A Quantum Annealer-Enabled Decoder and Hardware Topology for NextG Wireless Polar Codes

Srikar Kasi*†, John Kaewell†, Kyle Jamieson*

*Princeton University, †InterDigital, Inc.

Abstract

We present the Hybrid Polar Decoder (HyPD), a hybrid classical–quantum decoder design for Polar error correction codes, which are becoming widespread in today's 5G and tomorrow's 6G networks. HyPD employs CMOS processing for the Polar decoder's binary tree traversal, and Quantum Annealing (QA) processing for the Quantum Polar Decoder (QPD)–a Maximum-Likelihood QA-based Polar decoder submodule. QPD's design efficiently transforms a Polar decoder into a quadratic polynomial optimization form, then maps this polynomial on to the physical QA hardware via QPD-MAP, a customized problem mapping scheme tailored to QPD. We have experimentally evaluated HyPD on a state-of-the-art QA device with 5,627 qubits, for 5G-NR Polar codes with block length of 1,024 bits, in Rayleigh fading channels. Our results show that HyPD outperforms Successive Cancellation List decoders of list size eight by half an order of bit error rate magnitude, and achieves a 1,500-bytes frame delivery rate of 99.1%, at 1 dB signal-to-noise ratio. Further studies present QA compute time considerations. We also propose QPD-HW, a novel QA hardware topology tailored for the task of decoding Polar codes. QPD-HW is sparse, flexible to code rate and block length, and may be of potential interest to the designers of tomorrow's 6G wireless networks.

Index Terms

Polar codes, quantum computation, quantum annealing, channel decoding, wireless networks

I. Introduction

The Polar Code, a type of error control code, was discovered in 2009 [1], and subsequently shown to have a number of desirable features: achievement of Shannon capacity limits for a wide

range of channels [2], attainment of a low error floor (minimal bit error rate as a function of background noise) [3], and a simpler code construction process than other leading competitors, such as Low Density Parity Check (LDPC) codes [4]. While promising, Polar codes face several practical challenges if they are to manage decoder design complexity while at the same time maintaining their capacity achieving properties [5]. Low-complexity *Successive Cancellation* (SC) decoder can only achieve capacity on Polar codes that have been constructed *a priori* with knowledge of the communication channel [6], which is unfortunately impractical in a wireless network context where user mobility causes wireless channels to be largely unpredictable. The *Successive Cancellation List* (SCL) [7] decoder can approach Shannon capacity, but at the price of high complexity and high latency, thus compromising Polar codes' advantage over LDPC codes in this regard. Furthermore, SC/SCL algorithms are sequential in nature, which means their decoding latency grows at least linearly with code block length, leading to low throughput in the decoding process [5], [8].

Consequently, Polar Code use in 5G New Radio [9] is currently limited to control channels with short block lengths, but Polar codes' solid theoretical foundation, simple encoder implementation, and adjustable code rate from zero to one would make them viable candidates for high speed data channels such as for 5G *enhanced Mobile Broad-Band* (eMBB), and for 5G *ultra-reliable, low-latency channels* (URLLC), if the latency of the decoder could be minimized, while simultaneously maintaining a low bit error rate and near-capacity-limit rate performance. Overcoming processing throughput and latency limitations would also enable Polar codes' adoption in transformative 5G *Massive Machine Type Communications* (MMTC) technologies such as NB-IoT and LTE-M, which aim to scale cellular network coverage densities to millions of devices per square kilometer, and hence put them on the 6G roadmap with more certainty [10] .

It is with this vision in mind that this paper investigates a radically different processing architecture for a Polar code decoder, one based on *quantum annealing*, to see if this emerging technology can potentially speed up the decoding of Polar codes using the fundamental, quantum mechanical properties of superposition, entanglement, and tunneling. This would open up new possibilities in the design of the Polar code decoder. In our envisioned scenario, shown in Fig. 1, quantum processing units (QPUs) are co-located with CMOS processing units in a Centralized-RAN (C-RAN) data center, where QPUs are used for heavyweight computational tasks in the cellular baseband unit such as decoding and demodulation, and CMOS processing units undertake lightweight computational tasks such as handling the network's control plane [11].

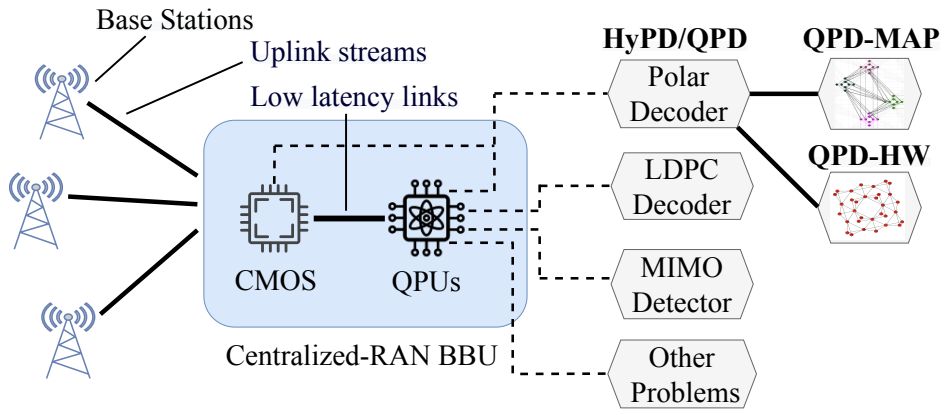


Figure 1: An envisioned **HyPD/QPD** deployment scenario in a Centralized RAN (C-RAN) context, where a QA server augments a C-RAN baseband unit (BBU).

This paper presents the ***Hybrid Polar Decoder (HyPD)***, a new classical–quantum hybrid Polar decoder design that considers error correction decoding from the fresh perspective of Quantum Annealing (QA). HyPD works by partitioning a long mother Polar code's binary tree into a number of shorter *sub-blocks*, where each sub-block is a perfect subtree with same leaf nodes as that of the mother Polar code (see Fig. 5). We structure HyPD's operation into classical and quantum processing modules. Our classical module considers CMOS hardware for the Polar decoder's binary tree traversal, and it operates between the mother Polar code's root node and the root nodes of the partitioned sub-blocks. Our quantum module implements a Quantum Polar Decoder (QPD) on a QA device to solve these partitioned sub-blocks, in order to decode the transmitted bits. HyPD's classical and quantum modules exchange bit-likelihood and bit-decision information, respectively, back and forth until all bits are decoded.

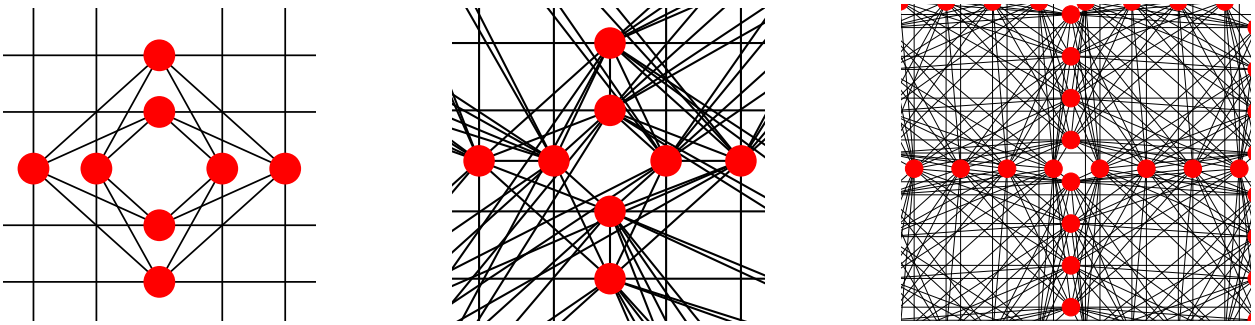
Our quantum module's ***Quantum Polar Decoder (QPD)*** is a new Maximum-Likelihood Polar decoder design that efficiently formulates a one-to-one mapping between the Polar code's binary tree-structured encoder and a quadratic polynomial form that a QA can solve at the receiver, in order to decode the transmitted bits. This polynomial is a linear combination of multiple *cost penalty* functions we have created, which we refer to as *Node*, *Frozen*, and *Receiver* constraints. A linear combination is chosen to ensure that the polynomial is quadratic and amenable to QA devices. The Node and Frozen constraints work by raising a positive cost penalty to candidate bit strings that do not agree with the Polar encoding conditions, thus ensuring that QPD outputs only valid bit strings. The Receiver constraints add a further cost penalty to all the valid bit

strings whose magnitude depends on the proximity of a candidate bit string to the received soft information, thus allowing QPD to select the most likely transmitted bit string. We map these cost penalty functions directly on to the physical grid of qubits present in the QA hardware, via ***QPD-MAP***, our customized problem mapping design tailored for QPD, taking into account the real-world physical qubit interconnections. QPD-MAP is flexible to code rate, block length, and embodies quantum annealing correction (QAC) to correct the QA computational errors introduced into the problem at hand, improving the solution quality. The QA returns the bit string with minimal cost penalty as its decoded solution. HyPD gathers all the bit strings (corresponding to sub-blocks) returned by QPD and concatenates them, to output the decoded user message.

We have experimentally evaluated HyPD on a state-of-the-art QA device with 5,627 qubits: Advantage_system4.1 [12], for CRC assisted Polar codes of block length 1,024 bits, in Rayleigh fading wireless channels, at low signal-to-noise ratio (SNR) regimes of practical interest. Our evaluations consider BPSK modulation employed in 5G-NR control channels, and 200 message data bits, which is the typical maximum payload size of uplink control information (UCI) in LTE and 5G-NR eMBB scenarios [13]. Our results show that HyPD operating at a 300 μs QA compute time outperforms leading edge SCL decoders operating at list size of eight by half an order of bit error rate magnitude, at 1 dB SNR. Further evaluations show that HyPD achieves a 1500-bytes frame delivery rate of 99.1% at 1 dB SNR, whereas SCL achieves only 95.5% frame delivery rate at the same SNR. We also analyze QA compute time at various code rates and with increased qubit numbers (§VI-C2). The lessons we have learned in the design and implementation of QPD on QAs point to alternate QA hardware structures that if realized, would be preferable to the qubit connectivity currently available. We present this ideal QA hardware structure tailored to Polar Codes, **QPD-HW**, in §VII, demonstrating its features and flexibility.

II. Primer: Quantum Architectures

While classical computation uses bits to process information, quantum computation uses *qubits*, physical devices that exhibit quantum mechanical properties such as superposition and entanglement. The current and near-term quantum computers can be classified into digital gate-model or analog annealing-model architectures. Gate-model devices are fully general purpose computers that perform computation using programmable logic (*i.e.,* gates) acting on qubits, whereas annealing-model devices are specialized computers that solve combinatorial optimization problems in their equivalent *Ising* specifications. While gate-model quantum devices of



(a) Chimera **(b)** Pegasus **(c)** Zephyr

Figure 2: The figure shows unit cell interconnection structures of **(a)** Chimera (recent) **(b)** Pegasus (state-of-the-art) and **(c)** Zephyr (next-generation) QA hardware topologies. Nodes in the figure are qubits and edges are couplers.

size relevant to practical applications are not yet generally available, today's annealing-model devices with about 5,000 qubits enable us to commence empirical studies at realistic scales [11]. Therefore, we conduct this study from the perspective of annealing-model devices.

