How Rapid is Rapid Prototyping? Analysis of ESPADON Programme Results

Bob K. Madahar

BAE SYSTEMS Advanced Technology Centre, West Hanningfield Road, Gt. Baddow, Chelmsford CM2 8HN, UK
Email: bob.madahar@baesystems.com

Ian D. Alston

BAE SYSTEMS Advanced Technology Centre, West Hanningfield Road, Gt. Baddow, Chelmsford CM2 8HN, UK
Email: ian.alston@baesystems.com

Denis Aulagnier

Thales Airborne Systems, 10 avenue de la 1ère DFL, 29283 Brest Cedex, France
Email: denis.aulagnier@fr.thalesgroup.com

Hans Schurer

Thales Naval Nederland, Zuidelijke Havenweg 40, P.O. Box 42, 7550 GD Hengelo, The Netherlands
Email: hans.schurer@nl.thalesgroup.com

Mark Thomas

Thales Underwater Systems, Dolphin House, Ashurst Drive, Bird Hall Lane, Cheadle Heath, Stockport, Cheshire, SK3 0XB, UK
Email: mark.thomas@uk.thalesgroup.com

Brigitte Saget

MBDA France, 20-22 rue Grange Dame Rose, BP 150, 78141 Velizy-Villacoublay Cedex, France
Email: brigitte.saget@mbda.fr

New methodologies, engineering processes, and support environments are beginning to emerge for embedded signal processing systems. The main objectives are to enable defence industry to field state-of-the-art products in less time and with lower costs, including retrofits and upgrades, based predominately on commercial off the shelf (COTS) components and the model-year concept. One of the cornerstones of the new methodologies is the concept of rapid prototyping. This is the ability to rapidly and seamlessly move from functional design to the architectural design to the implementation, through automatic code generation tools, onto real-time COTS test beds. In this paper, we try to quantify the term "rapid" and provide results, the metrics, from two independent benchmarks, a radar and sonar beamforming application subset. The metrics show that the rapid prototyping process may be sixteen times faster than a conventional process.

Keywords and phrases: rapid prototyping, COTS, model year, beamformer, EDA tools, heterogeneous platform, FPGA.

1. INTRODUCTION

The trinational European EUCLID/Eurofinder defence project called ESPADON (environment for signal processing application development and rapid prototyping) completed in September 2001 [1]. The ESPADON consortium comprised Thales and MBDA from France, Thales Naval Nederland, BAE SYSTEMS, and Thales Underwater Systems Ltd. from the United Kingdom. The primary objective of the three-years project was to significantly improve (reduced cost and timescales) the process, by which complex military digital processing systems are designed, developed, and supported. A new design methodology and supporting development environment has been reinvented to support this aim through reuse, concurrent engineering, rapid insertion of COTS technology and the key concepts of rapid and virtual

FIGURE 1: Iterative development process.

prototyping. The latter two concepts are an integral part of the model-year concept adopted by ESPADON and developed under the US RASSP programme [2].

A brief summary of these techniques and developments within ESPADON,[1] in the context of rapid prototyping (RP), is presented below.

1.1. The methodology

A risk-driven iterative development process has been identified. This is shown in abstract form in Figure 1 and is underpinned by the following 5-key processes stemming from the methodology MCSE (méthodologie de conception de systèmes electroniques) from IRESTE, Nantes [3].

  (i) Specification—refinement of the requirements into an engineering specification.
 (ii) Functional design—the functional parts of the component specifications are modelled and simulated and proven for correctness as a whole model. The model is independent of the implementation.
(iii) Architectural design—the critical characteristics of the reference functional model (computing power, rate, etc.) and the nonfunctional requirements (costs, volume, etc.) are identified. Cost/performance trade off studies are carried out and the most effective architecture is chosen.
 (iv) Implementation—the result of the current design iteration. Essentially the production and test of hardware and software, integration of the software on the target COTS platform, and validation of the component.

 (v) System review—the final process which determines whether the particular phase of the system development has met its requirements and ameliorated the major risks before proceeding to the next phase or iterating around the same phase again.

Each of the key processes above is itself composed of the generic abstract iterative process shown in Figure 2. This again is a risk-driven process where the risks are analysed and a plan formulated, the work defined, the developments undertaken, the results validated and the complete outcomes reviewed by the *exit* or *refine* review.

A spiral [4] can also represent the iterative development process described above as shown in Figure 3. Each turn of the spiral corresponds to one process. For each process, the same types of activities are carried out. The enlargement of the spiral at each process represents the refinement and the increase in the artefacts produced.

The overall aim of the methodology is to enable the developer to rapidly iterate to the final solution for the particular system development being undertaken. In the case of RP, it is to rapidly and seamlessly move from functional design, to the architectural design and finally to the implementation, through automatic code generation tools, onto real-time COTS test beds. This enables real behavioural and performance measurements to be made so as to refine the functional model and the architectural design solution to satisfy the system requirements.

1.2. Reuse and capitalisation

Reuse, alongside the iterative development process, is the other element of the signal processing methodology implemented to decrease development time and cost. Reuse applies at two levels.

(a) Reuse between iterative processes of development

---

[1]The ESPADON programme was presented at the Embedded Systems Show, Rapid and Virtual Prototyping Technical Seminar, London, 17th May 2001. The ESPADON programme and the ESPADON benchmarks results are presented in detail in the ESPADON website: www.espadon.org/.
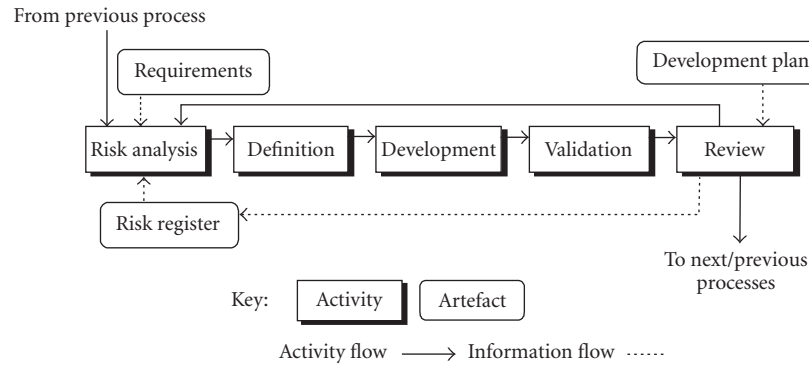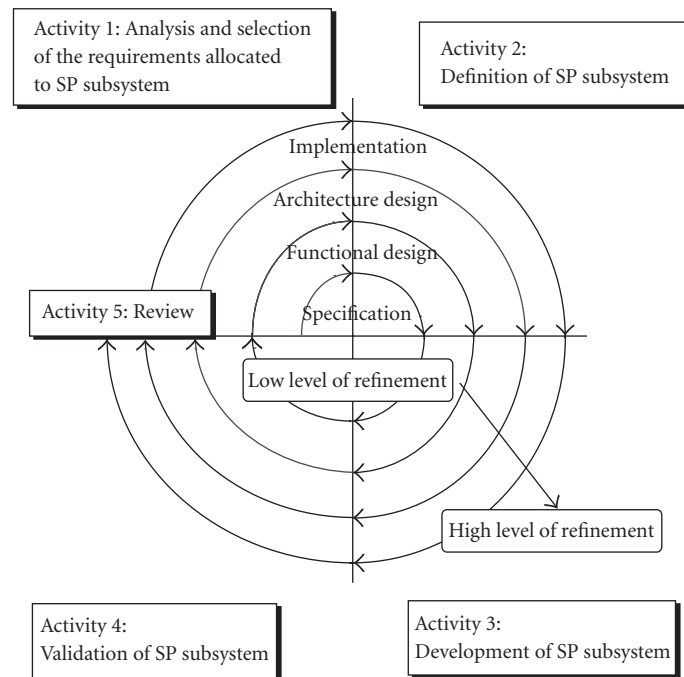
FIGURE 2: Abstract iterative process.

