Adiabatic Quantum Computing for Finding Low-Peak-Sidelobe Codes

Gregory E. Coxson
Connie R. Hill
Lockheed Martin Advanced Technology Laboratories, Cherry Hill, NJ
Jon C. Russo
Lockheed Martin Advanced Technology Laboratories, Cherry Hill, NJ

***Abstract*—Results are presented for an adiabatic quantum algorithm to compute low peak sidelobe binary and unimodular codes on a D-Wave 2 quantum computer. The quantum algorithm is benchmarked against a conventional genetic algorithm (GA). The quantum algorithm shows roughly a 100 times speedup relative to the GA for binary codes longer that 100 bits and is capable of producing low sidelobe binary codes up to length 426 on the current D-Wave 2 hardware. Results are presented for Doppler tolerant binary and quad-phase codes computed using the same quantum algorithm.**

Keywords: Autocorrelation sidelobes, quantum computing, Ising spin-glass model, genetic algorithm, Doppler tolerance, D-Wave, binary code, unimodular code.

I. Introduction

Codes with low peak sidelobe level are desired in applications such as radar and communications where Match Filtering is used for detection [1][2]. For a given length, it is useful to know the lowest achievable Peak Sidelobe (PSL), and some or all the codes which achieve it. Although there exist some well-known construction techniques for codes with low sidelobe levels, often the lowest-PSL codes must be found by random or exhaustive searches. As code length grows, random and exhaustive searches become prohibitively computationally costly.

In recent years, the development of adiabatic quantum computing technology has offered an intriguing, although still mostly unproven, avenue to finding low-sidelobe codes of longer lengths. The resolution of this question may not be a necessary step before exploiting this technology for the search for low-sidelobe codes and similar problems of high computational complexity.

This paper is intended to show that adiabatic quantum computing offers a useful alternative to standard approaches in some cases. In particular, it is shown that the D-Wave quantum computer is capable of computing low-sidelobe binary or unimodular codes of medium-to-long lengths (say, $N > 100$) for a similar computational effort and computation time required by a standard computer to find low-sidelobe codes for lower lengths (roughly $N \leq 70$). While the codes returned will not always have the best possible sidelobe level, the ability to find low-sidelobe codes at longer code lengths, where standard approaches remain impractical, will be attractive for some applications.

978-1-4799-6233-4/14/$31.00

II. Notation and Terminology

Let

$$x = [x_1, x_2, \ldots, x_N] \tag{1}$$

denote an N-length unimodular code each of whose elements resides on the unit circle. A special case are the m^{th}-root-of-unity codes, whose elements take values from the set of m^{th} roots of unity, $\{e^{2\pi i/m}, e^{4\pi i/m}, \ldots, 1\}$. The setting $m = 2$ corresponds to the class of binary ± 1 codes whose elements can take values either 1 or -1. The non-binary ($m > 2$) unimodular codes include the quad-phase codes ($m = 4$).

The aperiodic autocorrelation (AAC) sequence for an x has length $2N - 1$ and is defined by

$$\mathrm{A_x} = \mathrm{x} * \overline{\mathrm{x^c}} \tag{2}$$

where $*$ means acyclic convolution, $\overline{x}$ means the reversal of a code x, and x^c means element-wise complex conjugation. The elements of the AAC of x may be represented explicitly in terms of sums of pairwise products of elements of x in the following way:

$$\mathrm{A_x(k)} = \sum_{\mathrm{i=1}}^{\mathrm{N-|k-N|}} \mathrm{x_i x^c_{i+|k-N|}} \tag{3}$$

for $k = 1, \ldots, 2N - 1$.

The "peak" of the autocorrelation is $\mathrm{A_x(N)}$. The peak is always equal to N, since

$$\mathrm{A_x(N)} = \mathrm{x_1 x_1^c} + \ldots + \mathrm{x_N x_N^c} = |\mathrm{x}|^2 = \mathrm{N}. \tag{4}$$

Elements for indices $k \neq N$ are referred to as "sidelobes" of the autocorrelation. The autocorrelation is symmetric with respect to the peak; that is,

$$\mathrm{A_x(k)} = \mathrm{A_x^c(2N - k)} \tag{5}$$

for $k = 1, \ldots, 2N - 1$.