Quantum Annealing (QA) aims to find the lowest energy *spin configuration* (*i.e.*, solution) of an *Ising model* described by the time dependent energy functional (Hamiltonian):

$$H(s) = -\Gamma(s)H_I + L(s)H_P \tag{1}$$

where H_I is the *initial Hamiltonian*, H_P is the (input) *problem Hamiltonian*, s ($\in$ [0, 1]) is a non-decreasing function of time called an *annealing schedule*, $\Gamma(s)$ and $L(s)$ are energy scaling functions of the transverse and longitudinal fields in the annealer respectively. Essentially, $\Gamma(s)$ guides the probability of *tunneling* during the annealing process, and $L(s)$ guides the probability of finding the ground state of the input problem Hamiltonian H_P [12]. The QA hardware is a network of locally interacting radio-frequency superconducting qubits, organized in groups of *unit cells*. Figure 2 shows the unit cell structures of QA devices [14]. The nodes and edges in the figure are *qubits* and *couplers* respectively.

a) Annealing Process: Starting with a high transverse field (*i.e.,* $\Gamma(0) >> L(0) \approx 0$), the QA processor initializes the qubits in a pre-known ground state of the initial Hamiltonian H_I, then gradually interpolates this Hamiltonian over time (*i.e.,* decreasing $\Gamma(s)$ and increasing $L(s)$) by adiabatically introducing quantum fluctuations in a low-temperature environment, until the transverse field diminishes (*i.e.,* $L(1) \gg \Gamma(1) \approx 0$). The Adiabatic Theorem then ensures that

by interpolating the Hamiltonian slowly enough, the system remains in the ground state of the interpolating Hamiltonian [15], [16]. Thus during the annealing process, the system ideally stays in a low energy state and probabilistically reaches the global minima of the problem Hamiltonian H_P at its conclusion. The process of optimizing a problem in the QA is called *annealing*, and the time taken for this optimization is called *annealing time* [12].

b) Hamiltonian Forms: The initial Hamiltonian takes the form $H_I = \sum_i \sigma_i^x$, where σ_i^x is the *Pauli-X* spin operator acting on the i^{th} qubit. Thus, the initial state of the system is the ground state of this H_I, where each qubit is in an equal superposition state $\frac{1}{\sqrt{2}}(|-1\rangle + |+1\rangle)$. The problem Hamiltonian is $H_P = \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z$, where σ_i^z is the *Pauli-Z* spin operator acting on the i^{th} qubit, h_i and J_{ij} are the optimization problem inputs that the user supplies.

c) Input Problem Forms: QAs optimize Ising model problems, whose problem format matches the above problem Hamiltonian: $E = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j$, where E is the energy, $s_i \in \{-1, +1\}$ is the i^{th} solution variable, h_i is called *bias* of s_i, and J_{ij} is called *coupler strength* between s_i and s_j. Biases and coupler strengths are programmed onto qubits and couplers respectively using an on-chip control circuitry [12]. Ising format is equivalent to *quadratic unconstrained binary optimization* (QUBO) format, and it is obtained via the transformation $s_i \longrightarrow 2q_i - 1$, resulting in the energy function: $E_Q = \sum_i f_i q_i + \sum_{i<j} g_{ij} q_i q_j$, where E_Q is the QUBO form energy, $q_i \in \{0, 1\}$ is the i^{th} solution variable, f_i and g_{ij} are QUBO form linear and quadratic coefficients respectively. The QA probabilistically returns a solution variable configuration with minimum energy E at its output.

d) Embedding: To solve an Ising problem on a QA device, the problem must be mapped on to the physical QA hardware. This mapping process is called *embedding*, and it typically requires additional qubits. To understand embedding, let us consider an example Ising problem:

$$E = J_{12} s_1 s_2 + J_{13} s_1 s_3 + J_{23} s_2 s_3 \tag{2}$$

The connectivity structure of solution variables in Eq. 2 is a complete graph on three nodes, which does not exist in the Chimera graph (Fig. 2(a)). Thus, to implement Eq. 2 on Chimera-based hardware, the standard approach is to map one of the solution variables (*e.g.,* s_3) onto two physical qubits (*e.g.,* s_{3a} and s_{3b}), such that the resulting connectivity (*e.g.,* square graph) can be realized on the hardware. To ensure proper embedding, s_{3a} and s_{3b} must agree with each other, and this is achieved by chaining them with a strong negative coupler strength [17].

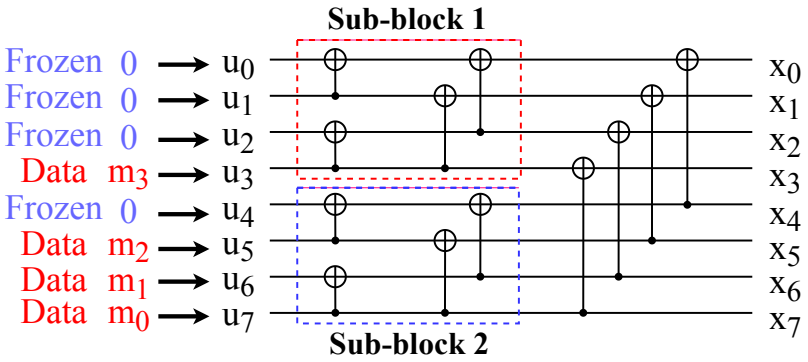


Figure 3: The encoding process of an example (N = 8) Polar code. m_i is a message bit, u_i is an encoder input bit, and x_i is an encoded bit. $\oplus$ represents the XOR operation.

QA processors exploit quantum properties such as superposition, entanglement, and tunneling during the annealing process, which under ideal conditions can speed up the optimization task [18], [19]. Realization of this speedup may be challenging today due to issues surrounding current QA hardware. In Section IV, we describe the unique challenges in applying QA computation for Polar Code decoding and demonstrate how we address these challenges in this work.

III. Primer: Polar Codes

A binary $(N = 2^d, K)$ Polar code is described functionally by a generator matrix $\mathbf{G}_N = \mathbf{G}_2^{\otimes d}$, where $\otimes d$ operation represents d-fold Kronecker product, and $\mathbf{G}_2 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ is called the *polarization kernel*. The *channel polarization* phenomenon in Polar codes is a transformation of N independent bits to N mutually interlinked bit-channels, where each bit-channel has a probability of being decoded correctly (*i.e.,* reliability). The sequence of these bit-channels in sorted order of their reliabilities is called the *reliability sequence*. Let N be the block length adapted for transmitting a message $\mathbf{m} = [m_0, m_1, ..., m_{K-1}]$ of length $K \leq N$ bits. The coding rate is then K/N. Using the reliability sequence, construct the encoder *input vector* $\mathbf{u} = [u_0, u_1, ..., u_{N-1}]$ by assigning the message bits to K most reliable bit-channels, and set the remaining $N - K$ bits to a zero value. The u_is that are set to zero value are called *frozen bits*. The encoded codeword is then $\mathbf{x} = \mathbf{u}\mathbf{G}_N$. Fig. 3 depicts the encoder operation for an example eight-bit Polar code.

To visualize QPD's decoding process (described later in §IV-B), we here demonstrate the encoder operation using a binary tree representation. Fig. 4 depicts this for Sub-block 1 of Fig. 3 with input vector $\mathbf{u} = [u_0, u_1, u_2, u_3]$ of length $N_L = 4$ bits. Construct a perfect binary tree

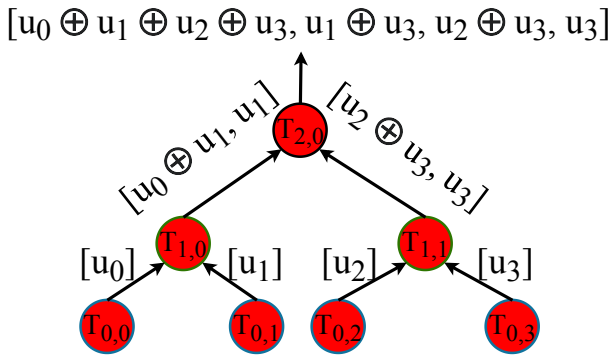


Figure 4: Encoder binary tree for Sub-block 1 of Fig. 3. $T_{h,i}$ is a node at height *h* and index *i*.

with N_L leaf nodes as shown in Fig. 4, and initialize each leaf node $T_{0,i}$ of the tree with bit u_i. Traversing the tree from height $h = 0$ (leaf) to $h = \log_2^{N_L}$ (root), each node $T_{h,i}$ $\forall h \in [1, \log_2^{N_L}]$ takes an input $[u_L|u_R]$ and generates an output $[u_L \oplus u_R|u_R]$, where u_L and u_R are the bit vectors of the left and right children of $T_{h,i}$ respectively, and $u_L \oplus u_R$ is a bit-wise XOR operation. The output vector obtained at the root is then the encoded codeword. In the example in Fig. 4, the encoded codeword is $\mathbf{x} = [u_0 \oplus u_1 \oplus u_2 \oplus u_3, u_1 \oplus u_3, u_2 \oplus u_3, u_3]$.

IV. Design

In this section, we first describe the proposed HyPD decoder design (§IV-A), and then present QPD's reduction of Polar decoding into a QUBO form (§IV-B). Next we present our customized problem mapping scheme, QPD-MAP, in §IV-C, alongside practical QA considerations in §IV-D.

HyPD, the hybrid classical–quantum decoder, is a sub-optimal decoder, whereas QPD, the pure quantum decoder, is the optimal Maximum-Likelihood decoder. The need for HyPD arises because of limited qubits available in today's QAs and the requirement to solve long 1,024 bits 5G Polar codes. Long codes can be entirely decoded via QPD, with sufficient qubits (§VII).

A. HyPD: Hybrid Classical-Quantum Polar Decoder

Let $\mathbf{u} = [u_0, u_1, ..., u_{N-1}]$ be the input vector, and $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]$ be the corresponding Polar-encoded codeword. Let $\mathbf{y} = [y_0, y_1, ..., y_{N-1}]$ be the respective received soft information. HyPD works by partitioning a long mother Polar code of block length N bits into N_{sub} number of shorter *sub-blocks*, where each sub-block is a largest perfect subtree with N_L leaf nodes of the mother Polar code tree ($N = N_{\text{sub}}N_L$). Fig. 5 depicts this partitioning scheme for an example 16-bit Polar code with $N_{\text{sub}} = N_L = 4$. We structure HyPD into classical and quantum processing

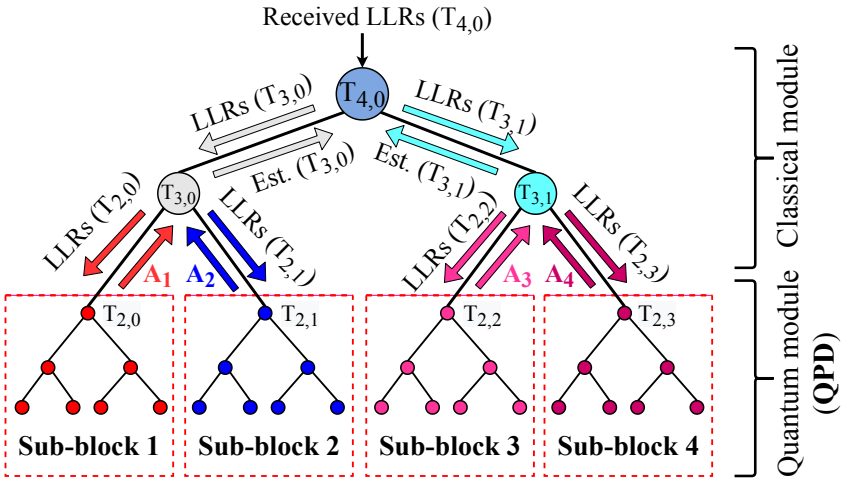


Figure 5: HyPD decoding of an example Polar code. The classical module traverses the tree, and the quantum module solves sub-blocks on QA. The downward and upward arrows show LLRs' and estimated bits' propagation, respectively. $\mathbf{A}_i$ is the QA solution for the i^{th} sub-block.

modules as in Fig. 5. The classical module operates between the root node of the mother Polar code and the root nodes of sub-blocks, and the quantum module operates on sub-blocks.

