\right]\boldsymbol{W_i} \quad (3.1)$$

where $\boldsymbol{B}$ is the segment size, $\boldsymbol{\varepsilon}$ is the learning rate whose initial value is 10^{-3}. In general, demixing matrices are estimated for segments separately, i.e. whenever the estimation of $\boldsymbol{W_i}$ is completed, the next demixing matrix $\boldsymbol{W_{(i+1)}}$ is estimated on basis of the last 12 seconds. During the estimation of $\boldsymbol{W_{(i+1)}}$, $\boldsymbol{W_i}$ is used for data decomposition. In the computation process, $\boldsymbol{I} + \boldsymbol{\varepsilon BI}$ is seen as a constant matrix. Based on the features of FPGA design, the Infomax implementation can be divided into input module, learning module, matrix multiplication module (data update), judge

module and output module, among which learning module is the core of the design. The learning module initializes demixing matrix $\boldsymbol{W}$ and performs the learning iterations in Eq. 3.1. The workflow of Infomax is illustrated in Fig. 3.1.

The input module reads out segments of data from the memory. The chosen sliding window with a window width of 12 s translates to 12,206 data points. After reading the 12,206 data points, the output of the input module is set to zero and waits for the trigger signal of the learning module.

The learning module performs the learning process of Infomax ICA and computes the demixing matrix $\boldsymbol{W}$ of the data segments. The identity matrix was chosen as the initial matrix when calculating the demixing matrix $\boldsymbol{W_1}$ of the first data segment. The previous demixing matrix $\boldsymbol{W_{i-1}\ with\ i = 2, 3, \ldots, n}$ is used as the initial matrix to compute the demixing matrix $\boldsymbol{W_i\ with\ i = 2, 3, \ldots, n}$ of the current segment, where $\boldsymbol{n}$ is the number of data segments. Then, the demixing matrix is transmitted to the matrix multiplication module for data update.

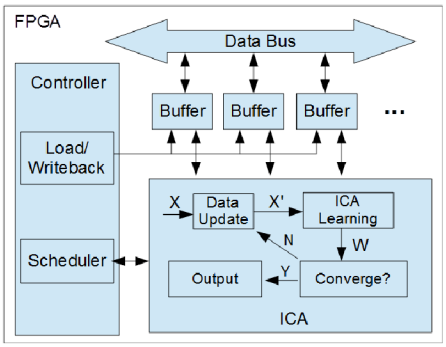

Figure 3.1. Architecture of data decomposition

The matrix multiplication module multiplies the MEG data segment with the corresponding demixing matrix $\boldsymbol{W}$ and transmits the demixing matrix $\boldsymbol{W}$ to the judge module.

The judge module checks the convergence of the demixing matrix $\boldsymbol{W}$. In the case of convergence, the computation stops, otherwise the computation is repeated.

3.2.3 Implementation of CTPS

The first step of CTPS is to transform the MEG signals into phase space. Our design adopted the Hilbert transformation to extract phase information. Then, the phase distribution is compared with a uniform distribution using the Kuiper test [32] to identify the components related to cardiac artifacts.

The Kuiper test is a statistical test that refines the well-known Kolmogorov-Smirnov test (K-S test) [33]. The Kuiper test is the adaption of the K-S test for cyclic problems. It quantifies the probability that two data sets are samples of the same distribution. For analyzing event-related phase-locked responses, the cross-trial cumulative phase distribution is compared with a cumulative uniform distribution [34]. The Kuiper test algorithm was realized in Verilog using the Xilinx intellectual property (IP) core library to ease the implementation of blocks as memories or arithmetic operators. First, the CDF of the phase samples is estimated, as shown in Fig. 3.2. After the CDF of the phase values is calculated, it is tested against a uniform distribution by the operations of subtraction and the maximum search.

3.2.4 Implementation of temporal and spatial correlation

Ocular artifacts are automatically identified by estimating the Pearson linear correlation between each independent component of the MEG signal and the reference electrooculogram (EOG) signal. As columns of the mixing matrix contain the spatial information, it is also taken into account by calculating the Pearson linear correlation between the template and the field maps as extracted by ICA [10]. The implementation mainly involves covariance and variance calculation, which is simple and is not presented here.

