A SIMD Solution to Guarantee Real-Time Requirements for Air Traffic on Associative Processor Architectures

Mike Yuan
Department of Computer Science
Kent State University
Kent, OH
Email: myuan@cs.kent.edu

Johnnie W. Baker
Department of Computer Science
Kent State University
Kent, OH
Email: jbaker@cs.kent.edu

Will Meilander(retired)
Department of Computer Science, Sys. Anal.
Kent State University, Goodyear Aerospace
Kent, OH Akron, OH
Email: willcm@charter.net

Frank Drews
School of Electrical Engineering and Computer Science
Ohio University
Athens, OH
Email: drews@ohio.edu

***Abstract*—This paper proposes a SIMD solution to air traffic control (ATC) using an enhanced SIMD machine model called an Associative Processor (AP). Automatic Air Traffic Control (ATC) is a goal of FAA's Next-Gen Project and is rapidly becoming more important. Previous ATC systems designed for MIMD computers have a great deal of difficulty meeting the predictability requirements for ATC, which are critical for meeting the strict certification standards required for safety critical software components. Instead, we implement this dynamic database problem on an enhanced SIMD system called an Associative Processor (AP), where interactions are much simpler and more efficiently controlled. Our proposed AP solution can support fully redundant, fail-safe, accurate and meaningful predictions of worst case execution times of 8 real-time ATC tasks and will guarantee all their deadlines are met. Also, the software will be much simpler and smaller in size than the current corresponding ATC software. An important consequence of these features is that the V&V (Validation and Verification) process will be considerably simpler than for current ATC software. Additionally, the associative processor is enhanced SIMD hardware and is considerably cheaper and simpler than the MIMD hardware currently used to support ATC. The ClearSpeed CSX600 accelerator is used to emulate the AP model, and a preliminary implementation of the proposed method has been developed. Experimental results comparing MIMD and CSX600 approaches are presented, and show that our solution can guarantee that all real-time ATC tasks will finish within their hard deadlines for all aircraft being tracked. The comparative performance of CSX600 demonstrates that AP (and SIMDs in general) has better scalability, efficiency, and predictability than that of MIMD.**

***Keywords*-Air Traffic Control (ATC); SIMD; MIMD; Associative Processor (AP); Conflict detection and resolution (CD&R); ClearSpeed CSX600; Validation and Verification(V&V);**

I. INTRODUCTION

The air traffic control (ATC) problem is a real-time dynamic database problem. It continuously monitors, examines, and manages space conditions for thousands of flights by processing large volumes of data that is dynamically changing due to reports by sensors, pilots, and controllers. The system gives the best estimate of position, speed and heading of every aircraft in the environment at all times. The ATC software consists of multiple real-time tasks that must be not only accurate for safety but also completed in time to meet their individual deadlines. The FAA has spent a great deal of effort on finding a predictable and reliable system to achieve *4-D trajectory* which would allow pilots to choose the best path to minimize fuel consumption and time delay rather than following pre-selected flight corridors [1], [2], [3]. In the past, ATC has been implemented on multiprocessor systems (MIMD) with records of various aircraft distributed over the distributed memory of this system. The distributed nature of this database adds considerably to the difficulty and complexity of handling ATC. Previous attempts to build a new system have repeatedly failed after about ten years work by a large team [4], [5].

The most critical issue of *free flight* is conflict detection and resolution (CD&R). The performance of all CD&R algorithms available depends on aircraft state estimation according to the comprehensive survey of Kuchar and Yang [6]. The Kalman filter [7], [8] is the central algorithm for the majority of all modern tracking systems, known as $\alpha - \beta$, $\alpha - \beta - \gamma$ filters. The major problem with the single Kalman filter is that it does not predict well when the aircraft makes an unanticipated change of flight mode such as making a maneuver, accelerating etc [9], [10]. Many adaptive state estimation algorithms have been proposed [11], [12], [13], [14]. The Interacting Multiple Model (IMM) algorithm [15], [16] runs two or more Kalman filters that are matched to different modes of the system in parallel. It uses a weighted sum of the estimates from the bank of Kalman filters to compute the state estimate. IMM and its variants have been

applied to single and multiple aircraft tracking problems in [11]. However, it becomes inaccurate for tracking multiple aircraft as the number of aircraft increases. Current MIMD implementation of this algorithm is computationally very intensive. Hwang et. al. [17], [18] propose that the mode likelihood function can be used to improve the estimation results of IMM algorithm. The likelihood function uses the mean of the residual produced by each Kalman filter. A heuristic algorithm that evaluates correlation error values has been shown to provide better results than the Kalman filter [10].