FIGURE 3: Spiral view of the development lifecycle.

cycle—use elements developed in an iterative process with a certain level of refinement for the development of the next iterative process having a higher level of refinement. The development strategy with reference to the abstract iterative process is

(i) definition activity—the same modelling formalisms or functional models are used at different levels of refinement but with dual libraries of components,

(ii) development activity—hardware is synthesised and code is generated for different target machines with the same synthesis techniques. These targets may be, for example, a workstation or a real-time multiprocessor machine according to the development stage,

(iii) validation activity—the stimulation or the results obtained from the previous iterative process

are used as a reference test set for the validation of the next iterative process.

(b) Reuse of existing components (SP algorithms, components, hardware architectures, etc.)—use in-house components already developed, or COTS components, for the development of an activity (or an iterative process) of the development cycle. The development strategy is

(i) development with reuse—development of an application must be able to reuse already developed existing constituent parts,

(ii) development for reuse (or capitalisation)—the new constituent parts of an application are developed in order to be reused in other systems.

The above reuse objectives are integral to the ESPADON development process and enables
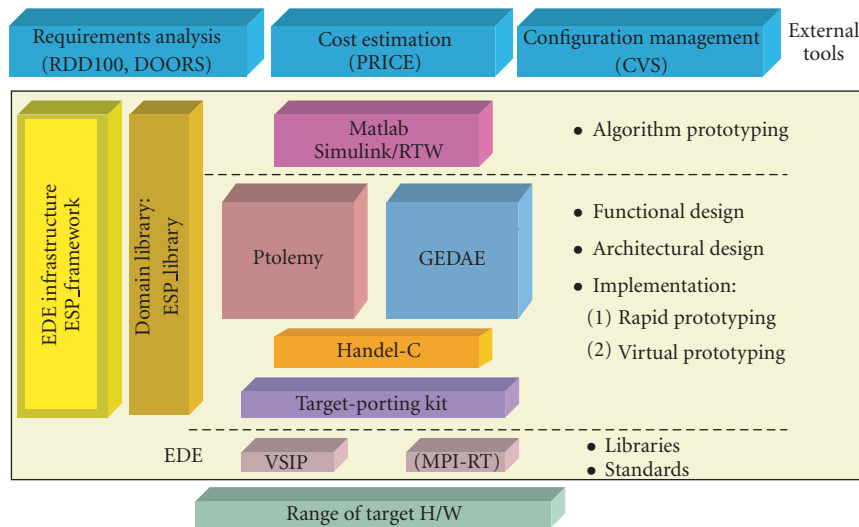
FIGURE 4: ESPADON design environment.

(a) increasing productivity and decreasing development time,

(b) providing additional architecture choices,

(c) using better quality constituent parts since they have already been tested and validated,

(d) capitalising on existing know-how.

1.3. The overall environment

An integrated software design environment, the ESPADON design environment (EDE), was developed to support the methodology and reuse and capitalisation policy described above. It is based on a collection of COTS software tools that were selected as the most suitable after a detailed review and evaluation of many commercial EDA (electronic design automation) tools. This environment is shown in Figure 4 below. Two of the tools, Ptolemy Classic [5] and GEDAE, Blue Horizon Development Software, Mount Laurel, NJ, USA, http://www.gedae.com/, form the core of the environment as they fully support and are integral to the concept of RP. Handel-C [6, 7, 8] was used for FPGA hardware synthesis to provide an analogous process route to programmable logic as to microprocessors. The domain library (ESP_Library) contains the reused elements for an SP application domain (radar, sonar, etc.). This library is based on the vector scalar image processing library (VSIPL) standard, http://www.vsipl.org/.

The radar and sonar benchmarks of the RP process were performed using the environment and Ptolemy Classic and GEDAE, respectively.

2. THE BENCHMARKS

The choice of signal processing applications for the benchmark was arrived at by considering two factors. One was that it must be a common application for the two domains,

radar and sonar, so that the conventional development process and associated metrics are known, or can be confidently estimated. The second was that a common subset of the application exists for both domains, for cross comparison, and is of small but sufficient size (functionality and development effort) to enable the benchmark measurements to be made within the time and effort available and to be acceptable.

Beamforming, the processing of sensor signals into directional beams, was selected as the application subset that would be benchmarked. It is a generic processing function for both domains and satisfies the selection criteria. The functional processing chain for the radar and sonar beamformer benchmarking applications is described in the next two sections.

2.1. Radar

The application subset is from a ship-based X-band air surveillance radar with a vertical array of, for example, eight transmit and receive elements. Each element is a horizontal linear stripline array of dipoles. Elevation beams are formed by the digital beamformer that performs an 8-point FFT algorithm on the outputs of the eight receiver channels. In this way, a multibeam receive system is formed, Figure 5. The benchmark concerns only the receiver beamforming function, the transmit beamforming function is implemented by an analogue system.

The beamformer is adaptive with respect to the ships course and speed, and the ships roll and pitch movement. This results in a phase correction that is applied to the complex data stream prior to the beamforming, together with windowing and calibration correction. The functional processing chain is shown in Figure 6.

The boxes shown as calibration and beam calculation were developed as part of the RP process using Ptolemy Classic. The other functions, except for the functions in the dashed boxes that were not used, were primarily for I/O and
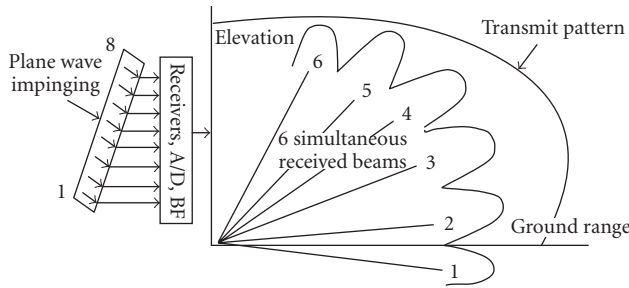
FIGURE 5: Multibeam receive system.

resident on the host machine and part of the stimuli generator and/or display.

