A Modular Pipeline Software Architecture for Radar Signal Processing

Krushi Deep M, S L Kumar, Kiran Manju D,
Traana Technologies Pvt Ltd, HBR layout, Bangalore 560 043, Karnataka, India

Suchith Rajagopal, Vaidya Dhavalkumar Bipinchandra
Electronics and Radar Development Establishment, DRDO, C.V.Raman Nagar, Bangalore-560093. Karnataka, India

krushi@traana.in

Abstract

This architecture is built around the concept of creating a processing pipeline, with each elemental signal processing algorithm being realized as an independent filter, and, the entire signal processing requirements for a mode being realized by stringing together the required filters as a pipeline. The architecture is scalable in terms of different processing pipelines being able to be created in different processing cores, thus, allowing the radar system designer to add or subtract processor cores based on the processing needs. Bringing in multiple processor cores brings in the additional challenges as the radar signal processing does switch between ranges based processing to pulse based processing. The other dimension of scalability that this architecture provides is the ability to add new signal processing steps or modes in form of new filters or pipeline.

Keywords— Modular, Pipeline Software, Radar Singal Processing, Multicore, Memory Manager, Error Handling, Architecture

I. Introduction

With the availability of ever-increasing processing bandwidths offered by the present day multi-core processors, Radar Signal Processing is being increasingly moved to the software realm from it being predominantly being implemented in hardware. The modern day airborne radars bring with them the added complexities of having to handle multiple modes of operations, and these modes of operations being rapidly switched.

This paper introduces a Modular Pipeline Architecture for implementing the airborne radar signal processing algorithms in software. This architecture introduces modular signal processing elements in the from of an independent filter, these elements are stitched together to from a required signal processing pipeline. Each filter has an active as well as a passive part. While the active part runs on an independent thread realising the actual processing algorithm, the passive part comes into play during the pipeline setup and destroy phases.

The setting up of a pipeline, and its subsequent tear down, is controlled by a System Manager, which also handles configuration management aspects. A Memory Manager module helps in managing the limited available memory across the different filters, and, is the main vehicle through which the data flows down the pipeline, as it is getting processed.

This architecture has been realized on Multi-core PowerPC processors on a Mercury HCD6220 COTS board and has been demonstrated to meet the hard real time constraints as well as the system memory constraints.

Subsequent sections of this paper explain the software architecture overview and its build blocks.

II. Software Architecture Overview

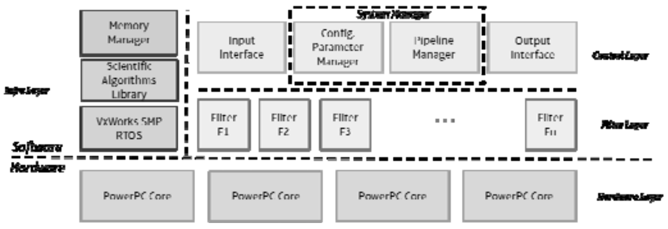


Figure 1 Modular Pipeline Software Architecture.

The hardware layer represents the underlying PowerPC cores available for signal processing on the Mercury HCD6220 COTS board.

The software itself is built on four different layers. The infra layer houses the essential RTOS VxWorks, the computational algorithms powered by Scientific Algorithms Library (SAL) developed by Mercury Inc and the memory manager catering for all the memory requirements.

The Filter Layer consists of all the algorithms required for signal processing in different modes of operation in the form of filters and they are used to form the processing pipeline as required.

The control layer is in charge of providing input and output interfaces along with the system manager. The pipeline manager and the configuration parameter manager together form the system manager to support and coordinate the mode specific activities like pipeline formation and then trigger the data processing.

III. Filter Architecture

Filters are the fundamental and independent blocks of the algorithm in this architecture. Though the functionalities of filters vary widely, but they all provide uniform & standard interfaces to the control layer above. This enables to describe all filters through a set of standard interface methods. These methods, called filter template methods, are implemented by each of the filters without violating the interface specifications.

