5.11 MIGRATION OF WSR-88D SIGNAL PROCESSING FUNCTIONALITY TO OPEN SYSTEMS

Sebastian M. Torres[(1)] and Allen Zahrai[(2)]
(1) Cooperative Institute for Mesoscale Meteorological Studies, University of Oklahoma
(2) NOAA/ERL/National Severe Storms Laboratory
Norman, Oklahoma

1. INTRODUCTION

From its inception, the WSR-88D program has recognized the need for system enhancement and growth. Under the NEXRAD Product Improvement (NPI) project, the National Severe Storms Laboratory (NSSL) designed and demonstrated an open-system architecture for the Radar Data Acquisition (RDA) subsystem of the WSR-88D radar (Zahrai et al. 2002). One of the objectives of the first phase of the open RDA (ORDA) project consisted of replacement of the signal processing subsystem (SPS) and the RDA status and control computer (RDASC) with a modern flexible and scalable system. Legacy Hardwired Signal Processor (HSP) and Programmable Signal Processor (PSP) functionalities were successfully migrated to a multiprocessor environment suitable for this type of application. This paper presents a general description of this migration process.

2. SIGNAL PROCESSING REQUIREMENTS IN THE WSR-88D

The WSR-88D signal processing subsystem employs time series data from the receiver to generate radial estimates of base data (reflectivity, Doppler velocity, and spectrum width) for each radial bin in the radar volume of coverage. In addition, during antenna retrace, it performs automatic calibration routines that monitor the status of the RDA and ensure that hardware imperfections are measured and compensated. The majority of the arithmetic calculations in the RDA fall within the SPS domain. Radar signal processing algorithms operate on a large number of range bins, and a pipeline of different procedures is applied to each set of data depending on the acquisition mode and atmospheric conditions.

Legacy WSR-88D signal processing functionality was implemented in the HSP and PSP subsystems (Unisys Corp. 1989). The HSP receives raw data from the receiver, processes it, and outputs it to the PSP. Processing at the HSP consists of (Unisys Corp. 1992b, Unisys Corp. 1993):

- conversion of digital I and Q video signals from logarithmic to linear scaling,
- suppression of ground clutter, and
- rejection of near-channel interference.

Corresponding author address: Sebastian M. Torres, CIMMS/NSSL, 1313 Halley Circle, Norman, OK 73069; email: Sebastian.Torres@noaa.gov

As such, the HSP outputs a stream of “clean”, linear, digitized I and Q video echo samples for consecutive range cells along successive beam sweeps.

On the other hand, the PSP handles base data generation and the computation of calibration parameters during antenna retrace (Unisys Corp. 1991, Unisys Corp. 1992a). Operations in the PSP include:

- matched filtering for long pulse data,
- power and pulse-pair sum computation,
- strong point clutter censoring,
- base data computation, and
- velocity ambiguity resolution.

Inputs to the HSP are digitized I and Q video, digitized LOG video, automatic gain control (AGC) value, interference tag, and ground clutter filter maps. Triplets of I, Q, and AGC values arrive to the HSP every 1.67μs. Outputs from the PSP are compressed and scaled reflectivity (Z), velocity (V), and spectrum width (W) estimates for every range bin in the beam sweep. These estimates are obtained with a resolution of 1 km for Z and 250 m for V and W. Output of radial base data to the radar product generation (RPG) group is required at the end of the dwell time

$$T_d = N_p . T_s \,, \tag{1}$$

where N_p is the number of sweeps (pulses) in a radial (usually one degree in extent), and T_s is the pulse repetition time. Other PSP output data include miscellaneous test and calibration measurement data; however, these occur only between elevation cuts or during antenna retrace periods.

3. MIGRATION TO OPEN SYSTEMS

System demands for intensive computations and data throughput in a real-time environment coupled with expandability and maintainability requirements can be met only with a design that accommodates multiple processors. Consequently, the ORDA signal-processing subsystem should consist of an array of processing elements connected through their own high-speed interconnect (Zahrai et al. 1999).

For real-time performance of the SPS, the signal processing algorithms have to be implemented such that the total time to process any one radial is less than the corresponding dwell time (1). Another migration objective is to devise an implementation that is easy to understand, code, debug, and port; and to provide a program that makes the most efficient use of the unique computational resources of a particular hardware platform, thereby running as fast as possible (Ackenhusen 1999).