For a first RP implementation, the target platform was an embedded VME system from Mercury Computer Systems (http://www.mc.com/). This comprised three Mercury motherboards, each connected via interlink modules through two ports to the high-speed RACEway interconnect between the boards. One motherboard was equipped with two daughter boards with two Motorola power PC altivec processors (http://e-www.motorola.com/) each, and the two other motherboards with one daughter card with two altivecs and one daughter card with 512 MB memory each. This resulted in a machine with eight processing nodes and bulk memory that was used for real-time data playback of the (stored) stimuli generator data and for logging of the intermediate/output results of the application.

In a second RP implementation, the time-critical part of the beamformer application, that is, the beamforming function, was ported onto an FPGA board. The rest of the application remains mapped onto the Mercury boards. The FPGA board was an existing board from Thales Airborne Systems with two Xilinx Virtex XCV400 devices.

Within Ptolemy Classic, the radar benchmark application has first been modelled and simulated with synchronous data flow (SDF) [9] and Boolean data flow (BDF) [10] modelling formalisms. Then the code has been generated with the code generation C (CGC) computation domains [11]. A target porting kit (Figure 4) has been developed in Ptolemy Classic to generate, compile, load, and run the code of the power PCs and the FPGAs.

In addition, other specific enhancements had to be carried out in Ptolemy Classic to support the RP process and the benchmark [12].

(i) Additional library components; radar library—five components; VSIPL core light library—11 components; and support library (e.g., components for parallel operation)—19 components. These components were developed for SDF, CGC, and HDLC (see below) computation domains. The components and the domains were identified in the early Ptolemy benchmark developments as being required if the functional requirements of the benchmark were to be addressed.

(ii) Support for the VSIPL altivec power PC optimised library and static memory allocation of VSIP views during the initialisation phase of the target machine.

(iii) Support for performance monitoring using Mercury trace tools.

(iv) Extension of the BDF code generation domain to support use within a single processor which was part of a multiprocessor architecture modelled using the multiproc CGC domain. The rationale was to enable support and integration of control functions which are integral to most sensor platforms.

(v) Addition of a new code generation domain "HDLC" to generate code in the Handel-C language instead of C so as to map directly to programmable logic. Handel-C is a hardware design language very similar to C, but Handel-C has some specific functionality dedicated to the design of hardware components.

(vi) Support for heterogeneous architecture code generation [13]. The characteristics of the codes generated for the power PCs or the FPGAs are different, that is, different languages (C or HDLC), different memory allocation, and different communication drivers. Nevertheless, these two types of codes implement the same communication protocol to interface the two architectures.

Only manual partitioning was used for the mapping of the functional SDF model into the targeted architecture. Although automatic methods of partitioning exist within the Ptolemy tool, these rely on estimates of the execution time of each of the low level building blocks used to create the model at schedule time. Sufficiently accurate estimates were not available because:

(i) the building blocks used VSIP library functions where the size of data to handle, and hence its execution time, is only known at runtime,

(ii) obtaining accurate estimates is complicated due to the complex cache policy of the PPC and Mercury architecture.

However, using rapid prototyping tools like Ptolemy and GEDAE, the generation of a new mapping is very quick/simple and the feedback on execution time performance using the efficient execution tracing methods available with these tools produces far more accurate mappings than could be achieved with automatic methods. It is intended in the future to build databases that contain the domain library functions and the measured attributes of the functions (e.g., latency, performance) against hardware architecture attributes (e.g., processor type, memory speed, cache size, clock speeds). These could then be used to support the automatic mapping of functions to the primary processing nodes of the architecture. New COTS components which are suitable can be benchmarked using some of the domain library functions and can be added to the database and the results extrapolated for the remainder of the library as appropriate.
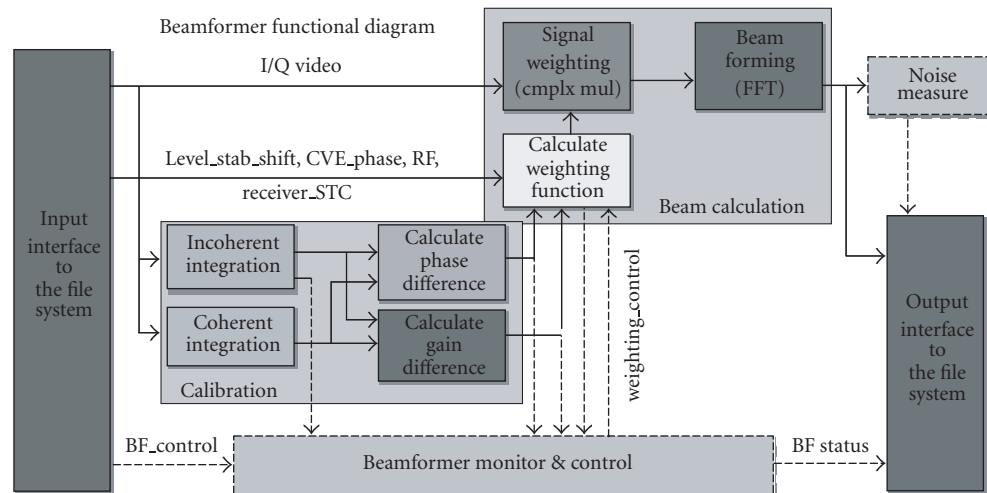
FIGURE 6: Radar functional processing chain.

2.2. Sonar

The application subset is from a generic ship/submarine-based active sonar system. It covers the core functionality of a conventional beamformer and an adaptive beamformer where the reverberation due to the propagation environment is estimated and cancelled. For a given sonar array heading and speed, the reverberant returns from any particular direction have an induced Doppler which depends only on the directions of transmission and reception, nonzero intrinsic Doppler being ignored. This means that all beams receive their reverberant noise in a given Doppler bin from the same direction. This includes the few beams, whose directions align with the direction of arrival of the reverberant returns being considered, which of course will have their zero Doppler ridges at the Doppler bin being considered. In fact they are these few beams which are used in the adaptive algorithm as reference beams to sense the reverberation at the Doppler bin in question, and cancel it from all other beams.

The overall functional processing chain (top diagram) and the beamformer functions (bottom diagram) are shown in Figure 7. The latter were developed for the RP benchmark and interfaced to other functions, primarily for I/O, such as the scenario generator and/or display resident on the host machine.

