Overview of Air Traffic Control using an SIMD COTS system

Stewart Reddaway
WorldScape Inc
Marlton, NJ
sfr@wscapeinc.com

Will Meilander (retired)
Johnnie Baker
Justin Kidman
Kent State University
Dept Computer Science
Kent, OH 44242-0001
jbaker@cs.kent.edu

Abstract

Air Traffic Control is an important application with demanding real-time database processing requirements. Systems that have been implemented using current approaches have typically been expensive, late, over-budget and have not performed up to specification. In part this is because, in attempting to meet the real-time requirements, developers have been driven to use complex algorithms and software for what are inherently relatively simple requirements. Our analysis indicates that the use of modern SIMD COTS systems will enable guaranteed real-time performance to be achieved with simpler algorithms on modest amounts of hardware. This paper covers the system, the approach to the application and some of the solution details.

1. Introduction

This paper provides an overview analysis of Air Traffic Control (ATC) using an SIMD COTS system, and covers the system, the approach to the application and some of the solution detail.

ATC is an important application with demanding real-time processing requirements. Systems that have been implemented using current approaches have typically been expensive, late, over-budget and not performed up to specification. In part this is because in attempting to meet the real-time requirements developers have been driven to use complicated algorithms and software for what are inherently relatively simple requirements. Our analysis indicates that the use of modern SIMD COTS systems will enable guaranteed real-time performance to be achieved with simpler algorithms on modest amounts of hardware. The overall requirements of ATC are illustrated in Figure 1.

1.1 A previous analysis

The following Tables 1 and 2 are tables 2. and 3. from [1]. That paper assumed currently available technology. The analysis of this paper is based on characteristics of a real modern SIMD chip and board.

Table 1. ATC – Worst-Case Environment

Reports per second	12,000
IFR flights	4,000
VFR/backup flights	10,000
Controllers	600

There are thus up to 14000 flights, 4000 of which are under the control of this system. There are up to 12000 radar reports/sec; 6000 are dealt with in each 0.5 sec interval.

