D. Gareth Loy
274 Sausalito St.
Corte Madera, California 94925, USA
dgl@GarethLoy.com

Life and Times of the Samson Box

Abstract: Peter Samson designed and built a real-time signal-processing computer for music applications in the 1970s. The Systems Concepts Digital Synthesizer ("Samson Box" for short) was installed at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University in 1977, where it served for over a decade as the principal music generation system. It was an important landmark in the transition from general-purpose computers to real-time systems for music and audio, and helped set the stage for the sea change in the music industry from analog to digital technologies that began in the 1980s and continues at a rapid pace today.

This article focuses on the historical context of the Samson Box, its development, its impact on the culture of CCRMA and the Stanford Artificial Intelligence Laboratory, its use for music research and composition at Stanford, and its role in the transformation of the music and audio industries from analog to digital practices. A list of compositions realized on the Samson Box is included, which shows that from 1978 to its decommissioning in 1992 it was used to create over 100 finished works, many of which were widely performed and were awarded prizes. A companion article provides a detailed architectural review and an interview with Pete Samson.

Introduction

The Systems Concepts Digital Synthesizer was a special-purpose signal-processing computer designed for music applications that was developed in the 1970s by Peter Samson of Systems Concepts, Inc., a computer equipment manufacturer in San Francisco. It helped usher in the modern age of computer music, which this journal especially celebrates. The Samson Box, as it was called, was an important landmark in the transition from general-purpose computers to real-time systems for music and audio, and it helped set the stage for the sea change in the music industry from analog to digital technologies that began in the 1980s and continues at a rapid pace today.

Commissioned in 1975 by John Chowning, John Grey, James ("Andy") Moorer, and Loren Rush, founders of the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University and funded by a grant from the National Endowment for the Arts, it was delivered 9 December 1977 and was placed into service in 1978. CCRMA was formed in June 1975. At first, it was a small project at the Stanford Artificial Intelligence Laboratory (SAIL), which was under the direction of John McCarthy (1927–2011). The SAIL facility was housed in the D.C. Power laboratory building (named for Donald C. Power, an executive at General Telephone and Electronics Corp, which had donated the building to Stanford University), located in the wild foothills behind Stanford University. The Samson Box served for over a decade as the principal music generation system at CCRMA. It was decommissioned 3 April 1992 (see Figure 1).

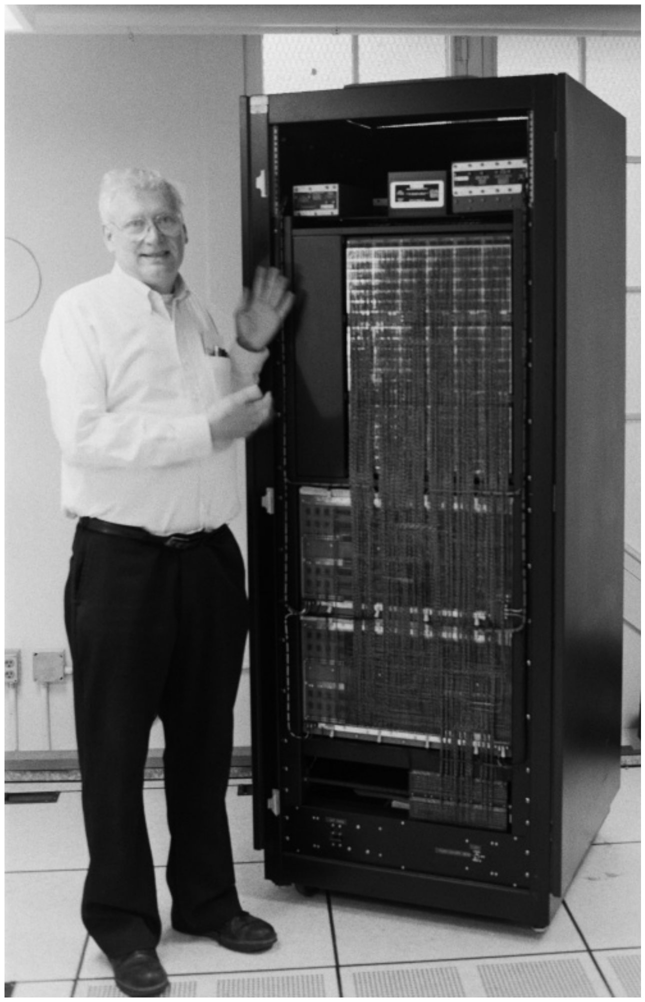
Computer Music Journal, 37:3, pp. 26–48, Fall 2013
doi:10.1162/COMJ_a_00192

In the 1970s, quantitative improvements in the density, speed, and cost of electronic components led to qualitative breakthroughs that allowed researchers to shift signal-processing tasks from slow general-purpose computers to machines specially designed for music and audio. This led eventually to the industrialization and diffusion of digital audio worldwide, bringing about the historic transformation of music that our field has witnessed over the last half-century.

The Samson Box was an important proof-of-concept for, and contributor to, these transformations. The Samson Box performed all then-popular synthesis and signal processing techniques in real time, yet it was sufficiently general-purpose to support some methods discovered after it was designed. Over its lifetime, it facilitated many research projects and compositions, and supported hundreds of users. Some received grants and prizes; others found wide audiences for works realized on the Samson Box. A list of works and composers is found at the end of this article.

Its elegant design still rewards study, but just as interesting are insights that its history provides about our unfolding musical culture. This article focuses on its historical context, its development, its impact on the culture of CCRMA and SAIL, its use for music research and composition at Stanford, and its role in the transformation of the music and audio industries from analog to digital practices.

Figure 1. Pete Samson with the Systems Concepts Digital Synthesizer ("Samson Box"). Photo taken on the occasion of the Box's decommissioning (3 April 1992), in the CCRMA machine room at The Knoll, Stanford University, Stanford, California. (Photograph by Patte Wood.)



A companion article (Loy 2013a) provides a detailed architectural review and an interview with Pete Samson.

Sea Change

There was a time not that long ago when analog audio techniques were all that were known, and the conventional wisdom of the time was that there would be more of the same in the future. Then, at Bell Laboratories in the early 1960s, Max Mathews (1926–2011) demonstrated that the problems of analog audio could be overcome by digital technology (Mathews 1963).

Samson (1980) described the advance this way:

> Digital synthesis offers numerous advantages over analog techniques. Every processing element can be controlled precisely, instantaneously, and repeatably. Digital processing, with its inherent accuracy and stability, can perform the tasks of current analog modules with substantially less noise and distortion, and introduces many new sonic resources, such as time-varying timbre and reverberation.

Of course, in hindsight, we know that digital audio has gone far beyond just overcoming the limitations of analog audio. In his 1963 article in *Science*, Max Mathews wrote, "There are no theoretical limits to the performance of the computer as a source of musical sounds." While this is also true, in hindsight, we also know that "the computer" now provides much more than just a source of musical sounds.

In fact, it is evident that no one in that time period foresaw the extent to which computer technology would touch every aspect of music technology as it does today. Before the mid 1970s, few not actively working in this field believed that computers had much relevance to music. Even if some relevance was granted, it was not clear whether computer music would ever amount to anything more than a laboratory curiosity. Disk drives as big as washing machines and temperamental analog-to-digital and digital-to-analog converters were required. Even in the unlikely event that one had access to such rare and expensive equipment, it was not possible to calculate music of any meaningful sophistication in anything remotely close to real time on even the fastest of these machines. Given that the sounds being generated were often highly experimental in nature, the slow turn-around time meant the pace of the work was glacial. "A time scale of 100—meaning 100 seconds of computational effort for each 1 second of sound—is not uncommon. This is despite the use of powerful computers, and numerous simplifying assumptions and restrictions in the software" (Samson 1980).

Given these circumstances, to suppose that digital signal processing (DSP) might somehow displace a century's worth of analog audio practices—dating back to the work of Alexander Graham Bell and Thomas Edison—stretched credibility, to say the least. But hindsight can show us now what foresight could not tell us then: The way ahead was already in progress.

Mathews' Prophecy

To outgrow the limitations of mainframe computers, digital computers specifically designed to generate music in real time were needed. At a conference on music and technology held in 1970, Max Mathews presciently observed,

> The future will add the digital computer to the equipment of today's electronic studio. . . . In the far future, analog devices may be swept away by more reliable and accurate digital synthesizers constructed from integrated circuits. The result will be real-time digital synthesizers which can be played with all the nuances of present-day performance and all the precision and range of sound quality achieved by present-day digital synthesis. The future grows from the past, and the past is now long enough to reveal at least the next step forward (Mathews 1971).

Mathews' "far future" actually showed up rather quickly. In the mid 1970s, specialized hardware systems for sound synthesis leapfrogged over sluggish software sound synthesis, demonstrating that digital technology was more than up to the challenge. The Samson Box provided a particularly powerful solution to real-time computation of digital audio that enabled enormous strides in computer music research and composition at Stanford.

A History of the Box

Andy Moorer (2012) recalls he met Samson in 1963 at the Massachusetts Institute of Technology (MIT) in the PDP-1 room, and that Samson introduced him to Steve Russell's Spacewar! game and also to software for music generation that could play multi-part Bach fugues in real time with square waves. "The tones were made by taking the top four bits of the accumulator (there was only one) of the PDP-1 and routing each one to an audio amplifier (vacuum-tube) and speaker. Each bit was one voice. You toggled the bits at the desired frequency and you got four square-wave tones." Subsequently, Michael Levitt and Stuart Nelson formed Systems Concepts, and made PDP-10 peripherals. Samson, who knew them from MIT, figures that he "was perhaps the fifth or sixth person" to join the company. Moorer ended up as a SAIL systems programmer in 1968, doing "a lot of different things, computer vision, networking. I programmed the Arpanet IMP [Interface Message Processor]—I was among the first hundred or so programmers to actually implement the early Internet—speech recognition, computer graphics, robotics, language design, and more. It was a great time!" Samson and Moorer stayed in touch, and remained interested in each other's work.

Chowning told me by electronic mail that he arrived at SAIL in 1964, "within months of having read Max's article [in *Science*], when the SAIL computer was still a PDP-1 on [the Stanford main] campus." He continued,

> I was supported by David Poole, an AI Lab systems programmer, who in 1966(!) wrote a music synthesis program native to the PDP-6 (the predecessor of the PDP-10) based on Max's Music IV. (It was in 1966 that SAIL moved to the D.C. Power Lab.) That is the program I used when I discovered FM in 1967. It resulted in the NOSCIL unit generator [an oscillator accepting negative increment or frequency values], that Andy remembers being "in the air" when he arrived in 1968.

