September 5, 2003

From: Dr. John Hunt

Brief History of MK1 and GP4

Readers: Please bear in mind that this all occurred forty years ago!

During the years of designing analog flight simulators we closely followed the progress of digital computer design with the hope of finding an affordable computer which would meet the needs of real time confutation in a full flight simulator. We were using existing commercial digital computers in the late 50s to analyze the validity of our mathematical models for the analog computers, and had developed a fairly complete understanding of the required speed of computation (i.e., the allowable time interval between successive computation cycle repetitions) to achieve acceptable solutions.

The principal problem which dictated minimum time between repetitions arose from the necessity to provide a large number of repetitions per cycle in the solution of differential equations whose solutions represented a damped cyclic oscillation such as short-term pitch stability (angle of attack) and yaw-roll dynamics (Dutch roll). Attempts to reduce the number of calculations per cycle are known to result in a marked reduction in damping and a distortion of the computed cyclic oscillation frequency. A further but considerably less demanding consideration was to avoid human perceptible steps in the solutions. Needless to say, neither of these problems existed in analog computer design.

We were fully aware of the enormous advantages in digital computation, especially in such areas as function generation (especially for functions of more than one variable), in accuracy, in all aspects of navigation, and in flexibility for change. Unfortunately, in 1961 the only available commercial computer which even approached sufficient computational speed was the IBM 9600/9604 family, at about two million dollars each. This was an intolerable cost in view of simulator economics at that time. Control Data had a roughly similar computer at about the same price. DEC had nothing remotely useable. None of the many so-called minicomputers existed at the time.

One of the major costs in the construction of suitable computers at that time was the cost of core memory, the only random access memory available with acceptable speed (6 microsecond access time in 1961). Semiconductor RAM and ROM memory were not available until the early seventies, and even then they were expensive for many years (Now, of course, semiconductor memory is practically free!) Inexpensive drum memories with relatively lavish capacity were available and represented a proven technology. Unfortunately, the access speed of the drum memories was intolerably slow. It is interesting to note that our exploratory calculations (for lateral stability of the 707/DC8 families) required about an hour to explore a few cycles of oscillation. These computations were on an IBM650 drum computer using an early interpretive language that was the grandparent of Basic. The absurdly slow computation was largely caused by the necessarily slow access time of the drum. It should be noted that today's best hard drives (the genetic successors of the old drums) only offer a threefold increase in access time! The IBM650, in spite of its slow memory and serial computation (vacuum tubes of course), still was the first really triumphant success in the business computer world.

I began thinking about the possibility of separating the storage of true, constantly changing variables such as angular and linear velocities on the three aircraft axes, aircraft attitude direction cosines, altitude, etc. separately from fixed data and instruction code. The latter non-variables required at least one hundred times as much storage as the true, constantly changing variables. Recognizing that storage of only the true variables in random access core was economically practical, I concentrated on the possibility of using a massively parallel readout drum to deliver the instructions and related stored numerical constants in an orderly sequence, with the instructions and data being delivered from the drum right at the instance they were needed for computation. This was the computer equivalent of today's "just in time" component delivery so dear to the heart of automobile industry economists. The drum was a read only device, written to only during the construction of the simulator.

Computer architects will say, wait a minute, how can you possibly perform a conditional branch (the vital heart of all business and scientific computations) on a computer whose instructions arrive in a predetermined unalterable string. The answer was to place the instructions for both possible branches in sequence following one another on the drum. Both sets of computations were wastefully accomplished, with only the desired set being retained. Analysis demonstrated that the inefficiency of this procedure was surprisingly modest for the total range of calculations required for a real time simulator.

The inability to branch ruled out subroutines and interrupts. The code which would normally exist in subroutines was simply replicated in the instruction stream whenever needed; this of course wasted memory space but only for the absurdly inexpensive drum storage. Interrupts were not really required, as all instructor and data inputs and outputs could be programmed into the metronymic sequence of instructions from the drum.

Inputs were dc (usually from high-resolution potentiometers), were fed to a commercially available sample-and-hold, and the outputs of the sample-and-hold were digitized by a commercially available product. The technological sophistication of these devices was surprisingly advanced at that time.

The MK1 emerged with three separate instruction strings on separate portions of the drum (unlike today's elegant hard drives, the old drums had a separate immovable head for each of hundreds of tracks, permitting massive parallel readout if the data obligingly arrived at the right time). The three banks of instructions were utilized at three different repetition rates of 20,5, and 1.25 times per second. The critical motion calculations were assigned to the 20 cycle repetition rate, and other calculations assigned to the two slower families according to the importance of fast response. The 5 per second instructions had four times as many track bands as their neighbor 20 per second tracks, and the 1.25 per second family had four times as many bands as their 5 per second neighbors.

The MK1 also had a separate function computation capability which used an entirely separate arithmetic processor. This portion of the computer spent all its time generating functions of one or more variables, obtaining the variables from, and storing the answers in, the core memory for availability of the main processor when function information was required.

A further independent processor accessed radio navigation data, with all of the voluminous data appearing on the drum just at the time it was required.

The design of the MK1 processor was fairly conventional. Since we had many designers skilled in adapting analog computers to flight simulation, a number of unconventional features were included. The MK1 was a magnitude/sign machine, to avoid exposing our designers to the conventional treatment of negative numbers. When the calculations resulted in an overflow, the MK1 electronically limited the resultant digital representation to the maximum value rather than folding into a negative number. This was exactly analogous to the manner in which operational amplifiers saturated at maximum value and servos ran against their limit stops in analog computation.

We could not afford the cost of the admittedly desirable floating point arithmetic, but fortunately our designers were highly experienced in establishing an acceptable maximum value for the various variables without wasting resolution by frivolously generous overestimation of maximums.

Integrated circuits were not available at the time of the design of the MK1. Individual circuit boards held multiple gates or flip-flops, each card being the rough equivalent of a single IC package five short years later. The MK1 circuit cards used germanium transistors; planar silicon transistors were just emerging but a high cost.

The GP4 followed with many of the design concepts of the MK1. If memory serves me correctly the core memory cycle time was an improved 2 microseconds. The very first integrated logic circuits were now available, although TTL was about two years in the future. We selected Motorola emitter coupled logic purely for economic reasons—the Motorola ECL units were about two dollars apiece while competitive saturated logic, to the limited extent available, was about ten times as **e**xpens**i**ve. We probably missed out on a lot of electronic noise heartache by using emitter coupled logic, as it is wondrously quiet. We used circuit board practices which would have been disastrous from a noise standpoint in the Fast TTL era.

Two or three years after the introductions of the first GP4 entered service the first marginally adequate (for simulator purposes) true general purpose computers became available commercially. The early versions required considerable ingenuity for programmers to cram full functional capability into an affordable computer. The programmers of these general purpose machines faced a substantially different task than that of MK1/GP4 programmers. None of the help from today's high level programming languages was available at that time.

John Hunt was born in Kansas City, MO in 1919.

He received BS degree in Engineering Physics from the University of Kansas in 1947.

He received a MS degree in Electrical Engineering from the Massachusetts Institute of Technology in 1948.

He received a Ph.D. degree in Electrical Engineering from Stanford University in 1960.

He worked in the simulator industry from 1948 until 1984, all of the time with Link and its parent company.

He was Senior Vice President and Technical Director of Link and of General Precision Systems (parent company).