Joint and Marginal Probability Analyses Of Markov Random Field Networks for Digital Logic Circuits

Jahanzeb Anwer, Usman Khalid, Narinderjit Singh, Nor H. Hamid and Vijanth S. Asirvadam
Electrical & Electronics Engineering Department
Universiti Teknologi PETRONAS
Bander Seri Iskander, Tronoh, Perak, Malaysia
Contact No: +60136057683
jahanzebanwer.utp@gmail.com

***Abstract*— With the device scaling up to nano-level, the integrated circuits are expected to face high computing error rates. This increased rate is the outcome of random and dynamic noise injected in the circuit which becomes more vulnerable due to low supply voltages and extremely small transistor dimensions. Markov Random Field (MRF) modelling is one approach to achieve noise-tolerance in integrated circuit design. As a general overview of fault-tolerance, we start with comparing on-going techniques for fault-tolerant design. Later, we explain the two basic terminologies of MRF i.e. Joint and Marginal Probability followed by their computation for M3 module of C432 Interrupt Controller (as our test circuit). The contribution of this paper is the derivation of circuit design rules based on the conclusions obtained by these two probability analyses.**

Keywords: Probabilistic computation, markov random field, joint probability, marginal probability, belief propagation algorithm.

I. Introduction

Nanoscale electronic circuits are suffering from both manufacturing defects and transient faults. Designing reliable system based on these devices is becoming harder. Hence, the need for defect- or fault-tolerant architecture is gaining importance amongst researchers. Several techniques, both practical and theoretical, to implement such architectures were investigated including the three major approaches discussed below.

Redundancy is the basic approach to design a fault-tolerant circuit model [1]. The idea of this technique is to introduce redundancy for each gate in the circuit (or for that portion of the circuit probable of being in error) and then taking the output from the majority output decision of the original and copied gates so that if one gate in the redundant combination is faulty, the output is not affected. This technique is further divided into Triple Modular Redundancy (TMR), Cascaded Triple Modular Redundancy (CTMR) and Triple Interwoven Redundancy (TIR).

The other significant approach is probabilistic computation [1], [2]. Here, we treat the logic levels between '0' and '1' to be attainable just like these two conventional levels and design our circuits such that the intermediate levels have minimum probability of occurrence. It is a mathematical approach in contrast to the direct hardware-based application model (i.e. redundancy). To name a few, this technique is further classified into Markov Random Field (MRF), Bayesian and Ensemble Dependent Matrix models.

Reconfiguration [1], [3] is another important approach widely in use. Again it is a direct hardware-based approach. Defect-tolerance is achieved through detection of faulty components during an initial defect map phase (defect mapping is the process of finding defective locations in the nanofabric) and excluding them during actual configuration.

Redundancy can be a possible approach to avoid manufacturing errors but it does not have robust error recovery mechanisms [1]. Similarly, reconfigurable architectures can deal with manufacturing defects but they also cannot provide tolerance for the transient faults. For transient error-tolerance, probabilistic computation is a suitable approach as the nature of noise injected in the circuit is random (or probabilistic). From its subdivisions we discussed earlier, our focus is on using MRF [2], [4]. Before starting our analysis, we briefly explain the joint and marginal probability concepts of MRF theory as understanding these terms is essential to understand the derivation of circuit design rules.

A. *Joint Probability*

The joint probability of a logic network, according to Hammersley-Clifford Theorem [5] can be written as,

$$P(X)=(1/Z)\prod_{c\in C} \exp(-Uc/kT) \qquad (1)$$

where 'X' is the set of all nodes in the neighborhood, 'C' is the set of cliques and 'U_c' is the clique energy function. The term 'Z' is called normalization constant which is required to normalize the probability function to [0, 1]. The term 'kT' is the thermal energy which controls the shape of the joint and marginal probability distribution graphs.

The system represented by MRF (as a dependence graph) can be decomposed into cliques. Since these cliques are independent of each other, we can compute joint probability of each separately. At the end, we multiply all these values to get the joint probability of the whole system. According to [6], the 'correct logic states' are those that maximize the joint probability of the overall network. (Correct logic states indicate to logic states achieved in a circuit without error). In our paper, we have provided a step by step procedure to compute and thus maximize the joint probability which will later be used to determine the correct logic states.

