DYNAMIC STABILITY MARGIN ANALYSIS ON SRAM

A Thesis

by

YENPO HO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2008

Major Subject: Electrical Engineering

DYNAMIC STABILITY MARGIN ANALYSIS ON SRAM

A Thesis

by

YENPO HO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Garng M. Huang
Committee Members,	Peng Li
	Takis Zourntos
	N. Sivakumar
Head of Department,	Costas Georghiades

May 2008

Major Subject: Electrical Engineering

ABSTRACT

Dynamic Stability Margin Analysis on SRAM.

(May 2008)

Yenpo Ho, B.S., University of Arizona

Chair of Advisory Committee: Dr. Garng Huang

In the past decade, aggressive scaling of transistor feature size has been a primary force driving higher Static Random Access Memory (SRAM) integration density. Due to the scaling, nanometer SRAM designs are getting more and more stability issues. The traditional way of analyzing stability is the Static Noise Margins (SNM). However, SNM has limited capability to capture critical nonlinearity, so it becomes incapable of characterizing the key dynamics of SRAM operations with induced soft-error. This thesis defines new stability margin metrics using a system-theoretic approach. Nonlinear system theories will be applied rigorously in this work to construct new stability concepts. Based on the phase portrait analysis, soft-error can be explained using bifurcation theory. The state flipping requires a minimum noise current ($I_{critical}$) and time ($T_{critical}$). This work derives $I_{critical}$ analytically for simple L1 model and provides design insight using a level one circuit model, and also provides numerical algorithms on both $I_{critical}$ and $T_{critial}$ for higher a level device model. This stability analysis provides more physical characterization of SRAM noise tolerance property; thus has potential to provide needed yield estimation.

DEDICATION

To *My parents, Jii-Chung Ho and Li-li Hu*

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my academic and research advisor, Dr. Garng Huang. Without his generous and constant guidance, I would not have able been to finish my program successfully. My family and I are greatly indebted to him.

I would also like to thank Dr. P. Li, Dr. T. Zourntos and Dr. N. Sivakumar for serving as my committee members and providing me valuable information.

Furthermore, I would like to thank my family for their constant support and encouragement. I would like to dedicate this work to my father to whom I owe everything. Lastly, I would like to thank all the people around me for their friendship and help. They made my years in College Station memorable ones.

TABLE OF CONTENTS

		Page
ABSTRACT		iii
DEDICATION		iv
ACKNOWLEDGEMENTS		v
TABLE OF CONTENTS		vi
LIST OF FIGURES		viii
LIST OF TABLES		xi
CHAPTER		
I	INTRODUCTION	1
II	SYSTEM MODELING ON SRAM	7
	2.1 How Does the SRAM Work	7
	2.2 Level-1 Spice Model	10
	2.3 Definition of S(x) and Modified Set of Current Equation	12
	2.4 SRAM Cell Modeling Equations	14
III	NONLINEAR SYSTEM THEORIES AND CELL DYNAMICS OF SRAM	17
	3.1 Equilibrium Points of General Nonlinear Systems	17
	3.2 Jacobian Matrix	18
	3.3 Qualitative Behavior of Linearized Systems	21
	3.4 System Stability Theories and Concepts	26
	3.5 Nullclines on Phase Portrait	32
	3.6 Noise Induced Bifurcation	34
	3.7 SRAM Cell Dynamics	37
	3.8 Nonlinearity of Separatrix	40
IV	ANALYTICAL SOLUTIONS FOR DYNAMICAL NOISE MARGIN ANALYSIS	50

CHAPTER Page

4.1 Introduction on Defied Regions 50
4.2 The Physical Interpretation of Regions 55
4.3 Stable Nodal Equilibria and Unstable Saddle 59
4.4 The Separatrix Equation 70
4.5 Movement of Equilibria When Perturbed by Noise 77
4.6 Analytical Solution of the Bifurcation Point and Critical Current . 110

V NUMERICAL ALGORITHMS AND IMPLEMENTATIONS 120

5.1 Numerical Implementation on Critical Current 120
5.2 Numerical Implementation on Critical Time 126

VI CONCLUSIONS AND FUTURE WORKS 130

6.1 Conclusions and Contributions 130
6.2 Recommended Future Works 130

REFERENCES 132

APPENDIX A 137

APPENDIX B 163

VITA 175

LIST OF FIGURES

FIGURE Page

1-1 Design-for-Manufacturability—Near-Term Years. 2

2-1 A Cell of SRAM. 7

2-2 (a) The Current Flow at the Left Inverter When Performing Read Operation. (b) The Current Flow at Right Inverter When Performing Read Operation. 8

2-3 (a) The Current Flow at the Left Inverter When Performing Write Operation. (b) The Current Flow at the Right Inverter When Performing Write Operation. 9

2-4 The Plot of Smooth Version of S(x) With A=100 (Left) A=50 (Right). ... 14

2-5 The 4-T SRAM Circuit Diagram 14

3-1 Phase Portrait of (a) Stable Node; (b) Unstable Node............................... 22

3-2 Phase Portrait for (a) Stable Focus; (b) Unstable Focus 23

3-3 Phase Portrait for a Saddle 24

3-4 Example of 65nm Technology SRAM Phase Portrait 25

3-5 The Linearization Plot for (a) Left-Top Equilibrium Point; (b) Center Point; (c) Right-Bottom Equilibrium Point. 26

3-6 An Example of Phase Portrait for an SRAM 29

3-7 Separatrix Plot of a 65nm SRAM Cell.. 32

3-8 SRAM Current Labels.. 33

3-9 An Example of Nullclines on 65nm Technology SRAM 34

3-10 The Example of Noise Bifurcation. Noise Level Are (a)0 A; (b)200 uA; (c)300 uA; (d)450 uA; (e)500 uA; (f)600 uA 36

FIGURE Page

3-11 (a) Phase Portrait of SRAM When I_{noise} is 500 µA; (b) The Timing Diagram of This Cell 38

3-12 (a) Phase Portrait and (b) Time Diagram for a Square Pulse Noise of 500µA Amplitude and 0.43ns Duration. 40

3-13 (Left) V_{dd} Decrease By 10%; (Right) V_{dd} Increase By 10% 41

3-14 (a) V_{th1} and V_{th3} Increase By 33%; (b)V_{th1} and V_{th3} Decrease By 33%; (c) K1 and K3 Increase By 20%; (d) K1 and K3 Decrease By 20% 42

3-15 Example of 0th Order Effect on Symmetrical Change on Multiple Pairs of Parameters. 43

3-16 Example of Separatrix (a) Linearly Shift Down; (b) Linearly Shift Up; When n=33 44

3-17 Example of Separatrix (a) Linearly Shift Down; (b) Linearly Shift Up; When n=90 44

3-18 A Single Parameter Change on (a) Threshold (V_{th1} Increase By 33%); (b) Capacitance (C_1 Increase by 5%); (c) More Than One Changes on Threshold Voltages. 46

3-19 Varying Only V_{th} Values When n is (a) 60; (b) 80 48

3-20 Varying Only K Values When n is (a) 20; (b) 40; (c) 60; (d) 80. 48

3-21 Varying V_{th} and K Values at n=70. 49

4-1 The Region Plot. 51

4-2 The Region Plot When All Threshold Voltages Are the Same. 53

4-3 Dark Regions Would Not Have Influence By the Term $S^2(V_{dd}-V_2-V_{th1})$. 54

4-4 Region 2 Plot and Its 6 Inequalities. 57

4-5 Three Equilibrium Points Sit in Region 1 2 and 3. 69

4-6 The 1st Order Effect Separatrix. 71

FIGURE Page

4-7 Actual Separatrix Vs. Linear-Separatrix 72

4-8 The Layout of Case I. 77

4-9 (Left) Positive Current Injection in Case I. (Right) Negative Current Extraction in Case I. 89

4-10 (LEFT) Simulated Equilibrium Point Movement When Is Running Positive I_{noise} From Zero to Positive $I_{critical}$; (RIGHT) Simulated Equilibrium Point Movement When Running I_{noise} From Zero To Negative $I_{critical}$. 90

4-11 The Layout of Case II.. 90

4-12 (Left) Positive Injection in Case II. (Right) Negative Injection in Case II 99

4-13 The Possible Regions to Have Bifurcation for Case I............................... 100

4-14 Output of Maple Software. 103

4-15 Equilibria in Region 9A When I_{noise}=4.7e-4(A). 106

4-16 (Left) Phase Portrait of a Symmetric Cell. (Right) Saddle Node Bifurcation as I_{noise} Increases. 112

4-17 The Bifurcation Point for 65nm SRAM. 113

4-18 The Plot of V_2f, V_2g and V_2h. 115

5-1 Newton Raphson's Algorithm Flow-Chart 122

5-2 The Bifurcation Location Plot and the Numerical Result on Positive I_c ... 123

5-3 The Bifurcation Location Plot and the Numerical Result on Negative I_c .. 124

5-4 The Bifurcation Location Plot and the Numerical Result on Positive I_c ... 125

5-5 The Bifurcation Location Plot and the Numerical Result on Negative I_c. . 125

5-6 (LEFT) Trajectory for an Initial State (V_{dd},0) Running for 17.5ns; and Its (RIGHT) Timing Diagram. 127

LIST OF TABLES

TABLE Page

1 L1-Spice Model of NMOS and PMOS Equations 11

2 A Different Representations for the Level-1 Model 11

3 Topological Classification of Hyperbolic Equilibra. 25

4 Physical Interpretation of Regions. 56

5 The Data of Analytical I_c and Real I_c Under Various Perturbed Parameters. 117

6 Few Sample Data for T_c on Different Set of Perturbed Parameters. 129

CHAPTER I

INTRODUCTION

Nowadays, the study of Static Random Access Memory (SRAM) design task becomes essential. The SRAM provides indispensable on-chip data storage for an extremely wide variety of electronic applications including microprocessor, ASICs, FPGAs, DSPs…etc. [1-9]. In today's technology devices, the silicon area occupied by SRAM caches is dominating over other logic devices. About more than 70% of chip area is used by SRAM in well known processors; and SRAM is expected to occupy more than 90% of silicon real estate in the future [10-11].

In the past decade, aggressive scaling of transistor feature size has been a primary force driven higher SRAM integration density. On the other hand, the supply voltage is scaled down to meet device reliability constraints and to reduce power consumption. However, the stability margin of SRAM has been significantly suffered by such aggressive scaling. [1-5] [11-21].

At the same time, nanometer SRAM designs are getting more and more susceptible to various noises. The state of SRAM would flip its state and produce soft errors by various noise injection mechanisms such as power supply noises, substrate noises and single event upsets (SEUs) [11] [18] [22-27]. Moreover, one significant SRAM design challenge in nano-scale CMOS is to ensure high parametric yield as the

---

This thesis follows the style of *IEEE Transactions on Automatic Control.*

size of SRAM arrays continues to grow. To insure low failure out of large cell array, SRAM must be designed to tolerate wide range of parametric variations, such as random dopant fluctuation (RDF), induced transistor threshold variations…etc. Hence, the impact of process variation is also crucial and can not be neglected. The process variations in highly scaled processes have inevitably introduced device parametric variations and mismatches [28]. According to the international technology roadmap for semiconductor [10], the parameter variation increases drastically as CMOS transistor device scale down. Fig. 1-1 shows some of the parameter variations for the next five years. The increasing of process variations in nanometer technologies produces significant fluctuations to all device aspect such as variations of threshold voltage, width, length, as well as the leakage currents. Because of that, the robustness evaluation and stability analysis of SRAM cells is one important design aspect.

	Year of Production	*2005*	*2006*	*2007*	*2008*	*2009*	*2010*	*2011*	*2012*	*2013*	*Driver*
	DRAM ½ Pitch (nm) (contacted)	*80*	*70*	*65*	*57*	*50*	*45*	*40*	*36*	*32*	
	Mask cost ($m) from publicly available data	**1.5**	**2.2**	**3.0**	**4.5**	**6.0**	**9.0**	**12.0**	**18.0**	**24.0**	*SOC*
	% V_{dd} Variability % variability seen at on-chip circuits	**10%**	**10%**	**10%**	**10%**	**10%**	**10%**	**10%**	**10%**	**10%**	*SOC*
	% V_{th} variability Doping Variability impact on VTH	**24%**	**29%**	**31%**	**35%**	**40%**	**40%**	**40%**	**58%**	**58%**	*SOC*
	% V_{th} variability Includes all sources	**26%**	**29%**	**33%**	**37%**	**42%**	**42%**	**42%**	**58%**	**58%**	*SOC*
IS	% CD variability CD for now, might add doping later	**12%**	**12%**	**12%**	**12%**	**12%**	**12%**	**12%**	**12%**	**12%**	*SOC*
	% circuit performance variability circuit comprising gates and wires	**41%**	**42%**	**45%**	**46%**	**49%**	**50%**	**53%**	**54%**	**57%**	*SOC*
	% circuit power variability circuit comprising gates and wires	**55%**	**55%**	**56%**	**57%**	**57%**	**58%**	**58%**	**59%**	**59%**	*SOC*

Manufacturable solutions exist, and are being optimized
Manufacturable solutions are known
Interim solutions are known
Manufacturable solutions are NOT known

Fig.1-1 Design-for-Manufacturability—Near-term Years

The traditional way of analyzing stability is the Static Noise Margins (SNM) [12] [29-30]. In the sense of Static Noise Margin, it presents the maximum amplitude of the voltage or the current perturbation on sensitive circuit nodes which the circuit can tolerate. However, static analysis has not considered an important fact that not all the injected noise pattern would cause state to flip. In other words, SNM has limited capability to capture critical nonlinear dynamic phenomenon. Not surprisingly, SNM may become incapable to characterize the key dynamics of SRAM operations with induced soft-error. For this reason, the dynamic noise margin (DNM) model has been proposed [31]. In order to capture critical nonlinearity, dynamic noise margin is proposed to take into account the spectral and time-dependent properties of the specific noise patterns described above. [32]

This thesis focuses on nonlinear dynamic behavior analysis using a system-theoretic approach and saddle node bifurcation of nonlinear dynamical equations [33-36]. Beginning with Chapter II, we first introduce the SRAM circuit and the read write process, and then we construct nonlinear differential equations base on Level-1 model for SRAM device.

In Chapter III, we introduce the needed background for our system theoretic approach. We introduce the fundamental concept of equilibrium point, phase portrait analysis, and the basic stability, stability region and stability boundary concept. Based on these fundamentals, a method to trace the stability boundary of SRAM (separatrix) [32] [37] can be easily applied to SRAM analysis that saves computation when compared to the brute force approach. During the operation of a SRAM cell, if a perturbed transient state trajectory passes across the stability boundary, the state will flip. For a perfectly symmetric SRAM, the stability boundary is a 45 degree line that passes through the origin. [31] However, stability boundary will become distorted and curved when SRAM becomes asymmetrical due to parameter perturbations. [32] Thus, the noise margin depends on the nonlinear stability boundary. Based on the phase portrait analysis and the developed separatrix for SRAM, we can explain why SRAM has lower current bound and minimum time requirement ($I_{critical}$ and $T_{critical}$) to flip a state using bifurcation theory. From the phase portrait analysis, when injected current amplitude reaches $I_{critical}$, we observed that two equilibria collide and result in a saddle-node bifurcation [34-36] [38]. The collision location is called the bifurcation point. When this happens, the two colliding equilibria disappear, and only the other remaining stable equilibrium point will survive. When I_{noise} is less than $I_{critical}$, the states will never cross the separatrix, so when the noise disappears, the states of the cell will return to its stable equilibrium point. However, I_{noise} being greater than $I_{critical}$ does not necessarily imply that the cell will flip its state [31]. I_{noise} must be greater than $I_{critical}$ for a certain period of time (defined as $T_{critical}$) to cross the original separatrix. Once the state of the cell crosses the separatrix, the state will flip even after the noise disappears. However, it is still not clear how the SRAM parameters physically influenced the phenomena observed from phase portrait analysis. Accordingly, we resort to analytical form solutions to find the relations.

In chapter IV, we use rigorous theorems to derive the stability margin analytically for the level-one transistor model. First of all, we partition the state space into 23 regions. We derive that three equilibria are located in three different regions. Then we prove the equilibria are two stable equilibria and a saddle. Focusing on the saddle, we derive separatrix equation analytically. Moreover, the equilibrium point locations in terms of a constant noise injection and system parameters are derived. From there, we prove that the saddle-node bifurcation [33-36] will only happen in certain region. Then, focus on the region of bifurcation; we derive bifurcation point and I_c analytically. However, the outcome of analytical solution on bifurcation point and I_c contain over a hundred terms. For that, we observe on the numerical property and propose a new method to derive analytical solution for I_c and that can greatly simplifies the equation but keep the accuracy. Most of cases, the error has kept within 10%, except only one or two cases has drastic error up to 27%. In chapter V, we present algorithms to find stability margin $I_{critical}$ and $T_{critial}$ for higher level models based on the fundamentals for the L-1 model.

In conclusion, this thesis defines a new dynamic stability margin (DSM) metrics for SRAM. Unlike SNM, our DSM is based on the dynamic cell behaviors and provides more physical characterizations of SRAM over the traditional based metrics. This thesis investigates DSM in an analytical form for the level-one transistor model. Although the device model is simple, this semi-analytical approach keeps the physical terms and provides much needed potential insights. The insights obtained for this simple model enable us to extend our numerical solution techniques to more general high level models, which may contain computer routines based on experimental data and thus are not differentiable function as we would like to have. In addition, it is rather complicated to obtain the analytic solutions by direct symbolic computation. Therefore, we use both numerical and symbolic computations. It turns out that a process to interact between numerical computations to keep rack on the complicated symbolic formulae that involves multiple roots for a 4^{th} order algebraic equation is needed. This summarizes the contribution of the thesis.

CHAPTER II

SYSTEM MODELING ON SRAM

2.1. HOW DOES THE SRAM WORK?

The Static Random Access Memory cell (SRAM) is often constructed by two cross-coupled inverters (labeled *Q1 Q2 Q3* and *Q4*) and two access transistors, labeled *Q5* and *Q6* from Fig. 2-1. [41] The access transistor acts as transmission gate allowing bidirectional current flow between the coupled inverters and bit line, denoted as *B*. The access transistors are turned on when the word line (denoted as *W*) is selected. In particular, the SRAM can hold their stored data indefinitely as long as the power supply provided. [39]

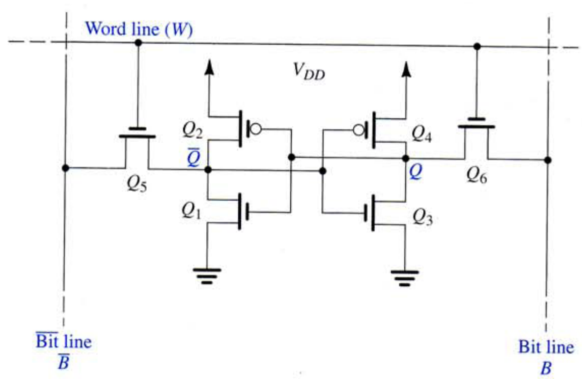


Fig. 2-1 A Cell of SRAM

2.1.1. The Read Operation

The goal of read operation is to retrieve the information stored in *Q* onto the bit lines. Assume the cell is initially stored a logic one. In this case, *Q* will be high at V_{DD}, and $\overline{Q}$ will be low at zero volt. Before the read operation begins, the bit lines are pre-charged to V_{DD}. When the word line is high and *Q5* and *Q6* are turned on, the current will flow from V_{DD} through *Q4* and *Q6* then onto line *B*; that will charge the capacitance, C_B, of line *B*. On the other side of circuit, current will flow from the pre-charged $\overline{B}$ line through Q5 and Q1 onto ground, thus discharging the bit line capacitance, $C_{\overline{B}}$. Fig. 2-2 summarized the operation.

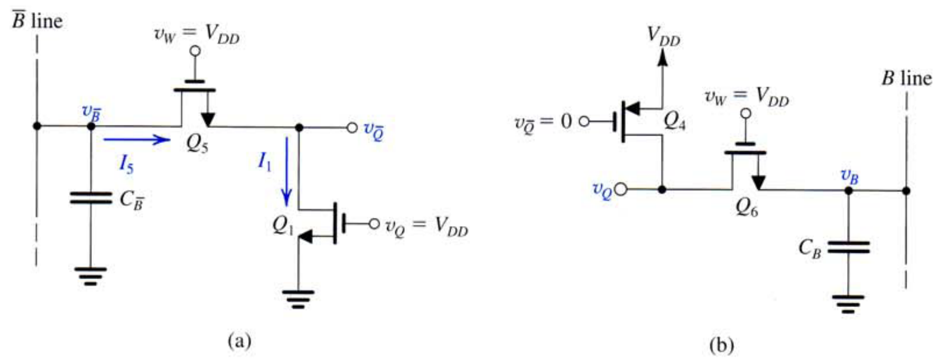


Fig. 2-2 (a) The Current Flow at the Left Inverter When Performing Read Operation. (b) The Current Flow at Right Inverter When Performing Read Operation.

Careful choice of transistor driving strength is necessary for correct operation. While $\overline{B}$ line is discharging through Q5 Q1 to ground, it would raise $V_{\overline{Q}}$ or the voltage at $\overline{Q}$. If $V_{\overline{Q}}$ has accidently been raised high to certain threshold, it might flip the voltage at Q. To avoid this, the pull down NMOS strength should design to be stronger than the access transistors, so it quickly drains out the rising voltage at $\overline{Q}$. This constraint will insure a stable read.

2.1.2. The Write Operation

The goal of write operation is to send the information on bit line into the cross-coupled inverters. In another word, write operation is making node Q to store the information on B line. Assume initially Q=V_{DD} and $\overline{Q}$=0, and the objective is to write Q=0 and $\overline{Q}$=V_{DD}. To do that, the B line will be discharged to zero, and the $\overline{B}$ line would be charged to V_{DD}. Once the write line goes high, the information on the bit lines would write into the inverters. Fig. 2-3 summarized the operation.

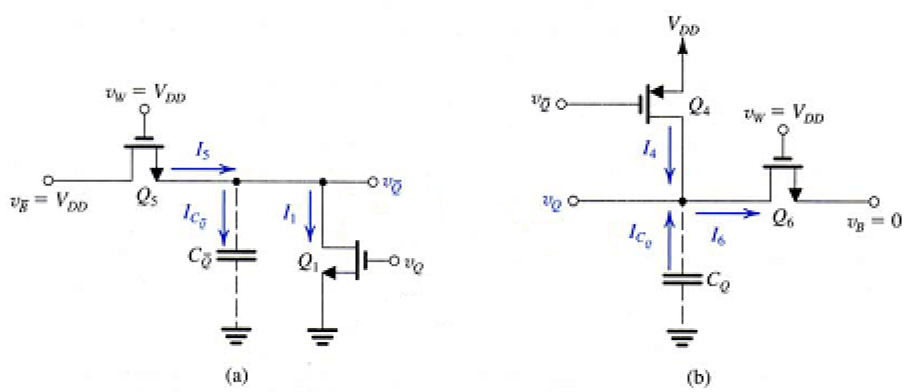


Fig. 2-3 (a) The Current Flow at the Left Inverter When Performing Write Operation. (b) The Current Flow at the Right Inverter When Performing Write Operation.

Since Q1 NMOS has strong pull down strength over Q5 to satisfy stable read constraint, Q5 transistor is unable to pull up $V_{\overline{Q}}$ because Q1 has greater pull down strength. Hence, the cell must be written by forcing V_Q low. Originally V_Q is V_{DD}, so the job of pulling V_Q down relies on the access transistor Q6. Transistor Q4 is opposing this operation because Q4 is supplying current to node Q. Thus, the strength of Q4 must be weaker than Q6, so Q6 is able to pull V_Q down.

Therefore, for SRAM to have correct operation and maintain sufficient read-ability and write-ability [16], the strength of MOS should be designed in this order: NMOS>ACCESS>PMOS. The driving strength depends on transistor sizing. For greater driving strength, designers tend to size up a transistor, and size down a transistor for low driving strength. The K values we will introduce later contain the transistor sizing factor. Therefore, for correct operation, the SRAMs often design to have $K_n > K_p$. [40]

2.2. LEVEL-1 SPICE MODEL

The modeling method is based on the most commonly known model. The level-1 MOSFET spice model consists with three regions: cut-off, linear, and saturate region. Table 1 summarizes the conditions for each region, and drain current equations for NMOS and PMOS. One thing to notice is that threshold voltage of PMOS from Table 1 is taken absolute value for simplicity later on. [39]

Table 1 L1-spice model of NMOS and PMOS equations

	NMOS	PMOS
Cut-off	$V_{gs} < V_{thn}$	$V_{sg} < \|V_{thp}\|$
	$I_{ds} = 0$	$I_{sd} = 0$
Linear	$V_{gs} > V_{thn}$	$V_{sg} > \|V_{thp}\|$
	$V_{ds} < V_{gs}\text{-}V_{thn}$	$V_{sd} < V_{sg}\text{-}\|V_{thp}\|$
	$I_{ds} = K_n(2\cdot(V_{gs}\text{-}V_{thn})\cdot V_{ds}\text{-}V_{ds}^2)$	$I_{sd} = K_p(2\cdot(V_{sg}\text{-}\|V_{thp}\|)\cdot V_{sd}\text{-}V_{sd}^2)$
Sat	$V_{gs} > V_{thn}$	$V_{sg} > \|V_{thp}\|$
	$V_{ds} > V_{gs}\text{-}V_{thn}$	$V_{sd} > V_{sg}\text{-}\|V_{thp}\|$
	$I_{ds} = K_n(V_{gs}\text{-}V_{thn})^2$	$I_{sd} = K_p(V_{sg}\text{-}\|V_{thp}\|)^2$

The term V_{ds} can be written as:

$$V_{ds} = V_d - V_s \quad (2.\ 1)$$

And V_d can be represented as:

$$V_d = V_{dg} + V_{gs} + V_s \quad (2.\ 2)$$

After substitute (2.2) to (2.1), the other way to write V_{ds} in NMOS is below:

$$V_{ds} = V_{gs} - V_{gd} \quad (2.\ 3)$$

Similarly, the V_{sd} in PMOS can be written in a similar manner as (2.4).

$$V_{sd} = V_{sg} - V_{dg} \quad (2.\ 4)$$

By substituting (2.3) for the NMOS equations, and (2.4) for PMOS equations, the level-1 current equations can be rewritten into the form shown in Table 2. [39]

Table 2 A different representations for the Level-1 model

	NMOS	PMOS
Cut-off	$I_{ds} = 0$	$I_{sd} = 0$
Linear	$I_{ds} = K_n((V_{gs}\text{-}V_{thn})^2\text{-}(V_{gd}\text{-}V_{thn})^2)$	$I_{sd} = K_p((V_{sg}\text{-}\|V_{thp}\|)^2\text{-}(V_{dg}\text{-}\|V_{thp}\|)^2)$
Sat	$I_{ds} = K_n(V_{gs}\text{-}V_{thn})^2$	$I_{sd} = K_p(V_{sg}\text{-}\|V_{thp}\|)^2$

The advantage of writing in the form in Table 2 is keeping variable inside the square term. Look at NMOS equation for now. Notice that V_{gs} is less than V_{thn} when in cut-off mode, and V_{gs} is higher than V_{thn} if not in cut-off mode. In another word, V_{gs}-V_{thn} is less than zero for cut-off mode and has zero current. That's the same as treating saturation equation with V_{gs}-V_{thn} equal to zero. If not in cut-off mode, V_{gs}-V_{thn} is higher than zero and the term V_{gs}-V_{thn} survive as shown in linear and saturation mode. An S(x) function will be discussed soon which can be a good representation for this kind of notation.

2.3. DEFINITION OF *S(X)* AND MODIFIED SET OF CURRENT EQUATION

S(x) function is defined to be zero if *x* is equal to or less than zero, and it is equaled to *x* if *x* is larger than zero.

$$S(x) = \begin{cases} 0 & X \le 0 \\ X & X > 0 \end{cases} \tag{2. 5}$$

The function S(x) is used to combine the three drain current equations of NMOS and PMOS transistors into one equation. For *S(x)* to serve this purpose, it satisfies the above approximation. Base on the property of (2.5), the current equations in Table 2 can combine three regions current equations (cutoff, linear, saturation regions) into one master equation. The equation (2.6) can represent the drain to source current of NMOS and (2.7) represent the source to drain current for PMOS.

$$I_{dsn} = K_n \cdot (S^2(V_{gs} - V_{thn}) - S^2(V_{gd} - V_{thn})) \tag{2. 6}$$

$$I_{sdp} = K_p \cdot (S^2(V_{sg} - |V_{thp}|) - S^2(V_{dg} - |V_{thp}|)) \tag{2. 7}$$

For continuously differentiable property, a smooth version of *S(x)* function in (2.8) can be used.

$$S(x) \approx \log(1+e^{A\cdot X})/A \qquad (2.\ 8)$$

The *A* is a constant. The suggested value for *A* is 100 for CMOS. The higher the *A* number is, the closer the log function approach to true *S(x)* function. Fig. 2-4 shows the plot of smooth version of *S(x)* function with *A*=100 and *A*=50. The transition is at zero sharper at higher *A* number. If smooth version of *S(x)* is desired to model MOS transistor current equation, the modified sets of current equations with smooth version of *S(x)* functions can be getting by plug (2.8) into (2.6) and (2.7). When use this smooth version of S(x) function, be ware of the digit of precision that machine need to handle to implement this type of function. When choosing a big *A* number like 100, assume V_{gs}-V_{thn} is 1, exponential of 100 is about ten to the power of 43. If V_{gs}-V_{thn} is zero, exponent of zero is one. That means this function deals numbers varying from one to ten to the power of 43, and not every compiler can deal this kind of precision.

$$I_{dsn} = \frac{K_n}{A^2}\cdot(\log^2(1+e^{A\cdot(V_{gs}-V_{thn})}) - \log^2(1+e^{A\cdot(V_{gd}-V_{thn})})) \qquad (2.\ 9)$$

$$I_{sdp} = \frac{K_p}{A^2}\cdot(\log^2(1+e^{A\cdot(V_{sg}-|V_{thp}|)}) - \log^2(1+e^{A\cdot(V_{dg}-|V_{thp}|)})) \qquad (2.\ 10)$$

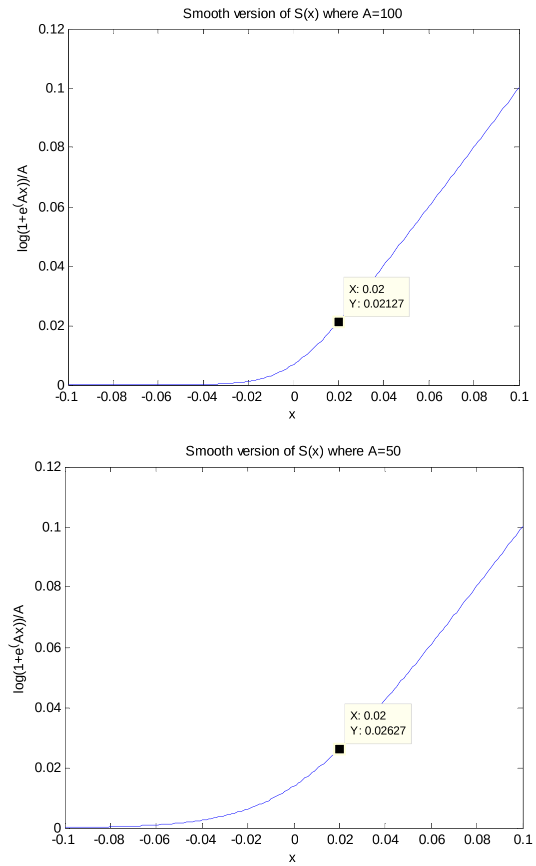


Fig. 2-4 The Plot of Smooth Version of S(x) With A=100 (Left) A=50 (Right)

2.4. SRAM CELL MODELING EQUATIONS

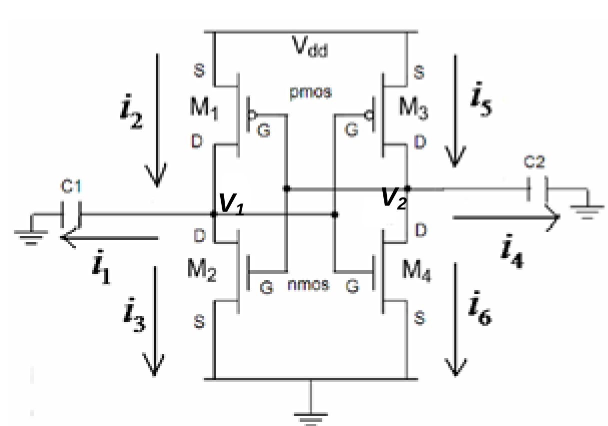


Fig. 2-5 The 4-T SRAM Circuit Diagram

The 6T SRAM cell consists of two cross-coupled inverters and two access transistors. The derivation is using Kirchhoff's current law evaluate at the two outputs.

$$\begin{aligned} i_2 &= i_1 + i_3 \\ i_5 &= i_4 + i_6 \end{aligned} \quad (2.\ 11)$$

Fig. 2-5 illustrates the SRAM circuit and labeling currents. The V_1 and V_2 are the outputs of SRAM. Both outputs connect to an access transistor and C_1, C_2 are the total lumped capacitances looking out from the output node. Without loosing generality, the noise disturbance is modeled as a current sources injected into the internal node V_1 V_2 as the output of CMOS inverter, labeled as I_{noise1} I_{noise2}. User can put noise source attached at other node and evaluate the voltage change at that particular node. The V_1 V_2 nodes are two sensitive nodes to noise, and the noise occurs on these two nodes directly effect the SRAM performance. So the thesis will focus on mostly the change at V_1 V_2 nodes. Moreover, in many books the PMOS threshold voltages are negative, and the PMOS drain to source current is negative. Here for simplicity, the derived equation treats threshold voltages all positive numbers included PMOS. It's the same thing but label differently. The labeling current for PMOS is source to drain, so everything in terms of mathematically is all positive in the equations.

KCL at V_1:

$$\begin{aligned} &K_1[S^2(Vsg - V_{th1}) - S^2(Vdg - V_{th1})] = C_1\dot{V}_1 \\ &\quad + K_2[S^2(Vgs - V_{th2}) - S^2(Vgd - V_{th2})] \end{aligned} \quad (2.\ 12)$$

KCL at V_2:

$$\begin{aligned} &K_3[S^2(Vsg - V_{th3}) - S^2(Vdg - V_{th3})] = C_2\dot{V}_2 \\ &\quad + K_4[S^2(Vgs - V_{th4}) - S^2(Vgd - V_{th4})] \end{aligned} \quad (2.\ 13)$$

The level-one modeling equation for SRAM is presented in (2.14) and (2.15). These equations will be analyzed and used through out the chapters. If smooth function is desired, the *S(·)* function can be substitute by the log function in (2.8).

$$\begin{aligned}\dot{V}_1 &= \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ &- \frac{1}{C_1} K_2 [S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] + \frac{1}{C_1} I_{noise1}\end{aligned} \quad (2.\ 14)$$

$$\begin{aligned}\dot{V}_2 &= \frac{1}{C_2} K_3 [S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] \\ &- \frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \frac{1}{C_2} I_{noise2}\end{aligned} \quad (2.\ 15)$$

CHAPTER III

NONLINEAR SYSTEM THEORIES AND CELL DYNAMICS OF SRAM

This chapter will focus on nonlinearity system theory and its application to the SRAM dynamic stability margin analysis. We will first introduce the concept of stability manifold, stability regions and separatrix mathematically for clarity. Then, we apply the general theorems to SRAM L1-Spice model, which is described by a two cross-coupled modified set of nonlinear differential equations. We will investigate its equilibrium points, linearized system equations and their qualitative behaviors. Furthermore, induced variations of separatrix will be investigated as device parameters varies. The change of separatrix categorizes to 0^{th} order, 1^{st} order and higher order effect.

3.1. EQUILIBRIUM POINTS OF GENERAL NONLINEAR SYSTEMS

We consider a nonlinear system represented by the following differential equation.

$$\dot{x} = f(t, x, u) \tag{3. 1}$$

where *x, u* are the state variable and input variable which are in multiple dimensions. The *t* is the time variable, and that referred system is time dependent. An N dimensional state variables and P dimensional input variable system state-space model will look like:

$$\begin{cases} \dot{x}_1 = f_1(t, x_1, x_2, \cdots, x_n, u_1, u_2, \cdots, u_p) \\ \dot{x}_2 = f_2(t, x_1, x_2, \cdots, x_n, u_1, u_2, \cdots, u_p) \\ \quad \vdots \\ \dot{x}_n = f_n(t, x_1, x_2, \cdots, x_n, u_1, u_2, \cdots, u_p) \end{cases} \tag{3. 2}$$

In particular, we consider the unforced state equation. That is,

$$\begin{cases} \dot{x}_1 = f_1(t, x_1, x_2, \cdots, x_n) \\ \dot{x}_2 = f_2(t, x_1, x_2, \cdots, x_n) \\ \quad \vdots \\ \dot{x}_n = f_n(t, x_1, x_2, \cdots, x_n) \end{cases} \tag{3. 3}$$

Note that as discussed in chapter II, an SRAM is an auntomonous system if no noise is involved. A system is an automonous system or time invariant system when the system is invariant to shift in time. A general automonous system equation would be as follows:

$$\begin{cases} \dot{x}_1 = f_1(x_1, x_2, \cdots, x_n) \\ \dot{x}_2 = f_2(x_1, x_2, \cdots, x_n) \\ \quad \vdots \\ \dot{x}_n = f_n(x_1, x_2, \cdots, x_n) \end{cases} \tag{3. 4}$$

The equilibrium points of the given nonlinear system can be solved by the following equations:

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0 \\ f_2(x_1, x_2, \cdots, x_n) = 0 \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \cdots, x_n) = 0 \end{cases} \tag{3. 5}$$

The equilibrium points, $x_{e,}$ which can be one or many points satisfied the above equation. In the case of SRAM, it has three equilibrium points when no noise is present. Two of them would be stable equilibrium points located around (0, V_{dd}) and (V_{dd}, 0), and one unstable equilibrium points normally located around the center in a phase portrait.

3.2. JACOBIAN MATRIX

The Jacobian matrix is an important matrix that can analyze local stability of an equilibrium point. The qualitative behavior of a nonlinear system near an equilibrium point can be determined via linearization with respect at that point.

Let p be an equilibrium point, $(p_1,\ldots,p_n)$ of the nonlinear system. Under the condition that functions f=f_1 …, f_n are continuously differentiable at p. By Taylor expansion of the right-hand-side of (3.4) around the equilibrium point, we have:

$$\dot{x} = f(p) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=p} \cdot (x-p) + \ldots higher\,order\,term \tag{3. 6}$$

Since p is an equilibrium point of the system, $f(p)$ is zero. If we restrict our attention to a sufficient small neighborhood of the equilibrium point, the first order term dominates the higher-order terms; accordingly, the higher-order terms can be dropped due to their insignificance. Moreover, since we are interested in the trajectories near p, define:

$$\Delta x = x - p \tag{3. 7}$$

Since p are constants, then

$$\Delta\dot{x} = \dot{x} \tag{3. 8}$$

And the state equation would be:

$$\Delta\dot{x} = \left.\frac{\partial f(x)}{\partial x}\right|_{x=p} \cdot \Delta x \tag{3. 9}$$

The Jacobian matrix is the term, $\left.\frac{\partial f(x)}{\partial x}\right|_{x=p}$, denoted by J_m. In general, the matrix looks like:

$$J_m = \begin{bmatrix} \frac{\partial f_1(x_1,..,x_n)}{\partial x_1} & \frac{\partial f_1(x_1,..,x_n)}{\partial x_2} & \cdots & \cdots & \frac{\partial f_1(x_1,..,x_n)}{\partial x_n} \\ \frac{\partial f_2(x_1,..,x_n)}{\partial x_1} & \ddots & & & \frac{\partial f_2(x_1,..,x_n)}{\partial x_1} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial f_n(x_1,..,x_n)}{\partial x_1} & \frac{\partial f_n(x_1,..,x_n)}{\partial x_2} & \cdots & \cdots & \frac{\partial f_n(x_1,..,x_n)}{\partial x_n} \end{bmatrix}\Bigg|_{\substack{x_1=p_1 \\ x_2=p_2 \\ \vdots \\ x_n=p_n}} \quad (3.\ 10)$$

For a second order system, let $p = (p_1, p_2)$ be an equilibrium point of the second order nonlinear system,

$$\begin{aligned} \dot{x}_1 &= f(x_1, x_2) \\ \dot{x}_2 &= g(x_1, x_2) \end{aligned} \quad (3.\ 11)$$

With the condition that function f and g are continuously differentiable. Just as before, Taylor series expansion around the equilibrium point (p_1, p_2) can be written into a form as below:

$$\begin{aligned} \dot{x}_1 &= f(p_1,p_2) + \frac{\partial f(x_1,x_2)}{\partial x_1}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot (x_1 - p_1) + \frac{\partial f(x_1,x_2)}{\partial x_2}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot (x_2 - p_2) + higher\,order\,term \\ \dot{x}_2 &= g(p_1,p_2) + \frac{\partial g(x_1,x_2)}{\partial x_1}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot (x_1 - p_1) + \frac{\partial g(x_1,x_2)}{\partial x_2}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot (x_2 - p_2) + higher\,order\,term \end{aligned} \quad (3.\ 12)$$

Since (p_1, p_2) is an equilibrium point of the system, $f(p_1,p_2)$ and $g(p_1,p_2)$ would be zero. Based on (3.7) and (3.8), second order system state equations can be written as:

$$\begin{aligned} \Delta\dot{x}_1 &= \frac{\partial f(x_1,x_2)}{\partial x_1}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot \Delta x_1 + \frac{\partial f(x_1,x_2)}{\partial x_2}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot \Delta x_2 \\ \Delta\dot{x}_2 &= \frac{\partial g(x_1,x_2)}{\partial x_1}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot \Delta x_1 + \frac{\partial g(x_1,x_2)}{\partial x_2}\bigg|_{\substack{x_1=p_1 \\ x_2=p_2}} \cdot \Delta x_2 \end{aligned} \quad (3.\ 13)$$

And the Jacobian matrix for a second order system becomes:

$$J = \left[\begin{matrix} \frac{\partial f(x_1,x_2)}{\partial x_1} & \frac{\partial f(x_1,x_2)}{\partial x_2} \\ \frac{\partial g(x_1,x_2)}{\partial x_1} & \frac{\partial g(x_1,x_2)}{\partial x_2} \end{matrix} \right]_{\substack{x_1=p_1 \\ x_2=p_2}} \tag{3. 14}$$

In summary, the matrix J is called the Jacobian matrix, and it is evaluated at the equilibrium point, p. Under some conditions, it is reasonable to expect the trajectories of nonolinear system in a small neighhorhood of an equilibrium point behaves like the trajectories of its linearization around that equilibrium point. [33] [41] This implies that if the trajectories of the nonlinear state equation around an equlibrium point behaves like a stable node, the linearized state equation around that equlibrium point would also be a stable node with distinct eigenvalues respectively.

Each equilibrium point is associated with a Jacobian matrix. The real part of the eigenvalues of the Jacobian matrix evaluated at a stable equilibrium point would be both negative. At least one of the real parts of eigenvalues of the Jacobian matrix evaluated at unstable equilibrium point will be positive. In conclusion, to analyze the stability of a particular equilibrium point, the Jacobian matrix is evaluated at that point, and the Eigen-values of Jacobian matrix reveal the local stability around the equilibrium point.