The simplest implementation of a process would be the one that accommodates the complete functionality within one processor. However, for complex processes with sizeable data volumes, latency and throughput requirements are usually difficult to meet in a single-processor environment. Decomposing a process into multiple threads is important in real-time applications since it allows these threads to execute simultaneously on a multiprocessor system.

Concurrency can be achieved by means of data partitioning, process partitioning, or a combination of both. Data partitioning involves process replication by creating clone processes, each with its own processing resources. With this partition, input data is equally distributed among clones, hence data throughput increases and latency decreases with the number of cloned processes. In cases where data partitioning is not enough to achieve acceptable levels of throughput and latency, a combination of data and process partitioning is usually required. Process partitioning entails dividing a process into simpler sub-processes to form a pipeline. Because sub-process threads can execute in parallel, the implementation exhibits additional throughput and latency decreases. Figure 1 shows the implementation of a process using a combination of data and process partitioning.

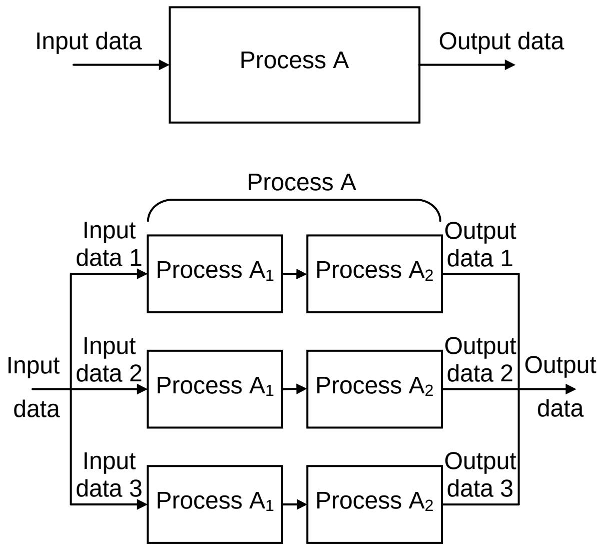

Figure 1. Implementation of a process using (top) single-processing environment and (bottom) a combination of data and process partitioning in a multiprocessor environment. Process A is partitioned into processes A_1 and A_2, input data is equally divided into 3 non-overlapping parts, and output data from each processing pipeline is recombined at the output end.

4. DESIGN DECISIONS

When trying to exploit concurrency in a multiprocessor environment, the designer needs to consider the issues of parallelism, clustering, and interprocess communications. Parallelism is inherent to sensor-based applications. In our case, each range radial goes through the same processing pipeline. Clustering is defined as the process of grouping partitioned tasks or sub-processes so that, in principle, each cluster can be implemented to execute in a single processor. Higher levels of partitioning will result in code that is more complex in terms of synchronization and control among the clusters. In addition, the overhead for data transfers between processors increases in direct relation to the number of clusters. Therefore, clustering must be implemented so that required latency and throughput levels are achieved with the minimum amount of partitioning.

For the WSR-88D signal processing subsystem the following functional tasks have been identified:

1. I and Q AGC correction and balance,
2. log channel linearization,
3. interference tag processing,
4. ground clutter filtering,
5. interference suppression,
6. matched filtering,
7. power sums accumulation,
8. pulse-pair sums accumulation,
9. strong point clutter censoring,
10. echo power computation,
11. Doppler velocity computation,
12. spectrum width computation,
13. reflectivity computation,
14. velocity unfolding,
15. data packing and formatting,
16. bias and noise power computation, and
17. fundamental Fourier coefficient computation.

The functions listed above can be classified as range bin operations, pulse operations, or radial operations. Functions in the first class are 1 through 6. Further, functions 1 through 4 operate on independent bins (regardless of their range location) while functions 5 and 6 operate on blocks of range bins. Functions in the second class are 7 through 13, 16, and 17. Functions 14 and 15 operate on radial base data, and therefore belong to the third class.