The sonar benchmarking application was developed within the RP process using GEDAE with an initial and final target embedded VME systems, each connected to a sun workstation. The initial embedded system was from Sky computers (http://www.skycomputers.com/) and was used to develop the benchmark application functions. This platform consists of a force SPARC CPU50 host machine and two-quad power PC altivec processor (Merlin) processing boards connected together by the backplane Sky channel interface. The final system was a subset developed on the EUROPRO project [14] and was used to complete all the sonar benchmark measurements. This platform was chosen because it is based on different processors to the other ESPADON target platforms and would demonstrate the hardware independence of the RP process. It has a number of DBV66 boards, Blue Wave Systems DBV66 ADSP-2106x Carrier Board Technical Reference Manual, Blue Wave Systems (http://www.bluews.com/) each with six analog devices "SHARC" DSP processors, Analog Devices (http://www.analogdevices.com/technology/dsp/). In the basic configuration, Figure 8, there are two DBV66 cards connected via the VME to the host board for code load. All further communication between SHARC processors are performed using point-to-point SHARC links which are capable of MB/s data transfer rates. Multiple DBV66 boards are connected using the internal SHARC links to support larger SHARC networks.

A target porting was developed for this platform for GEDAE. It is built on underlying support software for the platform, such as the blue wave IDE6000-V4.0, Blue Wave Systems (http://www.bluews.com/), the operating system Virtuoso 4.1-R2.04 for Solaris (http://www.windriver.com/) (formally owned & distributed by http://www.eonic.com/), and the analog devices compiler-V3.3 for Solaris. The overall EDE enabled the GEDAE data flow graphs to be developed for the benchmarking functions across a number of platforms, including the host, by different users and simply imported and/or ported to the benchmarking platform.

The key functions developed for the benchmark were

(i) data generator—read data in from a file that has been generated from the stimulator. The simulator provides hydrophone data (128 hydrophones) of complex baseband reverberant time series data across an array of specified elements;

(ii) control interface—control box responsible for defining parameters that would possibly be controlled through the operator/MMI. Examples are Dolph Chebychev coefficients, beam shading coefficient, FFT size, minimum number of Doppler bins, energy thresholding factor, and integration period;
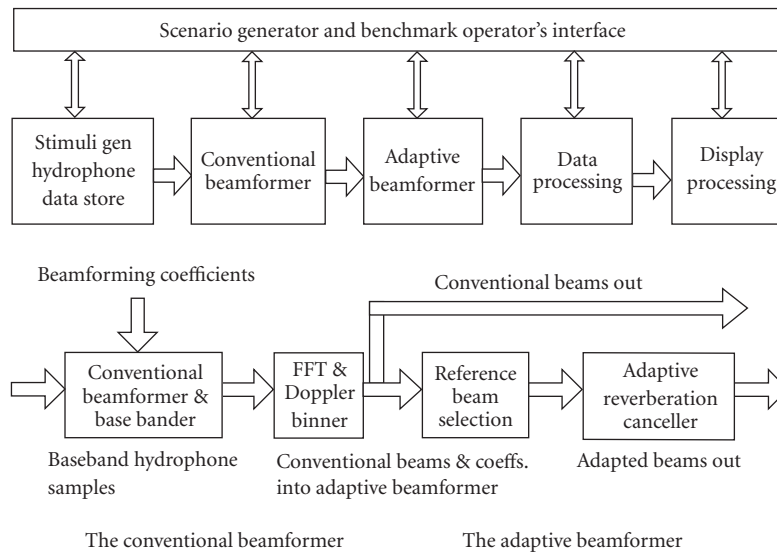
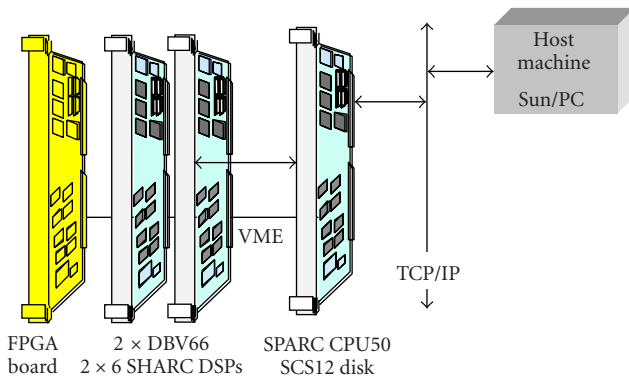FIGURE 7: Sonar functional processing chains.

FIGURE 8: Sonar hardware configuration.

(iii) conventional beamforming—comprises components that are to be used within the adaptive beamforming functionality. The current function box has three outputs:
  (1) beam samples (for each beam),
  (2) steering vectors (for each hydrophone/beam),
  (3) weighting vectors (for each hydrophone/beam).
  The last two output parameters are not essentially part of a conventional beamformer, but are formed, as they are required, parameters for the adaptive part of the benchmark functionality;

(iv) reference beam selection—algorithms to select the beams whose directions align with the direction of arrival of the reverberant returns being considered;

(v) adaptive reverberation cancelling—adaptive algorithms that use the reference beams to sense the reverberation at the Doppler bin in question, and cancel it from all other beams.

For the benchmark, the conventional beamformer, being fully understood and previously developed many times, could be implemented in its entirety within the GEDAE environment using the "embedded functions" available within the tool at both the host and optimised target level. The adaptive beamformer however required significant algorithm investigation and the development of specialist sonar library components, ten in total, to satisfy the needs of the specification. These components were all generated in high level C code and were not optimised for target execution.

Following the sonar benchmarking activity, an FPGA board was also connected to the VME to demonstrate heterogeneous implementation capability. A design flow linking the GEDAE and Handel-C tools was demonstrated, as a technology demonstrator only, and no benchmark metrics were collected for this activity.

3. THE METRICS

The objective of the benchmarks was to measure the performance of the ESPADON RP signal processing design environment with respect to the project goals: reduction in the product lifecycle cost and lifecycle time, and improvement of the product quality.

The measurements were through small design exercises, developing the benchmark applications described earlier, in order to capture basic metrics regarding the process and the product. The fundamental performance process metrics include design cycle time, nonrecurring cost, ease of use, and supportability. Performance product includes cost to manufacture, cost to maintain conformance to requirements, and dependability.

For each benchmark cycle, the benchmarks were structured to do the following.

(i) Evaluate the RP process:
    (1) comparison with current practice: develop current practice base-line costs and schedule;
    (2) performance metrics: employ metrics to measure and substantiate improvements;
    (3) improvement: identify weaknesses in the design process.

(ii) Evaluate the tool integration:
    (1) integration: verify the degree of tool integration, including libraries and databases;
    (2) ease of use: provide qualitative ease of use evaluation for the tools and processes;
    (3) improvement: identify weaknesses or missing elements in the tool set.

(iii) Evaluate the signal processing system (products):
    (1) architecture: assess the suitability and scalability of the HW and SW architectures;
    (2) compliance: measure the compliance of hardware and software to supplied requirements;
    (3) cost: provide current practice cost comparison.

Hence, the metrics that are needed to be measured are of different types. They can be summarised as belonging to one of the metric sets below.