A comprehensive survey of the CD&R algorithms is presented in Kuchar and Yang [6]. In [19], Krozel et. al. propose one centralized strategy that is controller-oriented and two decentralized strategies that are user-oriented. In the centralized approach, a central agent analyzes the trajectories of the aircraft and determines resolutions. In the two decentralized strategies, each aircraft resolves its own conflicts as they are detected. In [3], Yang et. al. propose a conflict alerting logic based on sensor and trajectory uncertainties, with conflict probability based on Monte Carlo simulation. Chiang et. al. [20] propose CD&R algorithms from the perspective of computational geometry. Paielli et. al. [21] and Prandini et. al. [22] propose analytic algorithms for computing probability of conflict. Many of the algorithms consider only two aircraft. For example, Krozel et. al. [19] show that neither their centralized nor decentralized CD&R algorithms can guarantee safety for multiple aircraft when the number of aircraft is growing. Furthermore, many algorithms propose optimization schemes that are not guaranteed to be completed within real-time deadlines. Due to the increasing number of FAA problems, FAA is inviting proposals for new and efficient CD&R [23].

On the other hand, several papers [24], [4], [5], [25], [26] have investigated the use of an Associative Processor (AP) to manage ATC computation. The assumed AP in these papers is an enhanced SIMD model which can execute several basic global operations such as MAX and MIN, AND and OR, Associative search, Any -Responders, Pick-One, and broadcast etc in constant time, as explained in detail in [27], [28]. As shown in [28], an AP can be built easily. In fact, the STARAN and ASPRO computers built by Goodyear Aerospace were associative computers. The STARAN was designed by Kenneth Batcher and for ATC applications and the ASPRO was a second generation of STARAN that was designed for the Navy to use for airborne early warning and command and control. In this paper, the assumed maximum number of aircraft being tracked by one air traffic control center is 4000 IFR (instrument flight rules) aircraft and 10000 VFR (visual flight rules) aircraft, for a total of 14000 aircraft [24], [4]. Currently, this number of flights would allow the air traffic control center to handle not only the air traffic in their sector but also to provide backup tracking for the aircraft in all adjacent sectors. The paper [25] predicted performance on the earlier ClearSpeed CS301 chip. Our papers [26], [29] implemented report correlation and tracking and CD&R tasks based on ClearSpeed CSX600 and compared its performance with a state-of-the-art MIMD system using a total of 8 system cores. The CSX600 architecture and its emulation of an AP using the CSX600 are also described in papers [26], [29].

However, our previous work only implemented two ATC tasks, report correlation and tracking, conflict detection and resolution, and cannot prove whether our ATC prototype can work well when the number of tasks and aircraft increase. In this paper, we have implemented 8 real-time ATC tasks such as report correlation and tracking, cockpit display, controller display update, sporadic requests, automatic voice advisory, terrain avoidance, conflict detection and resolution (CD&R) and final approach (runway optimization) on the emulating tool ClearSpeed CSX600. The experiment results show that all the tasks can be guaranteed to be finished within deadlines for up to 672 aircraft, and there is still some unused time in the total 8 second major cycle period. Because CSX600 emulates the properties of AP and its speedup and efficiency are almost optimal, we conclude that AP can guarantee the real-time requirements of all ATC tasks.

This paper is organized as follows. The overall system design and static scheduling is illustrated in Section II. Section III presents our approaches for 8 key ATC tasks. Section IV presents experimental results. Conclusions and future work are presented in Section V.

II. ATC System Design

A. ATC Data Flow

The overall system design is shown in Figure 1, which is a modification of a figure in [30]. The executive box controls the single instruction stream of AP using static scheduling. All control paths are from ATC in the executive box to all of the tasks, e.g., report correlation and tracking etc. Controller input simulates sporadic requests, e.g., weather change, controller input, etc. Radar reports data are simulated by data from data lines to two modems and transferred from host to CSX600 PEs. Flights are simulated from flight plans in PEs. The radar reports and tracks are used for report correlation and tracking task. The outputs of tracking task are used for cockpit display, controller display update, terrain avoidance, conflict detection and resolution (CD&R) and final optimization. The results of terrain avoidance, CD&R and final approach are used for cockpit display and controller display update. The results of terrain avoidance and CD&R are used for automatic voice advisory that transfers results to automatic voice advisory driver in host and produces voice output. The resolution advisories of CD&R task are sent to controllers.