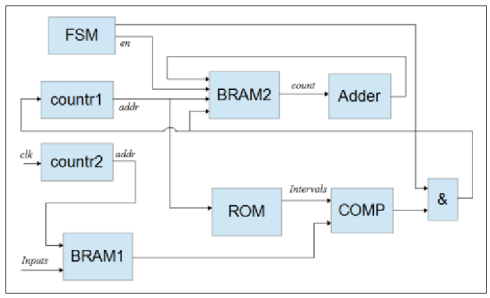

Figure 3.2. Architecture of CDF estimation

IV Results and Discussion

Cost analysis of resource and performance was carried out to assess the efficiency of the proposed prognostic (pre-implementation) model. The design was implemented on a Virtex-6 FPGA (running at 100 MHz) by optimizing the Verilog HDL code automatically generated by Enterprise Architect for further synthesis using the Xilinx ISE 14.1 tool. Questasim 10.5c and Xilinx Chipscope Pro 14.1 were used to validate the quantitative requirements and functional correctness. The test MEG data set was recorded with a whole-head magnetometer system (Magnes 3600WH, 4D-Neuroimaging) [8]. The configurable hardware design of MEG signal pre-processing was carried out by extensive parameterization of functional blocks to be generally applicable to other MEG systems.

4.1 Experiment data set

The design was evaluated using MEG data from a 248-channel whole-head magnetometer system (Magnes 3600WH, 4D-Neuroimaging).

During the experiments, MEG signals from five subjects were measured. The experiment setup is to record neuromagnetic field

changes due to finger tapping cued by auditory stimulation. The auditory stimulations were 50-ms sinusoidal tones (single clicks) of 1000 Hz [10]. The dataset is chosen because the experimental paradigm studies the auditory cortex, a well-known area in the brain. Detailed information on the dataset is listed in Tab. 4.1.

Table 4.1: Test MEG data details ([10])

sampling rate	1017.25Hz
bandwidth	0.1-400Hz
number of presented stimuli	120
stimulus frequency	1000 Hz
stimulus duration	50 ms
inter stimulus interval	2.0 ± 0.5 s
number of data samples (mean)	168098
experiment duration (mean)	165 s

4.2 Results and analysis of ICA design

4.2.1 Results of ICA design

For design evaluation, the 25 channels of mixed signals are stored in the block RAM on FPGA. Each channel consists of 12,206 data points. The ICA modules decompose the mixed signals into relevant components.

Tab. 4.2 shows the summary of synthesis for Infomax ICA implementation, which correlates highly with the results of the prognostic models created for system-level modeling. The reconfigurable hardware designs of Infomax ICA in this work are parameterized and easily adaptable to data of different dimensions, which results in different resource utilization.

Table 4.2: Resource utilization of ICA design

Item	Occupied	Available	Percentage
Number of slices	34,128	56880	60%
Number of DSP blocks	311	576	54%

The proposed Infomax ICA hardware design is driven by a system clock of 100 MHz; the target MEG data has a window size of 12 s and a sampling rate of 1017.25 Hz. The first 12 s of the data from each measurement was used as training data set. The data segments overlap within a time range of 10 s, so the dynamics of the underlying sources in two overlapping data segments are analogous. The processing of one segment is based on the prior calculations of the last segment, which speeds up the decomposition, and the signal separation can be done in one iteration. The maximum hardware computation time is 17,326,971 clock cycles. Thus, the throughput is 6103 points/s. Therefore, the implemented Infomax ICA architecture can support the real-time requirement of our applications well.

4.2.2 Result analysis and discussion

The requirements of ICA design are first traced to different models to ensure that all requirements are covered in system modeling. The requirements (*DataDecomposition and Method*) of ICA design are linked to model elements, including a state machine model *InfomaxICA*, a use case diagram ICA, and a sequence diagram *DataDecomposition*, which are colored yellow in Fig. 4.1. Various other traced model elements are not listed for the readability of the figure. Afterwards, the ICA requirements are realized and verified on an FPGA board. The implemented (covered) requirements are colored green in Fig. 4.1. Final realization is traced to relevant requirements to ensure the requirement coverage of SE.

