2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
Paper ID #40276

The Superstition Lecture: The Music Theory of Stevie Wonder as a Metaphor for Computing Levels of Abstraction

Dr. Matthew Morrison, University of Notre Dame

Matthew A. Morrison is an Associate Teaching Professor of Computer Science and Engineering at the University of Notre Dame. He is a Senior Member of the IEEE and the ACM, and the co-organizer of the Design Automation Conference Summer School. Dr. Morrison has won Best Paper Awards at the IEEE Integrated STEM Education Conference, Cadence CDNLive, and the IEEE VLSI Symposium. He was the recipient of the 2018 NACADA Global Academic Advising Award for Faculty.

The Superstition Lecture: The Music Theory of Stevie Wonder as a Metaphor for Computing Levels of Abstraction

Matthew Morrison
University of Notre Dame

Abstract

Effectively conveying the importance and breadth of computing at every level of abstraction to students through instruction is emerging as a critical challenge to cultivating the next generation of computer scientists. Students should not only learn the science and engineering of computing, but enjoy computer programming as an art form in order to effectively promote computational thinking. In this paper, an extended metaphor of the levels of abstraction in a computer and Stevie Wonder's seminal hit song *Superstition* is presented. *Superstition* is widely studied in introductory music theory classes to show how the repetitive grooves of funk music create a powerful and complex flow, and how that flow allows musical artists to use harmonic progression, vocals, and instrumentation to imbue songs with deeper meaning. Likewise, the repetitive processing in a computer architecture creates a power and complex data flow, which allows computer scientists to use data structures and algorithms, procedural and object-oriented programming, and logic design to imbue programs with efficiency, usability, and robustness.

The Superstition Lecture is presented as a course review for CS0, CS1, and CS2 computer science courses, as well as a preview for future course material. Because these courses are pre-requisite for most CS courses in academic curricula, reinforcing the importance of the concepts they have learned - and tying them to future concepts - is critical for setting students up for success. We present Stevie Wonder's use of synthesizers, drum figures, ostinato, and cadential progression in *Superstition* as a form of "musical computer programming". These comparisons provide introductory students insights into advanced computing concepts, including machine learning algorithms, hardware side-channel attacks, and the importance and career benefits of diversifying computing skills at several levels of abstraction.

Keywords

Levels of Abstraction, Metaphors, Computer Science, Computer Engineering, Paradigms

Motivation: Music and Fluency in Computing Levels of Abstraction

The notions of computer science as an art form itself and as a platform for creative minds to exercise new artistic direction have been intertwined since Ada Lovelace envisioned a future where computer served as more than calculator machines.[1] Her insight into the potential for computing machines stemmed in part from her dedicated study of the piano, singing, and as a harpist.[2] Since music and the "fundamental relations of pitched sounds" may be quantified as a science of the harmony of expression, she envisioned a computing machine that could compose elaborate pieces of music to any computable degree of complexity.

While the extent, capability, and ethical ramifications of a computing device to "think" and "create art" have long been debated by computer scientists, many pioneers in the field will argue that the process of designing a computer program is similar to *composing* music or poetry. Donald Knuth begins his magnum opus *The Art of Computer Programming* with the argument that a computer scientist who understands computer programming at several levels of abstraction will find the process aesthetically pleasing[3] "much like composing poetry or music."[3] Professor Knuth had three crucial characteristics in common with Ada Lovelace: a strong understanding of mathematics, a passion for music, and an understanding of the connection between the two.

In fact, universities have long observed a correlation between success in the computing field and musical proficiency, and that the highest scores on the admissions tests and best performers in the major are often "people with a background in music."[4] At the University of (removed for double blind), the highest percentage of majors in the Marching Band reside in Computer Science and Engineering. The fundamental skills of identifying and manipulating patterns are the same in musical composition and program development. Music and programming require the ability to view the composition at multiple levels of abstraction using spatial and temporal reasoning. *Schenkerian analysis* is an approach to music theory where tonality in a musical composition represented in "structural levels of abstraction", where music form itself is defined as "an energy transformation, as a transformation of forces that flow from background to foreground through all the levels."[5] Musical composers must understand how to compose a symphony using a variety of instruments, including drums, strings, brass, and woodwinds, and use those to convey emotion and meaning. Likewise, computer programmers understand these levels of abstraction - logic design, computer architecture and organization, compilers, and procedural and object-oriented programming, and computational algorithms all within an operating system – in order to effectively utilize computing devices towards solving complex problems.

