The ASR-9 Processor Augmentation Card (9-PAC)

James V. Pieronek

Since 1990, the Airport Surveillance Radar-9 (ASR-9) has been commissioned and installed at more than sixty of the largest airports in the United States, and future installations are planned at more than sixty additional airports. After the first several systems were put into daily operation, air traffic controllers began to lodge complaints about the radar's performance. Problems included the detection of "phantom" aircraft caused by the reflection of beacon interrogation signals off buildings and other aircraft, the radar's losing track of targets during parallel approaches and departures, the inability to track highly maneuverable military aircraft through high-G turns, radar clutter caused by highways and weather, and system overloading as a result of signal returns from flocks of migrating birds. An initial investigation of the sources of these problems focused on the radar's post-processor. Nearly all of the problems could be addressed by additions to the post-processor software, but the post-processor was already running near capacity and there was no means for expansion. Thus a new processor—the ASR-9 Processor Augmentation Card (9-PAC)—was designed to augment the existing system to allow for a significant increase in processing power. New algorithms were developed to run in 9-PAC to address the problems cited by the controllers.

The Airport Surveillance Radar-9 (ASR-9) is the newest radar in the FAA's air traffic control (ATC) system. The first FAA radar to use all-digital processing, the ASR-9 provides target reports that are free of the clutter and false alarms characteristic of earlier generations of airport surveillance radars [1]. Since 1990, the ASR-9 has been commissioned and installed at more than sixty of the largest airports in the United States, and future installations are planned at more than sixty additional airports.

Before commencing widespread installation of the ASR-9, the FAA subjected the radar to rigorous specification testing [2]. As the ASR-9 was deployed at operational sites around the country, however, air traffic controllers began to lodge complaints regarding the radar's performance. The most notable of these complaints were

- the detection of "phantom" aircraft caused by the reflection of beacon interrogation signals off buildings and aircraft in the vicinity of the radar,
- the radar's losing track of aircraft during parallel approaches and departures,
- the inability to track highly maneuverable military aircraft through high-G turns,
- radar clutter caused by highways and weather, and
- system overloading as a result of returns generated by flocks of migrating birds.

Lincoln Laboratory was the logical choice to address the problems listed above for several reasons. First, Lincoln Laboratory played a significant role in the development of the ASR-9. In fact, the primary radar section of the ASR-9 is based on the Moving Target Detector-II (MTD-II) radar that was developed by Lincoln Laboratory in the 1970s [3]. Second, Lincoln Laboratory was involved in testing the ASR-9's weather channel [4]. Third, Lincoln Laboratory

has extensive experience with beacon surveillance systems [5]. Lastly, Lincoln Laboratory has recently modified an ASR-8 and an ASR-9 with extensive digital signal and data processing to demonstrate windshear and gust-front detection using airport surveillance radars [6].

An initial investigation of the sources of the ASR-9 problems quickly narrowed our focus to the radar's post-processor. We found that nearly all of the problems could be addressed with substantial modifications to the post-processor software. Unfortunately, the post-processor's memory was almost completely filled by the existing software and no means had been designed into the system for expansion. As a result, we designed a new processor, called the ASR-9 Processor Augmentation Card (9-PAC), to augment the existing post-processor.

Because of the seriousness of the problems with the ASR-9, the 9-PAC hardware and software had to be developed expeditiously. Once the hardware and software tasks were determined, development of the processor card commenced on several fronts. Lincoln Laboratory, the University of Wisconsin, and the FAA Technical Center were responsible for algorithm and software development. Westinghouse Electric Corp., which was contracted by the FAA to develop a test plan for the 9-PAC, began writing test requirements from draft copies of algorithm descriptions that Lincoln Laboratory had provided. Lincoln Laboratory was also responsible for hardware development and, as the hardware design was finalized, it was transferred to Westinghouse (the prime contractor for the ASR-9) for an evaluation of the design's manufacturability. To date, Lincoln Laboratory has built more than thirty 9-PAC cards to be used for initial testing and evaluation by Westinghouse and the FAA at sites exhibiting critical problems. As a result of this concurrent engineering approach, the 9-PAC has gone from concept through development and into testing in two-and-a-half years. In 1995, we expect the FAA to award a contract for the production of more than four hundred 9-PACs to upgrade every ASR-9.

Background: Basics of Airport Surveillance

Two types of surveillance are used to monitor aircraft activity. *Primary,* or *skin,* surveillance refers to normal radar operation in which a radar transmits a pulse of electromagnetic energy that is reflected by the metal components of an aircraft. The reflected signal is received by the radar, which can then determine the aircraft's location: the direction in which the antenna is pointed indicates the direction to the aircraft, and the length of time between transmission of the pulse and reception of the reflection indicates the range to the aircraft. The primary radar is also capable of detecting and classifying precipitation in six levels, from light rain through severe thunderstorms.

In *secondary,* or *beacon,* surveillance, a second antenna mounted on top of the ASR's primary antenna transmits interrogation commands to aircraft. The interrogations, which are transmitted on a different frequency from that used by the primary radar, are received by a device called a *transponder.* All commercial and many general-aviation aircraft are equipped with transponders. Upon receiving an interrogation command, a transponder decodes the command and transmits a pulse-coded reply to the ASR. Depending on the interrogation command, the reply will contain either the aircraft's altitude or identification (ID) code. As with primary surveillance, the location of the aircraft can be determined from the antenna's azimuth and the round-trip time delay between the transmission of the interrogation pulse and the reception of the coded response.

Beacon surveillance gives controllers much better information about the aircraft that they are tracking, but not all aircraft are equipped with beacon transponders. Thus both surveillance systems are used. The redundancy also provides a backup mode when a radar or transponder component is malfunctioning.

The ASR-9

We must understand how the ASR-9 works before we can understand the sources of the radar's various problems. The ASR-9 is an almost fully redundant system. Although it has only a single antenna, the radar has two sets of most other components. These components are grouped into two channels referred to as A and B. Each channel contains all of the components necessary to operate the radar. This design allows one channel to be shut down for servicing while the other channel is left running. Figure 1 shows the

components contained in a single ASR-9 channel [7].

The primary surveillance function operates in the following way. The transmitter emits 1-MW pulses of S-band (2.7 to 2.9 GHz) radiation of approximately 1-μsec duration at 1-msec intervals. The pulses are transmitted in a block-staggered sequence of ten pulses at a high pulse-repetition frequency (PRF) followed by a sequence of eight pulses at a low PRF. Each of these sequences forms a coherent processing interval (CPI). A pair of high- and low-frequency CPIs are combined, and the PRFs of the two CPIs in the pair are related by a 7:9 ratio. Data from the two CPIs are used to unmask targets whose Doppler velocity might be obscured at a single PRF. The approximately 1-kHz PRF results in a maximum unambiguous Doppler-frequency detection range of ±500 Hz (from the Nyquist sampling theorem), which corresponds to an unambiguous velocity range of approximately ±69 nmi/hr at S-band. A target velocity outside this range will be folded into the range as the modulus of the target velocity with respect to the maximum unambiguous velocity. A target moving at a radial velocity that is an exact multiple of the maximum unambiguous velocity will have a detected Doppler velocity of zero and would be indistinguishable from stationary clutter. By utilizing two CPIs at slightly different PRFs, the system will always detect a Doppler velocity in one of the two CPIs (because of their different maximum unambiguous velocities) up to the lowest common multiple of the unambiguous velocities of the two PRFs used in the two CPIs. CPI pairs are transmitted on 256 evenly spaced boundaries in each 360° rotation of the antenna to align data from one radar scan to another. During the small amount of time that is left over between CPI pairs, fill pulses are transmitted. The data received from the fill pulses are not used.

The reflected signal received by the antenna is amplified, bandpass filtered, and quadrature converted to inphase and quadrature (I&Q) video components that are then sampled by 12-bit analog-to-digital (A/D) converters at a 1.29-MHz rate (772 nsec/sample). This processing results in a range resolution (range gate) of 1/16 nmi. The digital I&Q samples are then passed to the digital signal processor (DSP).

The ASR-9 is the first FAA radar to employ an all-digital signal processor. A key benefit of digital signal processing is the ability to utilize a moving target detector (MTD) to sort out moving aircraft from the ground-clutter return that makes up most of the received signal. The first processing step in the signal processor is to organize the I&Q data collected from each CPI pair into range-azimuth bins. Each bin corresponds to a radial patch of space that is 1/16 nmi long and 360°/256 = 1.4° wide. A total of 960 range bins is required to process the data over the 60-nmi coverage of the radar.

The I&Q data in each range bin are passed through 18 Doppler filters corresponding to 18 Doppler velocities. The output of each filter in the Doppler filter bank is compared against a constant false-alarm rate (CFAR) threshold. The CFAR thresholds

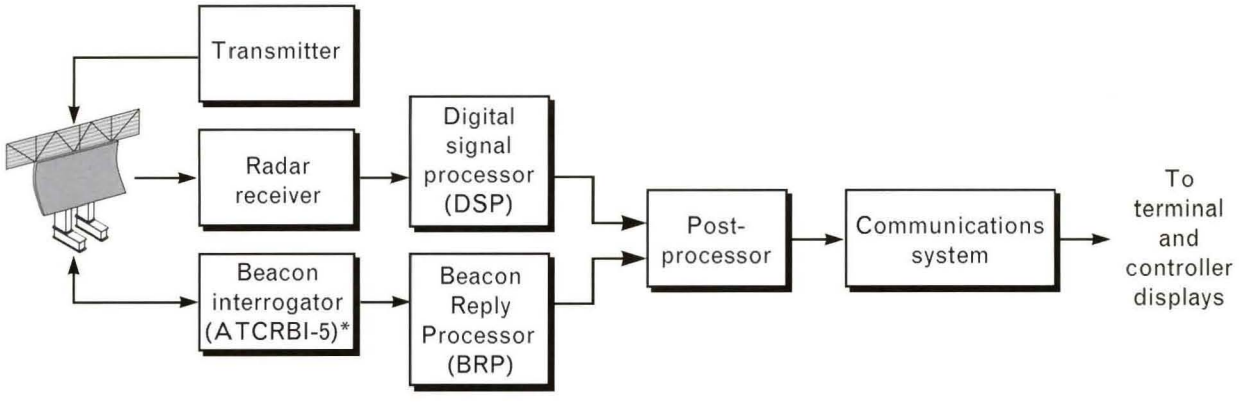

FIGURE 1. Simplified block diagram of one channel of the Airport Surveillance Radar-9 (ASR-9).