Many studies have been devoted to ICA hardware implementation. Hardware realizations of ICA algorithms are challenging with respect to resource, flexibility, and speed. Many studies on the real-time implementation of ICA have targeted data of much lower channel and sample numbers. Higher channel and sample numbers increase the computation time for ICA processing because of the greater computational complexity. In [24],[35],[36], the FPGA implementations can only achieve 2-channel ICA processing for speech signals. In [37],[38],[39], the proposed hardware architectures can perform ICA processing with a channel number of 4. In [40],[41],[42], 8-channel ICA implementations are reported. In [43],[44], the ICA implementation can achieve 16-channel processing. Thirty-two–channel convolutive ICA is first implemented on FPGA and applied to real-world signals in [45]. However, the post-layout results of the design are not reported. Another 32-channel ICA implementation on FPGA used for electroencephalogram (EEG) signal processing is reported in [46], but the implementation information is currently not accessible. In the present study, the proposed hardware implementation of Infomax ICA can decompose MEG signals with a channel number of 25, which is parameterized and can be scaled to applications with channel numbers > 25.

Speed is also an important criterion for evaluating a hardware implementation of ICA. The ICA design implemented in [36] and [45] works at the operating frequencies of 12.3 MHz and 50 MHz, respectively. The implementation in [36] is as short as 0.003 s for lower-dimension ICA computation, while the realization in [45] takes >60 s for adaptive noise canceling. A parallel ICA is implemented on FPGA in [37],[38]: it has the operating frequency of 20.2 MHz used for hyperspectral image analysis. The 4-channel Infomax ICA is implemented on FPGA in [39] with the operating frequency of 68 MHz applied to EEG signal processing. In our work, the operating frequency of the proposed ICA hardware architecture is 100 MHz with the computation time of 1.72s.

The present work has greater resource consumption than previous studies, as the existing implementations are only for signal processing of much lower channel and sample numbers. The processing capacity in the present study can be up to 25 channels and 12,206 samples, which are both higher than previous implementations. Additionally, our design can process data with dimensions larger than 25 × 12,206, as it is adaptable.

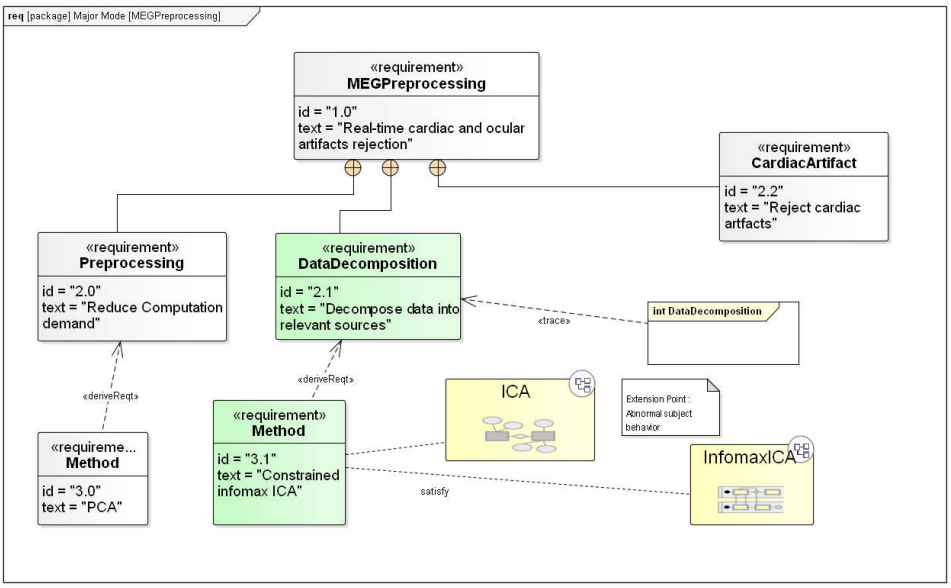

Figure 4.1. Requirement coverage of ICA design

4.3 Results and analysis of artifact rejection design

4.3.1 Results of artifact rejection design

Artifact components are identified and removed from recorded signals using CTPS. The design is evaluated using the MEG data which is windowed around the latency of the R-peak of the ECG signal using a window size of 500 ms. Each data window consists of 508 data points, which is of small volume and buffered in the block RAM on FPGA. The CTPS modules first transform the data into phase space using Hilbert transform, and then identify the cardiac components.