HyPD begins at the root node by computing the log-likelihood ratios (LLRs) of root node bits from the received soft data (Fig. 5). Each node in our classical module sends to its left and right children the LLRs of their corresponding bits, by computing F and G functions respectively [20]:

$$F(s,t) = 2\tanh^{-1}(\tanh L_s/2 \cdot \tanh L_t/2) \tag{3}$$

$$G(s,t,\widehat{s\oplus t}) = L_s(-1)^{\widehat{s\oplus t}} + L_t, \tag{4}$$

where L_s is the LLR of bit s, $F(s,t)$ is the LLR of bit $s \oplus t$, and $G(s,t,\widehat{s\oplus t})$ is the conditional LLR of bit t with respect to previously decoded bit $\widehat{s\oplus t}$. For a node at height *h*, the choices of $s \in \{s_1, s_2, ..., s_{2^{h-1}}\}$ and $t \in \{t_1, t_2, ..., t_{2^{h-1}}\}$ are the pairs $(s_i, t_i)\forall i$, where $[s_1, s_2, ...s_{2^{h-1}}, t_1, t_2, ..., t_{2^{h-1}}]$ is the bit vector of the node. To understand the meaning of the F and G functions, let us observe node $T_{2,0}$ of Fig. 4, with height $h = 2$ and bit vector $[s_1, s_2, t_1, t_2]$ = $[u_0 \oplus u_1 \oplus u_2 \oplus u_3, u_1 \oplus u_3, u_2 \oplus u_3, u_3]$. The bits corresponding to $T_{2,0}$'s left and right children are $[s_1 \oplus t_1, s_2 \oplus t_2]$ and $[t_1, t_2]$ respectively (*cf.* Fig. 4). By calculating the LLRs of $[s_1 \oplus t_1, s_2 \oplus t_2]$, the F function captures the LLRs of a node's left child bits. The intuition behind the G function is that if $\widehat{s\oplus t}$ is estimated to be zero (see Eq. 4), then $s = t$ and the conditional LLR of t

becomes $L_s + L_t$, otherwise $s \neq t$ and the conditional LLR of t becomes $L_t - L_s$, $\forall(s_i, t_i)$ pairs. Thus the G function captures the LLRs of a node's right child bits [20].

Similar to SC/SCL decoder operation, our classical module traverses the tree depth-first, with priority given to the left branches, propagating the corresponding LLRs downward [7]. In this process, we obtain the LLRs of the sub-blocks' root node bits, which are then our quantum module's input. Using this LLR input, we solve each sub-block on a QA device, and the solution returned by QA is fed back to the classical module. The solution feedback is necessary for the classical module to compute G functions (*i.e.,* $\widehat{s \oplus t}$ values in Eq. 4 are obtained using QA). Multiple solutions can be fed back to explore more decoding paths, and we refer the number of solutions fed back to N_{SF}. These bit-likelihood and bit-decision information exchanges between our classical and quantum modules, respectively, comprises our proposed HyPD decoder operation. The decoder terminates when all the bits are decoded (*i.e.,* all sub-blocks are solved). We next demonstrate HyPD more fully with a running example.

Consider a 16-bit Polar code as in Fig. 5, and input the LLRs of received bits at $T_{4,0}$, the root node. $T_{4,0}$ sends to $T_{3,0}$, and $T_{3,0}$ sends to $T_{2,0}$, the LLRs of their corresponding bits, by computing F functions (left child). Sub-block 1 is then solved on the QA, and the solution obtained, $\mathbf{A}_1$, is fed back to $T_{3,0}$. Using this solution, $T_{3,0}$ sends to $T_{2,1}$ its corresponding LLRs, by computing the G function (right child). Sub-block 2 is then solved on the QA, and the solution obtained, $\mathbf{A}_2$, is fed back to $T_{3,0}$. Using $\mathbf{A}_1$ and $\mathbf{A}_2$, $T_{3,0}$ estimates its bits (similar to encoding §III) and sends them back to $T_{4,0}$. A similar tree traversal process now happens at the right-hand branch of $T_{4,0}$, where $\mathbf{A}_3$ and $\mathbf{A}_4$ are the solutions obtained for Sub-blocks 3 and 4 respectively (see Fig. 5). HyPD terminates when all the sub-blocks are solved. The decoded answer is the bit decisions corresponding to leaf nodes wherein user data is located.

Design Analysis. In order to demonstrate the advantage of HyPD over SC/SCL decoders, we now discuss their decoding algorithms, referring to Fig. 6. In SC and SCL decoding, the tree traversal process described in Fig. 5 continues until the leaf nodes (*i.e.,* it is entirely classical), and at each leaf node we have one bit which can take two possible values (0 or 1). SC makes a hard decision on each leaf node bit (0 if LLR is positive-valued, 1 otherwise) and continues the tree traversal. Therefore, SC performs search over only one bit at a time and maintains only one decoding path (*cf.* Fig. 6). Unlike SC, SCL continues the tree traversal with both 0 and 1 decisions for each non-frozen leaf node bit (0 decision is made for frozen bits), where each decision leads to a distinct decoding path. The number of decoding paths therefore increases

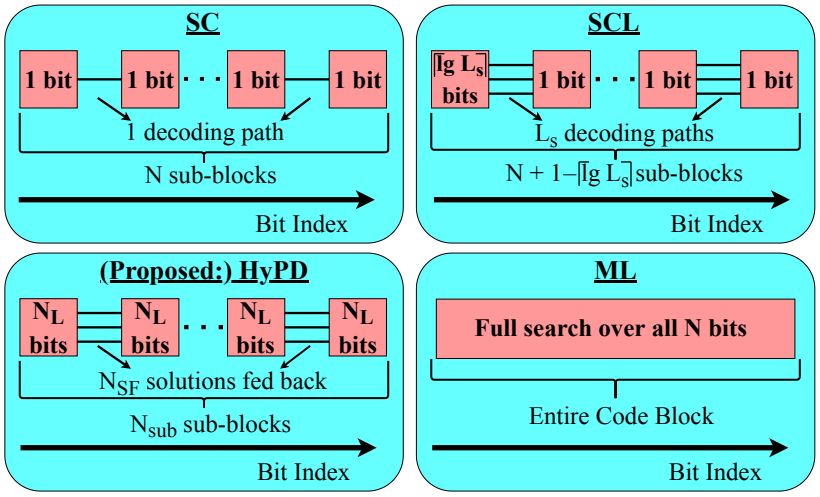


Figure 6: Algorithmic structures of various Polar decoders. Each square/rectangle block shows the number of bits over which a collective search is performed, and each edge connecting these blocks is a decoding path. Collective search indicates verifying all possible configurations.

exponentially with the number of non-frozen leaf nodes. In order to reduce SCL's computational complexity, a threshold is placed on the number of decoding paths, called the list size L_s. When the number of paths grows beyond L_s, only the best L_s paths continue (see Ref. [7]). Therefore, SCL performs a collective search over the first $\lceil \log_2 L_s \rceil$ bits (*i.e.,* until L_s decoding paths emerge), followed by a search over only one bit at a time, as shown in Fig. 6. Unlike SC and SCL, HyPD performs a collective search over all N_L bits present in each sub-block (§IV-A)–via QA. In HyPD, each solution fed back from the quantum module to the classical module leads to a distinct decoding path, as shown in Fig. 6. The Maximum Likelihood (ML) decoder performs a collective search over the entire code block, thus achieving the optimal performance.

Complexity Analysis. The complexity of SC decoder is $O(N\log N)$, and that of SCL decoder is $O(L_s N\log N)$. If $L_s > 1$, SCL decoder outperforms SC decoder, and if $L_s = 1$, SC and SCL are the same decoder. The complexity of ML decoder is $O(2^{NR} N\log N)$, where R is coding rate. If $L_s < 2^{NR}$, ML decoder outperforms SCL decoder, and if $L_s = 2^{NR}$, SCL and ML are the same decoder. The complexity of HyPD can be derived by analyzing its classical and quantum modules separately. HyPD's classical module's complexity is $O(N_{SF} N\log N_{\text{sub}})$. This is because our classical module processes N_{SF} decoding paths, and it operates on a subtree with N_{sub} number of leaf nodes, which are essentially the root nodes of sub-blocks (see Fig. 5). Our quantum module's complexity is $O(N_{SF} N_{\text{sub}} 2^{f(N_L(1+\log N_L))})$, where the factor $N_{SF} N_{\text{sub}}$ indicates the number of

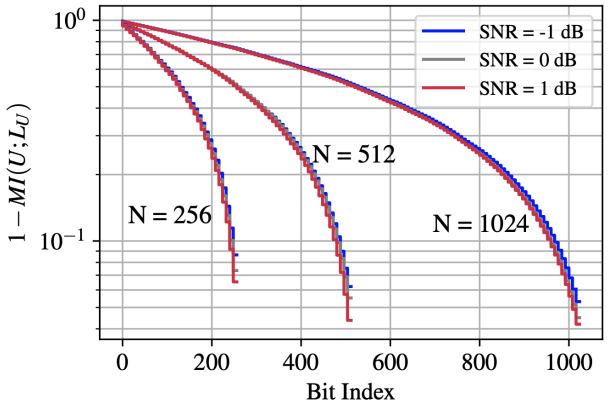


Figure 7: The behavior of mutual information between user data and their output LLR values as the HyPD decoding progresses. Mutual information is nearly zero at the start of decoding and it reaches > 0.9 at the end of decoding for all block lengths, demonstrating a high convergence.

sub-blocks processed via QA, and the exponent $N_L(1 + \log N_L)$ is the solution variable count for solving an N_L-bits sub-block (see §IV-B, §VII). We have conservatively used the function $f(\cdot)$ to capture the speedup of QA over classical methods, which is still yet to be precisely quantified (see [18], [19]). If $N_L = 1$, HyPD and SCL ($L_s = N_{SF}$) are the same decoder, and if $N_L = N$, HyPD and ML are the same decoder. For any N_L, if $N_{SF} < L_s = 2^{N_L}$, SCL decoder outperforms HyPD. For any $N_L > 1$, if $N_{SF} \geq L_s$, HyPD outperforms SCL decoder (see Fig. 6).

Convergence Analysis. As noted above, HyPD converges when all the sub-blocks are solved, which means that its convergence speed is proportional to the sub-block count. To understand convergence, we next observe how the mutual information (MI) between equiprobable user data U and their output LLRs L_U changes with the decoding progress. This is computed as [21]:

$$MI(U; L_U) = 1 - \frac{1}{N} \sum_{\forall i} H_b\Big(\frac{1}{1 + e^{-|L_i|}}\Big) \tag{5}$$

where H_b is the binary entropy function and L_i is the LLR value of i^{th} user data bit at the decoder output. In Fig. 7, the decoding starts at bit index 0 and ends at bit index N, and $MI(U; L_U)$ is computed after decoding every eight bits (*i.e.,* $N_L = 8$ sub-blocks), resulting in step functions of width eight bits. At the start of decoding, MI is nearly zero as the received data is highly corrupted. As the decoding progresses, MI increases, reaching 0.934, 0.956, 0.958 for block lengths 256, 512, 1024 bits respectively (SNR 1 dB) at the end of decoding, demonstrating a high convergence. We also note that the convergence speed is higher for shorter block lengths. To increase the convergence speed of long codes, larger sub-blocks must be solved.

B. QPD: Quantum Polar Decoder

We now present our QPD sub-module. Let $\mathbf{q}_{\text{sub}} = [q_0, q_1, \ldots, q_{N_L-1}]$ be the *solution* variables used to extract a sub-block's input bits $\mathbf{u}_{\text{sub}} = [u_0, u_1, \ldots, u_{N_L-1}]$ respectively. Let $\mathcal{F}$ be the set of frozen bits, $\mathcal{T}$ be the set of sub-block's binary tree nodes, and a_i be an *ancillary* variable used for calculation purposes. Any b_i is a generic binary variable (*i.e.,* solution or ancillary).