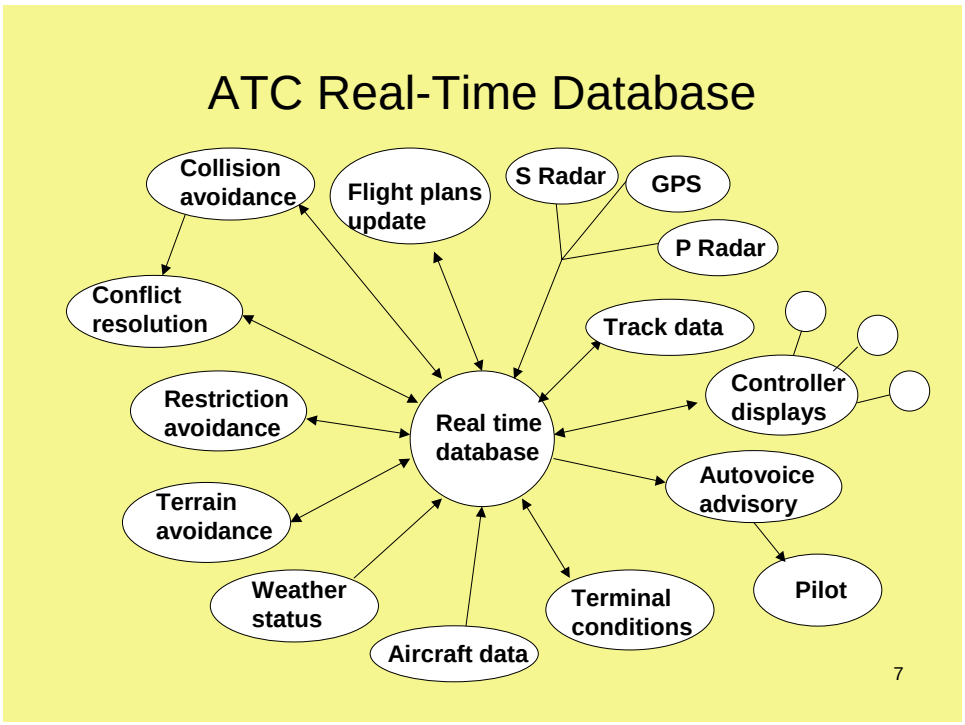

Figure 1

Table 2. Statically Scheduled Solution Time

Task	p	j	c	d	Proc time
1. Report Correlation & Tracking	.5	15	.09	.10	1.44
2. Cockpit Display 750 /sec)	1.0	120	.09	.20	.
3. Controller Display Update (7500/sec)	1.0	12	.09	.30	.72
4. Aperiodic Requests (200 /sec)	1.0	250	.05	.36	.4
5. Automatic Voice Advisory (600 /sec)	4.0	75	.18	.78	.36
6. Terrain Avoidance	8.0	40	.32	2.93	.32
7. Conflict Detection & Resolution	8.0	60	.36	3.97	.36
8. Final Approach (100 runways)	8.0	33	.2	6.81	.2
Summation of Tasks in a period P					4.52

The system period P (in which all tasks must be completed) is 8 seconds
p the task period time, is used to determine the next task release time $r_i + 1 = r_i + p$,
j is the execution time, in microseconds, for each jobset in a task,
c is the cost for each task for the worst-case set of jobsets,
d the deadline time for each task $r_i + c + .01$ (includes 10 ms interrupt processing per task)

Table 2 shows the computing work under 8 headings. The work of each heading is dealt with once every p secs. j, c and d characterize the timing for the suggested hardware. j is the time (usecs) for a unit of computing, c (secs) is the total time for all the units, d is the time from the start of an 8 sec period by which that computing is finished. For example, the Report Correlation & Tracking, has 6000 reports to correlate against all tracks. Each report process is estimated to take 15 usecs, so j is 15 and c is 0.09 (derived from 15 * 6000/10**6). With 10 msecs allowed for interrupt actions, and starting at the beginning of the period, d = 0 + 0.09 + 0.01 = 0.10. The "Proc time" column is the total processing time for an 8 sec period. As there are 16 half secs in 8 secs, the correlation time is 0.09 * 16 = 1.44 secs. This is ~32% of the 4.52 processing time, which is a total processing load of ~57% of available time.

1.2 Modern SIMD chips

The chips considered are the 200 MHz CS301 and a 250 MHz successor due in Q2 2005.

These SIMD chips have powerful PEs (Processing Elements) with both floating and fixed point hardware in every PE. For 16 to 32-bit work it is usually faster to use floating point, and this is assumed in this study. Other features have commensurate speed, such as I/O and moving data between the "poly" RAM (RAM in each PE) and the register files. The I/O is also flexible, with each PE able to specify its own address for external "mono" RAM. The poly RAM is 4kB/PE, rising to 6 kB/PE in the successor chip. There are 64 PEs in the current chip, and 96 in its successor. The generic core of the chip is shown in Figure 2. More detail can be found in [2].

In multi-chip systems, each chip runs its own program. Global SIMD operation is achieved by each chip running the same code and, when needed, synchronizing by software. (See below.)

Cycle estimates are for highly optimized assembly coding. Less optimized coding can still achieve the real-time requirements, as there are big margins over real time.

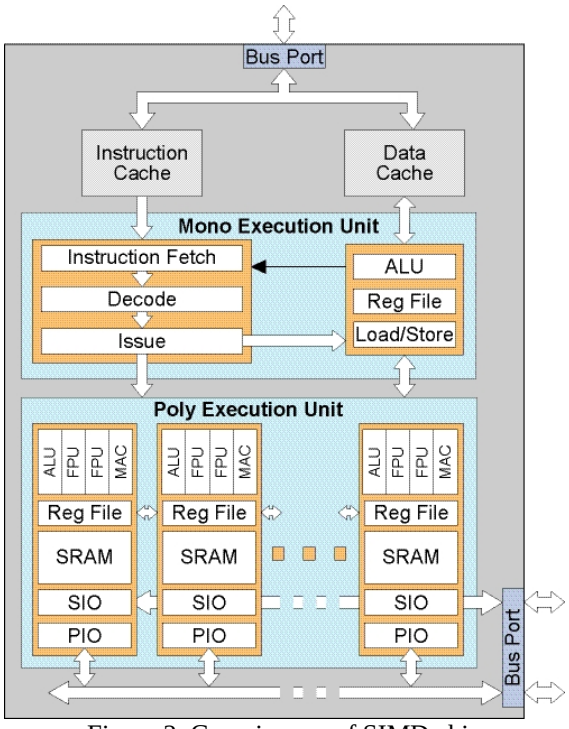

Figure 2. Generic core of SIMD chip

1.3 SIMD boards

The current CS301 boards contain 2 CS301 chips and 1 GB of "mono" DRAM. A proprietary ClearConnect bus runs from one CS301, across the other CS301 and, via an FPGA, to the DRAM. There is a PCI interface, which can be connected to a host computer such as a PC. Research at Kent State University will use this COTS board.