Ocular components are identified by computing the Pearson linear correlation between the MEG component and the reference EOG signal and also between the template T and the field maps. Mathematically, the calculations mainly involve divisions and vector–vector multiplications.

Tab. 4.3 shows summary of synthesis for the artifact rejection implementation. There is a strong correlation between the resource prediction model in Tab. 3.3 and Tab. 3.4 and the synthesis results. Meanwhile, it is difficult for the Resource Model to obtain the absolute accuracy in prediction, as the results of synthesis depend on the optimization algorithms, e.g., logic replication, typically exploited by modern synthesis tools for improving the timing characteristics of synthesized designs.

The proposed artifact rejection hardware design is driven by a system clock of 100 MHz; the data epochs (segments) during CTPS computation are buffered in on-chip BRAM, which reduces the memory access time. For ocular artifact rejection, the data channels are pre-fetched, which improves the data access speed. There is no data dependency between the covariance calculation and the computation of the standard deviations, which largely improves the computation parallelism. The design adopts the pipeline structure, further enhancing the throughput. Thus, the computation modules are fully occupied. The data is processed by one module, while the results are transmitted to the next unit for further computation in every clock cycle. Additionally, the computations are designed to overlap with the memory access to make best use of the pipelined architecture. The execution time is measured for each data segment during CTPS computation, and the average is 223 clock cycles. In the testing, an average two components were identified as related to cardiac activity. For ocular artifact rejection, the execution time is measured for each data channel, and the average is 106 clock cycles. In the testing, 1–3 components (average, 1.5) were identified as being attributed ocular activity. The rejection performance measure as introduced in [10] is used to evaluate the cardiac artifact rejection results. Cardiac artifacts were sufficiently removed with the hardware implementation of CTPS while keeping the signal of interest unchanged.

Table 4.3: Resource utilization of artifact rejection design

Item	Occupied	Available	Percentage
Number of slices	15742	56880	27%
Number of DSP blocks	95	576	16%

4.3.2 Result analysis and discussion

The requirements of CTPS design are satisfied in two ways. On the one hand, traceability between the requirements and different models is established, which ensures requirement coverage during

the system modeling process. The requirements (*CardiacArtifact and Method*) of CTPS design are traced to model elements including a state machine model *CAStructureAllocate*, a use case diagram *CAIdentification* and a sequence diagram ArtifactIdentification, which are colored yellow in Fig. 4.2. The other traced model elements are not shown for the readability of the diagram.

Subsequently, the CTPS is realized and verified on an FPGA board to further cover the relevant requirements. The covered requirements by CTPS design are colored green in Fig. 4.2. The hardware implementation of CTPS ensures the traceability of requirements to the final realization.

Again, the requirements of temporal and spatial correlation design (OcularArtifact, IdentifyEyeBlinks and Method) and data cleaning (DataCleaning) are first traced to different model elements guarantee the requirements coverage in system modeling. They are refined by tracing to both structural and behavioral diagrams.

Secondly, the requirements of temporal and spatial correlation and data cleaning are covered by implementation and verification on an FPGA board. The covered requirements in this section are colored green in Fig. 4.3. The requirements covered in previous sections are colored yellow in Fig. 4.3. Up to now, all the requirements of MEG signal pre-processing are fully covered, not only by traceability to different system model elements, but also by realization and verification on an FPGA board.

Real-time capability is achieved in this work by using a parallel computation platform FPGA, which makes best of the parallelism of the matrix-matrix multiplications or matrix-vector multiplications in MEG signal pre-processing. The computation modules of lower levels are designed in the pipeline structure, further enhancing the throughput. Data cleaning is performed in parallel to estimating a new demixing matrix (see Fig. 2.8), leading to the possibility to massively parallelize computation on FPGA. The data cleaning procedure is performed with a time delay of less than 1 ms.