Effectively conveying the importance and breadth of every level of abstraction to students through instruction is emerging as a critical challenge to cultivating the next generation of computer scientists. Imbuing fluency in computing at several levels of abstraction has always been challenging, but it has become more difficult for a novice programmer to visualize in their mind as computers advance in complexity. Students feel they send their program into a "black box", and the results are spit back out. Learning concepts like memory management, logic design, and computational complexity are increasingly viewed by students as unnecessary nuisances instead of critical skills. For example, students who grew up using Google Drive are having an increasingly difficult time understanding basic concepts of traversing file systems since they've never had to use them.[6] CS1 and CS2 instructors have to devote valuable lab time to teaching students what a file system is, yet students increasingly question the importance of understanding UNIX file pointers and hierarchies, a pedagogical challenge unthinkable a mere 5-10 years ago. In hardware specific fields, this phenomenon is a major factor in the decline in Computer and Microelectronics Engineering major enrollments, which now barely consist of 10% of all computing majors.[7,8] Counterintuitively, the easier it has become for the public to gain access to computing devices, the more challenging it has become for aspiring computing professionals to apply spatial and temporal reasoning in developing programming proficiency.

Challenges of Using Music Theory as a Computing Composition Metaphor

The use of classical or contemporary music in a computational metaphorical framework has its own challenges and pitfalls. While there are correlations between graduation outcomes of musically-inclined students and computer science degree recipients, it is *not* a one-to-one mapping, so computer science instructors should not assume musical ability, proficiency, or theoretical understanding. Furthermore, some students simply find artists like Mozart, Beethoven, and Bach “boring”, and using their compositions in a metaphorical framework may lead to reduced cognitive engagement, especially in an 8am class. Any music-computing metaphorical framework in a CS0-CS2 course should introduce the music theory “just enough” for the student to correlate what they hear with a visualization of a correlated programming concept.

Some popular songs contain content that may be considered vulgar, politically divisive, or discriminatory by some students. Editing lyrics may assuage some students concerns, but other students view those modifications as an unethical form of artistic censorship. Other popular songs – while catchy and interesting to students – are popular in part *because* of their simplicity.[9] Concision, brevity, and digestibility drive modern pop music. Songs composed using a simplistic approach may increase student cognitive engagement, but may not be sufficient for use as a metaphorical tool for conveying a concept as fundamental, yet extensive as the levels of abstraction in a computing device. Songs that briefly attained popularity are often considered awkward, unappealing, or inappropriate by future generations of students, meaning that any metaphorical framework using recently popular songs may need to be rapidly edited or removed, adding excessive or unnecessary work to future lesson planning.

Music composed with the aid of computer software, such as Autotune, is disliked by many students who feel those artists are using computers to overcome talent deficiencies.[10] This apprehension to exploring the overlap between computers and music by many students is a barrier to effective cognitive engagement in a computer science class. Addressing this taboo is crucial, since technologies like Moog synthesizers use underlying computational frameworks and levels of abstraction similar to what CS students observe in their classes.

Target Audience and Pedagogical Benefits of Stevie Wonder’s ‘Superstition’

Because of this myriad of challenges, many CS instructors understandably avoid using music as a pedagogical tool in the introductory CS courses entirely, missing an opportunity to engage with students in a way that they enjoy and has meaningful parallels to effective programming practices. In this section, we will describe the motivation for specifically using Stevie Wonder’s hit song *Superstition* as a metaphorical tool in computation composition, as well as how the song addresses the concerns raised in the previous section.