The "peak sidelobe level" for a code x is defined to be

$$\mathrm{PSL_x} = \max_{\mathrm{k \neq N}} |\mathrm{A_x(k)}|. \tag{6}$$

The lowest achievable value of $\mathrm{PSL_x}$ is 1 for unimodular codes x is 1. This is because when $k = 1$ or $k = 2N - 1$,

the sidelobe is a $x_1 x_N$, so its modulus is 1. The binary codes x that achieve $\mathrm{PSL_x} = 1$ are named Barker Codes, after the author of an early paper identifying these codes [3]. When $m > 2$, unimodular codes x that achieve $\mathrm{PSL_x} = 1$ are called Generalized Barker Sequences or Polyphase Barker Sequences [4][5].

Another metric is integrated sidelobe level (ISL), defined as

$$\mathrm{ISL_x} = \sqrt{\sum_{\mathrm{k}=1}^{\mathrm{N}-1} \left(\mathrm{A_x(k)/N}\right)^2}. \tag{7}$$

It is related closely to the Merit Factor (introduced by Golay [6]),

$$\mathrm{MF_x} = \mathrm{N}^2 / \left(2 \sum_{\mathrm{k}=1}^{\mathrm{N}-1} \left(\mathrm{A_x(k)}\right)^2\right). \tag{8}$$

III. Conventional Approaches

The search for low-sidelobe phase codes has decades of history, going back to the early days of radar. It began with the development of pulse compression to solve the trade-off between achieving long detection ranges and resolving closely spaced objects with systems limited in transmitter power [1][2]. An attractive aspect of pulse compression is that low-sidelobe codes may be developed off-line rather than in real-time. A variety of methods have been tried, some based on algebraically supported constructions (for instance, the maximal-length shift register sequences [7][8][9]) and others on searches. This paper focuses on search methods.

The simplest or most obvious approach is "brute-force" exhaustive search of the space of possible codes. This is often a necessity when optimizing the PSL, for which the objective surface is characterized by sharp peaks and valleys with many local minima; this kind of search landscape is often described as "fractured". Millitzer *et al* [10] likened the search for low-PSL codes to looking for a needle in a haystack. For a given code length N, the number of codes in the binary search space is 2^N and the number of codes in the m^{th} root of unity search space is m^N; hence, computational effort tends to grow exponentially with code length.

Another approach is to apply random search, where the N code elements are chosen randomly from the set of m choices. Rather than exhaustively searching the space, the search is continued until a reasonable fraction of the search space can be assured to have been checked. While this is a simple, intuitive approach, it tends to suffer from a similar combinatorial explosion as exhaustive search since as the code length grows, the search effort must either grow apace or become increasingly unable to produce results.

Another possibility is the use of genetic algorithms. These are iterative methods in which at each stage the best members of the population are found and used (often in pairs) to produce the members of the next generation. The success of these methods tends to depend on finding powerful methods for combining pairs of good codes. While an intuitive and interesting pursuit, these methods have not provided the best results historically.

Related to genetic algorithms are the evolutionary algorithms. These include a variety of algorithms, including such popular methods as simulated annealing [11] and Great Deluge [12], often inspired by industrial or natural processes. In recent years, evolutionary algorithms have started to provide an alternative to exhaustive and random methods for code lengths beyond about $N = 70$ [13].

In some hybrid approaches an advantage is achieved by noting that different sidelobe metrics achieve minima or near-minima at the same code or at closely-spaced codes. This is one property exploited in [14], where optimization switches between PSL and Integrated Sidelobe (ISL). When using ISL, gradient-based methods exploit the relative smoothness of the optimization surface; when using PSL, the search crawls along "valleys" in the fractured optimization landscape.

