Finding Optimal Synchronization Words for Digital Voice Protocols

An Exhaustive Search Yields Substantial Improvement Over Classical Sequences for Opulent Voice Protocol from Open Research Institute

By Michelle Thompson W5NYV

Abstract

Synchronization word selection is critical for reliable frame detection in digital communications protocols. While classical sequences like Barker codes are well-known for their autocorrelation properties, are they truly optimal for arbitrary lengths? This article presents a computational approach to finding optimal 24-bit synchronization sequences, achieving a Peak Sidelobe-to-Mainlobe Ratio (PSLR) of 8:1. This is 2.67 times better than concatenated Barker codes. We demonstrate that exhaustive search is feasible for moderate-length sequences. Does a much better PSLR translate to better performance improvements over traditional approaches? The answer depends on how you implement your sync word detector.

Introduction

In digital communications systems, synchronization words (sync words) are bit patterns decided upon and known in advance that mark the beginning of data frames. Receivers correlate, or match, the incoming bitstream against these expected sync words in order to detect frame boundaries. The quality of a sync word is characterized by its autocorrelation properties. Good autocorrelation means that we get a strong bit-to-bit match, or peak number of matching bits, when we exactly align our known sequence with the matching sequence in the incoming bit stream, and we don't get much correlation when it's not aligned exactly.

For example, if we are looking for a sequence, like 11111, and we compare it to our record of what we are looking for (also 11111), then we get the maximum number of 5 matches when it's perfectly lined up. This is what we call the main lobe. But, there's more to the story. We want weak responses at all of the other offsets of the known word to the word in the bitstream. Look again at 11111. We are shifting in our received bitstream one bit at a time, and moving it "past" the known word. So with one bit overlap we have one match. With two bits overlap we have two matches. With three we have three, and with four we have four. Then we get the maximum, which is five of five. Now we start shifting it away, and we get four, then three, then two, then one. In order to use this as a sync word, we'd have to insist on five matches to claim that we have things lined up. If there was any noise in our signal, and there is always noise, then the values that we receive over the air might cause some of these numbers to change, and we might get a

false alarm. The number of matches when a sync word is at any other offset other than perfectly lined up are very important to us. Those numbers of matches at all the other offsets are called side lobes. Just like in radio signals, we want a strong main lobe and smallest possible side lobes. We need to suppress our side lobes. 11111 does not have good autocorrelation.

The Challenge

For the Opulent Voice digital voice protocol, we needed a 24-bit synchronization word. The protocol operates most-significant-bit (MSB) first and requires excellent autocorrelation properties to maintain reliable frame synchronization in challenging propagation conditions. Our initial approach was to use the best sequences that we knew about, which are from a family of binary sequences called Barker codes. But, there are no 24 bit Barker codes. The longest Barker code is 16 bits. Therefore, we picked an 11 bit code and a 13 bit code and we stuck them together. This concatenated code seemed entirely reasonable given Barker codes' reputation for optimal autocorrelation properties. But was this truly the best we could do?

Background: Barker Codes and Beyond

What Are Barker Codes?

Barker codes are binary sequences with near-perfect autocorrelation properties. They were discovered by R.H. Barker in 1953. For a Barker code of length *N* bits, the autocorrelation side lobes are at most one bit. Compare that to the 11111 example above, where the max side lobe was four bits in the five bit code. Only 13 Barker codes are known to exist, with the longest being length 13. For arbitrary lengths like 24 bits, no Barker code exists. A common approach if you need something longer than 13 bits is to concatenate two shorter Barker codes, so that the total number adds up to the number of bits you need.

The PSLR Metric

We use Peak Sidelobe-to-Mainlobe Ratio (PSLR) to quantify sync word quality.

```
PSLR = Main Lobe / Peak Sidelobe
```

This is similar to the concept of directionality with an antenna. The main lobe value in PSLR is the autocorrelation result when the sync word has “zero lag”. We slide the received bit stream past the stored copy of the sync word, and measure how many bits match. Zero lag is the point in time where the sync word in the received bit stream is perfectly aligned with the copy of the sync word we’re looking for. Peak side lobe is the maximum value of the autocorrelation at any non-zero lag. This is the maximum number of bit matches at all the other offsets before and after perfect alignment. For our 11111 example, the main lobe was 5 and the peak side lobe was 4, for a PSLR of 5/4.

