Making Classical Ground State Spin Computing Fault-Tolerant

I. Crosson,[1] D. Bacon,[2,1] and K. R. Brown[3]

[1] *Department of Physics, University of Washington, Seattle, WA 98195 USA*
[2] *Department of Computer Science & Engineering, University of Washington, Seattle, WA 98195 USA*
[3] *Schools of Chemistry and Biochemistry; Computational Science and Engineering; and Physics, Georgia Institute of Technology, Atlanta, GA 30332 USA*

(Dated: June 24, 2010)

We examine a model of classical deterministic computing in which the ground state of the classical system is a spatial history of the computation. This model is relevant to quantum dot cellular automata as well as to recent universal adiabatic quantum computing constructions. In its most primitive form, systems constructed in this model cannot compute in an error free manner when working at non-zero temperature. However, by exploiting a mapping between the partition function for this model and probabilistic classical circuits we are able to show that it is possible to make this model effectively error free. We achieve this by using techniques in fault-tolerant classical computing and the result is that the system can compute effectively error free if the temperature is below a critical temperature. We further link this model to computational complexity and show that a certain problem concerning finite temperature classical spin systems is complete for the complexity class Merlin-Arthur. This provides an interesting connection between the physical behavior of certain many-body spin systems and computational complexity.

PACS numbers: 05.20.-y,05.10.-a,05.70.Fh,89.70.Eg

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
24 JUN 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Making Classical Ground State Spin Computing Fault-Tolerant	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Physics,University of Washington,Seattle,WA,98195**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**24**	

The creation of massive digital and deterministic computing systems represents one of the greatest triumphs of the last century. Increasingly, however, as the components in our information processing devices shrink to atomic scales [1], the two defining characteristics of these systems—that they are digital as opposed to analog, and that they are deterministic as opposed to probabilistic—are beginning to be revealed as approximations that arise from considering macroscopic or mesoscopic physical systems. As computing elements are made smaller they are increasingly subject to noise and an inability to be perfectly controlled [2]. Even further down this route lie quantum computers [3–7] where, instead of probabilistic time evolution, the computer is made of elements which function according to the laws of quantum information. These trends point toward the necessity of studying computing elements in settings where digital and deterministic functionality are not a priori guaranteed [8]. Here we consider a physical method for classical computing wherein a computation is encoded into the ground state of a classical many-body system of spins. At zero temperature in this model, computation is deterministic and thus we label this model classical *ground state spin computation*. The model has many predecessors, include quantum-dot cellular automata [9–11], the broadcast model on trees [12], quantum ground state computing [13, 14], and universal adiabatic quantum computing [15–19]. We focus our work here on this model when the system has thermalized at a non-zero temperature. At non-zero temperature, if we do nothing to compensate for the effects of thermal equilibrium, ground state spin computation fails. Here we show that it is possible to use the ideas of fault-tolerant classical computing to design ground state spin computers which function deterministically with high probability at non-zero temperatures below a critical temperature. We achieve this by mapping the thermal ensembles of our computational spin systems onto the distribution of ensembles produced during classical probabilistic computations. This is the main result of the paper: some classical spin systems in thermal equilibrium can be thought of as enacting classical probabilistic computations spatially across the system.

Finally, our mapping leads naturally to problems which are complete for the complexity class promise Merlin-Arthur [20, 21], a result which can be regarded as a finite temperature version of the Cook-Levin theorem [22, 23] from the theory of computational complexity. The new computational model we consider thus serves as a bridge between statistical mechanics and the classical computational complexity of probabilistic computation. We also obtain a computational complexity result concerning the difficulty of identifying when a system admits a mapping to a probabilistic circuit by showing that a restricted version of this problem is NP-complete. This implies that the problem of identifying when a physical system can be seen to be performing a computation is itself computationally intractable.

The outline of our paper is as follows. In Section I we introduce the model of computing with spins in the ground state using a particular energy function for this system. In Section II we show that at non-zero temperature the model from the previous section fails to properly compute. We do this for an extremely simple and previously explored model, but present the result using two different methods, one related to reinterpreting the transfer matrix and the other related to classical circuits applied to simple thermal ensembles. Generalizing these methods allows us to discuss the energy functions arising in Section I as probabilistic circuits. This more general formulation we discuss using two different methods in Section III. Motivated by the mappings discovered in Section III, we then return to our original model and define the broader class of energy functions consistent with these models in Section V. At this point we also point out how our models differ from classical models of quantum-dot cellular automata. The more general setting leads us to ask questions about how algorithmically hard it is to decide if a given physical system supports computation in these models. We show that this problem, even in a very weak form, is NP-complete and thus likely to be intractable. In Section VI we discuss how to design ground state spin models that, unlike the generic case, do compute fault-tolerantly at non-zero temperature. Our construction is intimately related to the original fault-tolerance construction of von Neumann [8] and we show that von Neumann's threshold for fault-tolerance is related to a critical temperature in our system. Finally in Section VII we discuss how the model we consider is related to the Cook-Levin theorem from computational complexity and show how this leads to certain natural problems about our model being **Promise-**MA-complete.

I. CLASSICAL GROUND STATE SPIN COMPUTING

Here we introduce the model of classical ground state spin computing. This model, in particular restricted cases, is relevant to a variety of different models considered elsewhere. For instance, quantum dot cellular automata can be partially modeled by classical ground state spin computing [9, 10]. However the model we consider is more general than these specific instances. This larger generality comes along with certain unphysical assumptions: our models contain, for example, three-body interactions which are not necessarily easily achievable in a physical device. Physically we imagine that models such as the one we consider might emerge in certain limits where effective many-body interactions can emerge. However, irrespective of the physical implementation, the model provides a set of classical many-body interacting systems which can be mapped onto probabilistic classical circuits. This connects statistical mechanics with classical probabilistic circuits thus bridging computer science and physics in a new and interesting manner.

We further note that a quantum version similar to this model has been studied by Mizel *et al* [13, 14]. This later model contains significant difference owing to the quantum nature of the computation: in particular the ground state does not contain the computation laid out spatially, but instead exists in a superposition of the computation being carried out spatially. These quantum models are also connected to universal adiabatic quantum computing schemes [15–19] and piecewise variations on these schemes [24–27]. By studying the classical analogy of these schemes, we hope to shed light on how these later quantum methods can be made fault-tolerant.

A. The Model

A *combinatorial circuit*, $C = (G, L)$, is a directed acyclic graph, $G = (V, E)$ with vertices V and edge set E, where the internal vertices, V_{in}, of the graph are logical gates, $L : V_{in} \to \mathcal{G}$ ($\mathcal{G}$ is the set of all logical gates), and the external vertices, V_{ex}, are the inputs and outputs to the circuit. External vertices that have no edges leading to them are inputs and external vertices that have no edges that lead away from them our outputs to the circuit. If a circuit has n input vertices and m output vertices, then to such a circuit we can assign a function $f : \{0,1\}^n \to \{0,1\}^m$ which results from propagating the input through the logic gates to the output vertices. Note that at this point we require that fan-outs in our circuit are represented by gates and do not allow circuits which have fan-in gates. A k-fan-out gate is the boolean function $f_k : \{0,1\} \to \{0,1\}^k$ given by $f_k(x) = (x, \ldots, x)$.

Consider the following physical system. For each edge of a combinatorial circuit associate a single subsystem with only two possible states, 0 and 1, i.e. associate a bit to every edge. Consider next a logic gate which has incoming edges labeled by bits $i_1, i_2, \ldots, i_j$ and outgoing edges labeled by $t_1, t_2, \ldots, t_k$. For such a logic gate, l, define the following energy function

$$E_l(i_1, i_2, \ldots, i_j, t_1, t_2, \ldots, t_k) = \begin{cases} 0 & \text{if } f_l(i_1, i_2, \ldots, i_j) = (t_1, t_2, \ldots, t_k) \\ \Delta & \text{otherwise} \end{cases}, \tag{1}$$

where f_l is the function the logic gate is computing. Note that this energy contributes 0 when the inputs and outputs follow the rules of the logic gate, but is Δ otherwise. Now define the energy for a particular configuration of the bits associated with edges as the sum over all logic gates of these energy terms:

$$E_C(s) = \sum_{l \in L} E_l \tag{2}$$

where each E_l acts on the appropriate input and output bits and s is the collection of bits for the entire system. This energy is a function of the labels on all of the edges. Its zero energy configurations are ground states which consist of configurations which correctly compute the logical function corresponding to the circuit C. Note that at this point the ground state is degenerate: every valid input (and corresponding output) defines a valid zero energy configuration. We call an energy function constructed in this fashion from a circuit a *circuit energy function.*

Suppose further that, in addition to imposing energy constraints to enforce logic gates, we also impose energy constraints to fix a particular input to the circuit. Suppose that the input vertices are $v_1, v_2, \ldots, v_n$ and we wish to force the input $x_1, x_2, \ldots, x_n$, where $x_i \in \{0,1\}$. Then to each edge whose bit is labeled s_i lead away from vertex v_i we can associate an energy term $E(s_i) = 0$ if $s_i = x_i$ and $E(s_i) = \Delta$ otherwise. Taking a sum over all of these terms for the input vertices (and corresponding edges) we can thus add a term such that the ground state configuration consists now only of the input $x_1, \ldots, x_n$ propagated to the output of the circuit, $f(x_1, \ldots, x_n)$. In particular for input $x \in \{0,1\}^n$ define the

$$E_{C,x}(s) = E_C(s) + \sum_{i=1}^{n} E_{x_i}(s_i) \equiv E_C(s) + E_x(s), \tag{3}$$

where

$$E_{x_i}(s_i) = \begin{cases} 0 \text{ if } s_i = x_i \text{ where } s_i \text{ corresponds to input } v_i \\ \Delta \text{ if } s_i \neq x_i \text{ where } s_i \text{ corresponds to input } v_i \end{cases}. \tag{4}$$

We call an energy function made up of a circuit energy function plus an input forcing energy function a *computed circuit energy function.*

The above description of how to take a combinatorial circuit, C, plus its input, $(x_1, \ldots, x_n)$, and converting it into a many-spin energy function is what we term the *classical ground state spin computing* (CGSSC) model. Just as in quantum dot cellular automata [9, 10], the computation occurs when the system is in its ground state and is

spatially spread out across the device. Unlike in quantum dot cellular automata, however, this model can implement logic gates by using interactions beyond just pairwise interacting spins (or pseudo-spins in the case of the quantum dot configurations.) For example, constructing a quantum wire in the CGSSC model will correspond directly to an identical model in the semi-classical limit of quantum dot cellular automata models. Gates in quantum dot cellular automata, however, are implemented in a way which is not directly analogous to the CGSSC model. In section V we return to this issue and define a more general set of energy functions wherein our results still hold and compare this with quantum cellular automata models.

B. Example

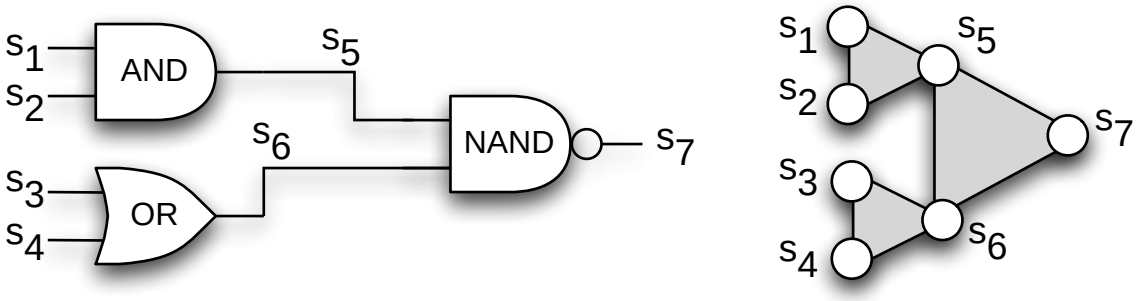

FIG. 1. Left: An example circuit whose circuit energy function is given in the text. The bits of the ground state spin energy function are labeled s_i. Right: the spin system corresponding to this circuit. Here the spins connected by a triangles have interactions between them as specified in the main text.