3.3. QUALITATIVE BEHAVIOR OF LINEARIZED SYSTEMS

Assume a general nonilnear system, the linear time-invariant system get from previous section looks like:

$$\Delta\dot{x} = J \cdot \Delta x \tag{3. 15}$$

Where J is Jacobian matrix as mention before, and the eigenvalue can be found by solving λ for the following equation:

$$\det(J - \lambda I) = 0 \qquad (3.\ 16)$$

If we restrict ourselves to second order automonous systems, the J would be two by two and the number of eigenvalues λ would be at most two. The qualitative behavior around an equilibrium point is highly depends on its eigenvalues. In mathmatics, there are three topological classification of generic *hyperbolic equilibra*: node, focus and saddle.

3.3.1 Node ($\lambda_1 \neq \lambda_2 \neq 0$)

The nodal behavior happens when both eigenvalues are real and non-zero. The stable node is the situation that eigenvalues, λ_1 and λ_2, are negative. The unstable node happen when both eigenvalues are positive. As shown in Fig. 3-1(a), suppose the origin is an equilibrium point, an initial condition start around the origin will converge to the origin. Hence, the idea of stable is when an initial point start near the origin and it will stay close to the origin. If $\lambda_1 < \lambda_2 < 0$, λ_1 would be called the fast eigenvalue and λ_2 is the slow eigenvalue. The corresponding v_1 and v_2 would be called fast and slow eigenvector. Fig. 3-1(b) shows an unstable node. The trajectories of an unstable equilibrium points away from the origin. Thus, an initial point starts near the origin would go away from the origin.

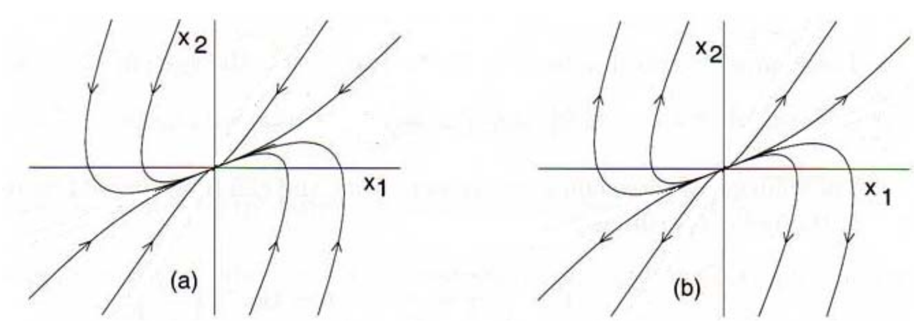


Fig. 3-1 Phase Portrait of (a) Stable Node; (b) Unstable Node.

3.3.2 Focus ($\lambda_{1,2} = \alpha \pm j\beta$)

The focus behavior happens when the eigenvalues are complex in the form of $\alpha \pm j\beta$ where α and β are real numbers. If α is a negative number, the equilibrium is said to have stable focus behavior. Whereas α is a possitive number, the equilibrium is said to have unstable focus behavior. Fig. 3-2 shows an example of stable focus and unstable focus. When α is a negative number, the spiral converge to the origin; when α is a positive number, the spiral diverge away from the origin.

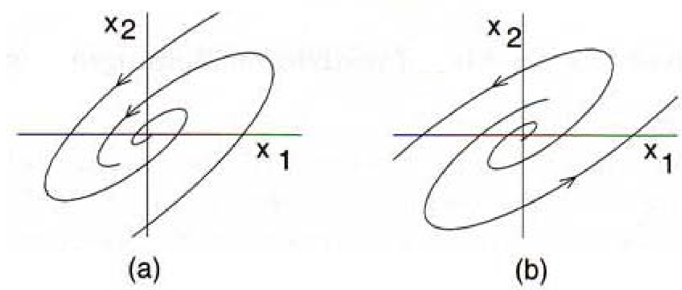


Fig. 3-2 Phase Portrait for (a) Stable Focus; (b) Unstable Focus.

3.3.3 Saddle ($\lambda_1 < 0 < \lambda_2$ or $\lambda_1 > 0 > \lambda_2$)

The saddle behavior happens when two eigenvalues are real number and have different sign. If λ_2 is a positive eigenvalue and λ_1 is a negative eigenvalue, λ_1 is called stable eigenvalue and λ_2 is called unstable eigenvalue of this equilibrium point. The corresponding eigenvector v_1 and v_2 would be called stable and unstable eigenvectors respectively. Fig. 3-3 shows an example of saddle. In this example, there has trajectories pointing inward to the origin and trajectories pointing outward away from the origin. Thus, the origin is a saddle point in this case.

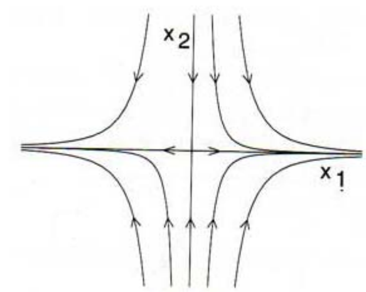
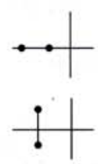
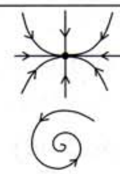
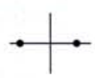
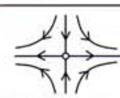
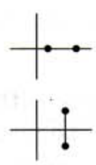
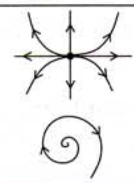


Fig. 3-3 Phase Portrait for a Saddle.

The idea of stable is when an initial condition start close to an equilibrium, it will stay around that equilibrium. Whereas unstable equilibrium points, an initial condition start near it, it will be driven away from the equilibrium. Therefore, the equilibra have the behaviors like stable node and stable focus would be considered as stable equilibrium points. In contrast, the equilibra have behaviors like unstable node and unstable focus would be unstable equilibrium points. A saddle point is also a case of unstable equilibrium point. Because a saddle has unstable eigenvalue and eigenvector, there would be a trajectory going away from the equilibrium. Table 3 gives a summary about the three important classifications.

Fig. 3-4 gives an example of 65nm SRAM phase portrait. The system paramters used to generate this 65nm phase portrait figure is posted in Appendix. The phase portrait for a typical SRAM has three equilibrium points. In this example, these points are located at (1,0) (0,1) and (0.44, 0.44). Fig. 3-5(a) (b) and (c) show the plots of linearized system around (1,0) (0,1) and (0.44,0.44). The small white arrows on the figure are the vector fields. Notice the vector field for Fig. 3-5(a) and (b) are all pointing inward to the equilibrium point, that also means, an initial point start close to it; the vector fields will pull it to the equilibrium. Hence, (1,0) and (0,1) are the stable equilibrium points. Whereas the center point is an unstable equilibrium. There are trajectories goes toward and away from the center point. Thus, the center point has the behavior like saddle.

Table 3 Topological Classification of Hyperbolic Equilibra

Eigenvalues	Phase portrait	Stability
	 node focus	stable
	 saddle	unstable
	 node focus	unstable

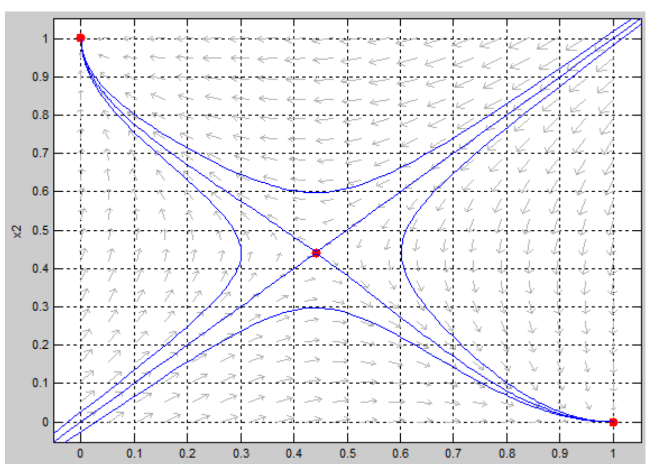


Fig. 3-4 Example of 65nm Technology SRAM Phase Portrait

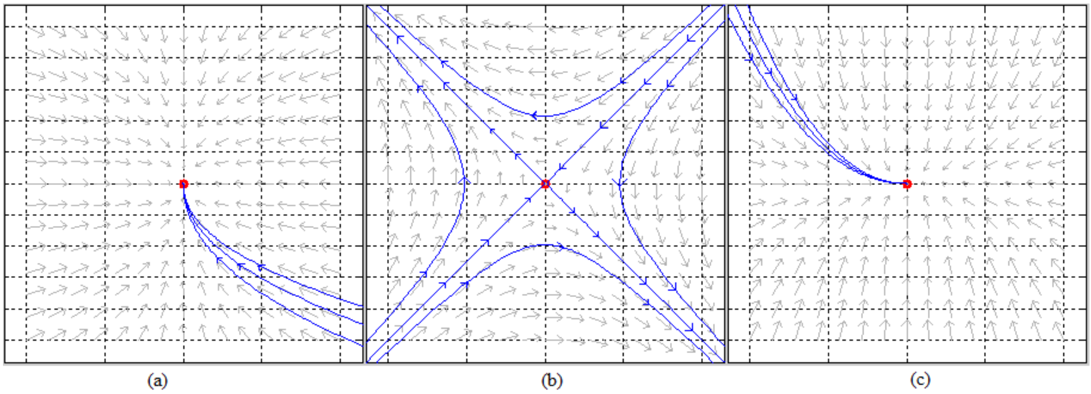


Fig. 3-5 The Linearization Plot for (a) Left-Top Equilibrium Point; (b) Center Point; (c) Right-Bottom Equilibrium Point.

3.4. SYSTEM STABILITY THEORIES AND CONCEPTS

The stability regions of associated equilibrium points determine the system behavior. In other words, an SRAM state starting anywhere within the stability region would converge to its equilibrium state. And the stability boundary is a border that separate stable regions. During the SRAM operations, a state flipping would occur if the state is perturbed across the stability boundary. In a symmetrical case, the stability boundary is simply a 45 degree line passing through the origin on the phase portrait [31-32]. The stability boundary for a given SRAM is also called *separatrix* because the stability boundary separates two stability regions. [32] In the case of SRAM cell, if the injected noise is higher than the stability margin, the state of the cell can deviate from the initial stable equilibrium and cross the separatrix after certain time period. If this happens, the cell state will fall into the stability region of the other stable equilibrium state and result in a state flip.

3.4.1 Definition of Stability Associated with an Equilibrium Point [41]

***Definition*:** *An equilibrium point* x_e *is said to be stable if, for any given* $\varepsilon > 0$*, there is a* $\delta > 0$*, depend on* ε*, such that, for an initial condition* $\| x(0) - x_e \| \leq \delta$*, the solution satisfied the inequality* $\| x(t) - x_e \| \leq \varepsilon$ *for all* $t \geq 0$*.*

***Definition*:** *An equilibrium point is said to be unstable if it is not stable.*

3.4.2 The Stability Boundary Theory

For a given dynamic equation $\dot{x} = f(x)$ with x in an N dimensional space, the equilibrium points are all the x_e's that satisfy $f(x_e)=0$. Its stable manifold and stability region can be described as below: [32]

3.4.2.1 General Theorems

The stable manifold of an equilibrium point x_e is defined as: [41-43]

$$W^s(x_e) = \{x \in R^N \mid \lim_{t \to \infty} \phi(t, x) = x_e\} \qquad (3.\ 17)$$

where $\phi(t, x)$ is the trajectory that starts from x and eventually converges to x_e. The stability region or region of attraction $A(x_e^s)$ of a stable equilibrium point x_e^s is the stable manifold of stable equilibrium point, x_e^s.

Definition of hyperbolic equilibria: [41]

An equilibrium is called hyperbolic if there are no eigenvalues on the imaginary axis.

Stable Manifold Theorem For a Fix Point: [33]

Suppose that $\dot{x} = f(x)$ *has a hyperbolic fix point* $\overline{x}$. *Then there exist local stable and unstable manifold* $W^s_{loc}(x)$ $W^u_{loc}(x)$ *of the same dimension* n_s, n_u *as those of the eigenspace* E^s, E^u *of the linearized system (3.15) and tangent to* E^s, E^u *at* $\overline{x}$. $W^s_{loc}(x)$ $W^u_{loc}(x)$ *are as smooth as the function f.*

The stability boundary of the stability region is denoted by $\partial A(x_e^s)$. Based on some generic assumptions, we have the stability boundary theorem [44]:

Assumptions for Stability Boundary Theorem:

- All equilibria in $\overline{A(x_s^e)}$ are hyperbolic.
- Every trajectory in $\overline{A(x_s^e)}$ converges to an equilibrium point.
- The stable and unstable manifold of the equilibria in $\overline{A(x_s^e)}$ intersect transversely.

Stability Boundary Theorem: [44]

The stability boundary $\partial A(x_e^s) = U_m W^s(x_m)$ *where* x_m, *m=1,2, …, are all equilibria of any order in* $\partial A(x_e^s)$.

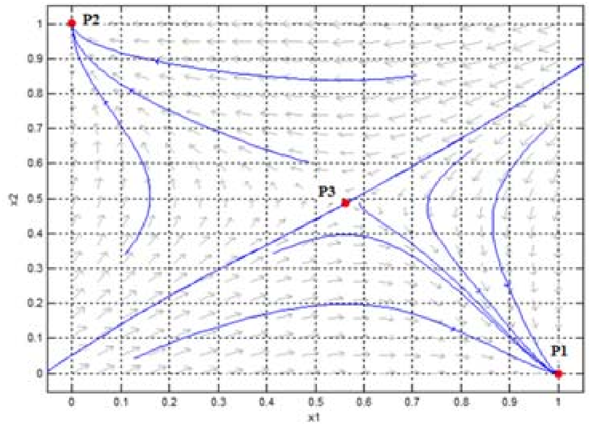


Fig. 3-6 An Example of Phase Portrait for an SRAM.

3.4.2.2 The Stable Manifold and Stability Boundary Theorem for SRAM

In particular case, from Fig. 3-6, P1 and P2 are two stable equilibrium points, x_e^s. The stability region of equilibrium point P1 is the region of all initial states whose trajectories will converge to P1. Accordingly, the stable region of P1 is the bottom right region in the phase portrait. Likewise, the stability region of P2 is the top left region of the phase portrait. The question remains on how to describe the stability region in a precise mathematical sense. From the same figure, we can see the stability boundary (the manifold passing through P3) naturally divides the state space into two stability regions. Accordingly, the stability boundary becomes one of the key components that decide the stability margin.

In SRAM case, stable equilibria are hyperbolic, every trajectory in $\overline{A(x_s^e)}$ converges to P3 and the stable and unstable manifold of P3 satisfies transversality. Thus, stability boundary theorem can be applied since SRAM satisfies the generic conditions. For the case of SRAM, saddle (P3) is the only one equilibrium on $\partial A(x_e^s)$, so the stability boundary is the stable manifold of saddle. Therefore, the stability boundary for SRAM can be described as:

$$\partial A(x_e^s) = W^s(x_e^u) \tag{3. 18}$$

where x_e^u is the unstable equilibrium point P3 on the boundary of *A*. Accordingly, to find the stability boundary, first is to identify the unstable equilibria on the stability boundary and find their stable manifolds. According to the Stable Manifold Theory [33], the stable eigenvectors of the linearized system around the equilibrium point will be tangent to its corresponding stable manifold. Thus, we can start in a small neighborhood of x_e^u along the directions of stable eigenvector to integrate reverse in time to find the stable manifolds. We need to reverse in time to bypass the stability nature of the trajectories that will converge to x_e^u in a short distance.

As an example, Fig. 3-6 illustrates the above theorem. In Fig. 3-6, P3 is an unstable equilibrium point. The trajectory pass through P3 is the separatrix that separate the state space into two stability regions. Points initially starts on the Separatrix will converge to the unstable equilibrium point, P3. The tangent vector on the Separatrix is the stable eigenvector with the stable Eigen-value of the linearized system around, P3.

3.4.3 An Algorithm to Find the Two Dimentional Stability Boundary or Separatrix

Based on the stability boundary theorem and the stable manifold theorem, we can see for a two dimensional nonlinear systems such as SRAM, the stability boundary can be found by the following procedure:

1. Find all the x_e^u and x_e^s.

2. Focus on the interested x_e^s.

3. Check if x_e^u are on stability boundary.

4. Find the stable eigenvectors, V_s, of the equilibrium point x_e^u, where the stable eigenvector is the eigenvector corresponding to the stable eigenvalue.

5. Choose initial condition as $x_0 = x_e^u \pm \varepsilon \cdot V_s$, where ε is a small positive number.

6. Integrated backward by $\dot{x} = -f(x)$.

In practice, we can bypass procedures 4 and 5 as long as the initial conditions are nearby the unstable equilibria since the unstable components will dissipate fast as we integrate reverse in time. In 65nm technology SRAM as example, the unstable equilibrium point is (0.44, 0.44) and stable equilibrium points are (1,0) and (0,1). In order to find the stable and unstable eigenvectors of unstable equilibrium point, one way is finding out the Jacobian matrix addressed previously and evaluated at (0.44,0.44). This Jacobian matrix gives eigenvalues of ($1x10^{-11}$) and ($-1x10^{-11}$); the corresponding eigenvector are (0.707,-0.707) and (0.707,0.707). As mention before, the eigenvalue ($1x10^{-11}$) is positive, so it's unstable eigenvalue and the corresponding eigenvector (0.707,-0.707) is unstable eigenvalue; for the eigenvalue ($-1x10^{-11}$), its stable eigenvalue and the eigenvector (0707,0.707) would be stable eigenvector. This stable eigenvector would be the V_s described in step 4. By following the procedures, integrating backward from the unstable equilibrium point as described in step 6, the Separatrix can be traced out as shown in Fig. 3-7.

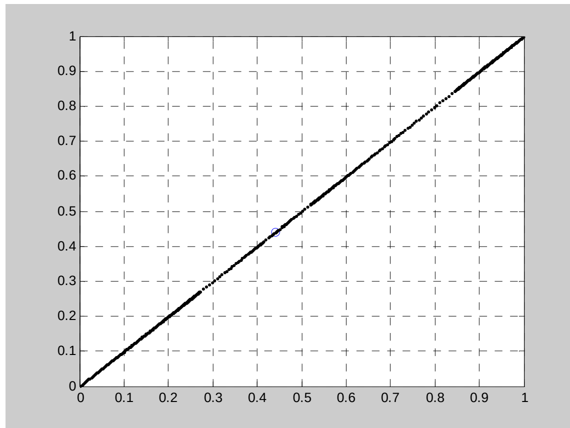


Fig. 3-7 Separatrix Plot of a 65nm SRAM Cell

3.5. NULLCLINES ON PHASE PORTRAIT

In general, consider a second order autonomous system:

$$\begin{cases} \dot{x}_1 = f(x_1, x_2) = c \\ \dot{x}_2 = g(x_1, x_2) = d \end{cases} \tag{3. 19}$$

The nullcline of x_1 or x_1-nullcline is the set of points satisfied the above equation with $c=0$, and nullcline of x_2 or x_2-nullcline is the set of points satisfied with $d=0$. As we know, equilibrium points are found by solving function f and g with both c and d are zero. In the other word, the points of intersection between $x1$-nullcline and $x2$-nullcline are exactly the equilibrium points.

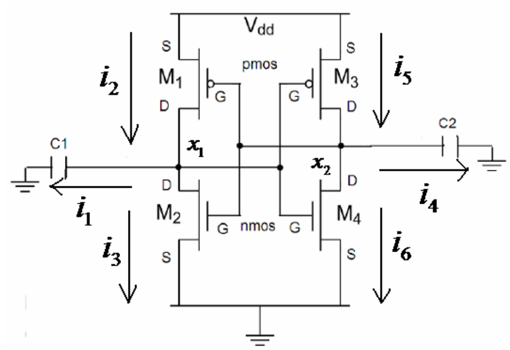


Fig. 3-8 SRAM Current Labels.

Recall the derived nonlinear differential equation without the noise input from previous chapter,

$$\dot{V}_1 = \frac{1}{C_1} K_1[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ - \frac{1}{C_1} K_2[S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] \tag{3. 20}$$

$$\dot{V}_2 = \frac{1}{C_2} K_3[S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] \\ - \frac{1}{C_2} K_4[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] \tag{3. 21}$$

the nullcline of V_1 can be found by setting (3.20), the derivative of V_1, to zero, and nullcline of V_2 can be found by setting (3.21), the derivative of V_2, to zero. Look at Fig. 3-8 under no noise current condition. The KCL at node V_1 comes from i_1=i_2-i_3 as illustrated previously. If $\dot{V}_1$ is set to zero by the definition of nullcline, that's the same thing as setting i_1 to zero which the current flow through capacitor is zero, and the plot of i_2-i_3=0 is the plot of the nullcline of V_1. By plotting out i_2=i_3, it gives input versus output plot of left inverter. In another word, the nullcline of V_1 is the input-output plot of left inverter. Similarly, the nullcline of V_2 is the input-output plot of right inverter. Fig. 3-9 shows the nullclines for 65nm parameter SRAM as an example.

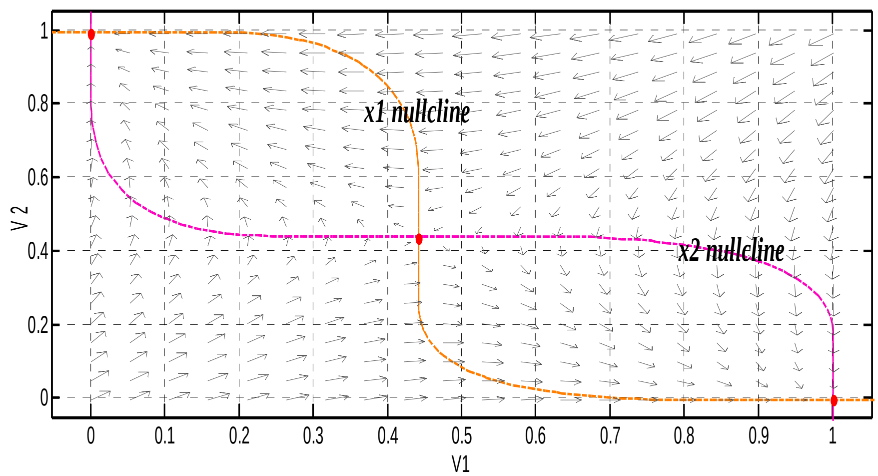


Fig. 3-9 An Example of Nullclines on 65nm Technology SRAM.

3.6. NOISE INDUCED BIFURCATION

In mathematics, a bifurcation occurs when a small smooth change to the parameter value, also called bifurcation parameters, of a system causes a sudden qualitative or topological change in its long-term dynamical behavior. Generally, there are two principle classes of bifurcation [33-34] [41] [35-37]:

3.6.1 Local Bifurcation

A local bifurcation occurs when a parameter change causes the local stability of an equilibrium or fixed point to change.

3.6.2 Global Bifurcation

Global bifurcation occurs when large invariant sets, such as periodic orbits, collide with equilibria and result change in phase portrait globally without local stability changes around equilibria.

For SRAM, one of the bifurcation parameters is the inject noise current I_{noise1} or I_{nois2}. When noise is present, the equilibrium points will change and the cell's phase portrait could be drastically changed. When noise is added to the SRAM circuit, the stable equilibrium points remain in their relative positions while the saddle point moves closer to one of the equilibrium point, depending on the direction of the noise current. After certain amplitude of noise, $I_{critical}$, the saddle point will collide with a stable equilibrium point, resulting in a *saddle-node bifurcation*. The location that bifurcation occurs is called the bifurcation point. Saddle-node bifurcation is a type of local bifurcation [33] [41]. When this happens, the two colliding equilibrium points disappear, and only the other remaining stable equilibrium point will survive.

Assume there is constant noise input at the V_2 node, a representation of the system equations is given below. With an initial condition starting from $(V_{dd}, 0)$, saddle-note bifurcation will happen as the noise current, I_{noise}, increase.

$$\begin{cases} C_1\dot{V}_1 = f(V_1, V_2) \\ C_2\dot{V}_2 = g(V_1, V_2) + I_{noise} \end{cases} \tag{3. 22}$$

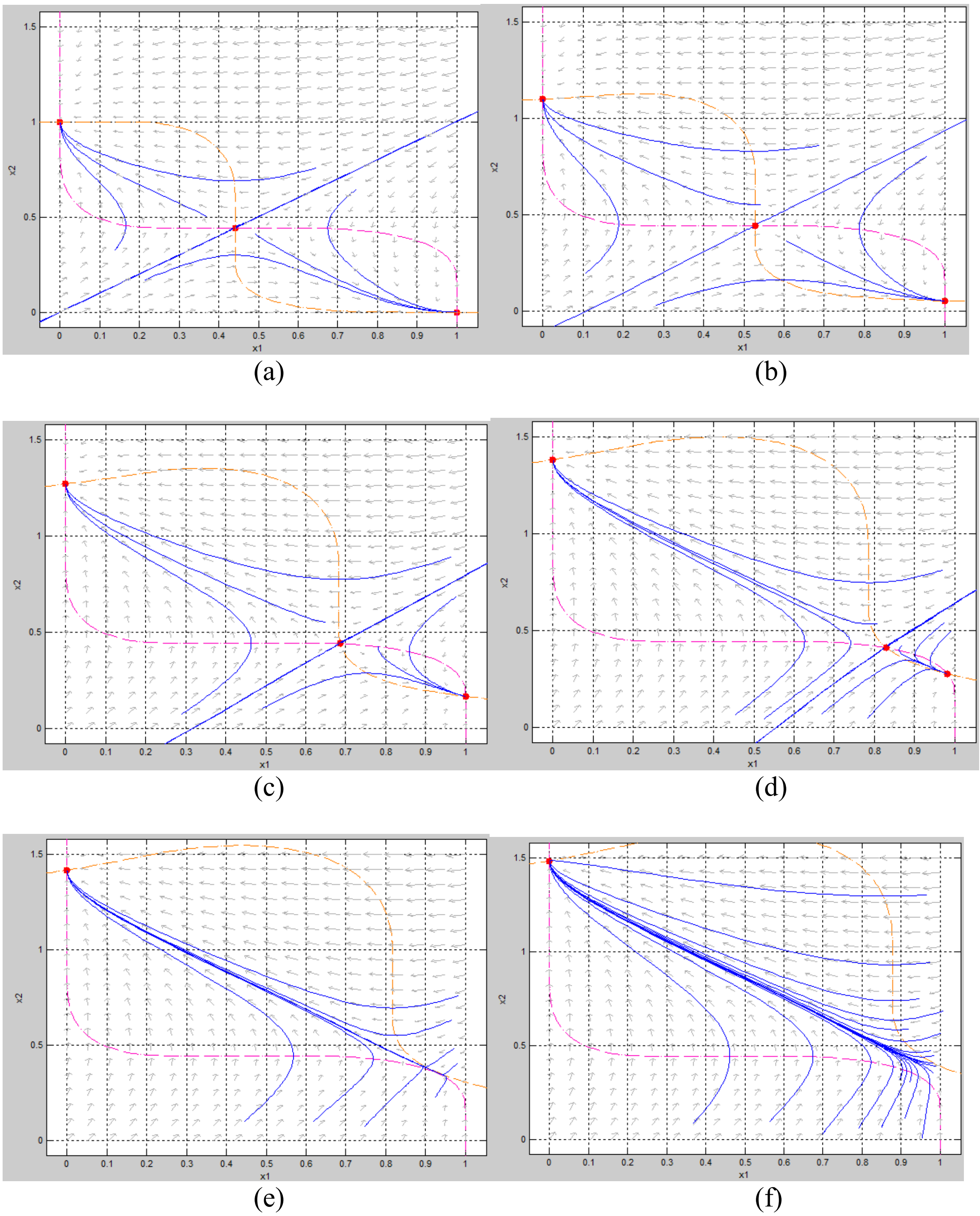


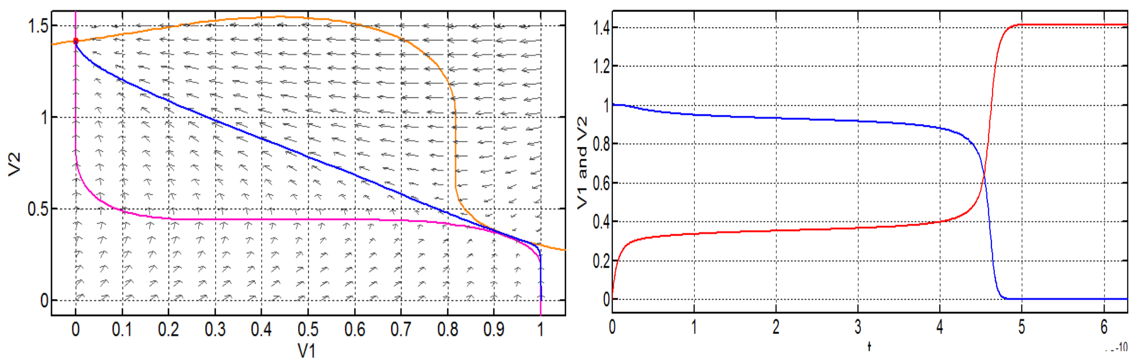
Fig. 3-10 The Example of Noise Bifurcation. Noise Level are (a)0 A; (b)200 uA; (c)300 uA; (d)450 uA; (e)500 uA; (f)600 uA.

Fig. 3-10 illustrates an example on the occurrence of noise induced bifurcation. As noise current increasing, the saddle node will gradually approach to the stable equilibrium state on the right side of the separatrix. When the noise amplitude reach the critical amplitude, the saddle point collides with the stable node, and the saddle node and the stable node along with separatrix disappear and result in a saddle-node bifurcation. In Fig. 3-10, the critical amplitude ($I_{critical}$) is 497uA and equilibria colliding point (bifurcation point) is located at (0.93,0.35) for 65nm nominal values. Thus, the only equilibrium point left is the equilibrium point originally on the left side of separatrix.

3.7. SRAM CELL DYNAMICS

One common way to model the characteristics of an SRAM cell is known as the *static noise margin* [20] [28-31] [45]. The noise margin is defined as the maximum amplitude of noise that can exist in a SRAM cell so that when the noise disappears, the cell does not flip states [31-32]. In the sense of $I_{critical}$, which was illustrated previously, represents the noise margin of a SRAM cell.

As long as I_{noise} is less than $I_{critical}$, the states will never cross the separatrix, so when the noise disappears, the states of the cell will always return to its stable equilibrium point. However, the static noise margin is not good enough to characterize the noise tolerance of this cell since the noise current, I_{noise}, being greater than $I_{critical}$ not necessarily implies that the cell will flip its state [31-32]. I_{noise} must be greater than $I_{critical}$ for a certain period of time (defined as $T_{critical}$). Once the state of the cell crosses the separatrix, the cell will flip states even though the noise disappears. For state flip to occur, the state of SRAM must cross the separatrix. Therefore, assume the noise amplitude is equal to or higher than $I_{critical}$, $T_{critial}$ defined to be the time it takes from initial state to the separatrix. If the present of noise current with amplitude $I_{critial}$ has shorter duration than $T_{critial}$, the state has not yet crossed the separatrix, and it will come back when the noise disappear. On the other hand, the presence of noise has greater duration over $T_{critial}$, the state of SRAM would cross the separatrix, and state flip is inevitable even though disturbance is gone.

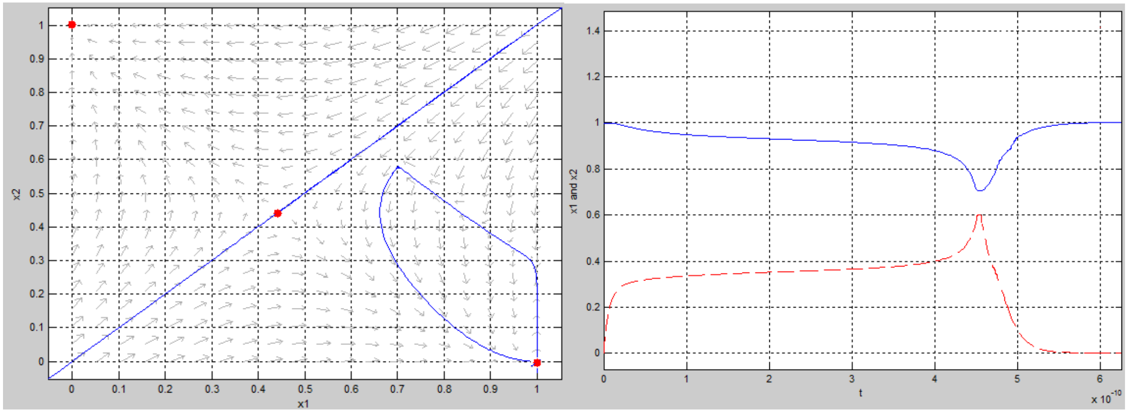


(a) (b)

Fig. 3-11 (a) Phase Portrait of SRAM When I_{noise} is 500 μA; (b) The Timing Diagram of This Cell.

Fig. 3-11(a) shows the phase portrait under nose level I_{noise} of 500 μA. Assume this noise acts as a step input to V_2 node, which holds constant without switching off. On the phase portrait, a trajectory starts from (1, 0) and gradually converge to the equilibrium point on (0, 1.4). From simulation result, the saddle-node bifurcation happens at approximately I_{noise} of 496 μA, so $I_{critical}$ would be 496 μA. Because in this case I_{noise} is 500 μA which is slightly larger than $I_{critical}$, there is only one equilibrium point located at *(0, 1.4)* on the entire phase portrait. So, if the cell's states initially start at *(1, 0)*, the state of the cell will be converging to *(0, 1.4)*. When plotting V_1 and V_2 in time diagram as shown in Fig. 3-11(b), one can see that it takes approximately 0.48 ns for the cell to reach the equilibrium point *(0, 1.4)* which results a state flip. In this case, the separatrix is simply the linear line $V_1 = V_2$ [31-32] across the origin, and it will take the cell about 0.45 ns to reach the separatrix. After the state of the cell crosses over the separatrix, the cell will not be able to come back to its original state. This means that a noise current pulse of constant amplitude 500 μA applied for less than 0.45 ns may not make the cell flip its state. However, when the noise duration is longer than 0.45 ns, the cell will flip its state. Therefore, 0.45ns is the $T_{critical}$ in this case. In addition, note that the transition time from the separatrix to the other equilibrium is only 0.03ns, which is only 1/15 fraction of the total transition time (.45ns), but the traveled distance is relatively long within such a short period of time.

Consider a type of square pulse noise that has constant amplitude of 500 μA and its duration is only last 0.43ns. The cell state will not flip because the duration is less than $T_{critical}$. From Fig. 3-11(b), the cell state is (0.7,0.582) at 0.43ns. Since the separatrix in this example is a straight line V_1=V_2 passing through the origin, after the noise is gone, the cell state (0.7, 0.582) is in the bottom right region of attraction of the equilibrium point $(V_{dd},0)$. The cell state will go back to $(V_{dd},0)$ gradually as shown in Fig. 3-12.



(a) (b)

Fig. 3-12 (a) Phase Portrait and (b) Time Diagram for a Square Pulse Noise of 500µA Amplitude and 0.43ns Duration.

3.8. NONLINEARITY OF THE SEPARATRIX

The separatrix is a nice straight line, $V_1=V_2$ passing through the origin, when an SRAM is symmetric. That is, the NMOS and PMOS parameters from each inverter are exactly the same. However, it's not always the case in reality. There will be mismatches in parameters such as threshold voltage mismatches, mismatch in device width, length…etc. When that happens, the separatrix may not be a straight line $V_1=V_2$ through the origin anymore. [32] The separatrix involved linearly shift, slope change effect, and nonlinearly curved.

3.8.1 0^{th} Order Effect

The 0^{th} order effect regards to the shift of the saddle; the location of the separatrix may also be shift but the slope does not change.

3.8.1.1 Deviation of V_{dd}

The variation of V_{dd} does not affect the separatrix. The saddle point change but the separatrix is still $V_1=V_2$ passing through origin. Fig. 3-13 shows an example on variation of V_{dd}.

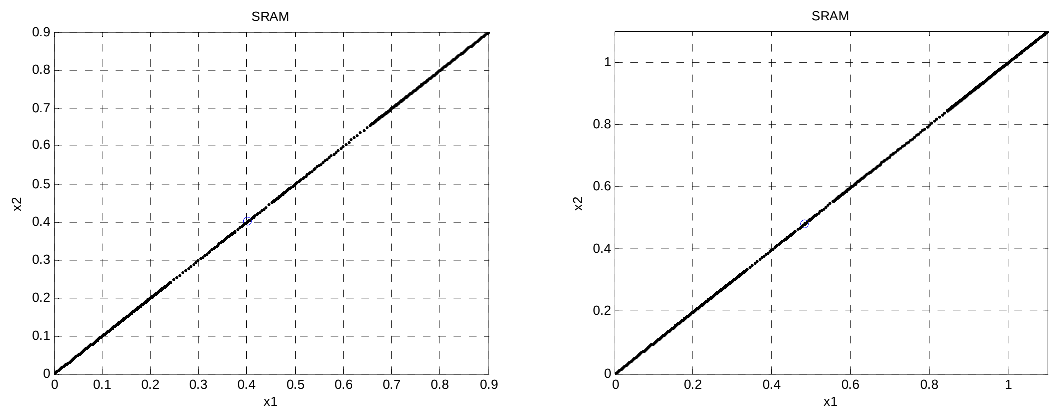


Fig. 3-13 (Left) V_{dd} Decrease By 10%; (Right) V_{dd} Increase By 10%

3.8.1.2 Symmetrical Change of Parameters

As long as the same diviation on both NMOS or PMOS parameters, the separatrix would not change its slope. For example as shown in Fig. 3-14, increase V_{th1} and V_{th3} both by 33% or decrease both by 33% would not affect the slope of separatrix. The reason is because V_{th1} is the threshold voltage of PMOS at left inverter, and V_{th3} is the threshold voltage of PMOS for the right inverter. Although they deviate from their nominal values, left inverter and right inverter are still symmetrical. It does not have to be a pair of V_{th1} and V_{th3}, it can also be a pair of V_{th2} V_{th4}, and K_1 K_3 as well.

Moreover, symmetrical change of parameters may move the saddle point along the separatrix, but the separatrix remain the same. In addition, it can also be multiple pairs of symmetrical change of parameters. There is an example shown in Fig. 3-15. Increase same amount on V_{th1} V_{th3} K_1 K_3 and decrease same amount on V_{th2} V_{th4} K_2 K_4, it would not affect the slope of separatrix.

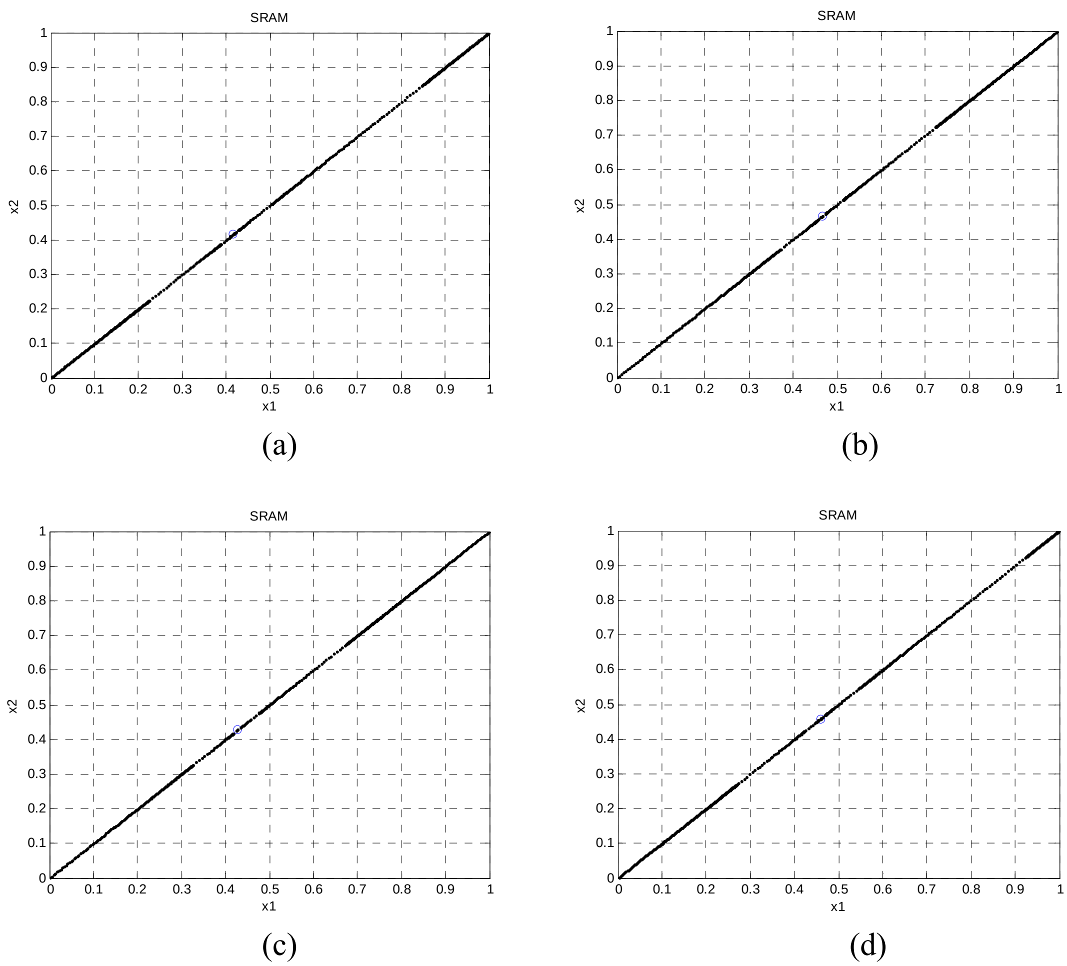


Fig. 3-14 (a) V_{th1} and V_{th3} Increase By 33%; (b)V_{th1} and V_{th3} Decrease By 33%; (c) K_1 and K_3 Increase By 20%; (d) K_1 and K_3 Decrease By 20%.

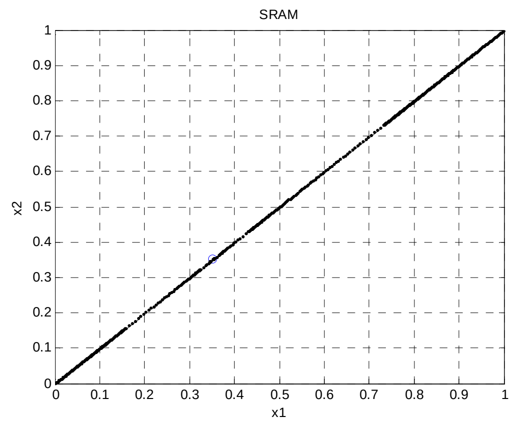
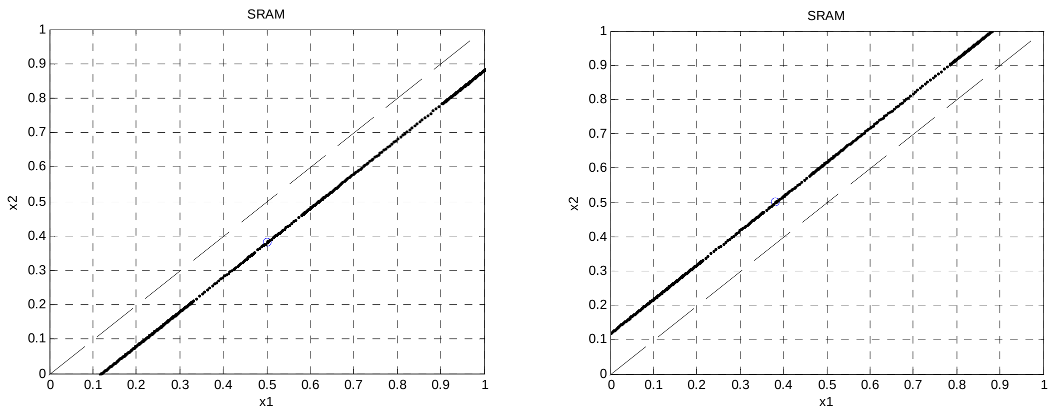


Fig. 3-15 Example of 0th Order Effect On Symmetrical Change On Multiple Pairs of Parameters.