Higher PSLR indicates better discrimination between true sync and false detections. In the

bipolar representation used for signal processing analysis (where we use 1 and -1 instead of binary 1 and 0) the autocorrelation is computed as:

$$R(\tau) = \Sigma\ s(i) \times s(i+\tau)$$

In the Opulent Voice digital hardware implementation, the equivalent operation uses binary representation with logical operations to count bit matches. Bipolar or binary give the same results. Bipolar is traditional to use in signal processing, and binary is used in our Opulent Voice hardware descriptive language implementations. This choice, to match the demodulated bits, turns out to be an important one. Improving the sync word without improving the method of detection means that the increased quality of the sync word doesn't immediately translate to better performance.

Alternative Sequences Considered

Before resorting to an exhaustive search, we evaluated several classical approaches:

1. Single Barker Codes The longest Barker code (length 13) is too short for our 24-bit requirement. Padding with zeros destroys the autocorrelation properties.

2. Concatenated Barker Codes Combining Barker-11 (binary 11100010010) and Barker-13 (binary 1111100110101) yields 24 bits (binary 1110 0010 1111 0011 0101 or hex 0xe25f35).

- PSLR is 3.00:1 (4.7 dB). The main lobe is 24 but the peak side lobe is 8.

The concatenation works reasonably well, but the two constituent codes creates side lobes larger than the ±1 ideal.

3. M-Sequences (Maximal Length Sequences) M-sequences have perfect *periodic* autocorrelation properties. All side lobes equal 1 when treated as circular sequences. However, they only exist at lengths of powers of two minus one (2^n - 1, so 7, 15, 31, 63...). For sync word detection, we need *aperiodic* (linear) autocorrelation since we detect the sequence once, not repeatedly. This is an important distinction, because it's also why we don't use a Zadoff-Chu sequence.

Testing 31-bit m-sequences truncated to 24 bits, the best truncation PSLR: 24:21 (1.2 dB).

The "textbook perfect" periodic property doesn't translate to our one-shot detection scenario and the length constraint ($24 \neq 2^n - 1$) forces truncation.

4. Zadoff-Chu Sequences A common question: "Why not use Zadoff-Chu sequences? They have perfect autocorrelation and are used in LTE/5G!"

Zadoff-Chu sequences are indeed excellent when they are used for their intended application. However, they're fundamentally mismatched for binary sync words. First, they are complex-

valued (IQ samples) with constant amplitude. Our sync word must be binary. Quantizing Zadoff-Chu to binary destroys the "perfect" autocorrelation properties.

A 24-bit quantized Zadoff-Chu sequence (root 5) yields only 2.18:1 PSLR with peak side lobe of 11. This is worse than the concatenated Barker code.

Zadoff-Chu sequences have perfect *periodic* (circular) autocorrelation, assuming the sequence repeats infinitely. Sync word detection requires *aperiodic* (linear) correlation. We match the sequence once in a bitstream, not as a repeating pattern. The "perfect" property doesn't apply to one-shot detection.

Another issue is that using Zadoff-Chu properly requires complex baseband processing (I/Q), frequency offset estimation and correction, complex correlation hardware, and magnitude threshold detection

Our binary approach requires a few logic gates, bit counters, and simple threshold comparison.

Zadoff-Chu sequences are brilliant for orthogonal frequency division modulation (OFDM) cellular systems with frequency-domain processing and multiple access requirements. For simple binary sync words in time-domain correlation, exhaustive search of binary sequences provides better PSLR with far simpler implementation.

5. The M17 Case Study: Sync Word Length Tradeoffs

The M17 digital voice protocol provides an instructive comparison. M17 operates at 9600 bps and uses 16-bit sync words (versus our 24-bit requirement):

LSF/Stream frames: 0x3243 (digits of π) - PSLR: 2.67:1
Packet frames: 0xFF5D - PSLR: 2.00:1
BERT frames: 0xDF55 - PSLR: 2.00:1

Why 16 bits instead of 24? Sync word length is a design tradeoff. Opulent Voice has nearly six times faster bitrate, so there's a faster acquisition with 24 bits at the faster bitrate, even though the sync word is longer. The longer the sync word the higher the main lobe, but 16 bits and 24 bits can achieve the same PSLR. 16 bits has less noise immunity, but can be considered lower overhead.

A 16-bit sync word is a reasonable choice for VHF/UHF operation where SNR is typically higher and lower overhead is valuable. However, M17's selection of a sync word without optimization represents a very big missed opportunity.

What if M17 had optimized their 16-bit sync word?

Exhaustive search of all 65,536 possible 16-bit sequences reveals 80 optimal sequences with PSLR = 8.00:1 (peak sidelobe = 2). Example: 0x066b.