To be explicit, let's consider an example circuit involving four inputs, a few logical gates, and one output as diagramed in Fig. 1. The energy function for the AND gate is then, for example,

$$\begin{aligned} E_{AND}(s_1, s_2, s_3, s_4, s_5) &= \Delta((1-s_1)(1-s_2)s_5 + (1-s_1)s_2s_5 + s_1(1-s_2)s_5 + s_1s_2(1-s_5)) \\ &= \Delta(s_5 + s_1s_2 - 2s_1s_2s_5). \end{aligned} \tag{5}$$

The full circuit energy function is given by

$$\begin{aligned} E_C(s_1, s_2, s_3, s_4, s_5) = &\overbrace{\Delta(s_3 + s_4 + s_6 - s_3s_4 - 2s_3s_6 - 2s_4s_6 + 2s_3s_4s_6)}^{\text{OR}} \\ &+ \underbrace{\Delta(s_5 + s_1s_2 - 2s_1s_2s_5)}_{\text{AND}} + \underbrace{\Delta(1 - s_7 - s_5s_6 + 2s_5s_6s_7)}_{\text{NAND}}. \end{aligned} \tag{6}$$

Suppose that we wish to force the input to be $s_1 = s_2 = s_3 = 0$ and $s_4 = 1$. Then we would add the term

$$E_{C,x} = E_C + \Delta(s_1 + s_2 + s_3 + (1 - s_4)). \tag{7}$$

II. UNPROTECTED GROUND STATE SPIN COMPUTING FAILS AT FINITE TEMPERATURE

Above we have defined an energy function whose ground state deterministically carries out a circuit. Having defined this energy model we can now consider the physically important question of what happens to this model when the physical system described by this energy function is in thermal equilibrium at a finite temperature. Thus we are led to consider the Boltzmann distribution corresponding to a non-zero temperature version of a circuit energy function.

It then makes sense to consider whether the conditional probability of output $f(i_1, i_2, \ldots, i_n) = (t_1, t_2, \ldots, t_m)$ given the forced input $(i_1, i_2, \ldots, i_n)$ for this circuit is large enough to distinguish this output from a completely random outcome. It is easy to see that for at least some circuits C and inputs x, ground state computation will fail to correctly evaluate the function corresponding to the circuit as the size of the circuit being implemented grows. This follows directly from examining one of the most basic models in statistical physics, the one-dimensional Ising model [28]. This was pointed out in the context of quantum-dot cellular automata in [10] (see also [11, 29]). Here we reproduce this argument presenting it in two different forms. We do this not just to be pedagogical, but also because these two methods will generalize from the one dimensional case to more general circuit computing energy functions.

Consider the circuit computing energy function corresponding to inputing the bit 0 into a series of n identity gates:

$$E_I(s_1, \ldots, s_{n+1}) = \Delta\delta_{s_1,1} + \Delta \sum_{i=1}^{n} \delta_{s_i, \neg s_{i+1}}, \tag{8}$$

where $\neg s$ represents the negation of s: $\neg 0 = 1$ and $\neg 1 = 0$. It is convenient here, and in the sequel, to work with $\{+1, -1\}$ valued variables instead of the $\{0, 1\}$ valued s_i's. Thus we will define uppercase spin variables to be ± 1 valued versions of the s_i's via $S_i = 1 - 2s_i$. Then the above energy function can be written as

$$E_I(S_1, \ldots, S_{n+1}) = \frac{\Delta}{2}(1 - S_1) + \frac{\Delta}{2} \sum_{i=1}^{n} (1 - S_i S_{i+1}), \tag{9}$$

which we recognize as the one dimensional Ising model with ferromagnetic couplings and a boundary term which is a local field. The ground state of this is simply all $s_i = 0$ ($S_i = +1$), i.e. the initial 0 has been copied by identity gates down the line.

The thermal ensemble arising from this energy function is given by

$$Pr(S_1, \ldots, S_{n+1}) = \frac{\exp(-\beta E_I(S))}{Z}, \tag{10}$$

where Z is the partition function,

$$Z = \sum_{S \in \{+1,-1\}^{n+1}} \exp(-\beta E_l(S)), \tag{11}$$

and $\beta = (k_B T)^{-1}$ is the inverse temperature. It is well known that the one dimensional Ising model does not order at finite temperature in the thermodynamic limit [28]. From our perspective, we observe that the system will fail to correctly transmit the 0 down the line at finite temperature except in a window of size $k_B T < \frac{\Delta}{n}$ which goes to zero as the system size goes to infinity.

A. Transfer Matrix Reinterpreted as a Probabilistic Circuit

The standard method for solving this model is the transfer matrix method which we review and then reinterpret. To the energy function add a term dependent on a variable γ for the last spin:

$$E'_I(S, \gamma) = E_I(S) + \gamma S_{n+1}. \tag{12}$$

If we calculate Z' for this energy function, we can then use this to calculate the probability that S_{n+1} is $+1$ via $Pr(S_{n+1} = +1) = \langle \frac{1}{2}(1 + S_{n+1}) \rangle = \frac{1}{2}(1 + \langle S_{n+1} \rangle)$ where

$$\langle S_{n+1} \rangle = -\frac{1}{\beta} \frac{\partial \ln Z'}{\partial \gamma}\bigg|_{\gamma=0} = -\frac{1}{\beta Z'} \frac{\partial Z'}{\partial \gamma}\bigg|_{\gamma=0}. \tag{13}$$

Define the two by two matrices

$$[M_i]_{S_{i+1}, S_i} = \exp\left[-\frac{b}{2}(1 - S_i S_{i+1})\right], \tag{14}$$

where $b = \beta\Delta$, as well as the column vector

$$[v]_{S_1} = \exp\left[-\frac{b}{2}(1 - S_1)\right], \tag{15}$$

and the row vector,

$$[w^T]_{S_{n+1}} = \exp[-\beta\gamma S_{n+1}]. \tag{16}$$

Explicitly the partition function Z' is

$$Z' = \sum_{S\in\{+1,-1\}^{n+1}} \exp\left[-\beta\gamma S_{n+1} - \frac{b}{2}\left(1 - S_1 + \sum_{i=1}^{n}(1 - S_i S_{i+1})\right)\right]. \tag{17}$$

This can then be written as

$$Z' = \sum_{S\in\{+1,-1\}^{n+1}} [w^T]_{S_{n+1}} \prod_{i=n}^{1} [M_i]_{S_{i+1},S_i} v_{S_1} = w^T M_n M_{n-1} \dots M_1 v, \tag{18}$$

where the last equation is a row vector, a matrix product, and a column vector thus giving a scalar. This is the transfer matrix form of the solution: the partition function is written as a product of "transfer matrices" and, in this case, sandwiched between an "initial" and "final" vector.

Define the following matrices and vectors:

$$[P_i]_{S_{i+1},S_i} = \frac{[M_i]_{S_{i+1},S_i}}{1+\exp(-b)} \tag{19}$$

$$[p_{in}]_{S_1} = \frac{[v]_{S_1}}{1+\exp(-b)}. \tag{20}$$

We can rewrite Z' as

$$Z' = (1-\exp(-b))^{-(n+1)} w^T P_n P_{n-1} \dots P_1 p_{in}. \tag{21}$$

It is at this point that we begin to see our *reinterpretation* of the transfer matrix method. In particular we note that p_{in} is a vector of probabilities (that is it sums to unity) and P_i is a stochastic matrix (its columns sum to unity). In other words encoded into Z' is a classical preparation of a probabilistic bit, followed by an evolution according to probabilistic gates. Suppose that we let $p_{out} = P_n P_{n-1} \dots P_1 p_{in}$ denote the output of this probabilistic circuit. Then

$$Z' = (1-\exp(-b))^{n-1} \langle w \rangle_{p_{out}} \tag{22}$$

where the sample is taken from the output of the probabilistic circuit. Notice that at $\gamma = 0$, w is a vector made up of all components equaling 1. In this case, $\langle w \rangle_{p_{out}}|_{\gamma=0} = 1$ since then this term is a sum over all the outputs to the probabilistic circuit. Further

$$-\frac{1}{\beta}\frac{\partial Z'}{\partial \gamma}\bigg|_{\gamma=0} = (1-\exp(-b))^{n+1} \langle \sigma \rangle_{p_{out}} \tag{23}$$

where $\sigma = (+1,-1)^T$. Thus

$$\langle S_{n+1} \rangle = \langle \sigma \rangle_{p_{out}}. \tag{24}$$

In other words the expectation value of the last spin in the chain is equal to the expectation value for the output of the circuit starting with p_{in} and then applying the probabilistic gates $P_1, \dots, P_n$. Note that the combinatorial factors $(1+\exp(-b))^{n+1}$ have canceled out.

The above description tells us that we can think about the one dimensional Ising spin chain with a forced boundary term as a probabilistic circuit related to our ground state spin computation. In particular the probabilistic circuit starts with the possibility of an incorrectly initialized boundary. This is formulated in the preparation probability vector p_{in}. Then with probability $\frac{\exp(-b)}{1-\exp(-b)}$ the ground state spin computation of an identity gate is replaced by a bit flip gate. The probability of the final spin of the chain being in a particular state is then directly given by the probability that the probabilistic circuit outputs a particular value.

In this model the probability of the ground state properly computing the desired identity circuit is such that the information is washed out at finite temperature. To see this we must actually calculate $\langle S_{n+1} \rangle$. This can be done most easily by diagonalizing the gates. In particular if we define the Hadmard matrix,

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{25}$$

then

$$P_i = HDH^{-1} = HDH \tag{26}$$

where

$$D = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1-\exp(-b)}{1+\exp(-b)} \end{bmatrix}. \tag{27}$$

Thus

$$\langle S_{n+1} \rangle = \sigma^T H D^n H p_{in} \tag{28}$$

which can be reduced to

$$\langle S_{n+1} \rangle = \left(\frac{1-e^{-b}}{1+e^{-b}} \right)^{n+1} = \left[\tanh \frac{b}{2} \right]^{n+1}. \tag{29}$$

From this expression we see that the probability that the final spin is aligned with the input quickly drops to $1/2$ as a function of n (unless $b > n$, i.e. in a small window of temperature below $\frac{\Delta}{n}$.) Thus the circuit fails with high probability as we make the chain longer and longer. This is exactly the result of [11]. In this respect, ground state spin computing in this model is not tolerant to errors arising from working at finite temperature. Note that we do not consider scaling of the temperature to stay within the small window where the computation is correctly performed to be a fault-tolerant method as it requires unreasonable physical resources as the system size gets larger.

B. A Second Approach Using Controlled-Not Gates

Having shown in the previous subsection how one can reinterpret the transfer matrix method as a probabilistic circuit for the one dimensional Ising model with forced boundary, we next present a second way to derive this observation. This method is a direct variation on the method used in [30] to study Ising models on Cayley trees.