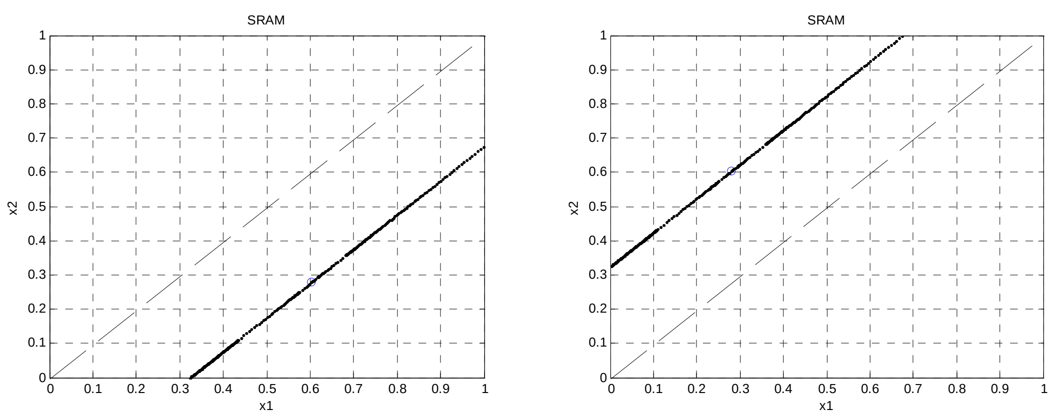
3.8.1.3 Special Case on Linearly Shift of Separatrix

This is an unusual situation observed from simulations. The separatrix seems to shift down linearly when the threshold voltages is deviated in the following manner: $V_{th1}=V_{th1nominal}*(1+n\%)$, $V_{th2}=V_{th2nominal}*(1-n\%)$, $V_{th3}=V_{th3nominal}*(1-n\%)$, $V_{th4}=V_{th4nominal}*(1+n\%)$. And the separatrix linearly shift up when the changes of threshold voltages is in the following manner: $V_{th1}=V_{th1nominal}*(1-n\%)$, $V_{th2}=V_{th2nominal}*(1+n\%)$, $V_{th3}=V_{th3nominal}*(1+n\%)$, $V_{th4}=V_{th4nominal}*(1-n\%)$. The n represents the percentage of deviation; the threshold voltages deviation from its nominal values by 33% if n=33. Fig. 3-16 demonstrate this case when n=33. Fig. 3-17 shows the case of n=90. The dash line represents the old separatrix $V_1=V_2$ passing through the origin.



(a) (b)

Fig. 3-16 Example of Separatrix (a) Linearly Shift Down; (b) Linearly Shift Up; When n=33.



(a) (b)

Fig. 3-17 Example of Separatrix (a) Linearly Shift Down; (b) Linearly Shift Up; When n=90.

3.8.2 1st Order Effect

The 1st order effect covers the slope change of separatrix.

This situation can happen very often. It appears to be slightly deviation on a single parameter, roughly within 50% on a single threshold votage and capacitance, about 0% to 15% on any of *K* values would change the slope of separatrix, and the separatrix also closely remains a continuous straight line passing through saddle. For instance, this case happens when increase V_{th1} by 33% and leave the rest parameter the same, or increase C_1 by 5% and leave the rest paratmers untouched. From The figure below, the dash line represent the separatrix at nominal value, which is $V_1=V_2$ passing through the origin. If the new separatrix has change of slope, and the separatrix may not pass through the origin. In addition, some other changes make the slope change more severe. Fig. 3-18(c) is the phase portrait for threshold voltages change in the following combination: $V_{th1}=V_{th1nominal}*(1+20\%)$, $V_{th2}=V_{th2nominal}*(1+20\%)$, $V_{th3}=V_{th3nominal}*(1-20\%)$, $V_{th4}=V_{th4nominal}*(1-20\%)$. The slope of separatrix been greatly altered.

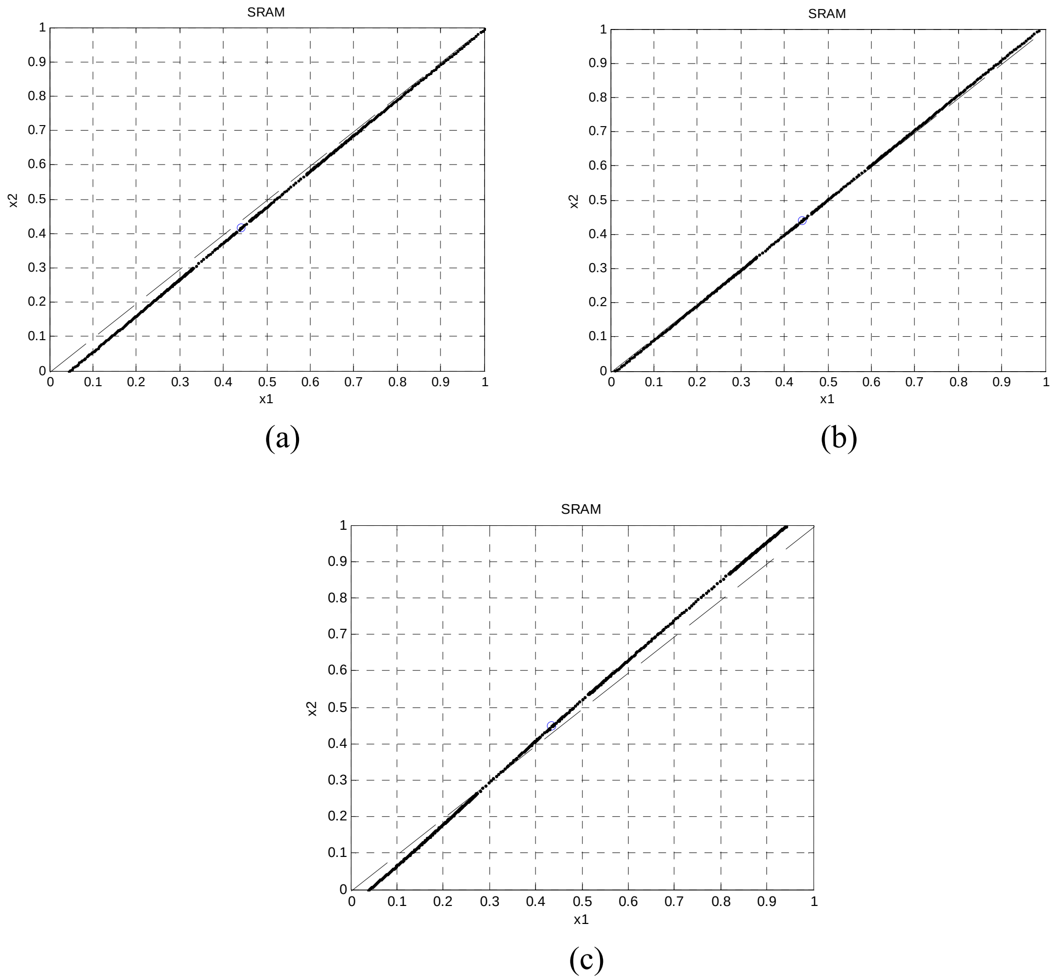


Fig. 3-18 A Single Parameter Change on (a) Threshold (V_{th1} Increase By 33%); (b) Capacitance (C_1 Increase By 5%); (c) More Than One Changes on Threshold Voltage.

3.8.3 Higher Order Effect

The higher order effect regards to curvature of separatrix.

This phenomenum occurs on curvature of separatrix often appears when the *K* values or threshold values are largely asymmetrically deviated. It appears to be that the separatrix would start to show nonlinearity if more than 50% deviation on a single threshold voltage or more than 15% on a *K* value. Since nonlinearity of separatrix is not showing so clear under varying a single parameter, we show a clear nonlinearity of separatrix in Fig. 3-19 by varying the thresholds in the manner of $V_{th1}=V_{th1nominal}*(1+n\%)$, $V_{th2}=V_{th2nominal}*(1+n\%)$, $V_{th3}=V_{th3nominal}*(1-n\%)$, $V_{th4}=V_{th4nominal}*(1-n\%)$, and Fig. 3-20 is varying *K* values in this manner, $K_1=K_{1nominal}*(1+n\%)$, $K_2=K_{2nominal}*(1+n\%)$, $V_{th3}=K_{3nominal}*(1-n\%)$, $K_4=K_{4nominal}*(1-n\%)$. By comparing those figures, variation of thresholds larger seems to give the separatrix an "S" shape, and large variation of *K* values gives the separatrix in a "C" curve shape. Depends on the combination of thresholds and *K* values, the separatrix can be "S" or mirrored "S" shape and "C" or mirrored "C" shape. If put together with the combinations from Fig. 3-19 and Fig. 3-20, the separatrix remains "C" shape for n=70 as shown in Fig. 3-21.

In summary, as the parameter deviation grows asymmetrically, the saddle point will change, and the separatrix would shift or change slope. Then, as the deviation grows slightly larger, it seems that the separatrix would have top and bottom segments with different linear regression slope but both segments are still looked linear. After that, the nonlinear distortion start to show up as the deviation becomes larger. When variation of thresholds become large, the "S" or mirrored "S" shape separatrix will be shown. If variations of *K* values are large, the separatrix shows a "C" or mirrored "C" shape. However, when thresholds and *K* values are varying in the same degree as show in Fig. 3-21, the separatrix remain "C" shape. Because of that, it seems that the effect to the separatrix on the variation of *K* values is greater than the variation of thresholds.

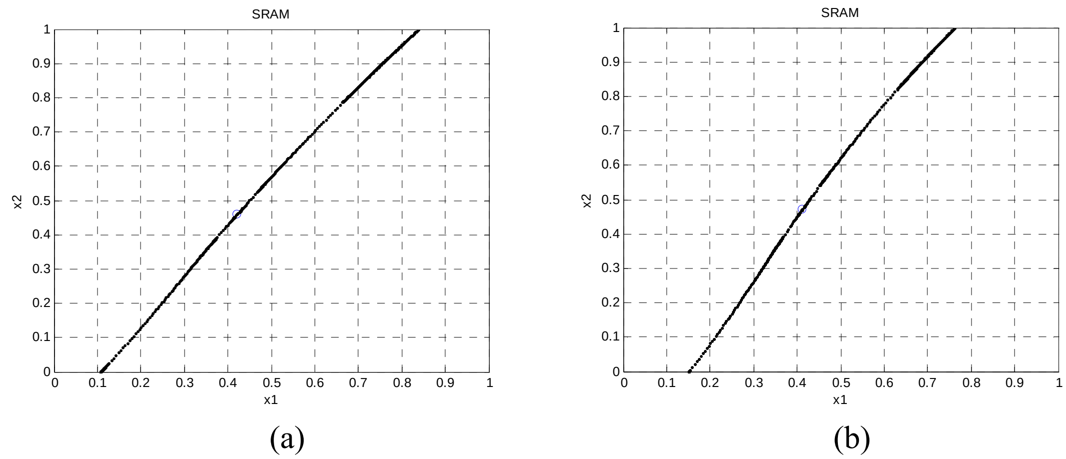


Fig. 3-19 Varying Only V_{th} Vvalues When n is (a) 60; (b) 80.

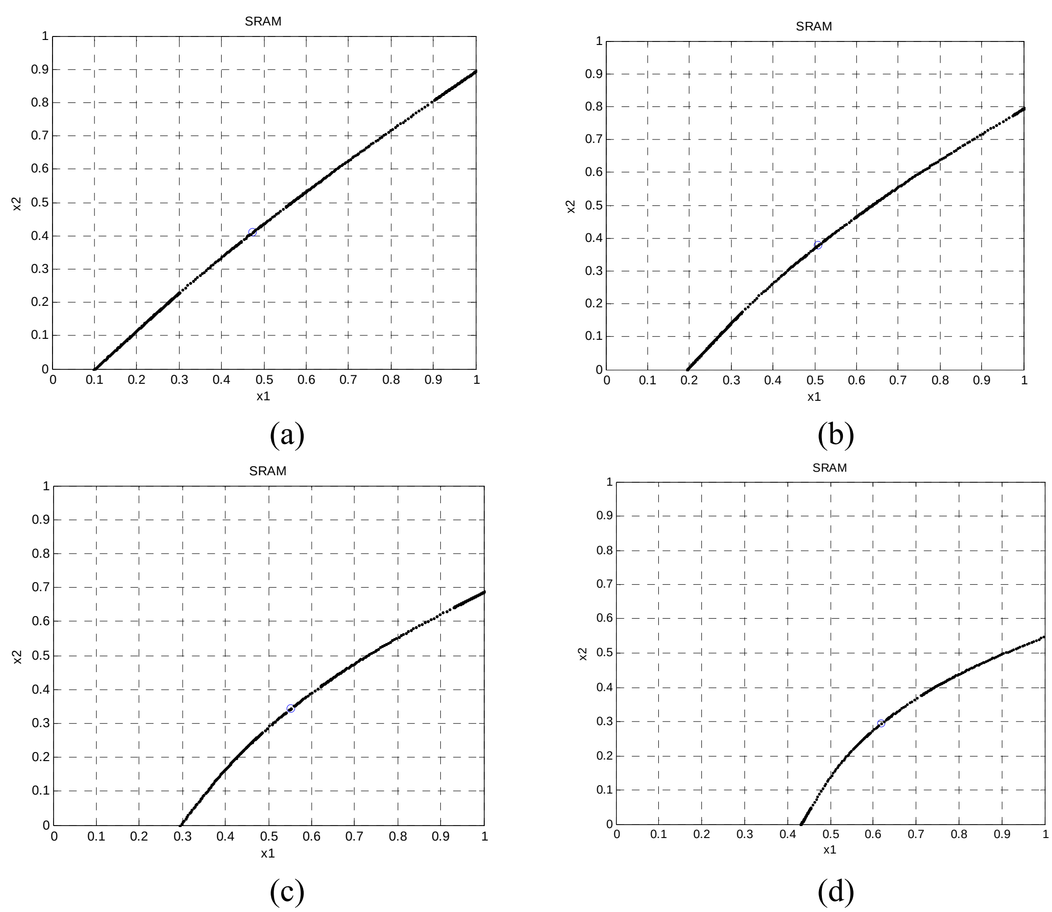


Fig. 3-20 Varying Only K Values When n is (a) 20; (b) 40; (c) 60; (d) 80.

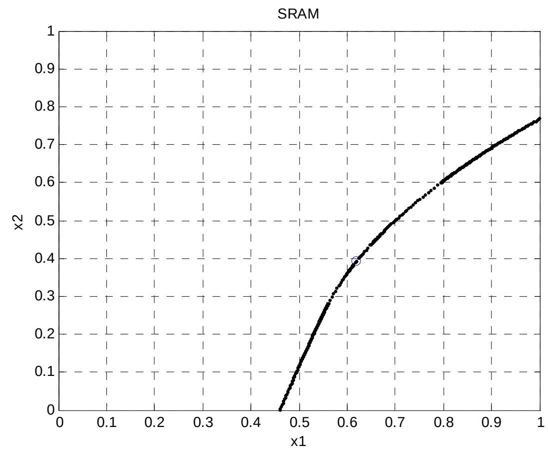


Fig. 3-21 Varying V_{th} and K Values at n=70.

CHAPTER IV

ANALYTICAL SOLUTIONS FOR SRAM DYNAMICAL NOISE MARGIN ANALYSIS

This chapter starts by introducing regions. For each region, we derived equilibrium point equations for the region. Then, we model noise disturbance as constant current source injected to either V_2 node (Case I) or V_1 node (Case II), and derive the equilibrium point equation in terms of I_{noise}. Lastly, bifurcation point and noise margin I_c are solved analytically.

4.1. INTRODUCTION ON DEFIED REGIONS

To find analytical solutions for our L-1 model, we simplify our equations by introducing regions based on the thresold parameters. We partition our state space into twenty three disjoint areas as shown in the figure below. Each area on the state space is a Region that is assigned by a number as shown in Fig. 4-1.

The equations that define the borderline of regions come from the SRAM modeling equations. Recall the SRAM modeling equations (4.1) (4.2) from chapter II, $\dot{V}_1$ and $\dot{V}_2$ equations can be written to combination of $S^2()$ terms. Assume $\dot{V}_1$=0, one possibility is four $S()$ terms are zero.

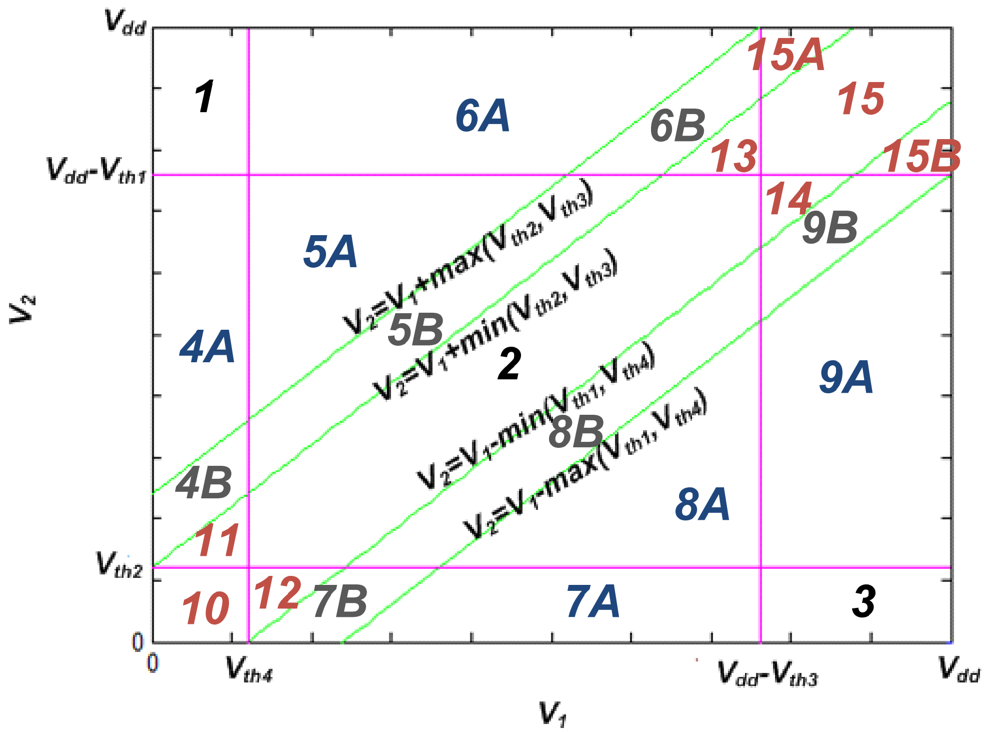


Fig. 4-1 The Region Plot

$$\dot{V}_1 = \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] - \frac{1}{C_1} K_2 [S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] \quad (4.1)$$

$$\dot{V}_2 = \frac{1}{C_2} K_3 [S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] - \frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] \quad (4.2)$$

$On\ \dot{V}_1:$

$$\begin{matrix} \frac{1}{C_1}K_1S^2(V_{dd}-V_2-Vth_1)=0 \\ -\frac{1}{C_1}K_1S^2(V_1-V_2-Vth_1)=0 \\ -\frac{1}{C_1}K_2S^2(V_2-Vth_2)=0 \\ \frac{1}{C_1}K_2S^2(V_2-V_1-Vth_2)=0 \end{matrix} \Rightarrow \begin{matrix} V_{dd}-V_2-Vth_1=0 \\ V_1-V_2-Vth_1=0 \\ V_2-Vth_2=0 \\ V_2-V_1-Vth_2=0 \end{matrix}$$

$$\Rightarrow \begin{cases} V_2=V_{dd}-Vth_1 \\ V_2=V_1-Vth_1 \\ V_2=Vth_2 \\ V_2=V_1+Vth_2 \end{cases} \qquad (4.\ 3)$$

$On\ \dot{V}_2:$

$$\begin{matrix} \frac{1}{C_2}K_3S^2(V_{dd}-V_1-Vth_3)=0 \\ -\frac{1}{C_2}K_3S^2(V_2-V_1-Vth_3)=0 \\ -\frac{1}{C_2}K_4S^2(V_1-Vth_4)=0 \\ \frac{1}{C_2}K_4S^2(V_1-V_2-Vth_4)=0 \end{matrix} \Rightarrow \begin{matrix} V_{dd}-V_1-Vth_3=0 \\ V_2-V_1-Vth_3=0 \\ V_1-Vth_4=0 \\ V_1-V_2-Vth_4=0 \end{matrix}$$

$$\Rightarrow \begin{cases} V_1=V_{dd}-Vth_3 \\ V_2=V_1+Vth_3 \\ V_1=Vth_4 \\ V_2=V_1-Vth_4 \end{cases} \qquad (4.\ 4)$$

The set of equations in (4.3) and (4.4) construct the region boundary lines in Fig4-1. In (4.3) and (4.4), there has two similar equations, $V_2=V_1-Vth_1$ and $V_2=V_1-Vth_4$. If V_{th1} and V_{th4} are the same, these two equations will merge into one line, and region 4B 5B 6B 7B 8B 9B will not exist. When all four transistor threshold voltages are the same, the region plot would look like Fig. 4-2.

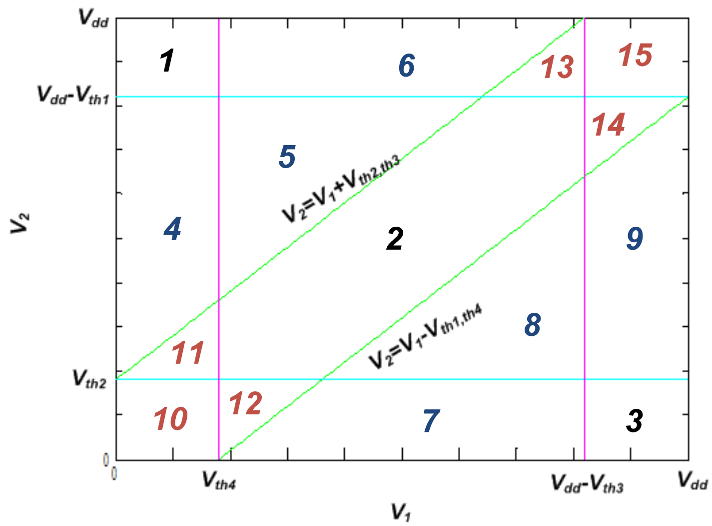


Fig. 4-2 The Region Plot When All Threshold Voltages Are The Same.

Our region partition can simplify our analysis. Recall the previous example, the term $S^2(V_{dd} - V_2 - Vth_1)$ is zero if $V_{dd} - V_2 - Vth_1 \leq 0$. Therefore, inequality $V_2 \geq V_{dd} - Vth_1$ would make $S^2(V_{dd} - V_2 - Vth_1)$ zero. The cooresponding region on the state space would be the top shaded area. It covers region 1, 6A, 6B, 13, 15A, 15 and 15B as shown in Fig. 4-3.

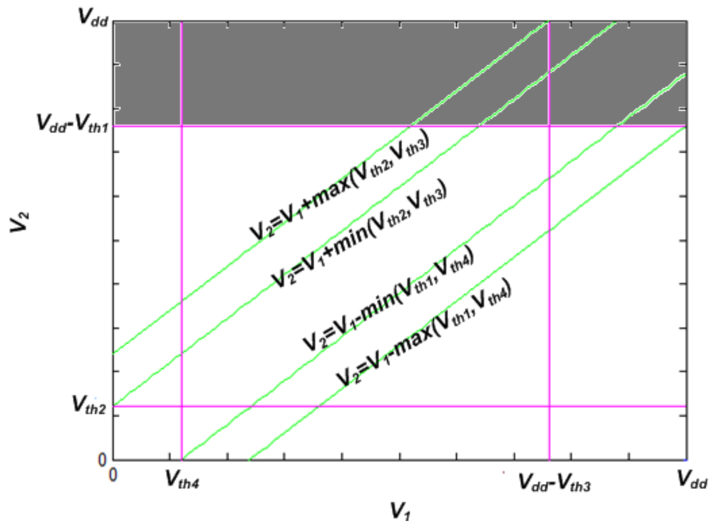


Fig. 4-3 Dark Regions Would Not Have Influence By The Term $S^2(V_{dd}-V_2-V_{th1})$

The modeling equation can be simplified by dropping this term $S^2(V_{dd} - V_2 - Vth_1)$ when we explore dynamics in the shaded region, in which the $\dot{V}_1$ equation would be simplified as:

$$\dot{V}_1 = -\frac{1}{C_1} K_1 S^2 (V_1 - V_2 - Vth_1) - \frac{1}{C_1} K_2 [S^2 (V_2 - Vth_2) - S^2 (V_2 - V_1 - Vth_2)] \tag{4. 5}$$

Accordingly, analytical solutions on equilibrium points can be easily obtained.

The other term on $\dot{V}_1$ equation, $S^2(V_1 - V_2 - Vth_1)$, would be zero if $V_1 - V_2 - Vth_1 \le 0$, and the condition would be $V_1 - Vth_1 \le V_2$. In a similar manner, $\dot{V}_2$ equation has the term $S^2(V_1 - V_2 - Vth_4)$, the condition to make the term zero is $V_1 - Vth_4 \le V_2$. Assuming threshold values V_{th1} and V_{th4} are unknown. Without losing generality, the expression written as $V_2 \geq V_1 - \max(Vth_1, Vth_4)$ guarantee satisfying both $V_1 - Vth_1 \leq V_2$ and $V_1 - Vth_4 \leq V_2$. The rest of borderline equations are built similarly.

4.2. THE PHYSICAL INTERPRETATION OF REGIONS

Each region corresponds to a transistor operation mode. For a T4 SRAM, we number left inverter PMOS as (M1), left inverter NMOS as (M2), right inverter PMOS as (M3) and right inverter NMOS as (M4). Table 4 summaries the transistor mode for each region.

The SRAM transistor's modes associate with each region can be determined from the system modeling equations. Each region relates to certain *S(.)* terms, that means, some of the *S(.)* terms survive inside that region and some doesn't. From the survived *S(.)* terms inside a particular region, its transistor's mode can be rigorously proved from basic MOS properties. Here give a sample proof showing that the transistor's modes associated inside region 2 are saturation mode. The following theorem states that the SRAM with all transistors saturated defines the region 2.

TABLE 4
Physical Interpretation of Regions

	M1	M2	M3	M4
Region 1	CUT	LIN	LIN	CUT
Region 2	SAT	SAT	SAT	SAT
Region 3	LIN	CUT	CUT	LIN
Region 4A	SAT	LIN	LIN	CUT
Region 4B	SAT	LIN (Vth3>Vth2)	SAT (Vth3>Vth2)	CUT
Region 4B	SAT	SAT (Vth2>Vth3)	LIN (Vth2>Vth3)	CUT
Region 5A	SAT	LIN	LIN	SAT
Region 5B	SAT	LIN (Vth3>Vth2)	SAT (Vth3>Vth2)	SAT
Region 5B	SAT	SAT (Vth2>Vth3)	LIN (Vth2>Vth3)	SAT
Region 6A	CUT	LIN	LIN	SAT
Region 6B	CUT	LIN (Vth3>Vth2)	SAT (Vth3>Vth2)	SAT
Region 6B	CUT	SAT (Vth2>Vth3)	LIN (Vth2>Vth3)	SAT
Region 7A	LIN	CUT	SAT	LIN
Region 7B	SAT (Vth1>Vth4)	CUT	SAT	LIN (Vth4>Vth1)
Region 7B	LIN (Vth4>Vth1)	CUT	SAT	SAT (Vth1>Vth4)
Region 8A	LIN	SAT	SAT	LIN
Region 8B	SAT (Vth1>Vth4)	SAT	SAT	LIN (Vth4>Vth1)
Region 8B	LIN (Vth4>Vth1)	SAT	SAT	SAT (Vth1>Vth4)
Region 9A	LIN	SAT	CUT	LIN
Region 9B	SAT (Vth1>Vth4)	SAT	CUT	LIN (Vth4>Vth1)
Region 9B	LIN (Vth4>Vth1)	SAT	CUT	SAT (Vth1>Vth4)
Region 10	SAT	CUT	SAT	CUT
Region 11	SAT	SAT	SAT	CUT
Region 12	SAT	CUT	SAT	SAT
Region 13	CUT	SAT	SAT	SAT
Region 14	SAT	SAT	CUT	SAT
Region 15	CUT	SAT	CUT	SAT
Region 15A	CUT	SAT (Vth2>Vth3)	CUT	SAT
Region 15A	CUT	LIN (Vth3>Vth2)	CUT	SAT
Region 15B	CUT	SAT	CUT	LIN (Vth1>Vth4)
Region 15B	CUT	SAT	CUT	SAT (Vth4>Vth1)

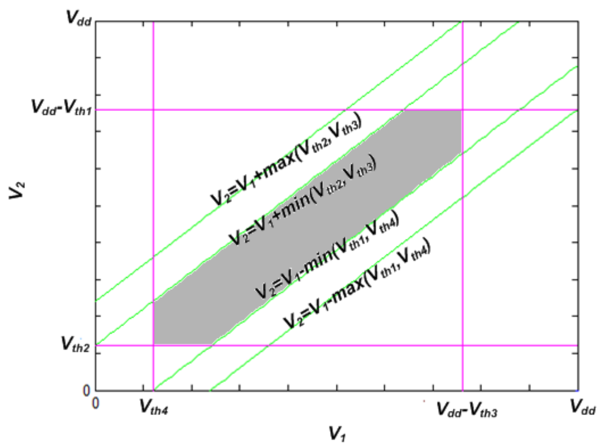
Assumption for theorem 4.2:

Assume all SRAM transistors are operating in the saturation mode.

Theorem 4.2:

For a L-1 model T4 SRAM, when all transistors are operating in the saturation mode,its SRAM state defines region 2, which is defined by the following six inequality constraints

:



1. $V_2 > V_1 - \min(V_{th1}, V_{th4})$
2. $V_2 < V_1 + \min(V_{th2}, V_{th3})$
3. $V_1 < V_{dd} - V_{th3}$
4. $V_1 > V_{th4}$
5. $V_2 > V_{th2}$
6. $V_2 > V_{dd} - V_{th1}$

Fig. 4-4 Region 2 Plot and Its 6 Inequalities

Proof:

For MOS transistors to operate at saturation mode, it must satisfy conditions from Table 1 in Chapter I:

NMOS:

$$V_{gs} > V_{thn} \tag{4. 6}$$

$$V_{ds} > V_{gs} - V_{thn} \tag{4. 7}$$

PMOS:

$$V_{sg} > |V_{thp}| \tag{4. 8}$$

$$V_{sd} > V_{sg} - |V_{thp}| \tag{4. 9}$$

Accordingly, (4.6) implies that $V_1 > V_{th4}$ and $V_2 > V_{th2}$ (inequalities 4 and 5 from Fig. 4-4) because V_1 is the gate voltage to (M4) NMOS and V_2 is the gate voltage to (M2) NMOS. From (4.8), V_{sg} for (M3) PMOS is V_{dd}-V_1 when looking into the right inverter, and V_{sg} for (M1) PMOS is V_{dd}-V_2 when looking into the left inverter. The (4.10) take V_1 as gate voltage when looking into right inverter number 3 PMOS, and (4.11) take V_2 as gate voltage looking into left inverter (M1) PMOS. Those inequalities can be rewritten as inequalities 3 and 6.

$$V_{sg} > |V_{thp}| \Rightarrow \textit{look in right inverter}$$
$$V_{dd} - V_1 > V_{th3}$$

$$V_1 < V_{dd} - V_{th3} \tag{4. 10}$$

$$V_{sg} > |V_{thp}| \Rightarrow \textit{look in left inverter}$$
$$V_{dd} - V_2 > V_{th1}$$

$$V_2 < V_{dd} - V_{th1} \tag{4. 11}$$

Inequalities 1 and 2 are derived from (4.7) and (4.9). After substituting variables looking to left and right inverter as describe above, we obtain four inequalities: (4.12) (4.13) (4.14) and (4.15). However, notice that (4.12) (4.15) and (4.13) (4.14) can be the same if thresholds are all equal. For all transistors to operate in saturation mode, all four inequalties must be satisfied. Therefore, we can combine (4.12) (4.15) together, and

(4.13) (4.14) together. Accordingly, V_1 minus V_{th4} must be less than V_2 . V_1 munus V_{th1} must be less than V_2. Then V_1 minus minimum of V_{th1} V_{th4} less than V_2 will satisfy both inequalities. That is $V_2 > V_1 - \min(V_{th1}, V_{th4})$ which is inequality 1. Similarly, inequality 2, $V_2 < V_1 + \min(V_{th2}, V_{th3})$, is the combined version of (4.13) and (4.4).

$$V_{ds} > V_{gs} - V_{thn} \Rightarrow look\ in\ right\ inverter$$

$$V_2 > V_1 - V_{th4} \tag{4. 12}$$

$$V_{ds} > V_{gs} - V_{thn} \Rightarrow look\ in\ left\ inverter$$
$$V_1 > V_2 - V_{th2}$$

$$V_2 < V_1 + V_{th2} \tag{4. 13}$$

$$V_{sd} > V_{sg} - |V_{thp}| \Rightarrow look\ in\ right\ inverter$$
$$V_{dd} - V_2 > V_{dd} - V_1 - V_{th3}$$
$$-V_2 > -V_1 - V_{th3}$$

$$V_2 < V_1 + V_{th3} \tag{4. 14}$$

$$V_{sd} > V_{sg} - |V_{thp}| \Rightarrow look\ in\ left\ inverter$$
$$V_{dd} - V_1 > V_{dd} - V_2 - V_{th1}$$
$$-V_1 > -V_2 - V_{th1}$$

$$V_2 > V_1 - V_{th1} \tag{4. 15}$$

4.3. THE STABLE NODAL EQUILIBRIA AND UNSTABLE SADDLE

Theorem 4.3.1:

There are three equilibria for L-1 modeling SRAM:

(V_{dd}, 0)

(0, V_{dd})

$$\left(\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}},\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}}\right)$$

Proof:

Each equilibrium point must satisfies $\dot{V}_1=0$ and $\dot{V}_2=0$ from (4.1) (4.2).

1. (V_{dd}, 0) is one of the equilibria:

In region 1, the original equations

$$\begin{cases}\dot{V}_1=-\dfrac{1}{C_1}K_2[S^2(V_2-Vth_2)-S^2(V_2-V_1-Vth_2)]\\ \dot{V}_2=\dfrac{1}{C_2}K_3[S^2(V_{dd}-V_1-Vth_3)-S^2(V_2-V_1-Vth_3)]\end{cases} \tag{4. 16}$$

are reduced to the following equations to find its equilibrium point

$$\begin{cases}-\dfrac{1}{C_1}K_2[(V_2-Vth_2)^2-(V_2-V_1-Vth_2)^2]=0\\ \dfrac{1}{C_2}K_3[(V_{dd}-V_1-Vth_3)^2-(V_2-V_1-Vth_3)^2]=0\end{cases}$$

$$\begin{cases}(V_2-Vth_2)^2-(V_2-V_1-Vth_2)^2=0\\ (V_{dd}-V_1-Vth_3)^2-(V_2-V_1-Vth_3)^2=0\end{cases}$$

$$\begin{cases}(V_2-Vth_2)=(V_2-V_1-Vth_2)\\ (V_{dd}-V_1-Vth_3)=(V_2-V_1-Vth_3)\end{cases}$$

$$\begin{aligned}&(V_{dd}-V_1-Vth_3)=(V_2-V_1-Vth_3)\\ &\Rightarrow V_2=V_{dd}\end{aligned}$$

$$\begin{aligned}&(V_2-Vth_2)=(V_2-V_1-Vth_2)\\ &(V_{dd}-Vth_2)=(V_{dd}-V_1-Vth_2)\\ &\Rightarrow V_1=0\end{aligned}$$

2. (0,V_{dd}) is another equilibrium point:

Similarly, in region 3:

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [S^2 (V_{dd} - V_2 - Vth_1) - S^2 (V_1 - V_2 - Vth_1)] \\ \dot{V}_2 = -\frac{1}{C_2} K_4 [S^2 (V_1 - Vth_4) - S^2 (V_1 - V_2 - Vth_4)] \end{cases} \quad (4.\ 17)$$

$$\begin{cases} \frac{1}{C_1} K_1 [(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] = 0 \\ -\frac{1}{C_2} K_4 [(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] = 0 \end{cases}$$

$$\begin{cases} (V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2 = 0 \\ (V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2 = 0 \end{cases}$$

$$\begin{cases} (V_{dd} - V_2 - Vth_1) = (V_1 - V_2 - Vth_1) \\ (V_1 - Vth_4) = (V_1 - V_2 - Vth_4) \end{cases}$$

$$(V_{dd} - V_2 - Vth_1) = (V_1 - V_2 - Vth_1)$$
$$\Rightarrow V_1 = V_{dd}$$

$$(V_1 - Vth_4) = (V_1 - V_2 - Vth_4)$$
$$(V_{dd} - Vth_4) = (V_{dd} - V_2 - Vth_4)$$
$$\Rightarrow V_2 = 0$$

3. On the third equilibrium point:

In region 2:

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [S^2 (V_{dd} - V_2 - Vth_1)] - \frac{1}{C_1} K_2 [S^2 (V_2 - Vth_2)] \\ \dot{V}_2 = \frac{1}{C_2} K_3 [S^2 (V_{dd} - V_1 - Vth_3)] - \frac{1}{C_2} K_4 [S^2 (V_1 - Vth_4)] \end{cases} \quad (4.\ 18)$$

$$\begin{cases} \frac{1}{C_1} K_1 (V_{dd} - V_2 - Vth_1)^2 - \frac{1}{C_1} K_2 (V_2 - Vth_2)^2 = 0 \\ \frac{1}{C_2} K_3 (V_{dd} - V_1 - Vth_3)^2 - \frac{1}{C_2} K_4 (V_1 - Vth_4)^2 = 0 \end{cases}$$

$$\begin{cases} K_1 (V_{dd} - V_2 - Vth_1)^2 = K_2 (V_2 - Vth_2)^2 \\ K_3 (V_{dd} - V_1 - Vth_3)^2 = K_4 (V_1 - Vth_4)^2 \end{cases}$$

Since $V_{dd} - V_2 - Vth_1$ and $V_2 - Vth_2$ terms are positive, the result comes out the square root would be positive.

$$\begin{cases} \sqrt{K_1}(V_{dd} - V_2 - Vth_1) = \sqrt{K_2}(V_2 - Vth_2) \\ \sqrt{K_3}(V_{dd} - V_1 - Vth_3) = \sqrt{K_4}(V_1 - Vth_4) \end{cases}$$

$$\begin{cases} \sqrt{K_1}(V_{dd} - Vth_1) - \sqrt{K_1}V_2 = \sqrt{K_2}V_2 - \sqrt{K_2}Vth_2 \\ \sqrt{K_3}(V_{dd} - Vth_3) - \sqrt{K_3}V_1 = \sqrt{K_4}V_1 - \sqrt{K_4}Vth_4 \end{cases}$$

$$\begin{cases} \sqrt{K_1}(V_{dd} - Vth_1) + \sqrt{K_2}Vth_2 = (\sqrt{K_1}\sqrt{K_2}) \cdot V_2 \\ \sqrt{K_3}(V_{dd} - Vth_3) + \sqrt{K_4}Vth_4 = (\sqrt{K_3} + \sqrt{K_4}) \cdot V_1 \end{cases}$$

$$\begin{cases} V_1 = \dfrac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} \\ V_2 = \dfrac{\sqrt{K_1} * (V_{dd} - Vth_1) + \sqrt{K_2} * Vth_2}{\sqrt{K_1} + \sqrt{K_2}} \end{cases}$$

The equilibria of SRAM only exist in region 1 2 and 3. We cannot find a solution satisfies $\dot{V}_1 = 0 \;\; \dot{V}_2 = 0$ for all other regions.