Accepted as a graduate student at Stanford in Computer Science in 1972, Moorer told me he gravitated to "learning the 'new' subject of digital signal processing pretty much as it was developing." When later John Grey and Loren Rush arrived at SAIL, they gravitated together to their common interest, and eventually formed CCRMA. Moorer said, "Since none of us were professors, Leland Smith took the role of the faculty sponsor, which was a mixed bag, but ultimately worked out OK. We applied to NSF [National Science Foundation] and NEA [National Endowment for the Arts] for grants. It took many months to write the proposals and many more months to get the deal."

Until their ship came in, they were an impoverished research project at SAIL, scavenging for spare instruction cycles on the PDP-10 by working nights and weekends. Chowning told me "we were the only ongoing project at SAIL that was not supported, rather well, by ARPA [Advanced Research Projects Agency] or NSF or some other government agency." Moorer (2012) said,

> As for the lab politics, recall that the music group were parasites on the system, which was funded by ARPA. . . . We were tolerated, and very modestly supported (spare change from the lab budget—all basically on the good will of Les Earnest, executive director of SAIL).

According to Chowning, ARPA had wanted CCRMA kicked off SAIL because of the inordinate number of machine cycles they consumed doing software sound synthesis. But when Les Earnest redesigned his famous Finger program (Harrenstien 1977) to include usage by time of day, it quickly became evident that CCRMA used only nighttime and weekend cycles that would otherwise have been wasted, and the issue subsided. Moorer goes on,

> I designed and hand built a set of 16-bit D/A [digital-to-analog] converters for the PDP-10 (four channels). We could only hit about 25 kHz sampling rate when using all four channels. It also had two channels of 14-bit A/D converters, which were as precise as you could get at that time. John Grey and I started working together on our dissertations—I would build analysis and DSP programs for John, as he needed them. That was also great fun and very productive. I published a mess of papers in that era. Chowning became convinced that we had to get the music computation off the PDP-10—that there just wasn't enough horsepower there to support what we wanted to do—and that we didn't see those big computers getting powerful particularly rapidly. Pete always had an interest in computer music and often came to our concerts and talks. Pete, on his own, decided to start the project to design the Box. We had talked a bit about it as he was developing it, but it was largely his idea from start to finish. We only made one relatively minor change to the design (adding ramping to the oscillator frequencies). Everything else was all Pete. He announced the device at the first computer music conference [1974], which was held [at Michigan State University] in [East] Lansing, Michigan, where Dave Wessel was a professor.

Shortly thereafter, in early 1975, CCRMA got its funding from the NSF and the NEA, and the principals then placed an order for one Samson Box on 12 January 1976, to be delivered in 110 days' time.

As it turned out, Max's prophecy was busy coming true in several places simultaneously. Hal Alles, of Bell Labs, showed a prototype of his digital synthesizer at the same conference (Alles 1977, 1979). Also in this time frame, Jon Appleton, Sydney Alonso, and Cameron Jones were developing the prototype of the Dartmouth Digital Synthesizer (Alonso, Appleton, and Jones 1976), which would later become the Synclavier. Peppino DiGiunio was developing his 4X series of synthesizers at the Institut de Recherche et de Coordination Acoustique/Musique (IRCAM). At Stanford, F. Richard ("Dick") Moore was also starting to develop another synthesizer prototype. The FRMbox, as it was called, was completed and operational at SAIL a few months before the Samson Box arrived (Moore 1977, 1985). Moore had worked at Bell Laboratories with Mathews on the GROOVE system in the late 1960s and early 1970s, and had come to Stanford for graduate work (Mathews and Moore 1970). During a conversation with Chowning in 2012, he told me that Mathews had strongly promoted the idea of building out Moore's FRMbox prototype to its full potential instead of commissioning the Samson Box, especially in view of the numerous delays in the Box's delivery. Originally scheduled for delivery 1 March 1976, its arrival was postponed numerous times by Systems Concepts, causing Stanford at one point to threaten breach of contract. Rather than taking 110 days to build, it took almost two years. However, Chowning also told me that he was persuaded by Moorer, whose technical credentials he trusted absolutely, to stick to their original plan and wait for the Samson Box. Moorer told me,

> Really, Pete's device was the only thing on the horizon that promised the appropriate horsepower. I looked at commercial DSP systems at that time as well. We had one in-house that was bought for speech research—the SPS-41. It was really designed for doing FFTs [fast Fourier transforms], and did less well on anything else. There were a few others (Culler-Harrison was one). They all had various problems making it difficult to do music, such as the lack of reverberation memory, not enough precision, and so on. There was really nothing that satisfied the needs like Pete's device did. Dick Moore also made a digital music synthesizer, but it was not a terribly powerful device—more of a demo device. We had heard Hal Alles's device, and noted that it had some terrible audio issues, such as that audio envelopes were calculated at 1/8th the sampling rate, so that attacks had noticeable 'chirps'—a classic rookie mistake that is repeated again and again, especially from people that think they know something from analog synthesizers. In analog, you just update the control voltages, say, 30 times a second and all is well. Of course, the analog circuitry smooths out all the glitches for you. No such luck in digital.

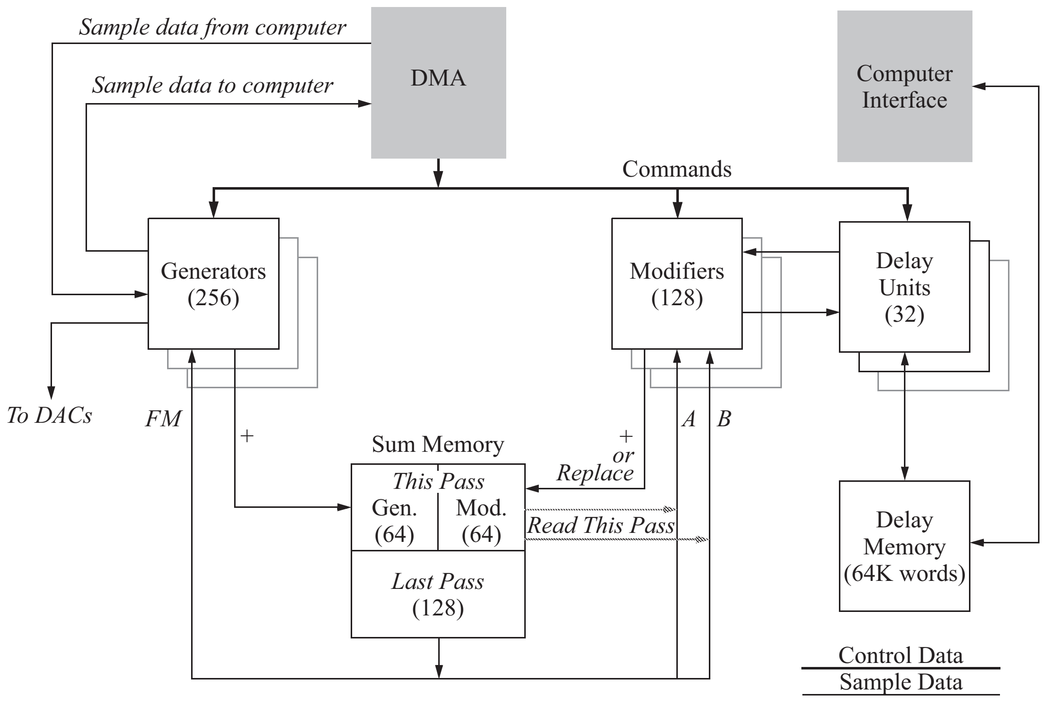
Designing the Box

At Systems Concepts, Samson had been surveying the literature and holding discussions with practitioners since about 1972, and had decided that the design would be organized by its repertoire of synthesis techniques. But it would also need to provide flexibility necessary to permit experimentation and development of new techniques. Just how big would it have to be? Samson decided that he would aim for a real-time capability on the order of a string quartet, both by additive and subtractive methods. A larger design ("symphonic") would be nice, but seemed infeasible with extant technology, and a smaller one ("violin") would probably be too constraining on users trying to prepare something larger-scale. An external host computer would control the synthesizer, sending it a musical score in the form of a stream of commands (Samson 1985). Samson envisioned that it could also be controlled interactively by use of real-time input devices for live performance, although there were some difficulties here (detailed in the following).

Synthesis techniques included additive, subtractive, and frequency modulation (FM) synthesis. Processing effects included filtering, mixing, reverberation, random noise generation, and miscellaneous signal testing functions. It could also read, process, and write streaming multichannel audio waveform data for sampling synthesis and *musique concrète*. It had a large delay memory for filtering, table storage, and reverberation. Four 14-bit digital-to-analog converters (upgraded to 16-bit DACs in 1984) provided state-of-the-art quadraphonic analog output for three-dimensional audio spatialization. The Samson Box was able to perform all these functions simultaneously, in real time, based on a design including 256 signal generators, 128 signal modifiers, 32 delay units for reverberation, about 64,000 words of delay memory, and 256 sum memory locations for combining signals. It used about 2,500 TTL integrated circuits, performed 20 million multiplications per second across 20-bit data paths, implemented a continuously variable sample rate, and supported up to 256 separate channels of read-data and write-data, and up to 16 channels of analog I/O (see Figure 2).

Arithmetic and data-path precision was designed in light of musical aesthetic considerations, as well as cutting-edge psychoacoustic research. For example, oscillator frequency precision was almost three magnitudes below the smallest just noticeable difference (JND) of pitch (usually given as 3 Hz for frequencies below 500 Hz). This fine frequency resolution would not only allow seamless resolution of pitch, but it would also allow beat frequencies between oscillators of musically useful duration. For another example, Samson faced a tradeoff between

Figure 2. Samson Box architecture. The Samson Box elements included generators, modifiers, sum memory, delay units, and delay memory. The 256 generators produced various band-limited and non-band-limited waveforms; applied linear or exponential envelopes; performed frequency modulation; swept frequency; read data from computer memory; and wrote data to computer memory or to digital-to-analog converters. The 128 modifiers performed filtering, resonance, antiresonance, two- and four-quadrant multiplication, mixing, clipping, sample-and-hold, and uniform noise. Modifiers interfaced to delay units, which implemented table lookup or delay lines. Each sum memory location accumulated all outputs directed to it on each pass. Generators wrote to generator-side sum memory, modifiers to modifier-side sum memory. Generators and modifiers wrote to this-pass sum memory and read from last-pass sum memory, but modifiers could also read from this-pass and could replace contents on writing sum memory.



the substantial cost of memory and oscillator quantization noise. With a sine wave table address size of 13 bits, the oscillators achieved a respectable 76.5 dB signal-to-noise ratio. As amplitude decreased, so did noise, because it was "signal with noise" (Blesser and Kates 1978). Oscillator glissando effects could be instantaneous or as slow as 9.3 Hz/sec at 20 kHz. Frequency sweep rate was also well below a JND of pitch so that glissandi sounded continuous.

Samson (1980) wrote about fabricating the Box as follows.