Superstition,[12] the first single from Stevie Wonder’s album *Talking Book*, was the #1 hit song in 1972, was awarded multiple Grammy Awards, and was ranked as the twelfth-greatest song of all time by *Rolling Stone Magazine* as recently as 2021.[11] *Talking Book* was the second album in a five-album sequence commonly referred to by Mr. Wonder’s fans and music theorists alike as his “Classic Period” due to his mastery and application of music theory. The “Classic period” has been referred by several critics as “the greatest creative run in the history of popular music.”[13, 14] *Superstition* has inspired documentaries [15, 16], theoretical discussions [17], lectures in introductory music theory classes[18], and dissertations.[19, 20] *Superstition* has been performed live by Mr. Wonder in environments as ranging as the educational Sesame Street program[21] and for the President at the White House[22]. *Superstition’s* pedagogical benefits and appropriateness for the classroom are well investigated and vetted by academic, government, and cultural organizations.

In 1971, Stevie Wonder turned 21 years old, which meant that his contract with Motown Records was set to expire. He was frustrated with the creative and artistic limitations of his original contract,[23] and he expanded his natural musical talents in two ways that lend his music well to computational metaphorical frameworks. First, he felt ideologically limited by the traditional keys, rhythms, and harmonic progressions of the Motown quality control board, saying "I wasn't growing. I just kept repeating the Stevie Wonder sound and it didn't express how I felt about what was happening in the world."[24] So he took both traditionally classical and modern twentieth century music theory classes at the University of Southern California. He applied his new-found understanding of both classical and non-classical songwriting order to strengthen and complement his already-prolific harmonic and rhythmic composition skills.

Similarly, many modern programming paradigms have their foundations in legacy computing systems, and understanding their origins strengthens a student's capability for applying advanced concepts at different levels of abstraction. For example, studying the infamous *Civilization* video game integer underflow error leading to Mahatma Gandhi's game character – renowned for his doctrine of non-violent resistance – to have an "aggressiveness" score of 255 out of 10 instead of -1, or simply rounding off at 0[25] can drive home the importance of studying data types. This amusing and relatively harmless example opens the door for more serious examples, such as the integer overflow error that caused six thousand 911 calls to not be routed to Public Safety Answering Points[26] and the inertial guidance system overflow that led to the explosion of the European Space Agency Ariane 5 rocket[27]. Students gain a deeper appreciation for emerging overflow issues, such as the Year 2038-time epoch bug plaguing Unix-based systems.[28] During *The Superstition Lecture*, we review these lessons and note that understanding traditionally classical programming challenges will help them adapt to modern twenty-first century challenges, just like Stevie Wonder did when he took music theory classes. The lesson is that even the most gifted musicians and programmers can improve their composing proficiency by gaining a deeper understanding of the capabilities and limitations of their instruments.

The second innovation in Stevie Wonder's music that lends itself well to a computer science metaphorical framework is his adoption of synthesizers into his music composition. His feelings of creative confinement were not limited to Motown's compositional control or ideological restrictions. He felt there were sounds *in his mind* that he couldn't get out into the real world with Motown's orchestra or conventional instruments.[29] He was introduced to Malcolm Cecil and Bob Margouleff, who produced albums on the world's largest and most advanced music synthesizer at the time "The Original New Timbral Orchestra." (TONTO) The technical innovation in TONTO was the combination of Moog analog synthesizers (invented only 5 years earlier), undulating ARP synthesizers for tuning modular systems, and guitar technology developed by Jimi Hendrix for *Electric Ladyland*. While the first commercial albums using synthesizers were *Switched on Bach* by Wendy Carlos (1968) and *Zero Time* by Cecil and Margouleff themselves (1971), the first albums to gain commercial and critical success using synthesizer technology were Stevie Wonder's albums from his "Classic Period". Introducing TONTO in this way helps break down the taboo that students may have towards using technology in composing music.