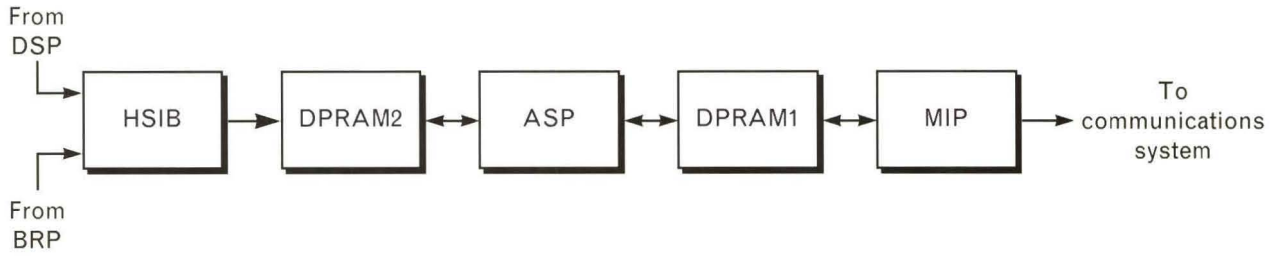

FIGURE 2. Block diagram of the ASR-9 post-processor shown in Figure 1. The post-processor consists of the high-speed interface buffer card (HSIB), two dual-port random-access memory cards (DPRAM1 and DPRAM2), Array Signal Processor (ASP), and Message Interface Processor (MIP).

for the zero-velocity filters are calculated by averaging the noise in each range-azimuth bin over a period of eight scans. The CFAR thresholds for the nonzero-velocity filters are calculated from the mean value of the filter over a 27-gate sliding window, i.e., from the filter outputs for the 13 range-azimuth bins preceding and 13 range-azimuth bins following the cell in range. If any of the Doppler filter outputs exceed their corresponding CFAR threshold, a *primitive target-detection report,* or *primitive report,* is generated for that range-azimuth bin containing the filter outputs. Next, the primitive reports are tested against a *geocensor map* of information about clutter sources on the ground, e.g., highways and buildings. The geocensor information is added to the primitive report and is used in later processing as an aid in determining whether a report was caused by clutter. The primitive report is then passed to the post-processor.

The secondary surveillance system contains components that are not actually part of the ASR-9. At most ASR-9 sites, an Air Traffic Control Radar Beacon Interrogator-5 (ATCRBI-5) [8, 9] is connected to the ASR-9. The ATCRBI-5 components used with the ASR-9 consist of a transmitter, antenna, and receiver. The ASR-9 commands the ATCRBI-5 to send an interrogation pulse at a PRF of approximately 400 Hz, interleaved between the primary-radar transmissions. A hardware function in the ASR-9 called the Beacon Reply Processor (BRP) extracts beacon transponder codes from the received signal, adds a range value calculated from the delay between the transmitted pulse and the received reply, and passes all of this information to the post-processor.

As is evident from the discussion above and Figure 1, the primary and secondary surveillance functions are completely independent of each other up to the point that primitive reports are generated. The primitive-report streams come together inside the post-processor, which generates the completed reports that are sent over the communications system and eventually are displayed on the controller's screens.

The ASR-9 Post-Processor

The block diagram in Figure 2 shows the major components of the ASR-9 post-processor. All of the components discussed here are located with the DSP and BRP in a large (3 ft × 3 ft) card rack in the ASR-9 receiver/processor cabinet.

The data streams of primary (or radar) and secondary (or beacon) primitive reports are supplied to the high-speed interface buffer (HSIB) card, which writes the data from the two streams into a dual-port random-access memory (DPRAM) card labeled DPRAM2 in Figure 2. (A dual-port memory is a memory device that can, in essence, be accessed independently by two devices connected to the memory's two ports.) Attached to the other port of DPRAM2 is the five-card Array Signal Processor (ASP), which performs all of the processing that converts the radar and beacon primitive reports into completed target reports. The ASP writes the completed reports into another DPRAM, which is labeled DPRAM1 in Figure 2. The other port of DPRAM1 is connected to the Message Interface Processor (MIP), which retrieves the completed reports and sends them through the communications system to the Terminal Radar Approach Control (TRACON) and tower for eventual display on the controller's screens.

Figure 3 shows all of the tasks that the ASP software performs. As the diagram indicates, the ASP converts the radar and beacon primitive reports into completed reports and then merges and tracks the reports to form the output report stream that is sent to the MIP. A detailed description of each of the ASP functions follows.

Correlation and Interpolation (C&I). As the ASR-9 antenna scans past a target, the DSP generates multiple primitive target-detection reports. These reports will occur over a small extent in range and azimuth, centered on the actual location of the target. C&I analyzes the multiple primitive reports and generates a completed target report that contains a single location, or centroid, for the group.

The first function of C&I is to group, or correlate, all of the primitive detections that belong to a target. C&I performs this function by using the proximity of the primitive detections in range. This simple grouping process becomes complicated when two targets are close together or when they overlap. For example, consider a situation in which an aircraft crosses directly over another aircraft. Because the primary radar portion of the ASR-9 has no means for separating returned signals by altitude, the primitive detections from the two targets will overlap. To address such situations, the correlation software detects primitive groups exhibiting an abnormally large extent in range, and subjects each of these groups to a range-resolution test to determine whether a group should be split into two target groups. A tentative range centroid is assigned to each group at this point.

As additional primitive detections are received from the DSP, the interpolation function of C&I updates report groups in the azimuth direction. When new primitive detections are received, they are compared with existing report groups and, depending on their range proximity, are added to the groups.

Once the primitive reports have been grouped properly, the groups have been split (if necessary), and the DSP has stopped sending reports that can be added to the groups, C&I can begin to generate reports. The range centroid of a group is determined by comparing the peak return magnitudes of all reports in the group and selecting the range associated with the largest peak return magnitude. If the peak magnitudes in two adjacent range bins are equal, a *straddle flag* in the completed report is set, indicating that the target falls on the 1/32-nmi boundary between the two range bins. Computation of the azimuthal centroid is somewhat more complicated. If primitives from two CPI pairs exist in the group, C&I interpolates the azimuth linearly between these two points by using the peak return magnitude of the two primitive reports. If primitives from three or more CPI pairs are present, a beamshape-match algorithm [10] is used to fit the three points to the shape of the beam, and the interpolated beam center is used as the azimuth centroid. If the group's azimuth profile differs substantially from the known beamshape, the group will be split and two target reports will be generated.

Doppler velocity is interpolated individually for the high- and low-PRF CPIs. For each CPI, the Doppler velocity is calculated by interpolating between the two largest adjacent filter outputs. If multiple CPI pairs are used to form the report, the interpolated high- and low-PRF velocities from the multiple primitives are averaged (with a single-pole filter) for the completed report.

The completed radar target report contains the centroid, the peak filter magnitudes from the primitive detections used to create the report, the interpolated Doppler velocity for each CPI, and a set of ancillary information that is used later in the processing. The ancillary information consists of a quality index and a set of confidence and miscellaneous flags. The quality index is based on the number of primitive detections used to create the report. The confidence flags indicate whether the target overlaps the geocensor road or clutter maps. Miscellaneous flags are used to indicate the presence of jammers and other sources of interference.

The target reports are subjected to a second test that uses adaptive thresholds to remove weak targets that may be caused by birds, insects, or anomalous propagation. These targets will have low return amplitudes. A CFAR function similar to that described for the signal processor performs the second adaptive-threshold test. (The CFAR signal processor test described earlier is considered the first adaptive-threshold test.) By lumping many range-azimuth cells together into large zones, the second adaptive-thresh-

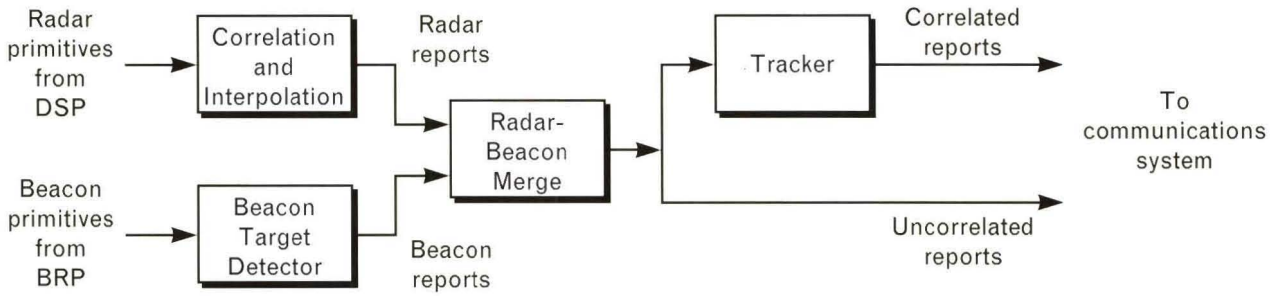

FIGURE 3. Data flow for the five-card ASP shown in Figure 2.

old test works on a very coarse scale. The threshold for each zone is determined by counting the rate at which targets are declared within the zone. As the detection rate goes up, the threshold level rises to adjust the false-alarm rate. Targets with high amplitudes and targets that are associated with beacons are excluded from this process to keep the thresholds from becoming so high that actual aircraft are rejected. The completed reports are passed to the Radar-Beacon Merge function.

Beacon Target Detector (BTD). As with the primary radar, the beacon surveillance interrogator/receiver will elicit multiple transponder replies as the antenna sweeps past an aircraft. During such a sweep, most transponders will receive and reply to approximately 18 interrogations. Two-thirds of the interrogations will be Mode 3/A, which requests the transponder's 12-bit identification code, and the remaining one-third will be Mode C, which requests the aircraft's altitude.

The function of the BTD is to group all of the replies from an aircraft into a single target report. BTD is even more susceptible than C&I to the problem of receiving responses from multiple aircraft because the transponder replies have a nominal duration of 21 μsec, which, given the speed of the transponder transmissions, results in each reply occupying approximately 1.5 mi in range. Thus, if two aircraft are within 1.4° of the same azimuth with respect to the radar and within 1.5 mi in range (we assume that the planes are at different altitudes), the transponder replies from the two aircraft will overlap. This problem is particularly vexing because the coded pulses from the two transponders may be garbled, i.e., mixed together in such a fashion as to be deemed incorrect. Usually, however, one of the transponders in this scenario will begin replying slightly before the other and the radar will receive a few clear replies that can be used to create a target report. Likewise, the other transponder will continue to generate replies after the first transponder has stopped, emitting a few clear replies for the creation of a second target report.

Radar-Beacon Merge (Merge). Target reports from C&I and BTD are passed to the Merge function, which combines the appropriate reports into single *merged* targets. A merged target imparts a higher amount of confidence than a *radar-only* or *beacon-only* target because a merged target indicates detection by both the primary and secondary surveillance systems.

Merge operates by maintaining a list of recent radar target reports from C&I and a list of recent beacon target reports from BTD. As new beacon target reports arrive they are checked against the list of radar target reports for geographic coincidence. When a match is found, a merged radar-beacon report is created from the two matching reports. A similar process takes place as new radar reports are received from C&I.

When a report has been on either the C&I or BTD list for approximately 11° of antenna rotation without being matched, Merge releases the report to the communications system as a radar-only or beacon-only report. This procedure is necessary to meet the FAA specification for maximum delay between the time that the antenna is pointing at a target to the time that the target is displayed on the controller's screen.

Tracker. After the Merge function has sent a report to the communications system, a copy of the report is

also sent to the Tracker function, which attempts to associate the report with one of the tracks that are maintained in a track file. If a geographic match is found, Tracker generates a *correlated report* and sends it to the communications system. Correlated reports indicate to the controller that the target on the display has been seen on previous rotations of the antenna, and that the target is behaving appropriately for a real aircraft. The Tracking function is very useful for determining whether a radar-only target is a real aircraft or if it is ground clutter or random noise. Controller displays are usually set to show all target reports except for uncorrelated reports of radar-only targets. The ASR-9 specification allows C&I to make 100 false-alarm uncorrelated radar-only reports per scan, but only one false correlated report per scan.