Branch-and-bound approaches tend to work well for low-sidelobe code searches. They take advantage of the property of autocorrelation sidelobes that the outermost sidelobe depend on outermost elements of the code. In the branching, a candidate code is formed element by element from the outside in; at each step the partial autocorrelation is computed. If any sidelobe is worse than a desired threshold, the process backs up to the last branch, or decision, point. This is the approach used in [15] and [16].

All of these conventional search techniques have limited use as the code length grows beyond about 70 to 100. Unfortunately, there exist applications requiring longer codes with low sidelobe levels. For these, adiabatic quantum annealing seems to offer some promise of extending the computational frontier.

IV. Quantum Approach

In adiabatic quantum annealing, the approach is to find the lowest-energy configuration of a system constructed so that this lowest-energy state solves a problem of interest. Typically, the system is created by forming appropriate connections (or "wiring") an arrangement of quantum bits, or "qubits," which are two-state quantum mechanical systems. This computational framework has a close relationship to the well-known Ising spin glass model [17][18][19][20]. The Ising model involves finding the set of spins $\overline{s} = \{s_1, s_2, \ldots, s_n\}$ to minimize an energy function

$$E(\overline{s}) = \sum_{i,j} J_{ij} s_i s_j + \sum_{i \in V} h_i s_i \tag{9}$$

where J_{ij} encodes the interaction field between neighboring spins s_i and s_j; V is the set of indices of molecules in the spin system; and h_i is the strength of the applied magnetic field at the i^{th} molecule in the system.

If adiabatic quantum computing has close ties to the Ising spin glass model, this can also be said about the Ising spin glass model and the problem of finding low-sidelobe binary codes [21][22]. The terms in the first sum in equation (9) are pairwise products of ± 1 spins. This would seem to correspond well to the problem of minimizing autocorrelation sidelobes, where the individual sidelobes can be written as sums of pairwise products of binary or unimodular bits. Depending on the sidelobe metric chosen, there remains work to do in matching the energy function $E(\overline{s})$ to the desired objective function.

Lockheed Martin Advanced Technology Laboratories has access to a 512-qubit D-Wave 2 quantum computer. This second-generation system, code-named Vesuvius, is the successor of the D-Wave 1 computer which only had 128 qubits. Each of the qubits in the device is made of a Superconducting Quantum Interfering Device (SQUID). Neighboring qubits have configurable coupling elements which provide the programmability to target different problems. For more detail on Vesuvius see [23].

The algorithm developed for autocorrelation sidelobe minimization on the D-Wave systems is a form of genetic algorithm, given the name WAM, for "Whack-a-Mole." The name, while tongue-in-cheek, describes the challenge of achieving uniformly low autocorrelation sidelobes. That is, as sidelobes of greater size are decreased, others tend to increase, or "pop up," like moles in the familiar boardwalk game. Like most genetic algorithms, WAM maintains a population of solutions. At each step, it generates new solutions by a crossover mechanism. The D-Wave computation is used as a directed mutation engine.

For comparison, a GA was run using MatLab on a standard computer (DELL Latitude E6400 laptop), and applied to the same optimization tasks as WAM. GA starts with a population built of several candidate codes. An initial popularion of 1000 random codes was used. From this initial set, the top half are kept according to their sidelobe performance and the bottom half are discarded. From the top subset, new codes are produced to repopulate the discarded portion by so-called mutation and crossover operations. Mutation, in this implementation, consists of taking a parent code and probabilistically flipping bits to result in a new "child" code. Crossover takes two parent codes and combines them by splicing, interleaving, or modulo-2 addition to produce an offspring. Uniqueness is optionally enforced to ensure that no two codes are identical within an interation. The process of successive ranking, sub-selection, and offspring production is repeated until a desired PSL is achieved. In practice, multiple instances of the algorithm may be run concurrently on isolated "islands" to ensure broader coverage of the search space and faster results.