Theorem 4.3.2:

The eigenvalue formula for the three equilibria are:

1. *The eigenvalue for* $(V_{dd}, 0)$:

$$\lambda_1 = -\frac{2K_2}{C_1}(V_{dd} - V_{th2}), -\frac{2K_3}{C_2}(V_{dd} - V_{th3})$$

2. *The eigenvalue for* $(0, V_{dd})$:

$$\lambda_3 = -\frac{2K_1}{C_1}(V_{dd} - V_{th1}), -\frac{2K_4}{C_2}(V_{dd} - V_{th4})$$

3. *The eigenvalue for* $(\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}}, \frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}})$:

$$\lambda_2 = \pm\sqrt{\frac{4\sqrt{K_1 \cdot K_2 \cdot K_3 \cdot K_4}(V_{dd}-Vth_1-Vth_2)\cdot(V_{dd}-Vth_3-Vth_4)}{C_1 C_2}}$$

Proof:

1. Proof for the eigenvalue of *(V_{dd}, 0)*

$$\begin{cases} \dot{V}_1 = -\frac{1}{C_1}K_2[S^2(V_2-Vth_2)-S^2(V_2-V_1-Vth_2)] \\ \dot{V}_2 = \frac{1}{C_2}K_3[S^2(V_{dd}-V_1-Vth_3)-S^2(V_2-V_1-Vth_3)] \end{cases}$$

$$\begin{cases} \dot{V}_1 = -\frac{1}{C_1}K_2[(V_2-Vth_2)^2-(V_2-V_1-Vth_2)^2] = f_1(V_1,V_2) \\ \dot{V}_2 = \frac{1}{C_2}K_3[(V_{dd}-V_1-Vth_3)^2-(V_2-V_1-Vth_3)^2] = g_1(V_1,V_2) \end{cases}$$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial V_1} & \frac{\partial f_1}{\partial V_2} \\ \frac{\partial g_1}{\partial V_1} & \frac{\partial g_1}{\partial V_2} \end{bmatrix}\Bigg|_{\substack{V_1=V_{dd}\\V_2=0}} = \begin{bmatrix} -\frac{2K_2}{C_1}(V_2-V_1-V_{th2}) & -\frac{2K_2}{C_1}V_1 \\ \frac{2K_2}{C_1}(V_2-V_{dd}) & -\frac{2K_3}{C_2}(V_2-V_1-V_{th3}) \end{bmatrix}\Bigg|_{\substack{V_1=V_{dd}\\V_2=0}}$$

$$= \begin{bmatrix} -\frac{2K_2}{C_1}(V_{dd}-V_{th2}) & 0 \\ 0 & -\frac{2K_3}{C_2}(V_{dd}-V_{th3}) \end{bmatrix}$$

The eigenvalues are obtained as follows:

$$\det(\lambda_1 I - J) = 0$$

$$\det(\begin{bmatrix} \lambda_1 + \frac{2K_2}{C_1}(V_{dd} - V_{th2}) & 0 \\ 0 & \lambda_1 + \frac{2K_3}{C_2}(V_{dd} - V_{th3}) \end{bmatrix}) = 0$$

$$[\lambda_1 + \frac{2K_2}{C_1}(V_{dd} - V_{th2})] \cdot [\lambda_1 + \frac{2K_3}{C_2}(V_{dd} - V_{th3})] = 0$$

$$\lambda_1 = -\frac{2K_2}{C_1}(V_{dd} - V_{th2}), -\frac{2K_3}{C_2}(V_{dd} - V_{th3})$$

2. Derive the eigenvalues of *(0, V_{dd})*

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \dot{V}_2 = -\frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] \end{cases}$$

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] = f_2(V_1, V_2) \\ \dot{V}_2 = -\frac{1}{C_2} K_4 [(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] = g_2(V_1, V_2) \end{cases}$$

$$J = \left. \begin{bmatrix} \frac{\partial f_2}{\partial V_1} & \frac{\partial f_2}{\partial V_2} \\ \frac{\partial g_2}{\partial V_1} & \frac{\partial g_2}{\partial V_2} \end{bmatrix} \right|_{\substack{V_1=0 \\ V_2=V_{dd}}} = \left. \begin{bmatrix} -\frac{2K_1}{C_1}(V_1 - V_2 - V_{th1}) & -\frac{2K_1}{C_1}(V_1 - V_{dd}) \\ -\frac{2K_2}{C_1}(V_2) & -\frac{2K_4}{C_2}(V_1 - V_2 - V_{th4}) \end{bmatrix} \right|_{\substack{V_1=0 \\ V_2=V_{dd}}}$$

$$= \begin{bmatrix} -\frac{2K_1}{C_1}(V_{dd} - V_{th1}) & 0 \\ 0 & -\frac{2K_4}{C_1}(V_{dd} - V_{th4}) \end{bmatrix}$$

$$\det(\lambda_3 I - J) = 0$$

$$\det\left(\begin{bmatrix} \lambda_3 + \frac{2K_1}{C_1}(V_{dd} - V_{th1}) & 0 \\ 0 & \lambda_3 + \frac{2K_4}{C_1}(V_{dd} - V_{th4}) \end{bmatrix}\right) = 0$$

$$[\lambda_3 + \frac{2K_1}{C_1}(V_{dd} - V_{th1})] \cdot [\lambda_3 + \frac{2K_4}{C_1}(V_{dd} - V_{th4})] = 0$$

$$\lambda_3 = -\frac{2K_1}{C_1}(V_{dd} - V_{th1}), -\frac{2K_4}{C_2}(V_{dd} - V_{th4})$$

3. Derive the eigenvalues of third equilibrium:

From the nonlinear equation (4.1) (4.2), focus on the equilibrium point inside region 2, 4 terms can be eliminated by the concept of defined region: $S^2(V_1 - V_2 - Vth_1)$, $S^2(V_2 - V_1 - Vth_2)$, $S^2(V_2 - V_1 - Vth_3)$, $S^2(V_1 - V_2 - Vth_4)$. The differential equation then become:

$$\begin{cases} \dot{V}_1 = f_s(v_1, v_2) = \frac{K_1}{C_1}(V_{dd} - V_2 - Vth_1)^2 - \frac{K_2}{C_1}(V_2 - Vth_2)^2 \\ \dot{V}_2 = g_s(v_1, v_2) = \frac{K_3}{C_2}(V_{dd} - V_1 - Vth_3)^2 - \frac{K_4}{C_2}(V_1 - Vth_4)^2 \end{cases}$$

Find the Jacobian matrix and evaluate at the third equilibrium point.

$$J = \left.\begin{bmatrix} \frac{\partial f_s}{\partial V_1} & \frac{\partial f_s}{\partial V_2} \\ \frac{\partial g_s}{\partial V_1} & \frac{\partial g_s}{\partial V_2} \end{bmatrix}\right|$$

$$\begin{aligned} \frac{\partial f_s}{\partial V_1} &= \frac{\partial}{\partial V_1}\left(\frac{K_1}{C_1}(V_{dd} - V_2 - Vth_1)^2 - \frac{K_2}{C_1}(V_2 - Vth_2)^2\right) \\ &= 0 \end{aligned}$$

$$\frac{\partial g_s}{\partial V_2} = \frac{\partial}{\partial V_2}(\frac{K_3}{C_2}(V_{dd} - V_1 - Vth_3)^2 - \frac{K_4}{C_2}(V_1 - Vth_4)^2)$$
$$= 0$$

$$\left.\frac{\partial f_s}{\partial V_2}\right| = -2\frac{K_1}{C_1}(V_{dd} - V_2 - Vth_1) - 2\frac{K_2}{C_1}(V_2 - Vth_2)$$
$$= -\frac{2}{C_1}[K_1(V_{dd} - \frac{\sqrt{K_1}*(V_{dd} - Vth_1) + \sqrt{K_2}*Vth_2}{\sqrt{K_1} + \sqrt{K_2}} - V_{th1})$$
$$+ K_2(\frac{\sqrt{K_1}*(V_{dd} - Vth_1) + \sqrt{K_2}*Vth_2}{\sqrt{K_1} + \sqrt{K_2}} - Vth_2)]$$
$$= -\frac{2}{C_1}[K_1(\frac{(\sqrt{K_1} + \sqrt{K_2})(V_{dd} - V_{th1}) - \sqrt{K_1}*(V_{dd} - V_{th1}) - \sqrt{K_2}*Vth_2}{\sqrt{K_1} + \sqrt{K_2}})$$
$$+ K_2(\frac{\sqrt{K_1}*(V_{dd} - Vth_1) + \sqrt{K_2}*Vth_2 - (\sqrt{K_1} + \sqrt{K_2})Vth_2}{\sqrt{K_1} + \sqrt{K_2}})]$$
$$= -\frac{2}{C_1}[K_1(\frac{\sqrt{K_2}(V_{dd} - V_{th1}) - \sqrt{K_2}*Vth_2}{\sqrt{K_1} + \sqrt{K_2}}) + K_2(\frac{\sqrt{K_1}*(V_{dd} - Vth_1) - \sqrt{K_1}Vth_2}{\sqrt{K_1} + \sqrt{K_2}})]$$

$$= -\frac{2}{C_1}[K_1(\frac{\sqrt{K_2}(V_{dd} - V_{th1} - Vth_2)}{\sqrt{K_1} + \sqrt{K_2}}) + K_2(\frac{\sqrt{K_1}*(V_{dd} - Vth_1 - Vth_2)}{\sqrt{K_1} + \sqrt{K_2}})]$$
$$= -\frac{2}{C_1}[\frac{K_1\sqrt{K_2}(V_{dd} - V_{th1} - Vth_2) + K_2\sqrt{K_1}*(V_{dd} - Vth_1 - Vth_2)}{\sqrt{K_1} + \sqrt{K_2}}]$$
$$= -\frac{2}{C_1}[\frac{(K_1\sqrt{K_2} + K_2\sqrt{K_1})}{\sqrt{K_1} + \sqrt{K_2}}(V_{dd} - Vth_1 - Vth_2)]$$
$$= -\frac{2}{C_1}[\frac{(\sqrt{K_1}\sqrt{K_2})\cdot(\sqrt{K_2} + \sqrt{K_1})}{\sqrt{K_1} + \sqrt{K_2}}(V_{dd} - Vth_1 - Vth_2)]$$
$$= -\frac{2}{C_1}[(\sqrt{K_1}\sqrt{K_2})\cdot(V_{dd} - Vth_1 - Vth_2)]$$

$$\left.\frac{\partial g_s}{\partial V_1}\right| = -2\frac{K_3}{C_2}(V_{dd} - V_1 - Vth_3) - 2\frac{K_4}{C_2}(V_1 - Vth_4)$$

$$= -\frac{2}{C_2}[K_3(V_{dd} - \frac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} - Vth_3)$$

$$+ K_4(\frac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} - Vth_4)]$$

$$= -\frac{2}{C_2}[K_3(\frac{(\sqrt{K_3} + \sqrt{K_4})(V_{dd} - Vth_3)}{\sqrt{K_3} + \sqrt{K_4}} - \frac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}})$$

$$+ K_4(\frac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} - \frac{(\sqrt{K_3} + \sqrt{K_4})Vth_4}{\sqrt{K_3} + \sqrt{K_4}})]$$

$$= -\frac{2}{C_2}[K_3(\frac{\sqrt{K_4}(V_{dd} - Vth_3) - \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}}) + K_4(\frac{\sqrt{K_3}(V_{dd} - Vth_3) - \sqrt{K_3}Vth_4}{\sqrt{K_3} + \sqrt{K_4}})]$$

$$= -\frac{2}{C_2}[\frac{K_3\sqrt{K_4}(V_{dd} - Vth_3 - Vth_4) + K_4\sqrt{K_3}(V_{dd} - Vth_3 - Vth_4)}{\sqrt{K_3} + \sqrt{K_4}}]$$

$$= -\frac{2}{C_2}[\frac{(K_3\sqrt{K_4} + K_4\sqrt{K_3}) \cdot (V_{dd} - Vth_3 - Vth_4)}{\sqrt{K_3} + \sqrt{K_4}}]$$

$$= -\frac{2}{C_2}[\frac{(\sqrt{K_3} + \sqrt{K_4}) \cdot (\sqrt{K_4} + \sqrt{K_3}) \cdot (V_{dd} - Vth_3 - Vth_4)}{\sqrt{K_3} + \sqrt{K_4}}]$$

$$= -\frac{2}{C_2}[\frac{(\sqrt{K_3} + \sqrt{K_4}) \cdot (\sqrt{K_4} + \sqrt{K_3}) \cdot (V_{dd} - Vth_3 - Vth_4)}{\sqrt{K_3} + \sqrt{K_4}}]$$

$$= -\frac{2}{C_2}[(\sqrt{K_3} + \sqrt{K_4}) \cdot (V_{dd} - Vth_3 - Vth_4)$$

The Jacobian matrix can be written in the form:

$$J = \begin{bmatrix} 0 & -A \\ -B & 0 \end{bmatrix} \qquad (4.\ 19)$$

Where

$$A = 2\frac{\sqrt{K_1} * \sqrt{K_2}}{C_1}(V_{dd} - Vth_1 - Vth_2)$$
$$B = 2\frac{\sqrt{K_3} * \sqrt{K_4}}{C_2}(V_{dd} - Vth_3 - Vth_4) \quad (4.\ 20)$$

Thus, the eigenvalues are derived:

$$\det(\lambda_2 I - J) = 0$$
$$\det(\begin{bmatrix} \lambda_2 & A \\ B & \lambda_2 \end{bmatrix}) = 0$$
$$\lambda_2{}^2 = A \cdot B$$

$$\lambda = \pm\sqrt{A \cdot B}$$
$$= \left(\pm\sqrt{\frac{4}{C_1 C_2}(\sqrt{K_1 \cdot K_2 \cdot K_3 \cdot K_4}(V_{dd} - Vth_1 - Vth_2) \cdot (V_{dd} - Vth_3 - Vth_4)} \right)$$

Theorem 4.3.3:

Among the three equilibria for SRAM, (V_{dd},0) and (0, V_{dd}) are stable nodes, and $\left(\frac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}}, \frac{\sqrt{K_1} * (V_{dd} - Vth_1) + \sqrt{K_2} * Vth_2}{\sqrt{K_1} + \sqrt{K_2}}\right)$ *is a saddle denoted as*:

$$\begin{cases} V_{1saddle} = \dfrac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} \\ V_{2saddle} = \dfrac{\sqrt{K_1} * (V_{dd} - Vth_1) + \sqrt{K_2} * Vth_2}{\sqrt{K_1} + \sqrt{K_2}} \end{cases} \quad (4.\ 21)$$

Proof:

(V_{dd},0) and *(0, V_{dd})* are stable nodes because their eigenvalues are negative, and $(\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}},\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}})$ has one positive and one negative eigenvalue proved by theorem 4.3.2. Thus, it's a saddle.

In summary, from the theorems derived in this section, we find three equilibria located in region 1, 2 and 3 for a typical SRAM. Two are stable equilibrium states at (V_{dd},0) and (0,V_{dd}) in regions one and three; and one is an unstable equilibrium state (a saddle) in region 2 as shown in Fig. 4-5.

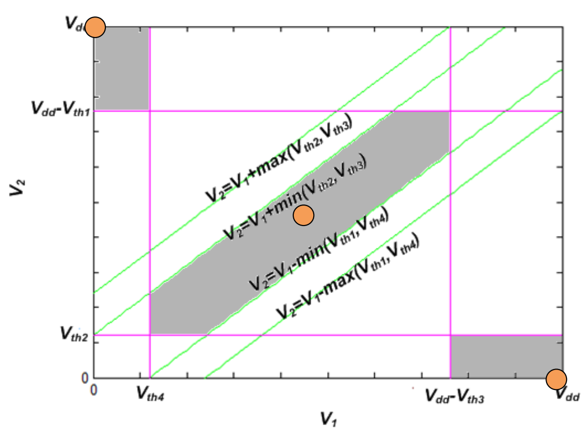


Fig. 4-5 Three Equilibrium Points Sit in Region 1 2 and 3.

4.4. THE SEPARATRIX EQUATION

4.4.1 The First Order Linear Approximation of Separatrix Equation

The Eigenvectors passing through the equilibrium point will be tangent to its corresponding manifold [42] [48-51] by the stable manifold theorem quoted in Chapter III section 3.4. Since the separatrix is the stable manifold of the saddle point, the stable eigenvector will be tangent to the separatrix.

Theorem 4.4.1:

The linearly approximated separatrix equation is:

$$V_2 = \frac{\sqrt{B}}{\sqrt{A}} V_1 + (V_{2saddle} - \frac{\sqrt{B}}{\sqrt{A}} V_{1saddle}) \tag{4. 22}$$

Proof:

From Jacobian matrix (4.19), the eigenvector at saddle express as:

$$\lambda_a = \sqrt{A \cdot B} \quad V_a = \begin{pmatrix} V_{a1} \\ V_{a2} \end{pmatrix} = \begin{bmatrix} \sqrt{A} \\ -\sqrt{B} \end{bmatrix} = \begin{bmatrix} \sqrt{2\frac{\sqrt{K_1} * \sqrt{K_2}}{C_1}(V_{dd} - Vth_1 - Vth_2)} \\ -\sqrt{2\frac{\sqrt{K_3} * \sqrt{K_4}}{C_2}(V_{dd} - Vth_3 - Vth_4)} \end{bmatrix} \tag{4. 23}$$

$$\lambda_b = -\sqrt{A \cdot B} \quad V_b = \begin{pmatrix} V_{b1} \\ V_{b2} \end{pmatrix} = \begin{bmatrix} \sqrt{A} \\ \sqrt{B} \end{bmatrix} = \begin{bmatrix} \sqrt{2\frac{\sqrt{K_1} * \sqrt{K_2}}{C_1}(V_{dd} - Vth_1 - Vth_2)} \\ \sqrt{2\frac{\sqrt{K_3} * \sqrt{K_4}}{C_2}(V_{dd} - Vth_3 - Vth_4)} \end{bmatrix} \tag{4. 24}$$

Note that λ_b is the stable eigenvalue and V_b is the stable eigenvector. From the stable manifold theorem we know the tangent vector of the separatrix is V_b. Accordingly, the slope of linear-approximated sepratrix is:

$$slope = \frac{V_{b2}}{V_{b1}} = \frac{\sqrt{B}}{\sqrt{A}} \tag{4. 25}$$

The first order approximation of the separatrix then has the following form:

$$V_2 = \frac{\sqrt{B}}{\sqrt{A}} V_1 + c \tag{4. 26}$$

Since the separatrix equation must pass through the saddle point, the constant c is to be found by evaluating the equation at saddle point as shown below, where $V_{1saddle}$ and $V_{2saddle}$ are the saddle node component describe previously.

$$c = \sqrt{A} \cdot V_{2saddle} - \sqrt{B} \cdot V_{1saddle} \tag{4. 27}$$

The separatrix can be approximated by the linear-separatrix equation. For example, the Fig. 4-6 shown below is the separatrix of 65nm technology parameter with variation of threshold $V_{th1}=V_{th1nominal}*(1+70\%)$, $V_{th2}=V_{th2nominal}*(1-70\%)$, $V_{th3}=V_{th3nominal}*(1-70\%)$, $V_{th4}=V_{th4nominal}*(1+70\%)$. The separatrix has first order property as shown in plot. By using the linear-separatrix equation on 65nm parameters, $\sqrt{A} = 3.26e5$ $\sqrt{B} = 3.26e5$, separatrix slope=1, and c= -8.2e4. The linear-separatrix equation is calculated to be $V_{2linear} = V_1 - 0.25182$. Compare to the separatrix on the plot obtained by using the algorithm described in chapter III, they are too close to tell a difference.

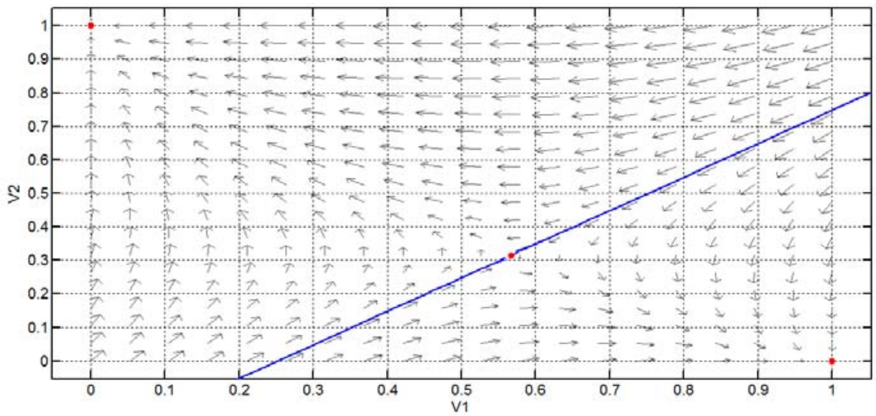


Fig. 4-6 The 1st Order Effect Separatrix.

4.4.2 The Exact Separatrix Equation

The linearly approximated equation is a good approximation if the separatrix's higher order effect is small. There are cases that the curvature of separatrix changes as the separatrix moves away from the saddle, and the separatrix will curve on the plot. Linearly apparoximation would introduce error as moving away from saddle. From Fig. 4-7, the threshold variation are in the manner of $V_{th1}=V_{th1nominal}*(1+70\%)$, $V_{th2}=V_{th2nominal}*(1+70\%)$, $V_{th3}=V_{th3nominal}*(1-70\%)$, $V_{th4}=V_{th4nominal}*(1-70\%)$, the black line is the calculated linear-separatrix equation, and the blue is the actual separatrix. As we can see, the separatrix appears nonlinear, it is curving as the separatrix moves away from the saddle.

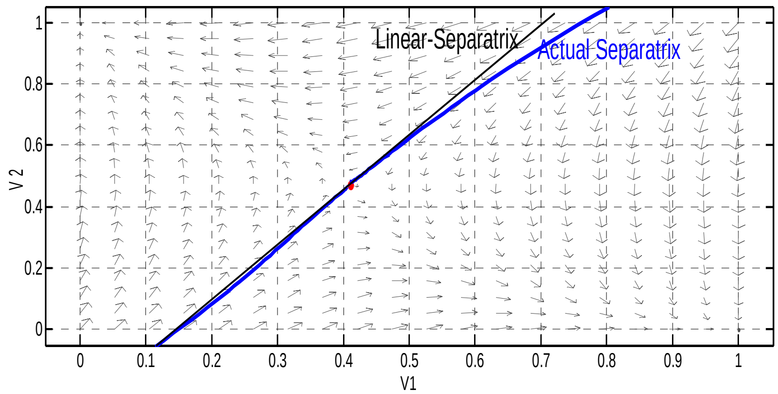


Fig. 4-7 Actual Separatrix Vs. Linear-Separatrix

The original SRAM model equation implies that the slope of actual-separatrix equation is:

$$\frac{dV_2}{dV_1} = \frac{\frac{K_3}{C_2}[S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] - \frac{K_4}{C_2}[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)]}{\frac{K_1}{C_1}[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] - \frac{K_2}{C_1}[S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)]} \quad (4.\ 28)$$

Intuitively, by cross-multiplying and integrating both sides of the equation just like the way did in theorem 4.4.1, the actual-separatrix equation will be derived. However, equation (4.28) involves integration of *S(·)* function, and the integration can not separate V_1 and V_2 variables. The integration is no longer valid.

Instead, consider the case that the separatrix remains in region 2. Then *S(·) in* the modeling equation can be removed as discussed earlier. From region 2 modeling equation (4.6), integration can be done since the functions extract out of *S()* and V_1 V_2 variables are separated.

Assumption for Theorem 4.4.2:

The separatrix remains entirely inside region 2.

Theorem 4.4.2:

The exact separatrix equation is:

$$\begin{aligned} & C_1K_3(V_{dd}-V_1-Vth_3)^3+C_1K_4(V_1-Vth_4)^3 \\ & -C_2K_1(V_{dd}-V_2-Vth_1)^3-C_2K_2(V_2-Vth_2)^3 \\ & =C_1(K_3K_4^{3/2}+K_4K_3^{3/2})\cdot(\frac{(V_{dd}-Vth_3-Vth_4)}{(\sqrt{K_3}+\sqrt{K_4})})^3 \\ & -C_2(K_1K_2^{3/2}+K_2K_1^{3/2})\cdot(\frac{(V_{dd}-Vth_1-Vth_2)}{\sqrt{K_1}+\sqrt{K_2}})^3 \end{aligned} \tag{4. 29}$$

Proof:

From the slope equation,

$$\frac{dV_2}{dV_1}=\frac{C_1[K_3(V_{dd}-V_1-Vth_3)^2-K_4(V_1-Vth_4)^2]}{C_2[K_1(V_{dd}-V_2-Vth_1)^2-K_2(V_2-Vth_2)^2]}$$

$$\begin{aligned} & C_2\int[K_1(V_{dd}-V_2-Vth_1)^2-K_2(V_2-Vth_2)^2]\cdot dV_2 \\ & =C_1\int[K_3(V_{dd}-V_1-Vth_3)^2-K_4(V_1-Vth_4)^2]\cdot dV_1 \end{aligned}$$

$$\begin{aligned} & C_2[-\frac{K_1(V_{dd}-V_2-Vth_1)^3}{3}-\frac{K_2(V_2-Vth_2)^3}{3}] \\ & =C_1[-\frac{K_3(V_{dd}-V_1-Vth_3)^3}{3}-\frac{K_4(V_1-Vth_4)^3}{3}]+C_{strip} \end{aligned}$$

After cancel out the one-third,

$$\begin{aligned} & -C_2K_1(V_{dd}-V_2-Vth_1)^3-C_2K_2(V_2-Vth_2)^3 \\ & =-C_1K_3(V_{dd}-V_1-Vth_3)^3-C_1K_4(V_1-Vth_4)^3+C_{strip} \end{aligned} \tag{4. 30}$$

Where constant C_{strip} is:

$$\begin{aligned}
C_{strip} &= C_1K_3(V_{dd}-V_{1saddle}-Vth_3)^3 + C_1K_4(V_{1saddle}-Vth_4)^3 \\
&\quad - C_2K_1(V_{dd}-V_{2saddle}-Vth_1)^3 - C_2K_2(V_{2saddle}-Vth_2)^3 \\
&= C_1K_3(V_{dd}-\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}}-Vth_3)^3 \\
&\quad + C_1K_4(\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}}-Vth_4)^3 \\
&\quad - C_2K_1(V_{dd}-\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}}-Vth_1)^3 \\
&\quad - C_2K_2(\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}}-Vth_2)^3 \\
&= C_1K_3(\frac{(\sqrt{K_3}+\sqrt{K_4})\cdot(V_{dd}-Vth_3)}{\sqrt{K_3}+\sqrt{K_4}}-\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}})^3 \\
&\quad + C_1K_4(\frac{\sqrt{K_3}*(V_{dd}-Vth_3)+\sqrt{K_4}*Vth_4}{\sqrt{K_3}+\sqrt{K_4}}-\frac{(\sqrt{K_3}+\sqrt{K_4})\cdot Vth_4}{\sqrt{K_3}+\sqrt{K_4}})^3 \\
&\quad - C_2K_1(\frac{(\sqrt{K_1}+\sqrt{K_2})\cdot(V_{dd}-Vth_1)}{\sqrt{K_1}+\sqrt{K_2}}-\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}})^3 \\
&\quad - C_2K_2(\frac{\sqrt{K_1}*(V_{dd}-Vth_1)+\sqrt{K_2}*Vth_2}{\sqrt{K_1}+\sqrt{K_2}}-\frac{(\sqrt{K_1}+\sqrt{K_2})\cdot Vth_2}{\sqrt{K_1}+\sqrt{K_2}})^3
\end{aligned}$$

$$\begin{aligned}
&= C_1K_3(\frac{\sqrt{K_4}\cdot(V_{dd}-Vth_3-Vth_4)}{\sqrt{K_3}+\sqrt{K_4}})^3 + C_1K_4(\frac{\sqrt{K_3}*(V_{dd}-Vth_3-Vth_4)}{\sqrt{K_3}+\sqrt{K_4}})^3 \\
&\quad - C_2K_1(\frac{(\sqrt{K_2})\cdot(V_{dd}-Vth_1-Vth_2)}{\sqrt{K_1}+\sqrt{K_2}})^3 - C_2K_2(\frac{\sqrt{K_1}*(V_{dd}-Vth_1-Vth_2)}{\sqrt{K_1}+\sqrt{K_2}})^3 \\
&= C_1K_3K_4^{3/2}(\frac{(V_{dd}-Vth_3-Vth_4)}{(\sqrt{K_3}+\sqrt{K_4})})^3 + C_1K_4K_3^{3/2}(\frac{(V_{dd}-Vth_3-Vth_4)}{\sqrt{K_3}+\sqrt{K_4}})^3 \\
&\quad - C_2K_1K_2^{3/2}(\frac{(V_{dd}-Vth_1-Vth_2)}{\sqrt{K_1}+\sqrt{K_2}})^3 - C_2K_2K_1^{3/2}(\frac{(V_{dd}-Vth_1-Vth_2)}{\sqrt{K_1}+\sqrt{K_2}})^3
\end{aligned}$$

$$C_{strip} = C_1(K_3 K_4^{3/2} + K_4 K_3^{3/2}) \cdot (\frac{(V_{dd} - Vth_3 - Vth_4)}{(\sqrt{K_3} + \sqrt{K_4})})^3$$
$$- C_2(K_1 K_2^{3/2} + K_2 K_1^{3/2}) \cdot (\frac{(V_{dd} - Vth_1 - Vth_2)}{\sqrt{K_1} + \sqrt{K_2}})^3 \quad (4.31)$$

Corollary: For the special case, all K, C and thresolds are equal separately, the separatrix becomes $V_1 = V_2$, which matches the linearly approximated separatrix from theorem 4.4.1

Proof:

$C_{strip} = 0$ for $K_1 = K_2 = K_3 = K_4$, $C_1 = C_2$ and $V_{th1} = V_{th2} = V_{th3} = V_{th4} = V_{th}$

Then equation (4.29) beccomes:

$$C_1 K_3 (V_{dd} - V_1 - Vth_3)^3 + C_1 K_4 (V_1 - Vth_4)^3$$
$$- C_2 K_1 (V_{dd} - V_2 - Vth_1)^3 - C_2 K_2 (V_2 - Vth_2)^3 = 0$$

$$(V_{dd} - V_1 - Vth)^3 - (V_{dd} - V_2 - Vth)^3 + (V_1 - Vth)^3 - (V_2 - Vth)^3 = 0 \quad (4.\ 32)$$

By setting

$$x_1 = V_{dd} - V_1 - Vth$$
$$x_2 = V_{dd} - V_2 - Vth$$
$$y_1 = V_1 - Vth$$
$$y_2 = V_2 - Vth$$

(4.32) can be written into

$$(x_1^3 - x_2^3) + (y_1^3 - y_2^3) = 0 \quad (4.\ 33)$$

$$(x_1 - x_2)(x_1^2 + x_1 x_2 + x_2^2) + (y_1 - y_2)(y_1^2 + y_1 y_2 + y_2^2) = 0$$
$$(x_1 - x_2)(x_1^2 + x_1 x_2 + x_2^2) - (y_2 - y_1)(y_1^2 + y_1 y_2 + y_2^2) = 0$$
$$(V_2 - V_1)(x_1^2 + x_1 x_2 + x_2^2) - (V_2 - V_1)(y_1^2 + y_1 y_2 + y_2^2) = 0$$
$$(V_2 - V_1)[(x_1^2 + x_1 x_2 + x_2^2) - (y_1^2 + y_1 y_2 + y_2^2)] = 0$$

Suppose $V_2 - V_1 \neq 0$,

$$(x_1^{\,2} + x_1x_2 + x_2^{\,2}) - (y_1^{\,2} + y_1y_2 + y_2^{\,2}) = 0$$
$$x_1^{\,2} - y_1^{\,2} + x_2^{\,2} - y_2^{\,2} + x_1x_2 - y_1y_2 = 0$$

$$2(V_2^{\,2} - V_1^{\,2}) - 2V_{dd}(V_2 - V_1) + (V_2^{\,2} - V_1^{\,2}) - 2V_{dd}(V_2 - V_1) = 0$$
$$3(V_2^{\,2} - V_1^{\,2}) - 3V_{dd}(V_2 - V_1) = 0$$
$$(V_2 - V_1)(V_2 + V_1) - V_{dd}(V_2 - V_1) = 0$$
$$(V_2 + V_1) = V_{dd}$$

Then it violates our assumption that it lies in the region 2, thus impossible.

Accordingly, $V_2 - V_1 = 0$, is the only candidate left, which also satifies our assumption.

$$V_2 = V_1 \qquad (4.\ 34)$$

(4.34) is the linear-approximated separatrix equation.

4.5. MOVEMENT OF EQUILIBRIA WHEN PERTURBED BY NOISE

4.5.1 Case I: Modeling Noise Input Attached At the Right Inverter

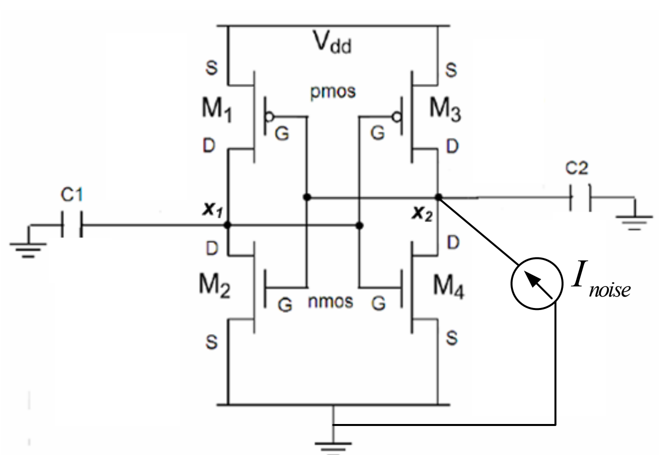


Fig. 4-8 The Layout of Case I.

Case I models noise injection as a constant current source attached to V_2 node (shown in Fig. 4-8). Let the state of SRAM device is denoted by (V_1 ,V_2). Assume the state is *(Vdd,0)*. Theoretically, an injecting positive noise current will raise the voltage at V_2 node. Because of the coupling effect, the voltage rising at V_2 node would bring down the voltage at V_1 node, and state flipping would occur eventually. If the initial SRAM condition is (*0,Vdd)*, injecting current to V_2 would not induce state flipping, it should extract current from V_2 node in order to have state flipping. Focus on the situation that would result state flipping, the modeling equation is presented as below. When the input noise current is zero, the SRAM equilibrium points sit in three regions shown in Fig. 4-5.

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \qquad - \frac{1}{C_1} K_2[S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] \\ \dot{V}_2 = \frac{1}{C_2} K_3[S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] \\ \qquad - \frac{1}{C_2} K_4[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \frac{I_{noise}}{C_2} \end{cases} \tag{4. 35}$$

Ideally, by introducing I_{noise} , the three equilibria will change their relative locations. The following theorems focus on the three equilibrium points. We specify I_{noies} in theorems so that theorem would work in a particular region.

4.5.1.1 Region 1

Assumption for theorem 4.5.1:

I_{noise} satisfy the condition:

$$-K_3(V_{dd} - V_{th3})^2 \le I_{noise} \tag{4. 36}$$

V_2 satisfy the condition:

$$V_2 \geq V_{dd} - V_{th1} \tag{4. 37}$$

Theorem 4.5.1:

Then the equilibrium state remains in region 1 and (V_1^{R1}, V_2^{R1}) *is a function of I_{noise} (as a special case,* V_1^{R1} *=0* V_2^{R1} *=V_{dd} when I_{noise}=0) and it remains a stable node.*

$$\begin{cases} V_1^{R1} = 0 \\ V_2^{R1} = \sqrt{(V_{dd} - V_{th3})^2 + \dfrac{I_{noise}}{K_3}} + V_{th3} \end{cases} \tag{4. 38}$$

Proof:

In region 1, four *S()* terms can be eliminated from the modeling equation (4.1) (4.2), they are $S^2(V_1 - V_2 - Vth_1)$, $S^2(V_1 - V_2 - Vth_4)$, $S^2(V_{dd} - V_2 - Vth_1)$ and $S^2(V_1 - Vth_4)$. The modeling equation is simplified to (4.38).

$$\begin{aligned} \dot{V}_1 &= -\frac{1}{C_1} K_2[(V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2] \\ \dot{V}_2 &= \frac{1}{C_2} K_3[(V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2] + \frac{I_{noise}}{C_2} \end{aligned} \tag{4. 39}$$

Solve for equilibrium point by setting $\dot{V}_1$=0 $\dot{V}_2$=0, and solve for V_1 V_2. The solution comes out nicely showing below in (4.39).

$$\begin{aligned} &-\frac{1}{C_1} K_2[(V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2] = 0 \\ &\frac{1}{C_2} K_3[(V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2] + \frac{I_{noise}}{C_2} = 0 \end{aligned}$$

$$\begin{aligned} &[(V_2 - V_{th2})^2 - (V_2 - V_1 - V_{th2})^2] = 0 \\ &K_3[(V_{dd} - V_1 - V_{th3})^2 - (V_2 - V_1 - V_{th3})^2] + I_{noise} = 0 \end{aligned}$$

$$(V_2 - V_{th2}) = (V_2 - V_1 - V_{th2}) \Rightarrow V_1 = 0$$

$$(V_{dd} - V_{th3})^2 - (V_2 - V_{th3})^2 + \frac{I_{noise}}{K_3} = 0$$

$$(V_{dd} - V_{th3})^2 - (V_2{}^2 - 2V_{th3}V_2 + V_{th3}{}^2) + \frac{I_{noise}}{K_3} = 0$$

$$V_2{}^2 - 2V_{th3}V_2 + V_{th3}{}^2 - (V_{dd} - V_{th3})^2 - \frac{I_{noise}}{K_3} = 0$$

Apply quadratic equation to solve for V_2:

$$V_2 = \frac{2V_{th3} \pm \sqrt{4V_{th3}{}^2 - 4(V_{th3}{}^2 - (V_{dd} - V_{th3})^2 - \frac{I_{noise}}{K_3})}}{2}$$

$$V_2 = V_{th3} \pm \sqrt{(V_{dd} - V_{th3})^2 + \frac{I_{noise}}{K_3}} \qquad (4.40)$$

Note that the equilibrium stay at (0, V_{dd}) in region 1 when I_{noise}=0. Therefore, when I_{noise}=0 the correct sign should give V_2=V_{dd}. Accordingly, equation (4.40) tells the sign should be plus.

$$V_2 = V_{th3} \pm (V_{dd} - V_{th3}) \qquad (4.41)$$

$$V_2 = \sqrt{(V_{dd} - V_{th3})^2 + \frac{I_{noise}}{K_3}} + V_{th3}$$

From the result, V_1 component is zero, which is independent of I_{noise} in this region; and I_{noise} term only appears in V_2 component. If I_{noise} increase to infinity, V_2 will also increase to infinity. Thus, equilibrium point in region 1 looks going upward as I_{noise} increase. If the I_{noise} is negative, the V_2 would decrease, and the equilibrium point is moving downward on the phase portrait. However, the assumption on I_{noise} is that the I_{noise} should satisfy the equation (4.35). Otherwise, the equilbirum point may move out of region 1; and the region 1 equation (4.37) can not be used, since the equilibrium point is no longer in region 1.

Now we prove that the region 1 equilibrium remains a stable node as long as the input source I_{noise} can be modeled as a constant that is independent of V_1 V_2.

Proof:

From (4.33), the Jacobian matrix,

$$\begin{cases} \dot{V}_1 = -\frac{1}{C_1} K_2[(V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2] = f_1(V_1, V_2) \\ \dot{V}_2 = \frac{1}{C_2} K_3[(V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2] + \frac{I_{noise}}{C_2} = g_1(V_1, V_2) \end{cases}$$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial V_1} & \frac{\partial f_1}{\partial V_2} \\ \frac{\partial g_1}{\partial V_1} & \frac{\partial g_1}{\partial V_2} \end{bmatrix}\Bigg|_{\substack{V_1 = V_1^{R1} \\ V_2 = V_2^{R1}}} = \begin{bmatrix} -\frac{2K_2}{C_1}(V_2 - V_1 - V_{th2}) & -\frac{2K_2}{C_1} V_1 \\ \frac{2K_3}{C_2}(V_2 - V_{dd}) & -\frac{2K_3}{C_2}(V_2 - V_1 - V_{th3}) \end{bmatrix}\Bigg|_{\substack{V_1 = V_1^{R1} \\ V_2 = V_2^{R1}}}$$

$$= \begin{bmatrix} -\frac{2K_2}{C_1}[\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}} + V_{th3} - V_{th2}] & 0 \\ -\frac{2K_3}{C_2}(\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}} + V_{th3} - V_{dd}) & -\frac{2K_3}{C_2}\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}} \end{bmatrix}$$

$$\det(\lambda_{R1} I - J) = 0$$

$$[\lambda_{R1} + \frac{2K_2}{C_1}[\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}} + V_{th3} - V_{th2}]] \cdot [\lambda_{R1} + \frac{2K_3}{C_2}\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}}] = 0$$

$$\lambda_{R1} = -\frac{2K_2}{C_1}[V_2^{R1} - V_{th2}], -\frac{2K_3}{C_2}\sqrt{(V_{dd} - V_{th3})^2 + \frac{I_n}{K_3}}$$

The square root expects to be positive by the assumption (4.36). Thus, the second eigenvalue is clearly negative. The first eigenvalue is negative if $V_2^{R1} - V_{th2}$ is positive. The V_2^{R1} is the V_2 component in region 1. From Fig. 4-1, the V_2 component in region 1 is greater than V_{th2}, so the first eigenvalue is also negative. The equilibrium is stable since both eigenvalues are negative.

4.5.1.2 Region 2

Assumption for theorem 4.5.2:

I_{noise} satisfy the condition:

$$I_{noise} \geq K_4 V_{th4}{}^2 - K_3 (V_{dd} - V_{th3})^2 - \frac{(K_4 \cdot V_{th4} - K_3 (V_{dd} - V_{th3}))^2}{(K_4 - K_3)} \tag{4. 42}$$

V_1 satisfy the condition:

$$V_{2saddle} - \min(V_{th2}, V_{th3}) \leq V_1 \leq V_{2saddle} + \min(V_{th1}, V_{th4}) \tag{4. 43}$$

Theorem 4.5.2:

Then the equilibrium state (V_1^{R2} =$V_{1saddle}$ V_2^{R2} =$V_{2saddle}$ when I_{noise}=0) remians in region 2 and is a function of I_{noise}. It remains a saddle.

$$\begin{cases} V_1^{R2} = \dfrac{B + \sqrt{B^2 + (K_4 - K_3) \cdot (I_{noise} + K_3 (V_{dd} - V_{th3})^2 - K_4 V_{th4}{}^2)}}{(K_4 - K_3)} \\ \quad where \quad B = K_4 \cdot V_{th4} - K_3 (V_{dd} - V_{th3}) \\ V_2^{R2} = \dfrac{\sqrt{K_1} * (V_{dd} - Vth_1) + \sqrt{K_2} * Vth_2}{\sqrt{K_1} + \sqrt{K_2}} = V_{2saddle} \end{cases} \tag{4. 44}$$

Proof:

The derivation is similar to theorem 4.5.1.

$$\begin{cases} \dot{V}_1 = \dfrac{K_1}{C_1}(V_{dd} - V_2 - Vth_1)^2 - \dfrac{K_2}{C_1}(V_2 - Vth_2)^2 = 0 \\ \dot{V}_2 = \dfrac{K_3}{C_2}(V_{dd} - V_1 - Vth_3)^2 - \dfrac{K_4}{C_2}(V_1 - Vth_4)^2 + \dfrac{I_{noise}}{C_2} = 0 \end{cases}$$

$$K_1(V_{dd}-V_2-Vth_1)^2 = K_2(V_2-Vth_2)^2$$
$$\sqrt{K_1}(V_{dd}-V_2-Vth_1) = \sqrt{K_2}(V_2-Vth_2)$$
$$\sqrt{K_1}(V_{dd}-Vth_1)-\sqrt{K_2}Vth_2 = (\sqrt{K_1}+\sqrt{K_2})V_2$$
$$V_2 = \frac{\sqrt{K_1}(V_{dd}-Vth_1)-\sqrt{K_2}Vth_2}{\sqrt{K_1}+\sqrt{K_2}} = V_{2saddle}$$

$$K_3(V_{dd}-V_1-Vth_3)^2 - K_4(V_1-Vth_4)^2 + I_{noise} = 0$$
$$K_3[(V_{dd}-Vth_3)^2 - 2(V_{dd}-Vth_3)V_1 + V_1{}^2] - K_4(V_1{}^2 - 2Vth_4V_1 + Vth_4{}^2) + I_{noise} = 0$$
$$K_3(V_{dd}-Vth_3)^2 - 2K_3(V_{dd}-Vth_3)V_1 + K_3V_1{}^2 - K_4V_1{}^2 + 2K_4Vth_4V_1 - K_4Vth_4{}^2 + I_{noise} = 0$$
$$(K_3-K_4)V_1{}^2 + (2K_4Vth_4 - 2K_3(V_{dd}-Vth_3))V_1 + K_3(V_{dd}-Vth_3)^2 - K_4Vth_4{}^2 + I_{noise} = 0$$
$$\rightarrow A\cdot V_1{}^2 + B\cdot V_1 + C = 0$$

$$V_1 = \frac{-B \pm \sqrt{B^2-4AC}}{2A}$$
$$= \frac{-B \pm \sqrt{B^2 - 4(K_3-K_4)(K_3(V_{dd}-Vth_3)^2 - K_4Vth_4{}^2 + I_{noise})}}{2(K_3-K_4)}$$

Since the driving strength for NMOS is stronger than PMOS in SRAM, K_3-K_4 is actually negative. The expression becomes:

$$V_1 = \frac{B \mp \sqrt{B^2 + 4(K_4-K_3)(K_3(V_{dd}-Vth_3)^2 - K_4Vth_4{}^2 + I_{noise})}}{2(K_4-K_3)}$$

When set I_{noise}=0, the V_1 suppose to be $V_{1saddle}$. Due to the equation complexity, the sign determination is done by plugging in 65nm system parameter values and match V_1=$V_{1saddle}$. Then the final set of equation is shown in (4.44).

$$V_1 = \frac{B + \sqrt{B^2 + (K_4-K_3)\cdot(I_{noise} + K_3(V_{dd}-V_{th3})^2 - K_4V_{th4}{}^2)}}{(K_4-K_3)} \qquad (4.\ 45)$$
$$where \qquad B = K_4\cdot V_{th4} - K_3(V_{dd}-V_{th3})$$

From the formula, there is a condition for I_{noise} for the square root to be legal.

$$(K_4 \cdot V_{th4} - K_3(V_{dd} - V_{th3}))^2 + (K_4 - K_3)\cdot(I_{noise} + K_3(V_{dd} - V_{th3})^2 - K_4 V_{th4}{}^2) \geq 0$$

$$\frac{(K_4 \cdot V_{th4} - K_3(V_{dd} - V_{th3}))^2}{(K_4 - K_3)} + (I_{noise} + K_3(V_{dd} - V_{th3})^2 - K_4 V_{th4}{}^2) \geq 0$$

$$-I_{noise} \leq \frac{(K_4 \cdot V_{th4} - K_3(V_{dd} - V_{th3}))^2}{(K_4 - K_3)} + K_3(V_{dd} - V_{th3})^2 - K_4 V_{th4}{}^2$$

$$I_{noise} \geq K_4 V_{th4}{}^2 - K_3(V_{dd} - V_{th3})^2 - \frac{(K_4 \cdot V_{th4} - K_3(V_{dd} - V_{th3}))^2}{(K_4 - K_3)}$$

Because $V_{th4}{}^2$ is smaller than $(V_{dd} - V_{th3})^2$, the expression comes out a negative number.