If no corresponding track is found for a report, a potential new track is started. For a potential track to be upgraded to an existing track, a target must be seen for several rotations of the antenna. If an existing track does not receive a target update from Merge on a particular scan, the track is kept (or *coasted*) for up to three scans before being deleted. A coasting track will not generate an output in the absence of a matching report; i.e., only a real radar, beacon, or radar-beacon report that matches an existing track can generate an output.

Output and Maintenance Functions. Completed reports from the Merge and Tracker processes are passed to MIP via DPRAM1, a dual-port memory. DPRAM1 also enables MIP to control and test the ASP with several special-purpose control signals. The most important of these control functions are the initialization of the ASP and the continuous background testing of the ASP functions and memory. To ensure that the ASP is functioning properly, the ASR-9 system inserts several test messages into the data streams that enter the ASP, and the messages are subsequently detected by MIP.

Technological Limitations of the ASP. The ASP processing element is built with bit-slice processor components, and all of the data paths in the ASP are 16 bits wide. The ASP processing element has no floating-point math capability, nor does it have an instruction for integer division. With a modified Harvard architecture (i.e., data and instructions are carried on separate buses), the ASP can perform limited simultaneous operations on its five internal buses. The ASP has access to only a very limited amount of memory—256 kB of random-access memory (RAM) populate DPRAM1 and DPRAM2.

With a clock speed of approximately 8 MHz, the ASP's processing capacity is similar to that of an Intel 386SX processor running at 8 MHz. For the time that it was designed (the mid-1970s), the ASP was a powerful machine. Bit-slice processors like the ASP, however, have been supplanted over the past decade by high-performance microprocessors and digital signal processors. Although the ASP is capable of executing an instruction on virtually every clock cycle, it is not as fast as modern microprocessors that run at clock speeds of up to (and even greater than) 100 MHz and that can execute multiple floating-point and integer operations per clock cycle.

The ASP is programmed in microcode, a device-specific language that is one level lower (i.e., one level further removed from human language) than assembly language. Working with ASP microcode is difficult: only a handful of programmers are familiar with the code, very few tools exist to help, and the tools that do exist run on computers that are becoming obsolete. Thus, from the outset of this project, a key goal has been to replace the ASP microcode with a high-level language for which there exists a large base of experienced programmers. Accomplishing this goal will ensure the availability of programmers with the skills that are necessary to support the maintenance and improvement of the ASR-9 system over its predicted lifetime of twenty years.

9-PAC Radar and Beacon Processing Algorithms

As mentioned at the beginning of this article, several unanticipated problems surfaced after the ASR-9 had been running in the field for a short time. Some of the problems that were reported were no different from the shortcomings of the older ASR-7 and ASR-8 systems but, because the ASR-9 provides a cleaner view of targets, the flaws were more evident.

It was immediately obvious that the way to correct the problems was to modify or, in some cases, rewrite the ASP software. In fact, for several years personnel at the FAA Technical Center in Atlantic City, New

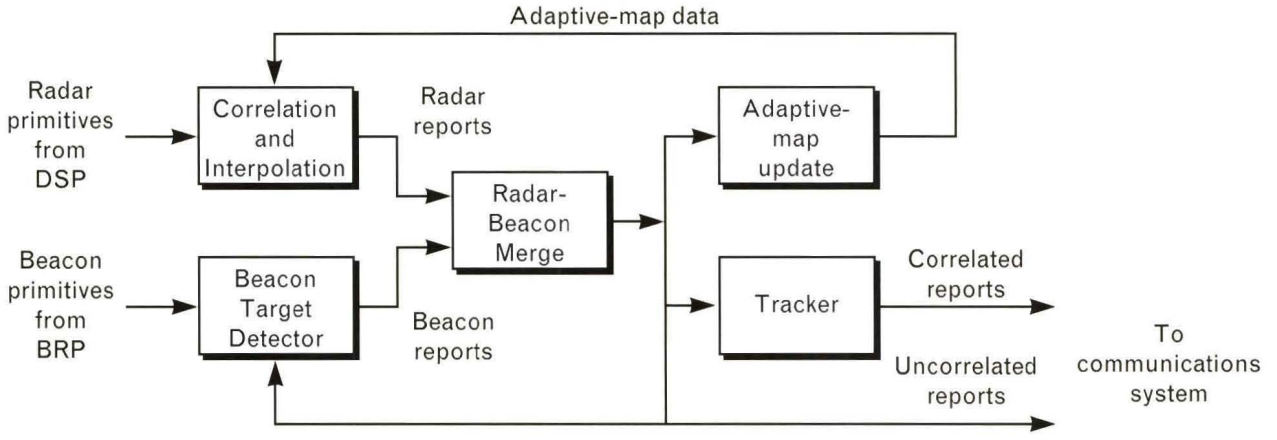

FIGURE 4. Data flow in 9-PAC.

Jersey, had been modifying the ASP code to correct problems. Such efforts, however, were limited by the ASP's capabilities and memory capacity.

The first step in developing 9-PAC was to define the new software functions that were needed to correct the problems. For the C&I function, the microcode was translated manually to the C programming language and then modified. The other three functions—BTD, Merge, and Tracker—required changes that mandated a complete rewrite of the code. The following subsections describe the enhancements made to the ASP software functions in the 9-PAC software. Figure 4 shows the new data flow used in 9-PAC.

Enhancements to C&I

The 9-PAC implementation of C&I is essentially a direct translation of the ASP microcode with the exception of two functions: geocensoring and the second adaptive-threshold testing.

The standard ASR-9 deals with road clutter by using a geocensor map that flags areas of known clutter. Using information obtained from the map, a processing step in the ASP tracker inhibits low-quality targets from initiating tracks in areas of known clutter. One of the main problems with the geocensor map is that it requires operator intervention for updating the map. For example, if a new road is opened or a new building is erected, an operator must go to the radar site, hook up a workstation to the radar to record the necessary clutter data, generate and edit the new map in the workstation, and load the completed map into the radar. This process can take several days if weather conditions are not amenable for collecting the clutter data.

The ASR-9 second adaptive-threshold test (the first adaptive-threshold test is the CFAR filter in the signal processor) utilizes a map of 121 adaptive-threshold cells. These cells are arranged radially in eight concentric rings with each cell incorporating an area of several square miles. The purpose of the second adaptive-threshold test is to censor targets that are likely to be bird flocks, insects, or atmospheric effects. The second adaptive-threshold test operates by keeping track of the number of radar targets that occur in each cell. When a cell shows an abnormally high occurrence of targets, the detection threshold for that cell is increased to bring the cell's target-detection rate back in line with the rest of the cells. The large size of the cells can cause problems in the vicinity of an airport. For example, a construction project involving large earth-moving equipment and/or cranes can cause the detection thresholds for a particular cell to increase to the point at which small aircraft passing through that cell will not generate a primary return of sufficient amplitude to be detected.

In 9-PAC, adaptive-threshold tests occur in two steps. In the first step, which occurs in C&I, target data are tested against the thresholds in the adaptive maps. If a target has a return amplitude that is lower than the threshold from the map, a flag is set in the target report indicating so. The reports then pass

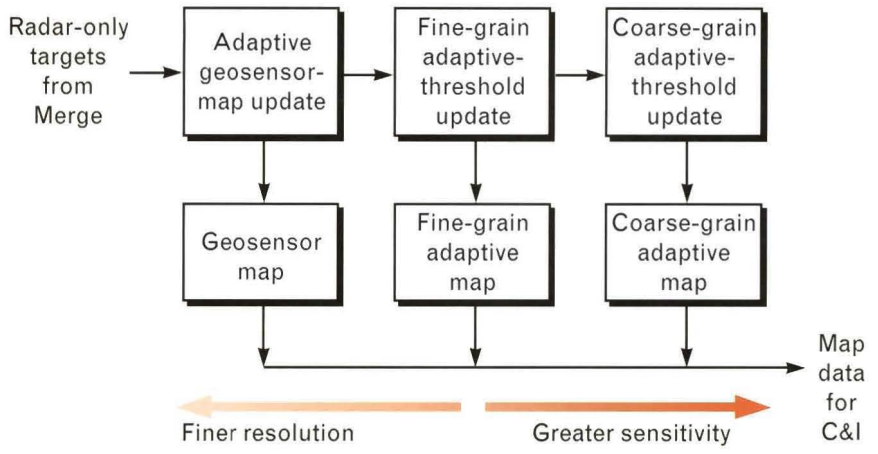

FIGURE 5. Three-stage adaptive-threshold process.

through the rest of C&I and the Merge function. In the second step, targets that come out of the Merge function are used to update the adaptive-threshold maps (Figure 4). Although all targets that come from Merge are passed to the map-update task, just the radar-only targets are used to update the maps.

To deal with the geocensoring and adaptive-threshold problems mentioned above, we deployed a three-layered adaptive-threshold map-update mechanism (Figure 5). The three steps—geocensor, fine-grain, and coarse-grain adaptive-threshold update—occur in order of increasing sensitivity and decreasing resolution. The geocensor map is generated and updated automatically by the system to eliminate the need for manual updates. The three maps work together to achieve the goal of providing censoring for small regions of strong clutter (such as the example of the construction project mentioned earlier), while also providing a means to detect and respond to weak and spatially diffuse clutter such as bird flocks, weather, and atmospheric effects.

The first step in the map-update process involves the geocensor map, which uses the smallest range-azimuth cell (1/16 nmi × 0.7°) of the three maps. The geocensor map requires a large rate of radar-only reports to occur in a cell before the cell is declared active. Once a cell has been declared active, however, it remains active for several days after the high rate of detection drops. The long hold time is designed expressly for maintaining the map, which consists principally of roads, over a four-day weekend, when traffic patterns tend to be less intense than during working days [11]. Figure 6 shows a comparison between the geocensor map used in a standard ASR-9 and a geocensor map generated by 9-PAC.

Targets that were not flagged by the geocensor map while passing through C&I are used to update the fine-grain adaptive thresholds. The fine-grain thresholds are stored in an *x-y* map with cells that are ½ nmi × ½ nmi. The fine-grain map, which adjusts the detection thresholds used by the C&I process, responds to lower detection rates than does the geocensor map. As the detection rate increases, the threshold also increases. Unlike the geocensor map, the threshold-update function in the fine-grain map does not require the detection rate to persist for a long time, nor does the fine-grain map keep the threshold values heightened for a long time after the detection rate has dropped. An influence region for each cell in the fine-grain map is defined as a square that is 1 nmi × 1 nmi centered on the cell (Figure 7). All targets falling in a cell's influence region are used to update the cell's histogram. This process gives a cell "advance notice" of slow-moving clutter like bird flocks and weather that might be present in adjacent cells.