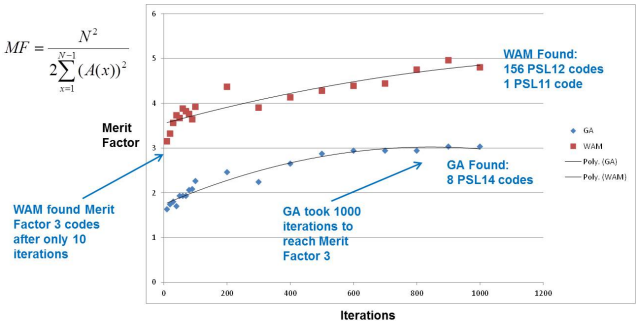

Figure 1. Merit Factor achieved versus number of iterations for WAM and GA algorithms.

V. Sidelobe Optimization Performance for Binary Codes

Figure 1 shows a comparison in Merit Factors achieved versus number of iterations for WAM, and for a GA running on a standard computer. It shows that the WAM algorithm found much better codes for the same number of iterations, both in terms of Merit Factor and PSL. For a code length of 256, WAM found codes with PSL 11 and 12 after 1000 iterations, while the lowest PSL found with GA was 14. WAM found codes with Merit Factor near 5, while the best GA achieved was a Merit Factor of about 3.

Figure 2 shows the PSL values achieved for five different lengths (64, 105, 128, 256 and 426) using WAM and the benchmark GA. The algorithms were run 1000 iterations and the best PSL values plotted, along with an estimate of the optimally low PSL for each case [Minimum peak-sidelobe-level (MPSL)]. The plot shows that a growing performance edge for WAM with increasing code length. SEED is the MLS (Maximal-Length Sequence) starting point for WAM.

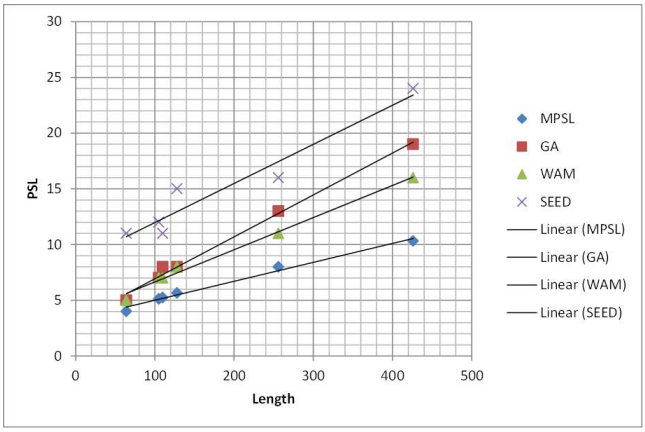

Figure 2. PSL values achieved with WAM and GA after 1000 iterations, for five code lengths.

VI. Computational Efficiency and Scaleability

Figure 3 displays a comparison of time per iteration for WAM with GA on a conventional computer. WAM runs on a combination of D-Wave and conventional computer. As mentioned already, the D-Wave 2 quantum computer operates as the mutation engine for WAM. Pre- and post-processing is approached using MatLab on a standard laptop. Timing for the conventional computer component of WAM processing is shown with red squares, while timing for the D-Wave 2 (Vesuvius) component of WAM processing is plotted in green triangles. Total running time for WAM is found by adding the times represented by the red squares and green triangles for each code length. Time for GA is shown with blue diamonds. Times are averaged over 1000 iterations.

The conventional computer times for GA show the familiar exponential growth with code length, while for WAM, the conventional component of per-iteration processing times shows a decreasing rate of growth with code length. Meanwhile, the Vesuvius component of WAM run times remains relatively constant with code lengths, starting at about 135 milliseconds per iteration for length 64 and growing only to 155 milliseconds per iteration for length 426.

Use of Vesuvius limits the code length that can be handled, to at most 426, due to a combination of overhead operations and qubits not available for use. A 2048-qubit D-Wave should

be available in several months, raising this limit on code length to about 2000.