Assuming that a single-processor implementation of the complete functionality is not affordable, a logical clustering for the functions listed above can be done in terms of the previous classification. Although functions in the first class are the simplest, the volume of input and output data is the largest, posing a high demand on the corresponding processors. Functions in the second category are moderately complex, but although the input data volume is still large, the bulk of data is reduced by a factor of N_p after processing. Lastly, functions in the third category are highly complex, but both input and output data streams have been reduced by a factor of N_p.

Additional partitioning within classes may be needed in order to reduce the overall processing times. The limit for partitioning is, of course, the functional unit as listed above. Still, if processing times are not acceptable with maximum partitioning levels, data partitioning must be implemented within the clusters exhibiting the most intensive computational demands. A trial-and-error approach might be inevitable in the process of searching for the optimum implementation on a specific hardware architecture.

5. MIGRATION TO SHARC PROCESSORS

For the first phase of the ORDA project, SHARC processors were adopted for the multiprocessor DSP subsystem. The proof-of-concept (POC) ORDA utilizes two boards from Mercury Computer Systems based on Analog Devices ADSP-21060 SHARC chips. The system has 18 processors (or compute environments), 13 of which are used by the POC DSP software. These processors run MC/OS, a proprietary, POSIX-compliant, real-time, operating system. Processors within a board and across boards are interconnected via the high-speed, crossbar-switch-based interconnect fabric called RACEway. An implementation of signal processing functionality that is specific to this hardware architecture is described next.

Clustering was performed as shown in Table 1, where functional tasks were grouped following the guidelines introduced in the previous section. In addition, due to the high volume of data, processing speed requirements, and memory availability, data partitioning had to be implemented for clusters 1 and 2. Clusters 1 and 2 take 5 and 3 processors, respectively. The goal for this partitioning was to reach processing times no greater than 60% of the dwell time, therefore allowing for future expansion and/or enhancements. In addition, it was verified that deeper levels of partitioning did not reduce processing time due to data transfers and synchronization overheads.

Cluster	Functions
1	1,2, and 3
2	4 and 5
3	6, 7, 8, 16, and 17
4	9, 10, 11, 12
5	13, 14, and 15

Table 1. Clustering of WSR-88D signal processing functions for the SHARC implementation of the ORDA signal processing subsystem.

6. CONCLUSIONS

This paper described the guidelines for migrating WSR-88D signal processing functionality to open systems. Based on signal processing requirements, the ORDA project adopted a multiple-processor architecture for the implementation of the SPS. The general ideas behind the implementation of a process using real-time multiprocessor systems were described, and several criteria for process partitioning were discussed. Based on the POC ORDA developed by NSSL, which has been operational since May of 2000, the migration of signal processing functionality to SHARC processors was presented last.

The POC design proved able to accommodate all real-time signal-processing functionality as well as satisfy maintainability and expandability requirements. Successful implementation required detailed knowledge of the processor characteristics and resources in order to maximize performance of each processing element. Unlike the implementation of higher-level applications, portability in real-time DSP applications is limited by the underlying hardware. Because of this, NSSL is currently working on porting the signal processing code to PowerPC processors, the proposed architecture for the first build of ORDA.

7. REFERENCES

Ackenhusen, J.G., 1999: *Real-time signal processing: Design and implementation of signal processing systems*, Upper Saddle River: Prentice Hall.

Unisys Corporation, 1989: Preliminary system technical manual NEXRAD. Vol III. Rev C.1.

Unisys Corporation, 1991: Computer program development specification for signal processing program (B5, CPCI 02), DV1208261F.

Unisys Corporation, 1992a: Computer program product specification for signal processing program (C5, CPCI 02), DV1208261D.

Unisys Corporation, 1992b: Critical item development specification for receiver/signal processor (B2, CI 04), DV1208254E.

Unisys Corporation, 1993: Critical item product fabrication specification for receiver/signal processor (C2b, CI 04), DV1208254G.

Zahrai, A., J. Carter, V. Melnikov, and I. Ivic, 1998: The design of a new signal processor subsystem for the WSR-88D (NEXRAD) radar. *Preprints 14th International Conference on IIPS*, Phoenix, AZ, Amer. Meteor. Soc., paper 7.13.

Zahrai, A., S. Torres, I. Ivic, and C. Curtis, 2002: The open radar data acquisition (ORDA) design for the WSR-88D. *Preprints 18th International Conference on IIPS*, Orlando, FL, Amer. Meteor. Soc., paper 5.10.