The successor board has two successor SIMD chips, each with an on-chip interface to their own DRAM. The ClearConnect bus again connects the two chips and, via an FPGA, the board's 64-bit PCI-X interface.

2. Analysis for a real modern SIMD system

A qualitative difference of this analysis to that in [1] is that here it is assumed that many tracks are mapped in one PE, whereas [1] assumes there is one PE for every track. This is possible because the modern chip is fast enough to do the work of many tracks per PE. The algorithm is still SIMD. Each PE can support about 100 tracks, so 14000 tracks require about 140 PEs. Either three of the 64-PE chips or two of the successor 96-PE chips are needed.

2.1 Reduction and broadcast operations

Using SIMD for ATC requires efficient implementation of global Reduction operations. It is these operations that require global synchronization. Arguably this is the "difficult part". ATC uses global tests and PickOne. The others are included for completeness.

2.1.1 Global test. The simplest operation is to test if a Boolean condition is true anywhere. An application may have a large array of Booleans, with many elements in each PE. A key for high performance is to first reduce these values in each PE to a single Boolean. Most SIMD chips will have hardware that produces a mono (scalar) Boolean by applying a Boolean operator across every PE in the chip. For current chips this instruction takes about 15 cycles to produce a result.

If a global test is to be applied across several chips, an across-PE mechanism is needed that first checks that all chips have finished and then combines the test results. There is no hardware help for this. It can be done by each chip recording both that it has finished, and its result, in the board mono DRAM. Each chip can then read the DRAM and, when it is found that all chips are recorded as finished ("synchronized"), read the chip results and compute the global result. With only a few chips on a board, this across-chip work is estimated at about 100 cycles. This assumes that all chips finished their previous computing at nearly the same time, which is likely to be the case if they have all been doing the same work.

Thus within-PE global tests take about 15 cycles, and across-chip tests about 115 cycles. A system with several boards (bigger than needed for ATC) would take longer.

2.1.2 PickOne. PickOne is an operation on a Boolean array that picks a unique F(alse) element. (Alternatively, a unique True element.) Usually the first F element is picked. There is no hardware to support this, and it is non-trivial. Each PE finds if there is any F in that PE. Using across-PE work, the first F PE (if any) is found. Then, within-PE Boolean work in the selected PE finds the first F in the PE. One way of doing the across-PE work is by a “binary chop”. A chip test finds if there is an F in the first half of the PEs. If not, the second half is chosen. The chosen half is then tested to find the quarter. In a logarithmic number (6 for 64 PEs) of such tests the first PE with an F is found. Carefully coded, this across-PE but within-chip work takes about 120 cycles.

With multiple chips, each chip writes to external mono RAM whether it has an F, and, after synchronization, each chip reads all results and computes if it is the selected chip. Each chip knows its position, treating the chips as a linear sequence. This takes about 100 cycles for chips on the same board. In some PickOne contexts the bulk of this across-chip work will already have been done as part of a global test. Thus an isolated PickOne takes about 220 cycles, and a PickOne after a global test takes about 100 cycles. These figures exclude the within-PE work when there are multiple Booleans in a PE.

2.1.3 Max and Min. There are two alternative algorithms for maximum that are almost equally good. One is described here. With big arrays, within-PE work should be done first, followed by a single across-PE stage.

The first algorithm treats the bits of the numbers in sequence, starting with the most significant. Numbers that cannot be the biggest in the chip are eliminated progressively, but always ensuring that at least one number is retained in the “competition”. A global test is applied to numbers that are still in the competition to find if the next bit is T(rue) anywhere. This result is broadcast, and if any were T, numbers with an F are eliminated. A record of the results of the global tests gives the scalar max. This within-chip work is constant time and takes about 20 cycles per bit. (e.g., about 320 cycles for a 16-bit array.) For multiple chips, the chip results are posted in mono RAM, so each chip can read them and work out the global maximum. This adds about 100 cycles.

The second algorithm will be in [3].