Now we prove that the region 2 equilibrium remains a saddle as long as the input source I_{noise} can be modeled as a constant that is independent of V_1 V_2

Proof:

$$\begin{cases} \dot{V}_1 = \dfrac{K_1}{C_1}(V_{dd} - V_2 - Vth_1)^2 - \dfrac{K_2}{C_1}(V_2 - Vth_2)^2 = f_s = 0 \\ \dot{V}_2 = \dfrac{K_3}{C_2}(V_{dd} - V_1 - Vth_3)^2 - \dfrac{K_4}{C_2}(V_1 - Vth_4)^2 + \dfrac{I_{noise}}{C_2} = g_s = 0 \end{cases}$$

$$J = \left.\begin{bmatrix} \dfrac{\partial f_s}{\partial V_1} & \dfrac{\partial f_s}{\partial V_2} \\ \dfrac{\partial g_s}{\partial V_1} & \dfrac{\partial g_s}{\partial V_2} \end{bmatrix}\right|_{\substack{V1=V_1^{R2} \\ V2=V_2^{R2}}}$$

$$= \left.\begin{bmatrix} 0 & -2\dfrac{K_1}{C_1}(V_{dd} - V_2 - V_{th1}) - 2\dfrac{K_2}{C_1}(V_2 - V_{th2}) \\ -2\dfrac{K_3}{C_2}(V_{dd} - V_1 - V_{th3}) - 2\dfrac{K_4}{C_2}(V_1 - V_{th4}) & 0 \end{bmatrix}\right|_{\substack{V1=V_1^{R2} \\ V2=V_2^{R2}}}$$

$$\det(\lambda_{R2} I - J) = 0$$

$$\lambda_{R2}{}^2 = 4\left(\frac{K_1}{C_1}(V_{dd} - V_2^{R2} - V_{th1}) + \frac{K_2}{C_1}(V_2^{R2} - V_{th2})\right)\cdot\left(\frac{K_3}{C_2}(V_{dd} - V_1^{R2} - V_{th3}) - \frac{K_4}{C_2}(V_1^{R2} - V_{th4})\right)$$

$$\lambda_{R2} = \pm 2\sqrt{(\frac{K_1}{C_1}(V_{dd} - V_2^{R2} - V_{th1}) + \frac{K_2}{C_1}(V_2^{R2} - V_{th2}))\cdot(\frac{K_3}{C_2}(V_{dd} - V_1^{R2} - V_{th3}) - \frac{K_4}{C_2}(V_1^{R2} - V_{th4}))}$$

Since (V_1^{R2}, V_2^{R2}) is remained in region 2 under our asumptions, the value inside the square root of λ_{R2} is positive, so λ_{R2} would have one positive eigenvalue and one negative eigenvalue. The equilibrium is a saddle due to one positive and one negative eigenvalues.

4.5.1.3 Region 3

Assumption for theorem 4.5.3:

I_{noise} satisfy the condition:

$$-\infty \le I_{noise} \le K_4(V_{dd} - V_{th4})^2 \tag{4. 46}$$

V_2 satisfy the condition:

$$V_2 \le V_{th2} \tag{4. 47}$$

Theorem 4.5.3:

Then the equilibrium state ($V_1^{R3}=V_{dd}$ $V_2^{R3}=0$ when $I_{noise}=0$) remians in region 3 and is a function of I_{noise}; and it remains a stable node.

$$\begin{cases} V_1^{R3} = V_{dd} \\ V_2^{R3} = V_{dd} - V_{th4} - \sqrt{(V_{dd} - V_{th4})^2 - \dfrac{I_{noise}}{K_4}} \end{cases} \tag{4. 48}$$

Proof:

$$\begin{cases} \dot{V}_1 = \dfrac{1}{C_1}K_1[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \dot{V}_2 = -\dfrac{1}{C_2}K_4[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \dfrac{I_{noise}}{C_2} \end{cases}$$

$$\begin{cases} \frac{1}{C_1} K_1[(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] = 0 \\ -\frac{1}{C_2} K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] + \frac{I_{noise}}{C_2} = 0 \end{cases}$$

$$\begin{cases} (V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2 = 0 \\ (V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2 - \frac{I_{noise}}{K_4} = 0 \end{cases}$$

$$(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2 = 0$$
$$(V_{dd} - V_2 - Vth_1)^2 = (V_1 - V_2 - Vth_1)^2$$
$$V_{dd} - V_2 - Vth_1 = V_1 - V_2 - Vth_1 \Rightarrow V_{dd} = V_1$$

$$(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2 - \frac{I_{noise}}{K_4} = 0$$

$$(V_{dd} - Vth_4)^2 - (V_{dd} - V_2 - Vth_4)^2 - \frac{I_{noise}}{K_4} = 0$$

$$(V_{dd} - Vth_4)^2 - (V_{dd} - Vth_4)^2 + 2(V_{dd} - Vth_4)V_2 - V_2{}^2 - \frac{I_{noise}}{K_4} = 0$$

$$V_2{}^2 - 2(V_{dd} - Vth_4)V_2 + \frac{I_{noise}}{K_4} = 0 \rightarrow AV_2{}^2 + BV_2 + C = 0$$

$$\begin{aligned} V_2 &= \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \\ &= \frac{2(V_{dd} - Vth_4) \pm \sqrt{4(V_{dd} - Vth_4)^2 - 4\frac{I_{noise}}{K_4}}}{2} \\ &= (V_{dd} - Vth_4) \pm \sqrt{(V_{dd} - Vth_4)^2 - \frac{I_{noise}}{K_4}} \end{aligned}$$

V_2 suppose to be 0 in region 3 when I_{noise}=0.

$$\begin{aligned} V_2 &= (V_{dd} - Vth_4) \pm \sqrt{(V_{dd} - Vth_4)^2} \\ &= (V_{dd} - Vth_4) \pm (V_{dd} - Vth_4) \end{aligned}$$

The sign is minus for V_2=0 for the above expression. Thus the final expression for V_2 is:

$$V_2 = V_{dd} - V_{th4} - \sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}}$$

Now we prove that the region 3 equilibrium remains a stable node as long as the input source I_{noise} can be modeled as a constant that is independent of V_1 V_2

Proof:

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] = f_2(V_1, V_2) \\ \dot{V}_2 = -\frac{1}{C_2} K_4 [(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] + \frac{I_{noise}}{C_2} = g_2(V_1, V_2) \end{cases}$$

$$J = \left. \begin{bmatrix} \frac{\partial f_2}{\partial V_1} & \frac{\partial f_2}{\partial V_2} \\ \frac{\partial g_2}{\partial V_1} & \frac{\partial g_2}{\partial V_2} \end{bmatrix} \right|_{\substack{V_1 = V_1^{R3} \\ V_2 = V_2^{R3}}} = \left. \begin{bmatrix} -\frac{2K_1}{C_1}(V_1 - V_2 - V_{th1}) & -\frac{2K_1}{C_1}(V_1 - V_{dd}) \\ -\frac{2K_4}{C_2}(V_2) & -\frac{2K_4}{C_2}(V_1 - V_2 - V_{th4}) \end{bmatrix} \right|_{\substack{V_1 = V_1^{R3} \\ V_2 = V_2^{R3}}}$$

$$= \begin{bmatrix} -\frac{2K_1}{C_1}\left(\sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}} + V_{th4} - V_{th1}\right) & 0 \\ -\frac{2K_4}{C_2}\left(V_{dd} - V_{th4} - \sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}}\right) & -\frac{2K_4}{C_2}\sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}} \end{bmatrix}$$

$$\det(\lambda_{R3} I - J) = 0$$

$$[\lambda_{R3} + \frac{2K_1}{C_1}(\sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}} + V_{th4} - V_{th1})] \cdot [\lambda_{R3} + \frac{2K_4}{C_2}\sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}}] = 0$$

$$\lambda_{R3} = -\frac{2K_1}{C_1}(V_{dd} - V_2^{R3} - V_{th1}), -\frac{2K_4}{C_2}\sqrt{(V_{dd} - V_{th4})^2 - \frac{I_{noise}}{K_4}}$$

The second eigenvalue is negative due to the condition (4.46). For the first eigenvalue, it is negative if $V_{dd} - V_2^{R3} - V_{th1} > 0$. That implies $V_2^{R3} < V_{dd} - V_{th1}$, which means that the V_2 component in region 3 is less than V_{dd}-V_{th1}, and it is true by checking Fig. 4-1. Therefore, the equilibrium is a stable node since both eigenvalues are negative.

Focus on equation (4.48), when I_{noise} increase/decrease slightly, V_2 would increase/decrease slightly. If I_{noise} is too large, the square root become imaginary, so I_{noise} should satisfy (4.45). For the square root to stay positively, I_{noise} must be equal to or less than $K_4(V_{dd}-V_{th4})^2$. The square root comes out zero if I_{noise} is equal to that value, so the maximun V_2 from the expression would be $V_{dd}-V_{th4}$. Normally, it should be bigger than V_{th2} (the vertical boundary of region 3). In modern technology like 65nm, the threshold voltages are low, and thus normally $V_{dd}-V_{th4}>V_{th2}$. Therefore, before the square root goes imaginary, the equilibrium point already run outside of region 3 and it should be modeling by another set of equations.

Fig. 4-9 shows the general case on how equilibrium point move in different regions as the noise changes. There are blue arrows pointing certain direction. The direction of arrows are the graphical representation of the derived equilibrium point equations. For instance, the region 2 equilibrium point equation only V_1 is involved I_{noise}. When put a small increment to I_{noise} , V_1 component is increasing. So Fig. 4-10(LEFT) region 2 equilibrium point attaches an arrow going to the right which represent the increasing of V_1. For large increment on I_{noise}, the equilibrium point might leave its original region. So far, the equilibrium point equations for region 1 2 and 3 are derived, but the equilibrium points for other regions are still unresolved. However, Fig. 4-10 demonstrates a movement of equilibrium points in 65nm SRAM. From Fig. 4-10 (LEFT), as sweeping I_{noise} from zero to infinity, the saddle node starts from region 2, then moves into region 8B, region 8A and finally reach region 9A. The stable equilibrium state in region 3 moves from region 3 to region 9A and meet with the saddle point in region 9A. This phenominum is called saddle-node bifurcation. The amount of positive I_{noise} that causes the bifurcation in region 9A is the positive $I_{critical}$. Similarly, Fig. 4-10 (RIGHT) is the plot of 65nm SRAM equilibria as I_{noise} sweep from zero to negative infinity. The bifurcation occurs in region 4A. The amount of negative I_{noise} that causes the bifurcation in region 4A is called the negative $I_{critical}$.

In summary, for postive current pumping into the V_2 node, the equilibria in region 2 and region 3 result in a bifurcation. In other words, those two equilibria will move and collide. Once they collide, they disappear and bifurcation results. This location where they collide is refered as “*bifurcation point*”.

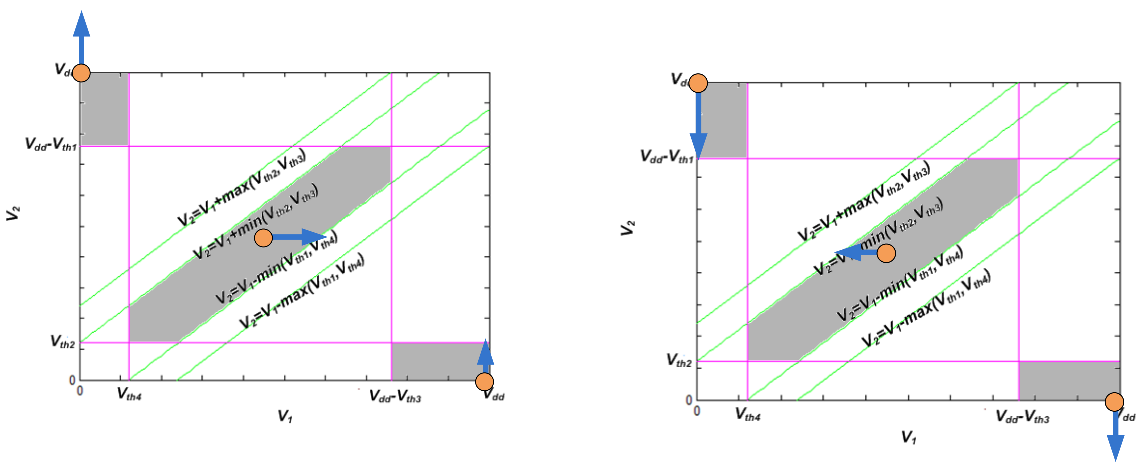


Fig. 4-9 (Left) Positive Current Injection in Case I. (Right) Negative Current Extraction in Case I.

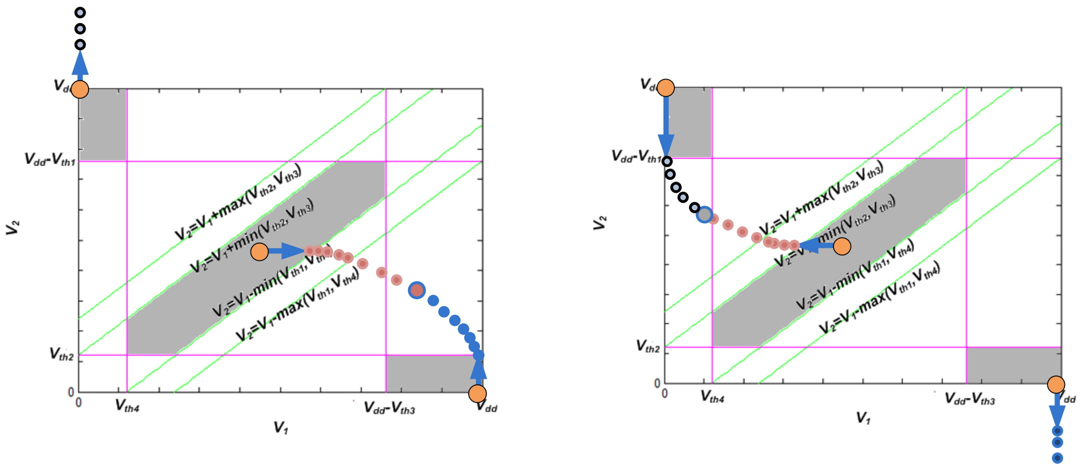


Fig. 4-10 (LEFT) Simulated Equilibrium Point Movement When Is Running Positive I_{noise} From Zero To Positive $I_{critical}$; (RIGHT) Simulated Equilibrium Point Movement When Running I_{noise} From Zero To Negative $I_{critical}$.

4.5.2 CASE II: Modeling Noise Input at the Left Inverter

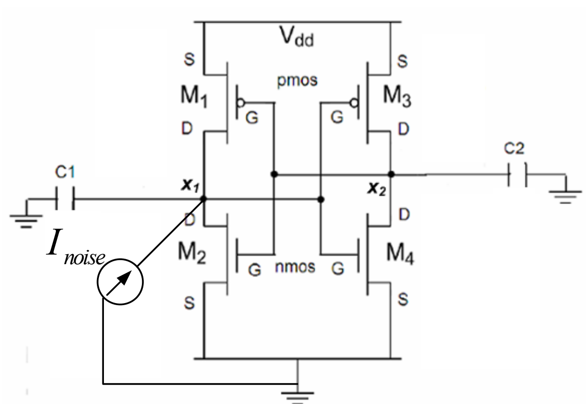


Fig. 4-11 The Layout of Case II.

This case assumes a current source modeling as noise attached to V_1 node (shown in Fig. 4-11). If the SRAM initially has state (0, V_{dd}), injecting positive current will raise the voltage at V_1 node, and state flipping would occur eventially. If the initial SRAM state is (V_{dd},0), extracting current from V_1 would result state flipping. This section will look at the change of equilibria position according to the change of I_{noise} in region 1 2 and 3 just like in Case I. Because the current source attachs to a different node, the modeling equation is slightly different than Case I.

$$\begin{cases} \dot{V}_1 = \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \qquad - \frac{1}{C_1} K_2 [S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] + \frac{I_{noise}}{C_1} \\ \dot{V}_2 = \frac{1}{C_2} K_3 [S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] \\ \qquad - \frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] \end{cases} \tag{4. 49}$$

Equation (4.51) (4.55) (4.59) below are the derived equilibria formulas in region 1 2 and 3 for Case II.

4.5.2.1 Region 1

Assumption to theorem 4.5.4:

The I_{noise} satisfy the condition:

$$-\infty < I_{noise} \le K_2 (V_{dd} - V_{th2})^2 \tag{4. 50}$$

V_1 satisfy the condition:

$$V_1 \le V_{th4} \tag{4. 51}$$

Theorem 4.5.4:

Then the following equations model the location of region 1 equilibrium state (V_1=0 V_2= V_{dd} when I_{noise}=0) in terms of I_{noise} and it remains a stable node.

$$\begin{cases} V_1 = V_{dd} - V_{th2} - \sqrt{(V_{dd} - V_{th2})^2 - \dfrac{I_{noise}}{K_2}} \\ V_2 = V_{dd} \end{cases} \quad (4.52)$$

Proof:

$$\begin{cases} \dot{V}_1 = -\dfrac{1}{C_1} K_2 [S^2(V_2 - Vth_2) - S^2(V_2 - V_1 - Vth_2)] + \dfrac{I_{noise}}{C_1} \\ \dot{V}_2 = \dfrac{1}{C_2} K_3 [S^2(V_{dd} - V_1 - Vth_3) - S^2(V_2 - V_1 - Vth_3)] \end{cases}$$

$$\begin{cases} -\dfrac{1}{C_1} K_2 [(V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2] + \dfrac{I_{noise}}{C_1} = 0 \\ \dfrac{1}{C_2} K_3 [(V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2] = 0 \end{cases}$$

$$\begin{cases} (V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2 - \dfrac{I_{noise}}{K_2} = 0 \\ (V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2 = 0 \end{cases}$$

$$(V_{dd} - V_1 - Vth_3)^2 - (V_2 - V_1 - Vth_3)^2 = 0$$
$$V_{dd} - V_1 - Vth_3 = V_2 - V_1 - Vth_3$$
$$\Rightarrow V_2 = V_{dd}$$

$$(V_2 - Vth_2)^2 - (V_2 - V_1 - Vth_2)^2 - \frac{I_{noise}}{K_2} = 0$$

$$(V_{dd} - Vth_2)^2 - (V_{dd} - V_1 - Vth_2)^2 - \frac{I_{noise}}{K_2} = 0$$

$$(V_{dd} - Vth_2)^2 - ((V_{dd} - Vth_2)^2 - 2(V_{dd} - Vth_2)V_1 + V_1^2) - \frac{I_{noise}}{K_2} = 0$$

$$+ 2(V_{dd} - Vth_2)V_1 - V_1^2 - \frac{I_{noise}}{K_2} = 0$$

$$V_1^2 - 2(V_{dd} - Vth_2)V_1 + \frac{I_{noise}}{K_2} = 0 \rightarrow AV_1^2 + BV_1 + C = 0$$

$$V_1 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$= \frac{2(V_{dd} - Vth_2) \pm \sqrt{4(V_{dd} - Vth_2)^2 - 4(\frac{I_{noise}}{K_2})}}{2}$$

$$= (V_{dd} - Vth_2) \pm \sqrt{(V_{dd} - Vth_2)^2 - \frac{I_{noise}}{K_2}}$$

V_1=0 when I_{noise}=0, so V_1 is:

$$V_1 = (V_{dd} - Vth_2) - \sqrt{(V_{dd} - Vth_2)^2 - \frac{I_{noise}}{K_2}} \tag{4. 53}$$

The I_{noise} must satisfy (4.49) to keep the square root to be positive.

4.5.2.2 Region 2

Asumption to theorem 4.5.5:

I_{noise} satisfy the condition:

$$I_{noise} \geq K_2 Vth_2{}^2 - K_1(V_{dd} - Vth_1)^2 - \frac{(K_2 Vth_2 - K_1(V_{dd} - Vth_1))^2}{(K_2 - K_1)} \tag{4. 54}$$

V2 satisfy the condition:

$$V_{1saddle} - \min(V_{th1}, V_{th4}) \leq V_2 \leq V_{1saddle} + \min(V_{th2}, V_{th3}) \tag{4. 55}$$

(These two conditions will keept the equilibirum in region2)

Theorem 4.5.5:

Then the following equation describes the location of perturbed region 2 equilibrium state (V_1=V_{1sadle} V_2= $V_{2saddle}$ when I_{noise}=0) in terms of I_{noise} and it remains a saddle.

$$\begin{cases} V_1 = \dfrac{\sqrt{K_3} * (V_{dd} - Vth_3) + \sqrt{K_4} * Vth_4}{\sqrt{K_3} + \sqrt{K_4}} = V_{1saddle} \\ V_2 = \dfrac{B + \sqrt{B^2 + (K_2 - K_1)(K_1(V_{dd} - Vth_1)^2 - K_2 Vth_2{}^2 + I_{noise})}}{(K_2 - K_1)} \\ \quad where \;\; B = K_2 Vth_2 - K_1(V_{dd} - Vth_1) \end{cases} \quad (4.56)$$

Proof:

$$\begin{cases} \dot{V}_1 = \dfrac{1}{C_1} K_1[S^2(V_{dd} - V_2 - Vth_1)] - \dfrac{1}{C_1} K_2[S^2(V_2 - Vth_2)] + \dfrac{I_{noise}}{C_1} \\ \dot{V}_2 = \dfrac{1}{C_2} K_3[S^2(V_{dd} - V_1 - Vth_3)] - \dfrac{1}{C_2} K_4[S^2(V_1 - Vth_4)] \end{cases}$$

$$\begin{cases} \dfrac{1}{C_1} K_1(V_{dd} - V_2 - Vth_1)^2 - \dfrac{1}{C_1} K_2(V_2 - Vth_2)^2 + \dfrac{I_{noise}}{C_1} = 0 \\ \dfrac{1}{C_2} K_3(V_{dd} - V_1 - Vth_3)^2 - \dfrac{1}{C_2} K_4(V_1 - Vth_4)^2 = 0 \end{cases}$$

$$\begin{cases} K_1(V_{dd} - V_2 - Vth_1)^2 - K_2(V_2 - Vth_2)^2 + I_{noise} = 0 \\ K_3(V_{dd} - V_1 - Vth_3)^2 - K_4(V_1 - Vth_4)^2 = 0 \end{cases}$$

$$K_3(V_{dd} - V_1 - Vth_3)^2 - K_4(V_1 - Vth_4)^2 = 0$$
$$\sqrt{K_3}(V_{dd} - V_1 - Vth_3) = \sqrt{K_4}(V_1 - Vth_4)$$
$$\sqrt{K_3}(V_{dd} - Vth_3) - \sqrt{K_4} Vth_4 = (\sqrt{K_3} + \sqrt{K_4})V_1$$
$$V_1 = \frac{\sqrt{K_3}(V_{dd} - Vth_3) - \sqrt{K_4} Vth_4}{\sqrt{K_3} + \sqrt{K_4}} = V_{1saddle}$$

$$K_1(V_{dd} - V_2 - Vth_1)^2 - K_2(V_2 - Vth_2)^2 + I_{noise} = 0$$
$$K_1(V_{dd} - Vth_1)^2 - 2K_1(V_{dd} - Vth_1)V_2 + K_1 V_2{}^2 - K_2 V_2{}^2 + 2K_2 Vth_2 V_2 - K_2 Vth_2{}^2 + I_{noise} = 0$$
$$(K_1 - K_2)V_2{}^2 + (2K_2 Vth_2 - 2K_1(V_{dd} - Vth_1))V_2 + K_1(V_{dd} - Vth_1)^2 - K_2 Vth_2{}^2 + I_{noise} = 0$$
$$\rightarrow AV_2{}^2 + BV_2 + C = 0$$

By applying the quadratic formula, it appears to have two roots.

$$V_2 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$= \frac{-(2K_2Vth_2 - 2K_1(V_{dd} - Vth_1)) \pm \sqrt{(2K_2Vth_2 - 2K_1(V_{dd} - Vth_1))^2 - 4(K_1 - K_2)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise})}}{2(K_1 - K_2)}$$

$$= \frac{-(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \pm \sqrt{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2 - (K_1 - K_2)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise})}}{(K_1 - K_2)}$$

Since K_1-K_2 are negative due to the fact that SRAM has strong NMOS drive strength over PMOS.

$$V_2 = \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \mp \sqrt{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2 + (K_2 - K_1)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise})}}{(K_2 - K_1)}$$

Only one root reflects the true solution. We need to choose the one that matches the root V_2=$V_{2saddle}$ when I_{noise}=0.

$$V_2 = \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \mp \sqrt{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2 + (K_2 - K_1)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + 0)}}{(K_2 - K_1)}$$

$$= \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \mp \sqrt{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2 - K_1{}^2(V_{dd} - Vth_1)^2 - K_2{}^2Vth_2{}^2 + K_1K_2((V_{dd} - Vth_1)^2 + Vth_2{}^2)}}{(\sqrt{K_2}^2 - \sqrt{K_1}^2)}$$

$$= \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \mp \sqrt{-2K_1K_2Vth_2(V_{dd} - Vth_1) + K_1K_2((V_{dd} - Vth_1)^2 + Vth_2{}^2)}}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

$$= \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1)) \mp \sqrt{K_1K_2}\sqrt{-2Vth_2(V_{dd} - Vth_1) + (V_{dd} - Vth_1)^2 + Vth_2{}^2}}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

$$= \frac{K_2Vth_2 - K_1(V_{dd} - Vth_1) \mp \sqrt{K_1K_2}(V_{dd} - Vth_1 - Vth_2)}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

If pick (+) sign, V_2 would equal to $V_{2saddle}$.

$$V_2 = \frac{K_2Vth_2 - K_1(V_{dd} - Vth_1) + \sqrt{K_1K_2}(V_{dd} - Vth_1) - \sqrt{K_1K_2}Vth_2}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

$$= \frac{(\sqrt{K_1K_2} - K_1)(V_{dd} - Vth_1) + (K_2 - \sqrt{K_1K_2})Vth_2}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

$$= \frac{\sqrt{K_1}(\sqrt{K_2} - \sqrt{K_1})(V_{dd} - Vth_1) + \sqrt{K_2}(\sqrt{K_2} - \sqrt{K_1})Vth_2}{(\sqrt{K_2} - \sqrt{K_1}) \cdot (\sqrt{K_2} + \sqrt{K_1})}$$

$$= \frac{\sqrt{K_1}(V_{dd} - Vth_1) + \sqrt{K_2}Vth_2}{(\sqrt{K_2} + \sqrt{K_1})} = V_{2saddle}$$

So the final expression for V_2 is:

$$V_2 = \frac{B + \sqrt{B^2 + (K_2 - K_1)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise})}}{(K_2 - K_1)} \tag{4. 57}$$

$$where\ \ B = K_2Vth_2 - K_1(V_{dd} - Vth_1)$$

The condition on I_{noise}:

$$(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2 + (K_2 - K_1)(K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise}) \geq 0$$

$$\frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2}{(K_2 - K_1)} + K_1(V_{dd} - Vth_1)^2 - K_2Vth_2{}^2 + I_{noise} \geq 0$$

$$I_{noise} \geq K_2Vth_2{}^2 - K_1(V_{dd} - Vth_1)^2 - \frac{(K_2Vth_2 - K_1(V_{dd} - Vth_1))^2}{(K_2 - K_1)}$$

4.5.2.3 Region 3

Assumption to theorem 4.5.6:

I_{noise} satisfies the condition:

$$-K_1(V_{dd} - Vth_1)^2 \leq I_{noise} \leq \infty \tag{4. 58}$$

V_1 satisfy the condition:

$$V_3 \geq V_{dd} - V_{th3} \tag{4. 59}$$

Theorem 4.7.3:

Then the following equations model the location of region 3 equilibrium state (V_1=V_{dd} V_2=0 when I_{noise}=0) in terms of I_{noise} and it remains a stable node.

$$\begin{cases} V_1 = \sqrt{(V_{dd} - V_{th1})^2 + \dfrac{I_{noise}}{K_1}} + V_{th1} \\ V_2 = 0 \end{cases} \tag{4. 60}$$

Proof:

$$\begin{cases} \dot{V}_1 = \dfrac{1}{C_1} K_1[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] + \dfrac{I_{noise}}{C_1} \\ \dot{V}_2 = -\dfrac{1}{C_2} K_4[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] \end{cases}$$

$$\begin{cases} \dfrac{1}{C_1} K_1[(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] + \dfrac{I_{noise}}{C_1} = 0 \\ -\dfrac{1}{C_2} K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] = 0 \end{cases}$$

$$\begin{cases} (V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2 + \dfrac{I_{noise}}{K_1} = 0 \\ (V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2 = 0 \end{cases}$$

$$(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2 = 0$$
$$(V_1 - Vth_4) = (V_1 - V_2 - Vth_4)$$
$$\Rightarrow V_2 = 0$$

$$(V_{dd} - Vth_1)^2 - (V_1 - Vth_1)^2 + \frac{I_{noise}}{K_1} = 0$$

$$(V_{dd} - Vth_1)^2 - V_1^{\,2} + 2Vth_1V_1 - Vth_1^{\,2} + \frac{I_{noise}}{K_1} = 0$$

$$V_1^{\,2} - 2Vth_1V_1 + Vth_1^{\,2} - (V_{dd} - Vth_1)^2 - \frac{I_{noise}}{K_1} = 0$$

$$\to AV_1^{\,2} + BV_1 + C = 0$$

$$V_1 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$= \frac{2Vth_1 \pm \sqrt{4Vth_1^{\,2} - 4(Vth_1^{\,2} - (V_{dd} - Vth_1)^2 - \dfrac{I_{noise}}{K_1})}}{2}$$

$$= Vth_1 \pm \sqrt{Vth_1^{\,2} - (Vth_1^{\,2} - (V_{dd} - Vth_1)^2 - \frac{I_{noise}}{K_1})}$$

$$= Vth_1 \pm \sqrt{(V_{dd} - Vth_1)^2 + \frac{I_{noise}}{K_1}}$$

$V_1=V_{dd}$ when $I_{noise}=0$

$$V_1 = Vth_1 + \sqrt{(V_{dd} - Vth_1)^2 + \frac{I_{noise}}{K_1}}$$

The condition on I_{noise} for stability:

$$(V_{dd} - Vth_1)^2 + \frac{I_{noise}}{K_1} \geq 0$$

$$I_{noise} \geq -K_1(V_{dd} - Vth_1)^2$$

Region 1 and 3 equilibra stay stable nodes and region 2 equilibrium stay as saddle as long as the input source I_{noise} is static and independent to V_1 V_2. The proofs are similar to section 4.5.1. Lastly, Fig. 4-12 summaried the derived equations in graphic. It demonstrates the movement of equilibria for 65nm technology SRAM. For postive current pumping into the V_1 node, the equilibra in region 1 and region 2 result bifurcation. If extracting current from V_1 node, the the equilibria in region 2 and region 3 meet and result bifurcation.

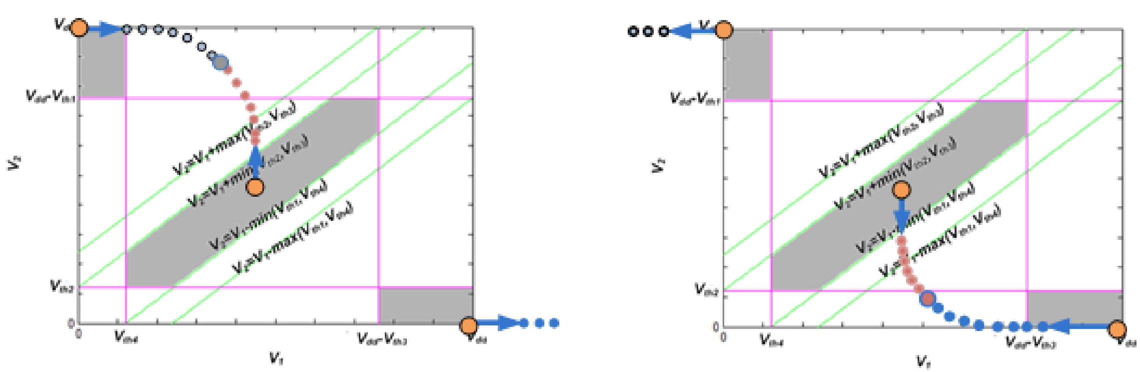


Fig. 4-12 (Left) Positive Injection in Case II. (Right) Negative Injection in Case II.

For Case I, Fig. 4-13 shows bifurcation point locations from 12 random samples. Those test samples generate from varying 65nm parameters by computer simulations. We observe test bifurcation points may occur in region 8A and 9A as shown in Fig. 4-13. In fact, the bifurcation point location for Case I psitive $I_{critical}$ will be in region 9A or 8A, it will not be in other regions such as region 2 or 3. The reason is because region 9A and 8A can have two equilibria but region 2 and 3 can only have one equilibrium (proved previously). Obviously, the region that only have one equilibrium will not have bifurcation. Therefore, region 9A and 8A are the possible regions to have a bifurcation.

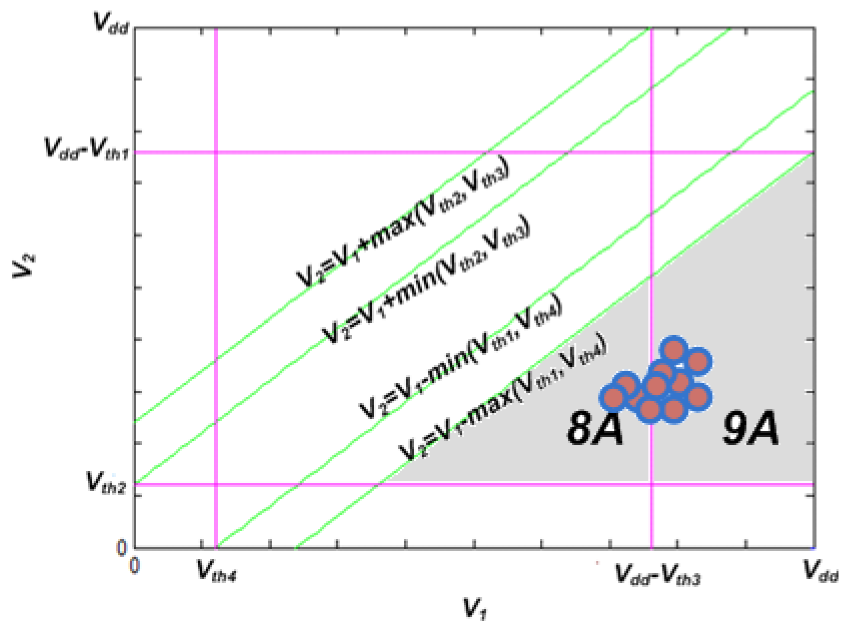


Fig. 4-13 The Possible Regions To Have Bifurcation for Case I.

Here we will demonstrate that when I_{noise} increases it moves the equilibria (the saddle and the node from region 3) to region 9A and 8A. Accordingly, it can have two equilibria in these two regions and their locations depends on I_{noise}. When these two equilibria collides into one, we have the bifurcation point. First, we focus on region 9A, and solve for equilibria.

4.5.2.4 Two equilibria in Region 9A

$$\begin{cases} \dot{V}_1 = \dfrac{1}{C_1} K_1[S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \qquad\qquad - \dfrac{1}{C_1} K_2[S^2(V_2 - Vth_2)] \\ \dot{V}_2 = -\dfrac{1}{C_2} K_4[S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \dfrac{I_{noise}}{C_2} \end{cases} \tag{4. 61}$$

$$\begin{cases} K_1[(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] - K_2[(V_2 - Vth_2)^2] = 0 \\ \qquad - K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] + I_{noise} = 0 \end{cases}$$

Continue solving the equations.

$$- K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] + I_{noise} = 0$$

$$(V_1 - Vth_4)^2 - ((V_1 - Vth_4) - V_2)^2 - \frac{I_{noise}}{K_4} = 0$$

$$(V_1 - Vth_4)^2 - ((V_1 - Vth_4)^2 - 2(V_1 - Vth_4)V_2 + V_2{}^2) - \frac{I_{noise}}{K_4} = 0$$

$$2(V_1 - Vth_4)V_2 - V_2{}^2 - \frac{I_{noise}}{K_4} = 0$$

$$V_2{}^2 - 2(V_1 - Vth_4)V_2 + \frac{I_{noise}}{K_4} = 0$$

$$V_2 = \frac{2(V_1 - Vth_4) \pm \sqrt{4(V_1 - Vth_4)^2 - 4\dfrac{I_{noise}}{K_4}}}{2}$$

$$= (V_1 - Vth_4) \pm \sqrt{(V_1 - Vth_4)^2 - \frac{I_{noise}}{K_4}}$$

$$V_2 = (V_1 - Vth_4) - \sqrt{(V_1 - Vth_4)^2 - \frac{I_{noise}}{K_4}} \tag{4. 62}$$

The (±) sign need to be determined. Intuitively, V_2 is small because $V_2 << V_1$ in region 9A, so minus sign makes sense. Next would be plugging V_2 back to solve for V_1.

$$K_1[(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] - K_2(V_2 - Vth_2)^2 = 0 \rightarrow substitute\ V_2$$

$$K_1[(V_{dd} - (V_1 - Vth_4) + \sqrt{(V_1 - Vth_4)^2 - \frac{I_{noise}}{K_4}} - Vth_1)^2$$

$$-(V_1 - (V_1 - Vth_4) + \sqrt{(V_1 - Vth_4)^2 - \frac{I_{noise}}{K_4}} - Vth_1)^2]$$

$$-K_2[((V_1 - Vth_4) - \sqrt{(V_1 - Vth_4)^2 - \frac{I_{noise}}{K_4}} - Vth_2)^2] = 0$$

This equation involves some square root terms inside square. To solve this equation for V_1, it will be transform to polynomial form, and the degree of polynomial is four. While solving with a computer software called Maple, Maple gives an output shown in Fig. 4-14. The output of Maple software shows a fourth order polynomial, and the software terminated due to its incapability of solving fourth order polynomial symbolically. The output from Fig. 4-14 is in 4th order polynomial form:

$$A4 \cdot V_1^{\ 4} + A3 \cdot V_1^{\ 3} + A2 \cdot V_1^{\ 2} + A1 \cdot V_1 + A0 = 0 \qquad (4.\ 63)$$

Where the coefficients A4 ~ A0 are:

A4=3*K1^2*K4^2-4*K1*K4^2*K2;
A3=8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd;
A2=-8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-8*K1*Vdd*Vth1*K4^2*K2;
A1=4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-8*In*K4*K1*K2*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In;

$$
\begin{aligned}
&\mathrm{RootOf}\big(-\mathit{K1}^2\,\mathit{Vdd}^4\,\mathit{K4}^2-\mathit{K2}^2\,\mathit{Vth2}^4\,\mathit{K4}^2\\
&\quad+8\,\mathit{Inoise}\,\mathit{K4}\,\mathit{K1}\,\mathit{Vdd}\,\mathit{K2}\,\mathit{Vth2}-4\,\mathit{K1}^2\,\mathit{Vdd}^3\,\mathit{K4}^2\,\mathit{Vth4}\\
&\quad+4\,\mathit{K1}^2\,\mathit{Vdd}^3\,\mathit{K4}^2\,\mathit{Vth1}-4\,\mathit{K1}^2\,\mathit{Vdd}^2\,\mathit{Vth1}^2\,\mathit{K4}^2\\
&\quad-4\,\mathit{K2}^2\,\mathit{Vth4}^2\,\mathit{K4}^2\,\mathit{Vth2}^2-4\,\mathit{K2}^2\,\mathit{Vth4}\,\mathit{Vth2}^3\,\mathit{K4}^2\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}^2\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\,\mathit{Vth2}-2\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\\
&\quad+2\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2+8\,\mathit{K1}^2\,\mathit{Vdd}^2\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{Vth1}\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth4}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2-\mathit{K2}^2\,\mathit{Inoise}^2\\
&\quad-4\,\mathit{Inoise}\,\mathit{K4}\,\mathit{K1}^2\,\mathit{Vdd}^2+\big(4\,\mathit{K2}^2\,\mathit{Vth2}^3\,\mathit{K4}^2\\
&\quad+4\,\mathit{K2}^2\,\mathit{Vth2}\,\mathit{K4}\,\mathit{Inoise}+8\,\mathit{Inoise}\,\mathit{K4}\,\mathit{K1}^2\,\mathit{Vdd}\\
&\quad-8\,\mathit{Inoise}\,\mathit{K4}\,\mathit{K1}\,\mathit{K2}\,\mathit{Vth2}+8\,\mathit{K1}^2\,\mathit{Vdd}\,\mathit{Vth1}^2\,\mathit{K4}^2\\
&\quad-4\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}+16\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\\
&\quad-4\,\mathit{K1}\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2+4\,\mathit{K1}^2\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{Vth4}\\
&\quad-4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}-12\,\mathit{K1}^2\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{Vth1}\\
&\quad+8\,\mathit{K2}^2\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{Vth2}^2+8\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\,\mathit{Vth2}\\
&\quad+8\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}-4\,\mathit{K1}\,\mathit{Vth4}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\\
&\quad+8\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}^2-4\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\\
&\quad+4\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2-4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2\\
&\quad-16\,\mathit{K1}^2\,\mathit{Vdd}\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{Vth1}+4\,\mathit{K1}^2\,\mathit{Vdd}^3\,\mathit{K4}^2\\
&\quad-8\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\big)\,\_Z+\big(\\
&\quad-8\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}-4\,\mathit{Inoise}\,\mathit{K4}\,\mathit{K1}^2\\
&\quad-4\,\mathit{K1}\,\mathit{Vth4}^2\,\mathit{K4}^2\,\mathit{K2}+8\,\mathit{K1}^2\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{Vth1}\\
&\quad+12\,\mathit{K1}^2\,\mathit{Vdd}\,\mathit{K4}^2\,\mathit{Vth1}-8\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}\\
&\quad+4\,\mathit{K1}^2\,\mathit{Vdd}\,\mathit{K4}^2\,\mathit{Vth4}-16\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\\
&\quad-4\,\mathit{K1}\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}+2\,\mathit{K1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2\\
&\quad-4\,\mathit{K1}^2\,\mathit{Vth1}^2\,\mathit{K4}^2-2\,\mathit{K1}^2\,\mathit{Vdd}^2\,\mathit{K4}^2-4\,\mathit{K2}^2\,\mathit{K4}^2\,\mathit{Vth2}^2\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}^2\,\mathit{K4}^2\,\mathit{K2}+6\,\mathit{K1}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\big)\,\_Z^2\\
&\quad+\big(8\,\mathit{K1}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}-4\,\mathit{K1}^2\,\mathit{K4}^2\,\mathit{Vth1}\\
&\quad+4\,\mathit{K1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}+8\,\mathit{K1}\,\mathit{Vth4}\,\mathit{K4}^2\,\mathit{K2}\\
&\quad-4\,\mathit{K1}^2\,\mathit{Vth4}\,\mathit{K4}^2-4\,\mathit{K1}^2\,\mathit{Vdd}\,\mathit{K4}^2\big)\,\_Z^3+\big(\\
&\quad-4\,\mathit{K1}\,\mathit{K4}^2\,\mathit{K2}+3\,\mathit{K1}^2\,\mathit{K4}^2\big)\,\_Z^4\\
&\quad-8\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}^2\\
&\quad-8\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth4}\,\mathit{Vth2}\\
&\quad+4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}\,\mathit{K2}\,\mathit{Inoise}\\
&\quad-4\,\mathit{K1}\,\mathit{Vdd}\,\mathit{Vth1}\,\mathit{K4}^2\,\mathit{K2}\,\mathit{Vth2}^2\\
&\quad-4\,\mathit{K2}^2\,\mathit{Vth4}\,\mathit{Vth2}\,\mathit{K4}\,\mathit{Inoise}-2\,\mathit{K2}^2\,\mathit{Inoise}\,\mathit{Vth2}^2\,\mathit{K4}\big)
\end{aligned}
$$

Fig. 4-14 Output of Maple Software.

```
A0=8*In*K4*K1*Vdd*K2*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2
*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vdd^2*Vth1*K4^2*Vt
h4-8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-
4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-
4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-K1^2*Vdd^4*K4^2-
K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In;
```

In order to solve fourth order polynomial roots, we apply Ferrari's theorem, which enable us to plug in the coefficient to obtain analytic solutions. The Ferrari's theorem is a general theorem for fourth order solutions. [46]

Ferrari's theorem:

For a general quartic equation (4th order polynomial equation) $Ax^4 + Bx^3 + Cx^2 + Dx + E = 0$, *the solution can be found by means of following calculations:*

$$\alpha = -\frac{3B^2}{8A^2} + \frac{C}{A}$$

$$\beta = \frac{B^3}{8A^3} - \frac{BC}{2A^2} + \frac{D}{A}$$

$$\gamma = -\frac{3B^4}{256A^4} + \frac{CB^2}{16A^3} - \frac{BD}{4A^2} + \frac{E}{A}$$

If β=0, then the solutions are:

$$x = -\frac{B}{4A} \pm_s \sqrt{\frac{-\alpha \pm_t \sqrt{\alpha^2 - 4\gamma}}{2}}$$

Otherwise,

$$P = -\frac{\alpha^2}{12} - \gamma$$
$$Q = -\frac{\alpha^3}{108} + \frac{\alpha\gamma}{3} - \frac{\beta^2}{8}$$
$$R = \frac{Q}{2} \pm \sqrt{\frac{Q^2}{4} + \frac{P^3}{27}} \quad \textit{either sign will do}$$
$$U = \sqrt[3]{R} \quad \textit{either root will do}$$

$$y = -\frac{5}{6}\alpha - U + \begin{cases} 0 & U = 0 \\ \frac{P}{3U} & U \neq 0 \end{cases}$$
$$W = \sqrt{\alpha + 2y}$$

The final solution:

$$x = -\frac{B}{4A} + \frac{\pm_s W \pm_t \sqrt{-(3\alpha + 2y \pm_s \frac{2\beta}{W})}}{2}$$

The solutions on V_1 for equation (4.61) come with 4 roots, and each root is few pages long equation. Not every root is meaningfuil in this case. We need to select the appropriete root by evaluating the solutions at nominal 65nm technology parameters. To do that, we give an estimate for a reasonable I_{noise}. From the data in Chapter III, we know the $I_{critical}$ is around 4.96e-4(A). That would be a reasonable number because we know $I_{critical}$ will make equilibria to bifurcate in region 9A. After evaluation, two roots are appropriate and two roots are not. The two appropriate roots are the solution for V_1. The reason is because there is a situation that region 2 equilibrium and region 3 equilibrium move to region 9A for a given constant I_{noise}. The equilibrium original from region 2 stay as saddle in region 9A and the equilibrium original from region 3 is also maintained as stable equilibrium in region 9A. Accordingly, one of the appropriate roots represents the saddle and the other one represents the original region 3 equilibrium. Fig.