Principal and supporting metrics

Considered to be the most important metrics. They must provide us with hard numbers regarding the improvement obtained by using the ESPADON methodology and directed toward specific issues of performance of both the ESPADON process and products, that is, reduced design cycle time, reduced cost, and improved quality.

Tool-oriented process metrics

Indicate more about the support which is given by the EDE and its constituent tools. While the principal and supporting metrics are important to the success of the ESPADON approach, the user's perception of ESPADON will be strongly influenced by the ease of use and uniformity of the EDE. Developing quantitative tool metrics to directly measure subjective attributes is a difficult if not an impossible task. However, certain attributes of the EDE can be measured, such as the consistency of the user interface, tool integration facilities, and so forth, bear some correlation with qualitative attributes such as ease of use.

Application complexity metrics

These primarily focus on the elements, products, or applications to be developed. The objective is to capture the inherent complexity of a given benchmark application, independent of the particular hardware and software implementation. The metrics will also serve as a reference for determining efficiency of the hardware and software realisations of a benchmark produced by the developer. In addition to complexity measures of the functions themselves such things as requirements of external interfaces, requirements for testability, reliability, and maintainability, and requirements constraints are also considered.

Product complexity metrics

Various types of products will be produced during the ES-PADON design process. These include software, hardware, and documentation. Even within a category, a variety of types of products may be produced. For example, in the software category products may include real-time application code, test code, simulation code, and so forth. For each significant product developed in the course of a benchmark, complexity metrics are required to characterise the efficiency of the product and the difficulty of implementation.

Product performance metrics

Measuring the performance of products produced using ES-PADON is different from measuring the performance of the ESPADON process itself. Misapplying a good process may produce poor products. These metrics aim to characterise the resulting performance of the individual products produced, for example, for a software product such things as computational efficiency, postrelease defect density, portability, and adherence to standards and testability are considered.

The measured metrics and summary of the important benchmark results are presented in Section 4.

4. THE MEASUREMENTS

There are distinct differences in the benchmark measurements of the sonar and radar applications. These are attributable to the difference of approaches of the two benchmarks. The sonar benchmark relied on GEDAE, a mature COTS tool for RP prototyping whereas the radar benchmark upgraded the open source Ptolemy Classic tool. In each case, the reference to conventional developments and processes refers to the existing methods and processes being used within the company performing the benchmarking activity. In general, these consisted of disparate groups of engineers performing a particular function within a typical V or waterfall development lifecycle with communication of requirements/specifications via paper documents. The baseline estimates refer to the time taken to perform the developments using these conventional approaches. They have been obtained using metrics available from previous developments of similar products and estimates produced from experienced engineers within the disparate groups mentioned above. In both cases, a final implementation using these conventional methods was not available to the developers and so a detailed comparison of performance between the baseline and the new developments was not possible. However, in both cases a level of performance was specified for the final implementation, and for the benchmarks to be successful, these had to be met.

4.1. Sonar

Table 1 shows the measurements from the first benchmark which is for the development of a conventional beamformer implemented on the initial target platform. Two overall design iterations were required before arriving at a compliant solution. That is the solution achieved real-time

TABLE 1

Activity (% of total)	Effort
Functional design: specification (14%), design (28%), implementation (42%), verification (13%), review (3%)	93 hours
Functional design 2: specification (14%), design (24%), implementation (24%), verification (24%), review (14%)	21 hours
Architecture design/implementation—5 iterations specification (11%), design (33%), implementation (11%), verification (11%), review (33%)	45 hours
(A) Total development time	159 hours
(B) Baseline estimate—including error correction	2537 hours
Performance improvement factor A/B	16
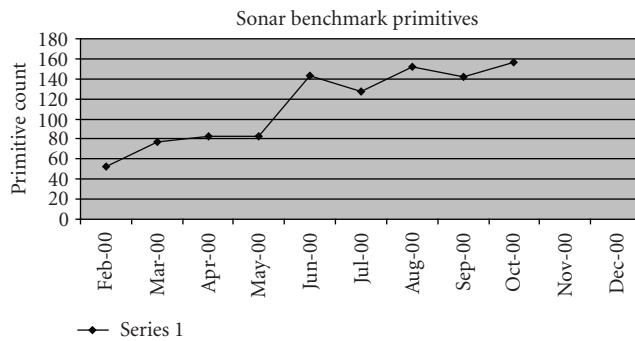
FIGURE 9: Sonar benchmark primitive count.

FIGURE 10: Sonar benchmark fault count.

performance, latency, and throughput, on the hardware architecture provided for the benchmark. The baseline estimate is from conventional developments for the same functions by Thales Underwater Systems which undertook the sonar benchmark activity. We attribute the significant productivity improvement to two factors. One is that the conventional beamformer is a fully understood and specified capability implemented many times within the company on a variety of different target architectures. The second is that the RP process enabled the application to be implemented in its entirety within the GEDAE environment using the "embedded functions" available within the tool at both the host and optimised target level. No specific sonar library components had to be generated for this application resulting in a high level of reuse and productivity.

In the case of the adaptive beamformer, the improvement factor is less because a number of new embedded functions had to be developed, as shown by Figure 9. Ten library components were developed, with approximately 1000 lines of code in total, across all components. These components were all generated in high-level C code and were not optimised for target execution. As can be seen in the middle of the graph it was sometimes necessary to remove boxes during redesign. This redesign usually required the development of new user-generated primitives to provide the required capability.

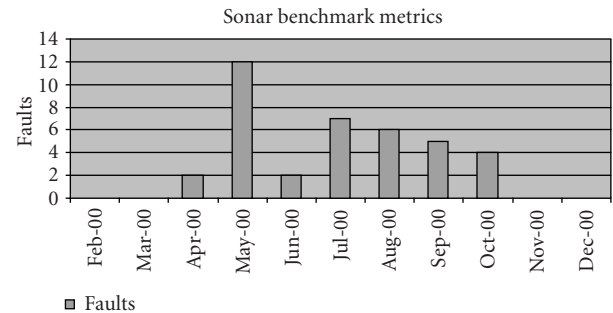A significant reduction in faults was identified, including reduced time to locate and remedy such faults, whilst implementing the adaptive beamformer once a level of understanding of the tool had been achieved by the developers (Figure 10). This meant that execution of the capability on the target machine resulted in no functional errors and allowed the developer the time to concentrate on the efficient partitioning of the capability rather than its algorithmic performance.

Another important metric measurement was the number of iterations of the design as shown in Table 2. Though more than planned, the rapidity with which designs could be iterated within the RP process enabled high productivity factors to be maintained. In fact, there were many more very short iterations that were not reported simply because they were too short but nevertheless important in converging to an acceptable solution. In the table below, an iteration is defined as being one pass through the process for the production of a product, for example, an algorithm, with particular functionality. Hence, additional iterations are required where the output of the particular process has not converged to an acceptable solution and has to be further refined during further iterations. So in the case of the time varying gain product, five iterations of the process were required, which covered initial development through improved functionality to optimised performance, before an acceptable solution that met its requirements was reached.