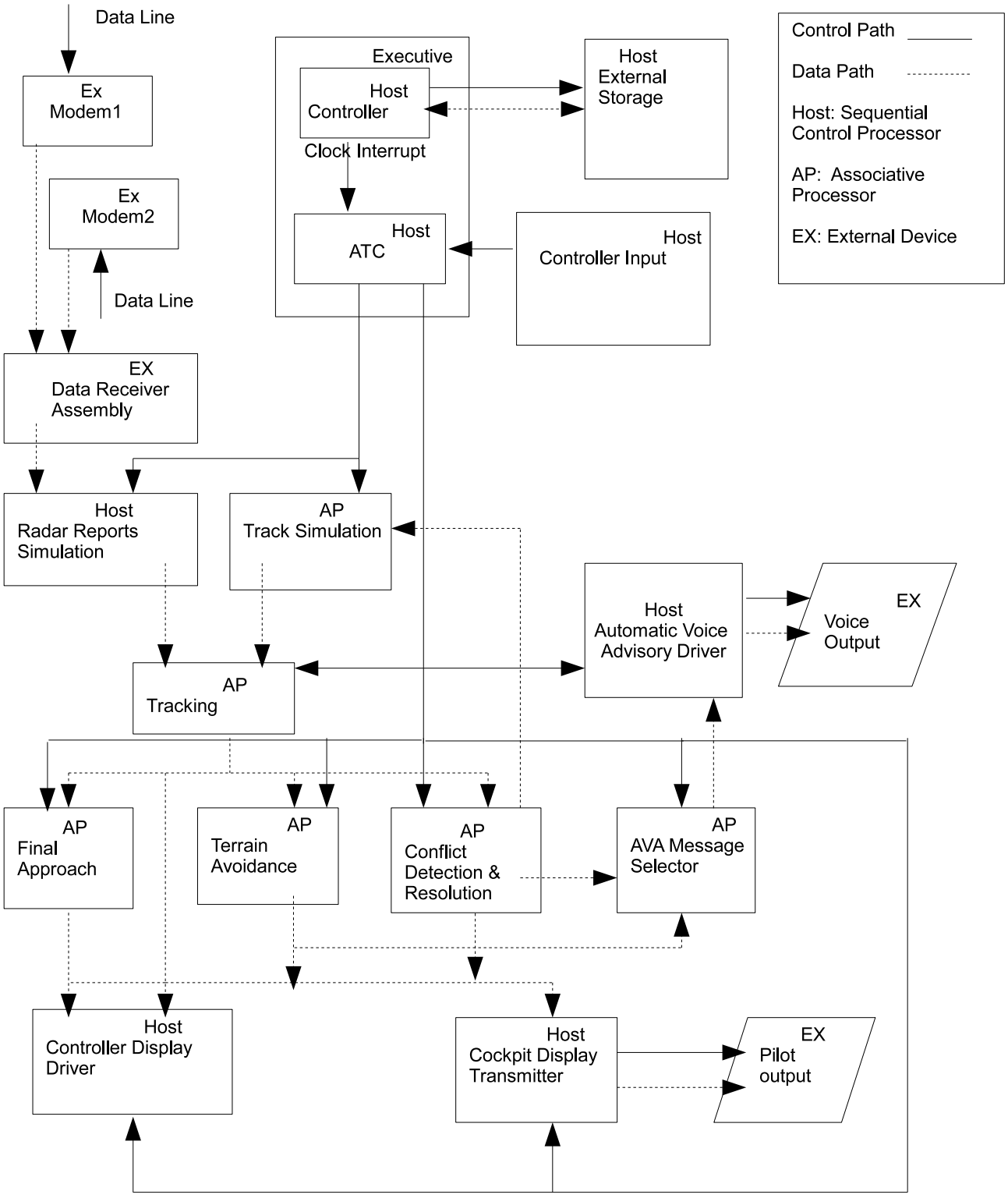

Figure 1. Overall ATC System Design.

B. Static Scheduling

The report correlation and tracking (1) is executed every 0.5 second, cockpit display (2), controller display update (3) and sporadic requests (4) are executed every one second, automatic voice advisory (5) is executed once every 4 seconds, terrain avoidance (6), conflict detection and resolution (7), and final approach (8) are executed every 8 seconds.

An 8 second period is split into 16 one-half second periods, which is called slots. Tasks 1, 2 and 3 are executed in the first, third, fifth, seventh, ninth, $11th$, $13th$, $15th$ *slot*. Tasks 1 and 4 are executed in the second, sixth and tenth *slot*. Tasks 1, 4 and 5 are executed in the fourth and $12th$ *slot*. Tasks 1, 4 and 6 are executed in the eighth *slot*. Tasks 1, 4 and 7 are executed in the $14th$ *slot*. Tasks 1, 4 and 8 are executed in the $16th$ *slot*.

III. An AP Solution for ATC Tasks Implemented on CSX600

This section describes the algorithms for eight real-time tasks, namely report correlation and tracking, cockpit display, controller display update, sporadic requests, automatic voice advisory, terrain avoidance, conflict detection and resolution (CD&R) and final approach (runway optimization). The solutions for these ATC tasks are implemented on the CSX600 using an emulation of an associative processor, but they could be implemented more easily and efficiently directly on an AP. Additionally, a large AP would be able to support a much larger number of tasks and aircraft than the CSX600. Since the CSX600 is a SIMD, it also follows that ATC could be managed by a large SIMD. However, additional constant time operations added to an AP makes it both simpler, more efficient, and more effective to use an AP rather than a SIMD to support an ATC system.

In some of these algorithms, an X-Y coordinate system is assumed. The assumed sector for controlled aircraft is a two dimensional airspace of 1024 by 1024 nautical miles. A central point in this sector is selected as the origin for this X-Y coordinate system and the X axis points towards north. The units used in this coordinate system is nautical miles.

A. Report Correlation and Tracking

The report correlation and tracking algorithm is shown in Algorithm 1. The input data are radar reports that are simulated in host and flight records in PEs. If total time consumed is considered, this is easily the ATC task that consumes the most time, as it is performed much more frequently than the other tasks.