More importantly, discussing Stevie Wonder's appreciation for TONTO is an opportunity to help students appreciate the foundations of computing itself. Malcolm Cecil recalled Stevie as saying "this is much more like the music that is in my mind" when he first started experimenting with composing music with TONTO.[29] The foundation of computing as a discipline is based on the study and mechanization of thought itself. The philosopher Thomas Hobbes wrote in his book *Leviathian* in 1641, "By ratiocination, I mean computation… all ratiocination is comprehended in these two operations of the mind, addition and subtraction." When Ada Lovelace wrote the sequence of operations for the Analytical Engine – widely considered the first computer program - the operations were addition, subtraction, multiplication and division (which were just repeated additions and subtractions), and storage[2], just as Hobbes posited. In the foundational paper of computer science, Alan Turing was studying the *Entscheidungsproblem*, which is German for "decision problem." And in 1958, when the members of the Association of Computing Machinery (ACM) debated the term that would be used for the emerging field (which eventually became *computer science*), two potential terms were "applied meta-mathematics" and "applied epistemology".[30] Based on these ideas, computer science is the mathematical theory of thought, computer engineering as the development of decision machines that effectively implement those algorithmic principles, and programming as the process of using the theory of thought to conduct the decision machine.

Stevie Wonder was leveraging *all three* paradigms when he recorded albums in TONTO during his "Classic Period". He was using a computing machine and his study of the science of harmony in concert, just as Ada Lovelace envisioned, which we may consider as a form of musical computing science. He worked with Cecil and Margouleff to modify TONTO's wiring to perform analog modulation to maximize the "funkiness" of his sound, which can be a musical computer engineering. By leveraging the technology of TONTO to overlay and compose melodies, he wasn't just playing music, he was programming the "music in his mind". It's not a coincidence that the first three albums he released while recording in TONTO were called *Music of My Mind*, *Talking Book*, and *Innervisions*. Specifically, on *Superstition*, Stevie Wonder is performing every instrument except the saxophone and trumpet, controlling every level of abstraction.

Target Audience, Prerequisites, and the Introduction of *The Superstition Lecture*

The Superstition Lecture has been presented to three sections of a CS2 Data Structures course taught primarily in C++ (395 total students), two sections of a CS1 Introduction to Computing course, where C and C++ are the primary languages (124 total students), and one section of a CS0 Principles of Computing for Freshman Business majors that covers Python, HTML, CSS, and a brief introductory sequence in PyRTL[29], a Python library for implementing hardware register transfer instructions (35 students). There are crucial fundamental concepts required for students to understand how to compose a computer program, regardless of programming maturity, so that the major concepts covered in *The Superstition Lecture* are relatable to material covered during the semester. But we note here that, because of the differences in topics and programming maturity across those courses, there are some necessary tweaks required between courses, which we outline below. We also assume no music theory prerequisite in the design of the lecture. In Table 1 below, we compare the levels of abstraction discussed in the CS courses with the levels of *Superstition*.

Superstition Lecture Outline

Superstition	Computing Abstractions
Application of TONTO	Operating Systems, Intersection of Levels of Abstraction
Vocals	Graphical User Interfaces, Data Structures, Paradigms, Algorithms
Saxophone and Trumpet	Object-Oriented Programming, Side-Channel Attacks
Clavinet Tracks and Cadential Progressions	Procedural Programming, Void and Cast Pointers Recursion, Graph Theory, AI
Funk Bass	Assembly Languages, Compiler Optimization, Finite State Machines, and Clocking
Funk Drum Figures	Computer Organization, Reduced Instruction Set Computer, Logic Design, Bootstrap programs
Music Synthesizers and Music Theory	Computer Science and Engineering, Logic Design, Turing Machines, Analog Devices

Figure 1: Comparison of Levels of Abstraction in Computer Science and Music Theory in *The Superstition Lecture*

The first part of the introduction of *The Superstition Lecture* is a brief discussion of Stevie Wonder's career, his study of music theory, and the history of TONTO. A brief clip of a documentary where Malcolm Cecil is interviewed discussing how Stevie Wonder used technology to perform "the music in his mind" is played.[30] A short clip of Cecil adjusting wires and knobs while Stevie Wonder performs[23] is played to emphasize the idea of music composition at different levels of abstraction on a synthesizer. In the CS0 course, the concepts of Analog and Digital signals are covered, so we review that material here. Next, we tell students that Malcolm Cecil was initially a Radar Technician in the British Royal Air Force and experimented with electronics and jazz based on what he learned in the military. The plugboards in TONTO are then compared to Alan Turing's Bombe machine developed to decode the Enigma cipher in World War II. Students learn that the same fundamental technology that enabled the creative process used by Stevie Wonder to record *Superstition* was the same that enabled the Allies employ the rapid counterintelligence used to ultimately defeat the Nazis, and that the differing approaches to programming on the same technology are known as programming paradigms.