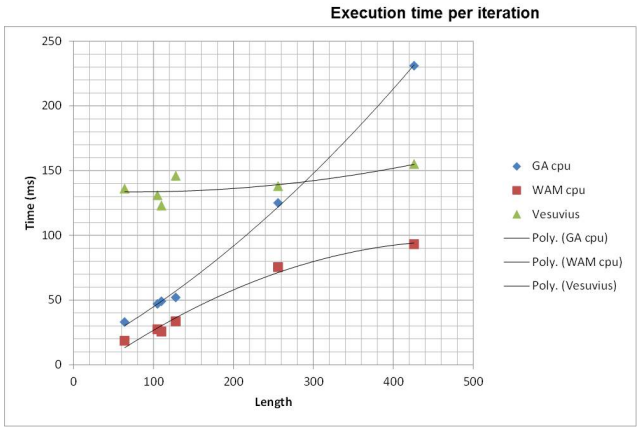

Figure 3. Execution time per iteration of WAM and the benchmark genetic algorithm.

VII. Optimizing Doppler Tolerance

A study was also performed to determine whether Doppler tolerance could be optimized, while maintaining reasonably low sidelobes at zero Doppler. The optimization was initiated on bi-phase and quad-phase seed sequences. Separate MLS codes were used for the real and imaginary parts of these sequences. Then WAM was applied to achieve an optimal tolerance to Doppler shifts up to a specified bound.

Figure 4 shows the performance achieved for length-64 bi-phase codes. When codes are optimized for Doppler tolerance at Mach 10, the optimal sidelobe level falls short of the optimal PSL of 4 achieved for zero Doppler. However, a value of about 7 is achieved and maintained out to Mach 10. Beyond Mach 10, the sidelobe performance rapidly worsens yet remains better than MLS codes to Mach 20.

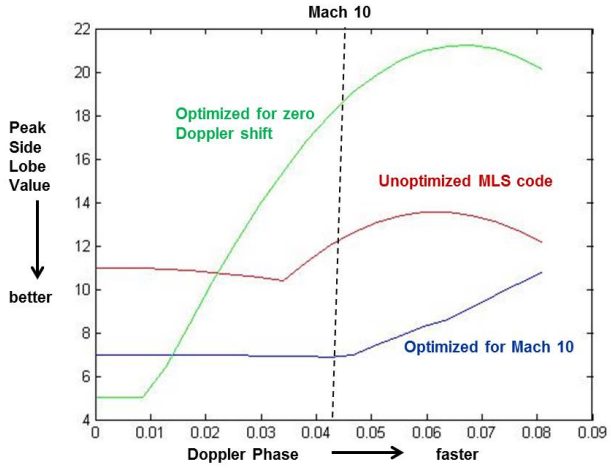

Figure 4. Peak Side Lobe values for optimized, unoptimized, and Doppler-tolerant 64-bit codes.

The plot shows the Doppler tolerance of three codes:

1) A MLS code used as the seed for the WAM algorithm: red
2) A WAM-generated code optimized for zero Doppler: green
3) A WAM-generated code optimized for Mach 10: blue

The next set of plots in Figure 5 displays the same comparison, but for 128-bit quad-phase codes, using Mach 5 as the desired optimization point for Doppler.

Figure 6 shows magnitude of the complex autocorrelation for the beginning seed and a code resulting after a number of iterations for 128-bit quad. The graph in red shows the seed (MLS sequences are used for real and imaginary); the graph in blue shows a resulting WAM solution with PSL of 7.6 (or -24.53 dB peak-sidelobe-to-peak ratio) and Merit Factor of approximately 4.

