Fundamental bounds on the interconnect complexity of decoder implementations

Pulkit Grover and Anant Sahai
{pulkit, sahai}@eecs.berkeley.edu

***Abstract*— Modern codes are often designed to attain the minimum possible error probability under a blocklength constraint. For instance, sparse-graph codes are often designed aiming for large girth in order to reduce the error-probability. In this paper, we show that such an improved performance comes at a fundamental cost: longer interconnects (wires) in the decoding circuit. Recent empirical results show why shorter interconnects are important: decoders with short interconnects can have significantly smaller power consumption because significant power is burned in wires, and this power is proportional to the length of the wire.**

We derive two bounds to demonstrate this cost: the first bound shows that for a belief-propagation decoder for a linear sparse-graph code, the wire-length must increase exponentially in the girth of the code. While this bound depends on the code construction, our second bound is fundamental: we derive lower bounds on the wire-length for decoding *any* code (even if it is nonlinear) and *any* message-passing decoding algorithm given the performance and the number of clock-cycles at the decoder. Under simplifying assumptions, we discuss novel small-girth code constructions that provide upper-bounds on the required interconnect-length.

I. Introduction

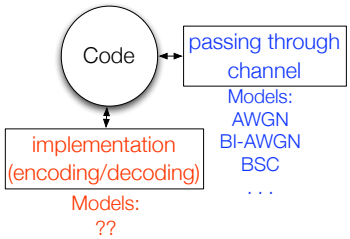

Fig. 1. A code's interfaces with the physical world. We need models for implementation just as we have models for channels.

Fig. 1 illustrates the disconnect that exists today between the theory community that designs codes, and the experimental community that implements them. A code interfaces to the physical world not only when it passes through a channel, but also when it is encoded and decoded. The theory community focuses almost exclusively on the first interface by designing codes that minimize the channel input power for given constraints on the channel bandwidth. The purpose of this paper is to show that the code design fundamentally affects the power consumption, complexity, and the area of implementation. Further, just as there are fundamental limits to the requirements on bandwidth and power of the channel that depend on the rate and error probability [1], there are fundamental limits on power, complexity, and area of implementation that also depend on the rate and error probability. Code design and implementation thus should not be done in isolation.

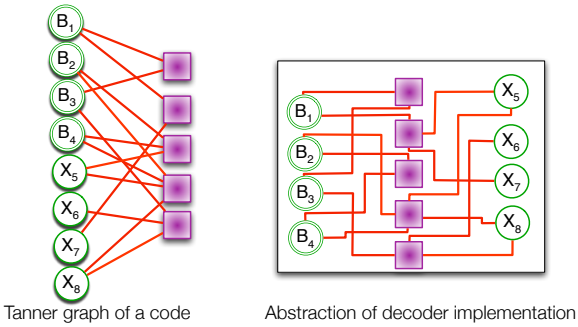

Fig. 2. Tanner graph of an LDPC code and a model of its decoder's implementation.

The focus of this paper is on the wiring complexity of the decoder implementation. Where are these wires on a decoder? Fig. 2 shows an abstraction for a possible decoder implementation of an LDPC code. In this example, the wires connect the variable nodes to the check nodes along the edges in the Tanner graph of the code. The variable nodes are placed on the two sides of the check nodes in order to minimize wiring complexity. Why do wires matter? Typically, longer wires consume more power, area, and are more complex to route than shorter wires (e.g. [2] and the references therein).

In Section III, we first show (Theorem 1) that for a fully-parallel implementation of belief-propagation decoding for any sparse-graph code, the length of the longest interconnect must increase exponentially in the girth of the code. While this suggests that the implementation complexity may depend on the desired performance, this result is extremely limited. Large girth is a property that is merely sufficient, but hardly necessary[1], for good code performance. The decoding architecture may also not be fully parallel, and there is no reason to believe that the decoding algorithm would always be belief-propagation either. Often, nodes do double-duty by acting as different nodes in the Tanner graph in different clock-cycles [4], which also has the potential to reduce interconnect-lengths. Can one therefore sacrifice on large girth and fully parallel decoding to have codes/decoders with reduced interconnect-lengths and yet good performance? Our second bound (Theorem 2) rules out this possibility as well:

[1]For instance, in [3] the authors mention that there is no proof that cycles in the decoding graph only degrade the performance.

we show that for *any* code (which may not even be a sparse-graph code) and any decoding algorithm, the product of the length of the longest interconnect at the decoder and the number of clock-cycles must diverge to infinity as the error probability converges to zero or as the rate approaches capacity.