B. *Marginal Probability*

In calculating marginal probability, we fix the value of one or more variables and sum it over non-fixed variables. For discrete random variables, the marginal probability function [7] can be written as shown below (2).

$$P(X{=}x) = \sum_{y} P(X{=}x, Y{=}y) = \sum_{y} P(X{=}x|Y{=}y)*P(Y{=}y) \quad (2)$$

where P(X=x,Y=y) is the joint distribution of X and Y, while P(X=x|Y=y) is the conditional distribution of X given Y.

The use of this statistical term is in determining the probability of achieving different logic states at each node. From this information, we can determine the most probable logic state for any node in the network. Since the inputs of the logic circuit have defined probabilities of being in logic state '0' or '1', the intermediate and output nodes have the probabilities, we have to calculate. For this purpose, we use Pearl's belief propagation algorithm [8]. This algorithm computes the marginal probabilities of intermediate and output nodes by marginalizing each node step by step unless we reach the desired node. Another use of marginal probability plots is that they help us to observe the variation of any logic state's probability (between 0 and 1) with temperature variation.

II. Computing Joint Probability

In this section, we will see how to compute and maximize joint probability. For our analysis, we have taken a test circuit, M3 module of C432 Interrupt Controller, from [9]. Fig. 1 shows its logic diagram, dependence graph and logic compatibility function of NAND gate (which will be used later in the clique energy function calculation).

We are using equation (1) for joint probability computation. After identifying cliques [$\{x_3,x_4\}$, $\{x_2,x_4,x_5\}$, $\{x_0,x_1,x_5,x_6\}$ and $\{x_6,x_7\}$], we calculate clique energy function, U_c for each. Then we evaluate the exponential in (1) for each clique and multiply all of the exponentials at the end to get the overall joint probability. Here we outline the steps for evaluating U_c for NAND gate, as an example. Similarly, the U_c for NOT (index 1 and 2 having x_3 and x_6 as inputs respectively) and NOR gates are calculated (with reference to Fig. 1(a)) and listed in Table I.

$U_c =$ $-\sum$ (Valid minterms (f=1) in the Logic Compatibility Function (Fig 1(c)))

$= -[\, x_0'x_1'x_5'x_6 + x_0'x_1'x_5x_6 + x_0'x_1x_5'x_6 + x_0'x_1x_5x_6 + x_0x_1'x_5'x_6 + x_0x_1'x_5x_6 + x_0x_1x_5'x_6 + x_0x_1x_5x_6' \,]$

$= -[\, x_0'x_1'x_6 (x_5 + x_5') + x_0'x_1x_6 (x_5 + x_5') + x_0x_1'x_6 (x_5 + x_5') + x_0x_1x_5'x_6 + x_0x_1x_5x_6' \,]$

$= -[\, x_0'x_1'x_6 + x_0'x_1x_6 + x_0x_1'x_6 + x_0x_1x_5'x_6 + x_0x_1x_5x_6' \,]$

$= -[\, x_0'x_6 (x_1' + x_1) + x_0x_1'x_6 + x_0x_1 (x_5'x_6 + x_5x_6') \,]$

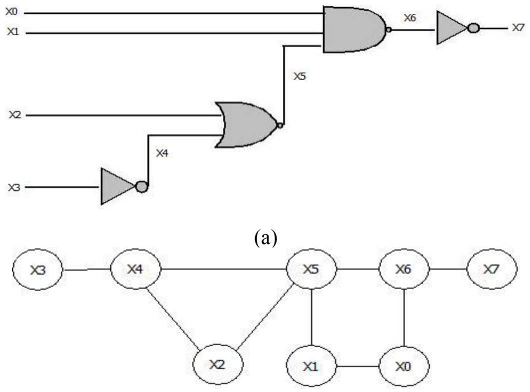
$= -[\, (1-x_0)x_6 + x_0x_6(1-x_1) + x_0x_1(x_6(1-x_5) + x_5(1-x_6)) \,]$

