

The case of the bad memory chip



Although this event happened years ago, its lesson still remains as one of the more important laws of the art of debugging: Bugs don't disappear with time. It all started one day when my boss called me and another co-worker into his office for an urgent task. It seems that our telephone switches were failing to perform the one crucial operation of a redundant system: to switch activity from one machine to another. The top

brass directed every lab location with such a switch to investigate the problem. The only clue was that the bug began to manifest itself with the latest memory boards.

Everyone thought the culprit was a bad batch of DRAM chips because the older boards had been in the field for years, and this problem had never occurred before. So, my boss assigned me, the hardware guy, to team up with my software buddy to see if we could diagnose the problem.

Considering the number of labs and the number of people working on the case, we never thought that we would be

of much use. But we headed toward the lab anyway. Our first step was to understand how a successful activity switch works by forcing one to occur with the good old memory boards installed in both halves of the system. Sure enough, the system switched correctly several times when we prompted it from the command console. Then, I replaced a memory board with one incorporating the "bad" chips on the standby side, and the switch command failed. When I asked my colleague what could cause the master to refuse to relinquish control to the standby side, he said that the master had most likely found the standby's data

tables to be corrupt, thereby preventing the switch from occurring.

Scanning the memory dump, it looked as if that corruption had occurred, except that the table pointers turned out to be the culprits. Instead of reading FFFF0000, as a good card should, the pointers read FF00FF00. This alternating pattern was a telltale sign of what occurs just after DRAM data powers up. The DRAM seemed to be not initializing. At this point, my colleague assured me that the boot software had been around since the start of the product, and it had always worked. And besides, the other pointer FFFF0000 was correct because it pointed to the proper location. I suggested we manually change the pointer value on the bad card and execute the boot code.

The value remained unchanged, as if we had never initialized it. My buddy insisted it was a hardware problem.

"You must be right," I said, to prevent the usual hardware-versus-software finger-pointing match. "But let's repeat the process on the good card."

"Why?" he asked. "We can see the correct value." At this point I had a hunch, and I urged him to try it out. To his amazement, the boot code failed to correctly initialize the pointers. I then told him that both pointers were simply different patterns of alternating FF and 00 and most likely the result of different internal geometries for both types of DRAM chips. He looked through the code and found the bug that the original DRAM pattern had masked and buried for years.

When we reported our findings, our boss did not believe us and sent us back down to the lab. I can't remember our getting any kind of recognition. It was as if the bug had never occurred. Most likely, some very important person hushed up the whole affair due to the embarrassing nature of the problem. But my buddy and I never forgot how great we felt when we found that bug. **EDN**

Like Pierre, you can share your Tales from the Cube and receive \$200. Contact edn.editor@reedbusiness.com.