Targets that were not flagged by the fine-grain map while passing through C&I are used to update the coarse-grain map. The process used to update the coarse-grain map is identical to that used to update the fine-grain map, with a few exceptions. The resolution of the coarse-grain map is 2 nmi × 2 nmi. The area of influence is correspondingly larger: 4 nmi ×

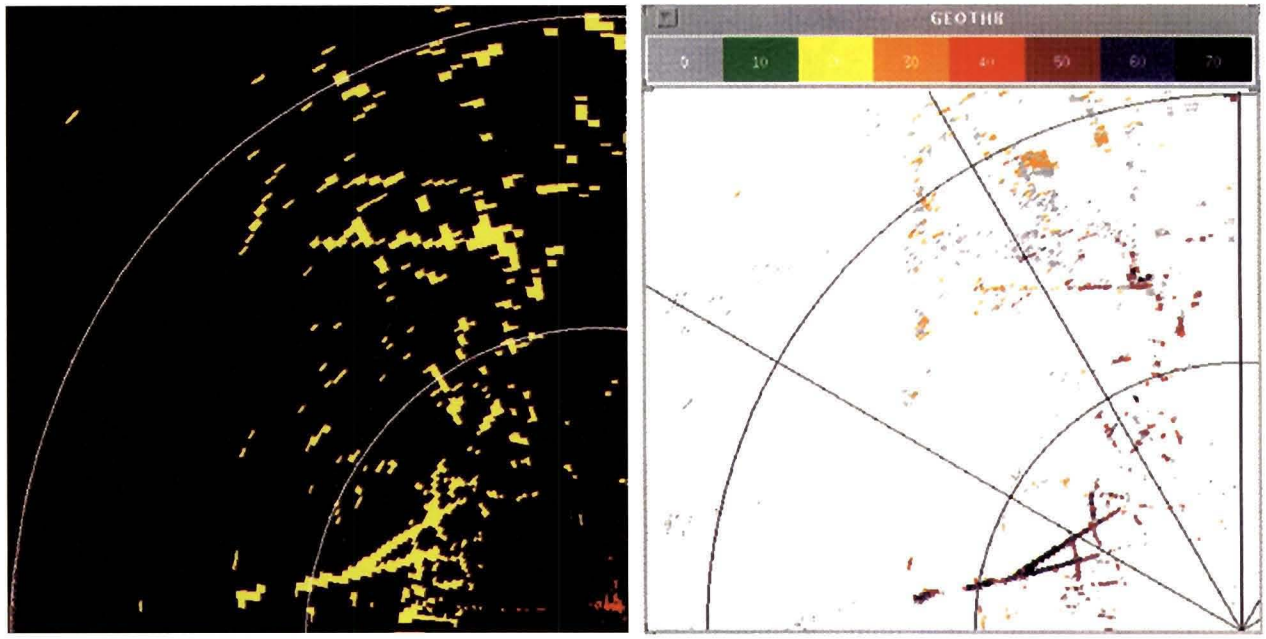

FIGURE 6. Comparison of geocensor maps for (a) standard ASR-9 and (b) ASR-9 with 9-PAC. The standard ASR-9 map is binary, whereas the 9-PAC map has variable threshold levels (as indicated by the color bar in relative dB). Note 9-PAC's increased detail of the roads just inside the 10-nmi range ring.

4 nmi. Also, the coarse-grain map responds to even lower detection rates than does the fine-grain map. Lastly, the coarse-grain map has shorter persistence and hold times than does the fine-grain map.

For an example of how the maps work together, we consider a flock of birds. As the birds take flight together, they will first cause the coarse-grain map to respond quickly by raising its thresholds, thus leading to the birds being censored. If the flock of birds is small and dense, the flock will next cause a smaller section of the fine-grain map to respond by raising its thresholds. Once the birds are being censored by the fine-grain map, the returns that they generate will no longer be supplied to the update process for the coarse-grain map and the coarse-grain thresholds will drop back to their original levels. This example can be extended to a construction site or highway interchange that would pass through the coarse- and fine-grain steps and eventually trigger a response by the geocensor map. Using this three-step process, 9-PAC provides an initial rapid response to the clutter, then localizes the clutter to as small an area as possible to lessen its impact on the surrounding area.

Enhancements to BTD

The beacon interrogation system uses a sophisticated method of multiple-pulse interrogation with directional and omnidirectional antenna patterns to minimize the chance of eliciting a response from an aircraft that is not in the main beam of the antenna pattern. Unfortunately, this system cannot prevent interrogations from being reflected in other directions by objects in the path of the antenna's main beam. In the example shown in Figure 8, interrogations are reflected off a building, causing an aircraft to be interrogated twice per antenna revolution—once when the antenna is pointed directly at the aircraft, and another time when the antenna is pointed at the building and the interrogation (and transponder reply) is reflected off the building. The aircraft will thus appear on the controller's screen in two places—one target appears in the correct location, the other target appears in the direction of the building at a slightly longer range than the actual range because of the longer path that the signal follows when it is reflected [12].

Because the transponder system is able to identify individual aircraft, a straightforward approach to solving reflection problems would be to store the range of all aircraft that are interrogated so that, when two aircraft with the same beacon identifier appear at different locations during the same antenna scan, the target with the greater range can be deleted as the target generated by reflection. This solution, however, presents two difficulties. First, not all beacon identifiers are unique. Because ID codes in the 1200s (e.g., 1207, 1235, 1262) and any code ending in 00 can be assigned to multiple general-aviation aircraft, targets with these codes cannot be removed from the display. Second, tests have shown that even the discrete codes that are supposed to be assigned to only one commercial aircraft at a time are occasionally assigned to multiple aircraft because the hardware can support only 4096 possible codes; i.e., not all discrete codes are truly unique. Thus a considerably more complicated algorithm is required to solve reflection problems.

To handle nondiscrete codes, we must construct a map of known reflectors by using information gleaned from reflections involving discrete-code aircraft. Once the locations of reflectors are known, we can check each aircraft against the map to determine if the aircraft is real or if it fits the geometry for a reflection. This test works for both discrete and nondiscrete codes. To resolve the case of multiple aircraft transmitting the same discrete code (and also flying at the same altitude), the BTD algorithm checks for the existence of a primary-radar report of a spatially matching target for the suspect duplicate code. If a corresponding primary-radar report is found, indicating that the aircraft is, in fact, really there, the BTD algorithm will not generate a reflector-map entry based on the target geometry of the multiple identical discrete codes, nor will the suspect reflected targets be deleted. A feedback path from the Radar-Beacon Merge function to the BTD is used to inform the beacon processing software that a matching primary report has been found.

But a further difficulty can arise with the use of reflector maps. If a large aircraft is parked such that its tail becomes a reflector, the aircraft will be saved in the reflector map and will remain in the map even after the vehicle has moved. Two maps are used to remedy such situations: a temporary map and a permanent map. The temporary map stores reflectors as they are discovered. New reflectors that remain for a period of twenty-four hours are promoted to the permanent map. Once a reflector has been placed in the permanent map, it can be removed only after it has not been observed for a period of twenty-one days.

As mentioned earlier, when two or more aircraft are in the same vicinity, interference of their transponder replies can produce garbled reports. In areas of high traffic density, particularly in areas where aircraft are making parallel runway approaches, garble is a constant problem. The original ASR-9 BRP and BTD take only moderate steps to deal with garble. Although not specifically cited as a problem with the ASR-9, garble had to be addressed to make the beacon reflection code more robust.

Garble can be dealt with in a number of ways. The easiest method attempts to unscramble garble by using adjacent codes. For example, when two aircraft are making a parallel approach to an airport, the aircraft may appear, from the radar's vantage point, to be side by side. As the beam of the interrogator antenna sweeps past the two aircraft, the beam may encounter only one of the aircraft at first, then both of the aircraft, and finally only the second aircraft. In such a scenario, the transponder returns will start with clear (ungarbled) replies from the first aircraft alone, followed by garbled replies resulting from the simultaneous interrogation of both aircraft, and finally clear

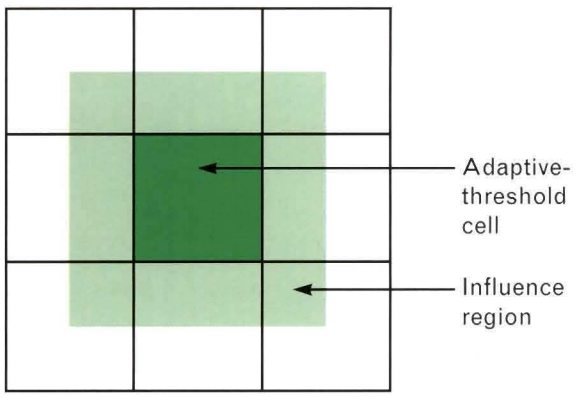

FIGURE 7. Adaptive-threshold cell (1/2 nmi × 1/2 nmi) and influence region (1 mi × 1 mi) used to update the fine-grain map. All targets falling in a cell's influence region are used to update the cell's histogram.

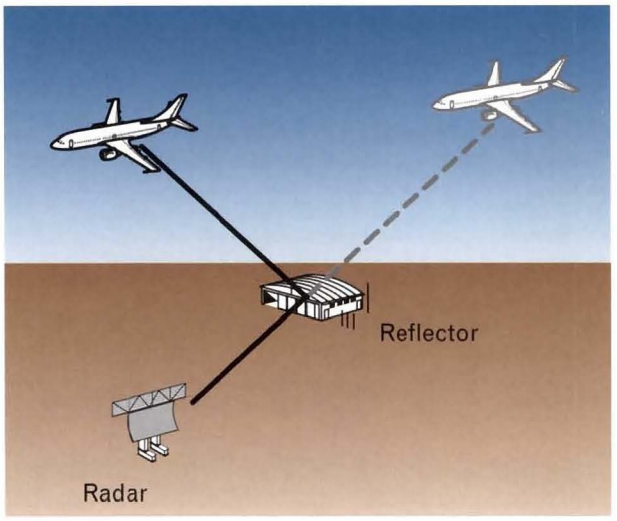

FIGURE 8. Example of an error caused by an interfering object. Beacon interrogations from the radar are reflected off a building, causing an aircraft to be interrogated twice per antenna revolution—once when the antenna is pointed directly at the aircraft, and another time when the antenna is pointed at the building and the interrogation (and transponder reply) is reflected off the building. The aircraft will thus appear on the controller's screen in two places.

replies from the second aircraft alone as the first aircraft passes out of the beam. If there are enough clear codes at the beginning and end of the reply stream, we can re-create the actual replies from the two aircraft and determine the correct centroid for each vehicle.

Data collected at several locations around the United States have shown that the above adjacent-code method will not correct all garble cases. For instance, garble that is not detected as such by the BRP can cause replies to be erroneously split into two reports, creating a report for a nonexistent aircraft. Thus a more robust method was developed to handle cases of garble that do not fit the simple scenario described above.

The new method requires the use of a separate beacon tracker that maintains tracks of targets whose ID codes are known. When garbled replies are received they are checked against this track map to determine if they closely match an existing track in that geographic vicinity. When a match is found, the garbled code is corrected and the processing continued. This track-matching method is considerably more expensive than the adjacent-code method described earlier because of the tracker's computational demands and memory requirements for storage of the tracks.

Enhancements to Radar-Beacon Merge

The Merge function in the ASP compares incoming radar reports with a list of non-merged beacon reports and, for a given radar report, will generate a merged report with the first beacon report that is found to fall within a specified rectangle, or *association box,* around the given radar report. An identical process is used to compare incoming beacon reports against a list of radar reports. This method can create an incorrect merge when there is more than one potential match in the association box. In an extreme case, the identification tags for two aircraft flying in parallel may hop between two aircraft. The 9-PAC Merge function seeks the best match within the association box by using a modified Munkres algorithm [13]. The algorithm uses a scoring approach to determine the best match with geographic proximity carrying the highest weight.