> Engineering of the Synthesizer made heavy use of computer-aided design techniques developed on the Systems Concepts in-house computing facility. All schematic drawings for the system were done by machine, and machine-checked for consistency. For the printed-circuit logic boards, computer programs did the parts placement, resulting in assembly drawings; and then laid out the complete wiring of each card. Results were full-size artwork and a paper tape to run the numeric-controlled drill making the printed circuit boards. For those portions of the Synthesizer on wire-wrap panels, the computer system produced data files for fully automatic machine wrapping: This avoids a significant source of error and ensures the best possible workmanship. As another part of the design automation process, various items of metalwork—brackets, chassis, etc.—were designed with the aid of computer programs which created paper tapes to run the actual metalworking machines.

Diagnostics testing were a central feature of the design. Samson added, "More than 10% of the hardware in the Synthesizer is strictly for diagnostic purposes. In support of this hardware, some 50,000 words of diagnostic software have been written."

Chance Favors a Prepared Mind

I can attest that, outside of a small handful of researchers, digital audio was almost completely under the radar in those times. Most academics and professionals were in the dark, and the public at large had no clue. No doubt we each have a story of how we came to computer music; mine is inextricably bound up with the creation of the Samson Box and its software, and I offer it to help get the flavor of the times. In 1974 I was a part-time lecturer in charge of a small classic electronic music laboratory, built by Herbert Bielawa at San Francisco State University (SFSU), with a Buchla 100 synthesizer and a collection of audio gadgets of our own devising. I'd read Lejaren Hiller and Leonard Isaacson's book *Experimental Music* (Hiller and Issacson 1959), had heard the *Illiac Suite*, and was familiar with Max Mathews's *Technology of Computer Music* (Mathews et al. 1969). Although these results were interesting, they seemed aesthetically primitive—and in any event, remote and unobtainable. I dismissed their meager results as a consequence of the unwieldy nature of computers. (And I was not incorrect.)

But one day in 1974 a colleague invited me to attend a public lecture at SAIL, where I met John Chowning and heard his charismatic work. The sophistication and advancement of his music, coupled with his penetrating insights on digital technology, lit me on fire. I saw the promise of his approach at once, and applied immediately to do doctoral study at Stanford with him. I approached Andy Moorer, then a systems programmer at SAIL, and basically begged him to give me access. He skillfully tested my resolve, handing me a two-foot thick stack of computer manuals, including such digital esoterica as the PDP-10 UUO Manual (Unimplemented User Operations, cf. Frost 1975), cheerily saying, "Here you go. Read these, and when you have understood them, give me a call back, OK?" I took it home, heartily wolfed it down (comprehending a little), and called him back two weeks later. My diligence was rewarded with a guest account at one of the most advanced computer laboratories on the planet, rubbing shoulders with the titans of artificial intelligence (AI). Believing I'd died and gone to heaven, I threw myself into the work, driving down from San Francisco on weekends, working insane hours to catch up with the technology.

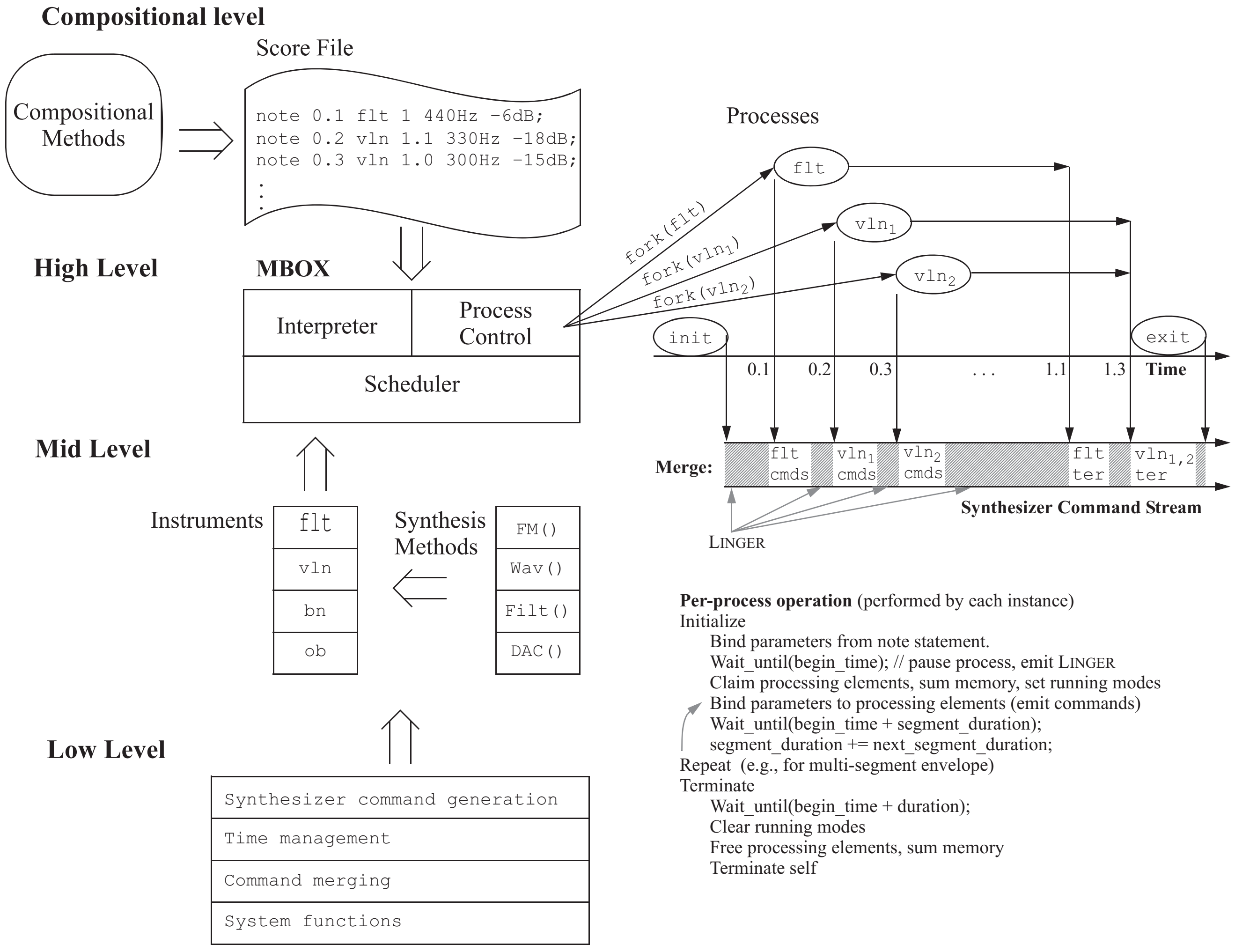
A few months later the phone rang in my office at SFSU, and the caller introduced himself as Pete Samson. He told me Chowning had given him my name, and that he wanted to come have a "look-see" at my studio to get a feel for analog electronic music practices. When he arrived, he required mere seconds of orientation before he more or less thoroughly understood what was possible. When I asked him his business, he told me he was developing a real-time digital synthesizer for Chowning, and did I want to come take a look at his studio, since I'd shown him mine? So the next week I visited Systems Concepts, which had its offices above a restaurant in San Francisco's SoMa (South of Market Street) district, and listened to Samson describe his synthesizer design.

I heard him as though he were speaking a foreign language, which was shocking, given that I considered myself to be a serious practitioner of electronic music. But I plainly understood little of what he said. Eventually, I would understand much more, but I would also come to appreciate why it had sounded so foreign: Digital audio required a fundamental reimagining of both the art and the science of sound. It would take much more than a little getting used to. Access to information was clearly key. I redoubled my efforts to study at Stanford, and was admitted to graduate study in the fall of 1975.

Prototyping the Software

Only months later I was chafing at the glacial turn-around time of software synthesis on SAIL's PDP-10 when I heard the news that the Samson Box had been ordered. Given my background in interactive analog performance, I was strongly attracted to the idea of doing improvised computer

Figure 3. Synthesizer command software architecture. The MBOX compiler incorporated three layers of synthesizer command software, as follows. Lower level: synthesizer command generation, time management, and command merging. Mid-level: user-provided (or library-provided) instrument definitions and support functions. Higher level: instrument processes and score interpreter. The output of this process was a file of Box commands stored on disk that would subsequently be transferred to the Box in real time to realize the musical performance.



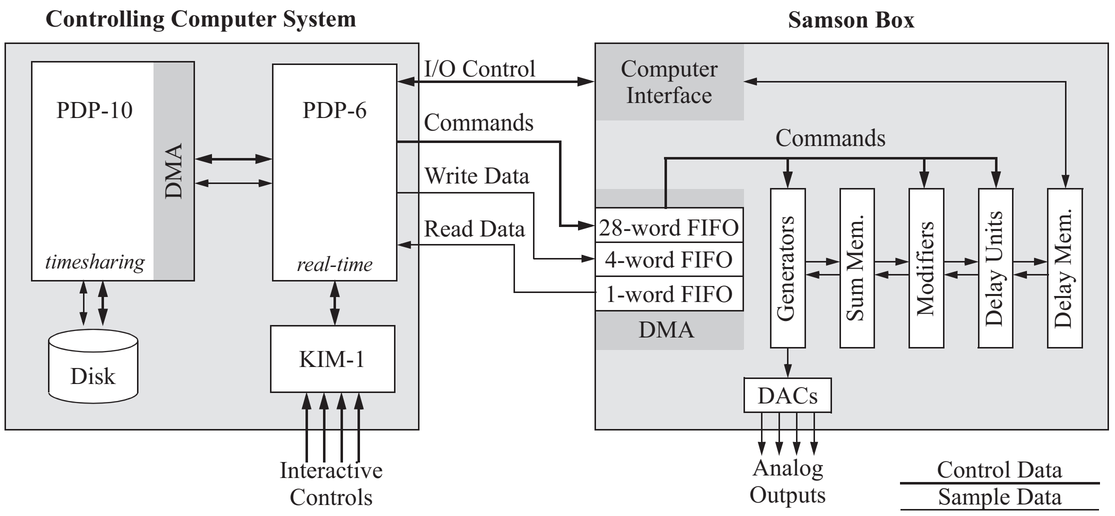
Per-process operation (performed by each instance)

```
Initialize
    Bind parameters from note statement.
    Wait_until(begin_time); // pause process, emit LINGER
    Claim processing elements, sum memory, set running modes
    Bind parameters to processing elements (emit commands)
    Wait_until(begin_time + segment_duration);
    segment_duration += next_segment_duration;
Repeat  (e.g., for multi-segment envelope)
Terminate
    Wait_until(begin_time + duration);
    Clear running modes
    Free processing elements, sum memory
    Terminate self
```

music in real time. Though I knew the Box might take years to show up—if it showed up at all—I decided to throw myself into preparing for it. But the only available documentation was an 18-page specification (Samson 1974) that we referred to as the "crib sheet" because it was so abstruse. I approached Moorer with a proposal: If he would help me comprehend it, I would write a tutorial introduction for the others (see Loy 1977 for the result of this bargain). He accepted, and, over time, inducted me into the Box's many digital mysteries. I eventually joined a small team of Box software developers: Andy Moorer and Mark Kahrs provided overall architectural guidance; Moorer provided systems programming; Kahrs wrote a command assembler, disassembler (DAB), and a library of code for resource management; Ken Shoemake further developed the command assembler; the compiler (Loy 1981) was my contribution.