On the $n+1$ spins, define the controlled-not on positions i and $i+1$ as the function, C_i, which maps $\{+1,1\}^{n+1}$ to $\{+1,-1\}^{n+1}$ via

$$C_i(S_1, \ldots, S_i, S_{i+1}, S_{i+2} \ldots, S_n) = (S_1, \ldots, S_i, S_{i+1}S_i, S_{i+2}, \ldots, S_n). \tag{30}$$

We call this a controlled-not because it functions as a deterministic gate which, controlled on the spin S_i either does nothing to the spin in position $(i+1)$ (if $S_i = +1$), or flips the spin in position $(i+1)$ (if $S_i = -1$.) Further define the function which is the composition of controlled-not's starting at the first spin and working forward (note that $C_i \circ C_{i+1} \neq C_{i+1} \circ C_i$):

$$C = C_n \circ C_{n-1} \circ \cdots \circ C_2 \circ C_1. \tag{31}$$

Notice that if we start with spin 1 in the state S_1 and all of the other spins $S_2, \ldots, S_{n+1}$ in $+1$, then the action of C is to *copy* S_1 down the line: $C(S_1, +1, \ldots, +1) = (S_1, S_1, \ldots, S_1)$. In this sense C is the operation of applying identity gates as information is propagated down the spin chain. Note that the C_i are bijections as is C and that C_i is its own inverse. Thus

$$C^{-1} = C_1 \circ C_2 \circ \ldots \circ C_{n-1} \circ C_n. \tag{32}$$

Explicitly,

$$C(S_1, S_2, S_3, \ldots, S_{n+1}) = (S_1, S_1S_2, S_1S_2S_3, \ldots, \prod_{i=1}^{n+1} S_i) \tag{33}$$

and

$$C^{-1}(S_1, S_2, S_3, \ldots, S_{n+1}) = (S_1, S_1S_2, S_2S_3, \ldots, S_nS_{n+1}). \tag{34}$$

Consider the energy function on $n+1$ spins:

$$E'(S'_1, S'_2, \ldots, S'_{n+1}) = \frac{\Delta}{2} \sum_{i=1}^{n+1} (1 - S'_i). \tag{35}$$

Then clearly a system in the thermal state described by this energy function has each spin in $+1$ with probability $1-p = 1/(1+\exp(-b))$ and -1 with probability $p = \exp(-b)/(1+\exp(-b))$. Suppose that we apply the C function to this system. That is consider the above system in its thermal ensemble and consider physically applying the controlled-not gates. Then certainly this can be thought of as a system in which the first bit is prepared in 0 with probability $1-p$, with probability $1-p$ the controled-nots successfully copy the bit down the line, and with probability p the output of the controlled-not is flipped. This is exactly the probabilistic circuit as described in the previous section.

How is this related to our original E? Let the unprimed variables be the spins after C has been applied to the system. Then

$$Pr(S_1, \ldots, S_{n+1}) = \sum_{S'_1, \ldots, S'_{n+1} \in \{\pm 1\}} Pr(S_1, \ldots, S_{n+1} | S'_1, \ldots, S'_n) Pr(S'_1, \ldots, S'_{n+1}) \tag{36}$$

where

$$Pr(S_1, \ldots, S_{n+1} | S'_1, \ldots, S'_n) = \delta_{C(S'_1, \ldots, S'_{n+1}), (S_1, \ldots, S_{n+1})} = \delta_{(S'_1, \ldots, S'_{n+1}), C^{-1}(S_1, \ldots, S_{n+1})}. \tag{37}$$

Therefore

$$Pr(S_1, \ldots, S_{n+1}) = Pr(C^{-1}(S_1, \ldots, S_{n+1})). \tag{38}$$

This implies that the probability distribution produced by taking the thermal ensemble for $E'(S'_1, \ldots, S'_{n+1})$ and applying C to the system is equivalent to a thermal ensemble with a new energy function

$$E'(C^{-1}(S_1, \ldots, S_{n+1})). \tag{39}$$

A quick calculation using Eq. (34) finds that

$$E'(C^{-1}(S_1, \ldots, S_{n+1})) = \frac{\Delta}{2}(1 - S_1) + \frac{\Delta}{2} \sum_{i=1}^{n} (1 - S_i S_{i+1}) \tag{40}$$

which we see is equal to our original energy function E. Reversing the above argument we thus see that the energy function E can equally well be thought of as the ensemble corresponding to E' followed by the application of the mapping C. Further this later ensemble and computation has a clear interpretation in terms of a probabilistic initialization followed by probabilistic gates: when we apply controlled-nots with error target values then this causes a bit flip error in copying the information down the line. Thus we see that we can derive the same probabilistic gate expression for the one-dimensional Ising model as that which arose by reinterpreting the transfer matrix as a probabilistic circuit.

III. THE PARTITION FUNCTION AS A PROBABILISTIC CIRCUIT

We now turn to the more general setting of a circuit energy function for a generic circuit $C = (G, L)$. We begin by focusing on circuit energy functions without a forced input.

Let E_C be the energy function for the circuit C. For each logic gate, l, with inputs $i_1, i_2, \ldots, i_j$ and output $t_1, t_2, \ldots, t_k$, define the following *tensor*,

$$M^{(l)}_{i_1, \ldots, i_j, t_1, \ldots, t_k} = \begin{cases} 1 & \text{if } f_l(i_1, i_2, \ldots, i_j) = (t_1, t_2, \ldots, t_k) \\ e^{-b} & \text{otherwise} \end{cases}, \tag{41}$$

where $b = \beta\Delta$ as before. Given this definition, we can now write the partition function Z as the contraction of a tensor network [31] plus a sum over all inputs and outputs. What is a tensor network? Take a graph and map a tensor to every vertex in such a manner that the tensor has an index for every edge of the relevant vertex. One can then perform a sum over two tensor indices connected via edges in the graph. A tensor network is thus a graph where vertices are tensors, and edges are tensor indices. Free edges are indices which have not been summed over and

connected edges are ones for which a sum has been performed. The entire network (graph) then represents itself a tensor with a number of unsummed indices equal to the number of free edges.

Given the tensors $M^{(l)}$ and the graph given by the circuit (which will have free edges for the inputs and outputs), we can build a tensor network out of these $M^{(l)}$s. This tensor network will itself be a tensor network having indices for the inputs and outputs. Call this tensor $M^C_{i_1,\ldots,i_n,t_1,\ldots,t_m}$ for inputs $i_1,\ldots,i_n$ and output $t_1,\ldots,t_m$. Then it is easy to see that the partition function is equal to

$$Z = \sum_{i_1,\ldots,i_n,t_1,\ldots,t_m \in \{+1,-1\}} M^C_{i_1,\ldots,i_n,t_1,\ldots,t_m} \tag{42}$$

In other words, the partition function is given by the sum over all inputs, and all outputs of the value of the tensor network.

What does this have to do with probabilistic circuits? Well now instead of using the tensors M, instead use

$$P^{(l)}_{i_1,\ldots,i_j,t_1,\ldots,t_k} = \frac{M^{(l)}_{i_1,\ldots,i_j,t_1,\ldots,t_k}}{1+(2^k-1)e^{-b}} \tag{43}$$

or in other words

$$P^{(l)}_{i_1,\ldots,i_j,t_1,\ldots,t_k} = \begin{cases} \frac{1}{1+(2^k-1)e^{-b}} \text{ if } f_l(i_1,i_2,\ldots,i_j) = (t_1,t_2,\ldots,t_k) \\ \frac{e^{-b}}{1+(2^k-1)e^{-b}} \text{ otherwise} \end{cases} \tag{44}$$

It is easy to check that this defines a *stochastic* matrix, i.e. a probabilistic gate. Indeed it is a probabilistic gate which performs the correct function of the gate with probability $\frac{1}{1+(2^k-1)e^{-b}}$ and randomly flips the output to one of the other outputs with equal probability $\frac{e^{-b}}{1+(2^k-1)e^{-b}}$.

Notice, importantly, that the difference between $P^{(l)}$ and $M^{(l)}$ is a constant which depends on only on the number of output bits. So suppose we consider the tensor network for the circuit made now with $P^{(l)}$s and not $M^{(l)}$s. Then we can write the partition function as

$$Z_C = K \sum_{i_1,;i_n,t_1,\ldots,t_m \in \{+1,-1\}} P_{i_1,\ldots,i_n,t_1,\ldots,t_m} \tag{45}$$

where K is the simple combinatorial factor

$$K = \prod_l (1+(2^{k_l}-1)e^{-b}) \tag{46}$$

and k_l is the number of outputs of the lth gate. Now examine $P_{i_1,\ldots,i_n,t_1,\ldots,t_m}$. This is nothing more than the probability that we get output $t_1,\ldots,t_m$ given that we had input $i_1,\ldots,i_n$ to the probabilistic circuit with probabilistic gates $P^{(l)}$. In other words, we can express the partition function as

$$Z_C = K \sum_{inputs} \sum_{output} \Pr(\text{output}|\text{input}) \tag{47}$$

Thus the partition function is really hiding a probabilistic computation. Notice further that we can explicitly calculate that partition function because $\sum_{output} \Pr(\text{output}|\text{input}) = 1$,

$$Z_C = K 2^n \tag{48}$$

where n is the number of input bits.

Now we turn to the case where we force the input to the circuit. Let $x_1,\ldots,x_n$ be the inputs to the circuit C and $E_{C,x}$ be the energy function for the forced computation. Define the modified energy function

$$E_{C,x,y} = E_{C,x}(s) + E_y(s), \tag{49}$$

where

$$E_y(s) = \sum_{i=1}^m \gamma_i E_{y_i}(s) \tag{50}$$

with

$$E_{y_i}(s) = \begin{cases} \gamma_i t_i & \text{if } y_i = 1 \\ \gamma_i (1 - t_i) & \text{if } y_i = 0 \end{cases}, \tag{51}$$

and the sum is over the output bits, t_i. Call the partition function associated with this setup $Z_{C,x,y}$.

Note that $Z_{C,x,y}|_{\gamma_1 = \cdots = \gamma_m = 0}$ is equal to the partition function for energy function with a forced input x, $Z_{C,x}$. Define

$$P_{in}(i_1, \ldots, i_n) = \prod_{j=1}^{n} P_{in,j}(i_j), \tag{52}$$

where

$$Pr_{in,j}(i_j) = \begin{cases} \frac{1}{1+e^{-b}} & \text{if } i_j = x_j \\ \frac{e^{-b}}{1+e^{-b}} & \text{if } i_j \neq x_j \end{cases}. \tag{53}$$

Then it is easy to see that

$$Z_{C,x} = K' \sum_{i_1, \ldots i_n, t_1, \ldots, t_m \in \{+1,-1\}} P_{i_1,\ldots,i_n,t_1,\ldots,t_m} P_{in}(i_1, \ldots, i_n), \tag{54}$$

where

$$K' = K(1 - e^{-b})^n. \tag{55}$$

Thus we see that we can interpret $Z_{C,x}$ as a sum over a probabilistic circuit, but now with a probabilistic input corresponding to each bit being flipped with a probability $\frac{e^{-b}}{1+e^{-b}}$. Again, because we are summing over all outputs, we can explicitly perform this sum

$$Z_{C,x} = K'. \tag{56}$$

We are interested in computing the probability that the thermal ensemble for the forced input circuit has output given by $y_1, \ldots, y_n$. This can be calculated by

$$\begin{aligned} Pr(t_1 = y_1, \ldots, t_m = y_m) &= -\frac{1}{\beta Z_{C,x,y}} \frac{\partial^m Z_{C,x,y}}{\partial \gamma_1 \cdots \partial \gamma_m}\bigg|_{\gamma_1 = \cdots = \gamma_m = 0} \\ &= -\frac{1}{\beta Z_{C,x}} \frac{\partial^m Z_{C,x,y}}{\partial \gamma_1 \cdots \partial \gamma_m}\bigg|_{\gamma_1 = \cdots = \gamma_m = 0} \end{aligned} \tag{57}$$

Note that

$$Z_{C,x,y} = K' \sum_{i_1, \ldots i_n, t_1, \ldots, t_m \in \{+1,-1\}} e^{-\beta E_y(s)} P_{i_1,\ldots,i_n,t_1,\ldots,t_m} P_{in}(i_1, \ldots, i_n) \tag{58}$$

so that the partial derivatives may be evaluated only over the first term. There we find that

$$\frac{\partial^m e^{-\beta E_y(s)}}{\partial \gamma_1 \cdots \partial \gamma_2}\bigg|_{\gamma_1 = \cdots = \gamma_n = 0} = -\beta \prod_{i=1}^{m} \delta_{t_i, y_i}. \tag{59}$$

Thus we find that

$$Pr(t_1 = y_1, \ldots, t_m = y_m) = \sum_{s_1, \ldots, s_n} P_{s_1,\ldots,s_n,y_1,\ldots,y_n} P_{in}(s_1, \ldots, s_n). \tag{60}$$

This is the main result of this section: the probability of the ground state spin computing model output bits is equal to the probability of running the probabilistic circuit described by P on inputs described by input probability distribution P_{in}. The gates in P are simply noisy versions of the deterministic gates which fail with a fixed probability related to the temperature.