M17 could have achieved the same 8:1 PSLR as Opulent Voice's optimal 24-bit sequence, with

zero additional overhead, just by running an exhaustive search on their chosen 16-bit length.

Both M17 (16-bit) and OPV (24-bit) made reasonable length choices for their respective applications. The mistake isn't the length. It's failing to optimize for the chosen length. The sync word value was selected from the digits of pi, written out in binary. While irrational numbers appear to be random, we can see from the m-sequence evaluation above that random or pseudorandom numbers do not necessarily have good aperiodic autocorrelation at all. It's just luck that the PSLR is as high as it is for one of the three sync words. Using randomly selected "digits of π" left a lot of performance behind. The packet and BERT sync words have even higher side lobes than the LSF/Stream words, and lower PSLR. This degrades detection performance in multi-path environments by quite a bit, even if the signals are strong. VHF and UHF have multi-path, so this is a real concern.

6. P25 Protocol Case Study

Project 25 (P25) is a suite of standards for digital two-way radio communications widely used by public safety organizations in North America. Developed through joint efforts of APCO International, federal agencies, and standardized by the Telecommunications Industry Association (TIA), P25 Phase 1 represents one of the most mature and widely deployed digital voice protocols in the public safety sector. P25 Phase 1 operates at 9600 bits per second using either C4FM (Continuous 4-level FM) or CQPSK (Compatible Quadrature Phase-Shift Keying) modulation with a symbol rate of 4800 symbols per second.

The P25 protocol uses a 48-bit frame synchronization pattern (a hex value of 0x5575F5FF77FF). This is sent at the beginning of every message and is then inserted every 180 ms during voice messages. The purpose of the frame synchronization pattern in P25 is to enable operators to monitor or join a conversation in progress. This is an important system feature in public safety use cases. While P25 uses twice as many bits for synchronization as Opulent Voice, it transmits these 48 bits at 4800 symbols per second. This takes 5 milliseconds to transmit the complete sync word. Opulent Voice 24-bit sync word transmitted at 54.2 kHz takes approximately 0.44 milliseconds.

P25's design reflects its public safety heritage. The longer sync word and aggressive repetition prioritize reliability and rapid synchronization over spectral efficiency. The protocol accepts higher overhead (48 bits every 180 ms) to ensure first responders can reliably join active transmissions, However, protocols operating at higher data rates (like Opulent Voice at 54.2 kHz) can achieve equivalent time-domain robustness with shorter bit sequences, as a correlation gain comes from processing bandwidth rather than pattern length alone.

When we analyze for the PSLR of P25, we don't compare single bits for the sync word analysis because P25 is not a binary mode. P25 symbols are two bits instead of one bit. P25 receivers operate at 4800 symbols per second. There are two bits (dibits) per symbol, for a bit rate of 9600 bits per second.

The matched filter or correlator in the receiver processes 24 symbols, not 48 bits.

The dibit pattern looks like this:

+3 +3 +3 +3 +3 -3 +3 +3 -3 -3 +3 +3 -3 -3 -3 -3 +3 -3 +3 -3 -3 -3 -3 -3

From dibits of:

01 01 01 01 01 11 01 01 11 11 01 01 11 11 11 11 01 11 01 11 11 11 11 11

Resulting in a PSLR of 3:1 (4.7 dB). This is very reasonable for a manually-designed sync word from the 1990s.

The Computational Approach

Is Exhaustive Search Feasible?

For 24 bits, the search space contains 2^{24} = 16,777,216 possible sequences. Modern computers with optimized numerical libraries make this fast and easy. We benchmarked the autocorrelation computation and put a simple test in the Jupyter lab notebook. The rate was ~175,000 sequences/second (NumPy on a newer laptop), and total time was ~96 seconds (1.6 minutes). This is a fully feasible search for ordinary computers.

Search Implementation

The search converts each integer from 0 to 2^{24}-1 into a bipolar sequence, computes its autocorrelation, and tracks sequences with maximum PSLR.

The bit ordering must match your protocol (most significant bit vs. least significant bit).

Results

The exhaustive search completed in 73 seconds and found 6,864 sequences with optimal PSLR of 8.00:1, or 18.1 dB.

Some of the top optimal sequences:

Hex Value	PSLR	Peak Sidelobe
0x00e564	8.00:1	3
0x7006ca	8.00:1	3
0x268b00	8.00:1	3
0x2e9c80	8.00:1	3
0x3c9a80	8.00:1	3

All optimal sequences share the same peak side lobe magnitude of 3, compared to 8 for the concatenated Barker code.