2.1.4 Sum. For large arrays, within-PE work is done first. The across-PE work in SUM is done by shifting, using the above "log(n)" approach. With 64 PEs, within-chip Sum requires about 200 cycles for up to 32 bits. For 96 PEs, the last 2 stages can be replaced by copying 3 partial results to mono RAM, and about 250 cycles are required. Across-chip adds about 100 cycles.

2.1.5 Broadcast. This is not strictly a reduction operation, but is important. Broadcasting data from mono RAM to all PEs is part of the instruction set of a chip. With more than one chip on a board, each chip can access all the mono RAM. When the mono data is stable, such as when correlating a sequence of radar reports, no validity check is needed, but in other cases validity needs to be semaphored to achieve synchronization.

2.2 Report Correlation & Tracking

An ATC system has radar reports from many radars, which give positions of objects in air space. There is a real-time database of the tracks of flights in 3D air space which are updated by correlating the position of each radar report with the position of a track extrapolated to current time. When a track correlates with a report the track information is updated to absorb this new information about the track. Over a 0.5 sec interval all radar reports are assembled, and then one at a time these reports are trial correlated against all tracks in the database. (About 6000 reports every one-half second.)

The track database is held in the mono DRAM, and at the start of the Correlation task each track loads 3 position coordinates, plus the estimated uncertainties in those coordinates into the on-chip poly RAM. With 14000 tracks and 192 PEs there are 73 tracks per PE.

It is assumed in each report that the location coordinates are x, y, and, often, h (height). There is also an uncertainty in x, y, and h. Reports are dealt with one at a time by broadcasting the 6 values and comparing the positions of every track with the report. Boxes of uncertainty are formed around both track and report positions, so each track can decide if there is an intersection between its box and a report box. If a unique track intersects, then the report data is stored with that track for later track update, and the track is marked not to correlate with later records. If two or more tracks correlate with the same radar report, then the correlating tracks are marked as part of a multiple hit. If a report does not correlate, it is marked to participate in a later round of correlations with wider tolerances.

With many tracks per PE, the broadcast of report data is stored only once per PE. Likewise, the report box is computed only once, after broadcast.

Computing the track boxes is done only once per “correlation period” (0.5 secs). Thus the only work per track per report is 6 comparisons and 5 Booleans.

For each report, each track's Boolean correlation result contributes to a count of the number of successful correlations in each PE, and to find the within-PE track number of the first correlated track. The count can be done with 2-bit numbers, representing 0, 1 and many.

The across-PE work is based on whether or not the PE total is 0. A global OR test is done for each radar report to find if there are any hits. If there are no hits, the report is marked for the next correlation round. If there are hit(s), a PickOne function is used to find the first PE with a nonzero total. The Picked PE changes its count to 0 if it was previously 1. Then a global test is done to see if any more tracks correlated. The result of this test is broadcast, and if, globally, at least 2 tracks correlated, all tracks are scanned to mark the correlated tracks as multiple correlations. If there was just one correlation, the Picked PE marks as successfully correlated the track for which it saved the within-PE address, and copies the report position data to the record for that track.

When all reports for the period have been dealt with, the process is repeated for all unmatched reports up to twice more, with wider track tolerances. After 3 rounds, any unmatched reports are used to start new tracks.

The following was done before any correlations:

- empty tracks are identified
- the empty tracks are counted within-PE
- a within-PE list is formed of positions of empty tracks. (1 byte of poly storage per track)
- a global "scan sum" gives each PE the 2-byte global "empty number" of its first empty track

As this work is done only once, it takes negligible time.

For each unmatched report:

- increment a mono count of new tracks
- compare this count with each PE's empty track numbers. (E.g. if this is the 57th new track, only one PE will contain this empty track number. It might e.g. be the 3rd empty track in that PE.)
- the within-PE address is used to initiate a track in this unique empty track with the report data

2.2.1 Higher quality Correlation. The error in a radar report has a component along the radar "radius" (from measuring the time of the radar response) and a component across the radius (from error in the azimuth angle). Other than for short-range reports, typical radars have much bigger errors in azimuth than range. The ideal "report box" is an eccentric ellipse, but an elongated rectangular box is quite a good approximation. However, in system coordinates the rectangle will not usually be aligned with the axes, and for computing efficiency a box aligned with the axes is needed. If such a box is to include all possible good correlations, it must be much bigger than the original box, but this will include dubious correlations. Better is to do the correlation in "radar coordinates" with axes along and perpendicular to the radar radius. This aligns the original radar box to the axes. Detail of this approach will be in [3].