The actual measurements of the adaptive beamformer benchmarks showed that the total time for development was 1113 hours compared to 2737 hours using conventional

Table 2

Activity	Phase	Iteration
Planning	Planning	1
Specification	Specification	1
Data generator	Functional design	1
Conventional beamformer	Functional design	1
Adaptive reverb cancellor	Functional design	15
Time varying gain	Functional design	5
Reference beam selection	Functional design	7
Control interface	Functional design	1
Conventional beamformer	Architecture design	2
Total iterations		34

Table 3

Bare beamformer	Design 1	Design 2	Design 3	Design 4
Nofchannel	8	8	8	8
Nofsweep	17	17	17	17
Nofproc	4 (+2)	5 (+2)	4 + 4	8
Input data	DMA	DMA	Pre-load	Pre-load
Output data	(DMA)	(DMA)	(DMA)	(Pb MCS)
Corner-turn	4− > 4	NO	4− > 4	8− > 8
RACE++ peak load	—	—	—	53%
Latency	1 burst	1 burst	2 burst	1 burst
Performance	25 ms	21 ms	9.5 ms	9 ms
Support var. burst L.	Yes	Yes	Yes	Yes
Design time	72 H	16 H	12 H	16 H

methods. This gives a productivity improvement of factor of 2.4. In terms of the reduction of coding errors, the comparison for the adaptive beamformer shows a factor of eight (conservative) reduction compared to conventional methods. This is attributable to the RP process providing algorithm developers/system analysts and software developers with the benefit of the use of a common development environment.

4.2. Radar [15]

In this case, a significant part of the benchmark work was tightly coupled to enhancing the Ptolemy Classic support environment for the target heterogeneous multiprocessor architecture and its integration within the overall EDE. Hence, the metrics covered more aspects of the RP process than the sonar benchmark. In particular

(i) more metrics were measured related to the application complexity, to the methodology and tool support, to the performance of the application and the libraries, to the validation process, and to the overall RP process;

(ii) metrics were measured for a heterogeneous RP architecture mixing FPGA and power PC.

For the first implementations on the Mercury platform, the benchmark application required in total eight design itera-

tions to complete before converging to a real-time performance compliant solution, that is, meeting the overall latency figure required for the processing. Results for four of the design iterations are shown in Table 3. For these cases, the improvements of the real-time performance (from 25 to 9 millisecond) between the iterations resulted from an iterative process based on

(i) an analysis of the real-time performance of the previous iteration implementation, in order to identify the time-consuming components and the communication bottlenecks;

(ii) an optimisation of the mapping of the radar application into the Mercury platform PPCs;

(iii) an upgrade of the code generators and the library components developed in Ptolemy Classic for the benchmark;

(iv) the execution of the new application implementation with the upgraded mapping and generated code.

During the radar benchmark iterations, different communication protocols and memory allocation mechanisms were experimented. Unexpected weakness of the COTS platform and VSIP library were identified and bypassed. These changes were mainly limited to Ptolemy code generators.

The following Figures 11, 12, and 13 summarise the measurements for the overall design of the first RP implementation, the test and validation and the application complexity, respectively. The right vertical axis presents cumulative totals represented by the dots. The figures present the results for a nonadaptive beamformer (bare BEFO) and for the full adaptive beamformer (full BEFO)

The overall time measured to implement (coding) and test/integrate the application was 354.5 hours. Thales Naval Nederland which undertook the radar benchmark have calculated a baseline figure of 481 hours for a conventional process. Hence, this gives an improvement factor of 1.4. The lower figure is attributable to the many more enhancements that had to be carried out with Ptolemy Classic thereby reducing the level of (re)-use of existing functions for the benchmark. However, future developments of similar applications will not have to incur all of these library component developments and hence this improvement factor will increase. In Figure 13, we finally present the complexity of the major iterations in terms of generated lines of code (LOC) as well as number of functions used in the designs.

The final benchmark implementation was based on a Mercury and FPGA heterogeneous platform. This Benchmark was not a real-time implementation since the communication between the Mercury boards and the FPGA board employed a low-bandwidth VME bus instead of the high-speed RACE++ interconnect between the Mercury boards.

The beamformer application after simulation (SDF/BDF) was mapped on three different configurations: 1 PPC/1 FPGA, 4 PPCs/1 FPGA, and 4 PPCs/4 FPGAs. The Handel-C code was generated for these three different implementations. Only the two first solutions where finally synthesised and tested into the FPGA.

Performance results of the FPGA implementation were for the 1 PPC 1 FPGA configuration:

  (i) 67% of SLICES (63536 gates, 3481 flip-flops) used,
  (ii) maximum frequency = 21.8 MHz,

and for the 4 PPC 1 FPGA configuration:

  (i) 99% of SLICES (97270 gates, 5052 flip-flops) used,
  (ii) maximum frequency = 17.2 MHz.

Due to the I/O limitation of the FPGA board, the performance of the beamforming FFT was tested as being a factor of three times slower on the FPGA than on the PPC (approx. 1.1 microsecond per FFT8). Although real-time performances were not demonstrated, the benchmark enabled the measurement of the design complexity and design time duration and the estimation of the development time speed up improvements.

In Figure 14, the complexity of the designs is summarised in terms of generated LOC as well as number of functions used in the designs.

Although the development for supporting the heterogeneous architecture was at first highlighted as a high-risk element, it actually took just over three engineer months. This included the development of the new Handel-C domain in
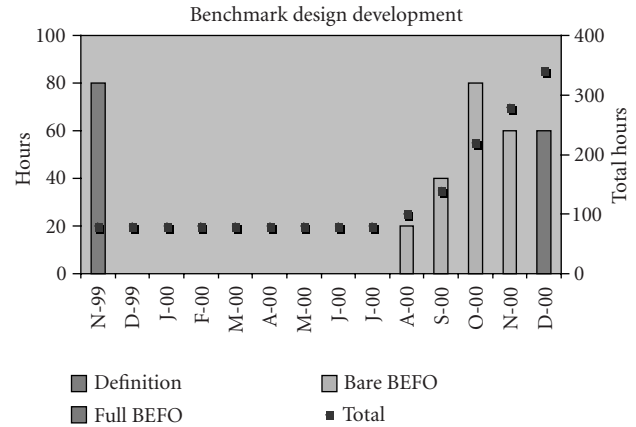

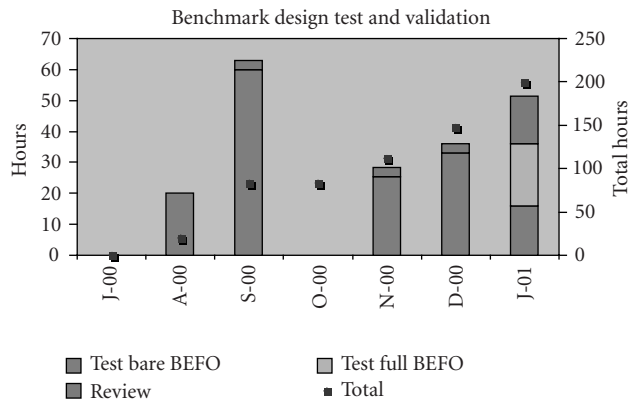FIGURE 11: Radar design and development duration.