What does this imply for code design? First, it changes the code design problem from merely attaining the desired performance (*i.e.*, rate and error probability) to introducing the goal of minimizing wire-lengths while attaining the performance objectives. Second, it shows how overdesigning can hurt. It is well known that the required blocklength, and hence the required area of the decoding chip, increases with improved performance. Even so, sparse-graph codes are often designed aiming for large girth in order to reduce the error-probability. This is because the on-chip area depends primarily on the blocklength, and the available area might be large enough that a code of larger girth can be accommodated. However, even when enough area is available, the code/decoder should be designed for the required performance, and no better, because an improved performance will require larger wire-lengths.

Interconnect complexity has been acknowledged as a problem in implementing decoders. Whereas earlier literature focused on reducing interconnect complexity for given LDPC codes (e.g. [5]–[7]), the works of Mansour and Shanbhag [8] and Thorpe [9] were the first to propose designing codes to reduce interconnect complexity. While Mansour and Shanbhag use quasi-cyclic LDPC constructions (first proposed by Gallager [10, Appendix C]), Thorpe proposes construction of protograph-based LDPC codes. These codes are nicely structured so that wiring is easy. While these codes have been used extensively in standards (such as 10Gbase-T [4]), there has been little exploration of the goodness of the interconnect length for their decoder implementations. This is where our bounds can help.

How far are our bounds from what is achievable? Because the constructions of [8], [9] are algorithmic, it is hard to obtain upper bounds on attainable wire-lengths for them. Constructing codes with small interconnect-lengths in general is an intriguing graph-theoretic problem. For the time-being, we side-step the real problem of minimizing interconnect-lengths by approximating interconnect-lengths with the blocklength of the code. Using this simplification, in Section IV we provide novel constructions of arbitrary girth. The resulting upper bound on wire-length increases exponentially in the girth, though the exponent is larger than that for the lower bound. The upper bound also serves to bring out one of the deficiencies of our model: because the wiring complexity introduced by intersecting interconnects is not accounted for, the bounds may be unrealistically optimistic. Perhaps results from existing VLSI theory (e.g. [11]) or the theory of embedding graphs into Euclidean spaces [12, Ch. 13] can help rectify this problem.

II. Notation, definitions, and system model

Notation: A (d_v, d_c)-LDPC code is a regular LDPC code [3], [13] where the degree of each variable node is d_v and the degree of each check node is d_c. The decoded average bit-error probability is denoted by $\langle P_e \rangle$. The length of the longest interconnect is denoted by W_{max}. The code rate is denoted by R, and the channel capacity by C. The girth of a graph is defined as the length of the shortest cycle in the graph. A regular graph (*i.e.*, a graph where all nodes have the same degree) is denoted by $G^{reg}(r, g)$, where r is the degree of each node. A regular bi-partite graph is denoted by $G^{bi}(d_v, d_c, g)$ where d_v is the left-node degree, and d_c is the right-node degree. The *order* of a graph is the number of nodes in the graph.

Channel model: The channel is assumed to be a memoryless Binary Symmetric Channel with channel flip-probability p, denoted by BSC(p). A test channel BSC(g) with flip-probability $g > p$ is used in our lower bounds.

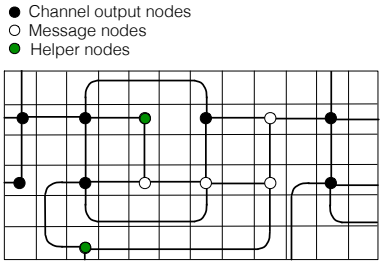

Fig. 3. The VLSI model of a decoder [14] (this model is based on Thompson's VLSI model of computation [15], [16]).

Decoding implementation model: As shown in Fig. 3, the decoder is modeled by a collection of computational nodes that are connected using interconnects [14]. The nodes are either (a) 'message' nodes that store the decoded bits after decoding, (b) 'channel output' nodes that store the channel outputs, (c) 'helper' nodes that act as intermediaries of processing by improving connectivity, or (d) any combination of (a), (b), and (c). In [14], we used limitations on wiring density to assume limited connectivity of each computational node. Here we do not make that assumption. Instead, the wiring complexity is reflected directly in the interconnect-length.