Moorer and Kahrs designed the synthesizer control system in three layers, as follows. Lower level: synthesizer command generation, time management, and command merging; mid-level: user-provided (or library-provided) instrument definitions and support functions; higher level: instrument processes and score interpreter. The compiler integrated these three layers, as described below (see Figure 3). Later, David Jaffe extended Kahrs's DAB into a

Figure 4. Samson Box control. In its original configuration, a PDP-10 provided time-sharing services and fed Samson Box command and audio data in real time to a PDP-6 on demand from the Samson Box. A KIM-1 microprocessor serialized interactive inputs (clavier keyboard and knobs) to the PDP-6. After 1983, the PDP-10 and PDP-6 were replaced by a Foonly computer system that provided both time-sharing and interactive real-time services.



command debugger (Jaffe 1983). Subsequent software, such as Bill Schottstaedt's Pla compiler for the Box (Schottstaedt 1983), and later interactive performance systems, shared essentially the same architecture.

The Day Finally Arrived. . .

At 9:00 AM on 9 December 1977, Samson wheeled a tall green metal box into the machine room at SAIL. He connected it to a disused PDP-6 computer, booted the "Six" from its console switches, and loaded diagnostic software. He ferociously pounded the keys of an ancient Model 33 Teletype, then jabbed his finger at an oscilloscope trace of a single sine wave. That, he told me, was a continuity test, the result of summing all of the Box's generators through all of its other components. At 5:00 PM, the acceptance agreement having been signed by both parties, he packed up and went back to San Francisco. Except for a few relatively minor hardware bugs we found over the next several months, it worked straightaway. The rest would be up to us.

Hooking It Up

Figure 4 shows the initial Samson Box setup at SAIL. The PDP-6 was a hand-me-down computer loaned to CCRMA by SAIL for hosting real-time control of the Samson Box. Moorer designed a direct memory access (DMA) controller to connect the PDP-6 to the PDP-10, which was the main time-sharing computer at SAIL. This required wire-wrapping an interface between the "Six" and the "Ten," which we put together one long weekend with a couple of six-packs of beer and wire-wrap guns. Moorer then wrote an application for the "Ten" to fetch synthesizer command files from the disk and relay them to the Box via the "Six."

We also built out the analog audio system to connect to new listening spaces and interoperate with the existing audio systems. It took us about a year to get all the software and hardware systems working well, and to find and flush out the few bugs in the Box and the many bugs in our software.

The Box at SAIL

At first, the Samson Box was attached to a commonly shared sound system in the rooms CCRMA used at SAIL, so that everyone working near connected speakers heard every sound. This had the great benefit of keeping us all apprised of our progress, and fostered cross-fertilization of research and a strong sense of community: if you heard something you liked, you went and talked to its creator. The practice did have occasional downsides, however. One day I was jolted by an immense blast of white noise that erupted from a nearby loudspeaker. My cup flew involuntarily out of my hand, spraying coffee across the room while Julius Smith fumbled frantically for the volume knob. He had been experimenting with very narrow band-pass filters—hence very quiet, hence the volume was all the way up—when they suddenly, and without warning, went unstable. It turned out the Box's filters were designed without saturating overflow or rounding control (see Loy 2013a). It was a vivid lesson to us all.

In fact, CCRMA grew up as an extension of the open, casual culture of SAIL, fostered by its leonine director, John McCarthy, and carried out with great civility and tact by executive director Les Earnest. The welcoming environment of SAIL was exemplified by the login banner on each terminal, which read, "Take Me I'm Yours!" This we did, lustily. Though there were some private offices, they were not actually very private: no one kept his or her door closed, and there were no locks. In fact, there was less than no privacy. Besides listening to what everyone did on the common sound system, there was also a custom video switch that connected all graphics terminals (Earnest 1972). An escape sequence on each keyboard allowed anyone to see anyone else's screen (even the director's) to check out what they were doing. And (if you were brave) you could then even interfere directly by typing on other people's screens, and they on yours. Sometimes we'd have heated discussions this way about digital etiquette and other subjects.

Which is to say, no, we didn't all just get along. There were intense, near-constant disputes over resources, even though the Samson Box largely relieved CCRMA's computational burden on SAIL. Despite the intensely diverse research interests that shared the same facilities, all SAIL research was conducted in close, intimate contact. It was the perfect hothouse for incubating creativity in every sphere. This atmosphere worked to the advantage of the CCRMA people because in those times most of us depended on what the SAIL research community knew about the nascent field of computer science, and there was really nowhere else to get that knowledge. What better way to absorb it than to rub elbows (and offices, keyboards, screens, machine cycles, and even volleyballs during after-hours games) with the "army of generals" that was the SAIL research community?

The Software

For the most part, we used the Samson Box as a special-purpose real-time batch processor of precompiled musical scores. We had hoped to use it interactively, but at first problems in its design interfered. In any event, batch processing in real time was what we needed most at that time, and it took longer for its real-time capability to be exploited for reasons discussed in the following.

My compiler, MUSBOX (MBOX for short; see Loy 1981), was designed to support real-time batch processing on the Samson Box. MBOX was written in SAIL (the Stanford Artificial Intelligence Language), developed by Swinehart and Sproull (1970) at the Stanford AI Laboratory. SAIL was a variant of ALGOL-60, including extensions such as a lightweight process discipline.

I adapted the classic "note statement" model of Music *N* software synthesis programs for score input to MBOX. This allowed composers to use existing score generation tools such as Leland Smith's SCORE program (Smith 1972) or Bill Schottstaedt's PLA compositional language (Schottstaedt 1983). Inputs to MBOX also included instrument definitions and user-provided procedures written in SAIL, time functions, and files of audio sample data. The compiler's output was a file of Box commands stored on disk that would subsequently be transferred to the Box in real time. An overview of the software architecture for constructing synthesizer command streams is shown in Figure 3.

The compiler read note statements from a text file; parsed the instrument name, timing information, and parameters from each note statement; forked an instance of the instrument named in each note statement using the SAIL lightweight-process discipline; and loaded timing and synthesis parameters into the instantiated instrument variables. The instruments themselves were special procedures written in the SAIL programming language. MBOX executed a time-based hierarchical scheduler that received requests from the instrument instances to run at particular times. When executed, instances claimed synthesizer processing resources from the low-level routines, then emitted synthesizer commands as appropriate to generate or process sounds, according to each instrument's design. If an instrument needed to execute at multiple time points—for example, to implement an attack-decay-sustain-release envelope—it would cause itself to be rescheduled as needed to generate its time-ordered list of synthesizer commands. Controlling the scheduler and the lightweight-process mechanism, MBOX automatically performed a time-ordered merge of each instance's output, and wrote the combined synthesizer command stream to a file (see Figure 3).

MBOX instruments could monitor the progress of other instruments using the lightweight process discipline, allowing instruments to track and modify each other's operations collaboratively. For example, a conductor instrument could describe a tempo map, or denote a path in space for other sounds to follow. This could be extended hierarchically. Other instruments could listen to this information and plot their tempo and/or position in space accordingly.

After the user had compiled a score to a disk file using MBOX, he or she would execute a separate program on the PDP-10, written by Andy Moorer, to direct the command stream (as well as any time functions and audio files needed) to the Box. This program would buffer blocks of Box commands (and time functions and audio data, if needed) to the PDP-6 computer to which it was connected. A real-time executive in the PDP-6 (also written by Moorer) fed the commands to the Box, which then played them in real time.

The Music

The first musical production we did on the Samson Box was to remaster Chowning's 1972 piece *Turenas* using its original score file. This step validated the architecture of the Box and proved that our software and hardware system worked for realistically sized projects. Chowning's records date this event to 16 December 1978. That means it took us a year from when the Box was delivered to get everything running. After that, the Box became the primary workhorse of digital synthesis at CCRMA and remained so for more than a decade. A list of works created on it is given at the end of this article.

Other Digital Synthesizer Designs of the Time

In 1977 there were a number of research machines and commercial designs in the making, some of which would eventually find their way to market. We can get a sense of where the field was headed and how the Samson Box stacked up by comparing the various tradeoffs of these designs (see also Moorer 1989 and Kahrs 2002).

A prototype of the Synclavier, the Dartmouth Digital Synthesizer, by New England Digital Corp. (NEDCO) was developed in 1976 (Alonso, Appleton, and Jones 1976); the first commercial version of the Synclavier was available in 1978. Up to four banks of sixteen oscillators could be paired to provide up to 32 simple FM instruments. No other synthesis techniques or reverberation were provided. It had no real-time digital audio output (analog out only). The Synclavier was controlled by a 16-bit minicomputer of NEDCO's own design, called ABLE. This was integrated directly with the synthesis hardware. Eight-inch floppy disk drives held software and a 25-MB moving-head disk held up to 3 minutes of digital audio, or score sequences of much greater length. A basic system in 1978 cost US$ 13,000. Fully built out, a Synclavier cost about as much as a Samson Box. The Synclavier price included the ABLE CPU, however, and it was shipped with software; the Samson Box came with neither CPU nor software (although Stanford's software was available free of charge—in the event you happened to have a PDP-10 available).

The Coupland Digital Music Synthesizer was a 16-voice polyphonic real-time instrument with a full organ-type keyboard, based on a TI9900 minicomputer, but the user controlled the system only through a fixed-function console that was not programmable. Though advertised, it was never released commercially (*Computer Music Journal* 1978).

The Bell Digital Synthesizer was the machine designed by Hal Alles (1979) mentioned by Moorer, above. Though it was strictly a research machine at Bell Labs, the General Development System (GDS) by Music Technology, Inc., was a reduced-feature version of this machine controlled by a Zilog Z80 microcomputer, scheduled for release in 1979. It implemented additive synthesis. Besides the limitations noted for Alles's architecture above, the GDS version did not scale beyond 32 oscillators, and was limited to 12-tone tempered tuning. Crumar, an Italian synthesizer manufacturer, commercialized it in the early 1980s.