Analysis and Verification

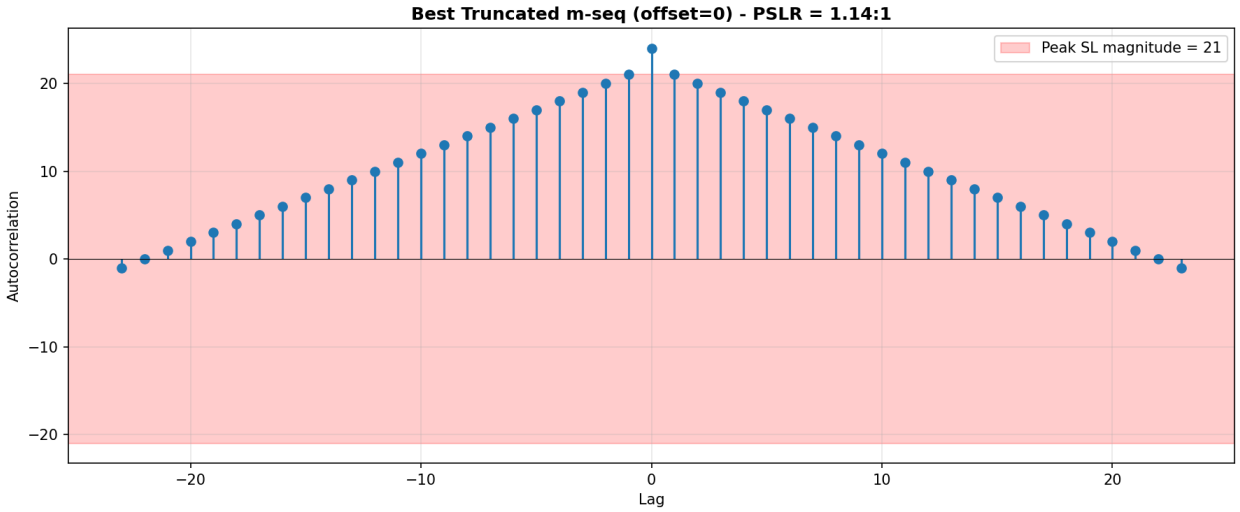
Sequence	PSLR	PSLR (dB)	Peak Sidelobe	Relative Performance
Concatenated Barker (0xe25f35)	3.00:1	9.5 dB	8	Baseline, ok
Best m-sequence truncation	1.14:1	1.2 dB	21	Really bad
Optimal (0x7006ca)	8.00:1	18.1 dB	3	Really good