For convenience, we assume that all computational nodes are circular[2] and occupy an area of at least A_{node} units. This area depends on the complexity of the message-passing decoding algorithm (for instance, the size of each message), the technology used (for instance, 65 nm CMOS), and even the length of the interconnects (as suggested above, the size of the driving transistors must increase with increase in interconnect length). For simplicity, in this paper we assume that this area is fixed. For our fundamental bound (Theorem 2), we assume that A_{node} is merely the area occupied by a memory-cell (*i.e.*, a channel output node). Yet we assume that these memory cells are connected using interconnects as illustrated in Fig. 3.

Modeling length of an interconnect: The length of the interconnect joining two computational nodes is defined as the distance between the centers of the two nodes (see Fig. 4).

[2]Any other model can be used in its place with a minor tweaking of our results.

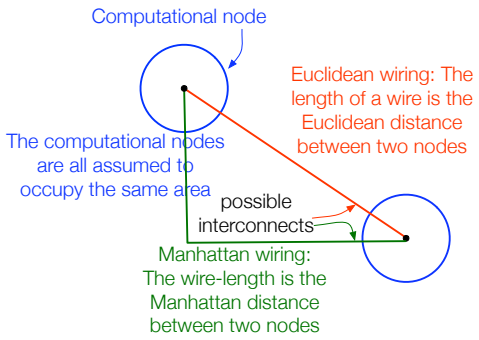

Fig. 4. Our model for an interconnect connecting two computational nodes. The interconnect-length on an actual implementation is always larger, usually because the wiring is Manhattan wiring. For more realistic upper bound calculations, we use Manhattan wiring in the upper bound. For broader applicability, the lower bound is allowed to have the any wiring.

The example in Fig. 3 shows that the actual length of the interconnect is always longer. This can happen due to many reasons: the wiring is usually performed in a "Manhattan wiring" pattern[3] (*i.e.*, the interconnects are aligned in a grid), the interconnects that cross each other have to cross by moving into different metal layers (thereby increasing their length), interconnects cannot cross nodes, etc. To better approximate the interconnect length in the upper bound, we will use the Manhattan distance in order to obtain those bounds.

III. Lower bounds on the interconnect-length

A. Lower bounds on the interconnect-length given the girth of a regular LDPC code

If a sparse-graph code of girth g is decoded using belief-propagation decoding, then every decoding neighborhood is a tree for at least $\frac{g}{2}-1$ iterations. The belief-propagation decoding algorithm is equivalent to the optimal (maximum-likelihood) decoding algorithm on a decoding neighborhood that is a tree[4]. The following theorem shows that an increased girth must come at the cost of an increased interconnect length.

Theorem 1: For decoding a regular (d_v, d_c)-LDPC code of girth g using belief-propagation decoding in a fully-parallel implementation, a lower bound on the length of the longest interconnect for a two-dimensional chip[5] is given by:

$$W_{max}(d_v,d_c,g) \geq \frac{\sqrt{A_{node}}}{\sqrt{\pi}(\frac{g}{2}-1)}\left(\sqrt{1+\psi(d_v,d_c,g)}-1\right) \quad (1)$$

$$\text{where } \psi(d_v,d_c,g) := \sum_{i=1}^{\frac{g}{2}-1} d_v(d_v-1)^{\lfloor \frac{i}{2} \rfloor}(d_c-1)^{\lfloor \frac{i-1}{2} \rfloor}. \quad (2)$$

The bound is also a lower bound on the longest interconnect involved in each decoding neighborhood.

Proof: See Appendix I. ∎

[3]With Manhattan wiring, the interconnect-length is better approximated by L_1 distance between two points.

[4]Even so, the maximum-likelihood (ML) decoding algorithm over the entire block can outperform BP decoding over trees. This is because the ML decoding algorithm performs optimal decoding over the entire block, not just the decoding neighborhood. The question of when the performance of ML decoding and BP decoding can be made the same was addressed recently in [17].

[5]The bound can easily be generalized to three dimensions.

B. Fundamental lower bounds on the interconnect-length given the code performance for any code

In this section, we provide bounds that are valid for any code (even nonlinear), decoding architecture (even a not-fully-parallel implementation), and any message-passing decoding algorithm. In order to derive bounds for implementations that are not fully parallel, the area A_{node} is assumed to be area of just the memory cells that store the channel outputs (and hence potentially much smaller than the area of the computational nodes in Theorem 1). The bounds derived here use the observation that each channel output needs to be stored in a different memory-cell.

The bounds are stated as lower bounds on the product of the length of the longest interconnect W_{max} and the number of clock-cycles t for which the decoding is run. Notice that the decoding power increases if any one of these two quantities is increased. The designer therefore needs to reduce both of them in order to minimize the power consumption. The following theorem shows that the two cannot be simultaneously small. The bound is really a bound on the required communication complexity [18] for decoding to a certain error-probability for a given rate.