$= -[\, x_6 + x_0x_1x_5 - 2x_0x_1x_5x_6 \,]$

$= 2x_0x_1x_5x_6 - x_0x_1x_5 - x_6$

TABLE I
CLIQUE ENERGY FUNCTIONS FOR NOT AND NOR GATES

NOT 1	$U_c = 2x_3x_4 - x_3 - x_4$
NOR	$U_c = x_2x_4 + 2x_4x_5 + 2x_2x_5 - 2x_2x_4x_5 - x_2 - x_4 - x_5$
NOT 2	$U_c = 2x_6x_7 - x_6 - x_7$



(a)

(b)

x_0	x_1	x_5	x_6	f
0	0	0	1	1
0	0	0	0	0
0	0	1	1	1
0	0	1	0	0
0	1	0	1	1
0	1	0	0	0
0	1	1	1	1
0	1	1	0	0
1	0	0	1	1
1	0	0	0	0
1	0	1	1	1
1	0	1	0	0
1	1	0	1	1
1	1	0	0	0
1	1	1	0	1
1	1	1	1	0

(c)

Fig. 1 (a) A sample logic circuit (b) Its Dependence Graph (c) Logic Compatibility Function for NAND

Now, computing joint probability,

$$P(x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7) = (1/Z).\,(e^{-Uc(NOT\ 1)/\ kT}.\ e^{-Uc(NAND)/\ kT}.\ e^{-Uc(NOR)/\ kT}.\ e^{-Uc(NOT\ 2)/\ kT})$$

$$= (1/Z).\exp[(x_2 + x_3 + 2x_4 + x_5 + 2x_6 + x_7 - x_2x_4 - 2x_2x_5 - 2x_3x_4 - 2x_4x_5 - 2x_6x_7 + x_0x_1x_5 + 2x_2x_4x_5 - 2x_0x_1x_5x_6)/\ kT]$$

Following the joint probability calculation, we need to determine the node label combinations that maximize its value. The simplified form of $P(x_0,x_1,\ldots,x_7)$ shows that its value would be maximum when the power of the exponential will be maximum. i.e. for maximum value of numerator of the power ($x_2 + x_3 + 2x_4 + x_5 + 2x_6 + x_7 - x_2x_4 - 2x_2x_5 - 2x_3x_4 - 2x_4x_5 - 2x_6x_7 + x_0x_1x_5 + 2x_2x_4x_5 - 2x_0x_1x_5x_6$).

We used MATLAB to determine the value of this power's numerator for its 256 (=2^8) possible node combinations. We have observed that the maximum value of the numerator is '4' and it exists for 16 combinations of node labels shown in Table II. These combinations are the same as the 16 combinations of this circuit's truth table, which shows that the joint probability is maximum only for correct logic combinations. For the rest of the combinations, its value is always lower.

TABLE II
NODE COMBINATIONS HAVING MAXIMUM JOINT PROBABILITY

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	0	1	1	0	0	1	0
0	1	1	1	0	0	1	0
1	0	1	1	0	0	1	0
1	1	1	1	0	0	1	0
0	0	0	0	1	0	1	0
0	1	0	0	1	0	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0
0	0	1	0	1	0	1	0
0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0
1	1	1	0	1	0	1	0
0	0	0	1	0	1	1	0
0	1	0	1	0	1	1	0
1	0	0	1	0	1	1	0
1	1	0	1	0	1	0	1

A. *Design Principle of Joint Probability*

From the joint probability analysis, we have concluded that for the perfect logic operation of a circuit i.e. with no errors at any nodes of the circuit; we need to design our circuit, as such to ensure at all times, that the joint probability of the circuit remains maximum.

III. COMPUTING MARGINAL PROBABILITY

We will do this analysis on the same test circuit that we used for joint probability case. We assume that all the inputs are equally likely to be in logic state '0' or '1'. For computing the probability of the hidden (intermediate and output) nodes, we use belief propagation algorithm (outline provided in [10]). We would show the steps of this algorithm with implementation on our test circuit.