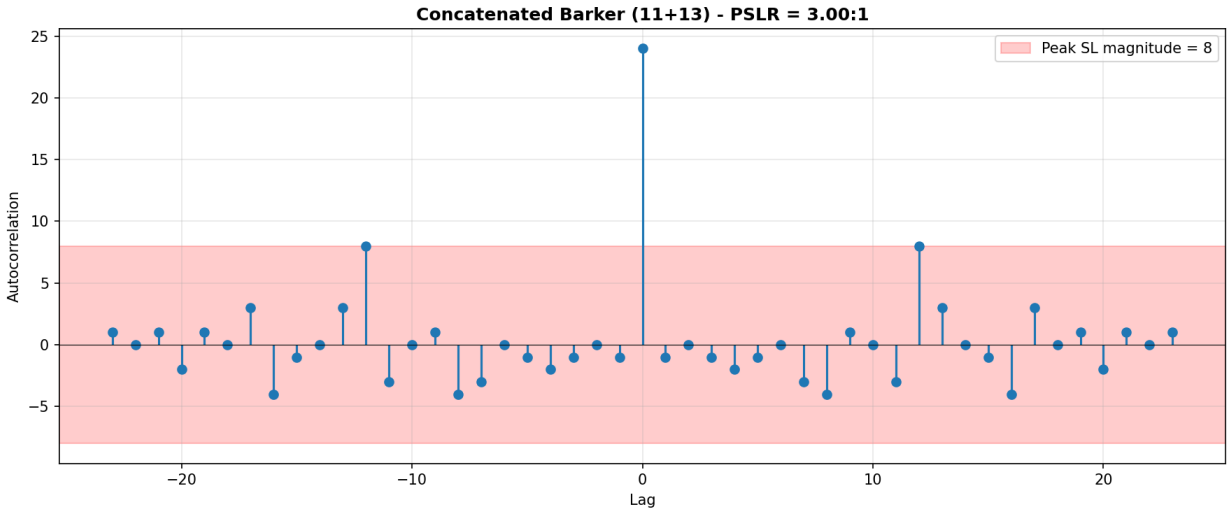
Autocorrelation Visualization

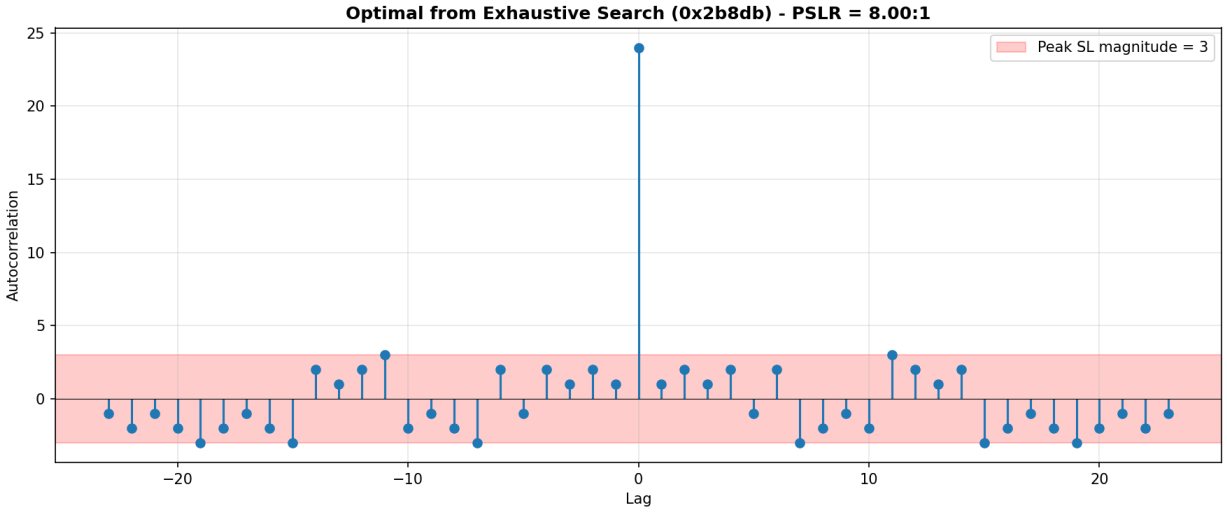
Figure 1 shows the autocorrelation functions for the three sequence types. The optimal sequence exhibits sharp main lobe at lag 0 (amplitude 24), suppressed side lobes (maximum ±3 in bipolar format), and a symmetric structure.

The improvement is dramatic: peak side lobes were reduced from ±8 to ±3, while maintaining the same 24-bit length and full main lobe amplitude.

Figure 1 three_way_autocorrelation_comparison.png







Practical Considerations

Implementation in Digital Hardware

While our analysis uses bipolar (±1) representation for mathematical convenience, hardware implementation typically uses binary logic with XOR operations. See the current implementation of sync word frame detection here: https://raw.githubusercontent.com/OpenResearchInstitute/pluto_msk/refs/heads/main/src/frame_sync_detector.vhd

Choosing a Detection Threshold

With PSLR = 8:1 and peak side lobe = 3, we can tolerate up to 3 bit errors while maintaining reliable sync detection at the bit level. This is using Hamming Distance as a function. Hamming Distance is the number of positions in a sequence that differs from another sequence we are comparing it to. For example, 11111 compared to 11001 has a Hamming Distance of 2.

A Hamming Distance of 3 means that 3 out of the 24 bits might be in error. We want to ensure detection even in this case of three errors. The sync word is not protected by forward error protection, as it is added after the payload is processed through randomization, forward error correction, and interleaving.

Errors	Correlation	Detection
0	24	Strong
1	22	Strong
2	20	Strong
3	18	Practical limit
4+	≤16	Risk of false detection

A threshold of 18 (Hamming distance ≤ 3) provides robust operation while minimizing false positives.

Which Optimal Sequence to Choose?

With 6,864 optimal sequences available, selection criteria include:

DC balance: Choose sequences with as equal numbers of 0s and 1s as possible
Run length: Try to avoid long strings of consecutive identical bits to relieve pressure on the tracking loops
Spectrum: Sequences with more transitions help some modulations in terms of detection
Aesthetics: Round hex values are easier to remember and type