The Fairlight CMI was, like the Synclavier, Coupland, and Bell synthesizers, a performance-oriented machine. It came equipped with a musical keyboard, computer keyboard, light pen, and bitmap graphic screen. Designed by Peter Vogel and Kim Ryrie, and based on a Motorola 68000 system design by Tony Furse, the CMI was introduced in 1979, costing about US$ 36,000. It stemmed from a 1976 prototype called the Quasar M8, a waveform modeling system using additive synthesis, that led to the development of the Fairlight CMI I as a sampling synthesizer, having eight voices of 8-bit samples each. It included sequencing ("Page R") and waveform editing software. It was soon followed by a host of other samplers from competitors including Ensoniq, E-mu, Roland, and others (Doornbusch 2009).

The DMX-1000, designed by Dean Wallraff and manufactured by Digital Music Systems, was a signal processor based on bit-slice microprocessors. This attractive design required a controlling computer and four to six units, costing US$ 8,000 each, to achieve an orchestra of 32 FM instruments. It came with software, the Music-1000 programming language, and example programs for FM synthesis and filtering. Like the Samson Box, it was a special-purpose computer optimized for signal processing, to be used as a coprocessor in a larger system. In production in 1979, it was mostly purchased by academic laboratories (Wallraff 1979).

Other non-commercial research machines in the years 1976–1979 included the FRMbox (Moore 1977); the SSSP Digital Synthesizer (Buxton et al. 1978); Iannis Xenakis's UPIC system in Paris; the 4A, 4B, and 4C machines made by DiGiunio at IRCAM (Moorer et al. 1979); and the Syter system at the Groupe de Récherches Musicales (GRM). The FRMbox was a programmable bus architecture framework with a controller that could communicate data and operation directives between collections of modules. So long as the modules obeyed the FRMbox bus protocol, they could perform any function—however, these modules would have to be designed and fabricated. As I recall, Moore implemented only an oscillator and envelope module. The SSSP synthesizer had a sophisticated graphical user interface and could perform VOSIM synthesis, additive synthesis, FM, and waveshaping in real time. It was controlled by an LSI-11 computer. It was used as a platform for research in user-interface design and for real-time music performance and composition (Fedorkow, Buxton, and Smith 1978). The 4C machine was a single-board DSP controlled by a PDP-11, implementing 64 oscillators, 32 envelopes, and various timing and control functions (Moorer 1989).

Some synthesis hardware systems, like the DMX-1000, 4C, FRMbox, and Samson Box, were designed as coprocessors. The rest were integrated single-use, interactive instruments, some with knobs, musical keyboards, and consoles. The keyboard-oriented machines such as the Synclavier and Fairlight mostly competed for placement with major recording studios, academic laboratories, or pop/rock stars such as Peter Gabriel (Fairlight) and Frank Zappa (Synclavier).

The market for these instruments never recovered from the introduction in the 1980s of mass-produced MIDI synthesizers such as the E-mu Emulator (1982), Yamaha DX-7 (1983), Ensoniq Mirage (1985), Korg M1 (1988), and Roland D-50 (1987). Just as low-cost personal computers replaced mainframes, so too did low-cost digital music instruments replace these first-generation hardware synthesizers.

The same commercial constraints also affected Systems Concepts: There was never another Samson Box sold. At around US$ 100,000, it was by far the most expensive digital synthesizer, and the software we developed for it was not portable because it was written for a computer architecture that soon became obsolete.

Critique

In an online article critiquing the Box, Julius Smith (2005) wrote, "The Samson Box was an elegant implementation of nearly all known, desirable unit-generators in hardware form, and sound synthesis was sped up by three orders of magnitude in many cases [compared to software sound synthesis]. It was a clear success."

The Samson Box largely delivered the synthesis design goals that Samson had initially outlined. One aim was implementation of all major synthesis techniques known at the time, prominently including additive and subtractive synthesis.

Additive synthesis was robustly implemented, with many unit generators (256), a flexible envelope architecture, and enough ways to connect them to realize dozens of voices in real time. Expressive articulation of individual harmonics was straightforward, implemented by sending updates via the command stream. Mike McNabb told me he used additive synthesis extensively in his Samson Box music. He said, "Pete once told me after hearing my piece *Love in the Asylum* that 'this [use of additive synthesis] is what the box was made for.'"

The Box supported Chowning-style FM synthesis with arbitrary multi-modulator and multi-carrier topologies. Subtractive synthesis was supported by the combination of digital filtering and the sum-of-cosines mode for oscillators.

The delay memory allowed for development of sophisticated reverberation systems. I was able to make a straightforward implementation of the reverberator design proposed by Moorer (1979), which I featured prominently at the climax of my thesis composition, *Nekyia*. Jaffe and Smith (1983) also used delay memory to realize plucked-string synthesis on the Box.

But from a design perspective, the Box's limitations were just as interesting as its capabilities, as they revealed to us a great deal of useful information with which to design better solutions. In the following sections, we consider several of the most prominent issues that its design presented.

Interactivity

A year before the Samson Box arrived, Andy Moorer, Robert Poor, and I had prepared a set of interactive controls for the Samson Box consisting of a clavier keyboard (a 61-note musical keyboard) and a set of knobs and switches. Moorer hooked them up to a KIM-1 single board microcomputer, which he programmed to transmit serialized control information to the PDP-6. But, to my knowledge, these controls were only used—and then but briefly—to control the FRMbox during Dick Moore's PhD thesis defense. Some architectural difficulties with the Box prevented its use for interactive real-time applications: The way the Box was designed to receive commands from the controlling computer effectively prevented interactive control.

The first Box limitation was that, by default, the Box continuously read commands via a DMA channel from the PDP-6's memory to set up a patch or change parameter values. If the score being performed required no new commands for a while, a "linger" command inserted into the command stream would cause the Box to suspend reading additional commands for a duration specified by the linger command; meanwhile, the Box continued running, generating samples. After the linger command expired, the Box would resume reading the command stream to update the state of the Box. This arrangement worked well for preprogrammed scores, but there was no way to interrupt or abort a linger command in progress, which might last for as much as a minute. This design prevented the Box from being able to respond to impromptu interactive inputs.

A work-around was to not use linger commands—that is, to disable timing within the command stream entirely, and depend upon the PDP-6 for all timing. In this scenario, the PDP-6 would receive interactive performance events from users and "immediately" assemble and send commands to the Box to synthesize the corresponding musical sounds. But "immediately" on a PDP-6 was much slower and less deterministic than on the Box. An interrupt service routine on the PDP-6 might take hundreds of microseconds to execute, depending on its complexity and whether any tasks with higher priority interfered. Stacking interrupts from multiple inputs could extend these delays to audible proportions. By contrast, the Box could process a new parameter update every 195 nanoseconds. Clearly, it would have been better if the finer timing resolution of the Box could have controlled timing on the PDP-6, rather than the other way around.

But the Box's second limitation made that solution impossible: Originally, the Box did not have a way to communicate sample-level timing or synthesis event information back to the controlling computer.

Later on, in 1984, I'm told that Samson retrofitted a way for the expiry of an envelope segment to trigger an interrupt to the controlling computer. This ultimately made it possible for the Box to provide sample-level synchronization to the controlling computer, and some serious—and not so serious—real-time applications eventually followed, described below.

By this time the controlling computer was a Foonly F4, however, not a PDP-6. Whereas the "Six" had been a dedicated real-time CPU for the Box, the Foonly also ran the time-sharing system for CCRMA. Although the Foonly had plenty of bandwidth to transfer blocks of commands and audio to the Box at interrupt level, its response to user input ordinarily went through the time-sharing layer of the operating system, which meant interactive applications competed with other time-sharing services, degrading their responsiveness. A work-around for this situation was to run the interactive application in "spacewar" mode, which guaranteed a modicum of real-time response to user inputs.

Named for Steve Russell's game mentioned earlier, spacewar mode had originally been developed at SAIL to support Russell's game in particular, and real-time interactivity under time-sharing in general. Andy Moorer told me via electronic mail that only one application at a time could avail itself of spacewar mode, which ran in "User I/O mode" to poll I/O devices every 1/60 second. "Address mapping was turned on, so you didn't have to write PC-relative (relocatable) code. On the other hand, you couldn't reference system memory, or any memory outside of your 'core image' (remember that term?)."

The fact that there was but one command stream to the Box was a related problem, because all polyphonic command updates had to be merged together in time order by the controlling computer. Moorer (1989) gives the example of merging *M* envelopes with *N* points each, resulting in computational complexity *MN*. He continues, "Notice that a consequence of the single command stream with embedded timing information is that keyboard performance is essentially ruled out. The pressing of a key is asynchronous, and the data (envelope control) for that note would have to be merged into the command stream, which is only possible for very simple (and musically trivial) command streams."

Nonetheless, the Box was eventually used for some fully interactive applications. For example, David Jaffe is shown in a YouTube video triggering notes on the Box in real time from a computer keyboard (Olczak 1984). Bill Schottstaedt (2012) told me, "We had compositional algorithms that we could control as they were generating sound via Pla . . . we were changing the music as the box computed it and played it." Doug Keislar told me, "I adapted code written by Bill [Schottstaedt] and David [Jaffe] to control the Box in real time, and put them in a SAIL program I called KEYBD that added a GUI and other functionality," enabling interactive polyphony for research in tuning systems (Keislar 2013; see Keislar 1987 for details). Keislar recalled "playing 3- and 4-note chords from the computer keyboard" using this software. (The keyboard scanning mechanism, not the Box, imposed the maximum of four polyphonic notes.) But "simultaneously pressed keys often resulted in glaringly different attack times." He attributed this to his application's not using spacewar mode, which he had not known about.

On the lighter side, David Jaffe reported implementing the then-popular computer game Pong using spacewar mode, and generating the sounds on the Samson Box using plucked-string synthesis (which, because it generated its own envelopes implicitly, did not require sample-level control).

Load-Balancing Fast Synthesizers and Slow CPUs

Lurking beneath the Samson Box's interactive difficulties lay a fundamental problem that affected virtually every design of this era: How to structure a slow CPU to provide musically uniform response times to user input gestures while simultaneously servicing the asynchronous demands of a high-speed synthesis engine for data and control. There were (and still are) many ways to get load balancing wrong. For example, Moorer (1989) documents critical computational load-balancing and timing problems encountered while developing the 4C machine.

He also told me anecdotally of similar difficulties he witnessed with the Bell Digital Synthesizer. It had many rows of sliders for interactive synthesis control. But moving a handful of sliders at the same time so "crowbarred" the machine that it stopped responding for several *minutes* while it worked through all the interrupts the sliders had generated. Moorer told me via electronic mail, "Recall that we are talking about gawd-awfully slow microprocessors—circa 1977 or so."

In the "bad old days" before general-purpose computers got fast enough, latencies in the controlling computer were very common problems that every designer of synthesizer systems struggled with (see also Moorer 1989; Kahrs 2002). The solution was to use sufficiently fast processors and high-bandwidth control interfaces that could mask the latencies—but in the 1970s, such machines were still decades away.