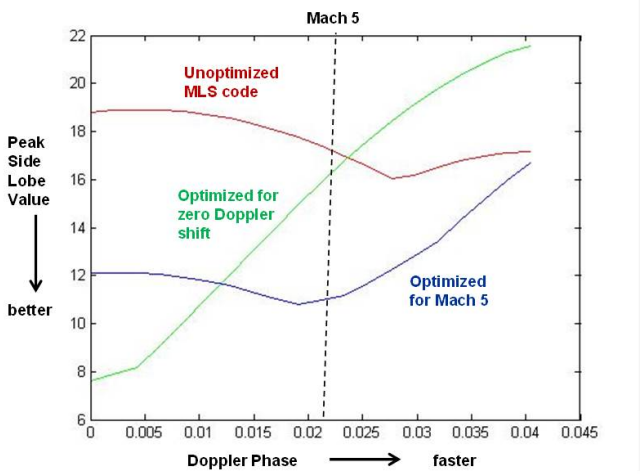

Figure 5. Peak Side Lobe values for optimized, unoptimized, and Doppler-tolerant 128-bit quad phase codes.

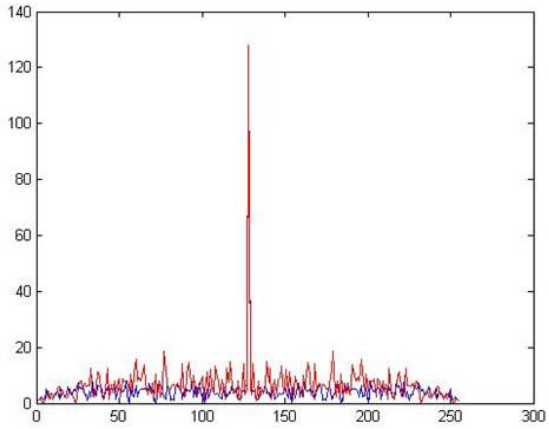

Figure 6. Magnitude of complex autocorrelation for a 128-bit quad MPSL code at zero Doppler. Red is the MLS seed and blue is the WAM solution.

VIII. New Codes

Some example codes are provided in this section to underline the power of both the standard GA algorithm and the WAM algorithm for finding low-sidelobe codes. Binary codes will be represented in hexadecimal format, where each character represents four binary bits. Any zero-fill bits are placed at the end of the hexadecimal representation.

An example of a long low-PSL code generated by the GA algorithm is the length-512 binary code listed below. It

achieves a PSL of 16 for a peak-sidelobe-to-peak ratio of -30.103 dB.

9C7D65FABF0F8FFB6D377947D6DA0AAD85B1B50C
F30356C3C67089FE586AB1D7025B8DC79B07CF314
8063EBFED5BA892A80E4889DB1A893BDB9DCDC8D
22E57B4

Table 1 lists PSL-8 (-24.08 dB) binary codes of length 128 found using WAM. An example 256-length binary code achieving PSL of 11 (or -27.3 dB relative to peak) found by the WAM algorithm is

3A6D5CB5AAB0593644D6E6608497F7253FD1FE1A72E8
A47A238C2E3984E3C.

Table 2 gives length-256 binary codes with PSL of 12, or -26.58 dB, found using WAM:

PSL-8 **Length-**128 **Binary Codes Found by WAM**
40BADA35B626CFF3FC8EEAD6287145E7
37CD25C84B59300C0BF31D2BD5818EAE
F7CD24C051D9310E0BB10D2B3529DE8E
0832DB3FAE26CEF1F44EF2D4CAD62171
8C21A4E949D9370D07FE158A1511DB8C
73DE5B1EB626C8F2F801EA75EAEE24F1
F3925831B622C8F2F801EA558EEE3775
BA8FFD1926020ED2E8A56ACD20673971
BE87FF1926060ED2E8A52ACD24663561
057220F7D9F9F1A51FD3D432D3BA46B3