Algorithm 1 Algorithm for Aircraft Tracking

1: All radar reports are transferred from host to mono memory. Next, they are transferred from mono to PE memories, with each PE receiving an equal share of the reports.
2: Boxes of sides of length 1 nautical mile (nm) and altitude 1000 feet are created around each radar report and each track in each PE to accommodate report and track uncertainties.
3: Check the intersection of each report box with every track box in each PE. If there is an intersection, the radar report and the track are correlated.
4: The radar reports in each PE are transferred to the next PE using the $swazzle$ (i.e., ring) network. After 96 iterations, all reports have been compared with all tracks.
5: Double the box sizes of tracks that have not correlated with any reports to increase their probability to intersect a report box and repeat the steps 3 and 4 above for unmatched reports.
6: Triple the original box sizes of tracks that have not correlated and repeat the steps 3 and 4 above for unmatched reports.
7: If two tracks correlate to the same radar report, this report is discarded because of ambiguity.

B. Cockpit Display

First the associative operation PickOne is used to select one aircraft. Next, the broadcast operation is used to broadcast the x, y and altitude coordinates of the plane picked in the previous step. For each of its aircraft records, each processor computes the x-distance, y-distance and altitude distance between the location of its selected aircraft and the location of the aircraft selected in the first step. Then select the aircraft that are approaching this aircraft and within conflict distance within the next 2 minutes. This is done using the conflict detection algorithm to find the aircraft that will be within 2 nm of the aircraft horizontally and within 1000 feet in altitude in 2 minutes. Next, transfer these selected aircraft's identity, x, y positions, altitude, velocity, heading and conflict information etc to the server of CSX600, which plays the role here of the cockpit display. Finally, use the conflict resolution algorithm in section III-F to compute the conflict avoidance advisory information and then transfers this information to the server.

C. Controller Display Update

First, transfer the updated flight identity, positions, altitude, speed, and heading, etc from PEs to the ClearSpeed server, which plays the role of the controller display in this simulation. Next, if there are any advisories for conflict avoidance maneuver, transfer this information to the server.

D. Automatic Voice Advisory

Automatic Voice Advisory (AVA) automatically advises an uncontrolled flight (VFR) of near term conditions of other aircraft and terrain by voice. This task is simulated by printing advisories of conflict detection and resolution, and terrain avoidance tasks etc. For example, if there is an aircraft that is approaching the aircraft called, the message

might be "aircraft at 4 miles, $4,500$ feet, in 1 minute"; if the aircraft called is heading for a terrain, the message might be "terrain, 4 miles, $3,100$ feet ahead".

E. Sporadic Requests

Sporadic requests include information requests or changes in data. For example, aircraft have to avoid an area that has bad weather, aircraft makes maneuver to avoid bad weather, or controllers make a request for runway usage, etc. This task is executed once every second. Although the requests are not processed immediately, they are processed very quickly. We simulate this task as follows. First, use associative operation PickOne to select one aircraft. Next, change its heading to avoid bad weather, e.g., turn right one degree.

F. Conflict Detection and Resolution(CD&R)

1) Conflict Detection: This paper considers a conflict to occur when two aircraft are predicted to be within a distance of 3 nautical miles in x and y and within 1000 feet in altitude. We want to determine the possibility of a future conflict between any pairs of aircraft within a twenty minute "look ahead" period (i.e., 1200 seconds). The conflict detection algorithm is shown in Algorithm 2.

Algorithm 2 Algorithm for Conflict Detection

1: Collision records are designed in each PE for copies of tracks records and conflict detection. Copy all track records to collision records, and for each collision and track record in each PE, check whether their flight *IDs* are different and their altitudes are within 1000 feet.
2: Project their positions 20 minutes into the future and add 1.5 nm to each x and y edge of the future position to provide a 3.0 nm minimal miss distance, as shown in Figure 2.
3: Calculate the min_x, max_x, min_y and max_y to obtain the minimum and maximum intersection times in x and y dimensions, as illustrated in equations 1, 2, 3 and 4.
4: Find the largest minimum time ($time\_min$) and smallest maximum time ($time\_max$) across the two dimensions using equations 5 and 6.
5: If $time\_min$ is less than $time\_max$, there is a potential conflict between the aircraft whose ID is $collision.ID$ and another aircraft whose ID is $track.ID$.
6: If $time\_min$ is less than $collision.time\_till$, $collision.time\_till$ is updated to $time\_min$.
7: All *collision* records in each PE are passed to next PE by *swazzle* function and steps 1 to 6 are repeated.
8: After 96 iterations, all *collisions* have been compared with all *tracks*. The $time\_till$ of each *collision* is its soonest collision time with another *track*.

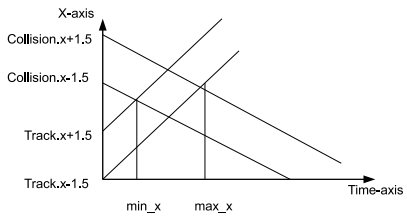

Figure 2. Conflict detection

Formulas 1 to 6 are used in Algorithm 2.

$$min\_x = \frac{|collision.X_c - track.X_t| - 3}{|collision.V_{xc} - track.V_{xt}|} \quad (1)$$

$$max\_x = \frac{|collision.X_c - track.X_t| + 3}{|collision.V_{xc} - track.V_{xt}|} \quad (2)$$

$$min\_y = \frac{|collision.Y_c - track.Y_t)| - 3}{|collision.V_{yc} - track.V_{yt}|} \quad (3)$$

$$max\_y = \frac{|collision.Y_c - track.Y_t)| + 3}{|collision.V_{yc} - track.V_{yt}|} \quad (4)$$

$$time\_min = max\{min\_x, min\_y\} \quad (5)$$

$$time\_max = min\{max\_x, max\_y\} \quad (6)$$

2) Conflict Resolution: The algorithm for conflict resolution is described in the Algorithm 3. The input data are the *tracks* and *collision* records in PEs. Each PE can have 1 to 17 *tracks* and *collision* records.