1) QUBO Formulation: QPD's objective function comprises multiple terms, classified into three types: *Node*, *Frozen*, and *Receiver* constraints. The Node constraints (C_N) ensures that a candidate decoding agree with the Polar encoding conditions. If a candidate decoding violates these constraints, a cost penalty is raised for that candidate (*i.e.,* the candidate is raised in energy). The Frozen constraints (C_F) ensures all a candidate decoding agree with the frozen bit conditions (*i.e.,* qubits that represent frozen bits must take a zero value). If a candidate decoding disagrees, a cost penalty is raised for that candidate. The Receiver constraints (C_R) introduce a further cost penalty to all valid candidates, whose magnitude depends on the proximity of an individual candidate to the received soft information. They thus encourage the decoder to find the decoding that most closely matches the received information. The entire QUBO objective function is a weighted linear combination of these cost functions:

$$\arg\min_{\mathbf{q}} \Big\{ W_N \sum_{\forall T \in \mathcal{T}} C_N(T) + W_F \sum_{\forall u_i \in \mathcal{F}} C_F(q_i) + W_R \sum_{\forall j} C_R(b_j) \Big\}. \tag{6}$$

A linear combination ensures that Eq. 6 is a quadratic polynomial. The weights W_N, W_F, W_R prioritize their respective constraints. We determine the best choices of these weights in §IV-D.

2) Node Constraints: From Section III, we observe that the Polar encoder performs only XOR operations. Let us define $\mathcal{E}_T$ as the set of all XOR operations the encoder performs at node T of the binary tree. For each $T \in \mathcal{T}$ (defined earlier in §IV-B), we define a Node constraint:

$$C_N(T) = \sum_{\forall \text{XOR}(b_i, b_j) \in \mathcal{E}_T} (b_i + b_j - a_k - 2a_{k+1})^2, \tag{7}$$

where b_i, b_j represent the variables whose equivalent bits are XORed at node T in the encoding process, and a_k, a_{k+1} are ancillary variables. The value of $k \in \{2p | p \in \mathbb{W}\}$ is chosen such that each ancillary variable is only introduced once. We observe that $C_N(T)$ is in sum-of-squares form, thus at the minimum energy (*i.e.,* $C_N(T) = 0$), the sum $b_i + b_j$ must be equal to the sum $a_k + 2a_{k+1}$. Since all the variables are binary, this implies that $a_k = b_i \oplus b_j$ in the minimum energy configuration. Upon expansion of Eq. 7, C_N introduces both linear and quadratic terms into the

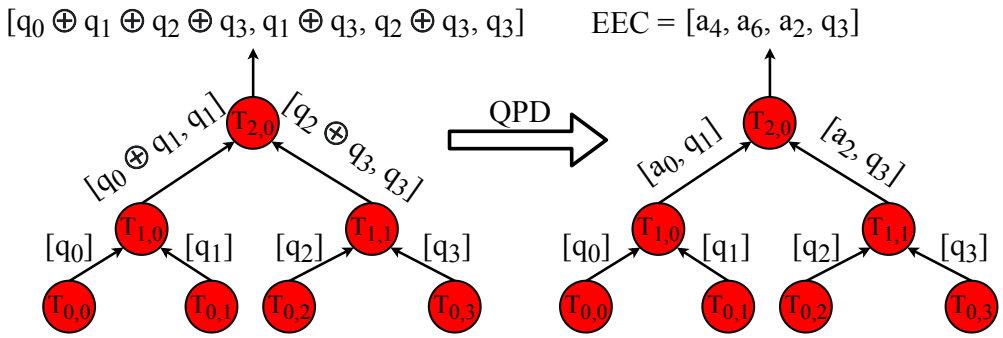


Figure 8: QPD decoding process of Sub-block 1. (*Left*) Direct representation of Fig. 4 using solution variables. (*Right*) QPD's equivalent representation of *Left* using ancillary variables.

objective QUBO (Eq. 6), with quadratic coefficient values in $\{-4, -2, +2, +4\}$ only. We next demonstrate the working process of Node constraints more fully with a running example.

Let us continue with Sub-block 1 whose encoder tree is shown in Fig. 4 with input vector $\mathbf{u}_{\text{sub}}$ = $[u_0, u_1, u_2, u_3]$. Let $\mathbf{q}_{\text{sub}}$ = $[q_0, q_1, q_2, q_3]$ be the solution variables used at the decoder to extract the bits $[u_0, u_1, u_2, u_3]$ respectively. Fig. 8 (*Left*) shows the direct representation of Fig. 4 using respective variables at the decoder, and Fig. 8 (*Right*) shows QPD's equivalent representation of Fig. 8 (*Left*). In this example, $\mathcal{T}$ = $\{T_{2,0}, T_{1,0}, T_{1,1}, T_{0,0}, T_{0,1}, T_{0,2}, T_{0,3}\}$. Similar to encoding, we traverse the tree from leaf to root for constructing QPD's Node constraints as follows.

At height = 0, we note that the nodes $\{T_{0,0}, T_{0,1}, T_{0,2}, T_{0,3}\}$ perform no computation, and so the Node constraints of these nodes are zero (*i.e.,* $C_N(T_{0,i}) = 0\ \forall i$). At height = 1, we have two nodes $\{T_{1,0}, T_{1,1}\}$ that perform one XOR operation each (see Fig. 8). In particular, $T_{1,0}$ computes $q_0 \oplus q_1$ and $T_{1,1}$ computes $q_2 \oplus q_3$. Thus using Eq. 7, we construct two Node constraints as: $C_N(T_{1,0})$ = $(q_0 + q_1 - a_0 - 2a_1)^2$ and $C_N(T_{1,1})$ = $(q_2 + q_3 - a_2 - 2a_3)^2$. Here a_0 equals $q_0 \oplus q_1$, and a_2 equals $q_2 \oplus q_3$ in the minimum energy solution. At height = 2, the root node $T_{2,0}$ performs two XOR operations (see Fig. 8): $q_0 \oplus q_1 \oplus q_2 \oplus q_3$, and $q_1 \oplus q_3$. We note that these computations are equivalent to $a_0 \oplus a_2$ and $q_1 \oplus q_3$ respectively. Hence, using Eq. 7 we construct the Node constraint: $C_N(T_{3,0})$ = $(a_0 + a_2 - a_4 - 2a_5)^2 + (q_1 + q_3 - a_6 - 2a_7)^2$. Here a_4 equals $a_0 \oplus a_2$, and a_6 equals $q_1 \oplus q_3$ in the minimum energy solution. The ancillary variables representing such XORs are reflected in Fig. 8 (*Right*). The output vector obtained at the root node $[a_4, a_6, a_2, q_3]$ is now bit-wise equivalent to the corresponding encoded data $[q_0 \oplus q_1 \oplus q_2 \oplus q_3, q_1 \oplus q_3, q_2 \oplus q_3, q_3]$. We hence refer this output vector to the *equivalent encoded codeword (EEC)*.

3) Frozen Constraints: From Section III, we note that frozen bits do not carry user information and that they are always assigned zero value. Hence we define a Frozen constraint as:

$$C_F(q_i) = q_i \qquad \forall u_i \in \mathcal{F} \tag{8}$$

C_F is minimum when the variables that represent frozen bits take a zero value (*i.e,* not one value). C_F introduces only linear terms into the objective QUBO (Eq. 6).

4) Receiver Constraints: We next consider the EEC obtained from the Node constraints, and compute its distance to the corresponding received data using a Receiver constraint as in [17]:

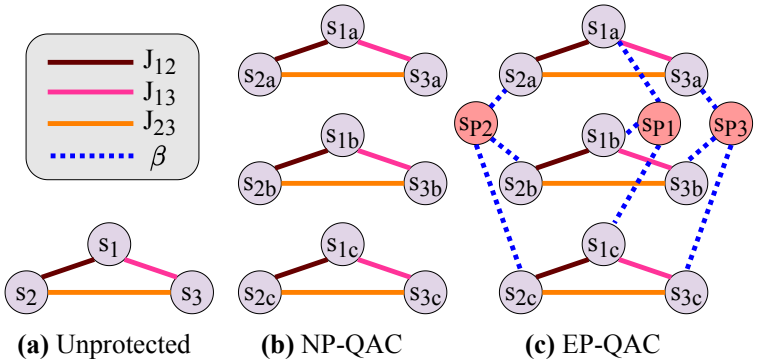
$$C_R(b_j) = (b_j - Pr(b_j = 1|\mathbf{y}))^2 \qquad \forall b_j \in EEC \tag{9}$$

where the probability $Pr(b_j = 1|\mathbf{y})$ can be computed for various modulations and channels, using the LLR information the HyPD's classical module supplies (§IV-A). For instance, for a BPSK-modulated information ($0 \rightarrow +1, 1 \rightarrow -1$) transmitted over a Rayleigh fading channel with AWGN noise, this is computed as $1/(1 + e^L)$, where L is the LLR of the bit (say c_j) that represents b_j. If the *channel state information* (CSI) is known at the receiver, at least statistically where the first and second order moments of the channel are characterized, L is computed as $2\mu_h c_j/(\sigma^2 + \sigma_h^2)$, where $\mu_h (= a\sqrt{\pi/2})$ and $\sigma_h^2 (= a^2(4 - \pi)/2)$ are the mean and variance of the Rayleigh distribution with scale parameter a, and σ^2 is the AWGN noise variance [22]. If the CSI is unknown at the receiver, this probability is computed using $L = \log(\psi(c_j/\sqrt{2\tilde{\sigma}})/\psi(-c_j/\sqrt{2\tilde{\sigma}}))$, where $\tilde{\sigma} = \sigma^2(1 + 2\sigma^2)$ and $\psi(x) = 1 + \sqrt{\pi}x\text{exp}(x^2)\text{erfc}(-x)$ [23]. We note that C_R is minimum for a $b_j \in \{0, 1\}$ that has a greater probability of being the corresponding bit at the encoder. C_R introduces only linear terms into the objective QUBO (Eq. 6).

Summary. In the above sections (§IV-A, §IV-B), we have described HyPD and QPD. These designs are purely algorithmic, and they can be implemented on hardware suitable for QUBO/Ising optimization problems [24]. While in this work we investigate QA technology, we note that implementing the same ideas using specialized classical hardware is also a promising possibility. The following sections (§IV-C, §IV-D) describe and address the unique challenges in the application of QA for Polar Code decoding, which include problem formulation, embedding, quantum annealing correction, and parameter tuning for real QA devices.

C. QPD-MAP: Problem Embedding

In this section, we first describe *quantum annealing correction* (QAC), and then explain QPD-MAP, our customized problem mapping scheme tailored for QPD.



(a) Unprotected **(b)** NP-QAC **(c)** EP-QAC

Figure 9: The figure shows QAC schemes with $N_{QAC} = 3$, where $s_{i\{a,b,c\}}$ represent $s_i \forall i$, and $s_{P\{1,2,3\}}$ are penalty qubits. Complete-QAC is EP-QAC with a majority vote decision (§IV-C1).