Table 1

PSL-12 **Length-**256 **Binary Codes Found by WAM**
5A2D1CD1BAF04937CDCD7F3AF628695F7A53FC 1B71BD260047A60EE2F19BCABC
6A2D54C5BAE049364C4D7F3AF608697F7A53FD1 FF1BD2E8267A228C2E39E5ABC
7A2D5CD5BAE049364C4D7F3AF608697F7A53FD1 FF1BD2E8267A228C2E3984ABC
3A6D5CD5AAF049364C4D7F7AF608697F7353FD1 FE1A52E82672238C2E39C4E3C
7A6D5CB5AAA059364C4D6E6AFE08497F7253FD1 FE1A52E8A47A238C2E3BC4E3C
7AED5C952AA059364C4D6E6AFE08497F7253FD1 FE1A52E8A67A238C2E3BC4E3C
3AED5C85AAA059364C4D6E7AFE08697F7253FD1 FE1A52E8267A238C2E39C4E3C
32ED5C95AAB059364C4D6E6AFE08697F7253FD1 FE1A52E8A67A238C2E39C6E3C
3E6D5CB5AAB159364C4D6E6AFE08497F7253FD1 FE0A52E8A67E238C2E3BC4E3C
6A2D5CC5BAE048364C4D7F3AF608697F7A53FD1 FF1AD2E8267A228C2E39E5EBC

Table 2

One of the length-426 codes found by WAM achieves a PSL of 16 (or -28.51 dB):

0741E6499434B0C2590A211CA58BE1E2F8D6069A
C176C7CC7BA957F5190DE07160E2C123FFD4D54
40CF9DFA46F6744088DAE45952D4

Table 3 displays length-426 PSL-17 (-27.98 dB) binary code found by WAM:

PSL-17 **Length-**426 **Binary Codes Found by WAM**
0FE1E60D9634F049792A203E87BB60EAF8D4469AC1 77879E79A957F5B14DE03369E6C12DFF94F545CCB9 DFA94E635008898CC9942D4
07E1E60D9634B009792B20BE85ABE0EAF8D4469AC1 66C79E7BA957F5B14DE03369E6C029FF94F5440CB9 DFAD4E635408898CC9942D4
0FF1E62D9634B049792B203E85ABE0EAF8D446BAC1 76C79E7BB957F5B14DE03369E6C029FF94F5440CB9 DFAD4E635408898CC9942D4
0FE1E60D9630B049792B203E85ABE0EAF8D4469AC1 76C79E73B857F5B14DE03369E6C029FF94F5440CB9 DFAD4E635408898CC9942D4
4F61E64D9434B041590B203E87ABE0EAF8D4479AC5 76C79E73A957F5914DE07169E6C121FF94F5440CB9 DFAD4E6754088DACCD942D4
4F21E6499434B041590B203E87ABE0EAF8D4479AC5 76C79E73A957F5914DE07169E6C121FF94F5440CB9 DFAD4E6754088DACCD942D4
0F21E6499434B041590B203E87ABE0EAF8D4479AC5 76C79E73A957F59149E07169E6C123FF94F5440CB9 DFAD4E6754088DACCD942D4
8F61E44D9434B041590B203E87ABE0EA78D4469AC1 76C79E73A957F5914DE07169E6C121FF94F5440CB9 DFAD4E6754088DACCD942D4
0F61E64C9434B041790B203E87ABE0EAF8C4469AC1 76C79E73A957F5914DE07169E6C121FF94F5440CB9 DFAD4E6754088DACCD942D4
0761E64D8434B041590B211E85ABE0EAF8D6469AC1 76C79E73A957F5990DE03169E6C121FF94F5440CB9 DFAD4E6754088D8ECD142D4

Table 3.

IX. Conclusions

This paper reports results for an adiabatic quantum algorithm to compute low-peak-sidelobe binary and unimodular codes on a D-Wave 2 quantum computer. The quantum algorithm is benchmarked against a conventional GA. The main results of this study are that the quantum algorithm:

- Produces roughly a 100 times speedup relative to the GA for binary codes longer that 100 bits
- Produces low-sidelobe binary codes up to length 426 on the current D-Wave 2 hardware
- Produces low-sidelobe Doppler-tolerant binary and quad-phase codes
- Scales in nearly constant time per iteration
- Uses up to approximately 80% of available qubits on the D-Wave computer

The next generation D-Wave hardware is projected at 2048 qubits. Extrapolating from the current results, it should be possible to produce length 1700 binary codes with PSL64

with approximately the same computational effort. The current quantum algorithm is in an early stage of development, and it is likely that substantial improvements are possible in both computational and PSL performance. Additionally, it should be possible to simultaneously optimize for low PSL and low cross correlation code groups.