G. Terrain Avoidance

The shapes that will be used to implement terrain avoidance will consist of a box or a sequence of boxes that contain the terrain feature, e.g., a TV tower is a 1.0 by 1.0 nm box with a height equal to $3,100$ feet. All terrains and tracks are entered in each PE. The terrain avoidance algorithm is shown in Algorithm 4 and is similar to conflict detection algorithm 2.

H. Final Approach (Runways Optimization)

The final approach task is to optimize runway usage. Each flight has a flight plan that specifies its departure terminal and planned departure time, its destination terminal and planned arrival time. The runways that occur in the region being managed by an ATC system could be distributed among the processors. Then each processor would manage the information for the runways assigned to it. Here, we assume that there are 96 runways in the sector being managed by this ATC system and assign one runway to each processor. Third, each runway collects departure and arrival time on it and sorts the time. Fourth, the flights will

Algorithm 3 Algorithm for Conflict Resolution

1: Find the minimum $time\_till$ of all *collisions* records in each PE, i.e., find which aircraft will collide soonest. This is the best or trial aircraft that will make the heading change.
2: Calculate the trial aircraft's *velocity* and *angle* using its velocity in x and y dimension in the mono memory.
3: Each PE will have a trajectory where the trial aircraft makes a different heading change from left to right 3 degrees to evaluate numerous different possible paths for the trial aircraft in parallel.
4: Use the conflict detection algorithm to find the furthest time when the trajectory collides with an another aircraft.
5: Find the maximum of the collision times of all trajectories. Its corresponding trajectory is the best heading change that the trial aircraft will make.
6: The track whose ID is the best aircraft's ID will change its x and y velocity to the best scenario trajectory's velocity.
7: Display the resolution advisory and change the flight plan in the host.

Algorithm 4 Algorithm for Terrain Avoidance

1: For each terrain and track in each PE, check whether the track's height is lower than the terrain's, if yes, go on to next step.
2: Project the track's position to 2 minutes into the future and add 1.5 nm to each x and y edge of the future positions in order to provide a 3.0 nm minimal miss distance. The terrains considered here are 1.0 by 1.0 nm boxes that contain towers.
3: Calculate the minimum and maximum intersection times in both x and y dimensions.
4: Record $time\_min$ as the larger of the two minimum intersection times in both x and y dimensions in step 3. Likewise, record $time\_max$ as the smaller of the two maximum intersection times in both x and y dimensions in step 3.
5: If $time\_min < time\_max$, there is a potential conflict between the track and the terrain.
6: All track records in each PE are passed to next PE by $swazzle$ function and steps 1 to 5 are repeated.
7: After 96 iterations, all track records have been compared with all terrain records for terrain avoidance.

increase or decrease their speed to optimize runway usage and also optimize fuel cost. The last step is currently done by controllers manually.

IV. Experimental Results

This section describes the results of a set of preliminary experiments that were conducted to achieve two different goals. First, we will show that our prototype can meet the deadlines for the hard real-time ATC tasks. Second, we will show that our prototype has a large degree of predictability.

A. Experimental Setup

We are creating a prototype solution since our implementations cannot manage the number of aircraft that occur in an ATC sector in a real-world situation. However, as explained in the Introduction, larger APs that can manage the real-world situation have been built in the past and can easily be built currently. In order to have information about flights that we can control, we simulate the real-world situation by generating aircraft flights in a two dimensional airspace of 1024 by 1024 nautical miles. The initial positions and velocities of the aircraft are generated randomly and trajectories of aircraft consist of a constant velocity mode and a coordinated turn mode. Some radar noise is randomly generated. Since we control this process, we can generate different numbers of aircraft to test algorithms and test the limits on the number of aircraft that can be processed fast enough to meet deadlines. Unlike live flight data, when two aircraft are on a collision course, we can alter the flight path of one aircraft to eliminate this problem.

The SIMD-based implementation is based on the ClearSpeed CSX600 board and implemented in the C^n language as described in [31]. Note that only one of the two CSX600 chips, i.e., one MTAP with a total of 96 PEs, has been used for the experiments. The reason for this is that with the CSX600, the two chips cooperate as two separate SIMDs. The latest SDK CSX600 software makes it easier to have the two chips operate as a single SIMD, but is still not as good a simulation of SIMD as obtained by using only one single chip.

B. Experiment Results

1) Performance on CSX600: We measure the timings in CSX600 SIMD environment in this section. The experiment results in Figures 3 and 4 show that the run-time on ClearSpeed is almost constant when the maximum number of aircraft is 1, 2, 3, and 4. Moreover, the jumps in graphs occur when the maximum number of aircraft per PE increases. These emulation results indicate that an AP's performance on ATC will become less efficient when the maximum number of aircraft assigned to each PE increases.