Special processing was added to the Merge function to deal with beacon reports generated by the BTD function that were flagged as false targets caused by reflections. If a beacon report has been flagged as a false target caused by reflection from a permanent reflector (a "supported" report), Merge will drop the report. If the flagged beacon report merges with a radar report, both reports will be deleted. Even if the flagged beacon report is not supported by a permanent reflector, the report will be dropped if it does not merge with a radar report. But, if an unsupported (i.e., not supported by a permanent reflector) flagged beacon report merges with a radar report, Merge will turn off the false-target flag, output the report as a radar-beacon merged report, and send a message back to the BTD process to indicate that the beacon target has been merged with a radar target. As described in the subsection "Enhancements to BTD," this information is used by BTD to determine whether an ID code has been assigned to multiple aircraft.

Enhancements to Tracker

The original ASR-9 Tracker was designed to accommodate targets with a maximum turning rate of

0.5G. Military aircraft, however, are capable of turning rates of considerably higher G. There are two problems with tracking these highly maneuverable targets. First, the original radar performed most of its tracking by using a polar ρ-θ coordinate system corresponding to the range-azimuth radar data because the ASP does not have the computational capacity necessary to convert all target reports to an x-y coordinate system. The ρ-θ system is acceptable for aircraft flying several miles away from the radar, but it has trouble handling high-speed targets flying near the radar site. For example, if a high-speed aircraft were flying due west and flew over the radar site (i.e., the origin of the ρ-θ system), the azimuth of the aircraft would change almost instantaneously from 90° to 270°. Second, the ASR-9 Tracker uses geographic association boxes to match incoming target reports with existing tracks. Tracking highly maneuverable targets requires the use of larger association boxes to handle the larger space of potential movement for a given target. The larger association boxes entail larger searches of the target and track databases, for which more processing power is necessary.

The 9-PAC Tracker is a full x-y tracker that utilizes conventional tracking in both coordinates. Parameters are adjusted automatically to deal with variations in report quality, track history, and track residual history. (Note: A residual is the difference between the predicted and actual location of a target.)

In the 9-PAC Tracker, the track-initiation process generates more trial tracks than the original tracker, and this enhancement can lead to the introduction of more false tracks and to the inclusion of false alarms in existing tracks. To mitigate these problems, we added several additional tests to the 9-PAC Tracker to prevent false-alarm reports from generating tracks or adding false information to existing tracks. A velocity and acceleration test is used to determine whether a target is moving in accordance with a simple model of aircraft performance. In addition, the correlation step in the 9-PAC Tracker uses a scoring mechanism to associate targets with tracks. The correlation step incorporates the quality and confidence information in the radar reports, and checks for agreement between the radial component of the tracked velocity and the radial Doppler velocity that is derived from the Doppler velocity estimates from each of the two CPIs in the report. As mentioned earlier, these velocity estimates are limited by the maximum unambiguous Doppler velocity for each CPI (approximately 69 nmi/hr). Because the ratio of the maximum unambiguous velocities of the two CPIs is fixed at 7:9, the radial Doppler velocity can be computed by applying the Chinese remainder theorem to the two CPI velocity estimates in the report. This estimate of radial velocity has a range of approximately ±600 nmi/hr.

A feature was added to the 9-PAC Tracker to deal with collimation error—the difference in target location as reported by the radar and beacon systems. By default, the ASR-9 selects the radar report's range and azimuth to use in a merged report. If a track that has had radar reports should miss a radar report and receive only a beacon report, the range and azimuth from the beacon report are used in the output. If the system contains collimation error, the tracked beacon-only report will reflect the discrepancy in that the track will appear to jump on the controller's screen. For such cases, the 9-PAC Tracker adjusts the range and azimuth values of the tracked beacon-only report to compensate for the collimation error. Collimation statistics are calculated during the formation of radar-beacon merged reports. Figure 9 is a comparison of the output from the standard ASR-9 tracker compared with the 9-PAC tracker for the same input dataset.

9-PAC Hardware

Once a basic set of algorithmic solutions to the major problems in the system had been determined, we needed to find a way to enhance the ASP's computing capability because the ASP was running near capacity in terms of memory and processor utilization. Several means were considered to achieve this enhancement.

An obvious option was to add a powerful workstation external to the system and use specialized hardware to connect the workstation to the ASP. One problem with this solution was that it would have been difficult for the ASR-9 processor to obtain complete control of the external hardware because most workstations are designed to operate autonomously. Furthermore, space where the workstation

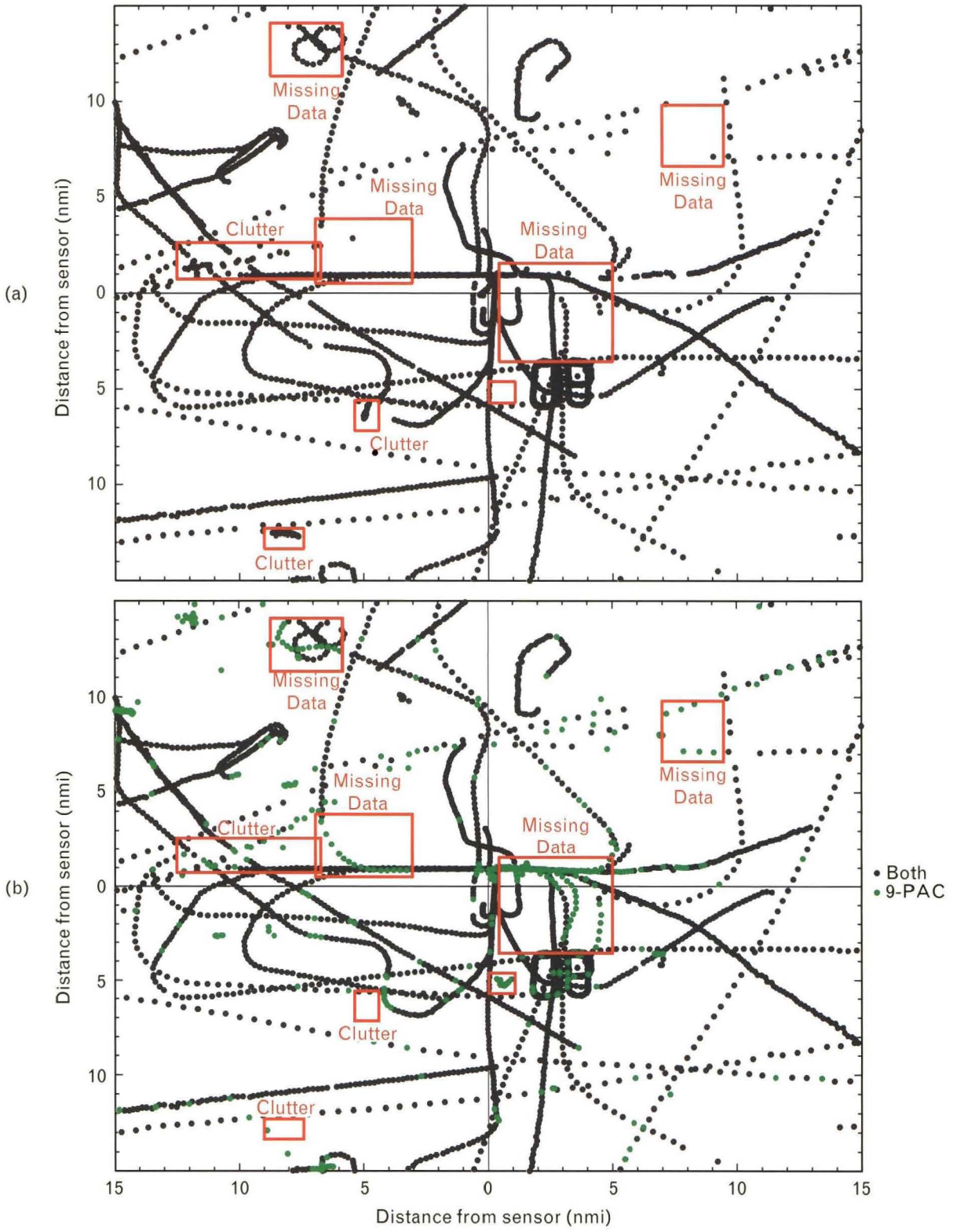

FIGURE 9. Comparison of primary-radar performance: (a) ASR-9 production unit and (b) 9-PAC output for the same 20-min set of data obtained at the Albuquerque, New Mexico, site. Note how 9-PAC enabled the rejection of clutter and enhancement of track detection.

could operate undisturbed would be needed in the radar shelter.

Another option considered was the addition of a small computer chassis inside the ASR-9 receiver/processor cabinet where the ASP resides. Again, specialized hardware would have to be built to connect the new processor to the system. A computer of this sort would be much more suited to the operational environment of the ASR-9, and the ASR-9 would have complete control over such a machine. The problem with this option was that it would have taken considerable effort to install such a computer.

A third option was to replace the ASP completely with a set of custom processing cards that would fit into the same slots in the card cage. Unlike modern computers, however, the backplane wiring that connects the ASP cards together does not consist of a uniform bus that connects the same pins together on each of the cards. Instead, the ASP backplane is point-to-point wired with a specialized interface between each card. Thus use of the existing wiring was considered very difficult. But removal of the ASP altogether meant that any modifications would have to duplicate fully the current functionality of the ASP; i.e., a large software effort would be required.

Because the three options considered required the construction of specialized hardware, we began studying the ASP schematics to see if there was a way to add a processor without disabling the ASP. Our investigation quickly focused on DPRAM1 and DPRAM2 because of several factors: the two boards are identical, they are each 7.5 in × 11.3 in, and, by current standards, they contain a small amount of memory. Each DPRAM board uses almost 60 in^2 to hold 64 integrated circuits (IC) containing a total of 128 kB. With current technology, the same 128 kB of memory can be supplied by a single IC that occupies less than 1 in^2 of space on a circuit board.

Having found as much as 120 in^2 of space that could be used for adding components on the two DPRAM boards, we investigated the next task of finding a way for the ASP to communicate with the added processor. Because the ASP already had access to the memory on both DPRAM1 and DPRAM2, all that was needed was another port to this memory, thus making it multi-ported, to allow the ASP to communicate with the added processor (Figure 10).

There are several benefits to this approach. First, should the added processor fail, the new board containing the processor could still function as a dual-port memory board; i.e., if the ASP determines that the added processor is not operating properly, the post-processor could return to the original operating mode and keep running. Another benefit is that, because DPRAM2 receives all of the radar and beacon input from the DSP and BRP directly via the HSIB, the added processor could also access these data directly if it were inserted in the DPRAM2 slot. Thus, by replacing DPRAM2 with a multi-port RAM that has an added processor connected to the additional memory port, we could provide additional processing to assist or augment the existing ASP without removing or disturbing the ASP. Hence the new board containing the multi-port RAM and added processor was named the ASR-9 Processor Augmentation Card, or 9-PAC.

