

Techniques for Designing Noise-Tolerant Multi-Level Combinational Circuits

K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, A. Zaslavsky
Brown University, Division of Engineering, Providence, RI 02912

Abstract

As CMOS technology downscales, higher noise levels, wider threshold variation, and low supply voltage will force designers to contend with high rates of soft logical errors and many defective devices. A probabilistic design framework based on Markov random fields (MRF) has been previously proposed to address dynamic fault and noise vulnerability of ultimate digital CMOS circuitry. The idea is to use additional transistors and feedback loops to achieve significant noise immunity and ensure correct logic operations at low V_{DD} . However, the extra reliability achieved in previously published work came at a cost of high transistor counts. In this paper, we present techniques to reduce the transistor count of larger multi-level combinational circuits built within the MRF framework by using variable sharing, implied dependence and supergates. Using these techniques we show an average reduction of approximately 28% in transistor counts over a range of combinational benchmark circuits built within the MRF framework compared to the best previously published results.

1. Introduction

For several decades, mainstream silicon technology has relied on scaling down CMOS transistors, with the miniaturization driven by Moore's Law and the continuously updated ITRS roadmap [1]. While CMOS devices with gate length $L_G < 10nm$ have been experimentally demonstrated [2], it is certain that shrinkage of devices to such extreme L_G , combined with the power-consumption constrained reduction of V_{DD} down to 0.5 V or even lower, will produce faulty systems. Both soft faults due to noise and signal coupling, and hard faults due to process variations and defects can be expected, making error-free logic and data retention a difficult proposition.

Probabilistic computing provides a possible approach towards building fault-tolerant nanoarchitectures and systems. Previous work has proposed the use of Markov random fields as a framework for probabilistic computation in nano-scale systems [3]. The mapping of the MRF-based probabilistic framework to ultimate CMOS circuitry was shown in [4, 5] and the approach was extended to state and memory protection against single event upsets in [6, 7]. These papers showed that subthreshold operation was viable for reliable computation at reduced dynamic power levels and with high level of noise immunity; however the improvements came with up to an order of magnitude increase in transistor count [4]. In this paper we address this transistor overhead problem for large multi-level combinational circuits by using different design strategies. The first strategy involves the creation of MRF elements using the concept of common clique variable sharing and implied dependence. The second strategy involves the creation and

use of supergates. Our results show a 28% improvement in transistor count compared to the area-optimal mapping reported in [5]. This translates to only a 3.5X overhead in transistor count in order to obtain reliable computation compared to an unreliable standard CMOS implementation.

2. Background

To better understand how our combinational circuits are designed, we first provide a brief background on Markov random fields.

2.1 Markov Random Fields

The Markov Random Field defines a set of random variables which can each take on various values and interact with other similar random variables in a finite neighborhood. Circuit networks can be expressed in terms of such neighborhoods [3] and the interaction of the logic states and variables can be represented as a dependence graph. Figure 1 shows a simple multi-level circuit and its corresponding dependence graph. In this case, the graph is equivalent to a Markov random field, where the nodes are random logic variables that can hold values ranging from 0 to V_{DD} and the edges are the conditional dependencies between the variables. Importantly, there is no notion of directed logic flow and causality, just statistical dependence. For instance, if the output of the first NAND gate is at logic 0, then both the inputs are constrained to be at logic 1 — i.e., there is a (backward) statistical dependency between the output state and the input state. This dependency between the inputs as well as the outputs is modeled with an edge connecting the nodes of the gate.

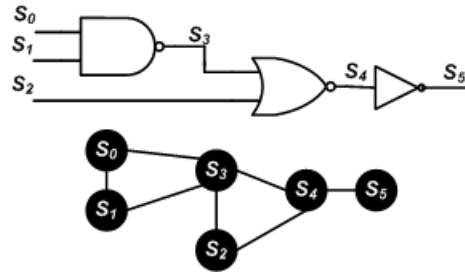


Figure 1. A logic circuit and its dependence graph for a simple Markov random field.