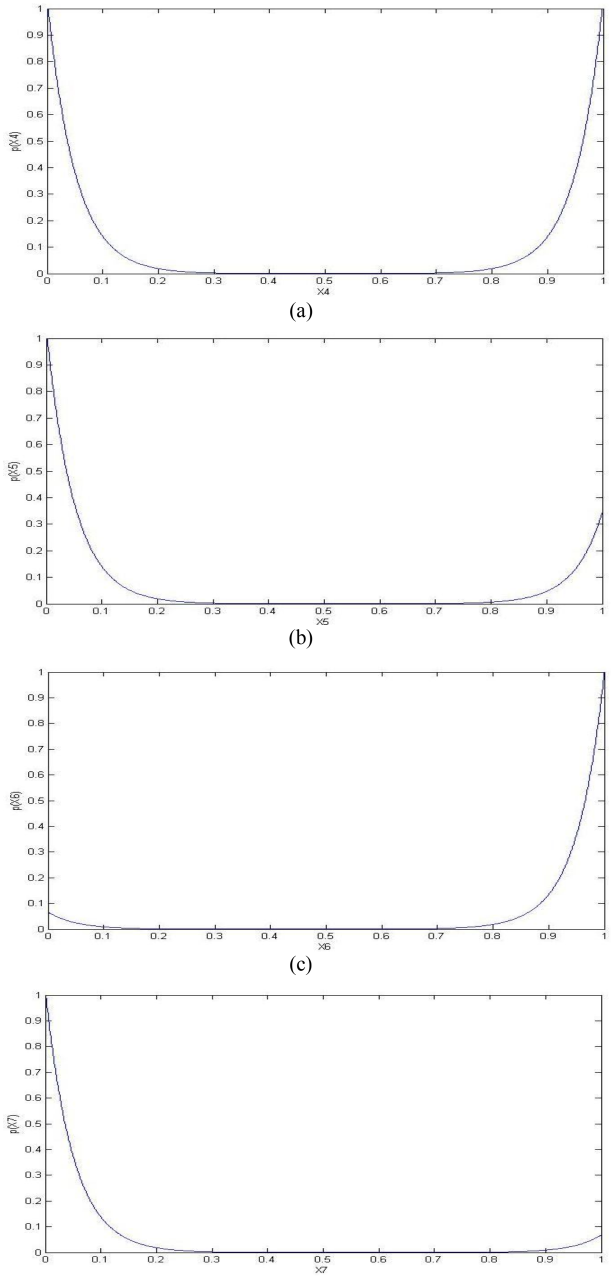
The first step is to assign Probability Distribution Functions (PDF) to all inputs and cliques as shown in Table III. In the process of computing marginal probability of output (x_7), we will be calculating the probabilities of all the intermediate nodes too (x_4, x_5 and x_6). (Note: Initially $p(x_7) = f_0\ f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7$). We will start from eliminating inputs followed by intermediate nodes until we reach the output node. In eliminating one node, two of the functions of that node eliminate and one new function forms. So, for each step, one function from $p(x_7)$ decreases unless we are left with only one function which would be dependent only on x_7. All the normalization constants (Z_n) are selected as to keep the value of the marginal probability between '0' and '1'. Re-using the clique energy functions calculated in section II, we have shown the algorithm steps in Table IV.

TABLE III
PDF FOR I/PS AND CLIQUES

Input	PDF	Clique	PDF
x_0	$f_0(s_0)$	$\{x_3,x_4\}$	$f_4(x_3,x_4)$
x_1	$f_1(s_1)$	$\{x_2,x_4,x_5\}$	$f_5(x_2,x_4,x_5)$
x_2	$f_2(s_2)$	$\{x_0,x_1,x_5,x_6\}$	$f_6(x_0,x_1,x_5,x_6)$
x_3	$f_3(s_3)$	$\{x_6,x_7\}$	$f_7(x_6,x_7)$

