

Distributed Radio Telescope as Massively Parallel Computing System: A Case Study

Laurențiu Nicolae
Leiden Institute for Advanced
Computer Science
Niels Bohrweg 1
2333 CA Leiden, Netherlands
nld@liacs.nl

Sylvain Alliot
Netherlands Foundation for
Research in Astronomy
Oudehoogeveensedijk 4
7991 PD Dwingeloo, The
Netherlands
alliot@astron.nl

Ed Deprettere
Leiden Institute for Advanced
Computer Science
Niels Bohrweg 1
2333 CA Leiden, Netherlands
edd@liacs.nl

ABSTRACT

A giant distributed radio telescope is an unusual kind of massive parallel computing system. The design of such a system is a multidisciplinary effort that is hard to accomplish within a reasonable period of time, and with a high level of confidence that the resulting system will ultimately be what it has to be, now and in the course of its entire life span. Issues like deriving specifications from sometimes incomplete and conflicting requirements, exploring alternatives in behaviors and organizations, and answering *what if* questions in an interactive way are not easily mastered without having a methodology and a framework in which all these crucial aspects of a design trajectory can be integrated. There is still a large gap between what that methodology and this framework should be and what the current approaches are. In an attempt to shed more light on that gap, a few telescope concepts, like the Low Frequency Phased Array (LOFAR), and the Square Kilometer Array (SKA) telescopes that have been proposed to be built in the coming decades have been analyzed, and down-scaled versions of these telescopes have been taken through an iterative exploratory specification process with the objective of identifying what the methods in the methodology and the tools in the framework should be for the design of the giant telescope to be successful. The paper reports on one of these case studies which is the Thousand Element Array (THEA), a down-scaled version of the SKA radio telescope.

Keywords

embedded systems design, design space exploration, system level modeling and simulation

1. INTRODUCTION

Designing a large scale system such as a massively parallel computing system, in particular a distributed real-time system, can be done in several ways. One approach is to design a small version prototype and, when successful, scaling that prototype up in a number of consecutive steps. This approach does not always work because the smaller prototype may not be scalable. Another way is to design

the large scale system itself, reusing as much as possible parts or components that have been designed and worked with before. This approach does also not always work, because if there is not much to be reused, then the design problem is too complex and many decisions are taken without knowing well what the consequences will be later on. To overcome this problem a new approach is advocated in this paper: dividing the design problem into two major steps. The first step is an iterative exploration step in which requirements and constraints are translated to specifications. The second step translates specifications into implementations. This paper focuses on the first step. The requirements and constraints are provided by the end-users. They express the user's expectations. For example, the user may require that the system (distributed radio telescope in our case) must be calibratable, have a particular sensitivity, and be capable of operating in a number of modes. Number and distribution of antennas, necessary processing of signals and trade-off between distributed and central processing are all issues that follow from the requirements and constraints and that are part of the specification.

The specification is expressed in terms of high-level (abstract) models. There are three models involved. The first one is a behavior model that consists of a number of partial behaviors and the interrelations between them. It is a possible decomposition of the system behavior that is obtained through an iterative, interactive and exploratory process that is based on metrics, such as the sensitivity of the requirements with respect to changes in the decomposition. The second model deals with the organization of the system and focuses on cost and performance. This model consists of a number of components and their interconnections. The third model relates the other two models and deals with possible mappings of the behavior into the organization. Part of the constraints or boundary conditions may be that certain entities in the model should be taken from a library. These entities must be so specified and described that they can be blindly incorporated in the models with a high level of confidence. The three models together constitute the system specification (or model). Included is the requirements-to-specification phase is a top-down and bottom-up flow of information which results from two processes that take place at a number of levels of abstraction. The first process is an exploration process, the second one is a refinement process. These are executed on *critical* subsystems, or parts of these. The exploration process explores different alternatives on a given level of abstraction. The refinement process takes a selected alternative one abstraction level down for a closer analysis.