Next, we investigate experimentally the speedup and efficiency of the system. The sequential program is simulated by programs that run on only one ClearSpeed PE, and

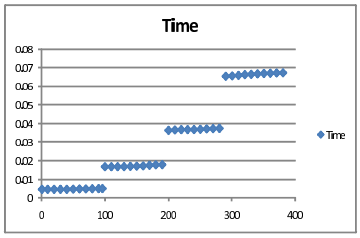

Figure 3. Time required for correlation for 10 to 380 aircraft.

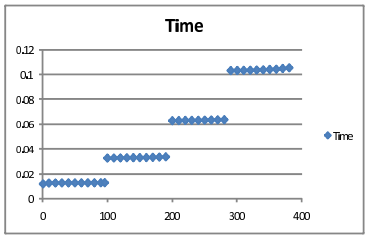

Figure 4. Time required for CD&R for 10 to 380 aircraft.

the running time is recorded as t_s. The parallel running time is recorded as t_p and p is the number of processors. One PE can only hold the data for 100 tracks because it only has $6k$ memory. The comparison of t_s and t_p for report correlation and tracking task is shown in Figure 5. We use the experiment data to calculate the speedup t_s/t_p and efficiency $t_s/(p \times t_p)$. The average of speedup is 96 and efficiency is 0.95, which are almost optimal. Because ClearSpeed processors emulate AP, the speedup and efficiency for AP should be even higher because the software emulation of the associative functions requires extra time on ClearSpeed. That is, for AP with $14,000$ PEs, the speedup should be fairly close to $14,000$ and the efficiency should be fairly close to 1.

Our current implementation allows a maximum of 17 tracks per PE, i.e., 1632 tracks in total. The results are shown in Figure 6 and Figure 7: the horizontal axis represents the number of tracks and the vertical axis the execution time

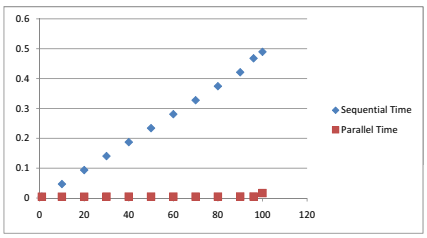

Figure 5. Speedup and Efficiency of Tracking.

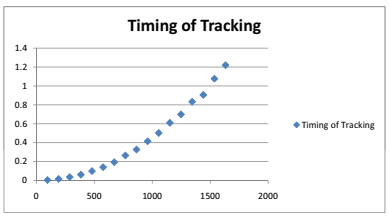

Figure 6. Timing of Tracking Algorithm

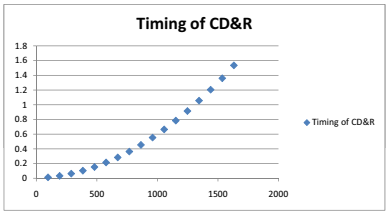

Figure 7. Timing for CD&R Algorithm

of the tasks in seconds. We assess the scalability of the tracking and CD&R algorithms by evaluating their running time over a wide range of number of tracks and plot the time each algorithm takes on a graph. We can see that the CSX approach can finish all the three tasks within deadlines even when the number of tracks is scaled up to 1500.

2) Timings for 8 Tasks: In this section we will show that our prototype can meet the deadlines for the hard real-time ATC tasks. Table I shows the performance of one flight per PE, i.e., 96 flights. The execution time (secs) is the time that is spent executing this task once. The processing time (secs) is the total time that is spent executing this task during an 8 second period. We can see that all tasks can be done within their deadlines. The total used time is 0.25508 seconds, which is only 3.19% of available time 8 second.

The maximum number of flights per PE is 7 to guarantee real-time requirements of the tasks on our CSX600 prototype. The performance is shown in Table II. Actually all tasks can easily be done within their deadlines, except for the 14th half-second cycle. The 3 tasks in that cycle are aircraft tracking, sporadic requests and CD&R and they take 0.498 second, which is essentially all (i.e., 99.6%) of the available time. All the other cycles have a reasonable amount of unused time. The used time is 6.67717 seconds, which is 83.46% of available time 8 second. We can put more tasks into some slots where there is unused time. The timing required for each of the various tasks is stable (i.e., is a constant), making this system very predictable.