All the logic variables, $\{s_0, s_1, s_2, s_3, s_4, s_5\}$ in this example, are varying in a random manner over the range of the logic signal levels. The correct logic states are those that maximize their joint probability, i.e., the correct logic operation for the example corresponds to the variables that maximize, $p(s_0, s_1, s_2, s_3, s_4, s_5)$.

This work is supported in part by NSF grants CCR-0304284, CCR-0403958, and CCR-0541106.

In the graph of Figure 1, three distinct sets of *cliques* (i.e., fully connected subsets of the nodes in the graph) $\{s_0, s_1, s_3\}$, $\{s_2, s_3, s_4\}$, $\{s_4, s_5\}$ are observed. These cliques represent the local statistical dependencies of the logic states. The crucial factor for probabilistic circuit design is that the full set of nodes (logic variables) in the circuit can be factored into a product of joint probabilities in the set of cliques that describe the local interactions. Using the Hammersley Clifford theorem [8], the joint probability distribution can be written as,

$$p(S) = \frac{1}{Z} \prod_{c \in C} e^{\frac{-U(s_c)}{U_0}} \quad (1)$$

where S is the set of all nodes in the dependence graph, C is the set of cliques, s_c is the set of nodes in a clique c , $U(s_c)$ is the clique energy function also referred to as the logic compatibility function, and U_0 is an abstract term that defines the sharpness of the probability distribution. The term Z is called the *partition function* and is a constant required to normalize the probability function to $[0,1]$.

2.2 State Propagation in MRF Networks

The general algorithm for finding individual site labels that maximize the probability of the overall network is called *belief propagation* [9] and provides an efficient means of solving inference problems by propagating marginal probabilities through the network. The basic idea of belief propagation is that the probability of state labels at a given node in the network can be determined by marginalizing (summing) over the joint probabilities for the node state given just the probabilities for site labels in the neighborhood. In our example of Figure 1 the probability $p(s_0, s_1, s_2, s_3, s_4, s_5)$ can be decomposed into:

$$p(s_0, s_1, s_2, s_3, s_4, s_5) = U(s_0, s_1, s_3) U(s_2, s_3, s_4) U(s_4, s_5) \quad (2)$$

For belief propagation, we start from the primary inputs of the circuit network. As a first step, s_0 and s_1 are eliminated by summing $U(s_0, s_1, s_3)$ over all states of s_0 and s_1 to obtain $U(s_3)$, i.e., s_0 and s_1 are marginalized out. Then s_2 and s_3 can be eliminated by summing $U(s_3)U(s_2, s_3, s_4)$ over all states of s_2 and s_3 , giving $U(s_4)$. Finally, s_4 can be eliminated similarly to obtain $U(s_5)$.

This example illustrates that achieving the correct state configuration in the network corresponds to propagating state values through the network and updating each node assignment with a node state having the maximum probability.

3. Mapping Multi-level Circuits to MRF elements

In our previous works, MRF encodings for simple logic elements were shown [4, 5]. Each encoding consisted of bistable elements and feedback circuitry to keep the circuit input and outputs at their correct values. The MRF encodings provided superior noise immunity but such immunity came at the cost of larger transistor counts. These simple elements were cascaded to form larger multi-level circuits resulting in extremely high area overhead. In this section, we identify various ways of reducing the transistor counts of multi-level circuit designs by using the concept of clique variable sharing, implied dependence and supergates.

3.1 Variable Sharing and Implied Dependence

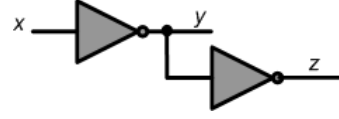


Figure 2. An inverter cascade.

Consider a cascade of two inverters as shown in Figure 2 for an application where the output of the first inverter as well as the second inverter are used elsewhere in the circuit. In such a case, the middle node y as well as the output node z need to be treated equally. One possible solution to create such a structure is to take two MRF inverter elements and cascade them together. The first inverter would implement the clique function $U_c(x, y) = x'y + xy'$ and the second inverter would implement $U_c(y, z) = yz' + y'z$. The cost of this cascaded implementation would be a total of 40 transistors.