For realizing this goal the following design is visualized. Each filter will be consisting of "Passive" and "Active" components. While "Passive" part is a set of static routines which are called and executed in the context of "caller", "Active" part has its own thread of control task for execution. All the method of invocations are dispatched as messages. It can be seen from the figure below that the data processing continues while commands are queued up in Message-Q by the passive component of the filter.

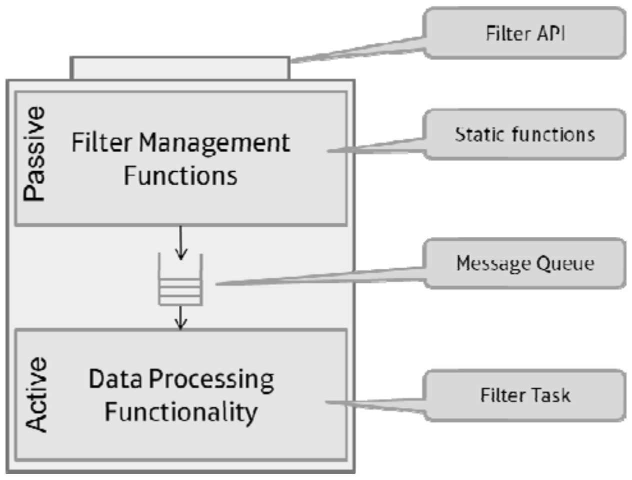


Figure 2 Filter structure

A filter, is a generic component, encapsulates a specific algorithm/functionality. Each filter can be instantiated to a max allowed number in a pipeline. Thus, if a "filter F" participates three times in a mode processing pipeline, there will be three instances of filter F. Apart from sharing the implementation other resources like Task/Thread and Memory Segments are claimed and released by the active instance and hence shared between instances.
.

IV. Pipeline manager

The major design pattern of SP filter layer is the pipes and filters pattern. A pipeline consists of a chain of processing filters or elements, arranged according to the data processing requirement of operating mode.

The pipe connects one filter to the next, sending output messages from one filter to the next. Thus the control or data can flow from one filter to the next through the pipe established during the connection process. These filters in a pipeline can be rearranged or recombined to achieve specific functional requirement of mode, all without having to change the filters themselves.

For a pipeline, filter connection sequence is captured through text files rather than embedding in code. This gives the flexibility of modifying Pipeline without re-doing the code. Thus this text file contains the list of filters involved in specific mode of operation

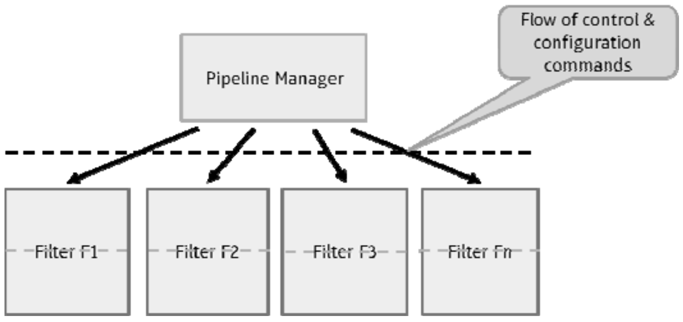


Figure 3 Pipeline data flow

The Pipeline Manager, using the above text file, establishes the connection between each of filters involved in specific mode of operation.