2.2.2 Estimating cycles. With 6000 reports, the estimated worst case is that 1000 fail to correlate on the first round, 300 fail on the second round and 100 fail on the last round (and hence start new tracks). Thus there are 7300 correlations. We assume there are 200 multiple hits.

There are 73 tracks/PE. The per track work of:

- 6 compares
- 5 Boolean ops
- one step of count hits and find the first address

is estimated at ~40 cycles with low level coding, or ~400 cycles in Cn, the extension of ANSI C used to program the chips. (The low level code assumes the data is floating point, and that the pipelined effect of the floating point subtracts is minimized by doing 4 tracks/PE as a batch.)

With low level coding there are 73 x 7300 x 40 = 21.3 M cycles per correlation period.

Each report (except those having multiple hits) can generate one PickOne, up to 4 global tests and one within-PE copying of report data. The average number of global tests is 2.2. (5000 reports require 2 tests, 700 3 tests, 200 4 tests and 100 3 tests.) The reduction work total is ~5800 x (100 + 2.2 x 115 + 20) = 2.2 M cycles.

Starting 100 new tracks cost about 100*50 = 5k cycles. Marking all hit tracks for the assumed 200 multiple-hit reports, is estimated as ~200 x 73 x 10 = ~146k cycles.

The estimated total is thus ~23.7 M cycles, a 19.7% load at 250 MHz in a 0.5 sec period.

2.2.3 Storage. Track storage is viewed as being primarily in mono RAM, with data brought into poly RAM as needed. The space per track needed is six 32-bit numbers, plus 1 status byte and 1 byte for the empty track list. If the report data for correlated tracks is to be saved in poly RAM, this adds ~20 bytes. Slightly slower is to save the report data in mono RAM. Thus either 26 or 46 bytes/track is needed.

There is 6 kB/PE of poly RAM in the successor chip, although some working space is needed. Thus each PE can hold a maximum of either ~210 or ~120 tracks.

The time to load the 26 bytes/track at the start of a period is negligible (~1 msec). Similarly negligible is outputting data that needs saving at the end.

2.2.4 Faster Reductions in bulk. It is possible to perform the reduction work in Correlation faster by dealing with several (e.g. 16) reports before doing

across-PE reductions. The technique can greatly reduce the time for reduction operations, and detail will be given in [3]

2.3 Conflict Detection & Resolution

2.3.1 Conflict detection. Each IFR track is projected 20 minutes to see if there is a conflict with any other projected track. This is done for each dimension in turn:

- Computing a min and a max closing velocity, which might be negative
- Tolerances are applied to compute the min and max current distance apart of the two tracks
- Division gives min and max times for that dimension to be identical within tolerance. Negative time means tracks are separating.

This produces 3 min times and 3 max times. If the biggest min time is smaller than the smallest max time, there is potentially a conflict. (This algorithm was devised by Ken Batcher.) Conflict is declared after potential conflicts with the same track in two successive periods.

2.3.2 Conflict resolution. If there is a conflict, the heading or altitude of the track is adjusted, and the algorithm run again. A schedule of adjustments is gone through until there is no conflict for that track. More detail will be in [3].

2.3.3 Implementation. With 192 PEs and 14000 tracks, there are 73 tracks/PE. If IFR tracks are held separately from VFR, then they occupy only 21 of the 73. When one IFR track in every PE (192 IFR tracks) has been dealt with, tests need only be done against 72 tracks/PE. Etc.

Each track has 6 coordinates and 6 velocity components, including both min and max. With a status byte, this is 49 bytes/track. 73 tracks take up 3577 bytes out of the 4k or 6k bytes of poly RAM/PE.

The active IFR track is broadcast to every PE, and then conflict detection is performed against up to 73 tracks/PE. Conflict detection can be done with:

- 6 subtracts to get min and max closing velocities
- 6 subtracts to get min and max distances
- 6 divides by closing velocities for the min and max distances
- Find max min and min max values. 4 compares
- Subtract max min from min max. Sign is result
- one step of within-PE OR (1 Boolean)

(Min and max mean nearest minus infinity and nearest plus infinity respectively.)

When all tracks/PE have been dealt with, a global across-PE OR test finds if there is a conflict. If so, another reduction finds the conflicting track, its' track number is stored and a decision made whether to declare a conflict.