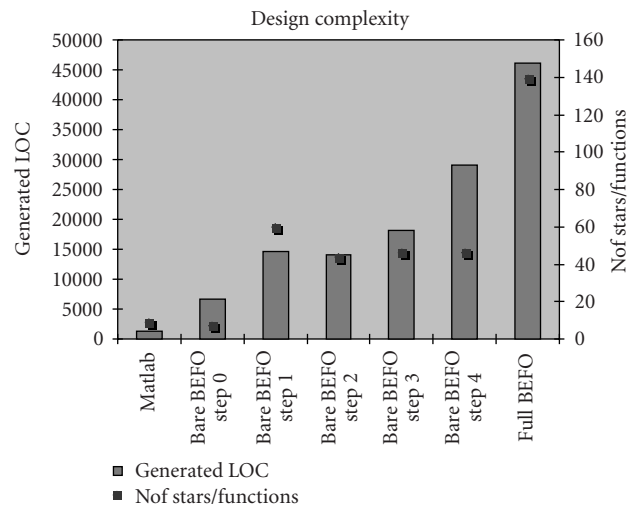FIGURE 12: Radar test and validation duration.

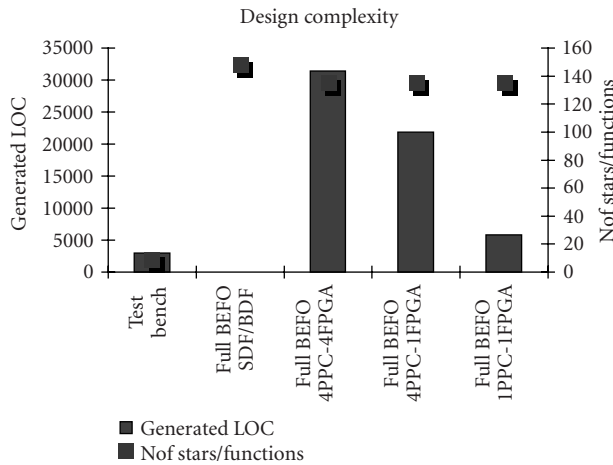FIGURE 13: Design complexity of radar benchmark implemented on a Mercury platform.

FIGURE 14: Design complexity of the RADAR benchmark implemented on the heterogeneous platform.

Ptolemy Classic, a new target supporting the mix of Mercury compute nodes and FPGAs, the needed Handel-C library developments, and HW/SW integration activities.

The heterogeneous platform benchmark was more limited than the mercury-based benchmark nevertheless a significant improvement of the development time by using an RP environment can be estimated. In the conventional development methodology, the design of an FPGA with this kind of complexity would take approximately 500 hours, including documentation. If we only focus on the development process, this reduces to approximately 300 hours. Compared to the development time using the EDE (40 hours), this means an improvement of 7.5 was achieved.

5. CONCLUSIONS

An adaptive beamformer application for a radar and sonar was successfully designed, implemented, tested, and benchmarked using the EDE and the ESPADON RP process. An improvement factor of 1.4 and 2.4 in productivity was demonstrated for the radar and sonar beamformer application, respectively. Hence, we can be confident that a halving of embedded signal processing system development lifecycle can be achieved using the RP methodology and support environment. A much higher factor of 16 was achieved with a conventional sonar beamformer. This implies that significantly higher factors are possible through the use of a common RP process and environment and the development of application domain libraries to maximise future (re)-use of signal processing functions.

An important finding and factor towards these productivity gains is that the functional, architectural and implementation design are done simultaneously instead of sequentially simply because it is so easy and fast to do these with the RP environment. This demands for a different process that allows for rapid and higher frequency types of iterations. Also many feedback loops are performed towards the

redesign of the environment. This allows for very rapid iterations to arrive at the correct design and reduce error propagation. The latter also benefited from a common RP environment and the exchange of information as graphical designs that can be quickly integrated into the overall functional design.

The approach has been found to be scalable to larger designs than the benchmark applications discussed in this paper, although this is to a certain extent a function of the tool rather than the rapid prototyping methodology. Indeed, both the tools discussed have a mean of automatically scaling elements of the design under parameter control. This is particularly useful in such sensor-based systems where a trade-off analysis with respect to say the number of beams and performance can be conducted without modifying the overall structure of the model.

Although the work has focused on rapid prototyping onto COTS processor-based components, the ESPADON methodology is also applicable to the development of algorithms for proprietary hardware platforms such as system-on-chip (SOC). This would also involve the use of techniques such as virtual prototyping, that is, the development of a model of the system to execute on a virtual model of the final hardware. These techniques and their supporting tools were found to be far less mature than those available for rapid prototyping. However, rapid prototyping is an important step towards defining the data required for virtual prototyping techniques in order to produce a sufficiently accurate performance model of the algorithms and to scope the proprietary developments.

Signal processing functions are only one part of an embedded sensor system. Further work needs to be done to extend the RP process to include other functions such as system control, front-end interfacing and processing, back-end data processing and the HCI for commensurate productivity improvements at the system level. These topics require further research and development.

NOTE

ACKNOWLEDGMENTS

presented at the internal BAE SYSTEMS Signal and Data Processing Conference, 5–7 March, 2002, Dunchurch Park Conference Centre, UK, Conference Proceedings pages 1–21.

REFERENCES

[1] ESPADON Programme Deliverable, "ESPADON: Final Report," Tech. Rep. Dex 45990, Thales Airborne Systems, Elancourt, France, September 2001, Project EUCLID-CEPA 2-EUROFINDER contract no. 97 34 391 00 470 75 65.

[2] J. Pridmore, G. Buchanan, G. Caracciolo, and J. Wedgwood, "Model-year architectures for rapid prototyping," *The Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 15, no. 1-2, pp. 83–96, 1997, Special Issue on Rapid Prototyping of Application Specific Signal Processors (RASSP).

[3] J. P. Calvez, *Embedded Real-Time Systems. A Specification and Design Methodology*, John Wiley, Chichester, Sussex, UK, 1993.

[4] B. W. Boehm, "A spiral model of software development and enhancement," *IEEE Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[5] Ptolemy, Department of Electrical Engineering and Computer Sciences of the University of California, Berkeley, Calif, USA, http://ptolemy.eecs.berkeley.edu/.