In the cascade of gates, signal y appears as the output of the first stage and an input to the second stage. The bistable element and feedback mechanism of the first stage will try to re-enforce the value at node y . The bistable element and feedback mechanism of the second stage would do the same. In this simple example, the value at node y is being re-enforced twice. There is nothing functionally wrong with the double reinforcement — it is simply wasteful.

In order to avoid the double cost of node storage and feedback, the node y can be shared and the two clique energy functions can then be written using the shared variable y as: $U_c(x, y, z) = x'yz' + xy'z$. Hence, an MRF encoding can be built by sharing the common label y as shown in Figure 3. Here the sharing occurs both in the bistable elements as well as the feedback circuitry and hence the total transistor count is reduced. For this particular circuit structure, the total cost in terms of area is just 28 transistors — a 30% savings compared to a naïve cascade of gates.

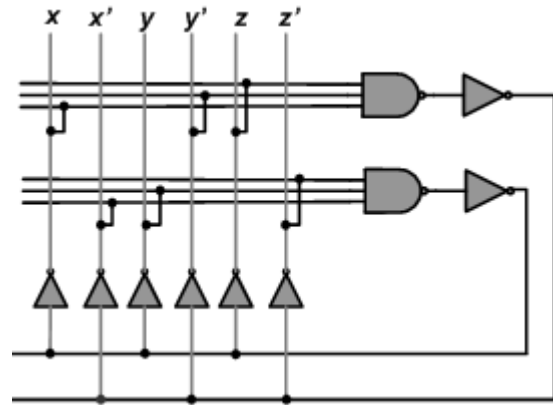


Figure 3. Clique encoding of the inverter cascade.

Sharing clique variables in both the bistable element and the feedback circuitry reduced the transistor count in this particular example. However, the possibility of sharing a node on the bistable element is constrained by the total number of transistors allowed on the bistable circuit stack. In general, an N -input/ M -output MRF element would require up to $(N + M)$ transistors in series in the transistor-level implementation of the bistable element. While larger logic functions could be realized using higher transistor stacks, for practical purposes this is generally not preferred.

When all the devices in the stack are turned on and conducting, the threshold voltage of each device increases due to the stack effect and causes the drive current to decrease. Circuit designers typically handle this problem by restricting the stack height of a design to four [10]. Higher number of clique variables could be handled using a multi-level bistable element.

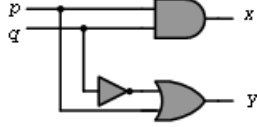


Figure 4. A circuit network with multiple outputs. p, q are the inputs and x, y are the outputs.

Often in real designs we encounter circuits that have multiple outputs. Usually these multiple outputs are all a function of the same primary inputs of the circuit. Consider the circuit shown in Figure 4 with inputs p, q and outputs x and y . The output x is defined by the logical AND of the two inputs, i.e., $x = p \cdot q$ and y is defined as $y = p + q'$. The clique energy function for these two relations can be written as:

$$U_c(p, q, x) = pqx + x'(q' + p') \quad (3)$$

$$U_c(p, q, y) = p'qy' + y(p + q') \quad (4)$$

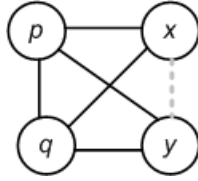


Figure 5. MRF dependence graph of the circuit from Figure 4.

The dependence graph in Figure 5 shows the relationship between the inputs and the respective outputs. The solid lines in the graph show the explicit dependence of the two separate outputs x , and y on the inputs. There is also a dashed line between outputs x and y representing the implied dependence between the two outputs. The logical state of output x is directly dependent on inputs p and q , but inputs p and q also directly depend on y . If output y were set to a 0 then inputs p and q would have to be logic 0 and 1 respectively, setting x to a logic 0. This implied dependence between all the nodes of the dependence graph adds some degree of complexity but it also confers an advantage: instead of treating these two outputs as two separate entities with two different clique functions, we can treat the entire system as one large entity governed by a fourth-order clique function consisting of nodes p, q, x and y .

$$U_c(p, q, x, y) = pqxy + p'qx'y' + q'x'y \quad (5)$$

Using this combined clique energy function, we can now create an MRF mapping for the entire circuit. The circuit encoding is shown in Figure 6. The total number of transistors required to implement this combined clique energy function is 50 compared to 84 if the individual clique energies were separately mapped. In other words, the transistor count overhead required to obtain noise

immunity dropped from 6 fold to about 3 fold relative to the circuit implemented with 14 transistors in standard CMOS.