A Grab Bag of Architectural Issues

We tended to stumble over problems as we gained mastery in using the Box, extending our reach into its capabilities. For example, we eventually realized that the size of the scratchpad memory for interconnecting processing elements (called sum memory) cramped the complexity of patches. There could only be 128 processing element outputs stored independently—64 for generators, 64 for modifiers—though there were 256 generators and 128 modifiers. As a result, there were many more processing elements than there were independent ways to connect them. The size of sum memory was limited because of the high bandwidth of access required of it by the processing elements.

We found out the hard way that the filtering algorithms lacked saturating overflow and rounding control. This meant they could "blow up" if pressed to their numerical limits.

It was difficult to generate very-low-amplitude sine waves, which would have been useful for generating vibrato signals. The higher the sampling rate the worse the problem. The eventual solution was to implement vibrato by periodic changes to oscillator frequency in the command stream. Though this cluttered up the command stream, it allowed more interesting vibrato to be realized through mixtures of filtered noise and sinusoidal vibration calculated in the host computer, and lessened the processing burden on the Box.

Critique of the Box's Generality

One of Samson's stated goals was to provide a flexible architecture of basic building blocks that would allow the Box to be used for synthesis techniques not yet discovered. In the white paper mentioned earlier, Samson (1974) wrote,

> This synthesizer is viewed as general-purpose, not only because its design encompasses the currently favored synthesis techniques, but also because it offers basic computational building blocks which can be interconnected, under program control, to perform at high speed new synthesis techniques as they are developed.

With plenty of hindsight, we can now ask how true that prediction was. For a fact, some synthesis algorithms invented after it was designed were adaptable to the Box, but some were not. Waveshaping (Arfib 1978; Lebrun 1979), for example, was developed after the Box was designed. It could be implemented by storing tables of Chebychev polynomial functions in delay memory, but the Box's limited number of delay memory ports restricted waveshaping to 32 independent voices. Likewise, plucked string synthesis, as developed by Jaffe and Smith (1983; see also Karplus and Strong 1983), depended on filtering wavetables, and so was also limited to 32 voices on the Box. But Jaffe found creative ways to overcome the 32-voice limit. He told me,

> *Silicon Valley Breakdown* and *May All Your Children Be Acrobats* both used a combination of plucked-string techniques with FM and other synthesis techniques, . . . so I got more than just 32 voices that way. Also, for the very high notes, I didn't use delay units at all! The delays needed to be less than the pipeline delay of the delay units, so I used simple modifiers and sum memory.

Techniques such as general waveguides (Smith 2006, 2008), Chant vocal synthesis (Rodet, Potard, and Barrière 1984), physical modeling synthesis (Smith 2012), and some forms of reverberation, however, did not fit at all.

Normative Musical Ideas

Not all the problems were Samson's fault. Normative ideas about music that we built into our software also caused problems. For example, at one point I was hired by Stanford psychoacoustician Roger Shepard to realize his signature acoustic illusion, the Shepard tone (Shepard 1964), on the Box, to accelerate his research. Though this was straightforward to program on a mainframe, implementation on the Box was tortured because we had carried over the Music *N* software synthesis model—which Shepard's illusion violated. In Music *N*, the note statements were all isolated events and, once triggered, statically evolved without further input. But this left out the possibility of dynamic interactions among notes, such as real musicians have with their instruments and each other. For MBOX, I eventually developed the system of meta-instruments described above, but by then Shepard had lost interest. Jaffe later developed a system where note statements could update running patches to re-strike a note that was already sounding. He told me,

> So, for example, in the opening of *Silicon Valley Breakdown*, a single string is plucked over and over with the dynamics filter and pick position filter changing, with alternating up and down picking. The energy from one note caries over into the next, builds up, and so on.

The Problem of Unique Software

The reason to go to the trouble of designing and implementing special-purpose hardware is if there is a significant performance benefit not obtainable any other way that makes it worth the effort and cost. For the Samson Box, the benefit was its capable real-time processing power. But we paid dearly in software development costs to gain this benefit. After all, the Box came without a scrap of software. In his critique, Julius Smith (2005) points out that

> [t]ens of man-years of effort went into software support. A large instrument library was written to manage the patching of hardware unit generators into instruments. . . . Debugging tools were developed for disassembling, editing, and reassembling the synthesizer command-stream data. Reading and manipulating the synthesizer command stream was difficult but unavoidable in serious debugging work. Software for managing the unique envelope hardware on the synthesizer was developed, requiring a lot of work. Filter support was complicated by the use of 20-bit fixed-point hardware with non-saturating overflow and lack of rounding control. General wavetables were not supported in the oscillators. Overall, it simply took a lot of systems programming work to make everything work right.

Hard on the heels of these costs was the cost of developing musical scores for the Box. Debugging a score by reading and understanding the synthesizer command stream was as non-trivial as it was unavoidable for all but the simplest scores—at least in the early days. This was an especially tricky dilemma for me as implementer and maintainer of the compiler, as I—and others—stumbled over bugs while I was trying to write my thesis, *Nekyia* (Loy 1979), to graduate. I must finish the composition to graduate; but I must fix the bugs in order to write the piece; but fixing the bugs could more than eat up all the composing time I had—and then I wouldn't graduate! It became so difficult, I eventually formulated a rule: If I'm fixing a problem, I'm debugging; if I'm working around a bug, I'm composing!

But whereas it might have been straightforward to conjure up a software-synthesis solution to a problem, the lure of the Box was its ability to do quite a lot of processing of a fairly general nature in real time. So we'd keep trying to adapt our ideas to the Box if we possibly could. A great deal of creativity was spent trying to find the right size shoehorn to make a new synthesis algorithm fit on the Box. This took a lot of time. Smith (2005) goes on to say,

> Research into new synthesis techniques slowed to a trickle. . . . Reconfiguring a complicated patch of Samson Box modules was much more difficult [than software synthesis], and a lot of expertise was required to design, develop, and debug new instruments on the Box. Many new techniques such as waveguide synthesis and the Chant vocal synthesis method did not map easily onto the Samson Box architecture. Bowed strings based on a physical model could not be given a physically correct vibrato mechanism due to the way delay memory usage was constrained. Simple feedback FM did not work because phase rather than frequency feedback is required. Most memorably, the simple interpolating delay line . . . was incredibly difficult to implement on the Box, and an enormous amount of time was expended trying to do it. While the Samson Box was a paragon of design elegance and hardware excellence, it did not provide the proper foundation for future growth of synthesis technology. It was more of a music instrument than a research tool.

I remember a hallway conversation (where most of our best work took place) one day with Chowning and Moorer after about a year's experience with the Box. Our consensus was that, if only general-purpose computers were 100 times more powerful, we would not need—nor want—specialized hardware synthesis. This sentiment was later borne out in spades with the limitations that were built into commercial MIDI synthesizers and PC sound cards, which made the Samson Box seem, in comparison, like the Starship Enterprise from *Star Trek*.

The Problem of Unique Hardware

And then there's the greatest limitation of them all: there was exactly one Samson Box in the whole world, it was not portable, and it would cost US$ 100,000 each to build more of them.

This was also a disincentive to explore real-time applications of the Box. Jaffe told me by electronic mail, "There was a strong motivation *not* to do anything performance-oriented with the Samson Box due to its lack of portability. A tape piece could be played worldwide (*Silicon Valley Breakdown* was played in over 25 countries), whereas a live Samson Box could be played in exactly one location."

Worse: When anyone left Stanford, all music and research done on the Box was left behind, stranded. Until at least the mid 1990s, no digital storage medium was big enough, portable enough, and cheap enough to allow researchers to carry away works of musically useful proportions in digital form. Jaffe told me,

> As you'll remember, in those days a 100 MB disk, a UDP [user disk pack], had to be shared among many users. For this reason, I have virtually no copies of my files from those days, as they were all on 7- and 9-track DART [dump-and-restore technique] tapes that become unreadable with time. This also discouraged sound-file-based pieces at CCRMA. It really wasn't practical to have more than one or two people working on them at the same time.

No one even bothered to create a driver for saving digital audio output from the Box until the mid 1980s because, in those days, storage was prohibitively expensive and limited in capacity. Though I was long gone from Stanford by that time, this was finally done at my instigation: I'd graduated and gone to teach at UC San Diego, and desperately wanted the bits to my thesis. A long-time CCRMA/SAIL staff, hacker, and friend, known to all simply as Tovar, wrote a magnetic tape driver for me. *Nekyia*, at 10 min duration, required ten nine-track "magtapes" to store. A decade later, we used the very last computer on the Stanford campus that had a nine-track magtape drive (in the Chemistry department) to read the tapes and store the bits on one writable compact disc (at a cost of US$ 25 a disc). Over the intervening years to the present, *Nekyia* has survived across a plethora of different kinds of media, fleeing from one to the next as each became obsolete.

Finally, all software written for the Box was also stranded at Stanford. All of our Box-specific code, plus the unique operating system and software development tools CCRMA inherited from SAIL, were all tied to Digital Equipment Corp. 36-bit hardware, were utterly non-portable, and died when the PDP-10 architecture died.

Conclusions

The Samson Box and the system we developed at CCRMA to control it provided by far the most powerful and general-purpose solution to real-time digital audio computation of its era, enabling enormous strides in computer music. It implemented all synthesis techniques known at the time, and was general-purpose enough to handle many new approaches.

Limitations in its design frustrated development of interactive applications for several years. Though these problems were eventually overcome, the Box was used mostly to realize precomputed musical scores. Because it was an expensive and unique piece of hardware controlled by a unique computer system running unique software, at a time when disk storage was still very primitive, one's work on it was chained to CCRMA.

Nonetheless, the Samson Box remained the gold standard for music synthesis for over a decade. It stood at the crossroads of the transformation of music and audio from analog to digital practices, a technological sea change that continues today at a rapid pace.

The ultimate testimony of the Samson Box is the music it was used to create. Over its lifetime, more than one hundred works were composed with it. Many of these works are listed at the end of this article. It served hundreds of composers and researchers. Numerous compositions won prestigious prizes. Many pieces are still performed, and many recordings are still commercially available 35 years later. The Box unleashed a burst of creative energy among composers of its day that helped put computer music on the map.