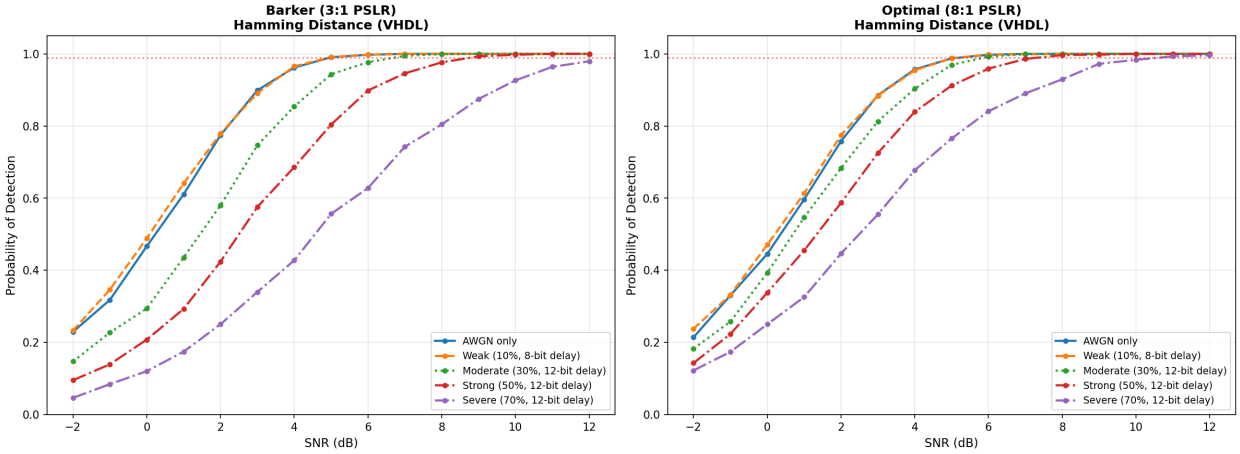
For Opulent Voice, we selected 0x02b8db. It's got ideal PSLR of 8:1, very low DC bias with 11 ones and 13 zeros, and a maximum run of 6 zeros in a row. These are good values, and the mnemonic can be "oh to be eight db".

Performance

Better PSLR provides better performance in two situations.

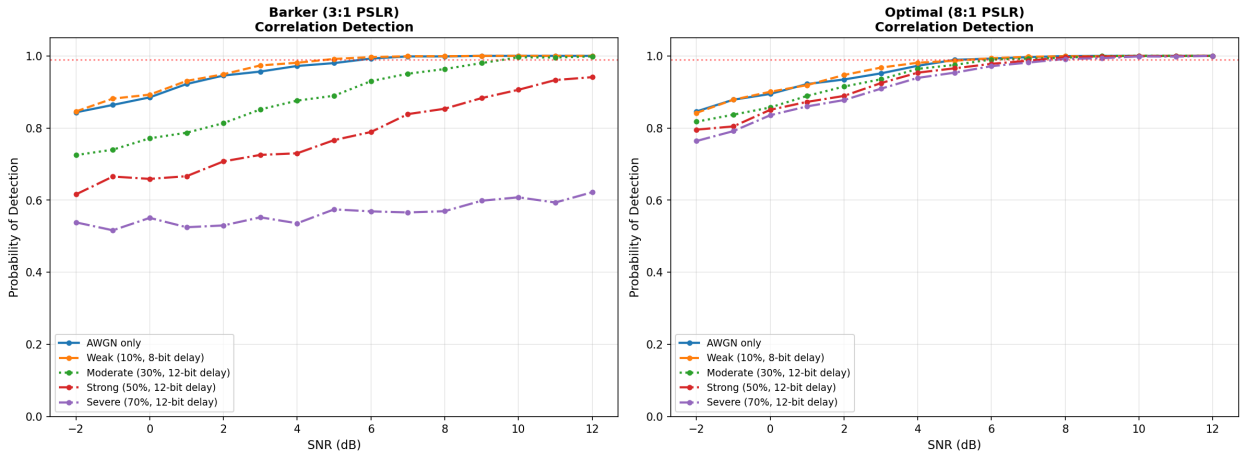
First, with multi-path conditions. Terrestrial features cause a particular type of interference where delayed copies of the signal arrive at the receiver. These echoes are from reflecting off of surfaces and taking longer paths than the line of sight transmission. Delayed copies of signals can destructively add and attenuate a signal. Worst case, they can largely cancel it out. VHF, UHF, and microwave communications systems quite often have to deal with multi-path, and many techniques have been developed to mitigate the damage. The better PSLR sync word, the better the multi-path performance. We show our optimal sync word compared to the concatenated Barker code using Hamming Distance sync detection. We vary the multi-path from none to severe.