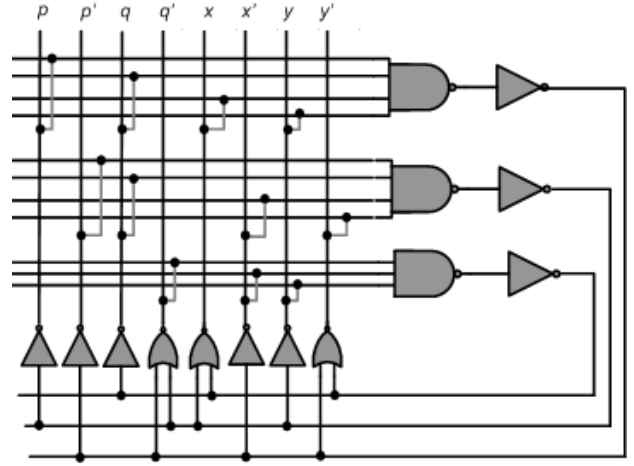


Figure 6. MRF encoding of clique energy function shown in Equation 5. p, q are the inputs and x, y are the outputs.

3.2 Supergates

The concept of supergates was introduced in [11] and its use as a preprocessing step before technology mapping was shown in [12]. In both these works, a supergate was introduced as a single-output network composed of several logic levels of basic logic elements found in a standard cell library. In our case, however, we use the term supergate to refer to both single and multiple output networks. Given a MRF circuit netlist consisting of simple logic elements, our goal is to find supergates that have the same logic functionality but lower transistor counts.

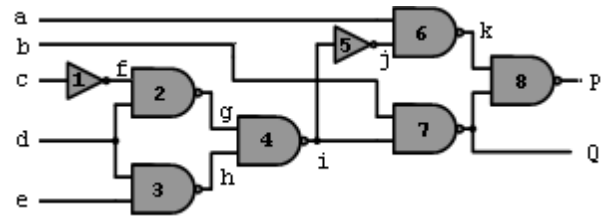


Figure 7. A circuit network.

Consider the circuit network shown in Figure 7 that implements an arbitrary function. For simplicity of illustration, the logic gates in this example are restricted to a simple inverter and two-input NAND; however these could be more complex logic gates (e.g., NAND4, AOI21, etc.). Larger multilevel circuits such as the one shown here can always be built by cascading simple MRF elements. The noise tolerance of the individual MRF element cascaded to form such multi-level elements will result in reliable signal at the primary outputs. The total cost of cascading such elements to form the network would come at a cost of 208 transistors compared to 28 in fault-susceptible standard CMOS.

We can consider various supergate structures that are functionally equivalent to the circuit network shown in Figure 7. Consider

the equivalent structure consisting of the following supergates:

$$\begin{aligned} SG_1 &= \{1, 2, 3, 4\} \\ SG_2 &= \{5, 6\} \\ SG_3 &= \{7, 8\} \end{aligned}$$