4-15 shows the two appropriete roots. Both equations are attached in Appedix A. After we have analytical solution for V_1, analytical solution for V_2 can be acquired by plug analytical solution of V_1 to (4.61).

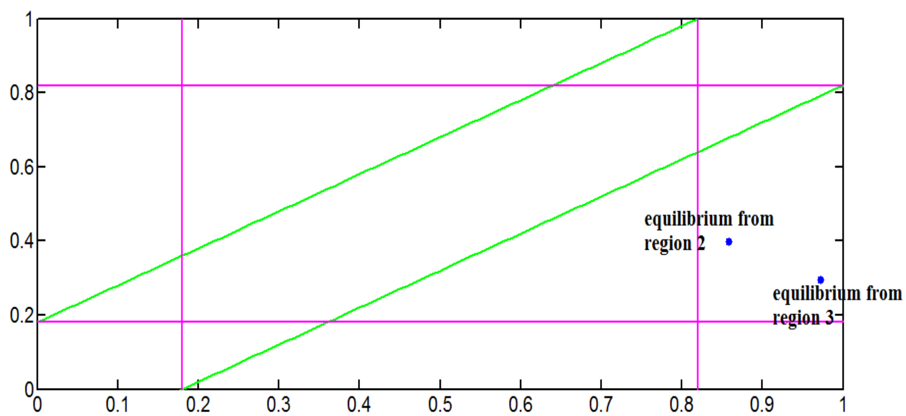


Fig. 4-15 Equilibria in Region 9A When I_{noise}=4.7e-4(A).

4.5.2.5 Region 8

The bifurcation point can occur in region 8A. For example, change V_{th3} = 0.05(V) from 65nm parameter and leave the rest unchange, bifurcation will occur in region 8A. The process to derive equilibrium point formula solution is quite similar. First, focus on region 8A that gives equation (4.64). Then find the equilibrium point by solving $\dot{V}_1 = 0$ and $\dot{V}_2 = 0$.

$$\begin{cases} \dot{V}_1 = \dfrac{1}{C_1}K_1[S^2(V_{dd}-V_2-Vth_1)-S^2(V_1-V_2-Vth_1)] \\ \qquad\qquad -\dfrac{1}{C_1}K_2[S^2(V_2-Vth_2)] \\ \dot{V}_2 = \dfrac{1}{C_2}K_3S^2(V_{dd}-V_1-Vth_3) \\ \qquad -\dfrac{1}{C_2}K_4[S^2(V_1-Vth_4)-S^2(V_1-V_2-Vth_4)]+\dfrac{I_{noise}}{C_2} \end{cases} \tag{4. 64}$$

$$\begin{cases} \dfrac{1}{C_1}K_1[(V_{dd}-V_2-Vth_1)^2-(V_1-V_2-Vth_1)^2] \\ \qquad\qquad -\dfrac{1}{C_1}K_2(V_2-Vth_2)^2=0 \\ \dfrac{1}{C_2}K_3(V_{dd}-V_1-Vth_3)^2 \\ \qquad -\dfrac{1}{C_2}K_4[(V_1-Vth_4)^2-(V_1-V_2-Vth_4)^2]+\dfrac{I_{noise}}{C_2}=0 \end{cases}$$

$$\begin{cases} K_1[(V_{dd}-V_2-Vth_1)^2-(V_1-V_2-Vth_1)^2]-K_2(V_2-Vth_2)^2=0\cdots\cdots f1 \\ K_3(V_{dd}-V_1-Vth_3)^2-K_4[(V_1-Vth_4)^2-(V_1-V_2-Vth_4)^2]+I_{noise}=0\cdots\cdots f2 \end{cases}$$

Use Maple to solve $f1$ for V_1:

$solve(f1=0, V1)$

$$\begin{aligned} V1 = \frac{1}{K1}\Big(&K1\ Vth1 + K1\ V2 \\ &+ \big(K1^2\ Vth1^2 + 2\,K1^2\ Vth1\ V2 + K1^2\ V2^2 + K1^2\ Vdd^2 \\ &- 2\,K1^2\ Vdd\ V2 - 2\,K1^2\ Vdd\ Vth1 + 2\,K1\ K2\ V2\ Vth2 \\ &- K1\ K2\ Vth2^2 - K1\ K2\ V2^2\big)^{1/2}\Big), \frac{1}{K1}\Big(K1\ Vth1 + K1\ V2 \\ &- \big(K1^2\ Vth1^2 + 2\,K1^2\ Vth1\ V2 + K1^2\ V2^2 + K1^2\ Vdd^2 \\ &- 2\,K1^2\ Vdd\ V2 - 2\,K1^2\ Vdd\ Vth1 + 2\,K1\ K2\ V2\ Vth2 \\ &- K1\ K2\ Vth2^2 - K1\ K2\ V2^2\big)^{1/2}\Big) \end{aligned}$$

The output shows two solutions. V_1 should be the first solution because V_1 suppose to be big in region 8A, and first solution has every term postivie. It can also be checked by evaluating at nominal 65nm values. Use that to plug into *f2* and solve *f2*=0. Maple responses the following message:

```
> solve(f2 = 0, V2)
```

```
RootOf((-3*K4^2*K1^2-
6*K4*K1*K3*K2+4*K1*K2*K4^2+K3^2*K2^2+4*K3*K1^2*K4)*_Z^4+(4*K3*K1^2*Vth3
*K4-4*K4*K1*Vth4*K3*K2-4*K3^2*K2^2*Vth2-16*K3*K1^2*Vdd*K4-
4*Vth4*K1^2*K4^2+4*K3^2*K1*Vth3*K2-4*Vth1*K1^2*K4^2+8*K3*K1^2*Vth1*K4-
4*K4*K1*Vth1*K3*K2+8*K4*K1^2*Vth4*K3+8*K1*K2*K3*Vdd*K4+8*K4^2*Vdd*K1^2+
12*K3*K2*Vth2*K4*K1-8*K1*K2*Vth2*K4^2-
8*K1*K2*K3*Vth3*K4)*_Z^3+(20*K3*K1^2*Vdd^2*K4+4*K3*K1^2*Vth1^2*K4-
8*K3^2*K2*Vth2*K1*Vth3+8*K3*K1^2*Vth3*K4*Vth4-6*K4*K1*K3*K2*Vth2^2-
24*K3*K1^2*Vdd*Vth1*K4+16*K1*K2*Vth2*K3*Vth3*K4+16*K3*K1^2*Vth1*K4*Vth4
+4*K1*K2*Vth2^2*K4^2-2*In*K1*K3*K2-
2*K3*Vth3^2*K1^2*K4+2*K3^2*Vth3^2*K1*K2-16*K3*K1^2*Vdd*K4*Vth4-
4*K3^2*Vdd*K1*Vth3*K2-
12*K3*Vdd*K1^2*Vth3*K4+4*K3*K1^2*Vth1*Vth3*K4+4*K3^2*K1*Vth1*Vth3*K2-
16*K1*K2*Vth2*K3*Vdd*K4+8*K3*K2*Vth2*K4*K1*Vth4+8*K3*K2*Vth2*K4*K1*Vth1
-4*K4^2*Vdd^2*K1^2+8*K4^2*Vth1*Vdd*K1^2-2*K4*K1^2*In-
8*Vth4*Vth1*K1^2*K4^2+4*Vth4^2*K1^2*K4^2+6*K3^2*K2^2*Vth2^2+4*K3^2*K1^2
*Vth3^2+4*In*K1^2*K3)*_Z^2+(-
8*K1^2*Vdd^3*K3*K4+8*K3*K1^2*Vth1*In+8*K3^2*K1^2*Vth1*Vth3^2+4*K4*K1^2*
Vth4*K3*Vth3^2+8*K1^2*Vdd^2*K3*Vth3*K4+8*K4*K1^2*Vth4*K3*Vth1^2-
4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*Vdd*K1*Vth3-
8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K3^2*K1*Vth3*K2*Vth2^2-
4*K3^2*K2^2*Vth2^3-4*K4*K1^2*Vth1*K3*Vth3^2-8*K3*K1^2*Vdd*Vth1^2*K4-
16*K3*K1^2*Vdd*Vth1*K4*Vth4+8*K1*K2*Vth2^2*K3*Vdd*K4-
8*K1*K2*Vth2^2*K3*Vth3*K4+16*K4*K1^2*Vth1*K3*Vdd^2+4*K3*K2*Vth2*In*K1-
4*K3^2*K2*Vth2*Vth3^2*K1-
4*K4*K1*Vth1*K3*K2*Vth2^2+8*K3*K1^2*Vth1*Vth3*K4*Vth4-
8*K3*Vdd*K1^2*Vth3*K4*Vth1-
8*K3*Vdd*K1^2*Vth3*K4*Vth4+8*K4*K1^2*Vth4*K3*Vdd^2+4*In*Vth4*K1^2*K4-
4*In*Vth1*K1^2*K4+4*K3^2*K1^2*Vth3^3-8*K3*K1^2*Vdd*In-
8*K3^2*K1^2*Vdd*Vth3^2+4*K3*K1^2*Vth3*In)*_Z-
4*K3^2*Vdd*K1^2*Vth3^3+4*K3^2*Vdd^2*K1^2*Vth3^2+In^2*K1^2+4*K3^2*K1^2*V
th1*Vth3^3+K3^2*K2^2*Vth2^4-8*K3*K1^2*Vdd*Vth1*In-
8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3*Vdd*K1^2*Vth3*In-
2*In*K1*K3*K2*Vth2^2+4*K3*K1^2*Vth1*Vth3*In+4*K3^2*K1^2*Vth1^2*Vth3^2+4
*In*K1^2*K3*Vdd^2+4*K3^2*K1*Vth1*Vth3*K2*Vth2^2+2*In*K1^2*K3*Vth3^2+K3^
2*Vth3^4*K1^2+4*In*K1^2*K3*Vth1^2+2*K3^2*Vth3^2*K1*K2*Vth2^2)
```

It appears to have 4th order polynomial form,

$$A4 \cdot V_2^{\,4} + A3 \cdot V_2^{\,3} + A2 \cdot V_2^{\,2} + A1 \cdot V_2 + A0 = 0 \qquad (4.\ 65)$$

```
A4=-K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4;
A3=-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8
*K1*K2*Vth2*K4^2-8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-
8*K1*K2*K4*K3*Vdd+8*K1*K2*K4*K3*Vth3-8*K4*K1^2*Vth4*K3;
A2=8*Vth1*Vth4*K1^2*K4^2-8*K4^2*Vth1*Vdd*K1^2+4*K4^2*Vdd^2*K1^2-
4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2-
6*K3^2*K2^2*Vth2^2-4*K3*K1^2*Vth1^2*K4-
2*K3^2*K2*Vth3^2*K1+2*In*K1*K3*K2+2*K3*Vth3^2*K1^2*K4-
20*K3*K1^2*Vdd^2*K4-
4*K1*K2*Vth2^2*K4^2+8*K3^2*K2*Vth2*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd-
8*K4*K1^2*Vth4*K3*Vth3-8*K4*K1*Vth4*K3*K2*Vth2-16*K4*K1^2*Vth4*K3*Vth1-
8*K3*K2*Vth2*K4*K1*Vth1+24*K3*K1^2*Vdd*K4*Vth1+4*K3^2*Vdd*K1*Vth3*K2+12
*K3*Vdd*K1^2*Vth3*K4-4*K3^2*K1*Vth1*Vth3*K2-
4*K3*K1^2*Vth1*Vth3*K4+6*K3*K2*Vth2^2*K4*K1+16*K1*K2*Vth2*K4*K3*Vdd-
16*K1*K2*Vth2*K4*K3*Vth3;
A1=-4*In*Vth4*K1^2*K4+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
8*In*K1^2*K3*Vth1-4*In*K1^2*K3*Vth3-8*K3^2*K1^2*Vth1*Vth3^2-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3+8*K1^2*Vdd^3*K4*K3+8*In*K1^2*K3*V
dd-4*K4*K1^2*Vth4*K3*Vth3^2+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3-8*K4*K1^2*Vth4*K3*Vth1^2-
8*K4*K1^2*Vth4*K3*Vdd^2+8*K4*K1^2*Vth4*K3*Vdd*Vth3-4*In*K1*K3*K2*Vth2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K3^2*K2*Vth2*V
th3^2*K1-8*K1^2*Vdd^2*K4*K3*Vth3+4*K4*K1^2*Vth1*K3*Vth3^2-
16*K4*K1^2*Vth1*K3*Vdd^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*K3*K2*Vth2^2*K4*K1*V
th1+8*K3*Vdd*K1^2*Vth3*K4*Vth1-
8*K1*K2*Vth2^2*K4*K3*Vdd+8*K1*K2*Vth2^2*K4*K3*Vth3+16*K4*K1^2*Vth4*K3*V
dd*Vth1-4*K3^2*K1*Vth3*K2*Vth2^2;
A0=4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-
2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K
3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*Vdd*K1^2*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3
^2-4*In*K1^2*K3*Vdd^2-4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-
4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2;
```

The V_2 solutions are attached in Appedix A. Unfortunately, the equilibrium point formula for region 8A and 9A are messy, which are pages long. If going through the same process at regions 9B and region 8B, there is only one meaningful solution, so bifurcation will not happen there. From all possible regions to result bifurcation (2, 8A, 8B, 9A, 9B, 7A, 3), only region 9A and 8A can have two equilibria, so bifurcation can only happen in region 8A or 9A. Therefore, for Case I, the positive $I_{critical}$ bifurcation point happen in region 8A or 9A. Likewise, the negative $I_{critical}$ bifurcation point happen in region 4A or 5A. For Case II, the bifurcation would happen in 5A 6A for positive $I_{critical}$, and 7A 8A for negative $I_{critical}$. (Proof will be similar)

4.6. ANALYTICAL SOLUTION OF THE BIFURCATION POINT AND CRITICAL CURRENT

The noise margin of a SRAM device is said to be the minimum amount of static current that would result in a bifurcation. This current is denoted as critical current, I_c. In order to have I_c analytically expressed only in terms of systerm parameters, the bifurcation point need to be known first. So it would have totally three unknowns, those are bifurcation point (V_1, V_2) and I_c. From the provided general state space equations for Case I (4.66) or Case II (4.67), the state equations can solve for equilibria in terms of I_c but I_c can not be expressed in terms of only system paramters. Thus, it needs one more condition to help solving I_c. Fortunately, theorem 4.6 [47] can be applied here.

$$\begin{cases} C_1\dot{V}_1 = f(V_1, V_2) \\ C_2\dot{V}_2 = g(V_1, V_2) + I_c \end{cases} \tag{4.66}$$

$$\begin{cases} C_1\dot{V}_1 = f(V_1, V_2) + I_c \\ C_2\dot{V}_2 = g(V_1, V_2) \end{cases} \tag{4.67}$$

Theorem 4.6: [47]

In general, for an N dimensional nonlinear system, the saddle-node bifurcation happens when its Jacobian matrix becomes a singular matrix.

Because of this theorem, a third equation can be constructed to solve all three unknowns. Since singular matrix has zero determinant, the third equation is taking the determinant of Jacobian matrix and set it to zero; it represent by $h(V_1,V_2)$. The third equation is shown below:

$$h(V_1,V_2) = \det\begin{pmatrix} \frac{\partial f}{\partial V_1} & \frac{\partial f}{\partial V_2} \\ \frac{\partial g}{\partial V_1} & \frac{\partial g}{\partial V_2} \end{pmatrix} = \frac{\partial f}{\partial V_1}\frac{\partial g}{\partial V_2} - \frac{\partial f}{\partial V_2}\frac{\partial g}{\partial V_1} = 0 \qquad (4.\ 68)$$

4.6.1 Analytical Solution of the Saddle Node Bifurcation Point in Region 9A

As dicussed previously, the SRAM will result saddle node bifurcation when the a noise injection current amplitude reaches critical current amplitude. In Fig. 4-16, it demonstrates saddle node bifurcation in phase portrait on a 65nm SRAM with noise injection to V_2 node. As I_{noise} increases, the nullcline would change. When I_{noise} is lower than I_c, the two nullclines (red and green lines) cross each other and form three equilibria. Once the I_{noise} reaches I_c, as shown in Fig. 4-17, two nullclines does not form three equilibria; instead, nullcline curves just tangent to each other at one point right at I_c. For further increment on I_{noise}, the two nullcline curves disconnect. In other words, the bifurcation point is the point that two nullcline curves tangent to each other.

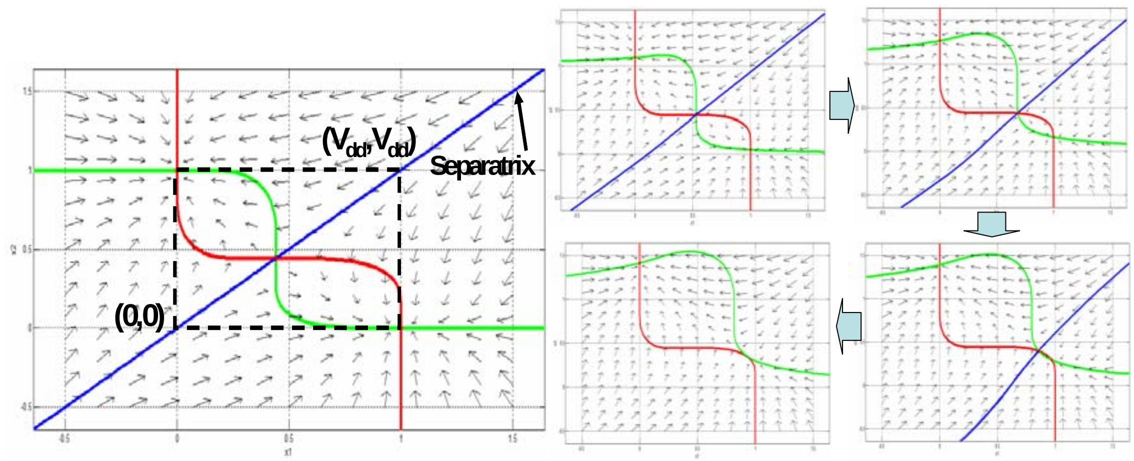


Fig. 4-16 (Left) Phase Portrait of a Symmetric cell. (Right) Saddle Node Bifurcation as I_{noise} Increases.

For the nominal 65nm technology SRAM parameters, the I_c is 4.96e-4(A) and the bifurcation occur in region 9A. From the derived equilibrium point equation for region 9A from section 4.6, the bifurcation point locates at (0.93,0.35). The goal is to find the analytical solution for I_c in this region. For that, we will get modeling equations for region 9A and apply theorem 4.7. Then we will try to get nullcline equations and look at the tangent part of the curve.

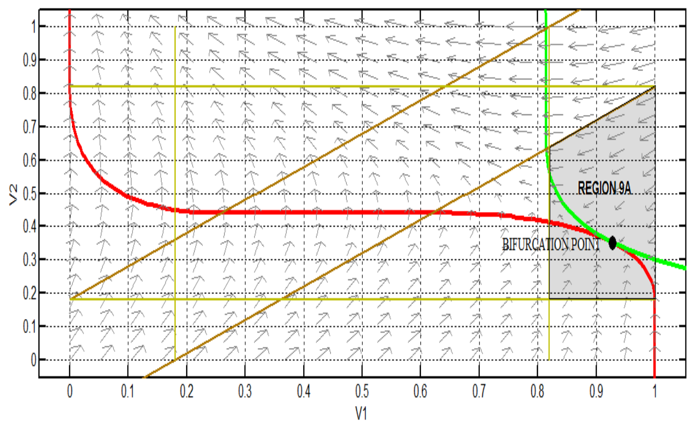


Fig. 4-17 The Bifurcation Point For 65nm SRAM.

.

$$\begin{cases} \dot{V}_1 = f(V_1, V_2, I_c) = \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \qquad - \frac{1}{C_1} K_2 [S^2(V_2 - Vth_2)] \\ \dot{V}_2 = g(V_1, V_2, I_c) = -\frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \frac{I_c}{C_2} \end{cases} \quad (4.69)$$

$$\begin{cases} f(V_1, V_2, I_c) = \frac{1}{C_1} K_1 [S^2(V_{dd} - V_2 - Vth_1) - S^2(V_1 - V_2 - Vth_1)] \\ \qquad - \frac{1}{C_1} K_2 [S^2(V_2 - Vth_2)] = 0 \\ g(V_1, V_2, I_c) = -\frac{1}{C_2} K_4 [S^2(V_1 - Vth_4) - S^2(V_1 - V_2 - Vth_4)] + \frac{I_c}{C_2} = 0 \\ h(V_1, V_2, I_c) = \det \begin{pmatrix} \frac{\partial f}{\partial V_1} & \frac{\partial f}{\partial V_2} \\ \frac{\partial g}{\partial V_1} & \frac{\partial g}{\partial V_2} \end{pmatrix} = \frac{\partial f}{\partial V_1} \frac{\partial g}{\partial V_2} - \frac{\partial f}{\partial V_2} \frac{\partial g}{\partial V_1} = 0 \end{cases} \quad (4.70)$$

The calculation would set $f(V_1, V_2, I_c)=0$ and $g(V_1, V_2, I_c)=0$ as solving for equilibrium point. With the third equation in hand, we have three equations to find bifurcation point (V_1, V_2) and critical current I_c.

$$\frac{\partial f}{\partial V_1} = -2\frac{K_1}{C_1}(V_1 - V_2 - V_{th1})$$

$$\frac{\partial f}{\partial V_2} = -2K_1[(V_{dd} - V_2 - Vth_1) - (V_1 - V_2 - Vth_1)] - 2K_2(V_2 - V_{th2})$$

$$= -2\frac{K_1}{C_1}(V_{dd} - V_1) - 2\frac{K_2}{C_1}(V_2 - V_{th2})$$

$$\frac{\partial g}{\partial V_1} = -2\frac{K_4}{C_2}[(V_1 - Vth_4) - (V_1 - V_2 - Vth_4)] = -2\frac{K_4 V_2}{C_2}$$

$$\frac{\partial g}{\partial V_2} = -2\frac{K_4}{C_2}(V_1 - V_2 - Vth_4)$$

$$\det\begin{pmatrix} \frac{\partial f}{\partial V_1} & \frac{\partial f}{\partial V_2} \\ \frac{\partial g}{\partial V_1} & \frac{\partial g}{\partial V_2} \end{pmatrix} = 0$$

$$\Rightarrow \det\begin{pmatrix} -2\frac{K_1}{C_1}(V_1 - V_2 - V_{th1}) & -2\frac{K_1}{C_1}(V_{dd} - V_1) - 2\frac{K_2}{C_1}(V_2 - V_{th2}) \\ -2\frac{K_4 V_2}{C_2} & -2\frac{K_4}{C_2}(V_1 - V_2 - Vth_4) \end{pmatrix} = 0$$

$$\Rightarrow 4\frac{K_1 K_4}{C_1 C_2}(V_1 - V_2 - V_{th1})(V_1 - V_2 - Vth_4) - (\frac{4}{C_1 C_2}(K_1(V_{dd} - V_1) + K_2(V_2 - V_{th2})) * K_4 V_2) = 0$$

$$\Rightarrow K_1(V_1 - V_2 - V_{th1})(V_1 - V_2 - Vth_4) - V_2(K_1(V_{dd} - V_1) + K_2(V_2 - V_{th2})) = 0$$

Simplified form for $f(V_1, V_2, I_c)$, $g(V_1, V_2, I_c)$ and $h(V_1, V_2, I_c)$.

$$f(V_1, V_2, I_c) = K_1[(V_{dd} - V_2 - Vth_1)^2 - (V_1 - V_2 - Vth_1)^2] - K_2(V_2 - Vth_2)^2 = 0 \quad (4.\ 71)$$

$$g(V_1, V_2, I_c) = -K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] + I_c = 0 \quad (4.\ 72)$$

$$h(V_1, V_2, I_c) = K_1(V_1 - V_2 - V_{th1}) \cdot (V_1 - V_2 - Vth_4) - V_2(K_1(V_{dd} - V_1) + K_2(V_2 - V_{th2})) = 0 \quad (4.\ 73)$$

In order to get nullcline equation, solve V_2 in terms of V_1 for *f, g, h* functions. This will be solving quardratic functions for V_2 for *f g* and *h* functions. Quardratic function gives two roots. Since we are focusing on 65nm SRAM I_c analytical solution, we evaluate the roots at 65nm values and pick the meaningful root to be V_2f V_2g *and* V_2h, then plot them on Fig. 4-18. The V_2f *and* V_2g in Fig. 4-18 are expected to look like the nullclines from Fig. 4-17; however, they are not quite the same. Since functions (4.71) (4.72) (4.73) are valid only for region 9A, only the curve inside region 9A is preserved as the nullclines.

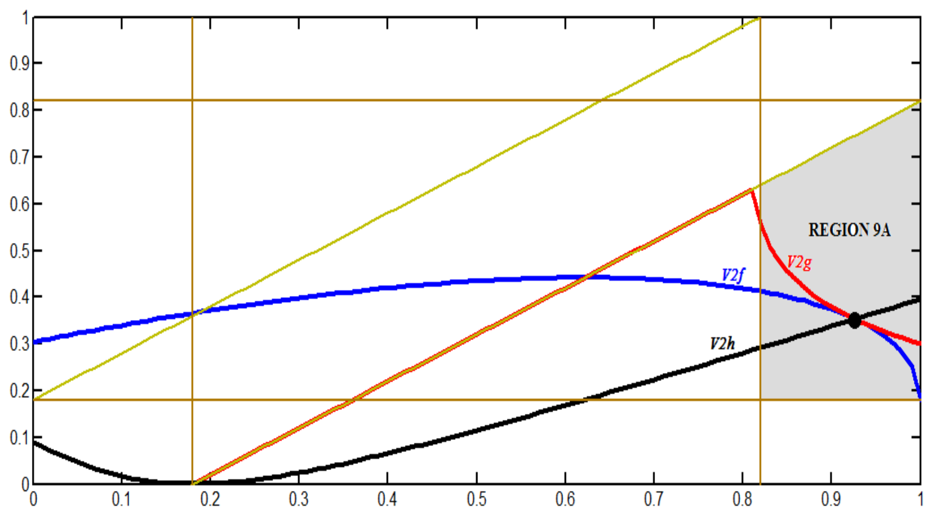


Fig. 4-18 The Plot of V_2f, V_2g and V_2h.

$$V_2 f = \left\{ \frac{1}{2K_2}(-2K_1V_{dd} + 2K_2V_{th2} + 2K_1V_1 \pm 2\sqrt{\begin{matrix}K_1{}^2V_{dd}{}^2 - 2K_1V_{dd}K_2V_{th2} - 2K_1{}^2V_{dd}V_1 + 2K_2V_{th2}K_1V_1 + K_1{}^2V_1{}^2 + K_2K_1V_{dd}{}^2 \\ + 2K_2K_1V_1V_{th1} - 2K_2K_1V_{dd}V_{th1} - K_2K_1V_1{}^2\end{matrix}}) \right.$$

$$V_2 g = \left\{ \frac{1}{2K_4}(2K_4V_1 - 2K_4 * V_{th4} \pm 2\sqrt{K_4{}^2V_1{}^2 - 2K_4{}^2V_1V_{th4} + K_4{}^2V_{th4}{}^2 - K_4I_{nose}}) \right.$$

$$V_2h = \begin{cases} \frac{1}{2(K_1 - K_2)}(-K_1V_{th4} + K_1V_1 - K_1V_{th1} + K_1V_{dd} - K_2V_{th2} \\ \pm \sqrt{\begin{matrix} 2K_1{}^2V_{dd}V_1 + 2K_1{}^2V_{th4}V_1 - 2K_1V_{dd}K_2V_{th2} - 2K_2V_{th2}K_1V_1 + K_1{}^2V_{dd}{}^2 + K_2{}^2V_{th2}{}^2 \\ -3K_1{}^2V_1{}^2 + 2K_1V_{th4}K_2V_{th2} + 2K_1V_{th1}K_2V_{th2} + K_1{}^2V_{th4}{}^2 + K_1{}^2V_{th1}{}^2 - 2K_1{}^2V_{th4}V_{th1} \\ -2K_1{}^2V_{th4}V_{dd} + 2K_1{}^2V_1V_{th1} - 2K_1\text{^}2V_{th1}V_{dd} + 4K_2K_1V_1{}^2 - 4K_2K_1V_1V_{th1} \\ -4K_2K_1V_1V_{th4} + 4K_2K_1V_{th1}V_{th4} \end{matrix}}\,) \end{cases}$$

Since bifurcation point is at tangent point of nullclines, and after the bifurcation, all the vector field vectors are pointing toward the remaining stable node; we observe that the slope at the bifurcation point on nullclines is around -1 from Fig. 4-17. With that observation, we can evaluate the slope by taking derivative on V_2f and V_2g and evaluate at nominal values since we know the bifurcation point and I_c at nominal value. The slope is -0.8925 for V_2f and -0.902 for V_2g and 0.579 for V_2h at nominal. When trying various of parameters, the slope are pretty much in the range [-0.89 -1.02] for both V_2f and V_2g, [0.43, 0.58] for V_2h. Therefore, we can use those observations to find analytical solution for I_c. Here is the proposed way to get I_c.

1. Take $\frac{dV_2f}{dV_1} = -s$, s=1 for default, then solve for V_1. It will be quardratic polynomial and have two V_1 solutions. Pick the one that gives the correct result when evalution at nominal value. V_1 should be in terms of system parameters.
2. Use that V_1 and plug in V_2h which gives V_2. V_2 is also in terms of system parameters.
3. Finally use (4.72) to get I_c.

This approach is based on the observation that the bifurcation point is located at a tangent point which has slope of -1. It finds the V_1 that gives V_2f the -1 slope, then V_2 is found based on that V_1. One bad news for this method is the slope may not be -1 once the system been perturbed. The slope varies in a range [-0.7 -1.1] when perturb 65nm SRAM. If the true slope is off from -1, the V_2 will be off and I_c will be off too. Fortunately, the variation on the slope does not change V_1 too much most of time. Table 5 shows 36 perturbed parameter samples, and the % error on I_c is around 7.7% for s=1.

Table 5 The Data of Analytical I_c and Real I_c Under Various Perturbed Parameters.

						Numerical			Analytical			% error on V1	% error on V2	% error on Ic
Vth1	Vth2	Vth4	K1	K2	K4	V1	V2	Ic	V1	V2	Ic	(V1n-V1a)/V1n	(V1n-V1a)/V1n	(Ic1-Ic2)/Ic1
1	1	1	1	1	1	0.926577	0.353311	0.000496	0.935581	0.358531	5.09E-04	9.72E-03	1.48E-02	0.026158132
			1-50%	1+50%		0.928096	0.286516	0.000427	0.9648	0.274616	4.38E-04	3.95E-02	4.15E-02	0.026061729
			1-50%	1+50%	1+50%	0.928096	0.286516	0.00064	0.9648	0.2993	7.02E-04	3.95E-02	4.46E-02	0.096971397
			1-50%	1+50%	1-50%	0.928096	0.286516	0.000213	0.9648	0.239859	1.96E-04	3.95E-02	1.63E-01	0.079750643
			1-50%	1-50%	1+50%	0.926577	0.353311	0.000744	0.935581	0.448753	8.81E-04	9.72E-03	2.70E-01	0.18385043
			1-50%	1-50%	1-50%	0.926577	0.353311	0.000248	0.935581	0.358531	2.55E-04	9.72E-03	1.48E-02	0.026158132
			1+50%	1-50%	1+50%	0.936597	0.43342	0.000865	0.915891	0.506198	9.03E-04	2.21E-02	1.68E-01	0.044402068
			1+50%	1+50%	1+50%	0.926577	0.353311	0.000744	0.935581	0.358531	7.64E-04	9.72E-03	1.48E-02	0.026158132
1+50%	1+50%					0.954025	0.388108	0.000555	0.953699	0.387912	5.54E-04	3.42E-04	5.03E-04	0.000897232
1+50%	1-50%					0.89722	0.285952	0.000405	0.935581	0.307525	4.56E-04	4.28E-02	7.54E-02	0.127084246
1-50%	1-50%					0.894925	0.32148	0.000439	0.917463	0.334097	4.69E-04	2.52E-02	3.92E-02	0.069683311
1+50%	1+50%	1+50%				0.963359	0.377874	0.00047	0.953699	0.372102	4.56E-04	1.00E-02	1.53E-02	0.028499561
1-50%	1-50%	1-50%				0.880392	0.332678	0.000511	0.917463	0.353354	5.66E-04	4.21E-02	6.21E-02	0.10764919
1-50%	1-50%	1-50%	1-50%	1-50%	1-50%	0.880392	0.332678	0.000256	0.917463	0.353354	2.83E-04	4.21E-02	6.21E-02	0.107649191
1-50%	1-50%	1-50%	1+50%	1+50%	1+50%	0.880392	0.332678	0.000767	0.917463	0.353354	8.50E-04	4.21E-02	6.21E-02	0.10764919
1-50%	1+50%	1-50%	1+50%	1-50%	1+50%	0.943891	0.513059	0.001133	0.915891	0.583041	0.001151265	2.97E-02	1.36E-01	0.016568891
1+50%	1-50%	1+50%	1-50%	1+50%	1-50%	0.903598	0.21009	0.000137	0.9648	0.170028	1.28E-04	6.77E-02	1.91E-01	0.066306281
1+50%	1-50%	1-50%	1-50%	1+50%	1+50%	0.874796	0.222859	0.000555	0.9648	0.254539	7.03E-04	1.03E-01	1.42E-01	0.267947218
1+50%	1-50%	1+50%	1-50%	1+50%	1+50%	0.903598	0.21009	0.00041	0.9648	0.231476	4.95E-04	6.77E-02	1.02E-01	0.207083707
1+50%	1-50%	1+50%	1-50%	1+50%	1-50%	0.903598	0.21009	0.000137	0.9648	0.170028	1.28E-04	6.77E-02	1.91E-01	0.066306281
1+50%	1-50%	1-50%	1-50%	1+50%	1-50%	0.874796	0.222859	0.000185	0.9648	0.184943	1.78E-04	1.03E-01	1.70E-01	0.035847943
1-50%	1+50%	1+50%	1+50%	1-50%	1+50%	0.963933	0.476749	0.000803	0.915891	0.51006	7.37E-04	4.98E-02	6.99E-02	0.082069779
1-50%	1+50%	1-50%	1+50%	1-50%	1+50%	0.943891	0.513059	0.001133	0.915891	0.583041	0.001151265	2.97E-02	1.36E-01	0.016568891
1+50%	1+50%	1-50%	1+50%	1+50%	1-50%	0.944861	0.396654	0.000321	0.953699	0.335142	2.87E-04	9.35E-03	1.55E-01	0.104121404
1+50%	1+50%	1-50%	1-50%	1+50%	1-50%	0.952738	0.343721	0.000292	0.9747	0.305879	2.76E-04	2.31E-02	1.10E-01	0.057433055
1+50%	1+50%	1-50%	1-50%	1-50%	1-50%	0.944861	0.396654	0.000321	0.953699	0.401948	3.28E-04	9.35E-03	1.33E-02	0.022902301
1+50%	1-50%	1-50%	1-50%	1-50%	1-50%	0.88425	0.29405	0.000234	0.935581	0.323008	2.72E-04	5.80E-02	9.85E-02	0.161024778
1-50%	1+50%	1-50%	1-50%	1-50%	1-50%	0.938441	0.431867	0.000336	0.935581	0.430179	3.34E-04	3.05E-03	3.91E-03	0.007083553
													average error	0.077495952

The analytical solution for V_1:

$$V1 = \Big(K2^2\, Vth1\, n^2 + K2\, Vth2\, K1 + 2\, K2\, Vth1\, K1\, n - 2\, K1^2\, Vdd\, n - K1^2\, Vdd + K2^2\, Vth2\, n^2 + 2\, K2\, Vth2\, K1\, n + K2\, K1\, Vth1 - K1\, Vdd\, n^2\, K2 + \big(8\, K2^3\, Vth1\, n^3\, Vth2\, K1 + 4\, K2^2\, Vth2\, K1^2\, Vth1\, n - 4\, K2\, Vth2\, K1^3\, Vdd\, n - 10\, K2^2\, Vth2\, K1^2\, Vdd\, n^2 - 4\, K2\, Vth1\, K1^3\, n\, Vdd + 2\, K2^2\, Vth1^2\, K1^2\, n - 2\, K2^3\, K1\, Vdd\, Vth1\, n^2 + 4\, K1^3\, n\, K2\, Vth2\, Vth1 - 2\, K2^3\, K1\, Vdd\, Vth2\, n^2 - 4\, K2^2\, K1^2\, Vdd\, Vth2\, n - 4\, K2^2\, K1^2\, Vdd\, Vth1\, n - 8\, K1\, n^3\, K2^3\, Vdd\, Vth1 - 8\, K1\, n^3\, K2^3\, Vdd\, Vth2 + 2\, K1^3\, K2\, Vth2\, Vth1 - 2\, n^4\, K2^4\, Vdd\, Vth1 + K2\, K1^3\, Vdd^2 + K2^4\, Vth2^2\, n^4 + n^4\, K2^4\, Vdd^2 + K1^3\, K2\, Vth1^2 + 2\, K2^2\, Vth2^2\, K1^2\, n + K2^3\, Vth2^2\, K1\, n^2 - 2\, K2\, Vth2\, K1^3\, Vdd + K2^3\, Vth1^2\, n^2\, K1 + 2\, K2^4\, Vth1\, n^4\, Vth2 + 4\, K2^3\, Vth1^2\, n^3\, K1 + K1^3\, K2\, Vth2^2 + 5\, K2^2\, Vth1^2\, K1^2\, n^2 - 2\, n^4\, K2^4\, Vdd\, Vth2 - 2\, K1^3\, Vdd\, K2\, Vth1 + 10\, K2^2\, Vth1\, K1^2\, n^2\, Vth2 + 4\, K2^3\, Vth2^2\, n^3\, K1 + 5\, K2^2\, Vth2^2\, K1^2\, n^2 + 2\, K2^2\, K1^2\, Vdd^2\, n + K2^3\, K1\, Vdd^2\, n^2 + 5\, K2^2\, K1^2\, Vdd^2\, n^2 + 2\, K2\, K1^3\, Vdd^2\, n + 4\, K1\, n^3\, K2^3\, Vdd^2 + K2^4\, Vth1^2\, n^4 + 2\, K2^3\, Vth1\, n^2\, Vth2\, K1 - 10\, K2^2\, Vth1\, n^2\, K1^2\, Vdd + 2\, K1^3\, n\, K2\, Vth1^2 + 2\, K1^3\, n\, K2\, Vth2^2\big)^{1/2}\Big) \Big/ \big(K2\, K1 + 2\, K1\, n\, K2 - 2\, K1^2\, n - K1\, n^2\, K2 - K1^2 + n^2\, K2^2\big)$$

V_1 for s=1:

$$V1 = \frac{1}{K2^2 - 3\, K1^2 + 2\, K2\, K1}\Big(3\, K2\, K1\, Vth1 + 3\, K2\, Vth2\, K1 - K1\, Vdd\, K2 + K2^2\, Vth2 + K2^2\, Vth1 - 3\, K1^2\, Vdd + \big(-14\, K2^2\, Vth2\, K1^2\, Vdd + 5\, K2^3\, K1\, Vth1^2 + 7\, K2^2\, Vth2^2\, K1^2 + 5\, K2^3\, Vth2^2\, K1 + 7\, K1^2\, Vdd^2\, K2^2 + 3\, K1^3\, Vdd^2\, K2 + 2\, K2^4\, Vth2\, Vth1 + 3\, K1^3\, K2\, Vth2^2 + 3\, K1^3\, K2\, Vth1^2 - 2\, K2^4\, Vdd\, Vth2 - 2\, K2^4\, Vdd\, Vth1 + 7\, K2^2\, K1^2\, Vth1^2 + 14\, K2^2\, K1^2\, Vth1\, Vth2 - 14\, K2^2\, K1^2\, Vth1\, Vdd + 10\, K2^3\, K1\, Vth1\, Vth2 - 6\, K2\, K1^3\, Vth1\, Vdd + K2^4\, Vdd^2 + 5\, K2^3\, K1\, Vdd^2 + K2^4\, Vth2^2 + K2^4\, Vth1^2 - 6\, K2\, Vth2\, K1^3\, Vdd - 10\, K1\, Vdd\, K2^3\, Vth2 - 10\, K1\, Vdd\, K2^3\, Vth1 + 6\, K1^3\, K2\, Vth2\, Vth1\big)^{1/2}\Big)$$

The solution for V_2:

$$V2 = \frac{1}{2}\frac{1}{K1 - K2}\Big(-K1\,Vth4 + K1\,V1 - K1\,Vth1 + K1\,Vdd - K2\,Vth2 - \big(2\,K1^2\,Vdd\,V1 + K1^2\,Vdd^2 + K2^2\,Vth2^2 - 3\,K1^2\,V1^2 - 2\,K1\,Vdd\,K2\,Vth2 - 2\,K2\,Vth2\,K1\,V1 + K1^2\,Vth4^2 + K1^2\,Vth1^2 + 2\,K1\,Vth4\,K2\,Vth2 + 2\,K1\,Vth1\,K2\,Vth2 + 2\,K1^2\,Vth4\,V1 - 2\,K1^2\,Vth4\,Vth1 - 2\,K1^2\,Vth4\,Vdd + 2\,K1^2\,V1\,Vth1 - 2\,K1^2\,Vth1\,Vdd + 4\,K2\,K1\,V1^2 - 4\,K2\,K1\,V1\,Vth1 - 4\,K2\,K1\,V1\,Vth4 + 4\,K2\,K1\,Vth1\,Vth4\big)^{1/2}\Big)$$

The solution for I_c:

$$I_c = K_4[(V_1 - Vth_4)^2 - (V_1 - V_2 - Vth_4)^2] \qquad (4.\ 74)$$

To conclude, we have demostrated how we derive critical current I_c in analytical form using nominal value approximations on the vector field slope at the bifurcation points.