We are currently putting exploration and refinement tools together in a framework in which the requirements-to-specification process can be performed and questions such as: *What if we modify this requirement?* The construction of the framework is succeeding a case study in which the process has been prototyped and validated. The case has been a down-scaled version of the SKA radio telescope, called the THousand Elements Array (THEA). For THEA, the system specification (the three models mentioned above) has been derived and simulated. The organization of the system was built on library blocks that themselves are defined in terms of components. At the lowest level of abstraction, behavior is mapped into components. At the level of abstraction above this one, compositions of behaviors are mapped on candidate library blocks that can support the corresponding component compositions. Higher levels of abstraction are the sub-subsystems or subsystems of THEA as they have been identified during the decomposition phase, starting at the highest level where the requirements and constraints are defined. The organization at the higher levels of abstraction is in terms of compositions of blocks. The blocks come in families and are so chosen that the system is scalable, that is, the SKA telescope can be specified in terms of these blocks as well. The components and blocks are not necessarily the ones that have to be used in the subsequent specification-to-implementation process. However, they are so chosen that the derived specification can indeed be implemented to get a prototype if so desired. This prototype has, in fact, been built - after the requirements-to-specification process was terminated - to demonstrate that the final specification of the system is sufficiently accurate that it can satisfy the requirements and constraints given by the end-users.

The rest of the paper is organized as follows. Section 2 is devoted to the THEA system requirements and constraints. Section 3 deals with the high-level system model and its exploration. Section 4 concludes the paper.

2. THE THOUSAND ELEMENTS ARRAY

The principles of phased array antennas were tested on the OSMA [2] system at Astron for 64 antenna elements. This study guided the THEA system specification and simulation. Being a small size telescope prototype, THEA had to be specified in a scalable way. Whatever the requirements and constraints that the end-user may provide, every radio telescope receives three types of signals, two of which have to be removed anyway. The first signal type is the signal of interest from the sky. The second signal type is the ubiquitous noise signal that may have a power level that is as high as 70 dB above the signal of interest. The third signal type is the strong man-made radio frequency interference signal that may have a power level as high as 70 dB above the noise level. Interference signals can be mitigated by means of so-called adaptive beamformers, noise signals can be suppressed by means of correlation techniques. Thus, a first rough model of THEA will be as shown in figure 1.

2.1 Requirements and constraints

The main requirements for a telescope specification are the observation modes and their frequency range, the sensitivity of the instrument, the calibration and on-line processing requirements, and the data reduction requirements. Top-level specification parameters are derived from these requirements. For example, the sensitivity of the instrument scales with the square root of the signal bandwidth B and the number of antenna n_a . Required signal processing speed and I/O rates are also linearly dependent on these factors. Number of RFI sources in the environment, RFI mitigation

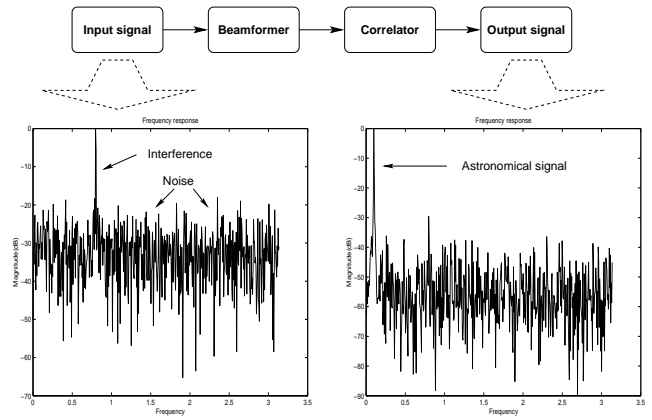


Figure 1: A non-constrained THEA model: the signals collected from the antennas are processed by an adaptive beamformer that detects and removes a spatial distribution of the RFI signals. A subsequent correlation extracts the astronomical signal from the surrounding noise

techniques, and system noise have all critical impact on the telescope performance. Typical top-level parameters are as shown in table 1 below.

A coarse and simple computational model, based only on a certain measure of computational complexity (such as the number of multiplications), can be derived from the top-level parameters. Interconnection complexity and other mapping parameters are excluded from this evaluation. As an example, table 2 contains the top-level computing rates of the beamformer for two of the possible techniques: hierarchical beamforming and Fourier beamforming at a certain level k in the hierarchy. These techniques are described for THEA in [3]. The constraints of a particular signal processing technique for data distribution will influence the detailed design of the processing nodes.