Figure 2 hamming_detection_all_conditions.png



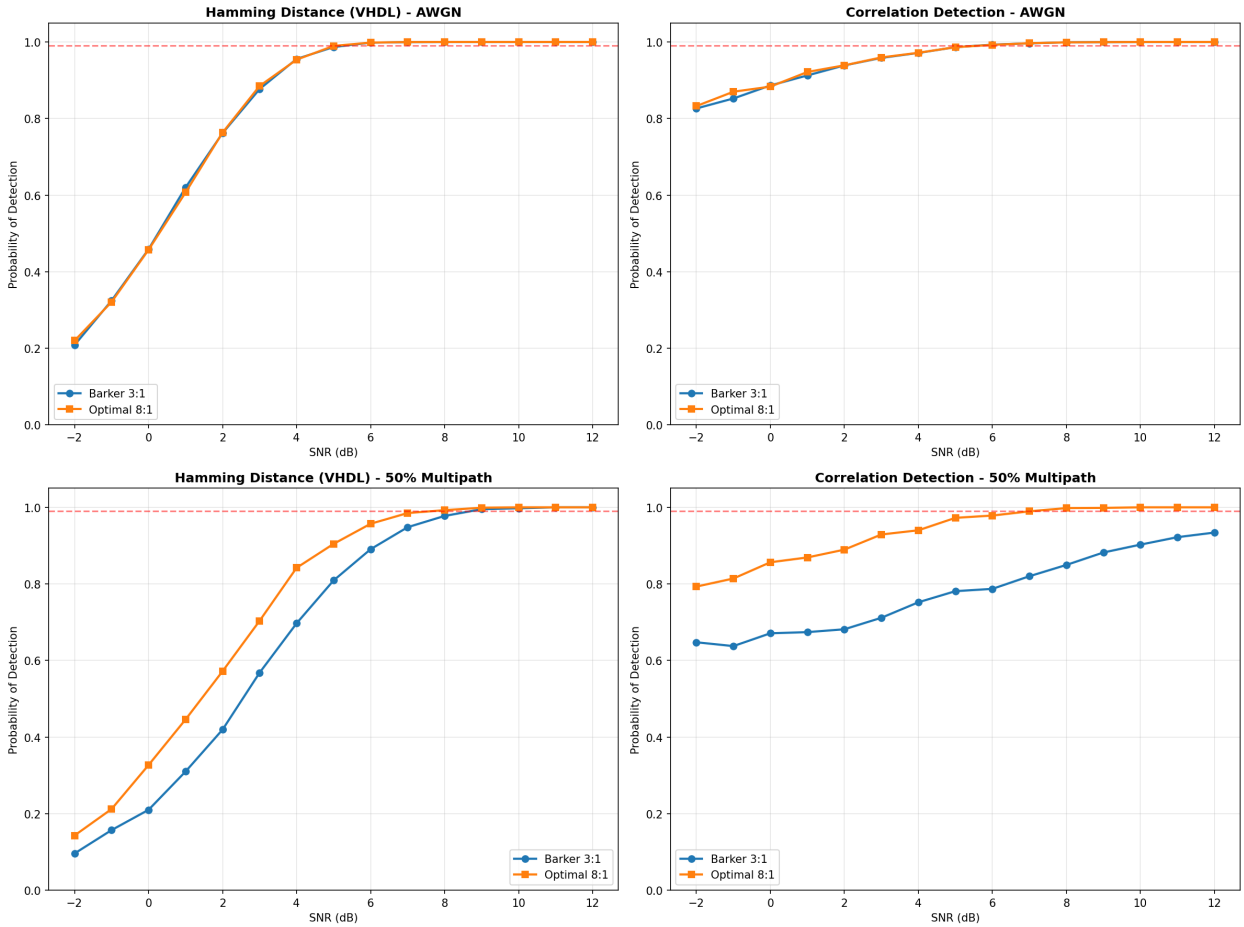
Second, if a correlator is used instead of a Hamming Distance calculator. If we use the more advanced correlation technique, then we see a performance improvement.

Figure 3 correlation_detection_all_conditions.png



Setting an optimized sync word in the protocol makes Opulent Voice future-proof. Multi-path performance significantly improves regardless of whether implementations use a Hamming Distance or a correlator to detect the sync word. If a correlator is used instead of a Hamming Distance calculation, then both classical concatenated Barker code and the optimal code have a significant performance improvement in additive white Gaussian noise (AWGN). The optimal sync word has a slight performance edge in very low SNR conditions. Higher PSLR doesn't give us much traction against AWGN. We check against AWGN to make sure we don't lose performance compared to other sequences. PSLR gives a substantial increase in performance when it comes to multi-path, and using a correlator improves both concatenated Barker and optimal sync word performance.