[6] Handel-C (DK1) Celoxica, Abingdon, Oxfordshire, UK, http://www.celoxica.com/.

[7] M. Fleury, R. P. Self, and A. C. Downton, "Hardware compilation for software engineers: an ATM example," *IEE Proceedings Software*, vol. 148, no. 1, pp. 31–42, 2001.

[8] M. Vasilko, L. Machácek, M. Matej, P. Stepien, and S. Holloway, "A rapid prototyping methodology and platform for seamless communication systems," in *12th International Workshop on Rapid System Prototyping (RSP '01)*, Monterey, Calif, USA, 2001.

[9] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous dataflow programs for digital signal processing," *IEEE Trans. on Computers*, vol. 12, no. 8, pp. 971–989, 1987.

[10] J. T. Buck, "Scheduling dynamic dataflow graphs with bounded memory using the token flow model," Tech. Rep. UCB/ERL 93/69, Ph.D. Dissertation, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Calif, USA, September 1993.

[11] S. S. Bhattacharya, P. K. Murthy, and E. A. Lee, *Software Synthesis from Dataflow Graphs*, Kluwer Academic Publishers, Norwell, Mass, USA, 1996.

[12] D. Aulagnier, P. Meyer, H. Schurer, and X. Warzee, "Rapid prototyping of RADAR signal processing systems using Ptolemy Classic," in *Proc. Ptolemy Miniconference*, University of California, Berkeley, Calif, USA, March 2001.

[13] J. L. Pino, M. C. Williamson, and E. A. Lee, "Interface synthesis in heterogeneous system level DSP design tools," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 1268–1271, Atlanta, Ga, USA, May 1996.

[14] EUROPRO Consortium, "EUROPRO Final Report," Tech. Rep. Project P21040-HPCN/EUROPRO, V1.1, Thomson Marconi Sonar, Sophia Antipolis, France, May 1999.

[15] H. Schurer and J. J. Hunink, "Rapid prototyping of radar signal processing algorithms," in *IEEE Workshop on Processing, Integrated Systems and Circuits (ProRISC '99)*, Mierlo, The Netherlands, November 1999.

Bob K. Madahar graduated with special honours in physics in 1976, followed by a Doctorate in space physics from the University of Sheffield before being appointed for a three-year postdoctoral research fellowship at Lancaster University in the Environmental Sciences Department. He joined industry in 1982, as a Scientist in a leading Scientific and Computer Consultancy Company in Cambridge before moving in 1991 to the Long Range Research Laboratory, Machine Vision Group, of the GEC-Marconi Hirst Research Centre. He has rapidly progressed from Team Leader, Group Leader to his current role of Chief Scientist of the Systems Department of the company, now BAE SYSTEMS Advanced Technology Centre. Dr. Madahar has worked over the last two decades for industry and the government in many areas concerned with real-time processing of signals and high-performance computing. These areas have included active control of sound and vibration, fluid dynamics, oceanography, data fusion, parallel processing, and computer security, particularly trusted networks. For the last ten years, he has been working on high performance distributed and embedded signal processing systems, particularly the systems and architectures, design methods, and tools and algorithms to reduce system through life costs.

Ian D. Alston graduated with a degree in mathematics from the University of Manchester in 1981. In 1984, he joined the Communications and Computing Laboratory at the Marconi Research Centre where he worked on several communications system design projects with particular emphasis on digital signal processing and protocols. Following the merger of British Aerospace and Marconi Electronic Systems, he is currently the Team Leader of the Future Architectures and Prototyping Technologies team within the Systems Department of the BAE SYSTEMS Advanced Technology Centre (Great Baddow). Most recently, he has been involved in the use of rapid prototyping techniques applied to embedded system design and was BAE SYSTEMS Project Manager of the Environment for Signal Processing Application Development and Prototyping (ESPADON) European Collaborative project. This work is now being extended to fully heterogeneous embedded architectures. He is currently the Project Manager and Technical Lead on the UK MoD CRP project on "Core-based DSP for FPGA," reference CRP/01/S&E/SPI/IPI. This is part of the BEACON programme and involves Qinetiq, Malvern, and BAE SYSTEMS ATC as the industrial partner.

Denis Aulagnier received an Engineer degree from l'École Nationale Supérieure des Arts et Métiers (1985) and an M.S. in electrical engineering from North Carolina State University (1988). He worked at Thales Airborne Systems since 1988 where he was in charge or participated in the development of multiprocessor computers for radar digital signal processing. Since 1996, he participated also in different research studies on design environment for signal processing and was the ESPADON Programme Technical Manager. He is a member of the IEEE Computer Society.

Hans Schurer was born in Nijeberkoop, The Netherlands, in 1967. In 1990, he received the B.S. degree in electrical engineering from the Leeuwarden Polytechnic Institute, The Netherlands. In 1993, he received his M.S. degree (cum laude) in electrical engineering from the University of Twente, Enschede. In November 1997, he obtained his Ph.D. from this same university. The subject concerned linearisation of electroacoustic transducers using nonlinear digital signal processing. During this research, a prototype of a digital loud speaker system with nonlinear distortion compensation was designed and built for Philips Research Labs, Eindhoven. At the moment, he is employed at Thales Naval Nederland B.V. (formerly Hollandse Signaalapparaten B.V.) in Hengelo, as a Digital Signal Processing Technical Authority. His main research interest is in antenna-based signal processing techniques for radar systems.

Mark Thomas is a Senior Systems/Software Design Consultant with over 18 years experience of the UK sonar industry. In depth knowledge of sonar processing with extensive experience in the design, development, integration, testing, and acceptance of large, and small, real-time sonar applications on programmes such as 2046, 2050, and 2076. He is contributing to current TUS open systems studies. Consultant on the ESPADON (Environment for Signal Processing Application Development and Rapid Prototyping) programme, a trination European EUCLID/EUROFINDER defence programme that successfully completed in September 2001, and researched rapid (DSP-based) and virtual (FPGA) programming.

Brigitte Saget is presently with MBDA-France, Airborne Electronics Engineering Directorate. She has 25 years of experience in research and development of embedded electronics and software for aerospace and defence applications. Following her graduation from Ecole Supérieure d'Electricité in 1978, she joined MATRA Defense which became Matra BAe Dynamics and recently MBDA. From 1978 to 1984, she was involved in software developments for Matra civil activities (transportation & tele-surveillance applications). Then, she joined the Defence sector where she was in charge of software development teams in the field of automatic test equipment for missiles from 1984 to 1989. Since 1989, she has been working in the field of airborne avionics & missile equipment. She is now in charge of research, methods & CAD tools in support of the electronic & software design teams as well as coordinating external R&D contracts. During the last 6 years, she has been involved in EU or EUCLID/EUROFINDER projects among them the RTP 2.29 ESPADON, a programme to define and validate a new development methodology and environment for rapid prototyping of signal processing applications which was successfully completed in September 2001.