This result is interesting in two manners. First it allows one to use techniques designed for probabilistically failing gates to be ported over to our model. For instance this will allow us to use methods for constructing fault-tolerant circuits. It is also interesting in that it shows that certain many-body quantum systems can be efficiently simulated. Computed circuit energy functions can be simulated by directly implementing the probabilistic gate that these systems correspond to. Here we have shown that output bits for the circuit energy function are related to the output bits for the probabilistic computation. A straightforward generalization shows that this is also true of other bits in the system. We can efficiently simulate systems with computed circuit energy functions by executing the probabilistic computation these energy functions represent.

IV. THE GATE MODEL DERIVATION

Next let us turn to the same analysis as the prior subsection, but now using the gate trick as we did for the one-dimensional model in Section II B. To do this we must first define the equivalent of the controlled-not gate. Consider a logical gate l with input $i_1, \ldots, i_j$ and output $t_1, \ldots, t_k$ which computes the function $(t_1, \ldots, t_k) = f(i_1, \ldots, i_j)$. Define the function C_l from $\{0,1\}^j \times \{0,1\}^k$ to $\{0,1\}^j \times \{0,1\}^k$ as

$$C_l(i_1, \ldots, i_j, t_1, \ldots, t_k) = \begin{cases} (i_1, \ldots, i_j, f_l(i_1, \ldots, i_j)) & \text{if } t_1 = \cdots = t_k = 0 \\ (i_1, \ldots, i_j, 0, \ldots, 0) & \text{if } (t_1, \ldots, t_k) = f(i_1, \ldots, i_j) \\ (i_1, \ldots, i_j, t_1, \ldots, t_k) & \text{otherwise} \end{cases} \tag{61}$$

This function thus computes the function on input $(i_1, \ldots, i_j, 0, \ldots, 0)$, uncomputes the function on input $(i_1, \ldots, i_j, f_l(i_1, \ldots, i_j))$, and does nothing otherwise. Note that C_l is a bijection and is self-inverse, $C_l^{-1} = C_l$. Defining an order to gates in our circuit such that the circuit computes properly, l_1 first, l_2 second, etc. Then we can define the function on all our bits of

$$C_C = C_{l_r} \circ \cdots \circ C_{l_2} \circ C_{l_1} \tag{62}$$

Suppose we initially start with input bits initialized to an input $i_1 = x_1, \ldots, i_n = x_n$ and all other bits initialized to 0. Then if we apply C_C to these bits we will place the result of the computation in their appropriate locations across the circuit.

Next define an energy function on all of our system. This is most easily defined in terms of the inputs to the full circuit and the outputs to the gates $l \in L$,

$$E'(s') = \sum_{i'_1, \ldots, i'_m} \Delta(\delta_{i'_1, \neg x_1} + \cdots + \delta_{i'_m, \neg x_m}) + \sum_{l \in L} E'_l(t'_1, \ldots, t'_{k_l}). \tag{63}$$

and

$$E'_l(t'_1, \ldots, t'_{k_l}) = \begin{cases} 0 & \text{if } t'_1 = \cdots = t'_{k_l} = 0 \\ \Delta & \text{otherwise} \end{cases} . \tag{64}$$

This energy function thus assigns for non-global inputs an energy penalty for output to gates being anything different that 0^{k_l}. For global inputs it adds a penalty for every input which is not the correct input from x. The thermal ensemble resulting from this system is then trivially described: the input qubits are in the correct input x_i with probability $\frac{1}{1+\exp(-b)}$ and the internal output bits of a gate are grouped together and have a probability $\frac{1}{1+(2^{k_l}-1)\exp(-b)}$ being all zeros and probability $\frac{e^{-b}}{1+(2^{k_l}-1)\exp(-b)}$ to be anything else.

Imagine applying C_C to the state described by the ensemble for $E'(s')$. Clearly the ground state of $E'(s')$ is such that the circuit C will be correctly computed. Further, because the ensemble related to $E'(s')$ has erred inputs, the resulting probability distribution will have erred inputs. Finally the gates will function properly only when the output was properly initialized to all 0s. This occurs with probability $\frac{1}{1+(2^{k_l}-1)\exp(-b)}$ for logic gate l and otherwise the gate fails by an equally probable error state with probability $\frac{e^{-b}}{1+(2^{k_l}-1)\exp(-b)}$. In other words the ensemble is exactly the one describing the probabilistic circuit in Section III. Let us now show that this corresponds to the thermal ensemble of our original E_C construction.

As in Section II B the trick here is to express the probability distribution for the ensemble after applying C_C as

$$Pr(s) = \sum_{s'} Pr(s|s') Pr(s'), \tag{65}$$

where

$$Pr(s|s') = \delta_{C_C(s'),s} = \delta_{s', C_C^{-1}(s)}, \tag{66}$$

such that

$$Pr(s) = Pr(C_C^{-1}(s)). \tag{67}$$

Thus the probability distribution resulting from after the application of C_C to the $E'(s')$ thermal ensemble is equal to the probability distribution arising from $E'(C_C^{-1}(s))$.

We will now show the $E'(C^{-1}(s))$ is exactly the energy function we would define for a ground state spin computing construction for our circuit C. Recall that

$$C_C^{-1} = C_{l_1} \circ C_{l_2} \cdots \circ C_{l_r}. \tag{68}$$

Note that in evaluating $E'(C^{-1}(s))$, which is a sum of terms, we can evaluate the action action of C_C^{-1} on each of these terms separately and then re-sum. Thus we first examine

$$C_C^{-1}\left[\sum_{i'_1,\ldots,i'_m} \Delta(\delta_{i'_1,\neg x_1} + \cdots + \delta_{i'_m,\neg x_m})\right] = \sum_{i'_1,\ldots,i'_m} \Delta(\delta_{i'_1,\neg x_1} + \cdots + \delta_{i'_m,\neg x_m}) \tag{69}$$

This is because the inputs to the circuit commute through the individual gate elements C_{l_i}. Indeed this observation along with the order of C_C^{-1} allows one to derive the term arising from each $E'_l(C_l(i_1,\ldots,i_{j_l},t_1,\ldots,t_{k_l}))$ term independently. In particular the only term which is not Δ for this energy function is

$$E'_l(C_l(i_1,\ldots,i_{j_l},f_l(i_1,\ldots,i_{j_l}))) = 0 \tag{70}$$

But this simply means that

$$E'_l(i_1,\ldots,i_{j_l},t_1,\ldots,t_{k_l}) = 0 \tag{71}$$

only when $(t_1,\ldots,t_{k_l}) = f_l(i_1,\ldots,i_{j_l})$ and is Δ otherwise. This is just our normal definition of the circuit energy function.

Thus we see that using the C_C construction one can construct a probabilistic circuit from the thermal ensemble for $E'_{C,x}(s')$ which is the probabilistic circuit described in Section III. Further we have shown that this can be thought of as arising from the original energy function $E_{C,x}$. This provides an alternative derivation of our main result.

V. GENERALIZED CLASSICAL GROUND STATE SPIN COMPUTATION MODELS

Having shown that classical ground state spin computing at non-zero temperature can be mapped to probabilistic circuits it is interesting to return to our original model and consider what generalizations of the energy function allow for this mapping to occur. The crucial assumption during our derivation of the probabilistic re-interpretation of the partition function was that we could convert the tensors $M^{(l)}$ of Eq. (41) to tensors $P^{(l)}$ of Eq. (43) in such a way that $P^{(l)}$ could be interpreted as a probablistic gate. This requires (1) the division of tensor indices into input and output indices and (2) for each input to the circuit the normalization to make the outputs probabilities (positive and sum to unity) is the same.

Let us begin by showing that not all tensors and divisions into input and output vertices can result in a probabilistic interpretation. This can be demonstrated with a simple example on two bits:

$$E(s_1,s_2) = \begin{cases} 0 & \text{if } s_1 = s_2 = 0 \\ \Delta & \text{if } s_1 = s_2 = 1 \\ 2\Delta & \text{otherwise} \end{cases}. \tag{72}$$

If we chose s_1 as an input and s_2 as an output, this gives rise to the tensor

$$M = \begin{bmatrix} 1 & e^{-2b} \\ e^{-2b} & e^{-b} \end{bmatrix}. \tag{73}$$

In order that the first column of this matrix sum to unity we must divide by $1 + e^{-2b}$. However when we do this the second column will not sum to 1. A similar argument occurs if we had chosen s_1 as the output and s_2 as the input. Thus there is no way to scale M in such a way that it can be interpreted as a probabilistic gate.

A more relevant example of a failure for a model to not be amenable to our scaling arising in quantum-dot cellular automata models. In quantum dot cellular automata models bistable quantum dots are engineered in such a way that the ground state enacts a classical computation as in our model. Most of the treatments of these models deal with the quantum mechanics of these devices [9, 10, 32]. One can, however, build a classical model of these systems as was done, for example, by Wang in [11]. In this approximation one derives a classical statistical mechanical many spin system as in our model. When one does this for a quantum-dot cellular automata wire one ends up with exactly the model we can consider above in Section II. However when one considers this for the majority gate constructions (see

[9–11]) one obtains a model for which our technique cannot be applied. In particular the majority gate that they propose has a classical energy function which is

$$E_M(S_1, S_2, S_3, S_m) = -\Delta(S_1S_m + S_2S_m + S_3S_m) = -\Delta(S_1 + S_2 + S_3)S_m. \tag{74}$$

Here S_1, S_2, S_3 are the input spins and S_m is the output spin. (Note that we have used one less spin than in the traditional presentation, but this does not affect our conclusions as one can think of the extra spin as being connected to the central spin by an identity gate.) Note that the lowest energy configuration *for a given* input correctly computes the majority function. That is if we fix S_1, S_2, and S_3, then the lowest energy value of S_m is given by the majority of the inputs, $S_m = MAJ(S_1, S_2, S_3)$. Note, however, that if we sum over all outputs for input $S_1 = S_2 = S_3 = +1$

$$\sum_{S_m|S_1=S_2=S_3=+1} \exp(-bE_M(S_1, S_2, S_3, S_m)) = e^{-3b} + e^{-3b}, \tag{75}$$

but if one uses input $S_1 = S_2 = +1$ and $S_3 = -1$ one obtains

$$\sum_{S_m|S_1=S_2=+1,S_3=-1} \exp(-bE_M(S_1, S_2, S_3, S_m)) = e^{b} + e^{-b}. \tag{76}$$

Thus we cannot properly globally normalize the entire tensor in such a way as to to obtain a probabilistic gate.

However, the same two-spin energy function can be used to describe a fanout gate, in contrast to the many-spin interaction used in Eq. (1),

$$E_F(S_1, S_2, \ldots, S_{k+1}) = -\Delta S_1 \sum_{i=2}^{k+1} S_i. \tag{77}$$

Here S_1 is the input and the other spins are the output. By changing the direction of the circuit, we can show that this element can be represented as a probabilistic gate. I.e. the majority gate used in quantum dot cellular automata can not be mapped to a probabilistic circuit, but reversing the role of the inputs and outputs, this gate can be used to construct a fan-out gate. To see this note that, as for our previous fanout model, the ground state of this system, where the computation is performed, is degenerate. However now, as opposed to a single excited state there are multiple excited states. For example if we consider a fan-out to two spins we obtain an M as given in Table I. From

S_1	S_2	S_3	$e^{-\beta E_F}$
$+1$	$+1$	$+1$	e^{2b}
$+1$	$+1$	-1	1
$+1$	-1	1	1
$+1$	-1	-1	e^{-2b}
-1	$+1$	$+1$	e^{-2b}
-1	$+1$	-1	1
-1	-1	1	1
-1	-1	-1	e^{2b}

TABLE I. The 2 output fan-out Boltzman factor.

this table we see that while the outputs that are in error have different Boltzman factors, we see that the sum over the output spins is indeed independent of the output. Indeed when we interpret this as a probabilistic gate by dividing out by $K = 2 + e^{2b} + e^{-2b}$, then we see that this is a probabilistic gate which, instead of having a global failure where all possible output failures have equal probability, the fan-out spins have independent error probabilities. In other words it is as if the fan-out occurred and then with probability $\frac{e^{-b}}{e^{-b}+e^{b}}$ each of the outputs is *independently* flipped. We will use this fan-out gate in the next section when we discuss fault-tolerance.