2.2 THEA data and control flow requirements

From the coarse computing requirements of the beamformer we can estimate also the complexity of the control flow and derive interconnection rates and real-time requirements.

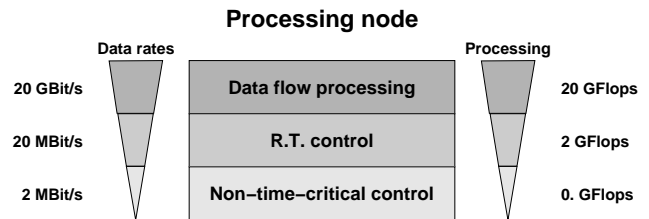


Figure 2: Beamformer data and control flow requirements

In figure 2 the processing requirements and module exchange rates are specified for a data flow processing layer, a real-time layer and a non-time-critical control layer. Data flow and control flow are strictly separated.

The signal processing layer performs massively parallel spatial filtering computing that will most likely have to be mapped into organizational block containing components in which distributed systolic like networks can be configured for efficient execution. Cost

Table 1: Top-level parameters for the THEA DSP back-end specification

| station | n_s | n_a | r_0 | n_p | n_c | n_B | n_b | n_{bB} | n_{sc} | t_i | n_{RFI} | t_u |
|---------|-------|-------|-------|-------|-------|----------|-------|----------|----------|-------|-----------|-------|
| THEA | 1 | 16 | 40 | 1 | 2 | n_c 16 | 12 | 12 | 1024 | 1 | 4/20 | 10 |

n_s -Number of station
 r_0 -Input beam sampling rate (MHz)
 n_p -Number of polarization
 n_b -Input sample word length
 n_{sc} -Number of spectral channels
 n_{RFI} -Max number of RFI (MHz)
 n_a -Number of antenna (or pre-formed beams) per station
 n_c -Number of independent band or RF beams
 n_B -number of digital beams formed
 n_{bB} -Output beam sample word length
 t_i -Integration time (ms)
 t_u -Update rate (ms)

Table 2: Coarse computing rates for the DBF

| Function | Formula | Computing rate (Giga Complex MAC/s) |
|-----------------|--------------------------------------|-------------------------------------|
| Hierarchical BF | $b_B(k-1)n_c n_B(k)n_a(k)r_0(k)$ | 10.24 |
| Fourier BF | $b_B(k-1)n_c(12\log_4 n_a(k))r_0(k)$ | 15.36 |

and performance numbers can be obtained by mapping parts of such networks into FPGAs and extrapolating these low level cost and performance numbers to obtain the numbers at the higher levels.

The real time control layer performs the necessary computations to update the spatial filter parameters at a rate that is lower than the processing rate in the signal processing layer. From the top-level parameters derived from the requirements, it follows that this processing could be mapped into a block containing components for extensive memory access and ISA like computation. Updating the filter weights and filtering run with tied real-time constraints.

The non-time-critical control consists of a low information rate to the front end of the telescope and the system scheduler. Connecting this processing control to a cluster of PCs is a compact and cost efficient solution for power supply, network connection, storage and acquisition.

3. SYSTEM MODEL EXPLORATION

Given the requirements and constraints for the THEA system, the top-level parameters, and the rough non-constrained THEA model of figure 1, we propose an exploration strategy that is based on the decomposition and refinement downwards from this level of abstraction, the analysis of alternatives at various levels of abstraction - at least for the critical parts encountered at these levels, and a subsequent composition of behaviors and organizations in the opposite direction of the levels of abstractions, including validations. The end result is an accurate specification of the system, including cost and performance numbers. This specification is to be taken as an input for a subsequent design and implementation phase. The whole process is schematically specified in figure 3.

A few specific steps in the process may illustrate the concept and methodology. Before giving them, we first present the end result, which is the system specification model. For THEA this model is shown in figure 4.