In [10], MEG data is tested on the Intel i5-2410M Core with the power dissipation of 35W. The reported processing time, including data decomposition and artifact rejection, is 1.1 seconds without taking the data training into account. The average computation time in this study is 1.4 seconds but with much lower power dissipation (maximum 7.5W). The CPU implementation (on an Intel Core i5-2410M, 2.3 GHz, 6 GB RAM) is not application specific and often slowed by the computation tasks of other threads. While the FPGA dedicated design has advantages in terms of both speed and resource consumption.

As described in section 5.1.1, signal processing systems with a delay of less than 200 ms are considered to be real-time in this work. Different computation times for real-time feedback design are indicated in previous studies. In the studies in [9],[13],[47], real-time designs a feedback update every 500 ms are reported. A real-time system with a feedback update every 300 ms is provided in [12]. Despite the differences in feedback delay, all authors described their systems as being real-time capable. In this work and in [10], the artifact rejection, not including data decomposition and data training, takes < 1 ms. In summary, the hardware implementation of MEG signal pre-processing offers effective real-time capabilities, with low power dissipation, high speed and reasonable resource consumption.

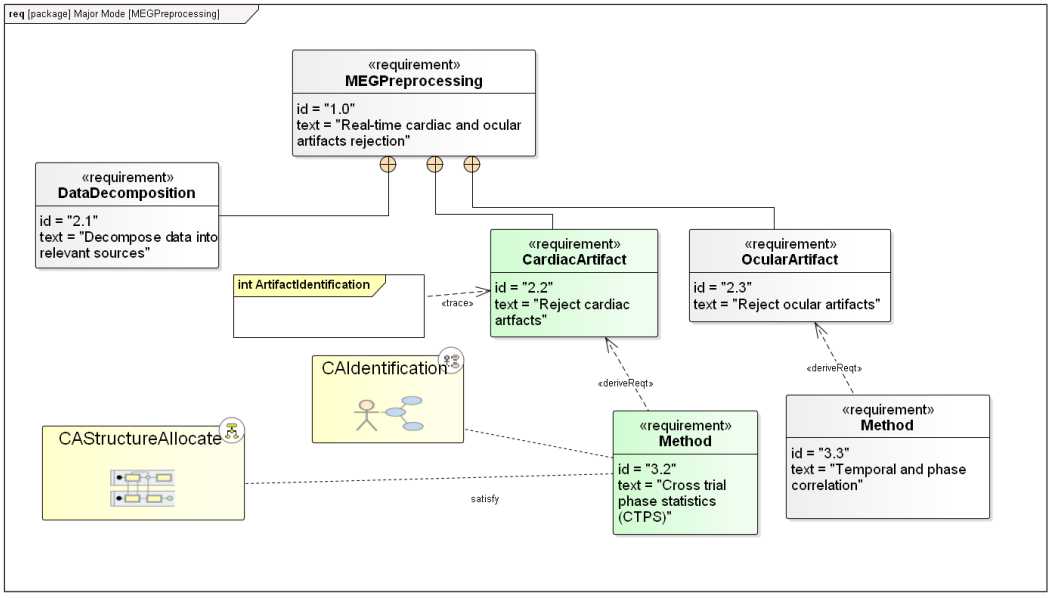

Figure 4.2. Requirement coverage of CTPS design

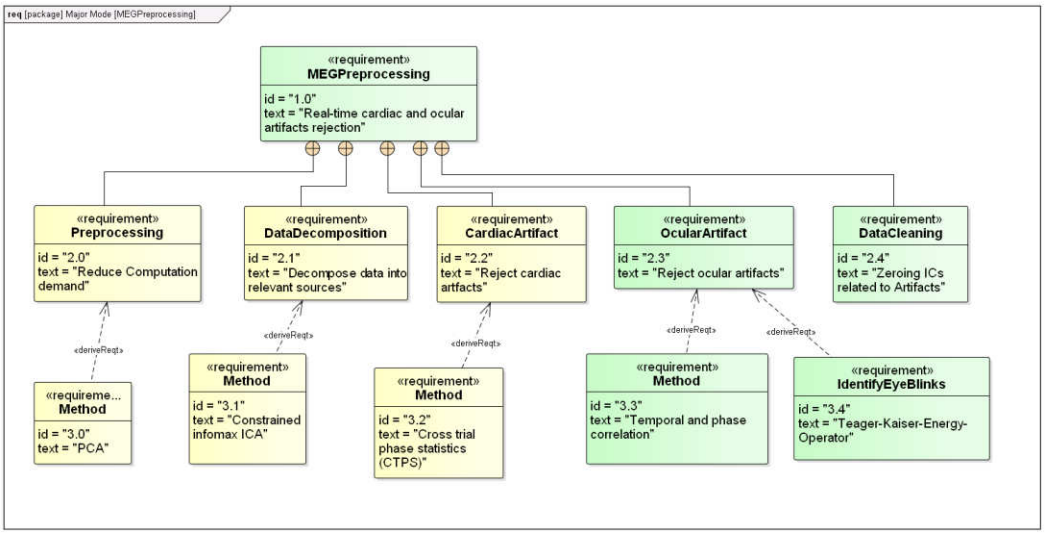

Figure 4.3. Requirement coverage of temporal and spatial correlation

V Conclusion

This work introduces an RDD & MBD methodology for real-time computation systems. It handles the ever-increasing complexity of computation systems by increasing the abstraction level in the design.

The present work focused on the design of a SoC capable of performing real-time artifact rejection in MEG data processing, because previous studies have either neglected or simplified real-time artifact rejection [9],[12],[13],[48]. The reconfigurable hardware designs of MEG signal pre-processing here were parameterized and could be modified externally within the same hardware architectures to be generally applicable to other MEG systems.