The first supergate now has three primary inputs c, d and e and produces the output i . The internal nodes f, g and h are no longer relevant. As such, one can write the clique energy function directly for the circuit, ignoring all internal nodes, as: $U_c(c, d, e, i) = id(c + e') + i'(c'd + e')$. Implementing this clique energy function directly requires a total of only 36 transistors. Similarly node j becomes irrelevant in the second supergate and the clique energy function of that supergate with respect to nodes a, i and k is $U_c(a, i, k) = ai'k' + k(i + a')$. This supergate implementation comes at a cost of 36 transistors, fewer than the 48 transistors that would have been required if the circuit were built by a cascade of gates 5 and 6. The third supergate consisting of gates 7 and 8 has two outputs. For this gate, output P is dependent on nodes $\{k, Q\}$ and Q is dependent on nodes $\{b, i\}$. Here node k is not directly dependent on node b or i but there is a direct dependence between k and P . Similarly, P and Q have a direct dependence and Q in turn is dependent on the nodes b and i . From the concept of implied dependence introduced in the preceding section, there is a dependence of Q on k . Hence, we can say that all the nodes of this supergate are implicitly or explicitly dependent on each other, enabling us to create a single clique energy function representing the five variables. The combined clique energy function is a little complex but can be written as: $U_c(b, i, k, P, Q) = kb'P'Q + ki'P'Q + k'b'PQ + k'i'PQ + biPQ'$. Implementing this clique energy function requires a total of 56 transistors. Using these three supergates, the multi-level combinational circuit shown in Figure 7 can be created using a total of 120 transistors, for a savings of 42% compared to the one mapped with cascaded simple MRF elements.

The supergate set shown in this example is not unique. For a given circuit, there exist multiple sets of supergates that can efficiently represent the original functionality. Consider the possibility of combining supergates SG_2 and SG_3 into a single supergate. This new supergate would involve gates $\{5, 6, 7, 8\}$ with inputs a, b and i and outputs P and Q . Using implied dependence, the combined clique energy function for this supergate can be written as $U_c(a, b, i, P, Q) = ab'PQ + a'b'P'Q + biPQ' + bi'P'Q$. This clique function can be implemented with a total of 66 transistors leading to the MRF implementation of the circuit of Figure 7 with only 102 transistors. This is about a 50% savings in terms of transistor count compared to a cascade of simple gates.

Note that for this particular circuit, the presence of multiple outputs on a supergate increases the total transistor count of that supergate. For instance, for the final supergate we had P and Q as the outputs. If the circuit design did not dictate the need for signal Q to be set as a primary output, we could have ignored it as an internal node and created the second supergate consisting of gates $\{5, 6, 7, 8\}$ with much fewer transistors. The absence of Q would have simplified the clique energy function to $U_c(a, b, i, P) = ab'P + a'b'P' + biP + bi'P'$. The implementation of this would have reduced the complexity of both the bistable elements and the feedback circuitry and required 20 fewer transistors.

Table 1 summarizes the area, power consumption and propagation delay of the different supergate sets for the circuit network

Supergate set	Area(# tran)	Power(μ W)	Delay(ns)
MRF cascade	208	19.6	13.4
$\{1,2,3,4\}\{5,6,7,8\}$	102	4.0	12.4
$\{1,2,3,4\}\{5,6\}\{7,8\}$	120	6.8	28.0
$\{1,2\}\{3,4\}\{5,6\}\{7,8\}$	148	9.1	27.9
$\{1,2\}\{3,4\}\{5,6,7,8\}$	130	6.0	15.4

Table 1. Area, power dissipation and propagation delay comparison at $V_{DD} = 200mV$ for different supergate sets of the circuit network shown in Figure 7.

shown in Figure 7. The supergate sets were simulated in SPICE using the 70nm predictive technology model [13] at a supply voltage of 200mV. All circuit nodes (internal and external) were subjected to noise modeled as a Gaussian distribution with mean 0V and standard deviation 60mV RMS [4]. From the table, it is clear that the supergate set of $\{1,2,3,4\}$ and $\{5,6,7,8\}$ not only requires the least number of transistors but also has the lowest propagation delay and has the lowest power consumption. Figure 8 shows the SPICE simulation of the output node P for a CMOS and MRF implementation of the circuit network. The MRF circuit implemented using the supergate set of $\{1,2,3,4\}$ and $\{5,6,7,8\}$ shows remarkable noise immunity compared to the regular CMOS implementation.