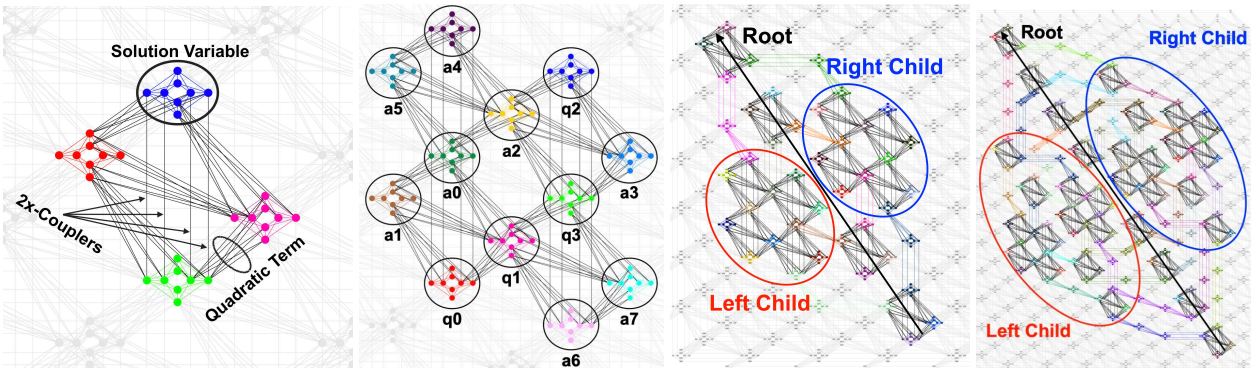
1) Quantum Annealing Correction: QAC is a strategy that aims to correct the QA computational errors introduced into the problem at hand, due to qubit decoherence, analog noise, among others [12], and it provides error protection to the problem in two ways: First, by increasing the energy scale of the problem, and second, by a majority vote decision for solution variables. QAC methods achieve increased energy scale by solving multiple copies of the same problem while correlating the qubits that represent the same solution variables (across copies). This can be expressed as the problem Hamiltonian transformation [25]:

$$\overline{H}_P = \sum_{i=1}^{N_{QAC}} (H_P)_i + \beta H_{penalty} \tag{10}$$

where $\overline{H}_P$ is the problem Hamiltonian resulted with QAC, N_{QAC} is the number of problem copies, $(H_P)_i$ is the problem Hamiltonian of the i^{th} copy, $H_{penalty}$ is the Hamiltonian resulted from correlating the qubits representing the same solution variable, and β is its energy scaling factor. If $\beta = 0$, then it is called *No-Penalty* QAC (NP-QAC), and if $\beta < 0$, then it is called *Encoded-Penalty* QAC (EP-QAC). EP-QAC with a majority vote decision for solution variables is called *Complete-QAC* [25]. To understand QAC, let us consider an example Ising problem:

$$E = J_{12}s_1s_2 + J_{23}s_2s_3 + J_{13}s_1s_3 \tag{11}$$

Figure 9 depicts various QAC graphs for this problem, where nodes represent qubits and edges represent quadratic coupler terms. In particular, Fig. 9(a) shows the direct, QAC-unprotected, connectivity of Eq. 11, and Fig. 9(b) shows its NP-QAC connectivity with N_{QAC} = 3 (*i.e.,* three copies of the same problem), where $s_{i\{a,b,c\}}$ represent s_i $\forall i$. In comparison to QAC-unprotected, NP-QAC provides better quality solutions due to increased problem energy scale



(a) Four-clique mapping **(b)** N_L = 4 bits **(c)** N_L = 8 bits **(d)** N_L = 16 bits

Figure 10: QPD-MAP embedding on Pegasus QA hardware, for various sub-block sizes (N_L). Nodes and edges in the figure are physical qubits and physical couplers, respectively.

resulting from multiple copies of the same problem. We next see in Fig. 9(c) the EP-QAC connectivity of Eq. 11 with N_{QAC} = 3, where the qubits representing the same variable (s_{1a}, s_{1b}, s_{1c}) are chained to a penalty qubit (s_{P1}) with a strong negative coupler strength (β). This chaining in EP-QAC forces the qubits representing the same variable to agree in the solution the machine outputs, therefore improving solution quality further in comparison to NP-QAC. Complete-QAC is EP-QAC with a majority vote decision for solution variables, thus it helps recover any errors across problem copies. QAC graphs also need minor embedding for running the problem.

2) QPD-MAP: The above sections have described minor embedding and QAC in general terms (§II-0d, §IV-C1). Here we describe how we bring together these strategies for mapping QPD's design onto QA hardware. QPD-MAP is our customized problem mapping scheme tailored for QPD, and it embodies minor embedding and Complete-QAC with $N_{QAC} = 8$. Let us recall from §IV-B that only Node constraints introduce quadratic terms into the QPD's QUBO design, and so they require embedding. Each Node constraint is a sum of *quadratic forms* (see Eq. 7), where each quadratic form consists of four solution variables, and so the connectivity of each such quadratic form is essentially a four-clique.

Construction. QPD-MAP realizes each such four-clique connectivity on the QA hardware using four unit cells as shown in Fig. 10(a), by mapping a single node of this four-clique (*i.e.,* a solution variable) on to eight physical qubits present in a unit cell, and a single edge of this four-clique (*i.e.,* a quadratic term) on to eight physical couplers connecting two unit cells, therefore realizing eight copies of the problem. Figure 10(a) depicts this construction on state-of-the-art

Pegasus QA hardware, showing eight qubits representing a solution variable and eight couplers representing a quadratic term. In order to make all the qubits within a unit cell behave like a single solution variable, QPD-MAP pairwise chains these qubits with a strong negative coupler strength, thus forcing them to agree. The vertically aligned unit cells (see Fig. 10(a)) have only four available couplers connecting them, and so we double the strength of each such coupler, effectively realizing an eight-coupler connectivity (see 2x-Couplers in Fig. 10(a)). QPD-MAP follows a majority vote decision for solution variables. Therefore, this construction of QPD-MAP embodies minor embedding and Complete-QAC with $N_{QAC} = 8$.

Placement. QPD-MAP next places the four-cliques that share common solution variables close to each other, then makes the qubits representing the same solution variable agree. To understand this, let us visualize a binary tree with N_L leaf nodes as a root node connected to two shorter binary trees with $N_L/2$ leaf nodes each. QPD-MAP follows this recursive property of binary trees: We first place the four-cliques corresponding to the root node along the top-left bottom-right diagonal of the Pegasus graph, and the four-cliques corresponding to its children trees symmetrically on either side of this diagonal as shown in Figs. 10(c) and 10(d). Figures 10(b), 10(c), and 10(d) depict QPD-MAP designs for sub-blocks with size N_L of four bits, eight bits, and 16 bits respectively. We observe that Fig. 10(b)'s qubit connectivity is same as that of Fig. 10(c)'s left and right children trees, and Fig. 10(c)'s qubit connectivity is same as that of Fig. 10(d)'s left and right children trees, therefore admitting the aforementioned recursive property of binary trees, making QPD-MAP flexible to varying sub-block sizes. We next describe Fig. 10(b).

Figure 10(b) shows the QPD-MAP design for Sub-block 1, whose Node constraints are (§IV-B2): $C_N(T_{0,i}) = 0 \forall i, C_N(T_{1,0}) = (q_0 + q_1 - a_0 - 2a_1)^2, C_N(T_{1,1}) = (q_2 + q_3 - a_2 - 2a_3)^2, C_N(T_{2,0}) = (a_0 + a_2 - a_4 - 2a_5)^2 + (q_1 + q_3 - a_6 - 2a_7)^2$, where $T_{2,0}$ is the root node, $\{T_{1,0}, T_{0,0}, T_{0,1}\}$ constitutes its left child tree, and $\{T_{1,1}, T_{0,2}, T_{0,3}\}$ constitutes its right child tree (see Fig. 8). We see in Fig. 10(b) the two four-cliques of the root node (*i.e.,* $C_N(T_{2,0})$) placed along the top-left bottom-right diagonal, with its left child (*i.e.,* $C_N(T_{1,0})$) and right child (*i.e.,* $C_N(T_{1,1})$) four-clique connectivities placed symmetrically on either side of this diagonal, where each four-clique follows the aforementioned construction scheme (Fig. 10(a)).

Let us note that upon expansion of C_N, the QUBO quadratic coefficients of terms involving two ancillary qubits (*e.g.,* $a_0 a_1$) take a value of $+4$, which in Ising form take a $+1$ value, which is the maximum supported value for a quadratic coefficient (see §IV-D). Therefore, such coupler strengths cannot be doubled. QPD-MAP avoids placing odd-indexed ancillary variables (*e.g.,*

a_1, a_3, a_5, a_7) in unit cells that require 2x-Couplers, therefore ensuring that all the coefficients fall into the supported range of QA. For instance, note that a_1, a_3, a_5, a_7 do not appear in unit cells that require 2x-Couplers (*cf.* Figs. 10(a) and 10(b)).

D. Practical Annealer Considerations

We now determine the best choices for QPD's weights W_N, W_F, W_R (Eq. 6), and then fine-tune the coefficient values for achieving a greater probability of decoding correctly.

To optimize a QUBO problem on a QA device, it must be specified as an equivalent Ising problem (§II). This conversion is characterized by the coefficient transformations [26]:

$$h_i = \frac{f_i}{2} - \sum_{\forall j} \frac{g_{ij}}{4}, \qquad J_{ij} = \frac{g_{ij}}{4} \tag{12}$$

where h_i and J_{ij} are Ising coefficients corresponding to linear (f_i) and quadratic (g_{ij}) QUBO coefficients respectively. Current QAs support values for $h_i \in [-4, +4]$ and $J_{ij} \in [-2.0, +1.0]$.

a) Choice of the weight W_N: As noted above in §IV-B2, all the QPD's quadratic QUBO coefficients (g_{ij}) take values in $\{-4, -2, +2, +4\}$ only. This implies that their respective Ising coefficients (J_{ij}) take values in $\{-1, -0.5, +0.5, +1\}$ (from Eq. 12). For any $W_N > 1.0$ (in Eq. 6), the quadratic Ising coefficients J_{ij} fall outside their supported values (*i.e.,* $> +1$), and hence must be normalized back again to bring coefficients into the supported range. For any $W_N < 1.0$, the priority of the Node constraints is undermined. Hence, we consider $W_N = 1.0$.

b) Choice of the weight W_F: Looking at Eqs. 6 and 8, we note that W_F is a linear coefficient, and that a high W_F strongly enforces the qubits representing frozen bits to take a zero value. Thus, we set W_F to the maximum supported linear coefficient value of $+4$ for every physical qubit that represents a frozen bit.

c) Choice of the weight W_R: The best choice of W_R depends on sub-block's size, coding rate, and channel SNR. We first demonstrate these dependencies, and then present our choice.

From §IV-B2, we see that the number of Node constraints QPD introduced into the QUBO objective function is proportional to the sub-block's size. Higher sub-block's size implies more nodes in the sub-block's binary tree, hence more Node constraints. Thus longer sub-blocks tend to agree on more Node constraints for error correction, implying that they balance greater W_R.

Code rate determines the number of frozen bits. When the number of frozen bits is high, more Node constraints tend to work correctly. For instance, let us consider the Node constraint: $C_N = (q_0 + q_1 - a_0 - 2a_1)^2$. If q_0 and q_1 represent frozen bits (*i.e.,* q_0, q_1 = 0), then it is certain

that a_0 (= $q_0 \oplus q_1$) takes the correct zero value in the minimum energy solution. If q_0 and q_1 do not represent frozen bits, then a_0 depends on the received information, and thus may or may not take a correct zero or one value. Hence, lower coding rates cause a greater number of Node constraints to work properly, and thus allow greater values for W_R.

A higher channel SNR implies that the received information has a lower probability of experiencing errors. Thus, a higher SNR allows greater values for W_R. Empirical evaluations are performed to select the best W_R value: At $W_R = R_{\text{sub}}, 1 - R_{\text{sub}}, 2 - R_{\text{sub}}, 3 - R_{\text{sub}}, 4 - R_{\text{sub}}$, the average correct answer probability of HyPD is 0.34, 0.42, 0.81, 0.56, 0.32 respectively. Therefore, we choose $W_R = 2 - R_{\text{sub}}$ (see §V for implementation details).

d) Choice of embedding coupler strength: The purpose of embedding is to make qubits agree with each other, and so we must prioritize embedding couplers. Therefore, we assign the minimum supported value for embedding coupler strengths ($J_F = \beta = -2.0$).

e) Fine-tuning the physical Ising coefficients: We select W_N, W_F, W_R, and J_F as aforementioned and obtain the physical Ising problem. Here we describe our approach in tailoring the physical Ising coefficients closely into the QA's supported bit precision of 4–5 bits [27].