In this figure, the upper part is the model that specifies the behavior. It is an untimed network consisting of processes that are interconnected through unbounded FIFO channels with blocking read synchronization (a so-called Kahn Process Network [5]). The beamformer in the model of figure 1 has already been refined into a DBF process (special filtering) and an AWE process (weight updating). A select process has been identified because of the requirement that

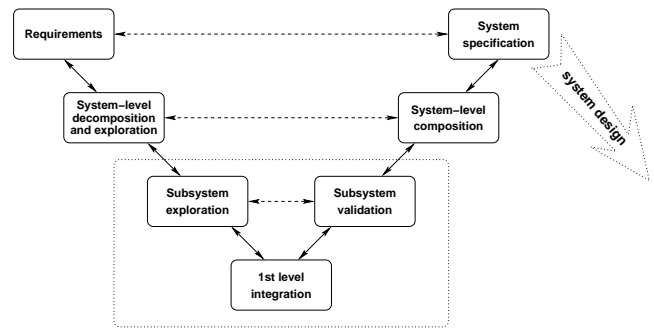


Figure 3: The proposed exploration strategy

multiple beams have to be provided simultaneously, some of which have to be selected depending on the selected mode of operation. The lower part in the figure is the model that specifies the organization of the system. It consists of blocks that are taken from a library of pre-specified blocks. The *Matrix shuffling* block is a composition of FPGA-like components. The *Control processing* unit is an ISA-like component that, together with the accompanying *Memory* component, constitutes another block from the library. The dotted lines connecting the two models represent the mapping model. This sort of modeling is known as the Y-chart modeling methodology [6]. This approach allows us to keep the behavior (or application) and the organization (or architecture) separated, for a plus of flexibility. By analyzing simulations, one can modify either the application, or the architecture or the mapping and reiterate the process.

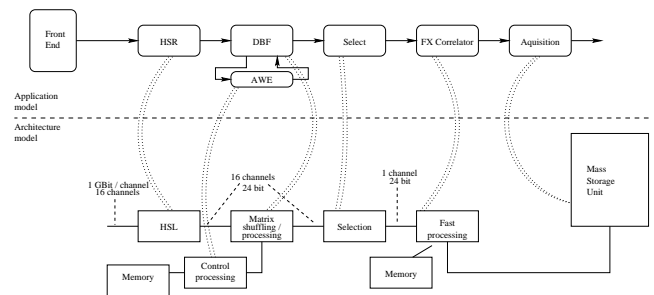


Figure 4: The THEA specification model

The system specification model shown in figure 4 has been simu-

lated in the *Sesame* framework created at the University of Amsterdam [8]. Some simulation results are displayed in table 3.

This table shows that the real-time constraints could not be met in two simulations (intersection of column four and rows one and three). This was due to the fact that the *fast processing* block that was selected from the library did not contain enough memory. Notice the short simulation times for the THEA system.

We are now ready to illustrate two typical steps in the exploration cycle. One step concerns behavior exploration (also called application exploration), the other one deals with organization exploration (also called architecture exploration).

3.1 Application exploration

Recalling that only critical parts are explored and possibly refined, we focus on the adaptive beamformer which is, indeed, a very critical part. No block in the library can execute the adaptive beamforming process in real time. Therefore the adaptive beamforming application is split in two processes that can run concurrently: the DBF (space filtering) process and the AWE (adaptive weight estimation) process. The specification of the two processes is shown in figure 5.