X. References

[1] Skolnik, M., *Radar Handbook*, edition 2, McGraw-Hill, NY, 1990.

[2] Levanon, N. and Mozeson, E., *Radar Signals*, Wiley, NY, 2005.

[3] Barker, R. H., "Group synchronization of binary digital systems", in Jackson, W. (editor), *Communications Theory*, Academic Press, London, 1953, pages 273 to 287.

[4] Golomb, S. and Scholtz, R., "Generalized Barker sequences (transformations with correlation function unaltered, changing generalized Barker sequences)", *IEEE Transactions on Information Theory*, volume 11, October 1965, pages 533 to 537.

[5] Friese, M. and Zottman, H., "Polyphase Barker sequences up to length 31", *Electronics Letters*, volume 30, number 23, 10 November 1994, pages 1930 to 1931.

[6] Golay, M.J.E., "Sieves for Low Autocorrelation Binary Sequences," *IEEE Transactions on Information Theory*, vol. 23, pp. 43?51, 1977.

[7] Goresky, M. and Klapper, A., *Algebraic Shift Register Sequences*, Cambridge University Press, Cambridge, 2012.

[8] Golomb, S., *Shift Register Sequences*, Holden-Day, Oakland, 1967.

[9] Golomb, S. and Gong, G., *Signal Design for Good Correlation*, Cambridge University Press, Cambridge, 2005.

[10] Militzer, B., Zamparelli, M. and Beule, D., "Evolutionary search for low autocorrelated binary sequences", *IEEE Transactions on Evolutionary Computation*, volume 2 (1998), number 1, pages 34 to 39.

[11] Kirkpatrick, S., Gelatt Jr, C. D., Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671?680, 1983.

[12] Dueck, G., "New optimization heuristics the Great Deluge algorithm and the record-to-record travel", *Journal of Computational Physics*, vol. 104, no. 1, p. 86-92, 1993.

[13] Deng, X., Fan, P., "New binary sequences with good aperiodic autocorrelation obtained by evolutionary algorithm," *IEEE Communication Letters*, vol. 3, no. 10, pp. 288-290, 1999.

noindent [14] Nunn, C. and Coxson, G. E., "Polyphase pulse compression codes with optimal peak and integrated sidelobes", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, number 2, pages 41 to 47, April 2009.

[15] Cohen, M., Fox, M. and Baden, J., "Minimum peak sidelobe pulse compression codes," *Proceedings of the IEEE International Radar Conference*, pp. 633-638, 1990.

[16] Coxson, G. and Russo, J., "Efficient exhaustive search for optimal-peak-sidelobe binary codes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 302-308.

[17] Bian, Z., Chudak, F., Macready, W., Rose, G., "The Ising model: teaching an old problem new tricks," D-Wave Systems, 2010.

[18] Brush, S., "History of the Lenz-Ising model," *Reviews of Modern Physics*, vol. 39, no. 4, pp. 883-893, 1967.

[19] Cipra, B., "An introduction to the Ising model," *American Mathematical Monthly*, vol. 94, no. 10, pp. 937-959, Dec. 1987.

[20] Lidar, D.A. and Biham, O., "Simulating Ising spin glasses on a quantum computer," *Physical Review E*, vol. 56, pp. 3661-3681, 1997.

[21] Ferreira, F.F., Fontanari, J.F. and Stadler, P.F., "Landscape statistics of the low autocorrelated binary string problem," *J. Physics A*, vol. 33, no. 48, pp. 8635-8647, 2000.

[22] Bernasconi, J., "Low autocorrelation binary sequences: statistical mechanics and configuration space analysis," *J. Physique*, vol. 48, pp. 559-567, 1987.

[23] D-Wave Systems Inc., Introduction to the D-Wave Quantum Hardware, http://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware.