COMPACT HIGH-SPEED DISK RECORDER

Timothy B. Bougan

Science Applications International Corporation
429 S. Tyndall Pkwy, Suite H
Panama City, FL 32404

ABSTRACT

In order to meet the high-speed and high-density recording requirements for today's development and testing environments, we are seeking to merge the cutting edge technologies of tiny, high-performance disk drives and field programmable gate arrays (FPGAs) to build a high-speed compact disk recorder (CHSDR). Specifically, we designed, built, and tested a multi-drive controller that handles the interleaving of data to eight inexpensive IDE drives. These drives and controller comprise a "cell" capable of transferring data at 2.45 MB/sec (4 to 5 times the rate of a single drive). Furthermore, these "cells" can be run in parallel (with a single controller interleaving data between the cells). This "tree" effect multiplies the data rate by the number of cells employed. For example, 8 cells (of 8 drives each) can reach nearly 20 MB/second (sustained) and can be built for less than $30,000. The drives we used are the size of match boxes (the Hewlett Packard KittyHawk). These tiny drives hold 42 megabytes each and can withstand 150 Gs while operating. The cell controller is a Xilinx 4005 FPGA. Furthermore, we've designed a 120 MB/sec RAM FIFO to buffer data entering the system (to account for unavoidable drive seek latencies).

In short, the compact high-speed disk array is a small, relatively low cost recording solution for anyone requiring high data speed but modest data volume. Missile shots, nuclear tests, and other short-term experiments are good examples of such requirements.

KEY WORDS

High-Speed Recording, Disk Arrays, Compact High-Speed Disk Recorder.

INTRODUCTION

Those of us involved in the development and use of avionics and telemetry equipment know that data recording is a key aspect of our profession. As technology has advanced, the recording requirements (in terms of data speed and volume) have increased almost exponentially. Today, multi-track analog and digital recorders are common components in our development and operational centers.

These high speed, high volume recorders (primarily tape drives) are bulky and quite expensive. Competing technologies either can't sustain the data rates (such as disk drives) or have insufficient storage capacity (like semiconductor RAM banks). Disk drive arrays, while promising, are currently quite large and very expensive.

THEORY OF OPERATION

We desired to design a low-cost alternative to high-speed, high-capacity tape drives to integrate with off-the-shelf and custom data acquisition equipment. Our immediate goal was to exceed 12 MB/sec (sustained), provide a data capacity of at least 2 GB, and keep the overall cost of the system in the $20-30,000 range. We furthermore desired to build a small system (preferably one that would occupy 6-12 6U VME slots in a standard VME chassis).

In order to meet these requirements, we made a number of design decisions early on. First, we decided not to "build in" data redundancy beyond each drive's Reed-Solomon error correction capability. This effectively designates our system as "RAID 0". (The reader should be aware, however, that our design allows for future incorporation of various redundancy schemes). We also opted to use very inexpensive "IDE" (Integrated Drive Electronics) drives rather than the faster and more expensive SCSI alternatives. We also found IDE drives available in very tiny footprints. In particular, the Hewlett Packard Kittyhawk Personal Storage Module is a 42 MB IDE drive with a tiny 1.3 inch form factor.

The IDE interface was originally designed for IBM PC-type computers. The intention was to "off-load" as much of the controller overhead as possible from the CPU. Hence, the interface is quite simple. Drives of this type are readily available, inexpensive, and reliable. Their major drawback is speed of operation... most such drives can sustain data rates of only 250 KB/sec to 1 MB/sec.

To keep the controller design simple, and to minimize fan-out and transmission line problems, we decided to limit the size of the array to eight drives. The controller could then be a single gate array (in this case a Xilinx 4005). Each array of eight drives

would comprise a "cell". The cells could, in turn, be connected together in parallel to create an array of drive arrays. Eight cells would comprise the complete system (for a total of sixty-four drives). The system controller could also be a single gate array, connecting the cells in much the same fashion as the cell controller connects the drives in each cell.

Finally, to handle unavoidable latencies due to drive head movement, we designed a 120 MB/sec RAM FIFO (to be positioned at just before the system controller). The FIFO is 64 MB deep and can sustain 60 MB/sec in and 60 MB/sec out simultaneously. The FIFO will fill while the controllers "wait" for the drives to be ready, and then empty as data "bursts" into the sector buffers. See figure 1 for a diagram of the system.

To test the feasibility of the system, we needed only to build and evaluate a single cell of eight drives. By prototyping and developing a baseline for a single cell, we could extrapolate the performance of the complete system. This approach limited our risk and made the overall task much simpler.

Because we used a field programmable gate array as the controller, the physical design of the cell was relatively easy (see figure 2). Most of the work was in the definition of the gate array, which was entered as a generic schematic and compiled with Xilinx's XACT software.