V. System Manager (SysMgr)

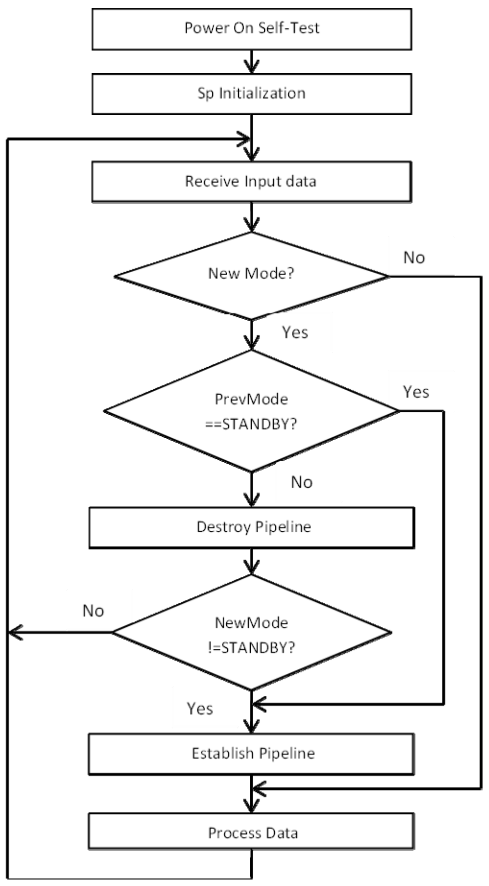


Figure 4 System Manager Operation sequence

System Manger(SysMgr) is the core component of the Signal Processing (SP) system. Figure 5 represents the normal operating sequence of the System Manager. The System Manager performs following functionalities: On power-on,the Power-on Self-Test (POST) is done during which pre-defined parameters of the system are checked. After successful POST, it continues with SP initialization sequence. SysMgr initializes the MemMgr and all other relevant modules in the system.

After this stage, system is ready to handle the valid data via input interface.

VI. MEMORY MANAGER

Memory Manager Component is responsible for controlling and handling the system memory requirements of the filters in the Signal Processing modes. It creates and maintains a pool of multi-sized memory segments. These memory segments are dynamically requested, used & released by the filters.

The sizes & number of different memory segments are configurable. Therefore it is scalable. The Memory Manager mainly emphasizes on reusability of the memory segments

The Memory Manager provides various APIs through which the memory segments can be dynamically requested or released. A filter which requires a new memory segment would request the Memory Manager for the memory segment of specific size by calling the API, AllocateSegment(). The Memory Manager on receiving the request provides suitable sized segment available from the pool.

The Memory manager keeps track of the number of filters which are currently using the memory segment. It also keeps track of memory segments under use and free to be allocated. In case, the subsequent filter wants to use the same memory segment, then it has to call the API, HoldSegment(). When the segment is no longer required, the filter can release it. This is accomplished by calling ReleaseSegment(). Only when all the filters which had held the memory segment release it, then it will be available for allocation on filter's request.

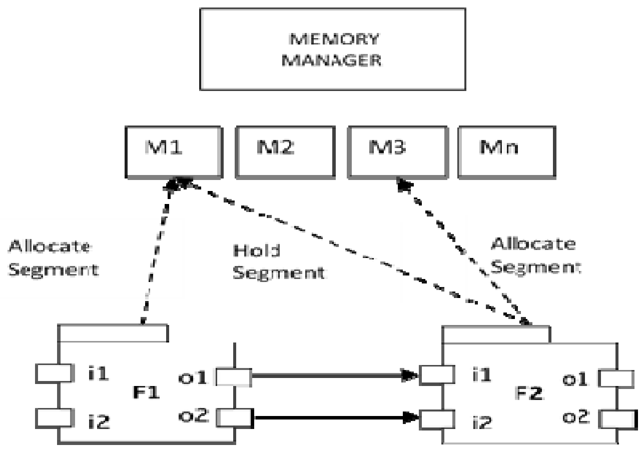


Figure 5 Memory Manager

Thus system memory is effectively used in the design. During Memory Manager termination, these memory segments are freed.

Bringing in multiple processor cores brings in the additional challenges as the radar signal processing does switch between range based processing to pulse based processing. The concept of 'scatter-gather' of data is built into the architecture, by which a single burst data is 'scattered' to the various available cores for processing, 'gathered' and assembled back before it is sliced in the other dimension and scattered back to the different cores for subsequent processing.

VII. CONFIGURATION PARAMETER MANAGEMENT

The data received from the Data Acquisition System will also contain Control information along with the target returns. This Control information is parsed by the System Manager and stored in a data structure. This structure contains various radar parameters which are used for Data Processing by filters in the pipeline.