For the CS0 course, we review the differences in implementation between Python, HTML, CSS, and PyRTL for introductory logic design. The CS0 students design their own basic Finite State Machines in this course, so this topic for the final exam is reviewed. For the CS1 course, we discuss the differences in the programming paradigms of the C and C++ languages in the context of procedural and object-oriented programming paradigms, and why they were studied. For the CS2 Data Structures students, a brief preview of the required *Programming Paradigms* course is presented, and which specific topics from the CS2 course will be crucial for their success.

Funk Drum and Figures, Computer Architectures, and Assembly Language

We use the funk drums in *Superstition* as a metaphor to help students understand the importance of understanding the underlying hardware in a computing device. In *Superstition*, the song begins with 8 measures of a famous funk drum beat. Stevie Wonder is playing the drums on a simple kit consisting of a bass drum, a snare drum, a high hat, and one cymbal, and he is only using three tracks to record the drums. And funk is all about timing. The drum beat is essentially a two-measure repeated drum phrase. The drum beat includes a bass "kick" on every beat (aka "four on the floor"), which means the bass drum is serving as the song's clock signal, with a snare drum on every other beat. But he is able to improvise flourishes with the confines of that steady beat in order to give the song depth and uniqueness.

In the CS0 class, the concepts of frequency, clock generation, and synchronization are covered at a high level, which introduces the opportunity to review those concepts. In the CS2 course, the concept of clock skew that they will cover in the Computer Architecture course is compared to Stevie Wonder's proclivity for improvising within the beat, yet maintaining the timing of the song. For the CS1 and CS2 students, I mention that they will study a Reduced Instruction Set Computer (RISC) in their Computer Architecture course, and that RISCs were designed to reduce the complexity of the hardware in exchange for reusing common assembly code instructions. Then, I review the memory management topics specific to those courses, indicating that simple microarchitectures are the underlying principles behind pointers, static and dynamic memory allocation, and stack and heap memory space. In the CS1 course, we review an assignment where students determined the difference in performance of storing the results of the Fibonacci sequence in static and dynamic memory.

Finally, the introductory drum solo is played again. I mention that this drum solo is being used to set up the funk rhythm for the rest of the song, and that operating systems in computers use bootstrap programs to perform a similar purpose. The introductory drum beat is 98 beats per minute, but the rest of the song itself is faster at 106 beats per minute. Likewise, a computer needs to load the BIOS, perform tests, instantiate simple driver programs, and load the entire operating system before running more complex programs. These are preview topics in the CS1 and CS2 class for the Operating Systems course for majors, and a review of the concept of bootstrapping covered in the OS overview lecture sequence in the CS0 course.

Next, we briefly cover that RISC architectures are dependent upon compiler optimizations which effectively translate high-level code into assembly and machine languages, just like funk drum beats and funk bass lines are interdependent. In each course, the students get a brief interaction with assembly language to introduce broader concepts of how code is interpreted by a computer (CS0), memory management and pointers (CS1), or how memory is dynamically allocated to build a data structure at run time (CS2). They do not write assembly language in these courses, but having some intuition of how to read assembly will help them with programming at different levels of abstraction. The purpose of the discussion of the bass in *Superstition* is to reinforce this concept.

The "bass" in *Superstition* is actually a clavinet track that Stevie Wonder plays instead of a bass guitar. The bass riffs are designed to build "anticipation" for the next cycle. There are variations on the riffs, but they fit within a clear pattern, a trait in funk music called *circumscribed variability*[18] Likewise, RISC architectures and assembly languages possess the trait of circumscribed variability through the implementation of Register (R-type), Immediate (I-type), and Jump (J-type) instructions. But they have specific constraints, such as the same size architecture, or having stages of the instruction completed by the next clock cycle. To drive this point home, we first play a sample of the bass clavinet track from *Superstition*, then the bass and drums together. In *Superstition*, the ostinato – the repeated musical phrase or rhythm - is established by the next 8 measures of the song with the bass clavinet track and the drums before any singing or horns are introduced. Then, we play the bass only with an animation of a set of assembly instructions. Finally, we play the bass and drums with an animation of a set assembly instructions and the representation of how those assembly instructions flow through a MIPS single-cycle architecture. The students can visualize the interaction of hardware and assembly language as the **hear** the interaction of the drum and bass clavinet track.