Basically, the controller gate array has to interface a 32-bit data bus with 6 control lines to 8 IDE drives (with a 16 bit data bus). The controller is responsible for initializing the drives, keeping track of the current track, cylinder, and head of the current sector, and managing the handshaking to and from the sector buffers. It also has to respond to "commands" from whatever external device is reading from or writing to the interface. In the absence of a processor, the controller incorporates a fair number of state machines and "solid state subroutines" to execute sequential operations. It also reads and writes to two drives simultaneously (to facilitate the 32 bit data path) and handles all drive "wait" conditions. By the time we were finished, the controller design occupied 80% of a 5000 gate FPGA (a Xilinx 4005).

Once the cell was complete, we had to develop a method to test and evaluate its performance. We did this (rather quickly and efficiently) with the help of another FPGA and an off-the-shelf PC compatible protoboard (with built in I/O decoding). Essentially, we interfaced a Xilinx 4005 with a PC. The FPGA was programmed to write an incrementing pattern as quickly as possible to the cell. It also maintains a free-running counter. The PC can "sample" both counters at any time. The ratio between the two gives us the instantaneous data rate for the cell.

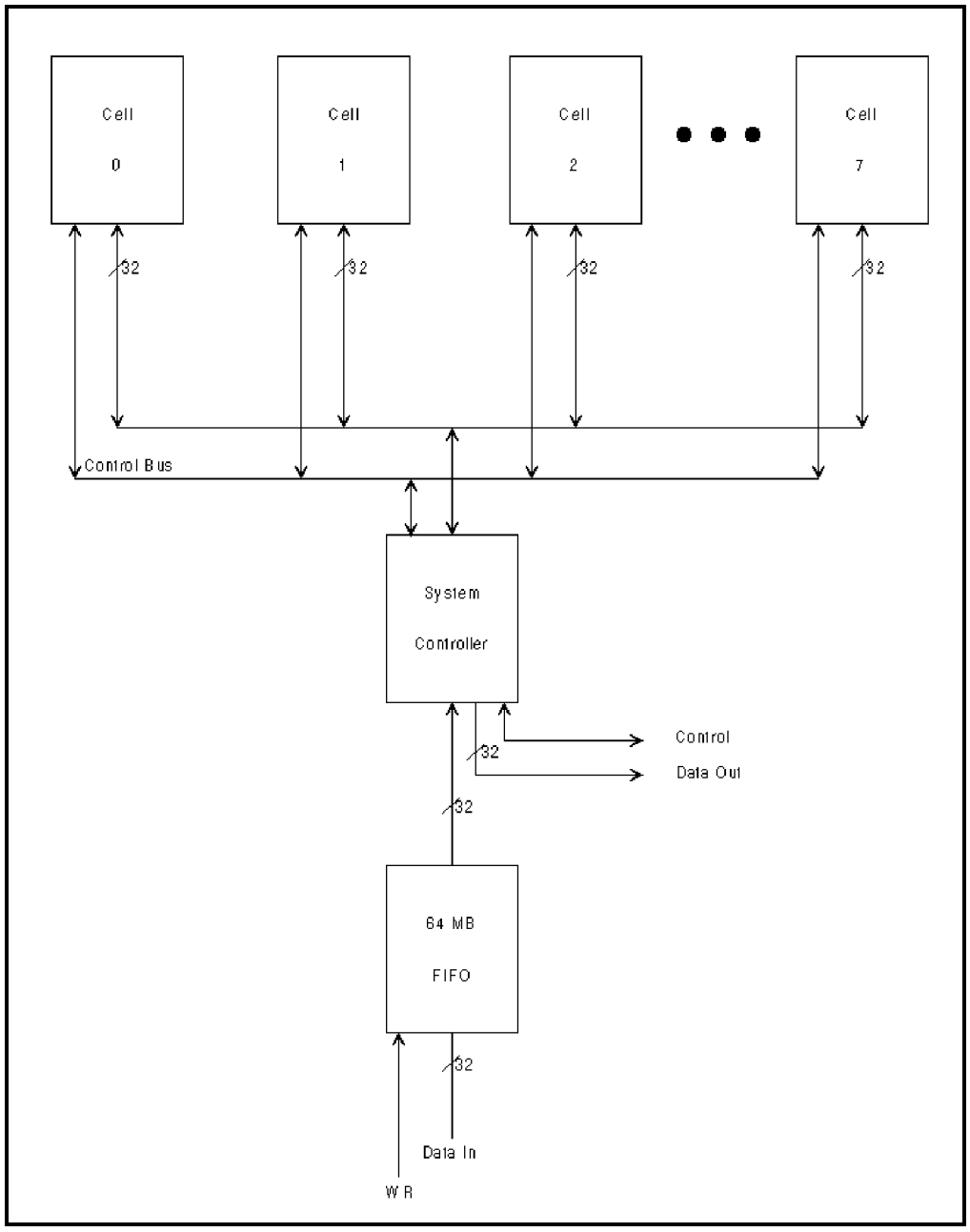

Figure 1

By observing the operation of the cell over its entire data depth, we determined the cell could record at a 2.45 MB/sec sustained rate. According to documents provided by Hewlett Packard, each drive should be able to record about 0.5 MB/sec. So, in effect, we witnessed a factor of five increase (rather than the theoretical maximum of eight). Further discussion with Hewlett Packard shed some light on the discrepancy... that is, the controller overhead between sector writes, the necessity to "empty" all