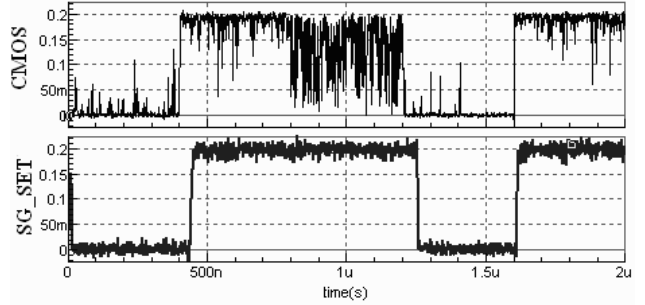


Figure 8. Voltage at output node P for a CMOS implementation and MRF implementation using supergate set $\{1,2,3,4\}\{5,6,7,8\}$.

4. Experimental Results

For a given circuit structure, there are numerous possibilities in generating the candidate supergates. Certain constraints must be imposed to keep the number of such supergate sets manageable. Constraints in the number of clique variables (i.e., the number of inputs/outputs), the total area, and the complexity of the feedback path of the MRF design were used as metrics for generating the supergates. The complexity of the feedback element depends not so much on the variable count but more on the ease of Boolean factorization of the terms in the clique energy function. If multiple minterms can be combined using factorization, as was shown in [5], the number of bistable elements can be reduced and the overall feedback path can be simplified. This in turn generates a savings in transistor counts. In this work, the number of clique variables was limited to five and the maximum number of transistors per supergate was limited to 120 to keep the feedback path complexity to a minimum. Using these constraints on the super-

Circuits	Input	Outputs	Circuit depth	Standard CMOS	w/o supergates	Overhead (X)	with supergates	Overhead (X)	% improvement
5xp1	7	10	11	532	2724	5.1	2140	4.0	21.4
9sym	9	1	15	988	5040	5.1	3940	4.0	21.8
alu4	14	8	23	5618	28472	5.1	24356	4.3	14.5
apex1	45	45	22	9838	47620	4.8	40100	4.1	15.8
apex2	39	3	26	1732	8152	4.7	6912	4.0	15.2
apex3	54	50	19	8652	42236	4.9	35500	4.1	15.9
apex4	9	19	19	13062	65416	5.0	51442	3.9	21.4
apex5	117	88	14	4876	22220	4.6	18432	3.8	17.0
b12	15	9	8	372	1816	4.9	1376	3.7	24.2
bw	5	28	9	748	3840	5.1	2912	3.9	24.2
clip	9	5	11	714	3544	5.0	2812	3.9	20.7
cordic	23	2	13	358	1712	4.8	928	2.6	45.8
duke2	22	29	21	2590	12192	4.7	10420	4.0	14.5
e64	65	65	10	3664	15292	4.2	13276	3.6	13.2
ex1010	10	10	21	12568	63832	5.1	47550	3.8	25.5
ex5	8	63	12	3264	15824	4.8	11894	3.6	24.8
inc	7	9	11	572	2632	4.6	2124	3.7	19.3
misex1	8	7	8	276	1416	5.1	1108	4.0	21.8
misex2	25	18	7	490	2288	4.7	1920	3.9	16.1
misex3	14	14	22	5948	29236	4.9	24496	4.1	16.2
o64	130	1	8	524	2832	5.4	2732	5.2	3.5
pdv	16	40	24	5626	26032	4.6	20280	3.6	22.1
rd53	5	3	9	242	1252	5.2	800	3.3	36.1
rd73	7	3	13	656	3260	5.0	2360	3.6	27.6
rd84	8	4	13	950	4836	5.1	3188	3.4	34.1
sao2	10	4	11	682	3392	5.0	2468	3.6	27.2
seq	41	35	23	8692	42108	4.8	35936	4.1	14.7
spla	16	46	24	5816	27060	4.7	20626	3.5	23.8
squar5	5	8	7	266	1372	5.2	752	2.8	45.2
t481	16	1	14	354	1544	4.4	1172	3.3	24.1
table3	14	14	24	7678	37880	4.9	32812	4.3	13.4
table5	17	15	23	7088	34216	4.8	29408	4.1	14.1
vg2	25	8	9	828	3636	4.4	3108	3.8	14.5
Z5xp1	7	10	10	798	3972	5.0	2344	2.9	39.2
Z9sym	9	1	12	794	3856	4.9	2760	3.5	28.4
C432	36	7	29	828	3424	4.1	2584	3.1	24.5
C499	41	32	23	1900	10672	5.6	3712	2.0	65.2
C880	60	26	23	1476	7764	5.3	5108	3.5	34.2
C1355	41	32	23	1964	10672	5.4	2632	1.3	75.3
C1908	33	25	31	1900	10176	5.4	3772	2.0	62.9
C2670	233	140	22	2976	14984	5.0	9292	3.1	38.0
C3540	50	22	41	4612	23388	5.1	16644	3.6	28.8
C5315	178	123	41	7138	35348	5.0	21220	3.0	40.0
C6288	32	32	119	9948	54828	5.5	25188	2.5	54.1
C7552	207	108	34	9220	47996	5.2	16904	1.8	64.8
AVERAGE						4.9		3.5	28.2