The filters query the Configuration structure for the required parameter which is necessary for data processing. The received Control Word information may change every burst (Bunch of transmitted pulses grouped together) and is updated by the Configuration Management. After each burst, the filters in the pipeline make use of updated Configuration structure.

The Configuration Management System provides scalability as any number of new Radar parameters can be added with least amount of software design change.

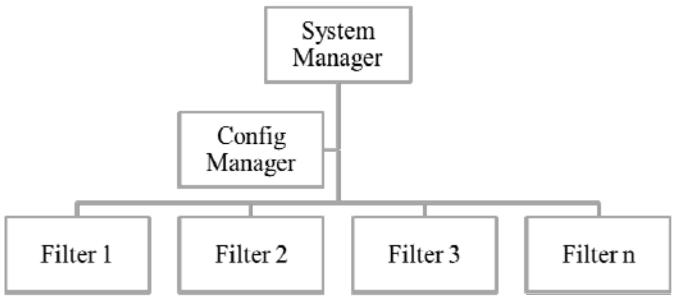


Figure 6 Configuration manager

VIII.CONCLUSION

The complex problems in Radar Signal Processing such as processing large amount of data, handling multiple modes of operation and switching rapidly between these modes are addressed by adopting the modular Pipeline Software Architecture. The signal processing steps/units called filters provide features such as scalability and reusability.

It was observed that the data was processed with less than 50% of the total available time.

ACKNOWLEDGEMENT

The authors would like to thank The Director LRDE and Radar V, Divisional Officer for all the encouragement provided for this implementation and study. The authors wish to acknowledge the support of Scientists and Technicians of LRDE, Bangalore, associated with the design and development of Active Electronically Scanned Array Radar.

REFERENCES

[1] Software Architecture Perspectives on an Emerging Discipline by Mary Shaw, David Garlan, Eastern Economy Edition, Prentice-Hall India

[2] An Embedded Software Premier, David E. Simon, LPE, Addison-WESLEY

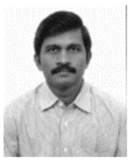
[3] "VxWorks Application programmer's guide "6.7, Wind River.

[4] "Scientific Algorithm Library (SAL): Reference Guide" TC-SAL-RM-750, 2006/08, Mercury Computer Systems, Inc.

[5] "Introduction to Radar Systems", Third edition, Skolnik, TATA McGraw Hill.

BIO DATA OF AUTHORS

Krushi Deep M received the M.S. degree in Embedded Systems and Design from Manipal University in 2012 and the B.E degree in Telecommunication from the VTU Belgaum in 2009. He joined Traana Technologies Private limited, Bangalore in 2012, where he is currently Systems Head. His areas of interest are Signal Processing, Design and development of Algorithms



S. L. Kumar, was born in 1972 at Kadukkarai, Kanyakumari Dist. He received his B.E., degree in Electronics & Communication Engineering from Bharathidasan University, Tiruchirappalli in 1993. Having over 20 years of comprehensive experience in Embedded System software, Architecture definition & deployment and Product creation for Defence, Semiconductors, Consumer Electronics and Mobile Industries. His current areas of interests include emerging technologies around IoT.

Kiran Manju D received the M.S. degree in Embedded Systems and Design from Manipal University in 2013 and the B.E degree in Electronics and Communication from the VTU Belgaum in 2010. He joined Traana Technologies Private limited, Bangalore in 2013, where he is currently Member of Technical Staff, R&D. His areas of interest are Signal Processing and Systems Engineering.

Suchith Rajagopal obtained his B.E. degree in Electronics & Communication Engineering in 1995 from Calicut University and M.E. degree in Computer Science Engineering in 2010 from IIT Madras. His Current area of interests includes radar system engineering

Vaidya Dhavalkumar Bipinchandra was born in 1984 at Surat, Gujarat. He obtained Electronics Engineering degree from NIT,Surat in 2005.
He joined Electronics and Radar Development Establishment (LRDE), Bangalore in 2006 as a Scientist. His areas of interests are design and development of signal processor for airborne Radar.