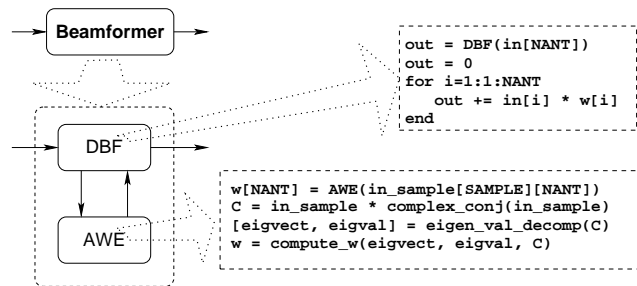


Figure 5: The THEA beamformer

The DBF process is a set of complex valued inner products that are required to run concurrently at a very high repetition rate. The AWE process that has to update the weight vectors in the DBF process at a relatively low rate is an eigenvalue decomposition of a covariance matrix built on the multiple antenna samples. Both specifications can be refined in various ways. To explore the alternatives, we use a toolset called Compaan [9] that allows us to transform automatically a given nested loop algorithm written in Matlab into a Kahn process network [5] specification. This toolset uses a few aggressive techniques, such as array data-flow analysis, data partitioning and linearization, in order to expose the inherent parallelism of the analyzed algorithms [1]. If the results of the transformations applied to the original algorithm are not as good as expected, the Compaan toolset offers the possibility to derive a set of alternative process networks. This can be achieved by applying algorithmic transformation techniques, such as skewing and unfolding [10], to the original application to expose various degrees of concurrency available in the given algorithm.

Moreover, the process networks derived by Compaan can be converted into synthesizable VHDL and implemented into an FPGA [4] to extract realistic cost and performance numbers for the given Kahn network. These numbers can then be imported in the higher-level model for simulation purposes at higher levels of abstraction. Tools that we use to simulate and explore at higher levels of abstraction are Spade [7] and Sesame [8].

3.2 Architecture exploration

The telescope application is distributed in space with remote antennas delivering a sustained data flow. For optimum use of the high speed link bandwidth, cost and power consumption, point-to-point (optical fiber) connections are the optimum choice. Such connections guarantee sustained input rates and low transmission errors. The optical fibers are also used to distribute a central sampling clock and synchronization information to the antennas. These choices originate from the user defined requirements and are part of the pre-defined boundary conditions for the organization exploration. They also take scalability requirements into account. Indeed, the same principles can also be used for the distribution of the data flow over clusters of processing nodes (THEA clones). The processes scheduling is then derived from the synchronization signals and the central clock. The processing and communication blocks in the library are specified in such a way that the high throughputs can be sustained with moderate to complex data processing, that interconnections can be made in a correct-by-construction way, and that upscaling of the system does not require blocks other than those in the library. The mapping of behavior to organization is a mapping of processes to blocks. Notice that although the application exploration is possibly going down to a mapping in FPGAs for calibration purposes, the final mapping has to make use of the library blocks which are not necessarily composed of interconnected FPGAs. However, these calibration numbers are useful to support and guide the organization or architecture exploration which is essentially a block selection process. The blocks that were specified and described for the THEA case study are reprogrammable/configurable and scalable. They allow for a great variety and scalable organization. An example of (part of) a large scale system built on the THEA blocks is shown in figure 6.

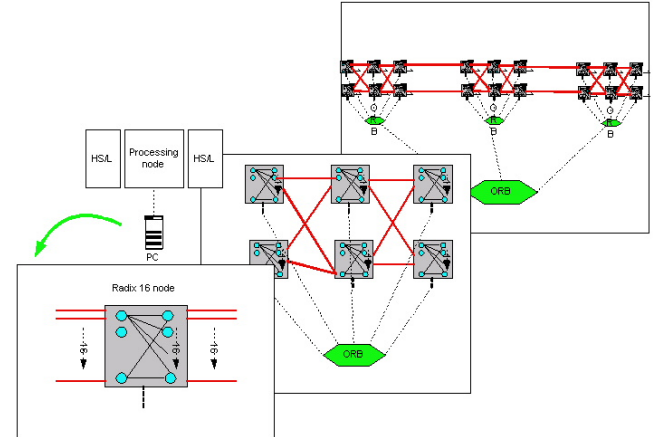


Figure 6: Network with processing node interconnections at a large antenna station scale.

This figure illustrates the deployment of processing nodes attached to a computer from a cluster. The high speed link connections via optical fibers on dedicated processing nodes allows a large variety of configurations, with inter block connections arbitrated by an object request broker. Clusters of nodes can virtually be expanded. The generic design of the THEA blocks with re-programmable and scalable configurations allows large architecture deployments.