Table 2. Comparison of transistor count for MRF mapping with and without the use of Supergates

gate complexity, a total of 657 gates (including the basic logic elements) were generated and used for synthesis.

We now present results obtained from experiments run on the MCNC and ISCAS combinational benchmark suites synthesized and mapped with and without the use of supergates. In Table 2, we show result for 45 different circuits with varying depth and size. All circuits were synthesized and mapped to gates using the ABC tool [14]. In columns 2–4 we list the number of inputs, outputs and circuit depth for each circuit. The depth reported in Table 2 is the maximum number of gates along any path from primary input to output using a standard CMOS mapping. Column 5 reports the transistor counts for the circuits mapped using a standard CMOS implementation, while columns 6 and 8 report transistor counts for the circuits mapped to MRF logic elements without and with the use of supergates, respectively. To obtain the results reported in column 8, each circuit was synthesized once and then put

through 10 iterations of mapping, where each iteration produced a functionally equivalent circuit using a different set of supergates chosen by the ABC tool. The mapping generated by the supergate set that resulted in the least area measured in terms of transistor counts was reported in the table.

The MRF implementation without supergates led to a 4.9 fold area overhead on average over a standard CMOS implementation. This is similar to what was reported in [5]. By using supergates during the mapping process, this area overhead was reduced to 3.5X on average relative to the standard CMOS implementation (see column 9). This translates to a 28% improvement in transistor count overhead by being able to exploit the supergates in the MRF mapping. Depending on the particular circuit, supergates may be used quite effectively to reduce the overall transistor count. In the best case, circuit C1355 required only a 30% overhead compared to a standard CMOS implementation (i.e., a 75% reduction

over the area-optimal method of [5]). This is a very modest overhead especially considering a standard CMOS implementation of C1355 is unlikely to produce reliable signal outputs under such noisy conditions, as reported in [4]. While area reduction is not always so impressive (e.g., in the worst case, the overall reduction in transistor count was only 3.5% for circuit *o64*), we do show that for 8 of the 47 circuits, the transistor counts were reduced by at least 40% compared to an MRF mapping without supergates. Still, the extent of area reduction greatly depends on the overall circuit structure and the availability of certain supergate structures in the library. As future work, enhancements to this tool may include selective MRF mapping directed in part by circuit structure to further reduce the area overhead.

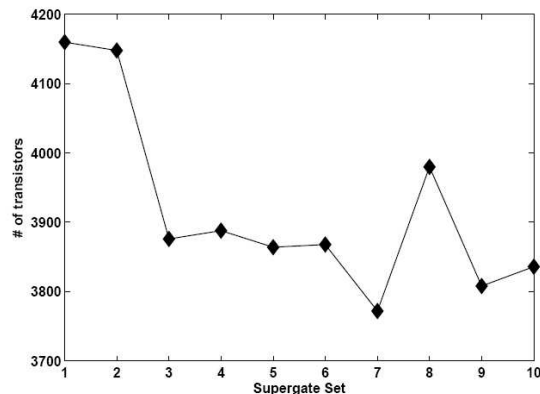


Figure 9. Transistor count variation for 10 different supergate sets for the C1908 benchmark