TABLE IV
BELIEF PROPAGATION ALGORITHM STEPS

<table>
<tr>
<td><b><u>Step 1: Eliminate x_3</u></b>
Eliminated: $f_3(s_3)$, $f_4(x_3, x_4)$
New: $f_8(x_4)$
$p(x_4) = \sum_{x_3 \in (0,1)} (1/Z_1)\, e^{-Uc\,(NOT\ 1)/\,kT}$
$=(1/Z_1)(e^{x4/kT}+e^{(1-x4)\,/kT})$
$= f_8(x_4)$
⇨ $p(x_7) = f_0\ f_1\ f_2\ f_5\ f_6\ f_7\ f_8$</td>
<td><b><u>Step 2: Eliminate x_2</u></b>
Eliminated: $f_2(s_2)$,
$f_5(x_2.x_4,x_5)$
New: $f_9(x_4,x_5)$
$p(x_5|x_4)=\sum_{x_2 \in (0,1)}(1/Z_2)\, e^{-Uc\,(NOR)\,/\,kT}$
$= (1/Z_2)(e^{(x4+x5-2x4x5)\,/kT} + e^{(1-x5)\,/kT})$
$= f_9(x_4,x_5)$
⇨ $p(x_7) = f_0\ f_1 f_6\ f_7\ f_8\ f_9$</td>
</tr>
<tr>
<td><b><u>Step 3: Eliminate x_4</u></b>
Eliminated: $f_8(x_4)$, $f_9(x_4,x_5)$
New: $f_{10}(x_5)$
$p(x_5)=\sum_{x_4 \in (0,1)}(1/Z_3)[p(x_5|x_4)^*\ p(x_4)]$
$=(1/Z_3)(e^{x5\,/kT}+3e^{(1-x5)\,/kT} + e^{(1+x5)\,/kT}\ + 3e^{(2-x5)\,/kT})$
$= f_{10}(x_5)$
⇨ $p(x_7) = f_0\ f_1 f_6\ f_7 f_{10}$</td>
<td><b><u>Step 4: Eliminate x_0</u></b>
Eliminated: $f_0(s_0)$,
$f_6(x_0,x_1,x_5,x_6)$
New: $f_{11}(x_1,x_5,x_6)$
$p(x_6|x_1,x_5) =\sum_{x_0 \in (0,1)}(1/Z_4)\, e^{-Uc\,(NAND)\,/kT}$
$= (1/Z_4)\,(e^{(x6)/\ kT} + e^{(x6+x1x5-2x1x5x6)/\ kT})$
$= f_{11}(x_1,x_5,x_6)$
⇨ $p(x_7) = f_1 f_7\ f_{10}\ f_{11}$</td>
</tr>
<tr>
<td><b><u>Step 5: Eliminate x_1</u></b>
Eliminated: $f_1(s_1)$,
$f_{11}(x_1,x_5,x_6)$
New: $f_{12}(x_5,x_6)$
$p(x_6|x_5)=\sum_{x_1 \in (0,1)}(1/Z_5)f_{11}(x_1,x_5,x_6)$
$=(1/Z_5)(e^{(x5+x6-2x5x6)/kT} + 3e^{(x6)\,/kT})$
$= f_{12}(x_5,x_6)$
⇨ $p(x_7) = f_7\ f_{10}\ f_{12}$</td>
<td><b><u>Step 6: Eliminate x_5</u></b>
Eliminated: $f_{10}(x_5)$,
$f_{12}(x_5,x_6)$
New: $f_{13}(x_6)$
$p(x_6)=\sum_{x_5 \in (0,1)}(1/Z_6)[p(x_6|x_5)^*p(x_5)]$
$= (1/Z_6)\ (28e^{(1+x6)\ /kT} + 13e^{(x6)\ /kT} +15e^{(2+x6)\ /kT} + 3e^{(1-x6)\ /kT} +4e^{(2-x6)/kT} + e^{(3-x6)\ /kT})$
$= f_{13}(x_6)$
⇨ $p(x_7) = f_7\ f_{13}$</td>
</tr>
<tr>
<td colspan="2"><b><u>Step 7: Eliminate x_6</u></b>
Eliminated: $f_7(x_6,x_7)$, $f_{13}(x_6)$
New: $f_{14}(x_7)$
$p(x_7) = \sum_{x_6 \in (0,1)} (1/Z_7)\ [\ p(x_7\,|\,x_6)\ ^*\ p(x_6)\ ]$
$= (1/Z_7)(31e^{(1+x7)/kT} +19e^{(2+x7)\ /kT}+e^{(3+x7)/kT} + 3e^{(1-x7)/kT} +17e^{(2-x7)\ /kT} + 29e^{(3-x7)\ /kT}+15e^{(4-x7)\ /kT})$
$= f_{14}(x_7)$
⇨ $p(x_7) = f_{14}$</td>
</tr>
</table>