Theorem 2: For a given BSC(p) channel and t time-steps of the message-passing decoding algorithm, the following bound holds on W_{max}, the length of the longest interconnect:

$$\pi\left(tW_{max}+\sqrt{\frac{A_{node}}{\pi}}\right)^2 \geq n(\langle P_e\rangle, R)A_{node}, \quad (3)$$

where $n(\langle P_e\rangle, R)$ is lower bounded by the following relation:

$$\langle P_e\rangle \geq \sup_{C_{bsc}^{-1}(R)<g\leq\frac{1}{2}} \frac{h_b^{-1}(\delta_{bsc}(g))}{2} 2^{-nD(g\|p)}\left(\frac{p(1-g)}{g(1-p)}\right)^{\epsilon\sqrt{n}}, \quad (4)$$

where $h_b(\cdot)$ is the binary entropy function; $D(g\|p) = g\log_2\left(\frac{g}{p}\right)+(1-g)\log_2\left(\frac{1-g}{1-p}\right)$ is the KL-divergence; and $\delta_{bsc}(g) = 1-\frac{C_{bsc}(g)}{R_{ch}}$, where $C_{bsc}(g) = 1-h_b(g)$; and $\epsilon = \sqrt{\frac{1}{K(g)}\log_2\left(\frac{2}{h_b^{-1}(\delta_{bsc}(g))}\right)}$, where $K(g) = \inf_{0<\eta<1-g}\frac{D(g+\eta\|g)}{\eta^2}$.

Proof: Fig. 5 shows the area covered using a

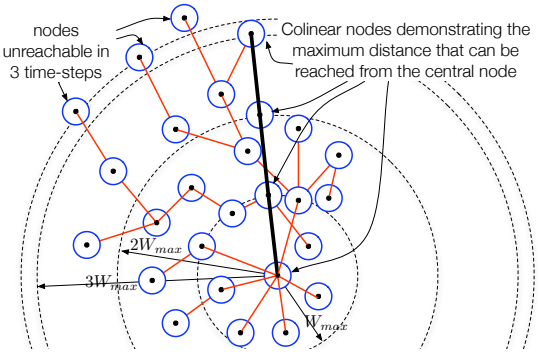

Fig. 5. An illustration of the bounding argument used in the proof of Theorem 1 and Theorem 2. The total area covered using interconnects is at most $\pi\left(tW_{max}+\sqrt{\frac{A_{node}}{\pi}}\right)^2$.

maximum interconnect-length of W_{max} in three time-steps. In general, the area covered is lower bounded by $\pi\left(tW_{max}+\sqrt{\frac{A_{node}}{\pi}}\right)^2$. This area should exceed the area covered by just the memory-cells, which is the product of the neighborhood size ($n(\langle P_e\rangle, R)$) and the area of each memory-cell (A_{node}). This yields (3). Equation (4) is taken from Theorem 1 of [19]. A lower bound on $n(\langle P_e\rangle, R)$ can be obtained by taking logarithms on both sides of (4) and solving the resulting quadratic equation in n. The best such lower bound can be found by optimizing over the allowed values of g. ■

Corollary 1: $\pi(tW_{max})^2 \gtrsim A_{node}\Omega\left(\frac{\log_2\left(\frac{1}{\langle P_e\rangle}\right)}{(C-R)^2}\right)$.

Proof: Follows directly from an approximation introduced in [19]. ■

Although the bound is provided for the BSC, similar bounds can be derived for the AWGN channel, the binary erasure channel, the Binary-Input AWGN channel, etc [19].

IV. Upper bounds on the required interconnect length

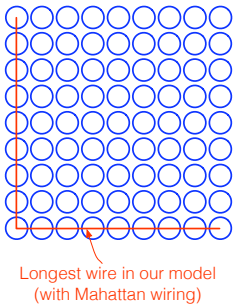

Fig. 6. Our upper bound on the interconnect-length arranges all the nodes in a grid configuration, and connects them using the shortest interconnect connecting two nodes (under Manhattan wiring). The longest interconnect in this configuration is one that connects the farthest nodes. If the number of nodes is not a perfect square, we take the smallest square larger than the number of nodes.

In this section we provide an extremely simple approach to construct LDPC codes to provide upper bounds on the required interconnect-length in our model of the decoder implementation. Obtaining complexity of wiring is in general a hard problem [11]. Instead, we obtain bounds on interconnect length within our simplified model (see Fig. 6).