2.3.4 Performance. Floating point divides take 47 cycles. With 4 tracks/PE done together, the estimate is about 332 cycles per track per PE. An average of 63 tracks/PE and 100 cycles for the global test, make 21k cycles per IFR track. With 4004 tests, the total is ~86 M cycles.

The conflict period is 8 secs (1600 or 2000 M cycles), so the processing load is about 5%. In Table 3 above, this processing is 0.36/8 = 4.5% of the available time.

2.4 Cockpit Display

The next set of up to 750 IFR flights for which displays are required are identified. The x, y and h positions of all tracks are transferred to poly RAM.

Display flights are dealt with sequentially. The x, y, h and velocity data is broadcast to all PEs. The position of the display flight is projected forward 10 secs. All track coordinates are transformed so they are centered on the broadcast track, and track coordinates and velocities are rotated to coordinates in which the broadcast track is heading "North".

Every track is tested to see if it is in a 10 mile x 10 mile box around the display flight, elongated to include a 30 sec projection of the display flight. It is assumed that the worst case average number of hit tracks is 12. In the worst case, all 12 could be in the same PE.

2.4.1 Algorithm and performance. More detail will be in [3]. The task is estimated at 5.9 M cycles.

2.5 Controller Display Update

~7500 out of 14000 tracks are selected as being within the controlled area. Information on these tracks is sent, on a pipe, to everyone, with each control station selecting the tracks relevant to its local area.

The information required is track position (x, y, h), heading and speed. Speed is in knots using BCD (3 or 4 decimals). Each track is made into a fixed size "message" of 16B, which includes a flight ID. All data is 16-bit.

There is no "all against all" component in the algorithm, so this task is fast, estimated at 360k cycles. Detail in [3].

2.6 Terrain Avoidance

All flights within the local control boundaries (assumed to be 7500, or 40 flights/PE) need to be warned if they risk running into ground terrain. At 8 sec intervals every flight is projected 1 minute and tested against every feature in the terrain map.

The terrain can be mapped (tiled) with thousands of triangles of various sizes, whose vertices are stored. This irregular mesh forms a surface in 3D, and covers both the natural topography and buildings/masts etc. What sticks up is more important than small valleys that stick down. The tiling should be as high or higher than every feature. Triangles are used because each triangle defines a planar surface; a shape with more points, such as a quadrilateral, might or might not lie on a single plane.

Either (a) the triangles are loaded into poly RAM and the flights are broadcast one at a time, or (b) flights are loaded and triangles broadcast. The latter is assumed here.

Overhangs are not included, so only the base of the flight projection box need be checked for intersection with any triangle (or restricted flight zone).

2.6.1 Algorithm. The projected bottom surface must be planar, ie the left and right bottom line projections must lie in a plane. The line forming the intersection of this plane and the plane containing a triangle is computed. The parts of this line (if any) that lie inside (a) the triangle and (b) the bottom of the projected box, are computed. If any part of the line lies in both the triangle and the bottom of the box, there is a collision, and the shortest time to collision is computed and reported. 30 arithmetic, comparison and Boolean operations are estimated.

2.6.2 Performance. The performance is nearly proportional to the number of triangles.

The x, y position of all ~14000 flights are transferred from mono to poly. Selection of participating flights is done based on x, y. The surviving track numbers are then packed to make them dense in the poly RAM, and used to construct mono addresses for loading the track data into this dense space. There are ~40 bytes/flight, and ~40 flights/PE, so there is plenty of poly space. The two bottom lines of the flight projections are computed. Due to velocity uncertainties, these lines may not be parallel. This setup work is estimated at ~900 usec (~450k cycles) to transfer ~420 kB, ~60k cycles to pack the track numbers and ~4k cycles to compute bottom surfaces.

For each triangle:

- Broadcast ~40 bytes (pre-computed from the three vertices), ~50 cycles
- Compute intersections, ~40x~70= ~2.8k cycles
- Within-PE OR of hits, ~40x~2=~80 cycles
- Single-chip global test for any hits, ~15 cycles

With 20k triangles, this totals ~60 M cycles.

For each flight with at least 1 intersection, the projected time to intersection and the triangle ID are extracted to mono lists (separate for each chip). ~1000 cycles, or 5k with a worst case of 5 intersections per chip.

The short mono lists of intersections are output to either the controller affected or to the automatic voice advisory system for general aviation activities.