Epilogue: The End—Or Not

Eventually, the hothouse environment of SAIL came to an end. Shortly after I left Stanford in 1980, SAIL moved from the D.C. Power building to new facilities on campus; CCRMA took over the whole building. The PDP-6 was replaced with a Foonly F2 (and later an F4) computer, designed by Dave Poole (the same man who'd helped Chowning develop synthesis software in the mid 1960s). Development of new Box software and music continued over the years. Bill Schottstaedt wrote his own compiler called sambox, which was widely used. Schottstaedt (2012) told me,

> I think some composers continued to use MBOX, in particular Chowning; others, like Mike McNabb, went off on their own path. Many people puttered around with the library code—Julius Smith, Ken Shoemake, David Jaffe. I have a list of the music group documentation files towards the very end of its life—say from 1990 or so. Besides those mentioned above, I see Andy Moorer, John Strawn, and Marc LeBrun. And many others developed instruments: Jan Mattox, Paul Wieneke, Xavier Serra, etc. David Jaffe wrote a debugger for the Samson box. It was a group effort.

With the advent of MIDI and networked personal computers, costs plummeted, and computational power multiplied. Eventually, individuals of ordinary means could afford systems for creating sophisticated digital music, and the function of CCRMA as a technology magnet dissipated. In fact, some technological developments at CCRMA actually discouraged collaboration: It was no longer possible to snoop other's screens; an audio crossbar switch was set up so that users only heard their own sounds from the Box. Chowning has told me several times that he regretted these developments for the isolation they encouraged. But CCRMA has remained a pre-eminent forum for collaboration between musicians and scientists—a function no technology can replace.

Eventually, as Moore's Law predicts, general-purpose computers became fast and cheap enough to sweep past the capabilities of specialized hardware synthesizers. Today, 35 years later, even the average cell phone has computational power comparable to the Samson Box; they fit in your hand; they run on precious little power; they cost several orders of magnitude less each than the Box did; and there are billions of them all around the globe. On 3 April 1992, the Samson Box was decommissioned and moved to the permanent collection of the Musée de la Musique, part of the Cité de la Musique, in Paris, France, ending an era.

Or was it the end? Certainly users' experiences informed subsequent practices in academia and industry. Speaking of the Music Kit for the NeXT computer (Jaffe and Boynton 1989), Jaffe told me that it "was my attempt to unify what at Stanford had been two divergent paradigms: the Music *N* legacy and MIDI." Further recent signs of life: In order to capture some old Box scores in digital format, Schottstaedt, assisted by Mike McNabb, recently wrote a Samson Box emulator in the C programming language, which runs in real-time on a standard laptop using 32-bit floating point arithmetic (Schottstaedt and McNabb 2012). Because 32-bit floating point has 24 bits of mantissa precision—4 bits more than the data paths in the Box (20-bit fixed-point fractional)—the sound from Schottstaedt's emulator surpasses the fidelity of the Box. It runs comfortably in real time on a standard laptop. McNabb and Schottstaedt used the emulator to remaster the soundtrack for the film *Mars in 3D* composed originally for the Box in the 1980s (see McNabb and Schottstaedt 2012; Loy 2013b).

The progress of computer music in the late 20th century can be compared to the development of the steam engine and its impact on the industrial revolution. Originally, steam engines were so bulky and inefficient that their only practical applications were stationary—pumping water from mines. But having even a single practical application was enough to give engineers and tinkerers a target of opportunity for improvements. When an improvement devised by James Watt increased their efficiency to the point that steam engines could be incorporated into moving vehicles, the age of steam was born. This history is reminiscent of the progress of computer music. Even the scarce availability of general-purpose computers was sufficient for engineers and tinkerers to explore the rich ramifications of this new resource for music. Just as a threshold of efficiency unleashed the age of steam, so too a threshold of integration unleashed the age of computer music. Just as knowledge sharing through publication among principals in the industrial revolution led to breakthroughs, so too the open intellectual framework of many institutions and individuals around the world—often shared in the pages of this journal—led quickly to the transformative music technologies our field now enjoys.

Acknowledgments

I wish to thank Peter Samson, Andy Moorer, John Chowning, Julius Smith, Mark Kahrs, Bill Schottstaedt, Mike McNabb, Doug Keislar, Andrew Nelson, and David Jaffe for their contributions to this article. Thanks to John Strawn for access to his trove of Samson Box documents, including some long thought lost. Thanks also to Leland Smith, Loren Rush, and John Grey, who all contributed, directly or indirectly, to the Samson Box project. Finally, thanks also to the composers and researchers at CCRMA and elsewhere whose perseverance has led to great success in our field.

List of Compositions

This listing of music composed on the Samson Box is a compilation of lists provided by Patte Wood, Bill Schottstaedt, Andrew Nelson, and others. Any inaccuracies or omissions are the responsibility of this author alone.

The listing below shows dates of composition and/or first performance. Also note, "quad" means "quadraphonic" (i.e., four-channel surround) sound. The CCRMA standard for distribution of four-channel audio was based on the conventional layout of the Cartesian coordinates, as follows. Channel 1: front right, channel 2: front left, channel 3: rear left, channel 4: rear right. "Web" means the music is available by searching for the composer or title on the World Wide Web.

1978 John Chowning, *Turenas* (quad; originally produced 1972, remastered for the Box December 1978; stereo reduction on *John Chowning*, Wergo, WER 2012-50); Stuart Dempster, *Standing Waves* (Web); Bill Schottstaedt, *Five Bagatelles*, *The Gong Tormented Sea*, *You're So Far Away*

1979 Jonathan Berger, *To the Lost History of Hope*; David Jaffe, *From a Sleeping Circus Animal's Perspective* (quad); Paul Kirk, *Desert Dance*; Gareth Loy, *Nekyia* (quad; Bourges prize 1981; stereo reduction on *Computer Currents 5*, Wergo, WER 2025-2); Janis Mattox, *Dragon's View*; Mike McNabb, *Mars Suite* (on *Mike McNabb*, Wergo, WER 2020-2); Mike McNabb with Bill Schottstaedt, *Mars in 3D* (on *Mars in 3D*, AIX Records, AIX86067), *Music for Altered States* (on *Altered States: Original Soundtrack*, most of track 8 [uncredited], RCA Victor, 3983-2-RG); Bill Schottstaedt, *Daily Life among the Phrygians* (on *Computer Music*, LP issued by IBM Deutschland GmbH in 1984, no catalog number; fragment used [without credit] in *Altered States*, directed by Ken Russell, 1980); Paul Weineke, *A Garden for Orpheus*

1980 Jonathan Berger, *A Pocketful of Posies* (Bourges Prize 1983); Tristram Cary, *Nonet*; Johannes Goebel, *Mother Goose*; David Jaffe, *May All Your Children Be Acrobats*; W. Andrew Schloss, *The Towers of Hanoi* (on *The Digital Domain: A Demonstration*, Elektra/Asylum, 9 60303-2; Web)

1981 Jonathan Berger, *Meteora* (Bourges prize 1987); Chris Chafe, *Solera* (Bourges Prize 1982); Thierry Lancino, *Static Arches* (Web); Janis Mattox, *Call to the Spirits* (on *Digital Digital Domain: A Demonstration*, Elektra/Asylum, 9 60303-2), John Chowning, *Phoné* (on *John Chowning*, Wergo, WER 2012-50); Michael McNabb, *Love in the Asylum* (quad; Bourge prize 1982; stereo reduction on *Michael McNabb*, Wergo, WER 2020-2); Bill Schottstaedt, *Colony* (excerpt from *Colony* V on *Digital Domain: A Demonstration*, Elektra/Asylum, 9 60303-2); Paul Weineke, *Attend*

1982 Jonathan Berger, *Diptych* (on *CDCM Computer Music Series Vol. 8*, Centaur CRC2091); David Jaffe, *Silicon Valley Breakdown* (available at the iTunes Music Store); Janis Mattox, *Song from the Center of the Earth* (excerpt on *Digital Digital Domain: A Demonstration*, Elektra/Asylum, 9 60303-2); Malcolm Singer, *Man versus Machine*

1983 Chris Chafe, *In a Word* (Bourges Prize 1984; on *Dinosaur Music*, Wergo, WER 2016-50), *Neriage;* Bill Schottstaedt, *From the Book of the Burning Mirror*, *Variation on Ives' Unanswered Question*; Paul Weineke, *A Garden for Orpheus*

1984 Jonathan Berger, *An Island of Tears* (Bourges prize 1991); Joanne Carey, *Gamelan R-gong Gong*; Chris Christensen, *Untitled*; Kent Deveraux,*Study No. 3*; Doug Fulton, *Red Cup and Rat*; David Jaffe, *Bristlecone Concerto No. 2*; Marc LeBrun, *Need to Know*; Bill Schottstaedt, *Daybreak*, *Dinosaur Music* (on *Dinosaur Music*, Wergo, WER 2016-50); Amnon Wolman, *Etude (Homage à Bartok No.2)*

1985 Douglas Fulton, *Bowling for Blood* (on *Computer Music Currents* 11, Wergo, WER 20312); David Jaffe, *Telegram to the President* (on *CDCM Computer Music Series Vol. 8*, Centaur, CRC 2091); Stanislaw Krupowicz, *Thus Spake Bosch* (Web); Fred Malouf, *Chromatonal* (quad); Michael McNabb, *Invisible Cities* (on *Michael McNabb*, Wergo, WER 2015-50); Adolfo Nuñez, *Canales*; Bill Schottstaedt, *Water Music* (first movement on *Dinosaur Music*, Wergo, WER 2016-50); Bill Schottstaedt, Jonathan Berger, David Jaffe, Doug Fulton, and Robert Shannon, *Fanfare*; Heinrich Taube, *JubJub*; Amnon Wolman, *A Circle in the Fire* for bass clarinet and tape (NewComp first prize 1988; on *Computer Music Currents 6*, Wergo, WER 2026-2), *If Thorns. . .* , *Mora* for soprano, mezzo-soprano, oboe, percussion, and tape (Minnesota Composers Forum prize 1988), *Perhaps, at Last, Some Such Hours Passed*

1986 Chris Chafe, *Quadro* on *CDCM Computer Music Series Vol. 8*, Centaur, CRC 2091); Doug Fulton, *You'll Never Walk Again*; David Jaffe, *The Fishing Trip*; Richard Karpen, *Eclipse* (Newcomp prize 1986; Bourges Prize 1987); Stanislaw Krupowicz, *Farewell Variations on a Thema by Mozart* for amplified string quartet and tape (Irino Prize 1987); Ira Mowitz, *Jubilum*; Bill Schottstaedt, *Put on a Happy Face*, *Sonata*

1987 Douglas Fulton, *Tip the Velvet*; David Jaffe, *Grass* for female chorus and tape; Richard Karpen, *Exchange* for solo flute and tape (Bourges prize 1987; Newcomp prize 1987; National Flute Association prize 1988), *Il Nome* for soprano and tape (Bourges prize 1989; on *Computer Music Currents 7*, Wergo, WER 2027-2); Servio Marin, *Fantasmas de los mundos*; Ira Mowitz, *Darkening* (Web); Bill Schottstaedt, *Brand X Music*; Amnon Wolman, *And Then She Said* for actress, tape, four pre-recorded voices, and live computer graphics, *M* for orchestra and tape, *Medea* (Ars Electronica finalist 1987; on *Medea*, Divided Records, Div03)