Clavinet Tracks, Cadential Progression, and Programming Paradigms

Superstition has three Clavinet tracks, as well as a delayed version of the main Clavinet riff, and Stevie Wonder strategically layers the tracks over each other to provide depth and feeling. The ostinato with the bass and drums provides the rhythm, but the clavinets are used to convey the foundations of the song's meaning. Likewise, a procedural programming language like C gives the programmer the capability to direct the computer to perform a task. In the CS1 and CS2 class, the casting of void pointers to types gives the allocated memory meaning in the context of the program. We convey this metaphor by playing the drums, bass, and one clavinet track as a void pointer, and then layer the main riff on top of them as an example of casting the pointer to a type, giving the song and program deeper meaning. In the CS0 course, they are shown an animation assembly language and basic Python programming fitting together in a similar manner.

In each class where we've introduced *The Superstition Lecture*, the main differences between procedural and object-oriented programming paradigms are detailed, although it is discussed in much more detail in the CS1 and CS2 courses, since both courses switch between C and C++ to emphasize course principles at different levels of abstraction. In the CS0 course, Python is the base language and supports procedural *and* object-oriented programming paradigms.

Next, we give a high-level overview of the composition of the clavinet tracks. In funk music, the groove is composed in order to compel the audience to anticipate the beginning of the beat to follow its ending. In essence, funk grooves *call themselves* until the pattern is no longer called.[17] At that point, I ask the students for a programming technique where a piece of code or task calls itself, unless it reaches a case where it no longer calls itself. In every class, several students immediately identify the solution as recursive calls. In Superstition, the clavinet tracks are used to promote the repetitive grove, and a stop-time effect followed by a crescendo is used to indicate the base case of the funk groove.

Finally, we select pairs of clavinet tracks to play together to show the students how Stevie Wonder "fit" them together. When select subsets of the tracks are played together, they can sound awkward, and students appear visibly confused. We explain that Stevie Wonder used his new understanding of music theory to combine the clavinet tracks to induce "ghost notes", where the listener "hears" notes in their own head that the artist *is not playing*.[14, 18] We use this opportunity to correct a common student misperception about artificial intelligence in computer science. AI algorithms are designed to derive approximate solutions that are not currently known, such as new art or essays. But these solutions are based on previous information, which means the computer is trying to return a solution that *isn't there*. We then present real-world ethical issues in AI, such as ChatGPT or using AI to select who to hire based on past hiring practices.

Horn Tracks, Object-Oriented Programming, and Side-Channel Attacks

The horn tracks are introduced as a metaphor for object-oriented programming. In the CS0 course, we review several Python libraries which allowed them to compose more robust and extensive programms, such as `os.path`, `Image`, `ipywidgets`, `PyRTL`, and `tornado`. In the CS1 and CS2 classes, this is an opportunity to discuss the differences between C and C++. We also use the horn tracks and vocals as a metaphor for the data structures covered in the course, as the horns and vocals are used to structure the meaning of the song. In the CS0 course, Python lists, dicts, and string operations are reviewed. In the CS1 course, we review the construction and freeing of memory when building singly and doubly linked lists in C++. And in the CS2 course, we review the lists, trees, queues, stacks, heaps, STL, and graph data structures covered in the class.

The lyrics of *Superstition* also reflect the recursive case and base case of the groove flow. When he is playing the "recursive case: groove, repetitive measures, he lists common superstitions. "Very superstitious, Writing's on the wall / Very superstitious, Ladders bout' to fall." But when he plays the "base case" groove, he implores with the listener to not fall for superstitions. As he sings "when you believe in things you don't understand / you will suffer", he builds up the tension that contrasts with regular groove. Then, implements the stop-time effect when he sings "Superstition ain't the way", immediately followed by a crescendo of horns, clavinet tracks, and drums. By singing the message of his song while playing the base case, he effectively contrasts that message with the superstitious behaviors he is critiquing in the recursive case.