Total time, ~61 M cycles. Over 8 secs, this is a load of 3.8%. [1] allocated ~7% of all computing to this task, so 20k triangles is manageable.

2.7 Sporadic (Aperiodic) requests

This task is primarily the input of various pieces of information into the various database tables held in mono RAM, and responding to queries to the database. The messages are buffered in the host computer over a 1 sec period. This happens simultaneously with the processing of other tasks in the SIMD chips, and so does not contribute to the cycle count. Once per second the host writes the message data to the correct places in the database. (Alternatively, the buffer can be transferred as a block to the DRAM on the board, and the database insertion done by one of the SIMD chips.) With no simultaneous tasks, there are no synchronization or scheduling issues. Queries extract data from the tables and transmit responses to requesters.

Most messages are small, but some, such as an update to the wind table, can be larger. There are estimated to be a maximum of 200 messages per second.

The task is estimated to take about 1 M cycles.

2.8 Automatic Voice Advisory

Automatic Voice Advisory (AVA) automatically advises, by voice, an uncontrolled (VFR) flight of near term conditions of other aircraft and terrain. It gives them a near equivalent of the cockpit display. The computing is similar to that for Cockpit Display, but rather simpler. It is estimated to be ~60% of the load of cockpit display. More detail in [3].

2.9 Final Approach (Runways)

Each flight has a flight plan which specifies (among many other things):

- departure terminal and planned departure time.
- destination terminal and planned arrival time.

A task is run once every 8 secs in which the information for each of the assumed 100 runways is gathered and, if needed, a queue is organized and any consequential modifications are inserted in each flight plan. Where a change is recommended for a flight, the relevant controller is informed. If possible, stacking is

avoided by delaying tracks in flight, but in emergencies tracks may be allocated to stacking, and the flight is told the stack detail by the controller.

This work applies primarily to the 4000 IFR flights, but some others may be included. Thus all 14000 flights participate, but some statistical estimates allow for most tracks not being actively involved.

Detail will be in [3]. The estimated cycle count is ~870k cycles. In 8 secs (1600 or 2000 M cycles) this is a load of < 0.1%.

2.10 Other parts of the computing

Flight Plan/Track Conformance has been studied in a comparable way to Table 1 of [1]. Table 3 below shows what can be achieved by programming in assembler, using 32-bit floating point and dealing with 4 tracks/PE at a time to minimize the effects of the pipeline.

Table 3

Operation	Memory (bytes)	Time (cycles/track)
Select all flight plans	1	1
Get Xf, Yf, Hf, xd, yd, hd	24	
Xf1 = Xf+xd, Yf1= Yf+yd, Hf1=Hf+hd		5.3
Get Xt, Yt, Ht, sin(hdg), cos(hdg)	20	
X'=(Xt-Xf)*cos + (Yt-Yf)*sin		7
Y'=(Yt-Yf)*cos - (Xt-Xf)*sin		3
If |X'| > K1, (set alert flag)		5
If |Ht – Hf| > K3, (set alert flag)		7
If |Y'|>K2, over-write Yf1 with Yt		6
Store Xf1,Yf1,Hf1	12	
Total	57	34.3

There is no per track work required to get K1, K2, K3, as they are fetched from mono and broadcast once. The ABS operations have been added. The 57 bytes of memory take ~15 cycles, but most of this (and the looping) can be overlapped with processing. The estimate for optimized code is ~38 cycles (0.19 usec)/track/PE. With 21 IFR tracks/PE this is 4 usec, compared with the 7.34 usec of [1].

3. Speedup with Sorting

Each of Terrain Avoidance, Correlation, Conflict Detection, Automatic Voice Advisory and Cockpit Display involve some kind of "all against all" matching, involving proximity. Sorting can greatly speed these up.

This is still SIMD computing. The sorting is partly based on Batcher's Bitonic Sort, is SIMD parallel, and has data-independent deterministic speed. The rest of the computing is somewhat data-dependent, but even in worst cases there is a big speedup.

The speed gains from sorting are dramatic. The pay-off can be taken as:

- less optimized coding
- more complex or bigger requirements
- less hardware

The disadvantage is more complex algorithms.

3.1 Correlation

The x coordinate of all tracks is input to poly RAM, sorted and mapped with PE LS and the poly address MS. Thus each PE has tracks from across the x range. Track numbers are retained through the sort and used to construct a mono address with which to fetch the rest of the track data from mono.