CHAPTER V

NUMERICAL ALGORITHMS AND IMPLEMENTATIONS

In chapter IV, we derived the dynamic noise margin analytically and rigorously for level-one model. In this chapter, we will use numerical implementation to find bifurcation point, $I_{critical}$ and $T_{critical}$ for higher level-models, in which the function f and g can be higher level models represented by differential equations or program routines that generate a vector field mapping based on laboratory data. We will extend the conceptual use of Jacobian matrix at bifurcation point to solve bifurcation point and I_c except that now we only handle conceptual differential equations, which may not be differentiable in practice. On the other hand, we do have numerical values around a particular point which we can use the data to estimate the Jacobian entries.

5.1. NUMERICAL IMPLEMENTATION ON CRITICAL CURRENT

5.1.1. Algorithm for Getting $I_{critical}$

Similar as before, the calculation would set $f(V_1,V_2,I_c)=0$ and $g(V_1,V_2,I_c)=0$ and determinant of Jacobian matrix to zero for solving three unknowns, $V_1,V_2,$ and I_c. Numerically, the Newton-Raphson iteration would be good enough for convergence.

$$\begin{cases} \dot{V}_1 = f(V_1,V_2,I_c) = 0 \\ \dot{V}_2 = g(V_1,V_2,I_c) = 0 \\ h(V_1,V_2,I_c) = f_{v1}\cdot g_{v2} - f_{v2}\cdot g_{v1} = 0 \end{cases} \tag{5. 1}$$

$$\begin{bmatrix} v_1^{(n+1)} \\ v_2^{(n+1)} \\ I_c^{(n+1)} \end{bmatrix} = \begin{bmatrix} v_1^{(n)} \\ v_2^{(n)} \\ I_c^{(n)} \end{bmatrix} - \begin{bmatrix} \frac{\partial f}{\partial v_1} & \frac{\partial f}{\partial v_2} & \frac{\partial f}{\partial I_c} \\ \frac{\partial g}{\partial v_1} & \frac{\partial g}{\partial v_2} & \frac{\partial g}{\partial I_c} \\ \frac{\partial h}{\partial v_1} & \frac{\partial h}{\partial v_2} & \frac{\partial h}{\partial I_c} \end{bmatrix}^{-1}_{\substack{v_1^{(n)} \\ v_2^{(n)}}} \cdot \begin{bmatrix} f(v_1^{(n)}, v_2^{(n)}, I_c^{(n)}) \\ g(v_1^{(n)}, v_2^{(n)}, I_c^{(n)}) \\ h(v_1^{(n)}, v_2^{(n)}, I_c^{(n)}) \end{bmatrix} \tag{5. 2}$$

Fig. 5-1 is the graphical representation of the iteration method. The Newton Raphson iteration formula is presented above in (5.3). The 3x3 Jacobian matrix involves first partial derivatives, second partial derivatives and mixed partial derivatives. We will apply the center difference formula for each partial derivative. Mathematically, center difference has higher order of accuracy over forward and backward differentiation. The first estimated partial derivatives using the numerical values around a point is given as:

$$\frac{\partial f}{\partial v_1} = \frac{f(v_{1o} + \Delta v, v_{2o}) - f(v_{1o} - \Delta v, v_{2o})}{2 \cdot \Delta v} \tag{5. 3}$$

$$\frac{\partial f}{\partial v_2} = \frac{f(v_{1o}, v_{2o} + \Delta v) - f(v_{1o}, v_{2o} - \Delta v)}{2 \cdot \Delta v} \tag{5. 4}$$

The second partial derivatives:

$$\frac{\partial^2 f}{\partial {v_1}^2} = \frac{f(v_{1o} + \Delta v, v_{2o}) - 2f(v_{1o}, v_{2o}) + f(v_{1o} - \Delta v, v_{2o})}{\Delta v^2} \tag{5. 5}$$

$$\frac{\partial^2 f}{\partial {v_2}^2} = \frac{f(v_{1o}, v_{2o} + \Delta v) - 2f(v_{1o}, v_{2o}) + f(v_{1o}, v_{2o} - \Delta v)}{\Delta v^2} \tag{5. 6}$$

And mixed partial derivatives:

$$\frac{\partial^2 f}{\partial v_1 \partial v_2} = \frac{\begin{array}{c} f(v_{1o} + \Delta v, v_{2o} + \Delta v) - f(v_{1o} + \Delta v, v_{2o} - \Delta v) \\ - f(v_{1o} - \Delta v, v_{2o} + \Delta v) + f(v_{1o} - \Delta v, v_{2o} - \Delta v) \end{array}}{4\Delta v^2} \tag{5. 7}$$

$$\frac{\partial^2 f}{\partial v_1 \partial v_2} = \frac{\partial^2 f}{\partial v_2 \partial v_1} \qquad (5.\ 8)$$

Here is the proposed Newton-Raphson iteration algorithms:

1. Start from an initial guess *v(o)*. And pick a reasonable Δv for differentiation.
2. Find the estimated Jacobian matrix entries using the above numerical values.
3. Iterate Newton's method: $v^{(n+1)} = v^{(n)} - J(v^{(n)})^{-1} \cdot func(v^{(n)})$
4. Take the difference against previous iteration (it would be initial guess for the first time), and take norm (this program uses two-norm). Then compare to a desire epsilon. $\| v^{(n)} - v^{(n-1)} \|_2 < \varepsilon$
5. If the norm is less than epsilon, the iteration terminated.
6. Otherwise, update the result and continue from step 2.

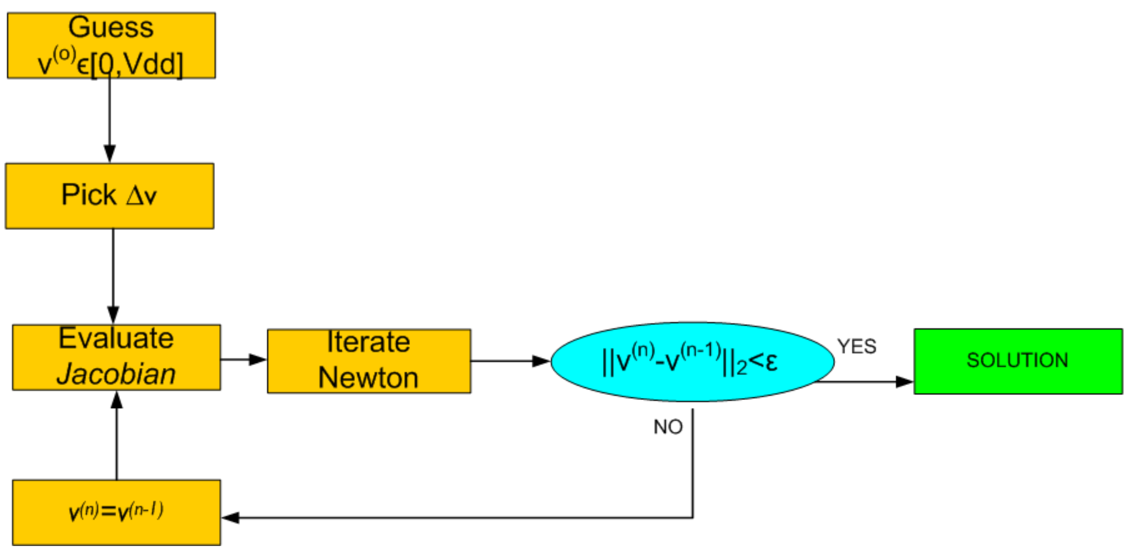
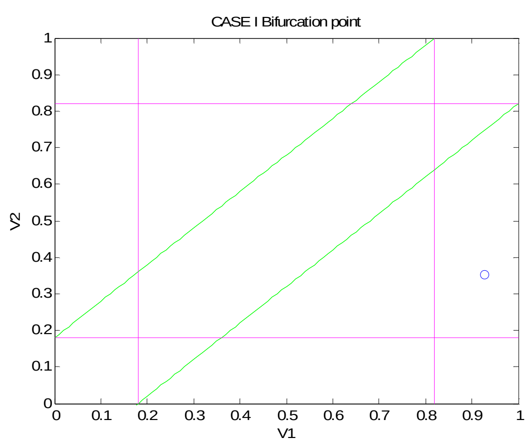


Fig. 5-1 Newton Raphson's Algorithm Flow-Chart

5.1.2. Test Result of $I_{critical}$ Implementation on L-1 Model

CASE I:

Here we will verify the above algorithm by programming in Matlab using L-1 model. Here, we use numerical data to estimate the Jacobian entries on all regions. The output of the code is based on 65nm level one model parameters. The code is posted in APPENDIX B. The program uses roughly 5 iterations to accuracy $1\text{x}10^{-6}$ on all cases. For case I, the current source attaches at the right inverter. On standard 65nm technology parameters, pick initial state (V_1 V_2) to be (V_{dd},0), and let I_c iterate from zero. The result gives I_c=$4.96\text{x}10^{-4}$ (A) and bifurcation point locate at (0.9266, 0.3533) as shown in Fig. 5-2. The I_c indicates positive $I_{critial}$. For a SRAM has initial state (V_{dd},0), an injected current $4.96\text{x}10^{-4}$ (A) to V_2 node will flip the state of SRAM as time goes to infinity.




```
k =

     5


V =

   0.92657731779815
   0.35331092550619
   0.00049603828393
```

Fig. 5-2 The Bifurcation Location Plot and The Numerical Result on Positive I_c for Case I.

Try different initial guess in Case I. Pick (V_1 V_2)=(0,V_{dd}) and still let I_c iterate from zero. The iteration result gives Ic=-2.878x10^{-4} (A) as shown in Fig. 5-3. Since V_2 node is originally V_{dd}, injecting current to V_2 node would not make state to flip. It has to be extracting current from V_2 node to have state flipping. Thus, the I_c should be negative as expected. Compared to the previous solution, the I_c magnitude for injecting and extracting current to make state flip are different. Since the magnitude of negative convergence I_c is smaller than positive convergence I_c, the noise margin for Case I is said to be the magnitude of negative I_c.

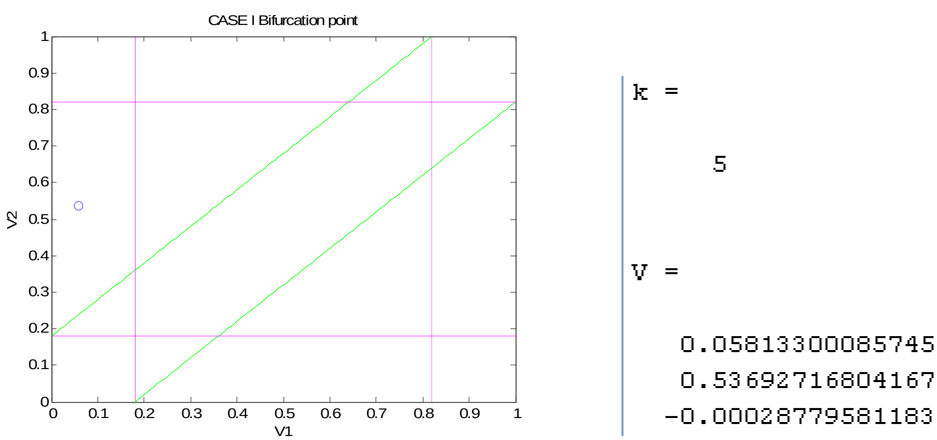


Fig. 5-3 The Bifurcation Location Plot and The Numerical Result on Negative I_c.

CASE II:

Using 65nm parameters, for the case that current source attaches at the left inverter, initial state of (0, V_{dd}) gets state flip at I_c=4.96e-4(A) when time run to infinity, and bifurcation point is at (0.3533, 0.9266) as shown in Fig. 5-4. For initial condition of

(V_{dd},0), the iteration result is I_c=-2.878x10^{-4} (A) and bifurcation is at (0.537,0.058) as shown in Fig. 5-5. Notice the results are similar to Case I. The positive I_c are the same in Case I and Case II, and their bifurcation points are inverted. The reason is because this SRAM parameters are balanced, it means the device parameters for left inverter are the same as the right inverter. So the current source attach from the left or the right, their resistance and capacitance look into the circuitry are the same from the attached source.

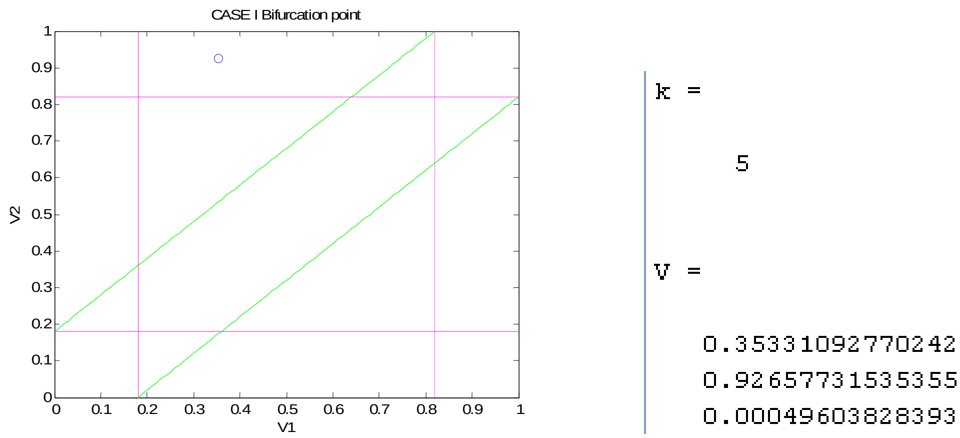


Fig. 5-4 The Bifurcation Location Plot and The Numerical Result on Positive I_c.

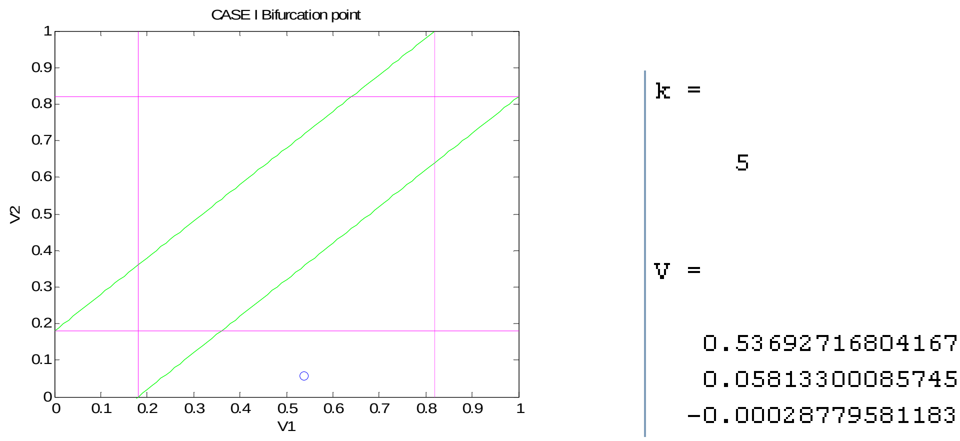


Fig. 5-5 The Bifurcation Location Plot and The Numerical Result on Negative I_c.

In summary, Case I and Case II both have two solutions on their Newton's iteration. One is positive I_c and one is negative I_c depending on the initial guess. The noise margin is the I_c that has mininum in magnitude. For example, in case I, one is -2.878x10^{-4} (A) and the other is 4.96x10^{-4} (A), and the one in smaller magnitude is considered to be the noise margin. In reality, when the noise comes into SRAM device, we do not know whether the noise acts as extracting current or injecting current, so the noise margin should pick the least in magnitude that will cause state flipping.

5.2. NUMERICAL IMPLEMENTATION ON CRITICAL TIME

5.2.1. Algorithm to Get $T_{critical}$

Assume the injected current reaches a constant amplitude of $I_{critical}$ or above, the state of SRAM will cross the stability boundary (the separatrix at I_{noise}=0) and flip state after a duration. This amount of time is called the critcal time, denoted by $T_{critical}$. When the noise disappear before $T_{critical}$, the state has not cross the boundary yet; and SRAM would not result state flip. If the noise has amplitude of $I_{critical}$ and longer duration than

$T_{critical}$, the state will cross the stability boundary and result state flipping. Thus, $T_{critial}$ is said to be the time it takes from initial state onto the separatrix once $I_{critical}$ is injected. Here is a proposed algorithm to find $T_{critical}$.

1. Trace out Separatrix using the method in Chapter III.
2. Take some samples on the separatrix.
3. Using curve fitting or interpolation to a polynomial. (Cubic least square is recommended).
4. Integrated the system with a given initial.
5. Check whether the state is above the curve fit polynomial.
6. If not, keep integrating the system.

5.2.2. Result of $T_{critical}$ Implementation On L-1 Model

Here we will demonstrate the above $T_{critical}$ algorithm by programming in Matlab. From section 5.1.2, an initial state (V_{dd},0) has I_c=4.96e-4 (A) when noise injecting to V_2 node. The state will reach the separatrix in 17.5ns as shown in Fig. 5-6. Furthermore, Table 5 shows the T_c for few sample set of different perturbed parameters. Each parameter is varying plus or minus 50%, and Table 6 totally generate 28 different combinations. By glance, the T_c doesn't seem to follow any patterns.

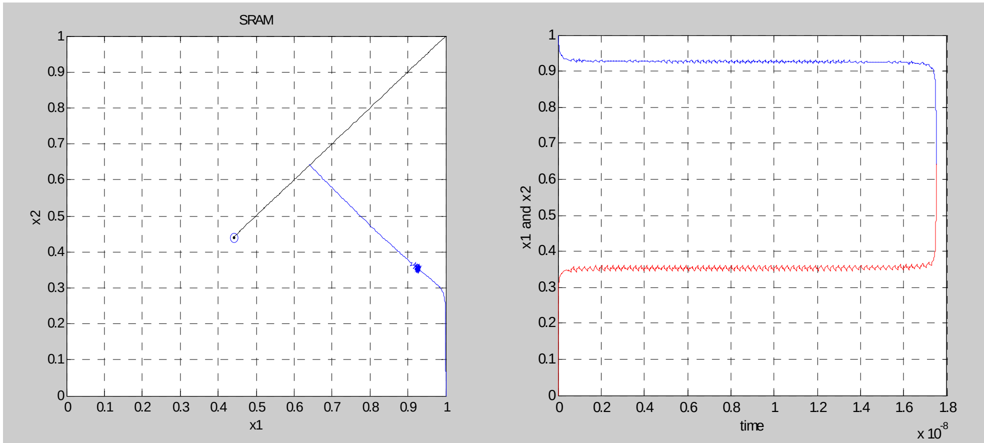


Fig. 5-6 (LEFT) Trajectory for an Initial State (V_{dd},0) Running for 17.5ns; and Its (RIGHT) Timing Diagram.

Lastly, The accuracy of simulated $I_{critical}$ and $T_{critical}$ can be varified by generating a square pulse that has amplitude of I_c and duration of T_c as noise injection. If the simulated I_c and T_c are exact, the state should just reach the separatrix. However, it can hardly happen in actual implementation due to truncation error from digital implemenation. In most of caes, the state will cross over the separatrix or come back to original equilibrium even with quite accurate I_c and T_c. Accordinly, we propose a way to verify I_c and T_c. Focus on I_c first. During transient integration, the accurate I_c will flip state when time goes to infinity; and it will not flip state if decreased by a tiny decrement. Thus, one can set a prefered decrement accuracy. If the I_c without the decrement would flip state but would not flip with the decrement, that I_c can say to be accurate to the decrement value. The same idea for $T_{critical}$. If the T_c with the decrement would not flip state but it would flip without the decrement, that T_c is accurate enough.

Table 6 Few Sample Data for T_c on Different Set of Perturbed Parameters.

Vth1	Vth2	Vth4	K1	K2	K4	V1	V2	Ic (A)	Tc (1e-7 sec)
1	1	1	1	1	1	0.926577318	0.353310926	0.000496038	0.187048222
			1-50%	1+50%		0.928095915	0.286515705	0.000426904	0.140916556
			1-50%	1+50%	1+50%	0.928095908	0.286515709	0.000640356	0.179296111
			1-50%	1+50%	1-50%	0.928095915	0.286515705	0.000213452	0.321394444
			1-50%	1-50%	1+50%	0.926577327	0.353310924	0.000744057	0.202686333
			1-50%	1-50%	1-50%	0.926577317	0.353310926	0.000248019	0.359201111
			1+50%	1-50%	1+50%	0.936597413	0.433420446	0.000864662	0.244865778
			1+50%	1+50%	1+50%	0.926577327	0.353310923	0.000744057	0.158330778
1+50%	1+50%					0.954025224	0.388107646	0.000554578	0.153990889
1+50%	1-50%					0.897219523	0.285952196	0.000404513	0.210583556
1-50%	1-50%					0.894925187	0.321479575	0.000438886	0.336603667
1+50%	1+50%	1+50%				0.963359379	0.377873687	0.000469552	0.194290667
1-50%	1-50%	1-50%				0.880391885	0.332678207	0.000511433	0.273377667
1-50%	1-50%	1-50%	1-50%	1-50%	1-50%	0.880391893	0.332678201	0.000255716	0.810727778
1-50%	1-50%	1-50%	1+50%	1+50%	1+50%	0.880391887	0.332678206	0.000767149	0.231742333
1-50%	1+50%	1-50%	1+50%	1-50%	1+50%	0.943891157	0.513058522	0.001132501	0.259550856
1+50%	1-50%	1+50%	1-50%	1+50%	1-50%	0.903597933	0.210090094	0.000136775	0.389234444
1+50%	1-50%	1-50%	1-50%	1+50%	1+50%	0.874796316	0.222858931	0.000554518	0.177380889
1+50%	1-50%	1+50%	1-50%	1+50%	1+50%	0.903597932	0.210090094	0.000410325	0.180155787
1+50%	1-50%	1+50%	1-50%	1+50%	1-50%	0.903597933	0.210090094	0.000136775	0.389234444
1+50%	1-50%	1-50%	1-50%	1+50%	1-50%	0.874796359	0.222858914	0.000184839	0.177785556
1-50%	1+50%	1+50%	1+50%	1-50%	1+50%	0.963932523	0.476749238	0.000802542	0.230574111
1-50%	1+50%	1-50%	1+50%	1-50%	1+50%	0.943891157	0.513058522	0.001132501	0.259550856
1+50%	1+50%	1-50%	1+50%	1+50%	1-50%	0.944860888	0.396654172	0.000320761	0.635293222
1+50%	1+50%	1-50%	1-50%	1+50%	1-50%	0.952737967	0.343720662	0.000292496	0.251216667
1+50%	1+50%	1-50%	1-50%	1-50%	1-50%	0.944860889	0.396654171	0.000320761	0.288166556
1+50%	1-50%	1-50%	1-50%	1-50%	1-50%	0.884250423	0.294050017	0.000234417	0.610974444
1-50%	1+50%	1-50%	1-50%	1-50%	1-50%	0.938440866	0.431867364	0.000336456	0.375917889

CHAPTER VI

CONCLUSIONS AND FUTURE WORKS

6.1. CONCLUSIONS AND CONTRIBUTIONS

This work defines a new SRAM dynamic noise margin based on rigorous nonlinear system theory. This new explored stability margin provides more physical characterization on SRAM with noise tolerance property, and fully preserves the dynamic cell behavior. For level-one transistor model, this work derives an analytical form for the stability margin; and it also provides an "implementable" basis using numerical algorithms to calculate the stability margin for higher level transistor model.

6.2. RECOMMENDED FUTURE WORKS

The stability margin analysis on memory device such as SRAM can be quite involved. The proposed extended work can be the following:

1. *Better approximation on analytical solution for stability margin.*

   This work uses approximation on the slope to derive an analytic expression for stability margin. The average error on I_c is less 10%; however, a more exact approximation is desirable.

2. *Useful design insight.*

   A useful design insight gives designers better information on how to adjust/design circuit parameters to achieve a desired performance criterion. The design insight is one of the main reasons to investigate analytical form on stability margin. It requires more research and more analysis to correlate the analytical insights with higher level models to verify and clarify our insights.

3. *Analytical solution and analysis for more complicated models*.

   This work starts from a simple level-one model. However, level one model is too ideal. Typical industry practices are using at least level 50 or higher models. For higher spice level, the complexity of transistor modeling increases. It will be great if we can extend our analytical solution for higher level models to include more complicated design parameters.

REFERENCES

[1] M. Khellah, Y. Ye, N. S. Kim, D. Somasekhar, G. Pandya, A. Farhang, K. Zhang, C. Webb and V. De, "Wordline & bitline pulsing schemes for improving SRAM cell stability in low-Vcc 65nm CMOS designs," *Symp. on VLSI Circuits*, pp. 9-10, June 2006.

[2] H. Pilo, J. Barwin, G. Braceras, C. Browning, S. Burns, J. Gabric, S. Lamphier, M. Miller, A. Roberts, F. Towler, "An SRAM design in 65nm and 45nm technology nodes featuring read and write-assist circuits to expand operating voltage," *Symp. on VLSI Circuits*, June 2006.

[3] M. Khellah, N. S. Kim, J. Howard, G. Ruhl, M. Sunna, Y. Ye, J. Tschanz, D. Somasekhar, N. Borkar, F. Hamzaoglu, G. Pandya, A. Farhang, K. Zhang, V. De, "A 4.2GHz 0.3mm^2 256kb dual-Vcc SRAM building block in 65nm CMOS", *IEEE Int. Solid-State Circuits Conf.*, pp 624-625, Feb. 2006.

[4] M. Yabuuchi, K. Nii, Y. Tsukamoto, S. Ohbayashi, S. Imaoka, H. Makino, Y. Yamagami, S. Ishikura, T. Terano, K. Hashimoto, A. Sebe, G. Okazaki, K. Satomi, H. Akamatsu, H. Shinohara, "A 45nm low-standby-power embedded SRAM with improved immunity against process and temperature variations," *IEEE Int. Solid-State Circuits Conf.*, pp. 326-327, Feb. 2007.

[5] K. Takeda, H. Ikeda, Y. Hagihara, M. Nomura and H. Kobatake, "Redefinition of write margin for next-generation SRAM and write-margin monitoring circuit", *IEEE Int. Solid-State Circuits Conf.*, pp. 2602-2603, Feb. 2006.

[6] B. H. Calhoun and A. P. Chandrakasan, "A 256kb sub-threshold SRAM in 65nm CMOS," *IEEE Int. Solid-State Circuits Conf.*, pp. 628-629, Feb. 2006.

[7] T. Kim, J. Liu, J. Keane and C. H. Kim, "A high-density subthreshold SRAM with data-independent bitline leakage and virtual ground replica scheme", *IEEE Int. Solid-State Circuits Conf.*, pp. 330-331, Feb. 2007.

[8] N. Verma, A. P. Chandrakasan, "A 65nm 8T Sub-V_t SRAM employing sense-amplifier redundancy", *IEEE Int. Solid-State Circuits Conf.*, pp. 328-329, Feb.

2007.

[9] B. Zhai, D. Blaauw, D. Sylvester and S. Hanson, “A sub-200mv 6T SRAM in 0.13µm CMOS”, *IEEE Int. Solid-State Circuits Conf.*, pp. 332-333, Feb. 2007.

[10] International Technology Roadmap for Semiconductors, 2005 Edition, http://www.itrs.net/home.html. (Accesed on March 2007).

[11] K. Chakraborty and P. Mazumder, *Fault-Tolerance and Reliability Techniques for High-Density Random-Access Memories*, Prentice Hall PTR, Englewood Cliffs, NJ, 2002.

[12] A. J. Bhavnagarwala, X. Tang and J. D. Meindl, “The impact of intrinsic device fluctuations on CMOS SRAM cell stability”, *IEEE J. of Solid-State Circuits*, vol. 36, no. 4, pp. 658-665, Apr. 2001.

[13] E. Grossar, M. Stucchi, K. Maex, and W. Dehaene, “Read stability and write-ability analysis of SRAM cells of nanometer technologies”, *IEEE J. of Solid-State Circuits*, vol. 41, no.11, pp. 2577-2588, Nov. 2006.

[14] R.V. Joshi, S. Mukhopadhyay, D.W. Plass, Y.H. Chan, C. Chuang, A. Devgan, “Variability analysis for sub-100 nm PD/SOI CMOS SRAM cell”, *European Solid-State Circuits Conf.*, pp. 211-214, Sept. 2004.

[15] R. Heald and P. Wang, “Variability in sub-100nm SRAM design,” *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 347-352, Nov. 2004.

[16] S. V. Kumar, C. H. Kim and S. S. Sapatnekar, “Impact of NBTI on SRAM read stability and design for reliability,” *IEEE Int. Symp. on Quality Electronic Design.*, pp. 27-29, Mar. 2006.

[17] K. Agarwal and S. Nassif, “Statistical analysis of SRAM cell stability,” *IEEE/ACM Design Automation Conf.*, pp. 57-62, July, 2006.

[18] B. Zhang, A. Arapostathis, S. Nassif and M. Orshansky, “Analytical modeling of SRAM dynamic stability,” *IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 315-322, Nov. 2006.

[19] S. Mukhopadhyay, H. Mahmoodi, K. Roy, “Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS”,

IEEE Trans. on Computer-Aided Design, vol. 24, no. 12, pp. 1859-1880, Dec. 2005.

[20] K. Takeda, Y. Hagihara, Y. Aimoto, M. Nomura, Y. Nakazawa, T. Ishii, H. Kobatake., “A read-static-noise-margin-free SRAM cell for low-VDD and high-speed applications”, *IEEE J. of Solid-State Circuits*, vol. 41, no. 1, pp. 113-121, Jan. 2006.

[21] L. Chang, D.M. Fried, J. Hergenrother, J.W. Sleight, R.H. Dennard, R.K. Montoye, L. Sekaric, S.J. McNab, A.W. Topol, C.D. Adams, K.W. Guarini, W. Haensch, “Stable SRAM cell design for the 32 nm node and beyond,” *Symp. on VLSI Technology*, June 2005.

[22] P.C. Murley and G. R. Srinivasan, “Soft-error Monte Carlo modeling, program, SEMM”, *IBM J. Res. Develop*. vol. 40, no. 1, pp. 109-118, Jan. 1996.

[23] L. B. Freeman, “Critical charge calculations for a bipolar array cell”, *IBM J. Res. Develop*. vol. 40, no. 1, pp. 119-129, Jan. 1996.

[24] H. H. Chen and J. S. Neely, “Interconnect and circuit modeling techniques for full-chip power supply noise analysis,” *IEEE Trans. Components, Packaging, and Manufacturing Technology*-Part B, 1998, pp. 209-215.

[25] R. Gharpurey and R. G. Meyer, “Modeling and analysis of substrate coupling in IC’s,” *IEEE Custom-Integrated Circuit Conf.*, 1995, pp. 125–128.

[26] P.E Dodd and L. W. Massengill, “Basic mechanisms and modeling of single-even upset in digital microelectronics,” *IEEE Trans. Nucl. Sci.*, 2003, pp. 583-602.

[27] M. Khellah, D. Khalil, D. Somasekhar, Y. Ismail, T. Karnik and V. De, “Effect of power supply noise on SRAM dynamic stability,” *Symp. on VLSI Circuits*, pp. 76-77, June 2007.

[28] B. H. Calhoun and A. P. Chandrakasan, “Static noise margin variation for sub-threshold SRAM in 65-nm CMOS”, *IEEE J. of Solid-State Circuits*, vol. 41, no. 7, pp. 1673-1679, July 2006.

[29] J. Lohstroh, E. Seevinck and J. D. Groot, “Worst-case static noise margin criteria

for logic circuits and their mathematical equivalence”, *IEEE J. of Solid-State Circuits*, vol. sc-18, no. 6, pp. 803-806, Dec. 1983.

[30] E. Seevinck, F. J. List and J. Lohstroh, “Static-noise margin analysis of MOS SRAM cells”, *IEEE J. of Solid-State Circuits*, vol. sc-22, no. 5, pp. 748-754, Oct. 1987.

[31] B. Zhang, A. Arapostathis, S. Nassif, and M. Orshansky, “Analytical Modeling of SRAM Dynamic Stability,” *ICCAD’06, San Jose, CA*, Nov 5-9, 2006

[32] Garng M. Huang, Wei Dong, Yenpo Ho, and Peng Li, “Tracing SRAM separatrix for dynamic noise margin analysis under device mismatch,” in *Proc. of IEEE Int. Behavioral Modeling and Simulation Conf.*, pp. 6-10, September 2007.

[33] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields*. New York: Springer-Verlag, 1983.

[34] A.A. Andronov, E.A. Leontovich, I.I. Gordon, A.G. Maier, "Theory of bifurcations of dynamical systems on a plane", Israel Program Sci. Transl. (1971) (In Russian)

[35] Yu. Kuznetsov, *Elements of Applied Bifurcation Theory*, New York: Springer, 1995.

[36] A. Vanderbauwhede, "Centre manifolds, normal forms and elementary bifurcations" *Dynamics Reported* , pp. 89–169, 1989.

[37] Bifurcation Theory, March 2008, http://en.wikipedia.org/wiki/Global_bifurcation , (Accessed on March 2008)

[38] Yu. Kuznetsov,“Saddle-node bifurcation”, http://eom.springer.de/s/s110010.htm, (Accessed on March 2008)

[39] *Microelectronic Circuits*, 5th ed. A. Sedra and K. Smith, New York: Oxford University Press, 2004.

[40] H. Pilo, C. Barwin, G. Braceras C. Browning, S. Lamphier and F. Towler, “An SRAM design in 65-nm technology node featuring read and write-assist circuits to expand operating voltage”, *IEEE J. of Solid-State Circuits*, vol. 42, no. 4, pp. 813-819, Apr. 2007.

[41] H.Khalil, *Nonlinear Systems*. 3rd ed. Upper Saddle River, NJ: Prentice-Hall., 2002.

[42] J. Carr, *Applications of Center Manifold Theory*. New York: Springer Verlag, vol. 35, 1981.

[43] A. Kelley, "The stable, center stable, center, center unstable and unstable manifolds" *J. Diff. Eq.*, vol. 3, pp. 546–570, 1967.

[44] J. Zaborszky, G. Huang, B. Zheng and T.C. Leung, “On the phase portrait of a class of large nonlinear dynamic systems such as the power system,” *IEEE Transactions on Automatic Control*, vol. 33, no. 1, pp. 4-15, January 1988.

[45] P. M. Carter and R. Wilkins, “Influences of soft error rates in static RAM’s”, *IEEE J. of Solid-State Circuits*, vol. sc-22, no. 3, pp. 430-436, June 1987.

[46] Stewart, Ian, *Galois Theory*, 3rd ed. Boca Raton, FL: Chapman & Hall/CRC Mathematics, 2004.

[47] David Gay, “Modifying singular values: Solution to systems of nonlinear equations having a possibly singular Jacobian matrix”, *Mathematics of Computation*, vol. 31, no. 140, pp 962-973, October 1977.