The MEG-2.0 RT project aims to develop an MEG real-time signal-processing device to be used as an add-on to existing MEG systems, thus enabling, for example, neurofeedback applications. The system model of the real-time MEG signal processing chain developed here and the real-time artifact rejection implementation that complied with the system/project requirement model is a first and essential component of the Jülich Research Center MEG-2.0 RT project.

As part of the future work, source localization is an important approach to image the electrical activities of deep brain structures in both fields of EEG and MEG. The RDD & MBD approach is ideal working on the source localization problem, because real-time and embedded systems can be easily designed using SysML [49]. In this context, the high level exploration process helps the developer effectively manage the design complexities. Additionally, the choice of high-speed hardware platforms well meets the demand of providing data processing results in real-time.

Acknowledgment

The presented research is supported by China Scholarship Council (CSC), in cooperation with the Central Institute of Engineering, Electronics and Analytics - Electronic Systems (ZEA-2) and the Institute of Neuroscience and Medicine (INM-4) at Forschungszentrum Jülich GmbH.

References

[1] Weilkiens, Tim. Systems engineering with SysML/UML: modeling, analysis, design. Elsevier, 2011.

[2] Office of the Deputy under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering. Systems Engineering Guide for Systems of Systems, Version 1.0. Washington, DC: ODUSD (A&T) SSE, 2008.

[3] Chen, Tao, et al. "Model-Driven Development Methodology Applied to Real-Time MEG Signal Preprocessing System Design." *2017 European Modelling Symposium (EMS)*. IEEE, 2017.

[4] Suslov, Sergey. Parallelisation Potential of Image Segmentation in Hierarchical Island Structures on Hardwareaccelerated Platforms in Real-time Applications. Forschungszentrum Jülich, 2013.

[5] Suslov, Sergey. Presentation: SysML for computing controlling system development, https://www.fz-juelich.de/SharedDocs/Downloads/ZEA/ZEA-2/DE/SysML.pdf?__blob= publicationFile, 2017.

[6] Biomagnetic Technologies Inc., “MAGNES 2500 WH-X and 3600 WH Hardware Reference Manual,” BTi San Diego, California, USA, 2006.