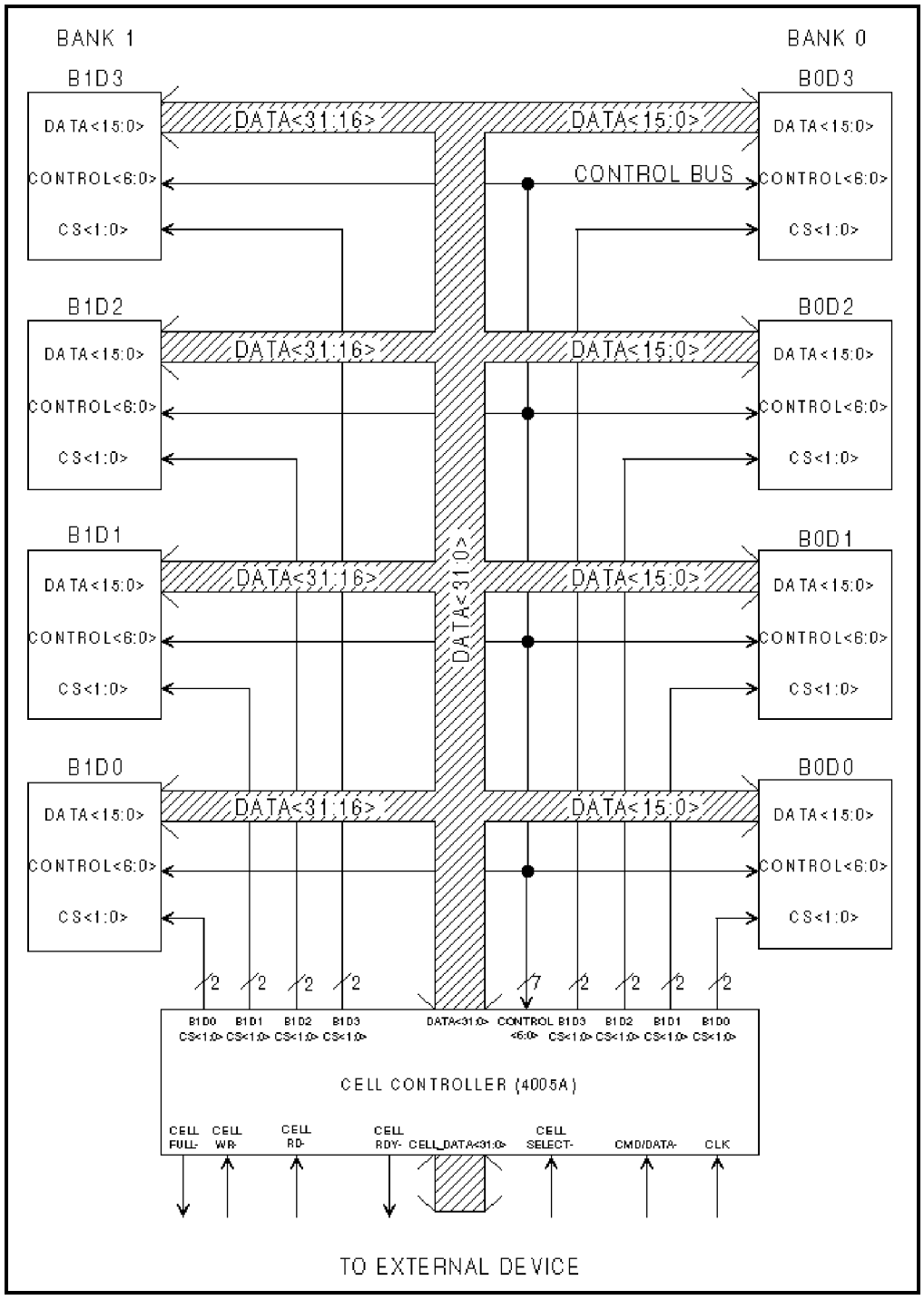

Figure 2

sector buffers before issuing new write commands, and drive delays due to "sparing" would all have negative effects on total performance. ("Sparing" refers to the drive's internal controller moving the head to the extreme edge of the drive to use a "spare" sector in place of a defective one. While it simplifies the drive interface, it requires movement out to and back from the edges of the platter).

From our testing experience, we've concluded that it is possible to build a low cost, compact, high-speed disk recorder based on plentiful and inexpensive IDE drives. If you are willing to use a larger form factor drive (as opposed to the tiny footprint drives we used), you can build a formidable system for surprisingly little money. For example, IDE drives with similar performance to the HP Kittyhawk (but with 425 MB capacity) run about $300 each. The Xilinx 4005 FPGA costs $250. If you through in about $300 for a circuit board and miscellaneous parts, a single cell will cost about $2950. Eight cells, then, would cost $23,600. The RAM FIFO we designed costs about $1,100 (mostly for the SIMMs) and the system controller (another Xilinx 4005) costs $250. All totalled, you would have a 19.6 MB/sec recorder with a 27.2 GB capacity for around $25,000.