Next, we show a visual representation of the horn track on Superstition (shown in Fig. 2 below). The large spikes represent the trumpet and saxophone sound, but notice on the left the small spike. We ask the students what they think that sound might be. After some guesses, we play the sound to reveal that is one of the horn players *coughing*. After the students laugh, we introduce the concept of side-channel attacks which use mathematical analysis of power spikes to steal information from a computing device, regardless of the secure software. They learn this is a common issue in smart card technology found on credit cards, and we share the story of how a North American casino was hacked when a side-channel attack was performed on an internet-connected fish tank.[31] The lesson imparted on the students is that software is not an amorphous blob. Software goes onto physical hardware, and understanding the entire computing ecosystem will strengthen their skills as a software engineer, even if they are remiss to learn hardware design.

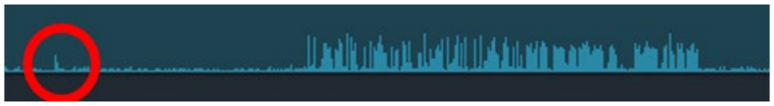
Figure 2: The cough (circled) in the horn track of *Superstition* used as a Side-Channel Attack Metaphor

Finally, we discuss how programming paradigms may be used in different ways to convey ideas, or as a different take on the same idea. *Superstition* was co-developed by guitarist Jeff Beck, who was collaborating with Mr. Wonder on songs in TONTO, and who wrote his own rock version.[13] Beck's version uses a reverb guitar to give a "superstitious" feeling. I point out that both Stevie Wonder and Jeff Beck use the same drum beat, and that funk drum beats can be used in both R&B and Rock to convey the same idea, just like different programming languages like Python, C, C++, or Java can use the same architecture to accomplish similar tasks.

Lecture Conclusion and Student Anecdotes

The *Superstition Lecture* is concluded with a discussion of a motivator for Stevie Wonder to learn music theory and experiment with TONTO in the first place. He didn't just feel ideologically limited in his music. He felt improperly compensated for his music. For each 98-cent album he sold, he made only 2 cents. But by programming his own music using TONTO, he was able to record *Music of My Mind* without the need for Motown's orchestra.[23,32] Since he was no longer under contract, he leveraged his skills to obtain unprecedented creative control and compensation for his music. We then tell the students that, even if they have their heart set on a computing specialization, that diversifying their understanding of computing will empower them to seek out a wider ranger of job opportunities, negotiate better salaries, and withstand shifts in the market.

The student feedback on *The Superstition Lecture* is primarily anecdotal and mostly euthanasic. Students regularly email feedback from when they remember a topic that we discussed in a future course or in an internship setting. Specifically, the CS2 students indicate the course helped them with their motivation for the subsequent *Programming Paradigms* course, and with picturing the movement of data with the MIPS architecture in the *Computer Architecture* course. Students in the CS1 course indicate they remembered lower-level concepts when they took the *Logic Design* course. And several CS0 students indicated on the Course Instructor Feedback that they loved the "unique and exciting" review. Occasionally, a student will indicate they did not like the lecture, and those comments are always couple with a comment similar to "I just don't learn very much from metaphors." Finally, several ROTC students indicated that they appreciated the connection between *Superstition*, the levels of abstraction, and military history.

References

1 Menebras, Luis and Ada Lovelace, “Notions sur la machine analytique de M. Charles Babbage”, Bibliothèque Universelle de Genève, Paris: Anselin, 1842, pp. 352-376.

2 “Ada Lovelace: how the maths genius saw the future of music”, Classical Music - BBC Music, Jan 31, 2023, https://www.classical-music.com/features/artists/ada-lovelace/.

3 Knuth, Donald, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, Mass, USA, 3rd edition, 1997.

4 K. Melymuka, “Why musicians may make the best tech workers”, CNN, July, 1998.

5 Jonas, Oswalt, “Introduction to the Theory of Heinrich Schenker: The Nature of the Musical Work of Art”, Musicalia Pr, 1934.