If a solution variable with linear Ising coefficient h_i is mapped onto A number of physical qubits, then the QA default auto-scaling methods assign h_i/A value to each qubit (*i.e.,* equal sharing) [28]. This auto-scaling approach therefore reduces the coefficient precision greatly when A is large. Thus, we opt for an unequal sharing of h_i, in steps of 1.0 value. For example, if a solution variable has a linear coefficient of 2.0, then eight copies of this variable requires an effective coefficient value of 16.0 ($N_{QAC} = 8$). If this solution variable is mapped on to 24 physical qubits in the embedding process, then 16 of the physical qubits take 1.0 coefficient value and the rest eight physical qubits take a zero coefficient value. Further, the linear coefficients are programmed only on to the physical qubits involved in the four-cliques (*i.e.,* not on to the qubits involved in the minor embedding chains), avoiding any errors due to long range chains. This uneven sharing ensures that the coefficient precision is much less disturbed than that of auto-scaling methods. By design, QPD's Ising quadratic coefficients fall into the supported range and precision of QA. As noted above, they take values in $\{-2, -1, -0.5, +0.5, +1\}$ only, where -2 is the embedding coupler strength. These values are sufficiently separated in magnitude for the QA to distinguish within the 4–5 bit precision.

V. IMPLEMENTATION

We implement HyPD's classical module on a 2.3 GHz eight-core Intel CPU with 14 nm CMOS process, and quantum module on 5,627-qubit Advantage_system4.1 QA. Our decoder targets CRC-assisted Polar codes in the 5G-NR Physical Uplink Control Channel (PUCCH) with block length of 1,024 bits. We consider PUCCH's BPSK modulation scheme and 200 message data bits, which is typically the maximum UCI payload in LTE and 5G-NR eMBB scenarios [13]. Our encoder implementation follows 5G-NR specifications [29]. In particular, a 11-bit CRC is attached to the user data, frozen bits and sub-channels are allocated, and then the mother Polar code encoding is performed as in §III. This encoded data is passed onto sub block and channel interleavers, and then transmitted over a wireless Rayleigh fading channel. At the receiver, we de-interleave the received soft information accordingly and then perform HyPD's decoding.

Current QAs have a 4–40 μs coefficient programming time, 0–10 ms post-programming thermalization time, 25–150 μs solution readout time, and 0–10 ms post-readout thermalization time. Thermalization times are user-specified, and we set it equal to default 1 ms. These overhead times, however, can be reduced several orders of magnitude by system integration [11]. In our particular QA device, there are 13 defective qubits, each in a different unit cell, and we use only 7 available physical qubits in such unit cells. Practical challenges include embedding, coefficient range and precision, and analog QA machine noise called *integrated control error* (ICE). ICE is caused by qubit flux-noise, quantization, among others [12], and it alters problem biases ($h_i \rightarrow h_i \pm \delta h_i$) and coupler strengths ($J_{ij} \rightarrow J_{ij} \pm \delta J_{ij}$). Although the errors δh_i and δJ_{ij} are currently on the order of 10^{-2}, these may disturb the solution quality in scenarios where ICE noise erases significant information from the problem. Nevertheless, we increase the solution quality via the standard method of running multiple anneals for a problem. Our end-to-end evaluation results capture all the sources of QA imprecision.

VI. EVALUATION

Our experimental evaluation begins with our experimental methodology description (§VI-A). We measure performance over Rayleigh fading channels at low SNR regimes of practical interest, in both known partial statistical CSI and unknown CSI scenarios at the receiver. QPD's evaluation provides detailed insights into QA performance (§VI-B). End-to-end experiments compare HyPD head-to-head against successive cancellation list (SCL) decoders with list size of eight (§VI-C).

A. Experimental Methodology

Let us define an *instance I* as a 1,024-bit Polar decoding problem. We partition each instance into N_{sub} = 128 sub-blocks with N_L = 8 bits each (as described in §IV-A). For each sub-block I_{sub}, we perform $2,000$ anneals with 1 μs annealing time, where each anneal potentially returns a distinct solution to the sub-block due to the heuristic sampling nature of the QA. If $N_s^{I_{\text{sub}}}$ is the number of distinct solutions returned for a sub-block, we rank these solutions in increasing order of their energies as $R_1, R_2, ..., R_{N_s^{I_{\text{sub}}}}$, and note the solutions' bit errors and occurrence probabilities. Eight sub-blocks are parallelized in a single QA anneal, by mapping these sub-blocks to distinct physical locations in the QA hardware.

1) BER Evaluation: If $R_{\min}$ is the rank of the minimum energy solution in a particular population sample of $N_a <$ 2,000 anneals, we compute the expected number of bit errors (N_B) in a sub-block I_{sub} over performing N_a anneals as:

$$E(N_B^{I_{\text{sub}}}|N_a) = \sum_{i=1}^{N_s^{I_{\text{sub}}}} \Pr(R_{\min} = R_i|N_a) \cdot N_B(R_i), \tag{13}$$

where the probability of $R_{\min}$ being R_i given N_a anneals can be computed using the cumulative distribution function $F(\cdot)$ of observed solution probabilities in 2,000 anneals as [17]:

$$\Pr(R_{\min} = R_i|N_a) = (1 - F(R_{i-1}))^{N_a} - (1 - F(R_i))^{N_a}, \tag{14}$$

If K is the number of message data bits in an instance I, and N_I is the total number of instances, then we compute the overall bit error rate (BER) as:

$$BER = \sum_{\forall I} \sum_{i=1}^{N_{\text{sub}}} E(N_B^{I_{\text{i}}}|N_{a,I_i}) \Big/ K \cdot N_I. \tag{15}$$

2) BLER Evaluation: A block is error-free iff all the bits in the block are decoded correctly, where each block is an instance. We compute the probability of instance I being error-free as:

$$\Pr(I_{ef}) = \prod_{m=1}^{N_{\text{sub}}} \Big\{ \sum_{\forall i} \Pr(R_{\min} = R_i|N_a, I_m, N_B^{I_m}(R_i) = 0) \Big\}, \tag{16}$$

Then we compute the overall block error rate (BLER) as:

$$BLER = \sum_{\forall I} \Big\{ 1 - \Pr(I_{ef}) \Big\} \Big/ N_I. \tag{17}$$

3) FER Evaluation: A frame is said to be error-free if and only if all the blocks in a frame are decoded correctly, where each block is decoded independently. If a given frame consists N_F blocks, we compute the overall frame error rate (FER) as:

$$FER = 1 - (1 - BLER)^{N_F}. \tag{18}$$

Figure 11: QPD's performance of partitioned sub-blocks at SNR 1 dB, in both *minorminer* and QPD-MAP embeddings, showing first 20 lowest energy solutions. Sub-blocks with low coding rates achieve high Rank 1 solution probability. At coding rate of 0.25 in QPD-MAP, only eight distinct solutions are returned by the QA. QPD-MAP achieves lower bit errors than minorminer.

B. QPD's Sub-block Performance

This section reports the performance of QPD's sub-blocks, in both QPD-MAP and the QA's default, state-of-the-art, *minorminer* embeddings [28].

In Fig. 11, we see the probability and bit error statistics of solutions returned by the QA for sub-blocks at various coding rates. In minorminer embedding, sub-blocks with 0.25 coding rate achieve a high probability of finding the correct answer (*i.e.,* Rank 1 solution), whereas sub-blocks with higher coding rates (0.5, 0.75, 1.0) achieve a significantly low correct answer probability (*i.e.,* barely 1–4 anneals out of 2,000 anneals returned the minimum energy solution). This is because in minorminer embedding, the absence of QAC makes the problems highly sensitive to QA ICE noise (§V), which in turn significantly degrades the solution quality. Further, we note that the number of bit errors in solutions increase with increased coding rate. Solutions with higher rank (> 1) and zero bit errors imply that ancillary qubits, but not solution qubits that represent user data, are errored (§IV-B). In QPD-MAP embedding, sub-blocks with 0.25 and 0.5 coding rates achieve high correct answer probability, whereas sub-blocks with 0.75 and 1.0 coding rates achieve relatively low correct answer probability. This is because at high coding rates the best choice of W_R is low (§IV-D), which leads to low energy gap between the

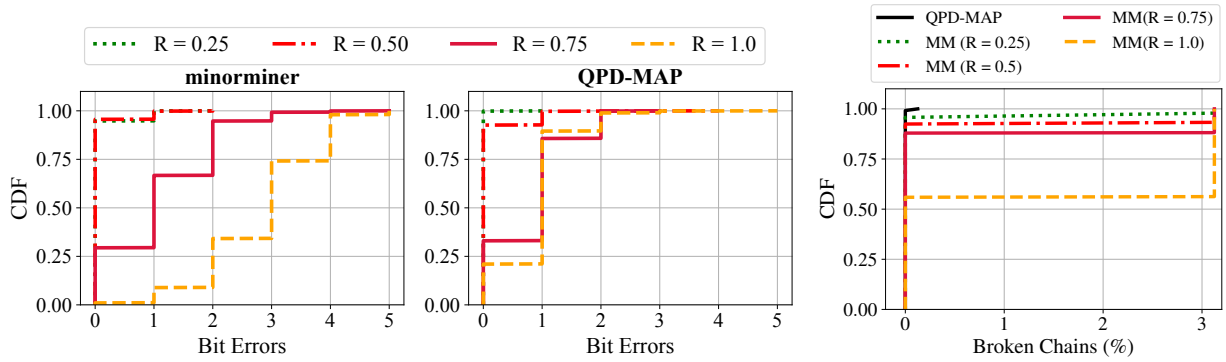


Figure 12: The figure shows bit errors and broken chains distribution, across solutions returned by the QA, in both minorminer and QPD-MAP, for sub-blocks at various coding rates. QPD-MAP has near zero broken chains at all coding rates.

minimum energy solution and the rest, thus making it difficult for the QA to distinguish the minimum energy solution. Nevertheless, QPD-MAP achieves higher correct answer probability and lower bit errors than the default minorminer embedding, at all coding rates.

We next see in Fig. 12 (*Left* and *Middle*) the distribution of bit errors for sub-blocks at various coding rates. The plots show that at 0.25 and 0.5 coding rates, most of the solutions have zero bit errors in both minorminer and QPD-MAP embeddings. At higher coding rates (0.75 and 1.0), the number of bit errors drastically increase in minorminer, reaching a worst case scenario where 25% of the solutions have more than four bit errors. In QPD-MAP embedding, only about 10–15% of the solutions have more than two bit errors, at high 0.75 and 1.0 coding rates. We next investigate broken chain statistics in Fig. 12 (*Right*). Broken chains are embedding chains where qubits do not agree, they thus degrade the solution quality [17]. The figure shows that broken chains are more frequent in minorminer embedding than in QPD-MAP embedding. In summary, our results show that QPD achieves higher correct answer probability, lesser bit errors, and lesser broken chains with our QPD-MAP embedding than with minorminer embedding. We therefore consider QPD in combination with QPD-MAP for HyPD's system performance evaluation heretofore.

C. HyPD's System Performance

This section presents HyPD's end-to-end system performance, comparing head-to-head against SCL decoders, in both known partial statistical CSI and unknown CSI scenarios at the receiver. Figures 13, 14, and 15 report these results.

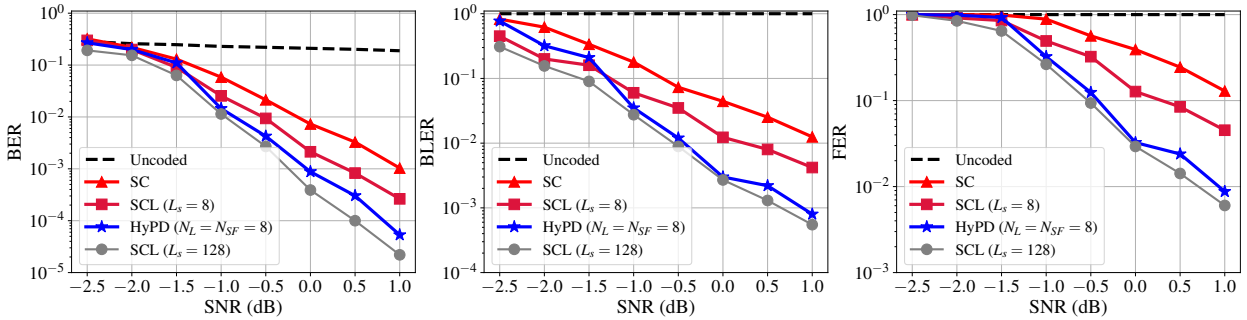


Figure 13: HyPD's end-to-end system performance when partial statistical CSI is known at the receiver, in Rayleigh fading channels. L_s is SCL decoder's list size, N_L and N_{SF} are HyPD's sub-block size and number of solutions fed back respectively. (*Right*) Frame size is 1500 bytes.