[7] Dammers, Jürgen, et al. "Integration of amplitude and phase statistics for complete artifact removal in independent components of neuromagnetic recordings." IEEE transactions on biomedical engineering 55.10 (2008): 2353-2362.

[8] Breuer, Lukas, et al. "Ocular and cardiac artifact rejection for real-time analysis in MEG." *Journal of neuroscience methods* 233 (2014): 105-114.

[9] Sudre, Gustavo, et al. "rtMEG: a real-time software interface for magnetoencephalography." *Computational intelligence and neuroscience* 2011 (2011): 11.

[10] Breuer, Lukas, “Identification of Neuromagnetic Responses for Real-Time Analysis in Magnetoencephalography”, RWTH Aachen University, Aachen, Germany, 2015.

[11] Rongen, H., V. Hadamschek, and M. Schiek. "Real time data acquisition and online signal processing for magnetoencephalography." Real Time Conference, 2005. 14th IEEE-NPSS. IEEE, 2005.

[12] Buch, Ethan, et al. "Think to move: a neuromagnetic brain-computer interface (BCI) system for chronic stroke." Stroke 39.3 (2008): 910-917.

[13] Mellinger, Jürgen, et al. "An MEG-based brain–computer interface (BCI)." *Neuroimage* 36.3 (2007): 581-593.

[14] Esch, et al. "MNE Scan: Software for real-time processing of electrophysiological data." *JOURNAL OF NEUROSCIENCE METHODS* (2018).

[15] Dinh, Christoph et al. 2015. "Real-Time MEG Source Localization Using Regional Clustering." *Brain Topography* 28.6(2015):771-784.

[16] Rosenberg, Doug, and Sam Mancerella. "Embedded systems development using SysML: an illustrated example using enterprise architect." Sparx Systems Pty Ltd and ICONIX (2010): 4-14.

[17] Hämäläinen, Matti, et al. "Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain." *Reviews of modern Physics* 65.2 (1993): 413.

[18] Dale, Corby L., et al. "Timing is everything: neural response dynamics during syllable processing and its relation to higher-order cognition in schizophrenia and healthy comparison subjects." *International Journal of Psychophysiology* 75.2 (2010): 183-193.

[19] Lauer, Richard T., et al. "Applications of cortical signals to neuroprosthetic control: a critical review." *IEEE transactions on rehabilitation engineering* 8.2 (2000): 205-208.

[20] Object Management Group. OMG System modeling Language (OMG SysML), v1.3. June 2012. Available at http://www.omg.org/spec/SysML/1.3/.

[21] Zamrini, Edward, et al. "Magnetoencephalography as a putative biomarker for Alzheimer's disease." *International journal of Alzheimer’s disease* 2011 (2011).

[22] Friedenthal, Sanford, Alan Moore, and Rick Steiner. A practical guide to SysML: the system modeling language. Morgan Kaufmann, 2014.

[23] Delligatti, Lenny. SysML distilled: A brief guide to the system modeling language. Addison-Wesley, 2013.

[24] Sattar, F., and C. Charayaphan. "Low-cost design and implementation of an ICA-based blind source separation algorithm." *ASIC/SOC Conference, 2002. 15th Annual IEEE International*. IEEE, 2002.

[25] Li, Zhongfeng, and Qiuhua Lin. "FPGA implementation of Infomax BSS algorithm with fixed-point number representation." *Neural Networks and Brain*, 2005. ICNN&B'05. International Conference on. Vol. 2. IEEE, 2005.

[26] Wang, Jia-Ching, et al. "VLSI Design for Convolutive Blind Source Separation." *IEEE Trans. on Circuits and Systems* 63.2 (2016): 196-200.

[27] Gupta, Suyog, et al. "Deep learning with limited numerical precision." International Conference on Machine Learning. 2015.

[28] Karamizadeh, Sasan, et al. "An overview of principal component analysis." *Journal of Signal and Information Processing* 4.03 (2013): 173.

[29] Shahrouzi, S. Navid, and Darshika G. Perera. "Dynamic partial reconfigurable hardware architecture for principal component analysis on mobile and embedded devices." *EURASIP Journal on Embedded Systems* 2017.1 (2017): 25.