In [29], an experimental comparison of the SIMD implementation of the Aircraft Tracking algorithm with a multi-threaded MIMD implementation was presented. For a number of planes ranging from 4000 to 14000 [1], the

[1]Since the number of planes that can be stored locally in each PE is limited, several iterations of moving data in and out of the local PE's memory had to be performed

Table I
PERFORMANCE OF ONE FLIGHT/PE

Tasks	Exec Time	Proc Time
Report Correlation & Tracking	0.00552	0.08832
Cockpit Display	0.00272	0.02177
Controller Display Update	0.00276	0.02209
Sporadic Requests	0.00155	0.01244
Automatic Voice Advisory	0.00544	0.01088
Terrain Avoidance	0.00782	0.0782
Conflict Detection & Resolution	0.01301	0.01301
Final Approach(96 runways)	0.00837	0.00837
Total		0.25508

execution times for the tracking algorithm for a large number of iterations were measured and collected. Based on these results, the Coefcient of Variation (COV), which is a common normalized measure of dispersion, and is dened as the ratio of the standard deviation to the mean of the measured execution times, was computed. The results in [29] clearly demonstrated the COV values for the SIMD implementation are several orders of magnitude below the ones for the multithreaded MIMD implementation. This allows the specification of meaningful and tight upper bounds on the SIMD task execution times, which is critical for an efficient design of a safety critical real-time system like ATC. The SIMD implementation did not only provide significantly more predictable execution times. In [29], it was also demonstrated that the scalability (with respect to the number of processing units) of the MIMD aircraft tracking task is significantly below the scalability of the SIMD solution. This was largely attributed to synchronization, context switching, scheduling, and load balancing overheads in the MIMD implementation. Compared to the unpredictability of real-time systems using MIMD [5], [29], the predictability of ATC system based on AP is far superior to the one based on MIMD.

V. CONCLUSION AND FUTURE WORK

The major purpose of this paper is to demonstrate that a SIMD or associative SIMD can provide a high quality solution to a challenging real-time task such as air traffic control. To accomplish this, we implemented an 8 task ATC prototype on a SIMD system and established that the worst case running time of this system is predictable and can meet the required deadlines for the various tasks. We use the ClearSpeed CSX600 accelerator to emulate an associative SIMD processor (AP). The details of how the CSX600 emulates the associative operations of an AP was given in the earlier paper [29]. The associative SIMD solution for the ATC prototype was developed on the CSX600 using the additional constant time operations. This system was able to meet every deadline for every task and has a near-optimal speedup and efficiency. Additionally, the system is predictable in that a worst case execution time can be determined for each task and this can be used to create a static schedule for this real-time problem. The worst case execution time for every task is stable in that it has a fixed running time. Algorithms or explanations are given to show how each of the tasks are computed using the ClearSpeed CSX600 emulation of an AP. While algorithms for three of these tasks appeared in [29], the algorithms given here have been improved and are simpler and easier to understand. This approach provides a solution for ATC that involves hardware and software that is considerably simpler and cheaper than what has been used in the past.

The ClearSpeed board is not powerful enough to handle the complete ATC problem nor handle the management of the magnitude of aircraft that such a system must support. However, this demonstration does show that an associative processor with $14,000$ processors should easily be able to handle the entire ATC problem for an ATC center. Since two associative processors, Goodyear Aerospace's STARAN and ASPRO, have been built and used for ATC-type applications, there is no reason to doubt that appropriate AP computers can be built for ATC. While the production of fully redundant, fail-safe systems can also be supported by this type of system, these topics are not addressed in this paper.

These contributions can provide major progress towards meeting the goals of FAA's NextGen Plan, which includes to fly more aircraft, more safely, more precisely, and more efficiently and to use less fuel [32]. Additionally, a V&V framework can easily be developed for this simpler ATC approach that can certify the ATC software, i.e., that all tasks meet their timing and resource requirements under all operational conditions.

In the future, we plan to implement all of the 8 ATC real-time tasks on the same MIMD system that had been used in [29], in the future and compare the performance of CSX600 and MIMD on this ATC prototype. In contrast with our AP solution, the MIMD-based systems are currently the approach used for the ATC problem. However, they suffer the disadvantages of using only average case running times due to their highly unpredictable worst case running times. In contrast with an AP, they normally use many MIMD programming techniques such as dynamic scheduling, load balancing, and pre-emption of tasks.

A possibly important extension to our current research would be to consider a GPU implementation using CUDA.

Table II
PERFORMANCE OF SEVEN FLIGHTS/PE

Tasks	Exec Time	Proc Time
Report Correlation & Tracking	0.20828	3.33261
Cockpit Display	0.10414	0.83315
Controller Display Update	0.11479	0.91832
Sporadic Requests	0.06887	0.55099
Automatic Voice Advisory	0.16663	0.33326
Terrain Avoidance	0.25761	0.25761
Conflict Detection & Resolution	0.29011	0.29011
Final Approach(96 runways)	0.16112	0.16112
Total		6.67717

Nvidia has many SIMD PE groups on its chips. The Nvidia technology including the latest FERMI chip has a lot in common with the MTAP approach of ClearSpeed, and implementing the CSX600 ATC algorithms on this architecture may provide another useful platform to use in this project.

REFERENCES

[1] S.Kahne and I.Frolow, "Air traffic management: Evolution with technology," *IEEE Control Systems Magazine*, vol. 16, no. 4, pp. 12–21, Nov. 1996.

[2] M. Nolan, *Fundamentals of Air Traffic Control*, 3rd ed. Wadsworth: Brooks/Cole, 1998.

[3] L. Yang and J. Kuchar, "Prototype conflict alerting system for free flight," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 4, Jul. 1997.

[4] W. Meilander, J. Baker, and M. Jin, "Predictable real-time scheduling for air traffic control," in *Fifteenth International Conference on Systems Engineering*, Aug. 2002, pp. 533–539.