We first provide an algorithm to obtain a bi-regular bi-partite graph $G^{bi}(d_v, d_c, g_{code})$ of girth g_{code} from a regular "skeleton" graph $G^{reg}(r, g_{skeleton})$ of n nodes and girth $g_{skeleton}$ with $r = d_v d_c$. The term "skeleton" is used to describe G^{reg} because the graph only provides the basic structure of the final bipartite graph G^{bi}. We will then show that the girth of the resulting bi-partite graph $G^{bi}(d_v, d_c, g_{code})$ is at least as much as the girth of the skeleton graph $G^{reg}(r, g_{skeleton})$.

We call each node in the skeleton graph G^{reg} a "super-node." A "cell-graph" is a (d_v, d_c)-regular bi-partite graph with $d_v + d_c$ nodes (*i.e.*, the minimum possible nodes for a (d_v, d_c) regular graph). Each super-node is embedded (see Fig. 7) with a cell-graph. This embedding can be thought of as mere placement of a cell-graph inside each supernode of the skeleton graph. On this structure, we run the following pseudocode:

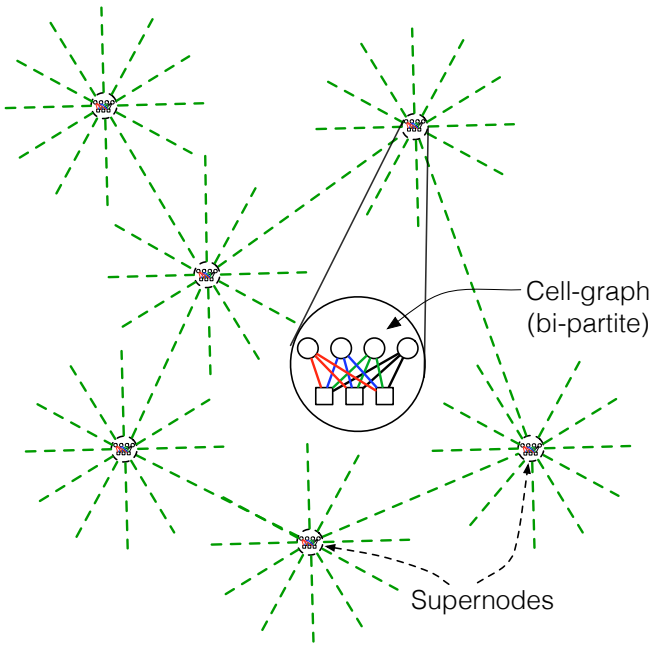

Fig. 7. The skeleton graph. In our construction, the skeleton graph is a regular graph of girth $g_{skeleton}$. Each node in this graph is embedded with a cell-graph (shown inset). To construct the code, each of the cell-graphs is linked via an edge-exchange procedure shown in Fig. 8.

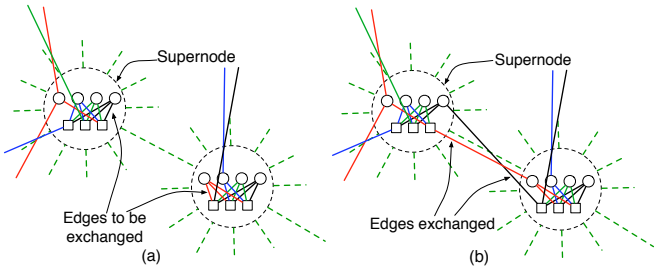

Fig. 8. The figure demonstrates how the edge exchange process is carried out. (a) Two super nodes that are neighbors, and have not yet exchanged an edge, are chosen. The indicated edges in the (remnant) cell graphs are chosen for the exchange (any choice of edges will work). (b) The indicated edges are exchanged — the variable node of one is connected to the check node of the other.

1) Initiate $i = 1$.
2) While $i < \frac{nd_v d_c}{2}$, do steps 3-5.
3) Pick two neighboring supernodes that have unconnected cell-graphs.
4) Pick one edge each from the two cell-graphs that have not been exchanged. Exchange these edges, *i.e.*, connect the variable node from one supernode to the check node in the other supernode (for both pair of nodes).
5) i=i+1.

Claim 1: When the algorithm finishes, the resulting graph is a bi-regular bipartite graph that has graph at least $g_{code} = g_{skeleton} + 1$ if $g_{skeleton}$ is odd, and $g_{code} = g_{skeleton}$ if $g_{skeleton}$ is even.