[30] Chen, Tao, et al. " Real-time MEG data-processing unit for online medical imaging and brain-computer interface: a model-based approach." *International journal of bioelectromagnetism : IJBEM* 209.1 (2018): 39-42.

[31] A. J. Bell and T. J. Sejnowski, “An information-maximization approach to blind separation and blind deconvolution,” *Neural Comput*., vol. 7, no. 6, pp. 1129–59, Nov. 1995.

[32] Stephens, Michael A. "Use of the Kolmogorov-Smirnov, Cramér-Von Mises and related statistics without extensive tables." *Journal of the Royal Statistical Society. Series B (Methodological)* (1970): 115-122.

[33] Smirnov, Nickolay. "Table for estimating the goodness of fit of empirical distributions." *The annals of mathematical statistics* 19.2 (1948): 279-281.

[34] Dammers, Jürgen, and Michael Schiek. "Detection of artifacts and brain responses using instantaneous phase statistics in independent components." *Magnetoencephalography*. InTech, 2011.

[35] Charoensak, Charayaphan, and Farook Sattar. "A single-chip FPGA design for real-time ICA-based blind source separation algorithm." *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. IEEE, 2005.

[36] Shyu, Kuo-Kai, et al. "Implementation of pipelined FastICA on FPGA for real-time blind source separation." *IEEE transactions on neural networks* 19.6 (2008): 958-970.

[37] Du, Hongtao, Hairong Qi, and Gregory D. Peterson. "Parallel ICA and its hardware implementation in hyperspectral image analysis." *Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks II*. Vol. 5439. International Society for Optics and Photonics, 2004.

[38] Du, Hongtao, and Hairong Qi. "An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images." *Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International*. Vol. 5. IEEE, 2004.

[39] Huang, Wei-Chung, et al. "FPGA implementation of 4-channel ICA for on-line EEG signal separation." *Biomedical Circuits and Systems Conference, 2008. BioCAS 2008. IEEE*. IEEE, 2008.

[40] Van, Lan-Da, Di-You Wu, and Chien-Shiun Chen. "Energy-efficient FastICA implementation for biomedical signal separation." *IEEE transactions on neural networks* 22.11 (2011): 1809-1822.

[41] Yang, Chia-Hsiang, Yi-Hsin Shih, and Herming Chiueh. "An 81.6 uW FastICA Processor for Epileptic Seizure Detection." *IEEE transactions on biomedical circuits and systems* 9.1 (2015): 60-71.

[42] Shih, Wei-Yeh, et al. "An effective chip implementation of a real-time eight-channel eeg signal processor based on on-line recursive ica algorithm." *Biomedical Circuits and Systems Conference (BioCAS), 2012 IEEE*. IEEE, 2012.

[43] Roh, Taehwan, et al. "A wearable neurofeedback system with EEG-based mental status monitoring and transcranial electrical stimulation." *IEEE transactions on biomedical circuits and systems* 8.6 (2014): 755-764.

[44] Kuriakose, Jini, and Jayan K. George. "Multilevel Power Optimization for ICA Processor."

[45] Kim, Chang-Min, et al. "FPGA implementation of ICA algorithm for blind signal separation and adaptive noise canceling." *IEEE Transactions on Neural Networks* 14.5 (2003): 1038-1046.

[46] Chen, Tsan-Yu. A System-on-Chip Design of 32-Channel EEG Acquisition System with Automatic Artifacts Rejection. MS Work. National Chiao Tung University, 2016. Web. 20 Jun. 2018.

[47] Florin, Esther, Elizabeth Bock, and Sylvain Baillet. "Targeted reinforcement of neural oscillatory activity with real-time neuroimaging feedback." *Neuroimage* 88 (2014): 54-60.

[48] Hesse, C., et al. "On the development of a brain-computer interface system using high-density magnetoencephalogram signals for real-time control of a robot arm." (2007).

[49] Basit-Ur-Rahim, Muhammad Abdul, Fahim Arif, and Jamil Ahmad. "Modeling of real-time embedded systems using SysML and its verification using UPPAAL and DiVinE." *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on*. IEEE, 2014.