6 Chin, Monica, “File Note Found: A generation that grew up with Google is forcing professors to rethink their lesson plans”, The Verge, Sep 2021.

7 Dillinger, Tom, “A Crisis in Engineering Education: Where are the Microelectronics Engineers?”, SemiWiki, July, 2022.

8 Morrison, Matthew, “EDA Education – Challenges and Opportunities”, CadenceLIVE, September, 2022.

9 Pareles, Jon, “Want a Hit? Keep It Simple: Pop Songwriting and the Spirit of a New Simplicity”, The New York Times, December 31, 2010, https://www.nytimes.com/2011/01/02/arts/music/02simplicity.html.

10 Sclafani, Toni, “Ow, my ears! Autotune is ruining music”, NBC Today, June 2, 2009, https://www.today.com/popculture/oh-my-ears-auto-tune-ruining-music-1C9424663

11 “The 500 Greatest Songs of All Time: #12 Stevie Wonder, Superstition”, Rolling Stone Magazine, September, 2021.

12 Wonder, Stevie, “Superstition”, Album: *Talking Book*, Motown Records,

13 Hamilton, Jack, “Stevie Wonder’s Classic Period: The Greatest Creative Run in the History of Popular Music”, Slate Magazine, Dec 19, 2016.

14 Jones, Bomani and Spencer Hall, “The Right Time: Bomani Jones & Spencer Hall talk Kwame Brown and celebrate Stevie Wonder's birthday”, ESPN, May 19, 2021, https://youtu.be/iUcihzoAVe8.

15 “Songs that changed Music”, Produce Like a Pro, May 25, 2022, https://youtu.be/J9_S-RESrV8

16 Keene, K.W., “The Fascinating Tale Behind the Making of Stevie Wonder’s Superstition”, Rock Music History Lesson, March 2022. https://youtu.be/syRygJ3nlnY

17 “Understanding Superstition”, 12tone, July 3rd, 2020, https://youtu.be/3dhtX5c45so.

18 Lam, George, “Syllabus for MUS 208 – Music Theory I”, CUNY York College, August 2020, https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=1013&context=yc_oers.

19 Lovell, Jeffrey, “An Exploration of the Melody, Harmony, and Improvisation in the Music of Stevie Wonder”, Dissertation, University of Oregon, May 2012.

20 Hughes, Timothy, “Groove and Flow: Six Analytical Essays on the Music of Stevie Wonder”, *Dissertation*, University of Washington, 2003.

21 Stevens, Dana, “Stevie Wonder Week: When Stevie Rocked Out on Sesame Street”, Slate Magazine, December 2016.

22 Bloom, Julie, “Gershwin Prize for Stevie Wonder”, New York Times, September 2008.

23 “Meet TONTO, the machine behind Stevie Wonder's Superstition”, CBC News, November, 2018.

24 Gibbs, Mya, “A Story of Stevie Wonder’s Journey Through Creative Expansion”, Black Music Scholar,

25 Plunkett, Luke, “This is why Mahatma Gandhi is so evil in Civilization”, Quzart: The A.V. Club, November, 5, 2014.

26 “April 2014 Multistate 911 Outage: Cause and Impact.”, Public Safety and Homeland Security, Federal Communications Commission, October, 2014.

27 Gleick, James, “A Bug and a Crash: Sometimes a Bug is More than a Nuisance”, New York Times Magazine, December 1, 1996.

28 Wagenseil, Paul "Digital 'Epochalypse' Could Bring World to Grinding Halt", July 28, 2017.

29 Sherwood, Timothy, “PyRTL: register-transfer-level hardware design and simulation”, UC Santa Barbara ARCHLAB, 2021.

30 Porter, Martin and David Goggin, “TONTO: The 50-Year Saga of the Synth Heard on Stevie Wonder Classics”, Rolling Stone Magazine, November 13, 2018.

31 Schiffer, Alex, “How a fish tank helped hack a casino”, The Washington Post, July 21, 2017.

32 Tayson, Joe, “How Stevie Wonder Beat Motown Records”, Far Out Magazine, May 13, 2021.