[5] ——, "Importance of simd computation reconsidered," in *Proc. of the 17th International Parallel and Distributed Processing Symposium (IEEE Workshop on Massively Parallel Processing)*, Nice, France, Apr. 2003.

[6] J. Kuchar and L. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[7] Y.Bar-Shalom and T.E.Fortmann, *Tracking and Data Association*. Academic Press, 1988.

[8] H.A.P.Blom, R.A.Hogendoorn, and B.A.vanDoorn, "Design of a multisensor tracking system for advanced air traffic control," in *Multitarget-Multisensor Tracking: Application and Advances*, Y.Bar-Shalom, Ed., vol. 2. Artech House, 1990, pp. 31–63.

[9] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin, "Multiple-target tracking and identity management in clutter for air traffic control," in *Proceedings of the AACC American Control Conference*, Boston, MA, Jun. 2004.

[10] K.M.Liu, "Composition of kalman and heuristic tracking algorithms for air traffic control," Master Thesis, Kent State University, Kent, OH, Aug. 1999.

[11] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in tracking: A survey," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, 1998, pp. 103–123.

[12] D. Lainiotis, "Partitioning: A unifying framework for adaptive systems i: Estimation," in *Proceedings of the IEEE*, vol. 64, Aug. 1976, pp. 1126–1142.

[13] Y.Bar-Shalom and X.R.Li, *Estimation and Tracking: Principles, Techniques and Software*. Boston, Massachusetts: Artech House, 1993.

[14] D. Sworder and J. Boyd, *Estimation Problems in Hybrid Systems*. Cambridge University Press, 1999.

[15] H. Blom and Y. Bar-Sharlom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, Aug. 1988.

[16] X. Li and Y. Bar-Shalom, "Design of an interacting multiple model algorithm for air traffic control tracking," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 3, September 1993.

[17] I. Hwang, J. Hwang, and C. Tomlin, "Flight-mode-based aircraft conflict detection using a residual-mean interacting multiple model algorithm," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, Texas, Aug. 2003.

[18] I. Hwang and C. Tomlin, "Protocol-based conflict resolution for finite information horizon," in *Proceedings of the AACC American Control Conference*, Anchorage, May 2002.

[19] J. Krozel, M. Peters, K. Bilimoria, C. Lee, and J. Mitchell, "System performance characteristics of centralized and decentralized air traffic separation strategies," in *Fourth USA/Europe Air Traffic Management Research and Development Seminar*, 2001.

[20] Y.-J. Chiang, J. T. Klosowski, C. Lee, and J. S. B. Mitchell, "Geometric algorithms for conflict detection /resolution in air traffic management," in *36th IEEE Conference on Decision and Control*, San Diego, CA, Dec. 1997, pp. 1835–1840.

[21] R. Paielli and H. Erzberger, "Conflict probability estimation for free flight," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 3, pp. 588–596, 1997.

[22] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–219, 2000.

[23] (2009) Faa grants for aviation research program solicitation. [Online]. Available: http://www.tc.faa.gov/logistics/grants/

[24] W. Meilander, M. Jin, and J. Baker, "Tractable real-time air traffic control automation," in *Proc. of the 14th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, Cambridge, MA, Nov. 2002, pp. 483–488.

[25] S. Reddaway, W. Meilander, J. Baker, and J. Kidman, "Overview of air traffic control using an simd cots system," in *Proc. of the International Parallel and Distributed Processing Symposium (IPDPS'05)*, Denver, CO, Apr. 2005.

[26] M.Yuan, J.W.Baker, F.Drews, and W.Meilander, "Efficient implementation of air traffic control (atc) using the clearspeed csx620 system," in *Proc. of the 21st IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, Cambridge, MA, November 2009, pp. 353–360.

[27] J. Potter, J. Baker, S. Scott, A. Bansal, C. Leangsuksun, and C. Asthagiri, "Asc: An associative-computing paradigm," *Computer*, vol. 27, no. 11, pp. 19–25, 1994.

[28] M. Jin, J. Baker, and K. Batcher, "Timings for associative operations on the masc model," in *Proc. of the 15th International Parallel and Distributed Processing Symposium (IEEE Workshop on Massively Parallel Processing)*, San Francisco, CA, Apr. 2001, pp. 193–200.

[29] M.Yuan, J.W.Baker, F.Drews, L.Neiman, and W.Meilander, "An efficient associative processor solution to an air traffic control problem," in *Large Scale Parallel Processing IEEE Workshop at the International Parallel and Distributed Processing Symposium (IPDPS2010)*, Atlanta, GA, Apr. 2010.

[30] J. A. Rudolph, "A production implementation of an associative array processor - staran," in *The Fall Joint Computer Conference (FJCC)*, Los Angeles, CA, Dec. 1972.

[31] (2007) Clearspeed technology plc. clearspeed whitepaper: Clearspeed software description. [Online]. Available: https://support.clearspeed.com/documents/

[32] (2009) Faa's nextgen implementation plan. [Online]. Available: http://www.faa.gov/about/initiatives/nextgen/media/ngip.pdf