What is a necessary and sufficient condition for an energy function to allow for a proper normalization to a probabilistic gate? If we insist that this work for all temperatures, it is easy to give this answer. Let $E^{(l)}(i_1, \ldots, i_j, t_1, \ldots, t_k)$ denote the energy function for a logic gate with inputs $i_1, \ldots, i_j$ and output $t_1, \ldots, t_k$. Let $\mathcal{E}(i_1, \ldots, i_k) = \{E(i_1, \ldots, i_k, t_1, \ldots, t_k) | t_1, \ldots, t_k \in \{0, 1\}\}$ denote the set of energies for a given input. Then the necessary and sufficient condition is that the $\mathcal{E}(i_1, \ldots, i_k)$'s must be permutations of each other. The sufficiency of this condition is obvious. Necessity follows from noting that we are requiring this to hold for all temperatures and thus the sum over outputs of the $\exp(-\beta E)$ must be a rearrangement of this sum and therefore a permutation of the output energies. Note that this condition implies that the ground states of a logic gate energy function must be degenerate for all possible inputs (but note that this alone is not sufficient for the condition to hold.)

A. Computational Complexity of Identifying Probabilistic Circuits in Thermal Systems

Having established a necessary and sufficient condition for the possibility of scaling the $M^{(l)}$ tensors as probabilistic gates, a natural follow up question is how one can determine whether it is possible to take an energy function $E(s_1, \ldots, s_n)$ and determine whether or not it can be interpreted as a probabilistic circuit. Here we will show that this problem is computationally intractable, even when we are given information about the breakdown of $E(s_1, \ldots, s_n)$ into terms which we wish to interpret as gates.

Suppose that we are told the $E(s_1, \ldots, s_n)$ is made up of a sum over logic gates and preparations

$$E(s_1, \ldots, s_n) = \sum_l E_l, \tag{78}$$

i.e. we are given a specification of the E_l in terms of a constant number of bits and we are told that each of these E_l will correspond to a logic gate or a preparation of initial inputs. We can then ask: can we interpret this as a probabilistic circuit with or without prepared inputs? Every E_l involves some inputs and output bits, but in general we will not be told which of these are inputs and which are outputs. In general an $M^{(l)}$ may thus have many different tuples of input bits and output bits such that we may interpret $P^{(l)}$ as a probabilistic gate by proper global normalization. For example gates which have doubly stochastic $P^{(l)}$ tensors may be run either backwards or forwards (a doubly stochastic matrix is a stochastic matrix whose rows and columns sum to 1.) Another example is given by a fan-out gate in our original energy function model

$$E_f(S_1, S_2, S_3) = \begin{cases} 0 & \text{if } S_1 = S_2 = S_3 \\ \Delta & \text{otherwise.} \end{cases} \tag{79}$$

In this case we may interpret the $M^{(l)}(S_1, S_2, S_3)$ arising from this energy function as having input S_1 and outputs S_2 and S_3, input S_2 and outputs S_1 and S_3, or input S_3 and outputs S_2 and S_3. Contrary to this, suppose that the energy function is that of the one-to-two fanout described in Eq. (77),

$$E_F(S_1, S_2, S_3) = -\Delta S_1(S_2 + S_3). \tag{80}$$

The $M^{(l)}(S_1, S_2, S_3)$ for this energy function has only one possible orientation between inputs and output: S_1 is an input and S_2 and S_3 are outputs. This in general will lead to the following question: given the possible input and outputs tuples for the $M^{(l)}$s is it possible to assign these in such a way as to interpret the resulting total tensor as a probabilistic circuit. We will now show that this problem is NP-complete and is therefore (unless P = NP), in general, intractable.

To see that the problem is NP-complete proceed as follows. Assign to each bit of our physical system a boolean variable, x_i. Assume that a given logical energy function E_l is involved in only a constant number of bits. Since E_l is involved in a constant number of bits we can explicitly calculate all of the sets of inputs and output bits for which $M^{(l)}$ can be globally normalized so as to make it a probabilistic gate. The variables x_is are going to represent whether the corresponding bit is associated is an input or and output. In particular, TRUE, will represent that the bit is an input and FALSE will represent that the bit is an output. Thus from the sets of input and output bits for which one can consistently label inputs to $M^{(l)}$ we can construct a boolean function on the relevant bits such that this function is true if and only if a valid assignment of boolean variables respects a possible input and outputs. Thus, for example, for E_f above in Eq. (79), the boolean function would be

$$[(x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3)] \tag{81}$$

Constructing one such boolean function for each logic gate we can then construct a boolean function on all bits x_i which is the conjunction (logical and) of all of the gate boolean functions. Thus we see that by calculating for each E_l the possible combinations of inputs and outputs that have an allowed interpretation in terms of a probabilistic circuit, we have a one-to-one mapping of the problem to the problem of deciding whether there is a satisfying assignment for the boolean function made up of the conjunction of logical terms with only a constant number of involved bits. In order to demonstrate that the problem is NP-complete we need now only show that the boolean function we can construct in this manner are general enough so as to yield a version of the satisfaction problem which is NP-complete. In particular it is clear that this problem is in NP, since we can evaluate in polynomial time whether a given assignment of inputs and outputs can be interpreted as a probabilistic circuit. To demonstrate NP-completeness we need to show that it is at least as hard as another NP-complete problem.

We will show that the problem is NP-complete by showing that some instance of the problem can be mapped to the one-in-three satisfiability problem of Schaefer [33] (now sometimes called the monotone one-in-three satisfiability

problem.) Define $R(x,y,z)$ as the boolean function which is true iff exactly one of (x,y,z) is true. Notice that this is exactly the boolean formula of Eq. (81) (with $x_1 = x$, $x_2 = y$, and $x_3 = z$.) In the one-in-three satisfiability problem one is given a boolean function which is the conjunction (logical and) of clauses each of which are of the form $R(x,y,z)$ for a choice of variables x,y,z in the problem. Schaefer proved that the one-in-three satisfiability problem is NP-complete. Actually there is a slight subtlety here as the problem Schaefer considers must allow repeated variables. We will now show that it is possible to take a one-in-three satisfiability problem and convert it into an instance of the problem of spotting whether a given energy function can support interpretation as a classical probabilistic circuit.

As mentioned above the one-in-three satisfiability problem has as input a conjunction of boolean functions which are all of the form $R(x,y,z)$, and where some boolean variables can be repeated,

$$f(x_1,\ldots,x_n) = \bigwedge_{j=1}^{r} R(a_i,b_i,c_i), \tag{82}$$

where $a_i, b_i, c_i \in \{x_1,\ldots,x_n\}$ and none of the a_i, b_i, and c_i are the same variable for fixed i. Let $X_i = \{a_j | a_j = x_i\} \cup \{b_j | b_j = x_i\} \cup \{c_j | c_j = x_i\}$ denote the set of a,b,c variables that are x_i. We will define an energy function on $3r + \sum_{i=1}^{n} |X_i|$ spins which, when converted into the problem of trying to assign consistent inputs and outputs will yield exactly the boolean function f of Eq. (82). Call the first $3r$ spins S_i and the last $X_{tot} = \sum_{i=1}^{n} |X_i|$ spins T_i. First define the energy function

$$E_M(S_1,\ldots,S_{3r}) = \sum_{j=1}^{r} E_f(S_{3j-2}, S_{3j-1}, S_{3j}), \tag{83}$$

where E_f is the fan-out energy function defined in Eq. (82). When one converts this into a logical expression for whether the spins are inputs or outputs, one gets exactly the one-in-three logical clause of Eq. (81).

However, at this point, the variables that will arise in these clauses are all different. This is what the other X_{tot} spins are for. Define the following energy function for the "identity" gate:

$$E_I^{(k)}(P_1,P_2,\ldots,P_k,Q_1,Q_2,\ldots,Q_k) = \begin{cases} 0 & \text{if } P_1 = Q_1, P_2 = Q_2, \ldots, P_k = Q_k \\ \Delta & \text{otherwise} \end{cases}. \tag{84}$$

This energy function corresponds to a logical reversible gate which acts as identity from the P_i spins to the Q_i spins. This energy function has the property that the P_i variables are all either inputs or all outputs (and the Q_i variables are also all either inputs or outputs, opposite to the P_i variable assignment.) Thus we can use gates of this form to effectively make the spins of the E_M gates constructed above the same variable. In particular define the following energy function

$$E_e(S_1,\ldots,S_{3r},T_1,\ldots,T_{X_{tot}}) = \sum_{i=1}^{n} E_I^{(|X_i|)}(S_{X_i[1]}, S_{X_i[2]}, \ldots, S_{X_i[|X_i|]}, T_{X_t^{(i)}}, T_{X_t^{(i)}+1}, \ldots, T_{X_t^{(i)}+|X_i|}), \tag{85}$$

where $X_t^{(i)} = \sum_{j<i} |X_i|$ and $X_i[j]$ is the jth element of X_i translated over to a S_i variable, i.e. a_i corresponds to S_{3i-2}, b_i corresponds to S_{3i-1} and c_i corresponds to S_{3i}.

Now consider the energy function

$$E(S_1,\ldots,S_{3r},T_1,\ldots,T_{X_{tot}}) = E_e(S_1,\ldots,S_{3r},T_1,\ldots,T_{X_{tot}})) + E_M(S_1\ldots,S_{3r}), \tag{86}$$

and stipulate that every term in the sum of E_e will correspond to one gate and every term in E_M will correspond to one gate. Then if we convert this problem over into a boolean satisfaction problem as describe above, where each spin is a boolean variable representing whether the spin is an input or an output, we will obtain a boolean expression that is equivalent to the problem described in Eq. (82). Thus we have shown that we can, for a given one-in-three satisfaction problem, Eq. (82) construct an energy function, along with labeling of the terms in this function corresponding to different gates, such that determining whether this energy function can be reinterpreted as a probabilistic computation is equivalent to the original one-in-three satisfaction problem. This implies that the problem of determining whether such probabilistic computations exist is at least as hard as this NP-complete problem. Combined with the fact the problem is in NP this means that the problem of deciding whether a given energy function can be thought of as enacting a probabilistic circuit is NP-complete.

Thus we have shown that the problem of determining a many-spin statistical mechanics system enacts a ground state spin computation, even given the decomposition of this system into logical gates, is NP-complete. We note in passing that this result is of perhaps some philosophical significance: determining whether a system can be thought of as a computer is shown to be computationally intractable (assuming $\mathsf{P} \neq \mathsf{NP}$).

B. Models that have a Probabilistic Gate Interpretation

So far we have considered spin models in a very general setting unrelated to physical theories. We point out, here, however, that there are a variety of models that have arisen in physics which admit such an interpretation. We have already seen that that one dimensional Ising model admits an interpretation as a series of identity gates which err with some probability. Here we point out some other models which admit such interpretations.