Design Goals

Once we decided to replace DPRAM2, we sought ways to make the new board as versatile as possible. Our objective was to place the maximum amount of processor power and memory on 9-PAC, and to add several high-speed interfaces to allow for data recording and future expansion. Non-volatile memory was

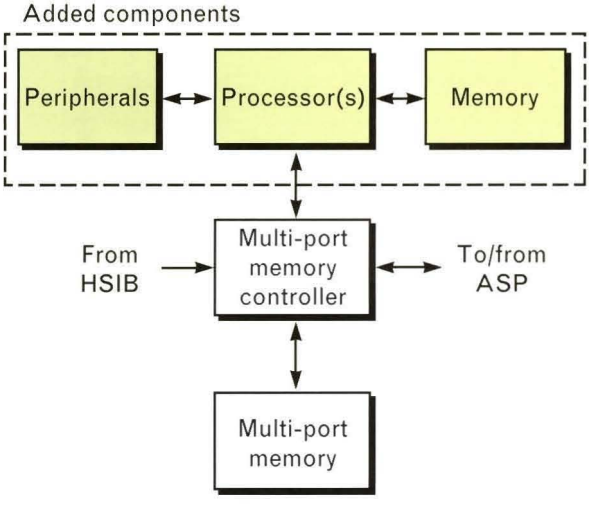

FIGURE 10. Multi-port random-access memory (RAM) with added components used in 9-PAC. This board was used to replace DPRAM2.

needed for storing the software and the various maps and databases used by the algorithms. In addition, a means for inexpensively loading new software into 9-PAC was required.

Hardware Realization

A standard ASR-9 printed-circuit board has about 68 in^2 of usable space on one side. With current surface mount technology (SMT) components, a redesign of the DPRAM portion of 9-PAC uses only 12 in^2 of board space. The remaining 56 in^2 of space is clearly sufficient for several large ICs. In addition, if thin small outline packages (TSOP), which are about 2 mm thick, were used for the memory components, the components could be placed on the back side of 9-PAC to increase board capacity further.

Of the many microprocessors and digital signal processing (DSP) chips available, we chose the Texas Instruments TMS320C40 processor for our design [14]. The main reasons for this choice were that the 'C40 performs floating-point math at a peak of 40 million operations per second, and the chip has built-in high-speed data links for communicating directly with as many as six other 'C40s, thus enabling a multi-processor design.

As work progressed on the 9-PAC design, we realized that there was enough room on the board to fit three processors with at least 9 MB of memory per processor. Figure 11 shows a block diagram of 9-PAC, and the following paragraphs describe the board's major features in reference to the block diagram.

Pairs of communications links—built-in high-speed serial ports capable of peak transfer rates of approximately 26 MB/sec—interconnect the three 'C40 processors. Although each communications link is bidirectional by design, we use the links unidirectionally by pairing them going in opposite directions. Unidirectional operation makes the software simpler and also avoids known problems associated with changing the direction of data transfer in a link.

Each 'C40 has 1 MB of zero-wait-state static random-access memory (SRAM) connected to one of the processor's two memory buses. The 1-MB SRAM holds processing stacks and other data structures that need to be accessed frequently. Each 'C40 also has a bank of dynamic random-access memory (DRAM) chips to hold the program code and data that do not need to be accessed frequently. Two of the 'C40s have 8 MB of three-wait-state DRAM and the third 'C40 (processor 3) has 16 MB of DRAM. Processor 3 requires this extra memory for map storage to run the geocensor-mapping function. To compensate somewhat for the slow DRAM used to store software, the 'C40 has an on-board instruction cache.

All peripheral devices on the board are connected to processor 1, which is often referred to as the "housekeeping" processor. Processor 1 is responsible for booting up the 9-PAC, running diagnostics, and communicating with the ASP and any other devices connected to the peripheral ports. The peripherals and special memories connected to processor 1 are

- The multi-port RAM that provides an interface to the ASP and the HSIB or MIP. The software running in processor 1 can access this memory directly.
- A 64-kB electrically programmable read-only memory (EPROM) that contains start-up diagnostics and the bootstrap code.
- 4 MB of flash memory (a form of electrically erasable non-volatile memory). The 4-MB flash memory can be used to hold the software that is loaded into the three processors, and can also be used to hold a history of modifications/revisions to the board.
- A Personal Computer Memory Card International Association (PCMCIA) card socket. A 20-MB solid state flash-memory disk card containing the software that the bootstrap program in EPROM loads into the three processors is normally plugged into this socket.
- Four high-speed serial ports that are capable of data rates up to 2 MB/sec. Two of the serial ports have RS-232 drivers and receivers; the other two have RS-422 differential drivers and receivers. The high-speed serial ports can be used for a variety of purposes, for example, to provide a data path to a workstation for recording input and output data as the data pass though the system. Other possible uses include expansion to add a satellite clock and a Surveillance Advanced Message Format (SAMF) output for direct communication to the FAA's

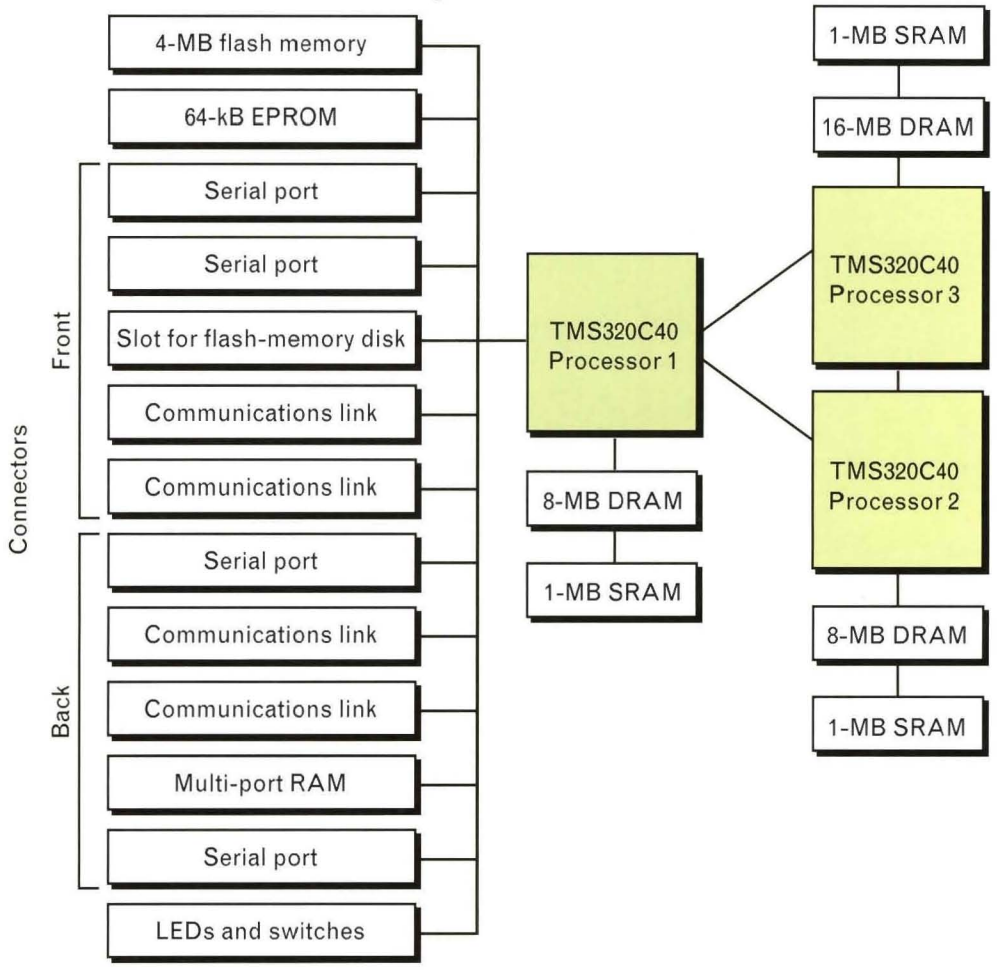

FIGURE 11. Block diagram of 9-PAC. Processor 1, which is often referred to as the "housekeeping" processor, is responsible for booting up the 9-PAC, running diagnostics, and communicating with the ASP and any other devices connected to the peripheral ports.

Advanced Automation System (AAS).

- A bank of eight light-emitting diodes (LED) for conveying status information, and nine switches for configuring board options in software.

The diagnostic interfaces of the 'C40 processors conform to the IEEE 1149.1 Joint Test Action Group (JTAG) [15]. The JTAG standard allows all of the processors to be connected by a few signal lines for diagnostic and debugging functions. A debugger and program loader can be connected to 9-PAC by a small connector at the front edge of the board. Via this JTAG port, the debugger has complete access to all of the 9-PAC's processors, memories, and peripherals.

All of 9-PAC's glue logic is contained in five Lattice Semiconductor ispLSI in-circuit programmable logic devices (PLD) [16]. Three of the ispLSI devices are used as DRAM controllers, one for each 'C40 processor's DRAM; the fourth ispLSI device controls the multi-port RAM interface; and the fifth controls all other peripheral devices. Unlike conventional PLDs, which must be inserted into a programmer to be programmed, the ispLSI devices are programmed in-circuit by a PC via a cable that plugs into a connector on 9-PAC. This feature allows the devices to be soldered in place instead of being socketed, and also makes it very simple to modify the programmable logic when necessary.

9-PAC is a 12-layer printed-circuit board with components—almost entirely SMT devices—on both sides of the board (Figure 12). 9-PAC has been

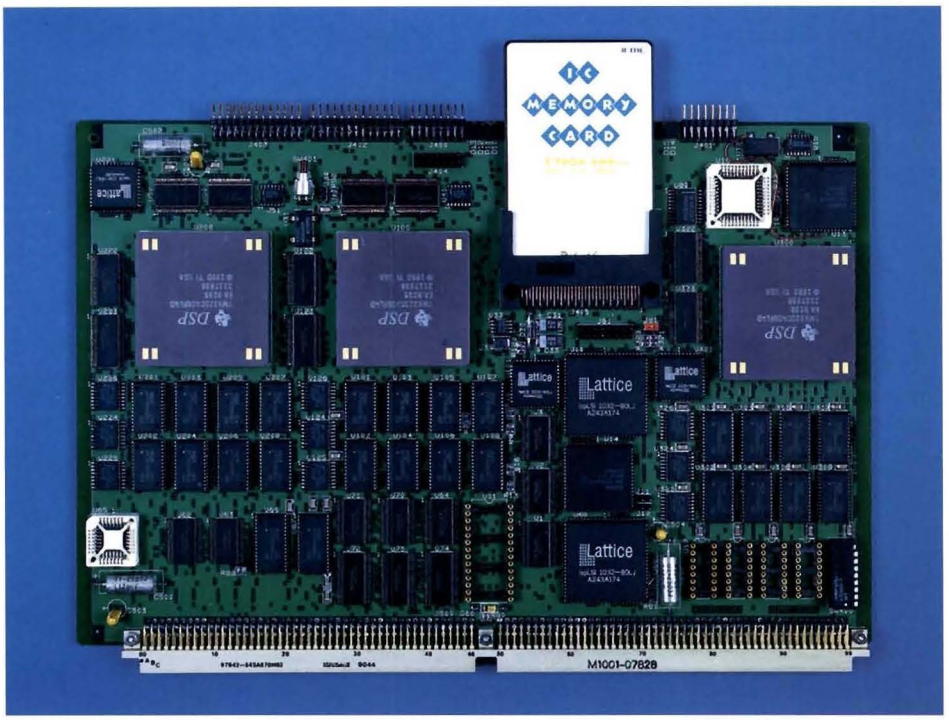

FIGURE 12. Photograph of 9-PAC.

manufactured by outside vendors and by Lincoln Laboratory's printed-circuit fabrication facility. The cost of 9-PAC with all parts installed is less than $10,000, and the cost of just the 9-PAC board without parts is approximately $1000.