Proof: The graph is bi-regular bipartite at each stage in the process. The girth can be smaller than $g_{skeleton}$ only if some super-node has any remaining internal edges when the algorithm finishes. If that is the case, it has not exchanged an edge with at least one neighboring super-node. Pick one such neighboring super-node. Since the skeleton graph is regular, this would mean that this neighboring super-node also has at least one internal edge remaining. Thus the algorithm has not finished, and we arrive at a contradiction.

Because there are no internal edges, the girth of the resulting bipartite graph is at least $g_{skeleton}$, the girth of the skeleton graph. Further, if $g_{skeleton}$ is odd, the the girth of the bipartite graph is at least $g_{skeleton}+1$, because a bipartite graph can only have even girth. This completes the proof. ■

Claim 2: The number of nodes in the code-construction above is upper bounded by

$$v(d_v, d_c, g_{code}) \leq 2(d_v + d_c)d_v d_c q^{\frac{3}{4}g_{code}-a}, \qquad (5)$$

where $a = 4, 11/4, 7/2, 13/4$ for $g_{code} = 0, 1, 2, 3 \bmod 4$ respectively, and $q < 2d_v d_c + 1$ is the smallest prime power larger than $d_v d_c$.

Proof: We use the following theorem from [20].

Theorem 3 (Theorem A in [20]): Let $r \geq 2$ and let $g \geq 5$, then the following upper bound holds on the order of the smallest regular graph of degree r and girth g:

$$v(r, g) \leq 2rq^{\frac{3}{4}g-a} \qquad (6)$$

where $a = 4, 11/4, 7/2, 13/4$ for $g = 0, 1, 2, 3 \bmod 4$ respectively, and $q < 2r + 1$ is the smallest prime power larger than r.

Using $r = d_v d_c$ in Theorem 3, we get a bound of $2d_v d_c q^{\frac{3}{4}g_{code}-a}$ on the number of super-nodes in our bi-partite graph. Since each super-node contains $d_v + d_c$ nodes, we get the bound of Claim 2. ■

Claim 3: In our model of decoder implementation, an upper bound on the interconnect-length for the decoding implementation (using Manhattan wiring) of the above code construction is given by:

$$W_{max} \leq 4\left(\sqrt{2(d_v + d_c)d_v d_c q^{\frac{3}{4}g_{code}-a}} + 1\right)\sqrt{\frac{A_{node}}{\pi}}, \qquad (7)$$

where $q \leq 2d_v d_c + 1$.

Proof: The proof follows immediately from the observation that the nodes can be arranged in the grid arrangement of Fig. 6. The technical detail where the number of nodes is not a perfect square can be dealt with easily. If N is the total number of nodes, a square larger than N is smaller than $(\sqrt{N}+1)^2$. Each side of the grid therefore has at most $\sqrt{N}+1$ nodes. ■

Comparing with the lower bounds of Theorem 1, W_{max} for our construction increases exponentially in girth with an exponent of $\frac{3}{8}g_{code}$. On the other hand, the lower bound has an exponent of $\frac{1}{4}g_{code}$.

Claim 4: An upper bound on the wire-length as a function of the achieved error-probability is given by:

$$W_{max} \leq 2\sqrt{2}\left(\delta\left(\ln\left(\frac{1}{\langle P_e\rangle}\right)\right)^{\eta} + 1\right)\sqrt{\frac{A_{node}}{\pi}}, \qquad (8)$$

where δ and η are positive constants that depend on the chosen degrees d_v, d_c of the code (as long as $d_v > 2$) and the chosen message-passing decoding algorithm.

Proof: It is shown in [21] that if the density-evolution analysis is valid for an LDPC code, the bit-error probability of decays double-exponentially with the number of iterations, I, *i.e.*, $\langle P_e\rangle \leq \exp(-\alpha e^{\gamma I})$ for some positive α and γ. The density-evolution analysis is exact for regular LDPC codes as long as all the decoding neighborhoods are trees[6] [3, Section III] [22], which is satisfied when the girth of the code-graph exceeds twice the number of iterations. Thus, $\langle P_e\rangle \leq \exp(-\alpha e^{\gamma g_{code}})$. The required girth is therefore upper bounded by:

$$g_{code} \leq \frac{1}{\gamma}\ln\left(\frac{1}{\alpha}\ln\left(\frac{1}{\langle P_e\rangle}\right)\right). \qquad (9)$$

Substituting in (7), we establish the claim. ■

Observe that the obtained product of the number of iterations $(\frac{g}{2} - 1)$ and the wire-length W_{max} is bounded above by $O\left(\left(\ln\left(\frac{1}{\langle P_e\rangle}\right)\right)^{\zeta}\right)$ for some $\zeta > 0$ which is similar to the behavior of the lower bound in Theorem 2. However, because regular LDPC constructions are bounded away from capacity, these codes operate at a farther gap from capacity than that predicted by Theorem 2, thus incurring a cost in transmit power.