1) Error Performance: In Fig. 13 (*Left*), we first investigate how HyPD's end-to-end BER behaves as the wireless channel SNR varies when partial statistical CSI is known at the receiver (§IV-B4). At regions of channel SNRs less than -1 dB, HyPD's BER barely lags behind that of SCL(8) decoder. As we meet SNRs greater than -1 dB, we observe HyPD's BER curve drops down, outperforming SCL(8) decoder by half an order of BER magnitude at 1 dB SNR. This performance improvement of HyPD is a result of QPD exploring all 2^{N_L} decoding paths within each sub-block. We next see in Fig. 13 (*Middle*) the BLER performance of HyPD in the same scenarios. The figure shows that at SNR 1 dB, HyPD achieves nearly an order of magnitude lower BLER than that of SCL(8), indicating that majority of bit errors HyPD experienced are distributed across a minority of code blocks. We next see in Fig. 13 (*Right*) the FER performance at a frame size of 1,500 bytes (*i.e.,* 11 blocks per frame). The figure shows that at 1 dB SNR, HyPD achieves a 99.1% frame delivery rate whereas SCL(8) achieves only 95.5%. In all the plots, SCL(128) curves are shown for reference, which is algorithmically more rigorous than HyPD when $N_L = N_{SF} = 8$. We next investigate how HyPD performs when CSI is unknown at the receiver, in Fig. 14, along the same error performance metrics. Fig. 14 (*Left*) depicts BER performance, showing that SC and SCL decoders achieve almost similar performance as that of known CSI scenario, and the performance gap between HyPD and SCL(8) is barely decreased at all SNRs (*cf.* Fig. 13). A similar analogy can be observed in Figs. 14 (*Middle*, *Right*) as well.

2) Timing Analysis: We now analyze QA compute time, describing current technology points and predicted future with increased QA qubit counts.

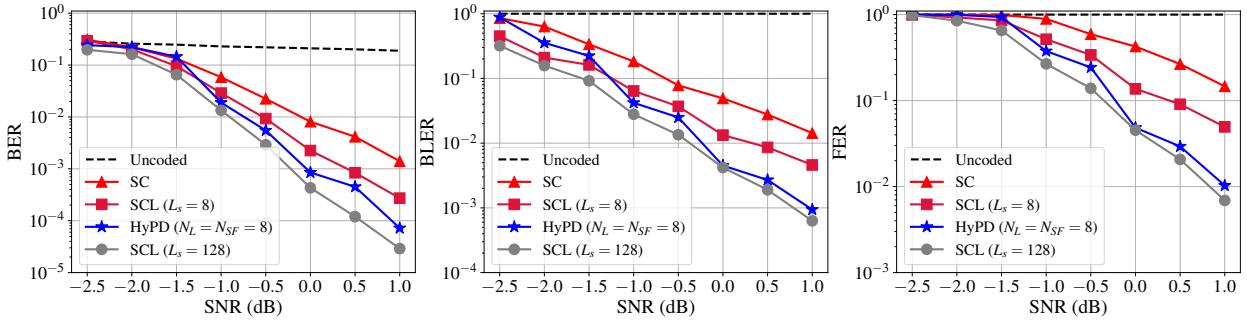


Figure 14: HyPD's end-to-end system performance when CSI is unknown at the receiver, in Rayleigh fading channels. L_s is SCL decoder's list size, N_L and N_{SF} are HyPD's sub-block size and number of solutions fed back respectively. (*Right*) Frame size is 1500 bytes.

We first measure *time-to-solution* (TTS), the time required to reach the ground state of input problem. Since QA is probabilistic in nature, TTS(P) can be used to understand the time required to reach the minimum energy solution with a target probability P. It is computed as [30]:

$$TTS(P) = T_a \cdot \log(1-P)/\log(1-P_1) \tag{19}$$

where T_a is the annealing time and P_1 is the probability of R_1, the minimum energy solution. The factor $\log(1-P)/\log(1-P_1)$ indicates the number of repetitions/anneals required to reach the desired success probability *P*. If P_f problems are solved in parallel, then the effective TTS is reduced by a parallelization factor P_f. In current QA devices with 5K qubits, eight QPD problems can be parallelized ($P_f = 8$), whereas the projected parallelization in near-term future QA devices with 14K, 35K, and 70K qubits is 20, 50, and 100 respectively.

In Fig. 15 (*Left*), we see TTS(99%) performance of QPD with $T_a = 1\ \mu s$. The figure shows that TTS scales proportionally with sub-block's coding rate, reaching a worst-case 70 μs for sub-blocks with a 1.0 coding rate (P_f = 8). This is because at high coding rates, R_1 solution probability is low (§VI-B). With P_f = 100, this worst-case TTS reaches to 5.6 μs. We further note that TTS at coding rate of 1/8 deviates from the trend. This is because at very low coding rates, the energy gap between the minimum energy solution and the rest becomes significantly low, making it difficult for the QA to distinguish the minimum energy solution. To overcome this issue, either QAC with $N_{QAC} < 8$ or finer coefficient settings may be employed to relax the effect of embedding chains–we leave for future work.

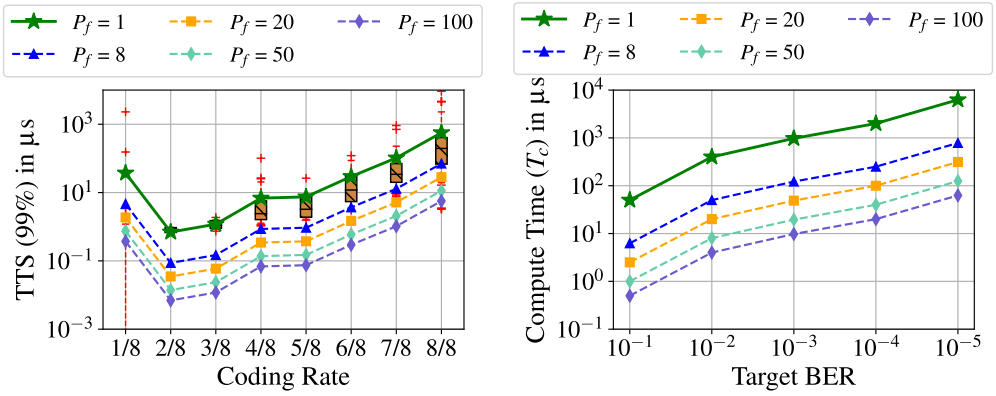


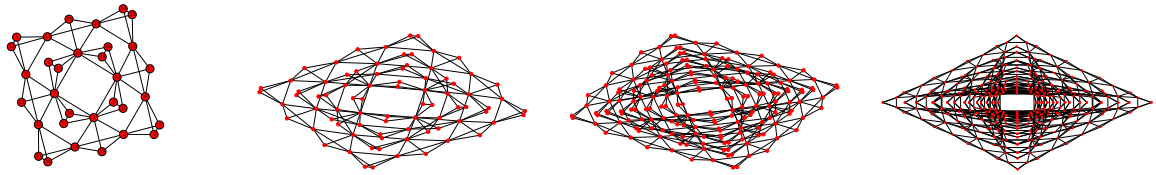
Figure 15: HyPD's timing analysis. The boxes' lower/upper whiskers and quartiles are $10^{th}/90^{th}$ and $25^{th}/75^{th}$ percentiles respectively. Line trends show averages, and P_f is parallelization factor.

While TTS shows the time required to reach the ground state of an input problem, it does not capture the effect of bit errors the ground state may have. Therefore, we next measure the *compute time* (T_c) required to reach a certain BER target. It is computed as:

$$T_c = \frac{1}{P_f} \sum_{i=1}^{N_{\text{sub}}} (T_a)_i \times (N_a)_i \tag{20}$$

where N_{sub} is the number of sub-blocks, $(T_a)_i$ and $(N_a)_i$ are annealing time and number of anneals of the i^{th} sub-block respectively. For this evaluation, we set a target BER value in Eq. 15 and then calculate back N_a and obtain T_c. Sub-blocks with only frozen bits do not contribute to the total compute time. Fig. 15 (*Right*) depicts HyPD's compute time requirements at SNR 1 dB, showing that higher compute times achieve lower BER. For a target BER of 10^{-4}, compute time required is 250 μs (P_f = 8). With P_f = 100, this time reduces to 20 μs.

3) Throughput considerations: HyPD solves eight eight-bit sub-blocks (*i.e.,* $N_L = P_f = 8$) in 1 μs annealing time, which means that the best case achievable throughput on current QAs is 64 Mbps only, which is well behind the achievable throughput on classical compute devices where SCL decoders with up to 4–5 Gbps throughputs have been demonstrated [31]. However, this throughput limitation on current QAs is not fundamental, and can be increased several orders of magnitude with faster anneal times, increased qubit counts, better qubit connectivity, among others [11], [32]. While these advancements are underway, we next demonstrate an ideal QA hardware connectivity structure for Polar codes, which with 10^7 qubits and 1 μs annealing time can achieve throughputs up to $524 \times R$ Gbps, where R is coding rate.



(a) N = 8 bits. **(b)** N = 16 bits. **(c)** N = 32 bits. **(d)** N = 64 bits.

Figure 16: QPD-HW: Quantum Polar Decoder's Hardware qubit connectivity design for decoding Polar codes. Nodes and edges represent qubits and couplers respectively.

VII. QPD-HW Hardware Design

In this section, we propose an ideal QA hardware structure tailored to the task of decoding Polar codes (**QPD-HW**). The motivation for this section stems from the challenges existing in current hardware implementation such as the need to boost decoder throughput (§VI-C3), circumvent embedding (§II-0d), and reduce hardware resource usage to mitigate ICE noise (§V).

The requirement of embedding is a major impediment to leveraging the QA technology for practical applications because of mapping difficulties. The problem of embedding arises because of the lack of all-to-all qubit connectivity in the QA hardware. While such an all-to-all qubit connectivity is desired for natively encoding generic problem graphs, near-term engineering considerations allow for scaling the QA hardware with local qubit connectivities. In this section, we envision a scenario where near-term quantum hardware is tailored for specific problem needs, where a part of qubit connectivity is dedicated for wireless community, for decoding Polar codes.