When power is applied to the 9-PAC board, processor 1 immediately begins running the program stored in the EPROM. This program contains a diagnostic that tests the functionality of most of the board's components. If a failure occurs, the diagnostic will halt and an error code will flash on the 9-PAC LEDs. When 9-PAC has passed the diagnostic without any failures, the program stored in the EPROM will load software from a PCMCIA flash-memory disk card into all three processors' memories. The flash disk is also used to load and update the various maps described in the section "9-PAC Radar and Beacon Processing Algorithms." If the 9-PAC software has to be upgraded in the field, the flash disk is simply replaced.

System Software and Development Process

Software development began simultaneously on the four main algorithm areas—BTD, C&I, Merge, and Tracker. The initial algorithms were written in ANSI C on Sun Microsystems workstations. A diagnostic tool called X-Windows Radar Analysis Package (XRAP) was used to test the algorithms with data collected at the Los Angeles and Salt Lake City ASR-9 sites. XRAP can record various combinations of the primitive report data that enter the ASP as well as the completed reports that the ASP sends to the MIP. With these data, we could test the new algorithms to see if they indeed corrected the existing problems without introducing new ones.

While the 9-PAC algorithms were under development, engineers at the FAA Technical Center wrote a specification consisting of data formats, communications protocol, and dedicated memory areas for the transfer of data between the ASP and 9-PAC via the multi-port memory. The engineers then modified the code in the ASP and the ASR-9's remote maintenance system (RMS) to operate with 9-PAC. Using the modified code, the ASP looks for 9-PAC at start-up. If a 9-PAC board is found, the ASP configures itself to communicate with the new board. If a 9-PAC board is not found, the ASP operates using its standard software. The ASP also monitors the functioning of 9-PAC and, if a problem is detected, the ASP can decide to shut 9-PAC down and revert to stan-

dard operating mode in which only 9-PAC's multiport RAM feature is used.

With the ASP communications specification in hand, our programmers wrote an ASP simulator that ran entirely within a Sun workstation. The simulator first reads data from XRAP tapes or generates simulated data, then transfers the data to a piece of shared memory that is formatted according to the ASP interface specification. A second program is then executed that reads the data in the shared memory in a manner identical to that in which data in the actual post-processor multi-port memory would be read. This second program can then execute the algorithm software. Most of the interface and algorithm software was developed and tested with this simulation technique before the 9-PAC hardware was ready for use.

9-PAC Operating System

Initial software work on the 9-PAC hardware was undertaken with a multitasking operating system obtained from an outside vendor. The operating system provided necessary services such as task and interrupt management. After several months of 9-PAC code development, however, the vendor of the operating system released a new version of the system in which the names of many of the routines that we were using had been changed, thus causing us to rethink our use of a commercial operating system. Our principal concern was in maintaining the software through a radar lifetime of perhaps twenty years. If the vendor of the operating system went out of business during that period and a problem was discovered in the code, the problem might be nearly impossible to fix.

To eliminate such potential difficulties, we decided to write our own operating system so that we could have complete control over the source code. The new operating system, called PAC-OS, is smaller, simpler, and faster (because of a reduced feature set) than the commercial operating system used earlier. Designed specifically for 9-PAC's needs, PAC-OS provides the following features:

- *Prioritized multitasking*—allows the software to be broken into small tasks that execute independently in order of importance.
- *Signals and semaphores*—provides a means to synchronize the activities of independent tasks, and control the access to shared resources.
- *Queues*—used to pass data between tasks in an orderly manner to avoid the potential conflicts that can occur when two tasks are simultaneously trying to pass data to a third task.
- *Memory management*—allocates memory to different tasks. Because the 9-PAC software can potentially run for years without being restarted, the de-allocation of memory is not allowed, thus avoiding memory fragmentation problems.
- *Input/output services*—used primarily for providing diagnostic outputs during debugging.
- *Flash-memory file services*—used to read and write data to the on-board and PCMCIA flash memories. These memories are organized to mimic the DOS file system used by IBM-compatible personal computers.
- *Intertask communications*—allows tasks to transfer data to and from other tasks. Via hardware communications links, the communications system can transfer data transparently to other processors when necessary. This feature allows tasks to be moved between processors during development to equalize the processor load.

In PAC-OS, the intertask communications software is of key importance because it allows transparent communications to take place between tasks whether they are located on the same or different processors. A driver in PAC-OS connects the intertask communications system to one of 9-PAC's high-speed serial ports, which is in turn connected to a Sun workstation on our development network. A program on the Sun workstation connects the intertask communications system to standard UNIX sockets. This setup allows multiple workstations to access the data streams that are available from 9-PAC.

Software and Hardware Development: Phases 1 and 2

Several months into the 9-PAC development effort, we realized that the time needed to develop the full software set was going to be considerably longer than the time required to develop the hardware. For this reason, the software task was split into two phases: phase 1 for the BTD and Merge functions, and phase 2 for the C&I and Tracker functions. The plan was to

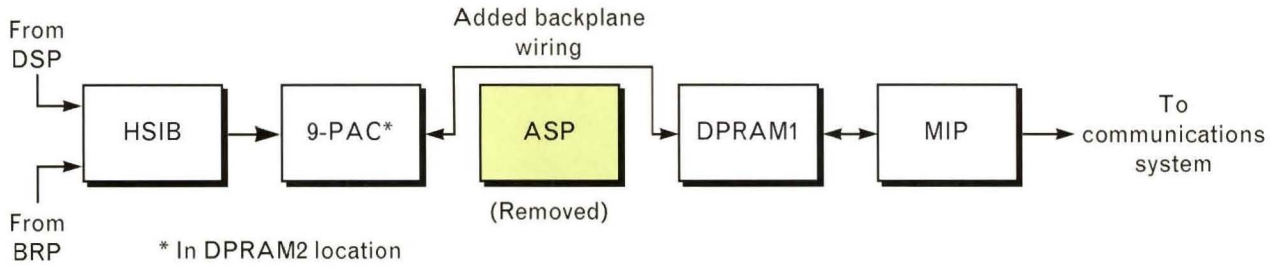

FIGURE 13. Configuration of the post-processor for phase 2 of the software and hardware development. Note that the ASP has been bypassed.

complete phase 1 as quickly as possible to get the system up and running, after which phase 2 would be completed. We selected BTD and Merge for phase 1 because those two functions provide the greatest benefit to controllers and are easier to test.

One of the goals of the 9-PAC program has been to reduce or eliminate the need to program the ASP. Although the replacement of all the algorithm modules in the ASP was a large step toward this goal, ASP software was still required to transfer output data from 9-PAC to MIP (see Figure 10). To meet the goal of eliminating all ASP software, we had to eliminate the ASP altogether. Analysis of the connections on the ASP backplane showed that we could remove the ASP and connect DPRAM1 with 9-PAC by jumping the backplane pins used by the ASP interface (Figure 13). With the removal of the ASP, 9-PAC could then communicate with MIP via DPRAM1. This new configuration would require the installation of approximately thirty jumpers on the wire-wrapped backplane in the post-processor, the replacement of a single IC on 9-PAC, and the reprogramming of the ispLSI glue logic on the board.

The modified version of the original hardware shown in Figure 13 is referred to as the phase 2 hardware, which is used with the phase 2 software to replace the ASP completely. The phase 2 software will run only on a phase 2 9-PAC, and this phase 2 configuration is the preferred version for final deployment.

Test and Development Tools

To test 9-PAC, we needed a means for recording different information such as the data going into 9-PAC, the internal data streams passing between the algorithm modules, the data completely internal to the processing modules (such as tracker updates), and the output data being sent to MIP. Also, the developers required a display for testing the algorithm code. Thus a task socket was created in 9-PAC into which all of the algorithms could write the necessary diagnostic and display data. The task socket was designed to pass data through the intertask communications system and high-speed serial port, and out to a Sun workstation that would then make the information available via Ethernet and a standard UNIX socket.

For the Sun workstation, we have developed three programs—Pacmon, Pacrec, and Pacview—for monitoring, recording, and displaying the 9-PAC data. The Pacmon program connects to the UNIX data stream and displays load statistics with bar graphs for each of the 9-PAC processors. Pacmon also monitors memory usage and the loading on the intertask communication channels. The Pacrec program runs as a UNIX X-Windows application and connects to the data stream via the UNIX socket. From a menu of twelve different types of data, a user can select any or all of the data types for recording to disk or tape. The Pacview program accesses the same data stream and generates a color display on a workstation (Figure 14). With Pacview, a user can zoom in to an area of interest on the display. Various types of targets can be shown with different colors and symbols to distinguish them. Statistics and alarms that 9-PAC generates can also be displayed.

For the IBM PC, we have built a demonstration tool for accessing the display and diagnostic data stream. To handle the data coming out of 9-PAC, the IBM PC required an inexpensive commercially available high-speed serial-interface board. For the IBM PC with this high-speed board, we have successfully