These upper bounds are unrealistic in part because we assume that there is no increase in length due to interconnects crossing interconnects or interconnects crossing nodes. As we noted in Section II, this is never the case in reality. How far are more realistic upper bounds? It is hard to say without estimating the wiring-complexity using a more realistic model, perhaps closer to the model explored in [11]. A question still remains: what is the best way to embed these large graphs (see, e.g., [23]) in a smaller dimensional space?

General constructions of bi-partite graphs of any specified girth have appeared previously in [24]. However, our bounds on the required number of nodes (Claim 2) are tighter (than those in [24, Theorem E]) and have a further advantage that any improvement in design of regular graphs directly yields improvements in design of bi-partite graphs. Constructions of regular graphs are better studied than bi-partite constructions [25] and therefore might offer good bounds for specific girths as well.

V. Discussions and conclusions

Based on ideas in this paper, how should we design code/decoder in order to minimize the power consumption in order to attain a specified performance? In our earlier work [14], [19], we noted that an increase in transmit power can reduce the decoding power by reducing the number of iterations required at the decoder. Here, we show that if we are indeed operating for a smaller number of iterations, the energy *per-iteration* can also be lowered because we can

[6] An ensemble of codes is considered in [3] only in order to arrive at code constructions that have large girth, and to show that almost all code constructions of given degrees behave nearly the same. However, if the decoding neighborhood has no cycles and the code is regular, then the prediction of density-evolution is exact (as also noted in [22]).

use code constructions that reduce interconnect lengths. In a good design, the designer will optimize over the interconnect length, the number of iterations, and the transmit power in order to minimize the system-level power consumption.

ACKNOWLEDGMENTS

We thank Karthik Ganesan, Jan Rabaey and Salim El Rouayheb for helpful discussions, and Vineet Abhishek for a careful read of the paper. Support of NSF grant CCF-0917212 is also acknowledged.

APPENDIX I
LOWER BOUND BASED ON CODE-GIRTH

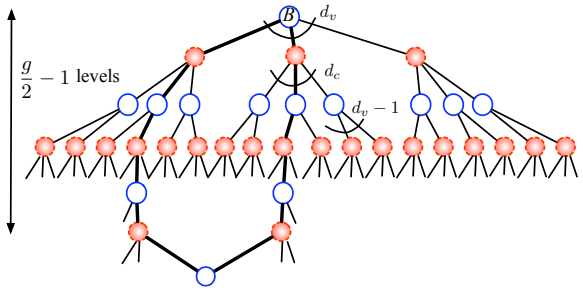

Fig. 9. The decoding neighborhood of a bit-node for a regular $(3,4)$-LDPC code. The check nodes are shaded. The girth of the code is 12. Two paths diverging from the root node converge at a node at $\frac{g}{2}=6$th level. Because there are no cycles until level $\frac{g}{2}-1$, the number of nodes in the neighborhood at $\frac{g}{2}-1$ time steps is the maximum possible for the given left and right degrees.

Because the graph is bipartite, g is even. Choose a node that participates in one of the cycles of length g. Call it node B. Expand the graph with node B as the root node until level $\frac{g}{2}$. At level $\frac{g}{2}$, there will be at least one node that is connected to at least two nodes at level $\frac{g}{2}-1$. But the sub-graph until level $\frac{g}{2}-1$ is a tree. The number of nodes until level $\frac{g}{2}-1$ is lower bounded by:

$$\psi(d_v,d_c,g) := \sum_{i=1}^{\frac{g}{2}-1} d_v(d_v-1)^{\lfloor \frac{i}{2} \rfloor}(d_c-1)^{\lfloor \frac{i-1}{2} \rfloor}. \quad (10)$$

Thus the neighborhood of node B explodes exponentially in the code girth. What about the neighborhood area-on-chip? If the longest interconnect has length W_{max}, the maximum area that the neighborhood can cover is given by the area of the circle: πr^2, where $r=(\frac{g}{2}-1)W_{max}+\sqrt{\frac{A_{node}}{\pi}}$ (see Fig. 5).