An ideal hardware structure that circumvents the requirement of embedding is the connectivity graph of the QUBO (*i.e.,* the graph of Eq. 6). We next demonstrate this connectivity which may be of interest for QPD-HW engineering considerations. Fig. 16 shows the connectivity structure of the QPD's QUBO design at various Polar code block lengths. In particular, nodes in the figure represent variables in Eq. 6, and edges represent the quadratic terms of Eq. 6. These nodes and edges then constitute qubits and couplers respectively in the proposed hardware design. In Fig. 16(a), we see QPD's qubit connectivity in a Polar code of block length eight bits. This code requires a total of 32 qubits, with several four-cliques. Looking at Eq. 7, we see that the connectivity of these four-cliques mirrors the connectivity of QPD's Node constraints. Since QPD's QUBO formulation is the exact representation of the Polar encoder binary tree structure, the proposed hardware also scales with the features of a perfect binary tree as follows:

a) Flexible support for block lengths: Since the children of a node in a perfect binary tree are also perfect binary trees, QPD-HW for a code of block length 2^d bits can be used to solve 2^k independent codes of block length 2^{d-k} bits, in parallel. For example, Fig. 16(b) shows QPD-HW for 16-bit codes, can also be used to solve two 8-bit codes in parallel. This is because two copies of Fig. 16(a) are present in Fig. 16(b) as independent subgraphs.

b) Flexible support for coding rates: A varying coding rate affects only the Frozen constraints of the QPD. Since the Frozen constraints do not introduce quadratic terms into the objective QUBO (§IV), the proposed hardware allows decoding of various coding rates.

c) Connectivity features: The proposed hardware qubit connectivity is highly sparse. In the proposed hardware, the number of couplers per qubit, for a code of block length 2^d bits, takes all values in positive integer multiples of three with a maximum value of 3*d*. For instance, observe in Fig. 16(a) that the number of couplers per qubit takes values in $\{3, 6, 9\}$ only. Further, if N_k is number of qubits with $3k$ edges ($k \in [1, d]$), then $N_d = 4$, $N_{d-1} = 8$, and $N_k = 2N_{k+1} + 2^{d-k}$ ($k \in [1, d-2]$). For instance, observe in Fig. 16(a) that the number of qubits with $\{3, 6, 9\}$ edges is $\{20, 8, 4\}$ respectively. To decode a Polar code of block length 2^d bits, the proposed hardware requires $(d+1) \times 2^d$ qubits and $3d \times 2^d$ couplers. This implies that with 1 μs annealing time and $10^4, 10^5, 10^6, 10^7$ qubits, the proposed hardware can achieve processing throughputs up to $0.12, 1.02, 65.5, 524 \times R$ Gbps respectively, where R is the coding rate. State-of-the-art D-Wave QA hardware has 5,627 qubits and 40,279 couplers, and so supports decoding Polar codes of block lengths up to 2^7 bits. While these block lengths are of useful sizes employed in the 5G-NR standard (2^5–2^{10} bits) [29], if QA designers were to reconnect the currently available qubits and couplers to form our proposed QPD-HW structure, Polar codes of block lengths up to 2^9 bits may be feasbily decoded in the near term.

VIII. Related Work

Polar codes' fundamental construction and properties are well studied [33]–[35] and though proven in theory to be capacity-achieving [1], their use is limited to short block lengths due to their computationally-complex decoding algorithms. Several studies to this end have proposed efficient decoder architectures based on the inferior SC algorithm [8], [36], whereas HyPD compares favorably in performance (§VI) against the superior SCL decoding algorithm. Reconfigurable decoders for LDPC and Polar codes also exist [37]–[39]: in this class of decoders, belief propagation and SC decoding perform similarly [38]. Further studies provide insights into

the comparison of Polar codes with other capacity achieving codes [40], [41], where [40] shows Polar codes outperform LDPC codes, and [41] shows that Polar codes decoded via the SCL algorithm match the performance of that of LDPC and Turbo codes.

QA machines have been recently used to successfully optimize wireless communication problems [17], [42]–[45]. Quantum Gate-based approaches are also being widely investigated, wherein quantum approximate optimization algorithm (QAOA), quantum search methods, and syndrome-based decoding schemes have been proposed [46]–[48]. A holistic cost and power feasibility analysis of QA-based wireless communications is conducted in [11]. Some of the prior work [17] studies QA-based LDPC decoding towards formulating a QUBO problem with a customized embedding scheme. However, the embedding design studied in Ref. [17] is applicable only to (2,3)-regular LDPC codes and not to Polar codes. Further, the embedding scheme presented in Ref. [17] does not generalize to higher LDPC check bit degrees. As a result, these studies will lose efficiency in their embeddings when their QUBO designs are employed for other code parameters. QPD instead proposes a QUBO design that can be mapped on to QA hardware for various code rates and block lengths flexibly, via QPD-MAP, and also enables a novel QA hardware structure, QPD-HW, that circumvents the requirement of embedding altogether, thus significantly advancing the state-of-the-art over prior work.

IX. Conclusion

HyPD leverages classical and quantum annealing computational structures for decoding Polar codes, proposing a fresh QA based design and evaluating it experimentally on real leading-edge QA hardware. Our studies demonstrate the practical challenges today in using such devices. We further present a customized problem mapping scheme on current hardware, and propose an ideal QA hardware structure that is sparse, flexible under different code rates and block lengths, and inherently meshes with the Polar code's binary tree construction. The ideas we propose here may inform NextG wireless networks and in the more distant future enable applicability of long Polar codes in practical protocol standards.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1824357. KJ and SK gratefully acknowledge a gift from the InterDigital Corporation that helped support this work.

REFERENCES

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] E. Şaşoğlu, E. Telatar, and E. Arikan, "Polarization for arbitrary discrete memoryless channels," in *2009 IEEE Information Theory Workshop*, pp. 144–148.

[3] A. Eslami and H. Pishro-Nik, "On bit error rate performance of Polar codes in finite regime," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 188–194.

[4] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[5] E. Arikan, N. ul Hassan, M. Lentmaier, G. Montorsi, and J. Sayir, "Challenges and some new directions in channel coding," *Journal of Communications and Networks*, vol. 17, no. 4, pp. 328–338, 2015.

[6] E. Şaşoğlu, "Polar coding theorems for discrete systems," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2011.

[7] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. on Info. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[8] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for Polar codes," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 289–299, 2013.

[9] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[10] P. Yang, Y. Xiao, M. Xiao, and S. Li, "6G wireless communications: Vision and potential techniques," *IEEE Network*, vol. 33, no. 4, pp. 70–75, 2019.

[11] S. Kasi, P. Warburton, J. Kaewell, and K. Jamieson, "A cost and power feasibility analysis of quantum annealing for NextG cellular wireless networks," *arXiv preprint arXiv:2109.01465*, 2021.

[12] D-Wave Systems User Manual, "Technical description of the D-Wave Quantum Processing Unit," pp. 09–1109A–O, 2019.

[13] 3GPP TSG-RAN WG1 #88 R1-1703106. Nokia, Alcatel-Lucent Shanghai Bell, "Polar design for control channels," 2017.

[14] D-Wave, "D-Wave QPU Architecture: Topologies," *Website*, 2021.

[15] C. Baldassi and R. Zecchina, "Efficiency of quantum versus classical annealing in non-convex learning problems," *Proceedings of the National Academy of Sciences*, vol. 115, no. 7, pp. 1457–1462, 2018.

[16] C. C. McGeoch, "Adiabatic quantum computation and quantum annealing: Theory and practice," *Synthesis Lectures on Quantum Computing*, vol. 5, no. 2, pp. 1–93, 2014.

[17] S. Kasi and K. Jamieson, "Towards Quantum Belief Propagation for LDPC Decoding in Wireless Networks," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[18] Y. Susa, Y. Yamashiro, M. Yamamoto, and H. Nishimori, "Exponential speedup of quantum annealing by inhomogeneous driving of the transverse field," *Journal of the Physical Society of Japan*, vol. 87, no. 2, p. 023002, 2018.

[19] S. Mukherjee and B. K. Chakrabarti, "Multivariable optimization: Quantum annealing and computation," *The European Physical Journal Special Topics*, vol. 224, no. 1, pp. 17–24, 2015.

[20] L. Xiang, Y. Liu, Z. B. K. Egilmez, R. G. Maunder, L.-L. Yang, and L. Hanzo, "Soft list decoding of polar codes," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 921–13 926, 2020.

[21] J. Hagenauer, "The exit chart-introduction to extrinsic information transfer in iterative processing," in *2004 12th European Signal Processing Conference*. IEEE, 2004, pp. 1541–1548.

[22] L. O. Espluga, M. Aubault-Roudier, C. Poulliat, M. L. Boucheret, H. Al-Bitar, and P. Closas, "LLR approximation for fading channels using a bayesian approach," *IEEE Communications letters*, vol. 24, no. 6, pp. 1244–1248, 2020.

[23] G. Hosoya, M. Hasegawa, and H. Yashima, "LLR calculation for iterative decoding on fading channels using padé approximation," in *2012 International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6.

[24] J. M. Arrazola, A. Delgado, B. R. Bardhan, and S. Lloyd, "Quantum-inspired algorithms in practice," *arXiv preprint arXiv:1905.10415*, 2019.

[25] K. L. Pudenz, T. Albash, and D. A. Lidar, "Error-corrected quantum annealing with hundreds of qubits," *Nature communications*, vol. 5, no. 1, pp. 1–10, 2014.

[26] D-Wave, "D-Wave Problem Solving Handbook," Github, 2018.

[27] J. E. Dorband, "Extending the D-Wave with support for higher precision coefficients," *arXiv preprint:1807.05244*, 2018.

[28] D-Wave, "D-Wave *minorminer* Embedding Tool," Github, 2018.

[29] 3rd Generation Partnership Project (3GPP), "Multiplexing and channel coding," *38.212*, vol. V.15.3.0, 2018.

[30] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Physical Review X*, vol. 8, no. 3, p. 031016, 2018.

[31] Y. Tao, S.-G. Cho, and Z. Zhang, "A configurable successive-cancellation list polar decoder using split-tree architecture," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 2, pp. 612–623, 2020.

[32] A. D. King, S. Suzuki, J. Raymond, A. Zucca, T. Lanting, F. Altomare, A. J. Berkley, S. Ejtemaee, E. Hoskinson, S. Huang *et al.*, "Coherent quantum annealing in a programmable 2,000 qubit ising chain," *Nature Physics*, 2022.

[33] R. Wang and R. Liu, "A novel puncturing scheme for Polar codes," *IEEE Communications Letters*, 2014.

[34] R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of Polar codes," in *2011 IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 11–15.

[35] K. Niu, K. Chen, and J.-R. Lin, "Beyond Turbo codes: Rate-compatible punctured Polar codes," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 3423–3427.

[36] O. Dizdar and E. Arıkan, "A high-throughput energy-efficient implementation of successive cancellation decoder for Polar codes using combinational logic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2016.

[37] T. Lin, S. Cao, S. Zhang, S. Xu, and C. Zhang, "A reconfigurable decoder for standard-compatible LDPC codes and Polar codes," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2019, pp. 73–76.

[38] W. Xu, X. Tan, Y. Be'ery, Y.-L. Ueng, Y. Huang, X. You, and C. Zhang, "Deep learning-aided belief propagation decoder for Polar codes," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2020.

[39] N. Yang, S. Jing, A. Yu, X. Liang, Z. Zhang, X. You, and C. Zhang, "Reconfigurable decoder for LDPC and Polar codes," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[40] B. Tahir, S. Schwarz, and M. Rupp, "BER comparison between Convolutional, Turbo, LDPC, and Polar codes," in *24th International Conference on Telecommunications (ICT)*, 2017, pp. 1–7.

[41] A. Balatsoukas-Stimming, P. Giard, and A. Burg, "Comparison of Polar decoders with existing Low-density parity-check and Turbo decoders," in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2017.

[42] N. Ide, T. Asayama, H. Ueno, and M. Ohzeki, "Maximum likelihood channel decoding with quantum annealing machine," in *2020 International Symposium on Information Theory and Its Applications (ISITA)*, 2020, pp. 91–95.

[43] M. Kim, D. Venturelli, and K. Jamieson, "Leveraging Quantum Annealing for Large MIMO Processing in Centralized Radio Access Networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019.

[44] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy, "Discrete optimization using quantum annealing on sparse Ising models," *Frontiers in Physics*, vol. 2, p. 56, 2014.

[45] S. Kasi, J. Kaewell, and K. Jamieson, "The design and implementation of a hybrid classical-quantum annealing polar decoder," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 5819–5825.

[46] T. Matsumine, T. Koike-Akino, and Y. Wang, "Channel decoding with quantum approximate optimization algorithm," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 2574–2578.

[47] Z. Babar, Z. B. Kaykac Egilmez, L. Xiang, D. Chandra, R. G. Maunder, S. X. Ng, and L. Hanzo, "Polar codes and their quantum-domain counterparts," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 123–155, 2020.

[48] C.-Y. Lai, K.-Y. Kuo, and B.-J. Liao, "Syndrome decoding by quantum approximate optimization," *arXiv preprint arXiv:2207.05942*, 2022.