1. Ising Models on Trees

A tree is a connected graph, $T = (V, E)$, with vertex set V, edge set E, and no cycles. Assign to each spin a vertex, S_v for $v \in V$. To each edge $e = (v, w)$ assign an non-zero energy for the Ising coupling, $J : E \to \mathbb{R} - \{0\}$. Then the Ising model on the tree has an energy function given by

$$E(s_1, \ldots, s_{|V|}) = \sum_{(v,w)\in E} J_{(v,w)} S_v S_w. \tag{87}$$

Fix a root vertex $v_r \in V$. First consider the model at zero temperature. At $T = 0$, following our discussion of the fan-out in Section V, it is easy to see that the ground state of this model can be thought of as a series of fan-out gates originating from the root vertex v_r with possible bit flips applied to the fan-out depending on the sign of $J_{(v,w)}$. At $T \neq 0$ we see that, as in the discussion of the fan-out gate, each of the individual fan-out wires (with possible bit flips), will fail independently with a probability of

$$p_{(S_v,S_w)} = \frac{e^{\beta J_{(v,w)}}}{e^{\beta J_{(v,w)}} + e^{-\beta J_{(v,w)}}}. \tag{88}$$

Note that a failure for an antiferromagnetic coupling (bit flip) corresponds to an identity gate. Ising models on trees have been studied in a variety of settings [34, 35] and in fact the interpretation of this model in terms of a probabilistic circuit has been used extensively in the computer science literature, where the model goes under the name of *broadcasting* [12].

2. Z_2 Lattice Pure Gauge Theories

Perhaps slightly more interesting that Ising models on trees is the fact that Z_2 lattice gauge theories [36, 37] can be cast as a probabilistic gate circuits. For definiteness, first consider a (pure) Z_2 lattice gauge theory on a two dimensional square lattice with closed boundaries. In this model one places spins on the edges of the lattice and defines the energy function

$$E = \sum_{p\in P} \sum_{S_1,S_2,S_3,S_4\in\delta p} J_p S_1 S_2 S_3 S_4 \tag{89}$$

where P is the set of all plaquettes of the square lattice, and δp are the spins surrounding plaquette p. Consider an individual term in this sum, and assume for simplicity that for all plaquettes, $p \in P$ that $J_p = J < 0$. We can view the individual term $JS_1S_2S_3S_4$ as a gate element with either 0, 1, 2, or 3 inputs and 4, 3, 2, and 1 outputs respectively. For example if we view it as a gate with 2 inputs and 2 outputs, then, in its ground state this corresponds to a probabilistic gate which, with equal probability, changes the input to a set of spins with the same parity of -1s as the input. In other words the logical evolution is

$$\begin{aligned}
&(+1,+1) \to 50\%\ (+1,+1) \text{ and } 50\%\ (-1,-1)\\
&(+1,-1) \to 50\%\ (+1,-1) \text{ and } 50\%\ (-1,+1)\\
&(-1,+1) \to 50\%\ (-1,+1) \text{ and } 50\%\ (-1,+1)\\
&(-1,-1) \to 50\%\ (+1,+1) \text{ and } 50\%\ (-1,-1).
\end{aligned} \tag{90}$$

The gate with 3 inputs and 1 outputs is deterministic and is given by $f(S_1, S_2, S_3) = S_1S_2S_3$. The gate with 0 inputs corresponds to a preparation of 4 spins which have an even number of -1 spins and this state is prepared with equal probability. The gate with 1 input and 3 outputs is a probabilistic gate which outputs an even number of -1s if the input is $+1$ all such possibilities occurring with equal probability. Similarly it outputs an odd number of -1s if

the input is -1 all such possibilities occurring with equal probability. By picking boundary terms to serve as input and output bits, it is clear that it is possible to order the energy terms in the above sum in such a way that we can interpret the ground state as a probabilistic cellular automata like model using the above gates. At finite gate these probabilistic gates all fail with some probability, where failure results in an equally likely failed output for the gate. While we have define this for the two dimensional square lattice, it is clear that the above construction can result in a probabilistic circuit for a far greater number of lattices, the main criteria being that there exists a way to label inputs and outputs in such a way that a proper circuit is constructed.

VI. MAKING CLASSICAL GROUND STATE SPIN COMPUTATION ROBUST AT FINITE TEMPERATURE

Having shown how the computed circuit energy functions give rise to thermal ensembles which can be interpreted as probabilistic circuits with probabilistic inputs, we now discuss how it is possible to make the model of classical ground state spin computation fault-tolerant at finite temperature. Indeed this is now rather simple given that we have a mapping between the behavior of these systems at finite temperature and probabilistic circuits which have gates and preparations failing with a fixed probability. We will need, however, to use the more general class of energy functions considered in Section V.

The general theory of computing in the presence of gates which fail goes back at least as far as the work of von Neumann [8]. Von Neumann considered a model in which each logic gate failed with exactly the probability ϵ. In order to get this model to compute reliably, von Neumann used two major techniques. The first techniques is that one needs to suitably encode information: in order to do this one encodes a single bit across a bundle of wires in a logic circuit and interprets this as 0 (or 1) when a majority of the wires are in the 0 (or 1.) Given such an encoding it is then easy to compute in a parallel manner such that the effect of gates failing is simply to decrease the proportion of 0s (or 1s) in an encoded 0 (or 1.) However, the fact that the gates fail does cause the ratio of correct bits in an encoded bundle of bits to decrease. In order to deal with this von Neumann used a second technique whereby faulty gates were used to perform error correction. Actually this portion of von Neumann's analysis is not quite satisfactory, as it requires the use of a random permutation. However Pippenger [38] has shown that one can de-randomize this construction (one can obtain reasonable parameters for this method by using the expanders defined in [39]). The end result of this is that one can show that if one wishes to compute a logical circuit of size n to an accuracy $1-\delta$ (that is, have the circuit fail with probability δ) with gates that fail at $\epsilon < \epsilon_t$ for some fixed threshold ϵ_t, one can do this using a circuit with the unreliable gates which is only a $O(\log^k \frac{1}{\delta})$ larger, where k is a fixed constant. This effectively means that one can create for all effective purposes reliable logic gates out unreliable logic gates with an overhead which is logarithmic in the inverse of the error desired.

Having described von Neumann's construction one can now see how it is possible to make classical ground state spin computing fault-tolerant. In particular we have shown above how the thermal ensemble for such systems gives rise to probabilistic gates. For gates constructed using energy functions like Eq. (1) we have shown that the thermal ensemble arising for these functions is equivalent to a circuit in which gates fail with probability $\frac{(2^k-1)e^{-b}}{1+(2^k-1)e^{-b}}$ if the gate has k output bits. For $k=1$ this is $\frac{e^{-b}}{1+e^{-b}}$. We would like to use this for von Neumann's ϵ failure probability. One complication arises, however, which is that in von Neumann's model the fan-out gates do not fail. These gates are needed during the error correcting stage of his procedure. We get around this by using fan-out gates constructed from energy functions like those in Section V. In particular define the fan-out gate via Eq. (77). As shown in Section V this gives rise in our probabilistic gate setting in which each fan-out resulting wire fails with probability $\frac{e^{-b}}{e^{-b}+e^{b}}$. These independent failures can then be reinterpreted as errors for the gates that follow them in von Neumann's error correcting construction. One should note that there is a subtlety here in that if we desire to recover von Neuman's error model where the gates fail with exactly probability ϵ then a complicated calculation is needed in order to figure out ϵ in this model. However, for large enough β the effect will mostly be that the error probabilities add such that ϵ will be the sum of the fan-out error and the gate error.

Putting this together we can conclude that there is a critical temperature below which one can use fault-tolerant constructions to design ground state spin energy functions which compute correctly with probability $1-\delta$ by using $O(\log^k \frac{1}{\delta})$ more interactions in the energy functions. This means that while the general model of ground state spin computing is not tolerant to errors, for a cleverly designed setup one can overcome this difficulty and build robust devices, assuming that the temperature is low enough to keep the error probability below the error threshold.

VII. RELATIONSHIP TO COMPUTATIONAL COMPLEXITY THEORY

Now we turn to a relationship between our model and computational complexity. This connection is provided because the ground state spin computing model at zero temperature is intimately related to the Cook-Levin theorem from computational complexity.

A. The Cook-Levin Theorem

The Cook-Levin theorem [22, 23] is a fundamental result in the theory of computational complexity. Here we briefly review this result. For more details one is referred, for example, to the classic textbook [40].

Traditionally the theory of computational complexity is founded upon the computational model of a Turing machine [41]. The Church-Turing thesis roughly states that the Turing machine model adequately captures the notion of what can be performed by any physical instantiation of a computer and decades of experience have yet to challenge this assumption. A deterministic Turing machine consists of the machine and a linear tape onto which elements from a finite alphabet, Γ, can be written into different cells on the tape. The machine itself has an finite set of internal states, Q, a method for reading a symbol written in a cell on the tape, a method to write a symbol in a cell on the tape, and the ability to move either left or right one cell. Formally the tape is assumed to be infinite or at least infinitely extendable. Initially an input, consisting of a string of symbols from a finite alphabet $\Sigma \subseteq \Gamma$ is placed on the tape and the physical machine is placed on the first symbol of this string. The Turing machine is assumed to start itself with its internal states in a special state called the start state, $q_0 \in Q$. A deterministic Turing machine then proceeds to "compute" as follows. At each time step the deterministic Turing machine examines the symbol of the cell it currently occupies and its internal state. Then conditional on these two variables the Turing machine writes a symbol onto cell (possibly from a larger, but still finite, alphabet than the alphabet of the initial strings) it occupies, moves either left or right, and modifies its internal state. These rules can be specified by a function, the transition function, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, where L and R denote moving left or right. The deterministic Turing machine does this indefinitely until it reaches one of a set of its internal states known as halt states. These halt states are labeled as either "reject" or "accept." If we initially input the string w and the deterministic Turing machine halts in accept state, then we say the deterministic Turing machine accepts w. Similarly if on input w the machine halts in a reject state, then we say that the machine rejects w. Note that on some strings the Turing machine may neither accept or reject a string, but continue indefinitely. The language of a Turing machine is the set of all strings taken from the input alphabet for which the Turing machine accepts for that string being input.

Deterministic Turing machines capture many important features of real modern computers. For example, the running time of most algorithms on a variety of hardware are closely related to the similar problems cast in terms of deterministic Turing machines. An important class of languages are those for which a deterministic Turing machine accepts a string from the language in a time bounded by a constant times a polynomial in the size of the string. Such languages are said to be in the complexity class P.

In contrast to a deterministic Turing machine, non-deterministic Turing machines are a model of computing which is not thought to properly encapsulate what can be efficiently computed on a modern computer. A non-determinisitic Turing machine works in a manner similar to a deterministic Turing machine: it has an internal state, it can read and write symbols onto a tape, etc. The difference is that at each step in the computation, a non-deterministic Turing machine doesn't have to chose one new state, new symbol and direction to move, but may branch to multiple such triples. The transition function is now not $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ but instead is of the form $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$ where P denotes the power set. Non-deterministic Turing machines thus explore multiple pathways at once. Such a Turing machine halts when one of the states of any of the multiple branches it is exploring enters a halt state. Again we can define whether the initial input is accepted or rejected depending on whether the state where the computation halts is a reject or accept state. Given this we can define the language of the non-determinstic Turing machine as the set of strings in which the machine accepts the given input. The time complexity of a non-deterministic Turing machine is the number of steps of executed by the Turing machine irrespective of the number of branches explored by the machine.

Because a non-deterministic Turing machine is allowed to explore multiple pathways in its computation it is not clear that such machines can be reasonably mapped (in terms of say, time complexity of problems) to modern computers. This question can be put on a more formal footing by defining the complexity class NP. NP is the class of languages which can be recognized by a non-deterministic Turing machine in a time bounded by a constant times a polynomial in the size of the string being decided. The statement of whether NP is strictly larger than P is the famous $\mathsf{P} = \mathsf{NP}$ problem, one of the most important open problems in the theory of computational complexity. That being said, most researchers do believe that $\mathsf{P} \neq \mathsf{NP}$ and that the model of a non-deterministic Turing machine is more powerful than computers we can physically construct.