Thus the area reachable by the interconnects is upper bounded by:

$$\pi\left(\left(\frac{g}{2}-1\right)W_{max}+\sqrt{\frac{A_{node}}{\pi}}\right)^2. \quad (11)$$

Using (10) and (11),

$$\begin{aligned}&\pi\left(\left(\tfrac{g}{2}-1\right)W_{max}+\sqrt{\tfrac{A_{node}}{\pi}}\right)^2\\ &\geq A_{node}\left(1+\textstyle\sum_{i=1}^{\frac{g}{2}-1} d_v(d_v-1)^{\lfloor \frac{i}{2} \rfloor}(d_c-1)^{\lfloor \frac{i-1}{2} \rfloor}\right)\\ &= A_{node}(1+\psi(d_v,d_c,g)).\end{aligned}$$

Thus, $W_{max} \geq \frac{\sqrt{A_{node}}}{\sqrt{\pi}(\frac{g}{2}-1)}\left(\sqrt{1+\psi(d_v,d_c,g)}-1\right)$. This completes our proof.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. Jour.*, vol. 27, pp. 379–423, 623–656, Jul./Oct. 1948.
[2] K. Ganesan, P. Grover, and J. M. Rabaey, "Interconnect power consumption can be significant in simplistic decoders," Feb. 2011, in preparation.
[3] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[4] Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "An efficient 10GBASE-T ethernet LDPC decoder design with low error floors," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 843 –855, Apr. 2010.
[5] X. Hu, E. Eleftheriou, D. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *IEEE Global Telecommunications Conference (Globecom)*, vol. 2, 2001, p. 1036.
[6] C. Howland and A. Blanksby, "Parallel decoding architectures for low density parity check codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 2001, pp. 742–745.
[7] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *IEEE Global Telecommunications Conference (Globecom)*, vol. 5, 2001, pp. 3019–3024.
[8] M. Mansour and N. Shanbhag, "High-throughput LDPC decoders," *IEEE Tran. Very Large Scale Integration Systems*, vol. 11, pp. 976–996, 2003.
[9] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *IPN Progress Report 42-154, JPL*, 2005.
[10] R. Gallager, "Low-Density Parity-Check Codes," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1960.
[11] F. Leighton, "A layout strategy for VLSI which is provably good," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. ACM, 1982, pp. 85–98.
[12] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc*, vol. 43, pp. 439–561, 2006.
[13] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2007.
[14] P. Grover, K. Woyach, and A. Sahai, "Towards a communication-theoretic understanding of system-level power consumption," *Arxiv preprint arXiv:1010.4855, submitted to IEEE Journal on Selected Areas in Communication*, 2010.
[15] C. D. Thompson, "Area-time complexity for VLSI," in *Proceedings of the 11th annual ACM symposium on Theory of computing (STOC)*. New York, NY, USA: ACM, 1979, pp. 81–88.
[16] ——, "A complexity theory for VLSI," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1980.
[17] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC," in *IEEE International Symposium on Information Theory (ISIT)*, 2010, pp. 684–688.
[18] E. Kushilevitz and N. Nisan, "Communication complexity," *Advances in Computers*, vol. 44, pp. 331–360, 1997.
[19] A. Sahai and P. Grover, "The price of certainty : "waterslide curves" and the gap to capacity," Dec. 2007. [Online]. Available: http://arXiv.org/abs/0801.0352v1
[20] F. Lazebnik, V. Ustimenko, and A. Woldar, "New upper bounds on the order of cages," *The electronic journal of combinatorics*, vol. 4, no. 2, 1997.
[21] M. Lentmaier, D. V. Truhachev, K. S. Zigangirov, and D. J. Costello, "An analysis of the block error probability performance of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3834–3855, Nov. 2005.
[22] A. Shokrollahi, "LDPC codes: An introduction," *Coding, cryptography and combinatorics*, p. 85, 2004.
[23] J. Matoušek, "On embedding expanders into L_p spaces," *Israel Journal of Mathematics*, vol. 102, no. 1, pp. 189–197, 1997.
[24] Z. Furedi, F. Lazebnik, A. Seress, V. Ustimenko, and A. Woldar, "Graphs of prescribed girth and bi-degree," *Journal of Combinatorial Theory, Series B*, vol. 64, no. 2, pp. 228–239, 1995.
[25] G. Exoo and R. Jajcay, "Dynamic cage survey," *Electron. J. Combin*, vol. 15, 2008.