(a)

(b)

(c)

(d)

Fig. 2 Marginal probability graphs for (a) x_4 (b) x_5 (c) x_6 (d) x_7

In Fig. 2, we have shown probability graphs for all the hidden and output nodes. The $p(x_4)$ graph shows that there is an equal probability of getting either logic state '0' or '1' whereas the $p(x_7)$ graph shows that the probability of achieving logic '0' at this node is almost sixteen times the probability for logic '1'. The probability of getting intermediate states between '0' and '1' is negligible. By this analysis, we can determine at any point in the network that whether it is more probable of being at logic state '0' or '1'. In Fig. 3, we can observe that, by increasing temperature, the graph moves upward and the probability of intermediate logic states start increasing thus making logic circuit more probable of achieving these states. And since in ideal case, the probability of intermediate states should be zero, the probability of error increases in nano-computation.

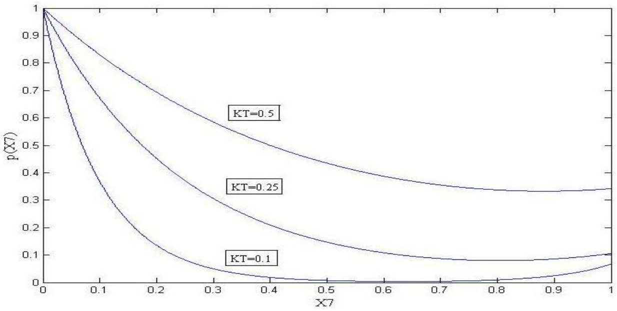


Fig. 3 Marginal probability variation for x_7

A. Design Principle of Marginal Probability

The key to design a fault-tolerant circuit is to ensure a good heat removal system for the integrated circuit which would make the probability of intermediate states between '0' and '1' close to zero and maintain sufficient noise margin as well.

Conclusions and Future Work

MRF is a design technique for nanoscale circuits that make them work in highly noisy conditions provided the design principles of joint and marginal Probability are followed. In future, we will show how to use these principles for digital hardware design.

References

[1] Sumit Ahuja, Gaurav Singh, Debayan Bhaduri and Sandeep K. Shukla, "Fault and Defect Tolerant Architectures for Nano-computing," in Bio-Inspired and Nanoscale Integrated Computing, Mary Eshaghian-Wilner, Wiley, 2009.

[2] R. I. Bahar, J. Chen and J. Mundy, "A Probabilistic-Based Design for Nanoscale Computation," in Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation, Sandeep K. Shukla and R.I. Bahar, Springer, 2004.

[3] Debayan Bhaduri and Sandeep Shukla, "Reliability Analysis of Fault-Tolerant Reconfigurable Architectures," FERMAT, Tech.rep. 2004-15, 2004.

[4] S. Z. Li, Markov Random Field Modeling in Computer Vision. Berlin: Springer -Verlag, 1995.

[5] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," Journal of the Royal Statistical Society, series B, vol. 36, No. 2, pp. 192-236, 1974.

[6] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson and A. Zaslavsky, "Designing Nanoscale Logic Circuits Based on Markov Random Fields," Journal of Electronic Testing: Theory and Applications, vol 23, pp. 255–266, Jun 2007.

[7] "A 'layman's explanation of Marginal Distribution," 2008. [Online] Available: http://en.wikipedia.org/wiki/Marginal_probability. [Accessed: Jun, 2009].

[8] J Yedidia, W. Freeman, and Y.Weiss, "Understanding belief propagation and its generalizations," in Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nebel, Morgan Kaufmann, 2003.

[9] I-Chyn Wey, You-Gang Chen, Changhong Yu, Jie Chen and An-Yeu Wu, "A 0.13μm Hardware-Efficient Probabilistic-Based Noise-Tolerant Circuit Design and Implementation with 24.5dB Noise-Immunity Improvement," in Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC-2007), Jeju, Korea, pp. 295-298, Nov 2007.

[10] Kundan Nepal, "Markov Random Field," in Designing Reliable Nanoscale Circuits Using Principles of Markov Random Fields, PhD dissertation, Brown University, RI 02912, US, 2007.