Figure 4 side_by_side_comparison.png



Lessons Learned

1. "Textbook Perfect" Isn't Always Practical

M-sequences and Zadoff-Chu are famous for their perfect autocorrelation properties, but this applies to *periodic* correlation in circular systems (like CDMA). For one-shot sync detection (aperiodic correlation), m-sequences offer no special advantage and are constrained to specific lengths, and Zadoff-Chu offers no special advantage and only works for complex samples. Concatenated Barker codes have drawbacks when it comes to multi-path.

2. Exhaustive Search is Underutilized

For moderate-length sequences, exhaustive search is entirely practical with modern computers. It guarantees finding the global optimum and eliminates guesswork about whether better sequences exist.

3. Classical Sequences Are Not Optimal for Arbitrary Lengths

While Barker codes are optimal for their specific lengths, extending them to arbitrary lengths through concatenation or truncation doesn't mean they're optimized for the new length. For custom lengths, dedicated optimization yields real improvements.

4. Bit Ordering Matters

Implementation must carefully match the bit ordering used during optimization. Our MSB-first protocol required specific code changes to ensure the hex value matched the transmitted sequence.

5. Symbol Rate Matters

Always analyze sync words at the level they were designed for. For binary PSK or MSK, analyze bits. For QPSK or 4-ary FSK, if the sync word is defined and transmitted at the symbol rate (dibits for P25), then analyze it at that symbol rate. For 8-ary PSK, if the sync word is defined at the symbol rate, then analyze tribits. And, so on, up the modulation order. If the sync word is defined at the demodulated bit level, then analyze it there.

Conclusion

This work demonstrates that exhaustive computational search can be done to provide optimized synchronization words that can outperform classical sequences in current and future designs. For the Opulent Voice protocol's 24-bit requirement, we achieved an 8:1 PSLR, which was substantially better than concatenated Barker codes and much better than truncated m-sequences.

The methodology was straightforward. We defined the search space (all 2^n sequences). We computed PSLR for each sequence. We selected from the list of all optimal sequences based on practical criteria, including spectral performance and prioritizing more frequent transitions over long runs of zeros or ones in the sequence in order to improve tracking loop behavior.

For sequence lengths up to 32 bits, this approach is entirely practical on modern computers and guarantees finding the global optimum. The resulting sequences provide significantly better performance than classical alternatives against multi-path and when using correlators in the receiver. Don't assume arbitrarily picked or classical sequences are optimal. Run exhaustive searches, verify independently using different implementations, understand where you are going to see performance wins, document the bit ordering clearly, and test in hardware to confirm that the correlation properties transfer correctly.

The code for this analysis is available at https://github.com/OpenResearchInstitute/interlocutor/blob/main/OPV_sync_word_study.ipynb and can be adapted for other sequence lengths and design constraints. All images in this article are from this Jupyter notebook.

Acknowledgments

Thank you to the amateur radio digital voice community for feedback on sync word selection, and to Matthew Wishek, Paul Williamson, Anshul Makkar, Jonathan Naylor, Wally Ritchie (SK), and Frank Brickle (SK) for helpful discussions about Opulent Voice protocol work.

References

1. Barker, R.H. "Group Synchronization of Binary Digital Systems." *Communication Theory*, Academic Press, 1953.

2. Golomb, S.W., and Gong, G. "Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar." Cambridge University Press, 2005.

3. Sarwate, D.V., and Pursley, M.B. "Crosscorrelation Properties of Pseudorandom and Related Sequences." *Proceedings of the IEEE*, vol. 68, no. 5, May 1980.

4. Skolnik, M.I. "Introduction to Radar Systems," 3rd Edition. McGraw-Hill, 2001. (Chapter on pulse compression and correlation is excellent)

5. Levanon, N., and Mozeson, E. "Radar Signals." Wiley-IEEE Press, 2004.

6. Ipatov, V. "Spread Spectrum and CDMA: Principles and Applications." Wiley, 2005.

7. Opulent Voice Protocol specification https://github.com/OpenResearchInstitute/interlocutor/blob/main/opulent_voice_protocol.md

8. TIA-102.BAAA-A, "Project 25 FDMA - Common Air Interface"

9. Daniels Electronics Ltd., "P25 Radio Systems Training Guide," TG-001, 2004

10. RadioReference.com P25 technical discussions and decoder analysis

11. APCO International, Project 25 Standards Documentation

About the Author

Michelle Thompson W5NYV@arrl.net is an electrical engineer and amateur radio operator specializing in digital communications. She holds an MSEE in Information Theory from USC and has contributed to a wide variety of open source digital radio projects and amateur radio organizations.