4. CONCLUSIONS

We have presented a case study in deriving specifications from requirements and constraints. An iterative, interactive exploration methodology to obtain specifications from requirements was tested

Table 3: Experimental results

| Model | Samples | FX Mem. | sel2fx stalls | Appl. time | Arch. time | Total time |
|-------|-------------|---------|---------------|-------------|------------|-------------|
| THEA | 128*1024 | none | 25.95% | 16.93 sec | 7.30 sec | 24.23 sec |
| THEA | 128*1024 | 1024 | 0.00% | 16.90 sec | 7.32 sec | 24.22 sec |
| THEA | 12.800*1024 | no | 33.03% | 1690.33 sec | 728.38 sec | 2418.71 sec |
| THEA | 12.800*1024 | 1024 | 0.00% | 1548.15 sec | 739.74 sec | 2287.89 sec |

and validated for a down-scaled version, called THEA, of a giant distributed radio telescope, called SKA. The methodology explores alternative decompositions of behaviors at various levels of abstraction, and corresponding alternative decompositions of organizations, where behavior is specified as a network of processes and the organization is specified as interconnected library blocks. The resulting specification is scalable, up to the size of the conceived giant telescope. The resulting specification can be taken to a subsequent design and implementation phase. This phase has not been tested for the THEA system. Instead, the specification has been implemented as such (using implementations of the library blocks). This provides a hands-on prototype for conducting user experiments. Because this prototype is a direct implementation of the specification, it can be used to convince the users that it meets the requirements, and that the specification of the upscaled system will do as well without having to implement it this way. Although the case study has been conducted strictly according to the proposed methodology, it was partially achieved by hand so that only a restricted number of alternatives have been explored. We are currently constructing a framework in which the various tools are integrated in such a way that many more alternatives can be explored, and answers to *what if* questions can be obtained in a reasonable time span.

The requirements-to-specification phase is to be succeeded by a specification-to-design and implementation phase. The system design and implementation will proceed in a way that is very similar to the requirements-to-specification methodology and will definitely be guided by that process. However, the ultimate system implementation will most likely use different implementations of the various blocks that were used in the specification process. The reason for that may be that the design and implementation process is performed later in time and that technology advances offer implementation options that were not available at the time the specification process was executed. The blocks used in the latter processes are also chosen with an eye on evolving technology.

5. REFERENCES

- [1] E. Deprettere, E. Rijpkema, P. Lieverse, and B. Kienhuis. High level modeling for parallel executions of nested loops algorithms. In *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'2000)*, Boston Massachusetts, USA, July 10 2000.
- [2] G. A. Hampson and B. A. Smolders and A. Joseph and H. Heutink and A. Doorduyn and K. Dijkstra. One Square Metre of a Million. *29th European Microwave Conference*, October 4-8 1999.
- [3] G.A. Hampson and R. de Wild and A.B. Smolders. Efficient Multi-Beaming for the Next Generation of Radio Telescopes. *Perspectives on Radio Astronomy: Technologies for Large Antenna Arrays*, pages 265–276, April 12-14 1999.
- [4] T. Harriss, R. Walke, B. Kienhuis, and E. Deprettere. Compilation from Matlab to process networks realized in FPGA. In *35th Asilomar conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, Nov.4-7 2001.
- [5] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. of the IFIP Congress 74*. North-Holland Publishing Co., 1974.
- [6] B. Kienhuis, E. Deprettere, P. van der Wolf, and K. Vissers. A methodology to design programmable embedded systems. In *SAMOS: Systems, Architectures, Modeling, and Simulation*, Nov. 2001.
- [7] P. Lieverse, T. Stefanov, P. van der Wolf, and E. Deprettere. System level design with spade: an m-jpeg case study. In *Int. Conference on Computer Aided Design (ICCAD'01)*, San Jose CA, USA, Nov. 4-8 2001.
- [8] A. Pimentel, P. Lieverse, P. van der Wolf, L. Hertzberger, and E. Deprettere. Exploring embedded-systems architectures with Artemis. *IEEE Computer*, 34(11):57, 2001.
- [9] E. Rijpkema, E. Deprettere, and B. Kienhuis. Deriving process networks from nested loop algorithms. *Parallel Processing Letters*, 10(2 - 3):165–176, 2000.
- [10] T. Stefanov, E. Deprettere, and B. Kienhuis. Exploring application model instances in system-level design. In *Proc. 2001 Workshop on Embedded Systems (PROGRESS'01)*, Valthoven, The Netherlands, Oct.18 2001.