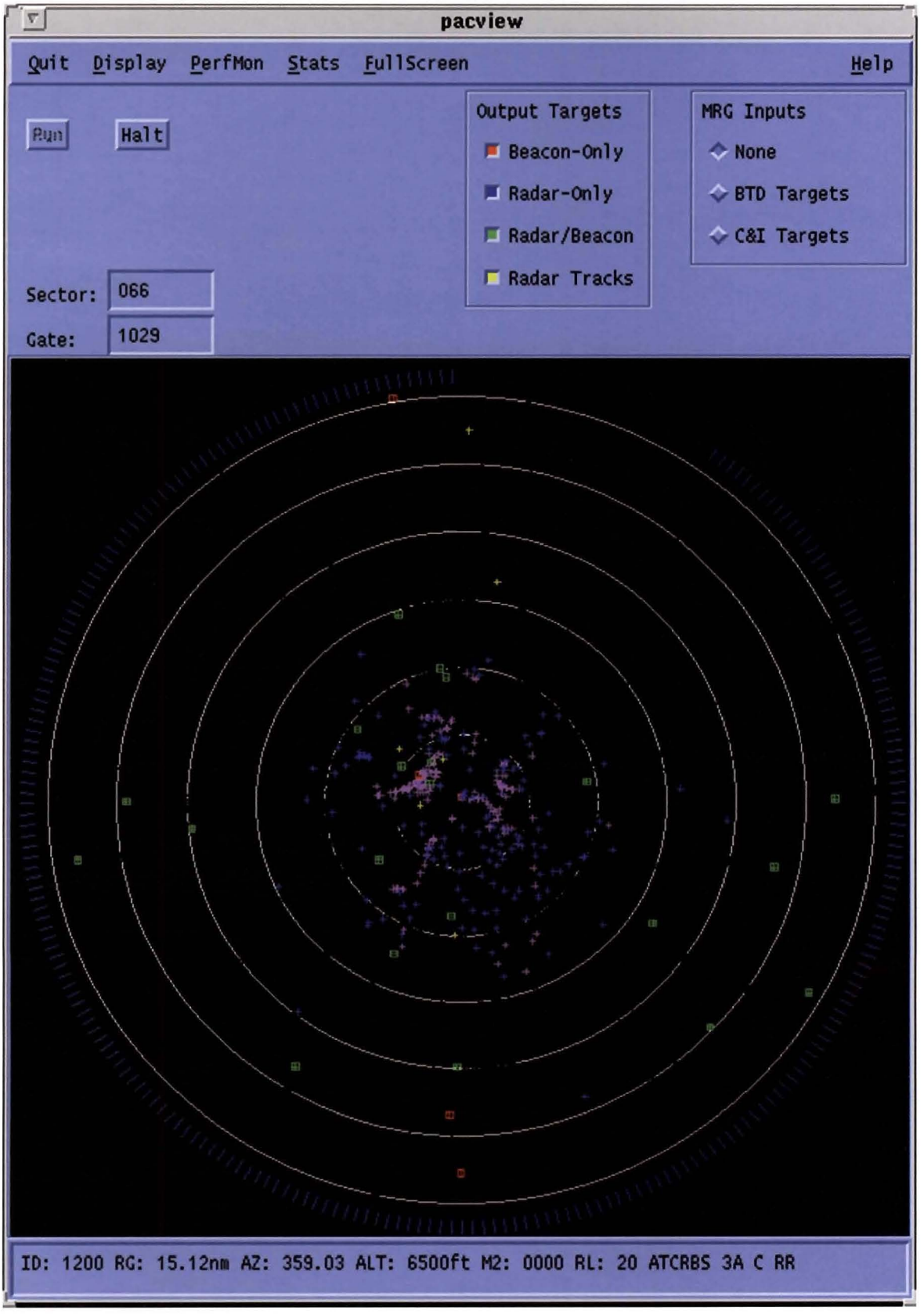

FIGURE 14. Typical display from the Pacview program on a workstation screen.

ported a version of the Pacmon program; we could also port Pacrec and Pacview by using the I/O code developed for this demonstration tool.

We have also developed three tools to test the 9-PAC hardware. The first tool, called the 9-PAC Tester, can exercise all of the signals present on the 9-PAC backplane connector with variable (user selected) timing. The primary use of this tool is for checking the 9-PAC backplane interface to ensure that it can operate over the wide range of system clock skews that was noticed during documentation of the DPRAM operation. The second tool, called the ASP Simulator, provides a VME interface to the multiport RAM on 9-PAC. When a VME chassis containing a Sun workstation is plugged into this interface, the workstation can use the ASP Simulator to read and write data directly from and onto 9-PAC's multiport RAM, thus simulating the ASP's function. With the Pacview and Pacrec programs, the output data from 9-PAC can be displayed or recorded. The ASP Simulator is capable of operating 9-PAC at real-time rates. The third tool, called the 9-PAC Simulator/Tester Version 2, is a combination of the first two tools with an added feature—it can simulate the MIP interface used by the phase 2 9-PAC hardware. An on-board TMS320C40 controls the Simulator/Tester's functions.

Field Testing

After the first functional 9-PAC was tested at Lincoln Laboratory with the above tester and simulator, the board was installed in the ASR-9 at Lincoln Laboratory's Terminal Radar Development Facility (TRDF) in Albuquerque, New Mexico. Integration of 9-PAC with the ASR-9 ASP required on-site software and hardware debugging sessions involving engineers from both Lincoln Laboratory and the FAA Technical Center. The TRDF site was used to test 9-PAC with real-time radar and beacon data over periods of several weeks. The ASR-9 at the TRDF site currently has a phase 1 9-PAC running in one channel and a phase 2 9-PAC running in the other channel. A 56-kb/sec digital data link connects the TRDF site with Lincoln Laboratory in Lexington, Massachusetts, allowing personnel at the laboratory to create and download new software into the 9-PAC in a matter of minutes. Also, the link allows remote operation of the Pacmon and Pacrec analysis tools. The 56-kb/sec link does not, however, have adequate bandwidth to provide a real-time display in Lexington.

At the TRDF site, we subjected the phase 1 9-PAC to a number of tests using a beacon test-pattern generator set to provide the maximum target loading for the ASR-9. In other tests, we stressed the phase 1 9-PAC further by defeating the sidelobe-suppression feature of the interrogator, thus causing the BRP to receive a large number of additional replies. As expected, we discovered software problems during this testing. Most of the problems were fairly straightforward coding errors that could be corrected through debugging and code modification over the communications link to Lexington. Occasionally, however, the real-world environment at the TRDF site would excite anomalous behavior in the algorithms. Such behavior required the developers to reassess and change some of the methods used in the algorithms.

The FAA Technical Center has also tested 9-PAC in a test version of the ASR-9. One of the initial runs in that system produced a dataset from which we first observed instances of multiple aircraft that had been erroneously assigned the same discrete identifier code. As a result of these observations, we modified the BTD and Merge algorithms to prevent 9-PAC from deleting such aircraft.

In other experiments, Westinghouse Electric Corp. has been testing 9-PACs in an ASR-9 that is connected to the production test equipment that was used to test the original ASR-9. Errors in report formatting were discovered in the initial testing at Westinghouse. These errors were reported to the staff at Lincoln Laboratory in Lexington, where the code was then corrected. The corrected code was then sent to Westinghouse over the Internet. Subsequent capacity testing at Westinghouse revealed that the BTD algorithm could not keep up with the target load under the worst-case conditions set forth in the ASR-9 specification. This discovery led to a profiling analysis of the BTD code and resulted in an overhaul of the database organization of the BTD internal tracker. Again the Lincoln Laboratory staff modified and updated the code over the Internet. The new code runs fast enough to pass the capacity tests with some addition-

al margin to allow for increasing the code's functionality in the future.

At this time, a noncommissioned ASR-9 at the Philadelphia Airport is being prepared to perform real-time evaluation of the phase 1 9-PAC. After the conclusion of these tests, 9-PAC will be installed on a trial basis at several of the airports that have been experiencing the problems that prompted the board's development.

Conclusion

Although the Moving Target Detector and its production in the ASR-9 represents a significant advance in the technology of airport surveillance radars, deployment of more than sixty such systems in a variety of operating environments exposed problems that were not apparent in the development program in the 1970s. Beacon multipath reports and radar echoes from birds, automobiles, and weather caused errors in the system's surveillance. New processing algorithms, incorporated in 9-PAC, were developed that took advantage of adaptive methods to mitigate these problems. Because these algorithms were realized in a high-level language, they should be portable to application in other radars.

With the current set of software, 9-PAC's memory and processing power are being utilized at roughly fifty percent of full capability. By these measures, 9-PAC has room for considerable future expansion.

Acknowledgments

Many talented people have worked to bring 9-PAC to its current state of development. At Lincoln Laboratory, Jeff Gertz developed the BTD algorithms; Gabe Elkin wrote the BTD code and designed and wrote the Pacrec and Pacview tools; Oliver Newell wrote the C&I code, the PAC-OS, and many other tools used in development; Jenifer Evans designed and wrote the Merge and Tracker algorithms; Walter Heath wrote the flash-memory file system and the support software for the IBM PC; Zahidul Hassan provided support for several data-collection tools; Ed Hall designed the 9-PAC hardware and the Simulator/Tester Version 2, as well as the start-up diagnostics; Greg Rocco designed the first 9-PAC tester; and Don Malpass facilitated the fabrication of all of the hardware, including more than thirty 9-PAC engineering models. In addition, John Anderson of the University of Wisconsin developed the automated geocensoring and second adaptive-threshold testing; Marc Edwards at Westinghouse Electric Corp. and Ed Pauley from the FAA Technical Center modified the ASR-9's ASP and RMS software to work with 9-PAC; and Bill Goodchild from the FAA headquarters spent many days at the Albuquerque site torture-testing the system to ensure its veracity.

REFERENCES

1. M.L. Stone and J.R. Anderson, "Advances in Primary-Radar Technology," *Linc. Lab. J.* **2**, 363 (1989).
2. "Airport Surveillance Radar (ASR-9)," Dept. of Transportation/FAA Specification (1 Oct. 1986), FAA-E-2704B.
3. D. Karp and J.R. Anderson, "Moving Target Detector (Mod II) Summary Report," *Project Report ATC-95,* MIT Lincoln Laboratory (3 Nov. 1981), DTIC #AD-A114709.
4. D.C. Puzzo, S.W. Troxel, M.A. Meister, M.E. Weber, and J.V. Pieronek, "ASR-9 Weather Channel Test Report," *Project Report ATC-165,* MIT Lincoln Laboratory (1989), FAA-PM-86-38, DTIC #AD-A211749.
5. J.L. Gertz, "The ATCRBS Mode of DABS," *Project Report ATC-65,* MIT Lincoln Laboratory (8 Jan. 1977), FAA-RD-76-39, DTIC #AD-A038543.
6. M.E. Weber and T.A. Noyes, "Low Altitude Wind Shear Detection with Airport Surveillance Radars: Evaluation of 1987 Field Measurements," *Project Report ATC-159,* MIT Lincoln Laboratory (31 Aug. 1988), FAA/PS-88/10.
7. *ASR-9 System, Field Maintenance,* Sections 1–2, 4–8, and 10–11 (Westinghouse Electric Corp., Baltimore, MD, 1 Dec. 1990), TI 6310.24.
8. M.C. Stevens, *Secondary Surveillance Radar* (Artech House, Norwood, MA, 1988).
9. A. Ashley and J.S. Perry, "Beacons," ch. 38 in *Radar Handbook,* ed. M.I. Skolnik (McGraw-Hill, New York, 1970).
10. *Moving Target Indicator and Detector System,* FAA Academy Training Manual 40392-4/2 (FAA, Oklahoma City, OK, July 1989).
11. O.J. Newell and J.R. Anderson, "Description of Radar Correlation and Interpolation Algorithm for the ASR-9 Processor Augmentation Card (9-PAC)," *Project Report ATC-236,* MIT Lincoln Laboratory (to be published).
12. J.L. Gertz and G.R. Elkin, "Documentation of 9-PAC Beacon Target Detector Processing Function," *Project Report ATC-220,* MIT Lincoln Laboratory (26 July 1994).
13. F. Burgeois and J.C. Lassalle, "An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices," *Commun. ACM* **14**, 802 (1971).
14. *TMS320C4x User's Guide* (Texas Instruments, Inc., Houston, TX, 1 June 1992), SPRU063.
15. "The Test Access Port and Boundary Scan Architecture," *IEEE Std. 1149.1-1990* (IEEE, New York, 1990).
16. *pLSI and ispLSI Data Book and Handbook* (Lattice Semiconductor Corp., Hillsboro, OR, 1992), B0006.

JAMES V. PIERONEK
works in the Weather Sensing Group, where he is the project leader for the Airport Surveillance Radar-9 (ASR-9) Processor Augmentation Card (9-PAC). His other work includes the development of the signal processor hardware for the ASR-9 Wind Shear Processor. Jim received a B.S. degree in electrical engineering from Michigan State University.