Variable	Represents
$L_{i,j,k}$	True if tape cell $-p(n) \leq i \leq p(n)$ contains symbol j at time $0 \leq k \leq p(n)$
H_{ik}	True if the Turing Machine is at tape cell $-p(n) \leq i \leq p(n)$ at time $0 \leq k \leq p(n)$
S_{qk}	True if Turing machine is in state q at time $0 \leq k \leq p(n)$

TABLE II. Variables in the Cook-Levin theorem.

Given the above brief overview of the basic models of deterministic and non-deterministic Turing machines we can now introduce the Cook-Levin theorem. To define the Cook-Levin problem we introduce the SAT problem.

> **Boolean Satisfaction Problem** (SAT): Given as input a description of a Boolean function from n boolean inputs to one output, $f : \{0,1\}^n \to \{0,1\}$, decide whether there exists a set of inputs $x_1, x_2, \ldots, x_n$, with each $x_i \in \{0,1\}$, such that $f(x_1, x_2, \ldots, x_n) = 1$.

Cast in terms of languages, the SAT language is the set of strings describing a Boolean function such that the Boolean function has some input which evaluates to 1. The Cook-Levin theorem then states that the SAT is complete for NP. This means that every language in NP can be converted by a deterministic Turing machine in polynomial time to a SAT language and every SAT language can be converted into a language in NP. Thus, in a real sense, SAT captures what it means for a lanuage to be in NP. For example, if we could efficiently solve a SAT problem in polynomial time we would be able to solve every problem in NP in polynomial time and vice versa. This is especially important considering that numerous problems beside SAT have been identified as NP-complete. This class of problems are essentially "the most difficult" in NP and are correspondingly the problems for which no known polynomial time algorithms for all instances of the problems in the class are thought to exist.

We are most interested in the proof of the Cook-Levin theorem. Before reviewing this it is useful to provide another equivalent characterization of NP which uses, instead of a non-deterministic Turing machine, a prover and a verifier. The prover is assumed to have unlimited computational power while the verifier is limited to running a polynomial time algorithm on a deterministic Turing machine. A language, L, is in NP if the prover can supply to a verifier a polynomial sized proof p that a instance x is in the language. This proof must always be correct, i.e. the verifier must always be convinced, by using a Turing machine from P, that the instance x is the in the language. Similarly the prover must never be able to convince the verifier incorrectly. In other words if x is not in L, then the verifier must always reject. For example SAT is in NP by this definition since the prover can simply supply a satisfying input to the boolean function if the function can be satisfied, and the verifier can compute the boolean function on this input in polynomial time. Similarly if the function does not admit a satisfying assignment the verifier can never be convinced that the instance is not in the language because no string the prover can give will lead to a satisfied boolean function. Note that in the prover-verifier setting we do not require that the prover try to convince the verifier that an instance x is not in L, only that if it is true that x is not in L that the prover cannot convince the verifier that x is in L.

Given the prover/verifier definition of NP we can now discuss the Cook-Levin theorem. The Cook-Levin theorem states that SAT is NP-complete. A problem, P, is NP-complete if and only if every problem in NP can be transformed into P by a deterministic Turing machine in polynomial time. Thus SAT, since it is NP-complete, captures much of what is difficult about problems in all of NP: if there were a polynomial time algorithm for SAT then there would be a polynomial time algorithm for every problem in NP.

The proof that SAT is NP-complete has two directions. We have already stated the first direction, that SAT is a NP problem. The meat of the proof is in the other direction, that every problem in NP can be converted into a SAT problem. Here we do this in a slightly non-standard manner, using the prover-verifier definition of NP. In particular for any problem, L, in NP we know that there exists a verifier who takes as input the problem instance, x and a certificate p, runs these on a deterministic Turing machine and verifies whether the instance is in the language. Imagine now the time history of such a verification by a Turing machine. Initially the Turing machine is at the beginning of the string, in its initial state, and the input string is on the tape. At the next time step, the machine can change its state, move left, right or stay in place, and possibly change the symbol on the tape where the machine is located. Notice, however that the contents of a cell on the tape, whether the machine is at that location, and the internal state of the machine if it is on that cell at the next time step is a function only of the contents of the cell and its neighbors at the prior step, whether the machine was on one of those cells, and its internal state if it was. In other words the computation can be described *locally* in space and time. Given this we can now describe how to convert a verifier, which is just a deterministic Turing machine and the strings representing the instance x and the proof certificate p into a problem in SAT.

Suppose that the verifiers deterministic Turing machine runs in time $p(n)$ where $n = |x| + |p|$ is the size of the instance and the proof and p is a polynomial function. Define the following boolean variables, $L_{i,j,k}$, $H_{i,k}$, S_{qk} to represent the history of the deterministic Turing machine in verifying the proof. We think about the first two variables

as being arranged in a large $O(p(n)) \times (p(n)+1)$ tableau onto which the computational history of the deterministic Turing machine is carried out. We can now define a SAT problem over these variables which is satisfiable iff the deterministic Turing machine accepts the input string x, p. This SAT formula is conjunction (logical and) of a clauses which enforce (a) that the initial contents of the tape are x, (b) that the machine is in the correct initial state, (c) the initial location of the head of the tape, (d) that there is only one symbol per tape cell, one state per time, one tape position per time, (e) that the tape cell is unchanged unless it is the one which can be written upon, (f) the machine properly updates according to the Turing machine transition function, and (g) the machine finishes in an accepting state. For example condition (b) consists of clauses with a single variable $S_{q_0,0}$ where q_0 is the initial state of the Turing machine. The most complicated clause in this construction is the one that encodes the local update rule for the machine, (f). This can be done by adding the clauses

$$(H_{i,k} \wedge S_{q,k} \wedge L_{i,j,k}) \rightarrow (H_{i+d,k+1} \wedge S_{q',k+1} \wedge L_{i,j',k+1}) \tag{91}$$

for valid transitions of the Turing machine (that is for q, q', j, j', d representing a valid change of state, symbol, and direction of movement for the read/write head of the Turing machine.) Putting this together one thus sees that once can take the deterministic verifier along with the instance and x and construct a polynomial sized boolean formula f which is satisfied iff the verifier accepts the proof on the instance. Note that we have not forced a constraint on the proof input to the circuit, but that the definition of NP in a prover-verifier setting leads to the satisfiability of the formula f iff the instance x is in the language.

We can now see that connection between the Cook-Levin theorem and the classical ground state spin computing model. In particular a central part of the proof of this theorem is that one can construct a satisfiability formula directly from the history of a computation. In a similar manner classical ground state spin computing constructs an energy function whose ground state is the history of a computation. The conversion of a logic circuit to such an energy function is essentially the arithmetization technique of computational complexity [42, 43]. Note however that the important role here of the input and outputs to the constructed formula: in the case where the output is forced, we are led to NP-complete problems, but when the input is forced then we are led to efficient constructions. Motivated by this it is interesting to consider how the above constructions works but now at finite temperature.

B. A Promise-MA Problem From Classical Ground State Spin Computing

We will now show that it is possible to use classical ground state spin computing to define, in analogy with $3-$SAT a **Promise-**MA-complete problem. MA stands for Merlin-Arthur and is essentially a probabilistic generalization of the complexity class NP [20, 21]. It is defined as the follows:

> **Promise-MA**: A language $L = L_{yes} \cup L_{no} \subset \Sigma^*$, $L_{yes} \cap L_{no} = \emptyset$, is a promise problem in Merlin-Arthur (**Promise-**MA) if there is a probabilistic polynomial-time Turing machine V and a polynomial p such that for all strings $x \in L$,
>
> - If $x \in L_{yes}$ then there exists a w such that $|w| = poly(|x|)$ and $V(x, w)$ accepts with probability at least $2/3$.
> - If $x \in L_{no}$ then for all w such that $|w| = poly(|x|)$ then $V(x, w)$ accepts with probability at most $1/3$.

Here Σ is the alphabet over which the language is defined. This version of the definition of MA is a promise problem, since $L_{yes} \cup L_{no}$ does not necessarily cover all possible elements of Σ^*. The reason that this class is called Merlin-Arthur is because it can be defined in terms of an interactive proof system between a computationally unbounded prover, Merlin, and a more limited verifier, Arthur, who can perform feasible probabilistic computations (polynomial time probabilistic computations with bounded error, BPP.) Merlin is trying to convince Arthur that a certain instance x is in a language L_{yes}. He does this by communicating a polynomial size proof w to Arthur. Arthur can then take this proof and run a polynomial time probabilistic computation on this proof the result of which is: if $x \in L_{yes}$, then Arthur is, with high probability, convinced the x is in L_{yes}, while if $x \in L_{no}$ no matter what proof Merlin supplies Arthur cannot be convinced that $x \in L_{yes}$ except with a small probability.

We define the following problem:

> **Boundary Expectation Value (BEV)**: Let k be a positive integer and β a k digit number. Suppose we are given an energy function on n bits, $E(s_1, \ldots, s_n)$ which can be written as a sum of energy functions
>
> $$E(s) = \sum_l E_l(s) \tag{92}$$

in such a way that this energy function can be interpreted as a circuit energy function (see Eq. (1)) with $j > 0$ input bits and 1 output bit, and each term in E_l is specified with k bits of precision. Consider the thermal ensemble generated by this energy function at inverse temperature β. We promise that either there exists input bits $i_1, i_2, \ldots, i_j$ and output bit t such that for the reduced probability distribution on these bits

$$Pr(i_1, i_2, \ldots, i_j, t = 1) > \frac{2}{3} \tag{93}$$

or for any input bits $i_1, i_2, \ldots, i_j$,

$$Pr(i_1, i_2, \ldots, i_j, t = 1) < \frac{1}{3} \tag{94}$$

The problem **BEV** is to determine whether the first of these conditions, Eq. (93), holds. In other words yes instances of this problem satisfy Eq. (93) for some inputs $i_1, \ldots, i_j$ and no instances satisfy Eq. (94) for all $i_1, \ldots, i_j$.

We claim that **BEV** is **Promise-**MA-complete. The proof of this is fairly straightforward given our prior discussion of the Cook-Levin theorem.

First we claim that **BEV** is in **Promise-**MA. To show this we must show that there exists an interactive proof of the form described above for the problem **BEV**. To see this note that a computationally unbounded Merlin can sample from the thermal ensemble for an instance of this problem and can check whether there exists an input $i_1, \ldots, i_j$ such that Eq. (93) holds or whether for all inputs $i_1, \ldots, i_j$, Eq. (94) holds. This will then serve as the proof, w, in the definition of MA. Given the description of β and the circuit energy function, Arthur can construct a probabilistic Turing machine which implements the circuit related to this energy function using our construction of a probabilistic circuit from Section III. Then Arthur can run this probabilistic Turing machine on the proof that Merlin supplies. It is clear that if x is a yes instance of **BEV**, then the proof w Merlin supplies will convince Arthur that he has a yes instance of this interactive protocol. Further if x is a no instance of **BEV** it is also clear that no proof that Merlin supplies will convince Arthur that x is a yes instance. Thus it is clear that **BEV** is in **Promise-**MA.