The supergate set used in the mapping of a particular benchmark circuit is not unique, as illustrated in Section 3.2. Across the 10 mapping iterations, the total number of transistors can vary depending on which supergate set was chosen by the ABC tool. To get a better appreciation of this effect, in Figure 9 we show a typical example of the variation in transistor counts across the 10 mapping iterations. For this particular circuit (C1908), the variation ranged from a maximum of 4160 transistors to a minimum of 3772. Across all benchmark circuits, an average of 7% difference in transistor count was seen between the maximum and minimum mapping. While certain constraints described earlier in this section were employed to keep the number of supergates available during technology mapping to a manageable number, the choice of an optimal supergate set during the technology mapping stage is a complex problem. In this work, a number of iterations were done and the mapping producing the least area was reported. However, as was shown in Section 3.2, the optimal supergate set in terms of area might not necessarily produce the optimal mapping in terms of delay or power. Our current work focuses on developing an efficient heuristic for determining the optimal supergate set in terms of reliability, area, delay and power.

5. Conclusions and Future work

As devices are sized down to the nanoscale and supply voltage scales down below 0.5V, circuit designers will need to account for significant signal noise in order to guarantee reliable computation. The MRF probabilistic model provides a framework for designing CMOS circuits that can operate effectively under conditions of ultra-low supply voltage and extreme noise conditions. In

this paper we showed how the area overhead for an MRF implementation might be reduced by using implied dependence, supergates and clique variable sharing. A complex logic function (with multiple inputs and/or outputs) may have a smaller overhead with respect to transistor count if implemented using supergates, however, tradeoffs in delay and power may not make this an optimal design choice. Our aim in the future is to develop a noise-aware logic synthesis and technology mapping tool. Given a functional description of a circuit, the tool will produce an error-tolerant design that balances area, power, delay, and reliability constraints when generating the final mapped circuit.

6. References

- [1] The latest update to the ITRS is available at <http://www.public.itrs.net>.
- [2] B. Doris *et al.* Extreme scaling with ultra-thin SOI channel MOSFETs. In *Technical Digest IEDM*, pages 267–270, 2002.
- [3] R. I. Bahar, J. Mundy, and J. Chen. A probabilistic-based design methodology for nanoscale computation. In *Proceedings of International Conference on Computer Aided Design*, Nov. 2003.
- [4] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky. Designing logic circuits for probabilistic computation in the presence of noise. In *Proceedings of Design Automation Conference*, June 2005.
- [5] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky. Optimizing noise-immune nanoscale circuits using principles of Markov random fields. In *Proceedings of Great Lakes Symposium on VLSI*, April 2006.
- [6] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky. MRF Reinforcer: A Probabilistic Element for Space Redundancy in Nanoscale Circuits. *IEEE Micro*, 26(5):19–27, Sept-Oct 2006.
- [7] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, and A. Zaslavsky. Designing MRF based error correcting circuits for memory elements. In *Proceedings of Design, Automation and Test in Europe*, March 2006.
- [8] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(3):192–236, 1994.
- [9] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *International Joint Conference on AI*, 2001. Distinguished Lecture.
- [10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 2003.
- [11] S.C. Seth, L. Pan, and V. D. Agrawal. PREDICT — probabilistic estimation of digital circuit testability. In *Proceedings of IEEE International Symposium on Fault Tolerant Computing*, pages 220–225, June 1985.
- [12] A. Mishchenko, X. Wang, and T. Kam. A new enhanced constructive decomposition and mapping algorithm. In *Proceedings of Design Automation Conference*, pages 143–148, June 2003.
- [13] Available at <http://www-device.eecs.berkeley.edu/~ptm/>.
- [14] Berkeley Logic Synthesis and Verification Group. ABC: A system for sequential synthesis and verification, release 51205. <http://www.eecs.berkeley.edu/~alanmi/abc/>.