The x range is divided into ~200 bins, and a mono array is formed giving the minimum and maximum poly addresses that have tracks for that bin.

Each radar report is dealt with in turn. The min and max x values are found, and, allowing for the worst track x uncertainty, are used to look up in the mono arrays the min and max poly addresses of tracks that might correlate. Only addresses in this range are used.

As the correlation boxes are not very big, only ~2 bins of tracks need be correlated, and the estimated worst case average number of tracks per PE that need to be processed is 3-4. (With 192 PEs, this is about 600-800 tracks.) This compares to 73 tracks/PE without the use of sorting, or about 20 times less. With an average 3.5 tracks, the core correlation processing is estimated at ~1.2 M cycles.

Four other contributions to the cycle total need considering. Sorting is estimated to take ~200k cycles. Reduction operations are still needed, and were estimated at ~2.2 M cycles, or ~220k cycles with advanced methods. The work to find the range of poly addresses will take ~200k. The final output of the update data from poly to mono will take ~400 usecs, or ~100k cycles.

Thus the total cycles (every 0.5 secs) is estimated to reduce from ~24M to ~3.9 M (or ~1.92 M with advanced Reduction processing), a speedup of over 6x (or 12x).

3.2 Terrain Avoidance

If terrain is represented by a surface in 3D of about 20k triangles, then the triangles can be permanently sorted on the minimum value of x in the triangle. The

sorted triangles are mapped onto the poly RAM, with PEs as the LS dimension and RAM address as the MS dimension. Details will be in [3]. The worst case speedup due to sorting is estimated at about 9x. Increasing the number of triangles increases the speedup.

3.3 Conflict Detection

Sorting by x velocity as well as x position enables an estimated speedup of about 3x. Detail in [3].

3.4 Cockpit Display and Automatic Voice Advisory

Sorting on x is estimated to produce a speedup of 9x for both these tasks.

4. Summary Table

Table 4 summarizes the overall performance estimates. Two next generation chips should do all the processing with a good margin.

5. A staged plan of work

1. Establish requirements and algorithms to implement.
2. Implement them in a high level language (Cn) using one CS301 chip.
3. Move key code to a mix of in-line assembler and assembler routines. Do some speed and space optimization. It is expected that real-time speed for ~5000 tracks can be achieved.
4. Extend the reduction codes to 2 SIMD chips, thus enabling ~10000 tracks in the application.
5. Project performance to a board with two successor chips. This should enable ~20k tracks at real-time speed.
6. Move the application to the successor board.
7. Demonstrations, reports and presentations.
8. Using sorting, develop faster code versions that will enable bigger applications with less hardware.

Kent State University are well placed to do an ATC project as they have a board with 2 CS301 chips and a substantial amount of off-chip DRAM. It is estimated that 1. to 6. above could be achieved in ~12 months. Extensions to the project could do 7. and 8. above, as well as further work to both study and implement the needs of a live system, including fault tolerance.

References

[1] W. Meilander, M. Jin, J. Baker. Tractable Real-Time Air Traffic Control Automation. Proceedings of the 14th IASTED International Conference Parallel and Distributed Computing and Systems, Cambridge, USA, 2002, pp. 483-488

[2] www.clearspeed.com

[3] To be published.

Table 4. Performance Summary

Task	Reference (sec)	3 x CS301	2 x next chip	Next chip with sorting
1. Report Correlation & Tracking	1.44	1.98(1.8)	1.58(1.44)	0.25(0.123)
2. Cockpit Display (750 /sec)	0.72	0.24	0.19	0.021
3. Controller Display Update (7500/sec)	0.72	0.015	0.006	0.006
4. Sporadic Requests (200 /sec)	0.4	0.04	0.033	0.033
5. Automatic Voice Advisory (600 /sec)	0.36	0.15	0.12	0.013
6. Terrain Avoidance	0.32	0.31	0.25	0.03
7. Conflict Detection & Resolution	0.36	0.44	0.35	0.12
8. Final Approach (100 runways)	0.2	0.005	0.004	0.004
Total	4.52	3.4(3.22)	2.54(2.4)	0.48(0.35)

1. All times are secs, and are the compute time over the total time available of 8 secs.
2. Times in ( ) are with the faster bulk reduction operations
3. The next chip figures allow for 250 MHz (rather than 200 MHz for CS301).