The other direction of the proof requires that we show that every problem in **Promise-**MA can be converted into a **BEV** problem. To do this we follow the idea of the Cook-Levin theorem in that, for a MAproblem $x \in L_{yes}$, we convert the probabilistic Turing machine V that verifies the proof w of the MAproblem into a thermal circuit for which the associated **BEV** problem has (x, w) as the satisfying input. Similarly, for an MAproblem $x \in L_{no}$, the thermal circuit corresponding to V must correspond to a no instance of **BEV**, since otherwise there would exist inputs $w = i_1, i_2, \ldots, i_j$ which would cause $V(x, w)$ to accept with probability greater than 1/3. Clearly to do this we will make use of our probabilistic circuit representation result from Section III. The main tool that we need in such a construction is that we need to be able to construct circuit energy functions which perform probabilistic gates of a form we desire, for a fixed inverse temperature β. To do this we can proceed as follows. We can use our fault-tolerant constructions to produce circuits which operate with near deterministic behavior. Then in order to augment a deterministic computation we need a source of randomness, indeed in most standard definitions of a probabilistic Turing machine one requires randomness which is close to uniformly random across the random bits being used (or in other words one requires bits which are nearly equal probability of being in 0 or 1.) To obtain such randomness in our constructions, notice that if we do not force our inputs to be 0 or 1 then there is, in a thermal ensemble, an equal probability to be 0 and 1. This allows one to create random bits in an circuit energy function. However in order to get this to work with the deterministic part of our Turing machine, we must find a method to get this randomness into *encoded* bits in our circuits. This can be accomplished by simply taking the equal probability 0 and 1 bits and running them through the circuit which performs error correction on the bundle corresponding to these bits. Thus we see that we can construct at finite β and circuit energy function which is near deterministic and which can also have input bits which are nearly uniformly random. Thus we can proceed just as in the Cook-Levin theorem. For a tableau describing the history of the probabilistic Turing machine, V, (when this probablistic Turing machine is simply a deterministic Turing machine aided by bits of randomness, we can construct an energy circuit function which implements the probabilistic Turing machine spatially across this tableau.

Thus we have shown that **BEV** is **Promise-**MA-complete. While the proof of this result was rather straightforward and followed quickly from understanding the Cook-Levin theorem it is interesting to note that very few MA-complete problems are known. In fact the only other non-natural problem which is **Promise-**MA-complete to our knowledge is the stoquastic 6-SATproblem [44]. The nearest comparable result is the classic result for the complexity of finding the ground state and computing the partition function for the Ising model of Barahona [45].

VIII. QUANTUM MODELS

Finally we would like to briefly mention the quantum models which show some similarity with our classical model. In particular these models were the inspiration for investigating this problem and thus this serves as a good location to introduce the open problems concerning these problem.

One of the quantum models relevant to this discussion are universal adiabatic quantum computing schemes. In these models one shows how to adiabatically drag a many-body quantum system from one easily preparable ground state to the ground state of another many-body quantum system whose ground state is a superposition over the history of a quantum circuit. Thus, similar to our models, the end result is a ground state which encodes a computation: but in this case the computation is a superposition over the history of the computation. These models show that adiabatic quantum computing, which previously had been only used for optimization problems [46–48] can also be used for universal quantum computation [15–19]. However the final state of these models are systems with small energy gaps and thus the effects of working at non-zero temperature will have a profound effect on these models. Indeed it is for exactly this reason that such universal adiabatic quantum computing schemes are not known to be fault-tolerant (see, however, [49].) Similar reasoning follows for an earlier model of ground state quantum computing due to Mizel and co-workers [13, 14].

The problem of having a system whose ground state correctly computes but whose excited states do not is exactly the problem we have addressed in this paper for the classical ground state spin computing. An interesting question then, when considering the quantum models, is whether there is a similar reinterpretation quantum thermal ensembles as spatially enacted quantum computations. This is an important open problem which might conceivably lead to fault-tolerant methods for adiabatic quantum computing.

IX. CONCLUSION

We have introduce a set of energy functions on many bits (spins) which have the property that their ground state can be thought of as a spatially distributed deterministic computation. At temperature greater than zero we have shown that the thermal ensemble arising from these models can be reinterpreted as a spatially distributed probabilistic computation. Further, above zero temperature we see that the gates of a ground state spin computer can become unreliable and fail to execute the desired computation with high-fidelity. However with the mapping of the thermal ensemble to probabilistic circuits we have shown how it is possible to make versions of the desired deterministic circuits which are fault-tolerant. We have shown that determining whether a given classical energy function can be thought of as enacting a ground state computation is NP-complete. Finally we have shown that a problem concerning the thermal ensembles arising in our model give rise to a rare complete problem for the complexity class **Promise-**MA. The models we have considered here are mostly devoid of connections to actual physical systems of interest, besides connections to quantum-dot cellular automata. An interesting an important open question about these models is whether they arise in naturally occurring physical systems, or a suitably engineered system. Another interesting question is the rate at which the ground state spin model thermalizes: a proof that the system reaches thermal equilibrium in a time polynomial in the size of the circuit being implement would be another step towards making this model more physically relevant.

ACKNOWLEDGMENTS

D.B. and I.C. are supported by NSF grants 0621621, 0803478, 0829937, and 091640 and DARPA grant FA9550-09-1-0044. K.R.B is supported by Georgia Tech. We acknowledge useful conversations with David Meyer and Maxwell Pierce.

[1] R. Feynman, "There's plenty of room at the bottom: an invitation to open up a new field of physics," Engineering and Science (California Institute of Technology), **23**, 22–36 (1960).
[2] Laszlo B. Kish, "End of Moore's law: thermal (noise) death of integration in micro and nano electronics," Phys. Lett. A, **305**, 144–149 (2002).
[3] R. Feynman, "Quantum mechanical computers," Optics News, 11–21 (1985).
[4] P. Benioff, "Quantum mechanical hamiltonian models of turing machines," J. Stat. Phys., **29**, 515–546 (1982).
[5] P. Benioff, "Quantum mechanical hamiltonian models of turing machines that dissipate no energy," Phys. Rev. Lett., **48**, 1581–1585 (1982).

[6] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," Proc. Roy. Soc. London Ser. A, **400**, 97–117 (1985).

[7] P. W. Shor, "Algorithms for quantum computation: Discrete log and factoring," in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, Los Alamitos, CA, 1994) pp. 124–134.

[8] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies* (Princeton University Press, Princeton, NJ, 1956) pp. 329–378.

[9] C.S. Lent, P.D. Tougaw, W. Porod, and G.H. Bernstein, "Quantum cellular automata," Nanotechnology, **4**, 49–57 (1993).

[10] C.S. Lent, P.D. Tougaw, and W. Porod, "Quantum cellular automata: the physics of computing with arrays of quantum dot molecules," in *Proceedings of 1994 Conference on Physics and Computation* (IEEE Computer Society Press, 1994) pp. 5–13.

[11] Y. Wang and M. Lieberman, "Thermodynamic behavior of molecular-scale quantum-dot cellular automata (qca) wires and logic devices," IEEE Transactions on Nanotechnology, **3**, 368–276 (2004).

[12] William Evans, Claire Kenyon, Yuval Peres, and Leonard J. Schulman, "Broadcasting on trees and the Ising model," The Annals of Applied Probability, **10**, 410–433 (2000), ISSN 10505164.

[13] A. Mizel, M. W. Mitchell, and M. L. Cohen, "Scaling considerations in ground-state quantum computation," Phys. Rev. A, **65**, 022315 (2002).

[14] A. Mizel, "Mimicking time evolution within a quantum ground state: ground-state quantum computation, cloning, and teleportation," Phys. Rev. A, **70** (2004).

[15] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," in *45th Annual IEEE Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, CA, USA, 2004) pp. 42–51.

[16] A. Mizel, D. A. Lidar, and M. W. Mitchell, "Simple proof of equivalence between adiabatic quantum computation and the circuit model," Phys. Rev. Lett., 070502 (2007), arXiv:quant-ph/0609067.

[17] J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local hamiltonian problem," SIAM Journal of Computing, **35**, 1070–1097 (2006).

[18] R. Oliveira and B.M. Terhal, "The complexity of quantum spin systems on a two-dimensional square lattice," Quantum Inform. Compu., **8**, 0900–0924 (2008).

[19] J. D. Biamonte and P. J. Love, "Realizable hamiltonians for universal adiabatic quantum computers," Phys. Rev. A, **78**, 012352 (2008).

[20] L. Babai, "Trading group theory for randomness," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (ACM Press, 1985) pp. 421–429.

[21] S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (ACM Press, 1986) pp. 59–68.

[22] S. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing* (1971) p. 151.

[23] B. A. Trakhtenbrot, "A survey of russian approaches to perebor (brute-force searches) algorithms," IEEE Annals of the History of Computing, **6**, 384–400 (1984).

[24] D. Bacon and S. T. Flammia, "Adiabatic gate teleportation," Phys. Rev. Lett., **103**, 120504 (2009).

[25] D. Bacon and S. T. Flammia, "Adiabatic cluster state quantum computing,," arXiv:0912.2098 (2010).

[26] O. Oreshkov, T. A. Brun, and D. A. Lidar, "Fault-tolerant holonomic quantum computation," Phys. Rev. Lett., **102**, 070502 (2009).

[27] O. Oreshkov, "Holonomic quantum computation in subsystems," Phys. Rev. Lett., **103**, 090502 (2009).

[28] E. Ising, "Beitrag zur theorie des ferromagnetismus," Z. Physik, **31**, 253–258 (1925).

[29] C. Ungarelli, S. Francaviglia, M. Macucci, and G. Iannaccone, "Thermal behavior of quantum cellular automaton wires," Journal of Applied Physics, **87**, 7320–7325 (2000).

[30] T. P Eggarter, "Cayley trees, the Ising problem, and the thermodynamic limit," Phys. Rev. B, **9**, 2989–2992 (1974).

[31] Igor L. Markov and Yaoyun Shi, "Simulating quantum computation by contracting tensor networks," SIAM Journal on Computing, **38**, 963–981 (2008).

[32] H. Wu and D. Sprung, "Three-dimensional simulation of quantum cellular automata and zero dimensional approximation," J. Appl. Phys, 4000–4005 (84).

[33] Thomas J. Schaefer, "The complexity of satisfiability problems," in *Proceedings of the tenth annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 1978) pp. 216–226.

[34] R. Lyons, "The Ising model and percolation on trees and tree-like graphs," Comm. Math. Phys., **125**, 337–353 (1989).

[35] C. R. Viteri, Y. Tomita, and K. R. Brown, "Monte carlo analysis of critical phenomenon of the Ising model on memory stabilizer structures," Phys. Rev. A, **80**, 042313 (2009).

[36] R. Balian, J. M. Drouffe, and C. Itzykson, "Gauge fields on a lattice. ii. gauge-invariant Ising model," Phys. Rev. D, **11**, 2098–2103 (1975).

[37] Michael Creutz, Laurence Jacobs, and Claudio Rebbi, "Experiments with a gauge-invariant Ising system," Phys. Rev. Lett., **42**, 1390–1393 (1979).

[38] N. Pippenger, "On networks of noisy gates," in *Proc. of the 26th Annual Symposium on Foundations of Computer Science* (1985) pp. 30–38.

[39] A Lubotzky, R Phillips, and P Sarnak, "Explicit expanders and the ramanujan conjectures," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 1986) pp. 240–246, ISBN 0-89791-193-8.

[40] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. (Course Technology, 2005).
[41] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," Proc. Lond. Math. Soc. 2, **42**, 230–265 (1936).
[42] L. Babai and L. Fortnow, "Arithmetization: A new method in structural complexity theory," Compuational Complexity, **1**, 41–66 (1991).
[43] Adi Shamir, "IP = PSPACE," J. ACM, **39**, 869–877 (1992), ISSN 0004-5411.
[44] Sergey Bravyi and Barbara Terhal, "Complexity of stoquastic frustration-free hamiltonians," SIAM Journal on Computing, **39**, 1462–1485 (2009).
[45] F. Barahona, "On the computational complexity of Ising spin glass models," J. Phys. A: Math. Gen., **15**, 3241–3253 (1982).
[46] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," (2000), arXiv:quant-ph/0001106.
[47] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," Science, **292**, 472–475 (2001).
[48] A.M. Childs, E. Farhi, J. Goldstone, and S. Gutmann, "Finding cliques by quantum adiabatic evolution," Quantum Inform. Compu., **2**, 181 (2002).
[49] D. Lidar, "Towards fault tolerant adiabatic quantum computation," Phys. Rev. Lett., **100**, 160506 (2008).