1988 Rachel Boughton, *Pigeon Fantasies* soundtrack for short film (Bourges honorable mention); Joanne Carey, *Cloud's Lament* (Danish Institute for Electronic Music [DIEM], Audioarkiv CD 89-002); Fred Malouf, *Sacrifice* for celletto and tape; Heinrich Taube, *Tremens* (Bourges honorable mention); Amnon Wolman, *Ladders and Plains* music for multimedia performance (on *Medea*, Divided Records, Div03), *Nautilus* for three singers, radio operator and tape (Minnesota Composers Forum award), *Play by Samuel Beckett* (incidental music)

1989 Anthony Holland, *Die Heiligenstadter Testament* (Bourges finalist 1991); Geir Johnson, *RADAR*, *Some of Sam*; Ira Mowitz, *Shimmering* (on *A la Memoire d'un Ami*, New Albion, NAR 047; Web); Bill Schottstaedt, *Busted Pipes*, *I'm Late*, *Icicles*, *Idyll*, *Long Ago and Far Away*, *Pastorale*, *Wait For Me!*, *Windmills*; Amnon Wolman, *The Day the Bank Came Through* for narrator, percussionist, and tape, *Distant Images*, for oboe and tape, *Distorted Reflections* for oboe, chamber ensemble, and tape, *Reflections on Pedestals*, for orchestra and tape

1990 Joanne Carey *Intonations of the Wind*; Judith Schatin, *Tenebrae super faciem abyssi*; Bill Schottstaedt, *Brown Music*; Amnon Wolman, *The Many Faces of Marilyn—Four Requiems*, for improvisation group and tape (revised 1991, on *The Marilyn Series*, Centaur, CRC 2573)

1992 Stanley Jungleib, *Earth Sighs*

References

Alles, H. 1977. "A Portable Digital Sound Synthesis System." *Computer Music Journal* 1(4):5–6.

Alles, H. 1979. "An Inexpensive Digital Sound Synthesizer." *Computer Music Journal* 3(3):28–37.

Alonso, S., J. H. Appleton, and C. Jones. 1976. "A Special Purpose Digital System for Musical Instruction, Composition, and Performance." *Computers and the Humanities* 10(4):209–215.

Arfib, D. 1978. "Digital Synthesis of Complex Spectra by Means of Multiplication of Non-Linear Distorted Sinewaves." In *Proceedings of the International Computer Music Conference*, pp. 70–84. Available online at quod.lib.umich.edu/i/icmc/bbp2372.1978.009?view=toc. Accessed April 2013.

Blesser, B., and J. M. Kates. 1978. "Digital Processing in Audio Signals." In A. V. Oppenheim, ed. *Applications of Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall.

Buxton, W., et al. 1978. "An Introduction to the SSSP Digital Synthesizer." *Computer Music Journal* 2(4):28–38.

Computer Music Journal. 1978. "Products of Interest: Micor Inc. Coupland Digital Synthesizer." *Computer Music Journal* 2(1):64–66.

Doornbusch, P. 2009. "Early Hardware and Early Ideas in Computer Music—Their Development and Their Current Forms." In R. Dean, ed. *The Oxford Handbook of Computer Music*. Oxford: Oxford University Press, pp. 44–84.

Earnest, L. 1972. *Video Switch*. Available online at www.saildart.org/VDS.LES%5BS,DOC%5D1. Accessed April 2013.

Fedorkow, G., W. Buxton, and K. C. Smith. 1978. "A Computer Controlled Sound Distribution System for the Performance of Electroacoustic Music." *Computer Music Journal* 2(3):33–42.

Frost, M. 1975. "UUO Manual (Second Edition)." Operating Note 55.4. Stanford, California: Stanford University, Stanford Artificial Intelligence Laboratory.

Harrenstien, K. 1977. "Finger Protocol." RFC 742. Available online at tools.ietf.org/html/rfc742. Accessed April 2013.

Hiller, L. J., and L. Issacson. 1959. *Experimental Music*. New York: McGraw-Hill.

Jaffe, D. A. 1983. "A Synthesizer Debugger." In *Proceedings of the International Computer Music Conference*. Available online at quod.lib.umich.edu/i/icmc/bbp2372.1983.011?view=toc. Accessed April 2013.

Jaffe, D. A., and L. R. Boynton. 1989. "An Overview of the Sound and Music Kits for the NeXT Computer." *Computer Music Journal* 13(2):44–55.

Jaffe, D. A., and J. O. Smith. 1983. "Extensions of the Karplus-Strong Plucked String Algorithm." *Computer Music Journal* 7(2):56–69.

Kahrs, M. 2002. "Digital Audio System Architecture." In M. Kahrs and K. Brandenburg, eds. *Applications of Digital Signal Processing to Audio and Acoustics*. New York: Kluwer.

Karplus, K., and A. Strong. 1983. "Digital Synthesis of Plucked String and Drum Timbres." *Computer Music Journal* 7(2):43–55.

Keislar, D. 1987. "History and Principles of Microtonal Keyboards." *Computer Music Journal* 11(1):18–28. Published with corrections in 1988 as "History and Principles of Microtonal Keyboard Design." Stanford University: Report STAN-M-45. Available online at ccrma.stanford.edu/STANM/stanms/stanm45. Accessed April 2013.

Keislar, D. 2013. Personal communication.

Lebrun, M. 1979. "Digital Waveshaping Synthesis." *Journal of the Audio Engineering Society* 27(4):250–266.

Loy, D. G. 1977. "LRNSAM—Systems Concepts Digital Synthesizer Operations Manual and Tutorial." Internal Report STAN-M-6. Stanford University: Center for Computer Research in Music and Acoustics, Department of Music.

Loy, D. G. 1979. *Nekyia*. DMA thesis, Stanford University, Department of Music.

Loy, D. G. 1981. "Notes on the Implementation of MUSBOX, a Compiler for the Systems Concepts Digital Synthesizer." *Computer Music Journal* 5(1):13–33.

Loy, D. G. 2013a. "Systems Concepts Digital Synthesizer: an Architectural Retrospective." *Computer Music Journal* 37(3):49–67.

Loy, G. 2013b. "Mars in 3D". Review of McNabb and Schottstaedt Blu-ray disc *Mars in 3D*. *Computer Music Journal* 37(3):101–103.

Mathews, M. V. 1963. "The Digital Computer as a Musical Instrument." *Science* 142(3592):553–557.

Mathews, M. V. 1971. "The Electronic Sound Studio of the 1970's." In W. Skyvington, ed. *Music and Technology*. Paris: La Revue Musical, pp. 129–141.

Mathews, M. V., and F. R. Moore. 1970. "GROOVE—a Program to Compose, Store, and Edit Functions of Time." *Communications of the ACM* 13(12):715–721.

Mathews, M. V., et al. 1969. *The Technology of Computer Music*. Cambridge, Massachusetts: MIT Press.

McNabb, M., and W. G. Schottstaedt. 2012. *Mars in 3D: Images from the Viking Mission*. Los Angeles, California: AIX Records, AIX 86067, Blu-ray 3D.

Moore, F. R. 1977. "Real Time Interactive Computer Music Synthesis." STAN-M-7. PhD dissertation, Stanford University, Department of Electrical Engineering.

Moore, F. R. 1985. "The FRMbox: A Modular Digital Music Synthesizer." In J. Strawn, ed. *Digital Audio Engineering: An Anthology*. Los Altos, California: Kaufmann, pp. 95–107.

Moorer, J. A. 1979. "About This Reverberation Business." *Computer Music Journal* 3(2):13–28.

Moorer, J. A. 1989. "Synthesizers I Have Known and Loved." In C. Roads, ed. *The Music Machine.* Cambridge: MIT Press, pp. 589–598.

Moorer, J. A. 2012. Personal communication.

Moorer, J. A., et al. 1979. "The 4C Machine." *Computer Music Journal* 3(3):16–24.

Olczak, G. 1984. "David A. Jaffe: Silicon Valley Breakdown; A Film by George Olczak." Available online at www.youtube.com/watch?v=_p4DGE5t3x0. Accessed April 2013.

Rodet, X., Y. Potard, and J.-B. Barrière. 1984. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General." *Computer Music Journal* 8(3):15–31.

Samson, P. R. 1974. "Systems Concepts Digital Synthesizer Specifications." Programming Specification. San Francisco, California: Systems Concepts. Revised 24 November 1974.

Samson, P. R. 1980. "A General-Purpose Digital Synthesizer." *Journal of the Audio Engineering Society* 28(3):107.

Samson, P. R. 1985. "Architectural Issues in the Design of the Systems Concepts Digital Synthesizer." In J. Strawn, ed. *Digital Audio Engineering: An Anthology*. Los Altos, California: William Kaufmann, pp. 61–94.

Schottstaedt, W. G. 1983. "Pla: A Composer's Idea of a Language." *Computer Music Journal* 7(1):11–20.

Schottstaedt, W. G. 2012. Personal communication.

Schottstaedt, W. G., and M. McNabb. 2012. "Samson Box Emulation Software." Available online at ftp://ccrma-ftp.stanford.edu/pub/Lisp/sam.c. Accessed April 2013.

Shepard, R. N. 1964. "Circularity in Judgments of Relative Pitch." *Journal of the Acoustical Society of America* 36(12):2346–2353.

Smith, J. O. 2005. "Experiences with the Samson Box." Keynote Paper. In *Proceedings of the International Computer Music Conference*, pp. 1–10. Revised with Curtis Roads for publication in *Cahiers de l'IRCAM*, 1992. Available online at https://ccrma.stanford.edu/~jos/kna/Experiences_Samson_Box.html. Accessed April 2013.

Smith, J. O. 2006. *A Basic Introduction to Digital Waveguide Synthesis*. Available online at https://ccrma.stanford.edu/~jos/swgt/. Accessed April 2013.

Smith, J. O. 2008. "Digital Waveguide Architectures for Virtual Musical Instruments." In D. Havelock, S. Kuwano, and M. Vorländer, eds. *Handbook of Signal Processing in Acoustics*. Berlin: Springer, pp. 399–417.

Smith, J. O. 2012. *Physical Audio Signal Processing*. Available online at https://ccrma.stanford.edu/~jos/pasp. Accessed April 2013.

Smith, L. 1972. "SCORE—A Musician's Approach to Computer Music." *Journal of the Audio Engineering Society* 20(1):7–14.

Swinehart, D., and B. Sproull. 1970. "SAIL." Available online at ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/76/574/CS-TR-76-574.pdf. Accessed April 2013.

Wallraff, D. 1979. "The DMX-1000 Signal Processing Computer." *Computer Music Journal* 3(4):44–49.