APPENDIX A

The 65nm System Parameter Used Throughout This Paper

	NMOS	PMOS
μ_0	0.06 $m^2/V/S$	0.014 $m^2/V/S$
t_{ox}	1.7 nm	1.7 nm
V_{th}	0.18 V	0.18 V
L_{eff}	30nm	30nm
Width	60 nm	125.5 nm
K	1.23×10^{-3}	5.86×10^{-4}
	$C_1 = C_2 = 10.2$ fF	
	$V_{dd} = 1.0$ V	

The Results for Analytical Soluton of V_1 for Section 4.7:

Region 9A:

The first solution:

V1=(-2*K1*Vth4*K4^2*K2-2*K1*Vth1*K4^2*K2-
K1*K4^2*K2*Vth2+K1^2*Vth1*K4^2+K1^2*Vth4*K4^2+K1^2*K4^2*Vdd)/(3*K1^2*K4^2-
4*K1*K4^2*K2)+1/12/(3*K1^2*K4^2-
4*K1*K4^2*K2)*((9*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2*((288*K1*Vth1*K4^2*K2*Vth2-72*K1*K4^2*K2*Vth2^2-
288*K1^2*Vth1*K4^2*Vth4-144*K1^2*Vth4*K4^2*Vdd-
144*K1*Vdd^2*K4^2*K2+144*K1*Vth4^2*K4^2*K2+144*K1^2*Vth1^2*K4^2-
216*K1*K4*K2*In+576*K1*Vth1*K4^2*K2*Vth4+72*K1^2*K4^2*Vdd^2-
432*K1^2*Vth1*K4^2*Vdd+144*In*K4*K1^2+144*K2^2*K4^2*Vth2^2+144*K1*Vth4*K4^2*K2*Vth2+288*K1*Vdd*Vth1*K
4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)-
(2304*K1*Vdd*K4*K2*In*Vth2+1152*K1*Vdd*Vth4*K4*K2*In+1152*K1*Vdd*Vth4*K4^2*K2*Vth2^2+1152*K1*Vdd^2*K4
^2*K2*Vth4^2+1152*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
576*K1*Vdd^2*K4*K2*In+576*K1*Vdd^2*K4^2*K2*Vth2^2+2304*K1^2*Vth1*K4^2*Vdd^2*Vth4-
2304*K1*Vdd*Vth1*K4^2*K2*Vth4^2-2304*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+1152*K1*Vdd*Vth1*K4*K2*In-
1152*K1*Vdd*Vth1*K4^2*K2*Vth2^2-1152*K1^2*Vdd^2*Vth1^2*K4^2+1152*K1^2*Vdd^3*Vth1*K4^2-
1152*In*K4*K1^2*Vdd^2-288*K2^2*In^2-1152*K1^2*Vdd^3*Vth4*K4^2-1152*K2^2*Vth4^2*K4^2*Vth2^2-
1152*K2^2*Vth4*Vth2^3*K4^2-576*K2^2*In*Vth2^2*K4-288*K1^2*Vdd^4*K4^2-288*K2^2*Vth2^4*K4^2-
1152*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+108*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(864*K1*Vdd*K4*K2*In*Vth2+432*K1*Vdd*Vth4*K4*K2*In+432*K1*Vdd*Vth4*K4^2*K2*Vth2^2+432*
K1*Vdd^2*K4^2*K2*Vth4^2+432*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
216*K1*Vdd^2*K4*K2*In+216*K1*Vdd^2*K4^2*K2*Vth2^2+864*K1^2*Vth1*K4^2*Vdd^2*Vth4-
864*K1*Vdd*Vth1*K4^2*K2*Vth4^2-864*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+432*K1*Vdd*Vth1*K4*K2*In-
432*K1*Vdd*Vth1*K4^2*K2*Vth2^2-432*K1^2*Vdd^2*Vth1^2*K4^2+432*K1^2*Vdd^3*Vth1*K4^2-
432*In*K4*K1^2*Vdd^2-108*K2^2*In^2-432*K1^2*Vdd^3*Vth4*K4^2-432*K2^2*Vth4^2*K4^2*Vth2^2-
432*K2^2*Vth4*Vth2^3*K4^2-216*K2^2*In*Vth2^2*K4-108*K1^2*Vdd^4*K4^2-108*K2^2*Vth2^4*K4^2-
432*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2+8*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3+12*3^(1/2)*(-(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+128*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth
2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-

2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+6*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K
1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(1536*K1*Vth4*K4^2*K2+1536*K1*Vth1*K4^2*K2+768*K1*K4^2*K2*Vth2-768*K1^2*Vth1*K4^2-
768*K1^2*Vth4*K4^2-
768*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3*(3*K1^2*K4^2-4*K1*K4^2*K2)-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-

K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)-
(1152*K1*Vdd*K4*K2*In*Vth2+576*K1*Vdd*Vth4*K4*K2*In+576*K1*Vdd*Vth4*K4^2*K2*Vth2^2+576*K1*Vdd^2*K4^2*
K2*Vth4^2+576*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
288*K1*Vdd^2*K4*K2*In+288*K1*Vdd^2*K4^2*K2*Vth2^2+1152*K1^2*Vth1*K4^2*Vdd^2*Vth4-
1152*K1*Vdd*Vth1*K4^2*K2*Vth4^2-1152*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+576*K1*Vdd*Vth1*K4*K2*In-
576*K1*Vdd*Vth1*K4^2*K2*Vth2^2-576*K1^2*Vdd^2*Vth1^2*K4^2+576*K1^2*Vdd^3*Vth1*K4^2-
576*In*K4*K1^2*Vdd^2-144*K2^2*In^2-576*K1^2*Vdd^3*Vth4*K4^2-576*K2^2*Vth4^2*K4^2*Vth2^2-
576*K2^2*Vth4*Vth2^3*K4^2-288*K2^2*In*Vth2^2*K4-144*K1^2*Vdd^4*K4^2-144*K2^2*Vth2^4*K4^2-
576*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2-
144*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*V
th4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2-
(128*K1*Vdd*K4*K2*In*Vth2+64*K1*Vdd*Vth4*K4*K2*In+64*K1*Vdd*Vth4*K4^2*K2*Vth2^2+64*K1*Vdd^2*K4^2*K2*
Vth4^2+64*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
32*K1*Vdd^2*K4*K2*In+32*K1*Vdd^2*K4^2*K2*Vth2^2+128*K1^2*Vth1*K4^2*Vdd^2*Vth4-
128*K1*Vdd*Vth1*K4^2*K2*Vth4^2-128*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+64*K1*Vdd*Vth1*K4*K2*In-
64*K1*Vdd*Vth1*K4^2*K2*Vth2^2-64*K1^2*Vdd^2*Vth1^2*K4^2+64*K1^2*Vdd^3*Vth1*K4^2-64*In*K4*K1^2*Vdd^2-
16*K2^2*In^2-64*K1^2*Vdd^3*Vth4*K4^2-64*K2^2*Vth4^2*K4^2*Vth2^2-64*K2^2*Vth4*Vth2^3*K4^2-
32*K2^2*In*Vth2^2*K4-16*K1^2*Vdd^4*K4^2-16*K2^2*Vth2^4*K4^2-64*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-
4*K1*K4^2*K2)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^4+4*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3+(32*K1*Vdd*K4*K2*In*Vth2+16*K1*Vdd*Vth4*K4*K2*In+16*K1*Vdd*Vth4*K4^2*K2*Vth2
^2+16*K1*Vdd^2*K4^2*K2*Vth4^2+16*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
8*K1*Vdd^2*K4*K2*In+8*K1*Vdd^2*K4^2*K2*Vth2^2+32*K1^2*Vth1*K4^2*Vdd^2*Vth4-
32*K1*Vdd*Vth1*K4^2*K2*Vth4^2-32*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+16*K1*Vdd*Vth1*K4*K2*In-
16*K1*Vdd*Vth1*K4^2*K2*Vth2^2-16*K1^2*Vdd^2*Vth1^2*K4^2+16*K1^2*Vdd^3*Vth1*K4^2-16*In*K4*K1^2*Vdd^2-
4*K2^2*In^2-16*K1^2*Vdd^3*Vth4*K4^2-16*K2^2*Vth4^2*K4^2*Vth2^2-16*K2^2*Vth4*Vth2^3*K4^2-
8*K2^2*In*Vth2^2*K4-4*K1^2*Vdd^4*K4^2-4*K2^2*Vth2^4*K4^2-
16*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3-

256*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^3*(3*K1^2*K4^2-
4*K1*K4^2*K2)^3+27*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^4*(3*K1^2*K4^2-
4*K1*K4^2*K2)^2+27*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-
4*K2^2*Vth4*Vth2*K4*In)^2*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^4+4*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3+(640*K1*Vth4*K4^2*K2+640*K1*Vth1*K4^2*K2+320*K1*K4^2*K2*Vth2-320*K1^2*Vth1*K4^2-
320*K1^2*Vth4*K4^2-
320*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-4*K1*K4^2*K2)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2)^(1/2))^(1/3)-(-
192*K1*Vth1*K4^2*K2*Vth2+48*K1*K4^2*K2*Vth2^2+192*K1^2*Vth1*K4^2*Vth4+96*K1^2*Vth4*K4^2*Vdd+96*K1*Vdd^2*K4^2*K2-96*K1*Vth4^2*K4^2*K2-96*K1^2*Vth1^2*K4^2+144*K1*K4*K2*In-384*K1*Vth1*K4^2*K2*Vth4-
48*K1^2*K4^2*Vdd^2+288*K1^2*Vth1*K4^2*Vdd-96*In*K4*K1^2-96*K2^2*K4^2*Vth2^2-96*K1*Vth4*K4^2*K2*Vth2-
192*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-4*K1*K4^2*K2)*((288*K1*Vth1*K4^2*K2*Vth2-72*K1*K4^2*K2*Vth2^2-
288*K1^2*Vth1*K4^2*Vth4-144*K1^2*Vth4*K4^2*Vdd-
144*K1*Vdd^2*K4^2*K2+144*K1*Vth4^2*K4^2*K2+144*K1^2*Vth1^2*K4^2-
216*K1*K4*K2*In+576*K1*Vth1*K4^2*K2*Vth4+72*K1^2*K4^2*Vdd^2-
432*K1^2*Vth1*K4^2*Vdd+144*In*K4*K1^2+144*K2^2*K4^2*Vth2^2+144*K1*Vth4*K4^2*K2*Vth2+288*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)-
(2304*K1*Vdd*K4*K2*In*Vth2+1152*K1*Vdd*Vth4*K4*K2*In+1152*K1*Vdd*Vth4*K4^2*K2*Vth2^2+1152*K1*Vdd^2*K4^2*K2*Vth4^2+1152*K1*Vdd^2*K4^2*K2*Vth4*Vth2-

576*K1*Vdd^2*K4*K2*In+576*K1*Vdd^2*K4^2*K2*Vth2^2+2304*K1^2*Vth1*K4^2*Vdd^2*Vth4-
2304*K1*Vdd*Vth1*K4^2*K2*Vth4^2-2304*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+1152*K1*Vdd*Vth1*K4*K2*In-
1152*K1*Vdd*Vth1*K4^2*K2*Vth2^2-1152*K1^2*Vdd^2*Vth1^2*K4^2+1152*K1^2*Vdd^3*Vth1*K4^2-
1152*In*K4*K1^2*Vdd^2-288*K2^2*In^2-1152*K1^2*Vdd^3*Vth4*K4^2-1152*K2^2*Vth4^2*K4^2*Vth2^2-
1152*K2^2*Vth4*Vth2^3*K4^2-576*K2^2*In*Vth2^2*K4-288*K1^2*Vdd^4*K4^2-288*K2^2*Vth2^4*K4^2-
1152*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+108*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(864*K1*Vdd*K4*K2*In*Vth2+432*K1*Vdd*Vth4*K4*K2*In+432*K1*Vdd*Vth4*K4^2*K2*Vth2^2+432*
K1*Vdd^2*K4^2*K2*Vth4^2+432*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
216*K1*Vdd^2*K4*K2*In+216*K1*Vdd^2*K4^2*K2*Vth2^2+864*K1^2*Vth1*K4^2*Vdd^2*Vth4-
864*K1*Vdd*Vth1*K4^2*K2*Vth4^2-864*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+432*K1*Vdd*Vth1*K4*K2*In-
432*K1*Vdd*Vth1*K4^2*K2*Vth2^2-432*K1^2*Vdd^2*Vth1^2*K4^2+432*K1^2*Vdd^3*Vth1*K4^2-
432*In*K4*K1^2*Vdd^2-108*K2^2*In^2-432*K1^2*Vdd^3*Vth4*K4^2-432*K2^2*Vth4^2*K4^2*Vth2^2-
432*K2^2*Vth4*Vth2^3*K4^2-216*K2^2*In*Vth2^2*K4-108*K1^2*Vdd^4*K4^2-108*K2^2*Vth2^4*K4^2-
432*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2+8*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3+12*3^(1/2)*(-(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+128*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth
2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+6*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K
1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(1536*K1*Vth4*K4^2*K2+1536*K1*Vth1*K4^2*K2+768*K1*K4^2*K2*Vth2-768*K1^2*Vth1*K4^2-

768*K1^2*Vth4*K4^2-
768*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3*(3*K1^2*K4^2-4*K1*K4^2*K2)-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)-
(1152*K1*Vdd*K4*K2*In*Vth2+576*K1*Vdd*Vth4*K4*K2*In+576*K1*Vdd*Vth4*K4^2*K2*Vth2^2+576*K1*Vdd^2*K4^2*
K2*Vth4^2+576*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
288*K1*Vdd^2*K4*K2*In+288*K1*Vdd^2*K4^2*K2*Vth2^2+1152*K1^2*Vth1*K4^2*Vdd^2*Vth4-
1152*K1*Vdd*Vth1*K4^2*K2*Vth4^2-1152*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+576*K1*Vdd*Vth1*K4*K2*In-
576*K1*Vdd*Vth1*K4^2*K2*Vth2^2-576*K1^2*Vdd^2*Vth1^2*K4^2+576*K1^2*Vdd^3*Vth1*K4^2-
576*In*K4*K1^2*Vdd^2-144*K2^2*In^2-576*K1^2*Vdd^3*Vth4*K4^2-576*K2^2*Vth4^2*K4^2*Vth2^2-
576*K2^2*Vth4*Vth2^3*K4^2-288*K2^2*In*Vth2^2*K4-144*K1^2*Vdd^4*K4^2-144*K2^2*Vth2^4*K4^2-
576*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2-
144*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*V
th4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-

K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2-
(128*K1*Vdd*K4*K2*In*Vth2+64*K1*Vdd*Vth4*K4*K2*In+64*K1*Vdd*Vth4*K4^2*K2*Vth2^2+64*K1*Vdd^2*K4^2*K2*
Vth4^2+64*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
32*K1*Vdd^2*K4*K2*In+32*K1*Vdd^2*K4^2*K2*Vth2^2+128*K1^2*Vth1*K4^2*Vd... Output truncated. Text exceeds maximum line length of 25,000 characters for Command Window display.

Second solution:

V1=(-2*K1*Vth4*K4^2*K2-2*K1*Vth1*K4^2*K2-
K1*K4^2*K2*Vth2+K1^2*Vth1*K4^2+K1^2*Vth4*K4^2+K1^2*K4^2*Vdd)/(3*K1^2*K4^2-4*K1*K4^2*K2)-
1/12/(3*K1^2*K4^2-4*K1*K4^2*K2)*((9*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2*((288*K1*Vth1*K4^2*K2*Vth2-72*K1*K4^2*K2*Vth2^2-
288*K1^2*Vth1*K4^2*Vth4-144*K1^2*Vth4*K4^2*Vdd-
144*K1*Vdd^2*K4^2*K2+144*K1*Vth4^2*K4^2*K2+144*K1^2*Vth1^2*K4^2-
216*K1*K4*K2*In+576*K1*Vth1*K4^2*K2*Vth4+72*K1^2*K4^2*Vdd^2-
432*K1^2*Vth1*K4^2*Vdd+144*In*K4*K1^2+144*K2^2*K4^2*Vth2^2+144*K1*Vth4*K4^2*K2*Vth2+288*K1*Vdd*Vth1*K
4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)-
(2304*K1*Vdd*K4*K2*In*Vth2+1152*K1*Vdd*Vth4*K4*K2*In+1152*K1*Vdd*Vth4*K4^2*K2*Vth2^2+1152*K1*Vdd^2*K4
^2*K2*Vth4^2+1152*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
576*K1*Vdd^2*K4*K2*In+576*K1*Vdd^2*K4^2*K2*Vth2^2+2304*K1^2*Vth1*K4^2*Vdd^2*Vth4-
2304*K1*Vdd*Vth1*K4^2*K2*Vth4^2-2304*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+1152*K1*Vdd*Vth1*K4*K2*In-
1152*K1*Vdd*Vth1*K4^2*K2*Vth2^2-1152*K1^2*Vdd^2*Vth1^2*K4^2+1152*K1^2*Vdd^3*Vth1*K4^2-
1152*In*K4*K1^2*Vdd^2-288*K2^2*In^2-1152*K1^2*Vdd^3*Vth4*K4^2-1152*K2^2*Vth4^2*K4^2*Vth2^2-
1152*K2^2*Vth4*Vth2^3*K4^2-576*K2^2*In*Vth2^2*K4-288*K1^2*Vdd^4*K4^2-288*K2^2*Vth2^4*K4^2-
1152*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+108*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(864*K1*Vdd*K4*K2*In*Vth2+432*K1*Vdd*Vth4*K4*K2*In+432*K1*Vdd*Vth4*K4^2*K2*Vth2^2+432*
K1*Vdd^2*K4^2*K2*Vth4^2+432*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
216*K1*Vdd^2*K4*K2*In+216*K1*Vdd^2*K4^2*K2*Vth2^2+864*K1^2*Vth1*K4^2*Vdd^2*Vth4-
864*K1*Vdd*Vth1*K4^2*K2*Vth4^2-864*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+432*K1*Vdd*Vth1*K4*K2*In-
432*K1*Vdd*Vth1*K4^2*K2*Vth2^2-432*K1^2*Vdd^2*Vth1^2*K4^2+432*K1^2*Vdd^3*Vth1*K4^2-
432*In*K4*K1^2*Vdd^2-108*K2^2*In^2-432*K1^2*Vdd^3*Vth4*K4^2-432*K2^2*Vth4^2*K4^2*Vth2^2-
432*K2^2*Vth4*Vth2^3*K4^2-216*K2^2*In*Vth2^2*K4-108*K1^2*Vdd^4*K4^2-108*K2^2*Vth2^4*K4^2-
432*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2+8*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3+12*3^(1/2)*(-(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-

8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+128*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth
2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+6*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K
1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(1536*K1*Vth4*K4^2*K2+1536*K1*Vth1*K4^2*K2+768*K1*K4^2*K2*Vth2-768*K1^2*Vth1*K4^2-
768*K1^2*Vth4*K4^2-
768*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3*(3*K1^2*K4^2-4*K1*K4^2*K2)-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-

16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)-(1152*K1*Vdd*K4*K2*In*Vth2+576*K1*Vdd*Vth4*K4*K2*In+576*K1*Vdd*Vth4*K4^2*K2*Vth2^2+576*K1*Vdd^2*K4^2*K2*Vth4^2+576*K1*Vdd^2*K4^2*K2*Vth4*Vth2-288*K1*Vdd^2*K4*K2*In+288*K1*Vdd^2*K4^2*K2*Vth2^2+1152*K1^2*Vth1*K4^2*Vdd^2*Vth4-1152*K1*Vdd*Vth1*K4^2*K2*Vth4^2-1152*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+576*K1*Vdd*Vth1*K4*K2*In-576*K1*Vdd*Vth1*K4^2*K2*Vth2^2-576*K1^2*Vdd^2*Vth1^2*K4^2+576*K1^2*Vdd^3*Vth1*K4^2-576*In*K4*K1^2*Vdd^2-144*K2^2*In^2-576*K1^2*Vdd^3*Vth4*K4^2-576*K2^2*Vth4^2*K4^2*Vth2^2-576*K2^2*Vth4*Vth2^3*K4^2-288*K2^2*In*Vth2^2*K4-144*K1^2*Vdd^4*K4^2-144*K2^2*Vth2^4*K4^2-576*K2^2*Vth4*Vth2*K4*In)*(-8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-4*K1*K4^2*K2)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2-144*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(-8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-4*K1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2-(128*K1*Vdd*K4*K2*In*Vth2+64*K1*Vdd*Vth4*K4*K2*In+64*K1*Vdd*Vth4*K4^2*K2*Vth2^2+64*K1*Vdd^2*K4^2*K2*Vth4^2+64*K1*Vdd^2*K4^2*K2*Vth4*Vth2-32*K1*Vdd^2*K4*K2*In+32*K1*Vdd^2*K4^2*K2*Vth2^2+128*K1^2*Vth1*K4^2*Vdd^2*Vth4-128*K1*Vdd*Vth1*K4^2*K2*Vth4^2-128*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+64*K1*Vdd*Vth1*K4*K2*In-64*K1*Vdd*Vth1*K4^2*K2*Vth2^2-64*K1^2*Vdd^2*Vth1^2*K4^2+64*K1^2*Vdd^3*Vth1*K4^2-64*In*K4*K1^2*Vdd^2-16*K2^2*In^2-64*K1^2*Vdd^3*Vth4*K4^2-64*K2^2*Vth4^2*K4^2*Vth2^2-64*K2^2*Vth4*Vth2^3*K4^2-32*K2^2*In*Vth2^2*K4-16*K1^2*Vdd^4*K4^2-16*K2^2*Vth2^4*K4^2-64*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-4*K1*K4^2*K2)*(-8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-8*K1*Vdd*Vth1*K4^2*K2)^4+4*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)*(-8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-8*K1*Vdd*Vth1*K4^2*K2)^3+(32*K1*Vdd*K4*K2*In*Vth2+16*K1*Vdd*Vth4*K4*K2*In+16*K1*Vdd*Vth4*K4^2*K2*Vth2^2+16*K1*Vdd^2*K4^2*K2*Vth4^2+16*K1*Vdd^2*K4^2*K2*Vth4*Vth2-8*K1*Vdd^2*K4*K2*In+8*K1*Vdd^2*K4^2*K2*Vth2^2+32*K1^2*Vth1*K4^2*Vdd^2*Vth4-32*K1*Vdd*Vth1*K4^2*K2*Vth4^2-32*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+16*K1*Vdd*Vth1*K4*K2*In-

16*K1*Vdd*Vth1*K4^2*K2*Vth2^2-16*K1^2*Vdd^2*Vth1^2*K4^2+16*K1^2*Vdd^3*Vth1*K4^2-16*In*K4*K1^2*Vdd^2-
4*K2^2*In^2-16*K1^2*Vdd^3*Vth4*K4^2-16*K2^2*Vth4^2*K4^2*Vth2^2-16*K2^2*Vth4*Vth2^3*K4^2-
8*K2^2*In*Vth2^2*K4-4*K1^2*Vdd^4*K4^2-4*K2^2*Vth2^4*K4^2-
16*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3-
256*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*V
th4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^3*(3*K1^2*K4^2-
4*K1*K4^2*K2)^3+27*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^4*(3*K1^2*K4^2-
4*K1*K4^2*K2)^2+27*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-
4*K2^2*Vth4*Vth2*K4*In)^2*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^4+4*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3+(640*K1*Vth4*K4^2*K2+640*K1*Vth1*K4^2*K2+320*K1*K4^2*K2*Vth2-320*K1^2*Vth1*K4^2-
320*K1^2*Vth4*K4^2-
320*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-4*K1*K4^2*K2)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2)^(1/2))^(1/3)-(-
192*K1*Vth1*K4^2*K2*Vth2+48*K1*K4^2*K2*Vth2^2+192*K1^2*Vth1*K4^2*Vth4+96*K1^2*Vth4*K4^2*Vdd+96*K1*Vdd
^2*K4^2*K2-96*K1*Vth4^2*K4^2*K2-96*K1^2*Vth1^2*K4^2+144*K1*K4*K2*In-384*K1*Vth1*K4^2*K2*Vth4-
48*K1^2*K4^2*Vdd^2+288*K1^2*Vth1*K4^2*Vdd-96*In*K4*K1^2-96*K2^2*K4^2*Vth2^2-96*K1*Vth4*K4^2*K2*Vth2-
192*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-4*K1*K4^2*K2)*((288*K1*Vth1*K4^2*K2*Vth2-72*K1*K4^2*K2*Vth2^2-
288*K1^2*Vth1*K4^2*Vth4-144*K1^2*Vth4*K4^2*Vdd-
144*K1*Vdd^2*K4^2*K2+144*K1*Vth4^2*K4^2*K2+144*K1^2*Vth1^2*K4^2-
216*K1*K4*K2*In+576*K1*Vth1*K4^2*K2*Vth4+72*K1^2*K4^2*Vdd^2-
432*K1^2*Vth1*K4^2*Vdd+144*In*K4*K1^2+144*K2^2*K4^2*Vth2^2+144*K1*Vth4*K4^2*K2*Vth2+288*K1*Vdd*Vth1*K

4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)-
(2304*K1*Vdd*K4*K2*In*Vth2+1152*K1*Vdd*Vth4*K4*K2*In+1152*K1*Vdd*Vth4*K4^2*K2*Vth2^2+1152*K1*Vdd^2*K4
^2*K2*Vth4^2+1152*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
576*K1*Vdd^2*K4*K2*In+576*K1*Vdd^2*K4^2*K2*Vth2^2+2304*K1^2*Vth1*K4^2*Vdd^2*Vth4-
2304*K1*Vdd*Vth1*K4^2*K2*Vth4^2-2304*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+1152*K1*Vdd*Vth1*K4*K2*In-
1152*K1*Vdd*Vth1*K4^2*K2*Vth2^2-1152*K1^2*Vdd^2*Vth1^2*K4^2+1152*K1^2*Vdd^3*Vth1*K4^2-
1152*In*K4*K1^2*Vdd^2-288*K2^2*In^2-1152*K1^2*Vdd^3*Vth4*K4^2-1152*K2^2*Vth4^2*K4^2*Vth2^2-
1152*K2^2*Vth4*Vth2^3*K4^2-576*K2^2*In*Vth2^2*K4-288*K1^2*Vdd^4*K4^2-288*K2^2*Vth2^4*K4^2-
1152*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+108*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(864*K1*Vdd*K4*K2*In*Vth2+432*K1*Vdd*Vth4*K4*K2*In+432*K1*Vdd*Vth4*K4^2*K2*Vth2^2+432*
K1*Vdd^2*K4^2*K2*Vth4^2+432*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
216*K1*Vdd^2*K4*K2*In+216*K1*Vdd^2*K4^2*K2*Vth2^2+864*K1^2*Vth1*K4^2*Vdd^2*Vth4-
864*K1*Vdd*Vth1*K4^2*K2*Vth4^2-864*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+432*K1*Vdd*Vth1*K4*K2*In-
432*K1*Vdd*Vth1*K4^2*K2*Vth2^2-432*K1^2*Vdd^2*Vth1^2*K4^2+432*K1^2*Vdd^3*Vth1*K4^2-
432*In*K4*K1^2*Vdd^2-108*K2^2*In^2-432*K1^2*Vdd^3*Vth4*K4^2-432*K2^2*Vth4^2*K4^2*Vth2^2-
432*K2^2*Vth4*Vth2^3*K4^2-216*K2^2*In*Vth2^2*K4-108*K1^2*Vdd^4*K4^2-108*K2^2*Vth2^4*K4^2-
432*K2^2*Vth4*Vth2*K4*In)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2+8*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^3+12*3^(1/2)*(-(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-
4*K1^2*Vth1*K4^2-4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+128*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth
2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)^2+6*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-

12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^2*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K
1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)*(3*K1^2*K4^2-
4*K1*K4^2*K2)+(1536*K1*Vth4*K4^2*K2+1536*K1*Vth1*K4^2*K2+768*K1*K4^2*K2*Vth2-768*K1^2*Vth1*K4^2-
768*K1^2*Vth4*K4^2-
768*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(3*K1^2*K4^2-4*K1*K4^2*K2)^2-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)^3*(3*K1^2*K4^2-4*K1*K4^2*K2)-(-
144*K1*Vth1*K4^2*K2*Vth2+36*K1*K4^2*K2*Vth2^2+144*K1^2*Vth1*K4^2*Vth4+72*K1^2*Vth4*K4^2*Vdd+72*K1*Vdd
^2*K4^2*K2-72*K1*Vth4^2*K4^2*K2-72*K1^2*Vth1^2*K4^2+108*K1*K4*K2*In-288*K1*Vth1*K4^2*K2*Vth4-
36*K1^2*K4^2*Vdd^2+216*K1^2*Vth1*K4^2*Vdd-72*In*K4*K1^2-72*K2^2*K4^2*Vth2^2-72*K1*Vth4*K4^2*K2*Vth2-
144*K1*Vdd*Vth1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-
4*K1^2*K4^2*Vdd)^3*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-
8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-
4*K1*Vth4*K4*K2*In)*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*
Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)-
(1152*K1*Vdd*K4*K2*In*Vth2+576*K1*Vdd*Vth4*K4*K2*In+576*K1*Vdd*Vth4*K4^2*K2*Vth2^2+576*K1*Vdd^2*K4^2*
K2*Vth4^2+576*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
288*K1*Vdd^2*K4*K2*In+288*K1*Vdd^2*K4^2*K2*Vth2^2+1152*K1^2*Vth1*K4^2*Vdd^2*Vth4-
1152*K1*Vdd*Vth1*K4^2*K2*Vth4^2-1152*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+576*K1*Vdd*Vth1*K4*K2*In-
576*K1*Vdd*Vth1*K4^2*K2*Vth2^2-576*K1^2*Vdd^2*Vth1^2*K4^2+576*K1^2*Vdd^3*Vth1*K4^2-
576*In*K4*K1^2*Vdd^2-144*K2^2*In^2-576*K1^2*Vdd^3*Vth4*K4^2-576*K2^2*Vth4^2*K4^2*Vth2^2-
576*K2^2*Vth4*Vth2^3*K4^2-288*K2^2*In*Vth2^2*K4-144*K1^2*Vdd^4*K4^2-144*K2^2*Vth2^4*K4^2-
576*K2^2*Vth4*Vth2*K4*In)*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^
2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)^2*(4*K2^2*Vth2*K4*In+8*K2^2*Vth4*K4^2*Vth2^2+8*K1*Vth1*K4^2*K2*Vth4^2-
8*K1*Vdd^2*K4^2*K2*Vth4-4*K1*Vdd^2*K4^2*K2*Vth2+4*K1*Vth1*K4^2*K2*Vth2^2+4*K1^2*Vdd^3*K4^2-

8*K1*K4*K2*In*Vth2-4*K1*Vdd*K4^2*K2*Vth2^2+8*K1^2*Vth1^2*K4^2*Vdd-
16*K1^2*Vth1*K4^2*Vdd*Vth4+4*K1^2*Vth4*K4^2*Vdd^2+8*K1*Vdd*Vth1*K4^2*K2*Vth2+8*In*K4*K1^2*Vdd-
4*K1*Vdd*K4*K2*In+4*K2^2*Vth2^3*K4^2+8*K1*Vth1*K4^2*K2*Vth4*Vth2-4*K1*Vth1*K4*K2*In-
12*K1^2*Vth1*K4^2*Vdd^2+16*K1*Vdd*Vth1*K4^2*K2*Vth4-4*K1*Vth4*K4^2*K2*Vth2^2-4*K1*Vth4*K4*K2*In)^2-
144*(8*K1*Vdd*K4*K2*In*Vth2+4*K1*Vdd*Vth4*K4*K2*In+4*K1*Vdd*Vth4*K4^2*K2*Vth2^2+4*K1*Vdd^2*K4^2*K2*Vth4^2+4*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
2*K1*Vdd^2*K4*K2*In+2*K1*Vdd^2*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vdd^2*Vth4-
8*K1*Vdd*Vth1*K4^2*K2*Vth4^2-8*K1*Vdd*Vth1*K4^2*K2*Vth4*Vth2+4*K1*Vdd*Vth1*K4*K2*In-
4*K1*Vdd*Vth1*K4^2*K2*Vth2^2-4*K1^2*Vdd^2*Vth1^2*K4^2+4*K1^2*Vdd^3*Vth1*K4^2-4*In*K4*K1^2*Vdd^2-
K2^2*In^2-4*K1^2*Vdd^3*Vth4*K4^2-4*K2^2*Vth4^2*K4^2*Vth2^2-4*K2^2*Vth4*Vth2^3*K4^2-2*K2^2*In*Vth2^2*K4-
K1^2*Vdd^4*K4^2-K2^2*Vth2^4*K4^2-4*K2^2*Vth4*Vth2*K4*In)^2*(-
8*K1*Vth1*K4^2*K2*Vth2+2*K1*K4^2*K2*Vth2^2+8*K1^2*Vth1*K4^2*Vth4+4*K1^2*Vth4*K4^2*Vdd+4*K1*Vdd^2*K4^2*K2-4*K1*Vth4^2*K4^2*K2-4*K1^2*Vth1^2*K4^2+6*K1*K4*K2*In-16*K1*Vth1*K4^2*K2*Vth4-
2*K1^2*K4^2*Vdd^2+12*K1^2*Vth1*K4^2*Vdd-4*In*K4*K1^2-4*K2^2*K4^2*Vth2^2-4*K1*Vth4*K4^2*K2*Vth2-
8*K1*Vdd*Vth1*K4^2*K2)*(3*K1^2*K4^2-
4*K1*K4^2*K2)*(8*K1*Vth4*K4^2*K2+8*K1*Vth1*K4^2*K2+4*K1*K4^2*K2*Vth2-4*K1^2*Vth1*K4^2-
4*K1^2*Vth4*K4^2-4*K1^2*K4^2*Vdd)^2-
(128*K1*Vdd*K4*K2*In*Vth2+64*K1*Vdd*Vth4*K4*K2*In+64*K1*Vdd*Vth4*K4^2*K2*Vth2^2+64*K1*Vdd^2*K4^2*K2*Vth4^2+64*K1*Vdd^2*K4^2*K2*Vth4*Vth2-
32*K1*Vdd^2*K4*K2*In+32*K1*Vdd^2*K4^2*K2*Vth2^2+128*K1^2*Vth1*K4^2*Vd... Output truncated. Text exceeds maximum line length of 25,000 characters for Command Window display.

Region 8A:

First solution:

V1=(K3^2*K1*Vth3*K2+K3*K1^2*Vth3*K4+3*K3*K2*Vth2*K4*K1-K3^2*K2^2*Vth2+2*K4^2*Vdd*K1^2-
K4*K1*Vth4*K3*K2-4*K3*K1^2*Vdd*K4-Vth1*K1^2*K4^2-2*K1*K2*Vth2*K4^2+2*K3*K1^2*Vth1*K4-
K3*K2*K4*K1*Vth1-Vth4*K1^2*K4^2+2*K3*K2*K4*K1*Vdd-2*K3*K2*K4*K1*Vth3+2*K4*K1^2*Vth4*K3)/(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+1/12/(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*((9*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*((216*K3^2*K2^2*Vth2^2+144*In*K1^2*K3+144*K3^2*K1^2*Vth3^2-
72*K4*K1^2*In+144*Vth4^2*K1^2*K4^2-144*K4^2*Vdd^2*K1^2+288*K3*K2*Vth2*K4*K1*Vth1-
576*K3*K2*Vth2*K4*K1*Vdd+288*K4*K1*Vth4*K3*K2*Vth2-
432*K3*K1^2*Vdd*Vth3*K4+144*K3*K1^2*Vth1*Vth3*K4+288*K4*K1^2*Vth4*K3*Vth3-
216*K3*K2*Vth2^2*K4*K1+576*K4*K1^2*Vth4*K3*Vth1-864*K3*K1^2*Vdd*K4*Vth1-
288*K3^2*K2*Vth2*K1*Vth3+144*K3^2*K1*Vth1*Vth3*K2-576*K4*K1^2*Vth4*K3*Vdd-
144*K3^2*Vdd*K1*Vth3*K2+576*K3*K2*Vth2*K4*K1*Vth3+144*K3*K1^2*Vth1^2*K4-
72*In*K1*K3*K2+144*K1*K2*Vth2^2*K4^2-288*Vth1*Vth4*K1^2*K4^2+72*K3^2*K2*Vth3^2*K1-
72*K3*Vth3^2*K1^2*K4+288*K4^2*Vth1*Vdd*K1^2+720*K3*K1^2*Vdd^2*K4)*(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)-
(1152*K3^2*Vdd*K1*Vth3*K2*Vth2^2-1152*K3^2*K1*Vth1*Vth3*K2*Vth2^2-576*K3^2*K2*Vth2^2*Vth3^2*K1-
288*K3^2*Vth3^4*K1^2-288*In^2*K1^2-576*In*K1^2*K3*Vth3^2-
288*K3^2*K2^2*Vth2^4+576*In*K1*K3*K2*Vth2^2+2304*In*K1^2*K3*Vdd*Vth1+1152*In*K1^2*K3*Vdd*Vth3-
1152*In*K1^2*K3*Vth1*Vth3+1152*K3^2*K1^2*Vdd*Vth3^3+2304*K3^2*K1^2*Vdd*Vth1*Vth3^2-
1152*In*K1^2*K3*Vdd^2-1152*K3^2*Vdd^2*K1^2*Vth3^2-1152*In*K1^2*K3*Vth1^2-1152*K3^2*K1^2*Vth1*Vth3^3-
1152*K3^2*K1^2*Vth1^2*Vth3^2)*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-
4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-
8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-4*K3*K1^2*Vth1*Vth3*K4-
8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-

16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+108*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(432*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
432*K3^2*K1*Vth1*Vth3*K2*Vth2^2-216*K3^2*K2*Vth2^2*Vth3^2*K1-108*K3^2*Vth3^4*K1^2-108*In^2*K1^2-
216*In*K1^2*K3*Vth3^2-
108*K3^2*K2^2*Vth2^4+216*In*K1*K3*K2*Vth2^2+864*In*K1^2*K3*Vdd*Vth1+432*In*K1^2*K3*Vdd*Vth3-
432*In*K1^2*K3*Vth1*Vth3+432*K3^2*K1^2*Vdd*Vth3^3+864*K3^2*K1^2*Vdd*Vth1*Vth3^2-432*In*K1^2*K3*Vdd^2-
432*K3^2*Vdd^2*K1^2*Vth3^2-432*In*K1^2*K3*Vth1^2-432*K3^2*K1^2*Vth1*Vth3^3-432*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2+8*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-
4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-
8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-4*K3*K1^2*Vth1*Vth3*K4-
8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)^3+12*3^(1/2)*(-(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2*(-6*K3^2*K2^2*Vth2^2-
4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-
20*K3*K1^2*Vdd^2*K4)^2+128*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-
4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)^2+6*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-

8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(-768*K3^2*K1*Vth3*K2-
768*K3*K1^2*Vth3*K4-2304*K3*K2*Vth2*K4*K1+768*K3^2*K2^2*Vth2-
1536*K4^2*Vdd*K1^2+768*K4*K1*Vth4*K3*K2+3072*K3*K1^2*Vdd*K4+768*Vth1*K1^2*K4^2+1536*K1*K2*Vth2*K4^2-
1536*K3*K1^2*Vth1*K4+768*K3*K2*K4*K1*Vth1+768*Vth4*K1^2*K4^2-
1536*K3*K2*K4*K1*Vdd+1536*K3*K2*K4*K1*Vth3-1536*K4*K1^2*Vth4*K3)*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2-(-108*K3^2*K2^2*Vth2^2-72*In*K1^2*K3-
72*K3^2*K1^2*Vth3^2+36*K4*K1^2*In-72*Vth4^2*K1^2*K4^2+72*K4^2*Vdd^2*K1^2-
144*K3*K2*Vth2*K4*K1*Vth1+288*K3*K2*Vth2*K4*K1*Vdd-
144*K4*K1*Vth4*K3*K2*Vth2+216*K3*K1^2*Vdd*Vth3*K4-72*K3*K1^2*Vth1*Vth3*K4-
144*K4*K1^2*Vth4*K3*Vth3+108*K3*K2*Vth2^2*K4*K1-
288*K4*K1^2*Vth4*K3*Vth1+432*K3*K1^2*Vdd*K4*Vth1+144*K3^2*K2*Vth2*K1*Vth3-
72*K3^2*K1*Vth1*Vth3*K2+288*K4*K1^2*Vth4*K3*Vdd+72*K3^2*Vdd*K1*Vth3*K2-288*K3*K2*Vth2*K4*K1*Vth3-
72*K3*K1^2*Vth1^2*K4+36*In*K1*K3*K2-72*K1*K2*Vth2^2*K4^2+144*Vth1*Vth4*K1^2*K4^2-
36*K3^2*K2*Vth3^2*K1+36*K3*Vth3^2*K1^2*K4-144*K4^2*Vth1*Vdd*K1^2-360*K3*K1^2*Vdd^2*K4)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^3*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)-(-108*K3^2*K2^2*Vth2^2-72*In*K1^2*K3-
72*K3^2*K1^2*Vth3^2+36*K4*K1^2*In-72*Vth4^2*K1^2*K4^2+72*K4^2*Vdd^2*K1^2-
144*K3*K2*Vth2*K4*K1*Vth1+288*K3*K2*Vth2*K4*K1*Vdd-
144*K4*K1*Vth4*K3*K2*Vth2+216*K3*K1^2*Vdd*Vth3*K4-72*K3*K1^2*Vth1*Vth3*K4-
144*K4*K1^2*Vth4*K3*Vth3+108*K3*K2*Vth2^2*K4*K1-
288*K4*K1^2*Vth4*K3*Vth1+432*K3*K1^2*Vdd*K4*Vth1+144*K3^2*K2*Vth2*K1*Vth3-
72*K3^2*K1*Vth1*Vth3*K2+288*K4*K1^2*Vth4*K3*Vdd+72*K3^2*Vdd*K1*Vth3*K2-288*K3*K2*Vth2*K4*K1*Vth3-
72*K3*K1^2*Vth1^2*K4+36*In*K1*K3*K2-72*K1*K2*Vth2^2*K4^2+144*Vth1*Vth4*K1^2*K4^2-
36*K3^2*K2*Vth3^2*K1+36*K3*Vth3^2*K1^2*K4-144*K4^2*Vth1*Vdd*K1^2-360*K3*K1^2*Vdd^2*K4)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-

8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^3*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)-
(576*K3^2*Vdd*K1*Vth3*K2*Vth2^2-576*K3^2*K1*Vth1*Vth3*K2*Vth2^2-288*K3^2*K2*Vth2^2*Vth3^2*K1-
144*K3^2*Vth3^4*K1^2-144*In^2*K1^2-288*In*K1^2*K3*Vth3^2-
144*K3^2*K2^2*Vth2^4+288*In*K1*K3*K2*Vth2^2+1152*In*K1^2*K3*Vdd*Vth1+576*In*K1^2*K3*Vdd*Vth3-
576*In*K1^2*K3*Vth1*Vth3+576*K3^2*K1^2*Vdd*Vth3^3+1152*K3^2*K1^2*Vdd*Vth1*Vth3^2-576*In*K1^2*K3*Vdd^2-
576*K3^2*Vdd^2*K1^2*Vth3^2-576*In*K1^2*K3*Vth1^2-576*K3^2*K1^2*Vth1*Vth3^3-576*K3^2*K1^2*Vth1^2*Vth3^2)*(-
6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2-
144*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-
K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2-(64*K3^2*Vdd*K1*Vth3*K2*Vth2^2-64*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
32*K3^2*K2*Vth2^2*Vth3^2*K1-16*K3^2*Vth3^4*K1^2-16*In^2*K1^2-32*In*K1^2*K3*Vth3^2-
16*K3^2*K2^2*Vth2^4+32*In*K1*K3*K2*Vth2^2+128*In*K1^2*K3*Vdd*Vth1+64*In*K1^2*K3*Vdd*Vth3-
64*In*K1^2*K3*Vth1*Vth3+64*K3^2*K1^2*Vdd*Vth3^3+128*K3^2*K1^2*Vdd*Vth1*Vth3^2-64*In*K1^2*K3*Vdd^2-
64*K3^2*Vdd^2*K1^2*Vth3^2-64*In*K1^2*K3*Vth1^2-64*K3^2*K1^2*Vth1*Vth3^3-64*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-
4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-

2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-
20*K3*K1^2*Vdd^2*K4)^4+4*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-
4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-
20*K3*K1^2*Vdd^2*K4)^3+(16*K3^2*Vdd*K1*Vth3*K2*Vth2^2-16*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
8*K3^2*K2*Vth2^2*Vth3^2*K1-4*K3^2*Vth3^4*K1^2-4*In^2*K1^2-8*In*K1^2*K3*Vth3^2-
4*K3^2*K2^2*Vth2^4+8*In*K1*K3*K2*Vth2^2+32*In*K1^2*K3*Vdd*Vth1+16*In*K1^2*K3*Vdd*Vth3-
16*In*K1^2*K3*Vth1*Vth3+16*K3^2*K1^2*Vdd*Vth3^3+32*K3^2*K1^2*Vdd*Vth1*Vth3^2-16*In*K1^2*K3*Vdd^2-
16*K3^2*Vdd^2*K1^2*Vth3^2-16*In*K1^2*K3*Vth1^2-16*K3^2*K1^2*Vth1*Vth3^3-16*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-
4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-
8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-4*K3*K1^2*Vth1*Vth3*K4-
8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)^3-
256*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-
K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^3*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^3+27*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^4*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2+27*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^4+4*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^3*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-

4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^3+(-320*K3^2*K1*Vth3*K2-
320*K3*K1^2*Vth3*K4-960*K3*K2*Vth2*K4*K1+320*K3^2*K2^2*Vth2-
640*K4^2*Vdd*K1^2+320*K4*K1*Vth4*K3*K2+1280*K3*K1^2*Vdd*K4+320*Vth1*K1^2*K4^2+640*K1*K2*Vth2*K4^2-
640*K3*K1^2*Vth1*K4+320*K3*K2*K4*K1*Vth1+320*Vth4*K1^2*K4^2-
640*K3*K2*K4*K1*Vdd+640*K3*K2*K4*K1*Vth3-640*K4*K1^2*Vth4*K3)*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-
4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-
8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-
4*K3*K1^2*Vth1*Vth3*K4-8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)^2)^(1/2))^(1/3)-(-
144*K3^2*K2^2*Vth2^2-96*In*K1^2*K3-96*K3^2*K1^2*Vth3^2+48*K4*K1^2*In-
96*Vth4^2*K1^2*K4^2+96*K4^2*Vdd^2*K1^2-192*K3*K2*Vth2*K4*K1*Vth1+384*K3*K2*Vth2*K4*K1*Vdd-
192*K4*K1*Vth4*K3*K2*Vth2+288*K3*K1^2*Vdd*Vth3*K4-96*K3*K1^2*Vth1*Vth3*K4-
192*K4*K1^2*Vth4*K3*Vth3+144*K3*K2*Vth2^2*K4*K1-
384*K4*K1^2*Vth4*K3*Vth1+576*K3*K1^2*Vdd*K4*Vth1+192*K3^2*K2*Vth2*K1*Vth3-
96*K3^2*K1*Vth1*Vth3*K2+384*K4*K1^2*Vth4*K3*Vdd+96*K3^2*Vdd*K1*Vth3*K2-384*K3*K2*Vth2*K4*K1*Vth3-
96*K3*K1^2*Vth1^2*K4+48*In*K1*K3*K2-96*K1*K2*Vth2^2*K4^2+192*Vth1*Vth4*K1^2*K4^2-
48*K3^2*K2*Vth3^2*K1+48*K3*Vth3^2*K1^2*K4-192*K4^2*Vth1*Vdd*K1^2-480*K3*K1^2*Vdd^2*K4)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-
4*K3*K1^2*K4)*((216*K3^2*K2^2*Vth2^2+144*In*K1^2*K3+144*K3^2*K1^2*Vth3^2-
72*K4*K1^2*In+144*Vth4^2*K1^2*K4^2-144*K4^2*Vdd^2*K1^2+288*K3*K2*Vth2*K4*K1*Vth1-
576*K3*K2*Vth2*K4*K1*Vdd+288*K4*K1*Vth4*K3*K2*Vth2-
432*K3*K1^2*Vdd*Vth3*K4+144*K3*K1^2*Vth1*Vth3*K4+288*K4*K1^2*Vth4*K3*Vth3-
216*K3*K2*Vth2^2*K4*K1+576*K4*K1^2*Vth4*K3*Vth1-864*K3*K1^2*Vdd*K4*Vth1-
288*K3^2*K2*Vth2*K1*Vth3+144*K3^2*K1*Vth1*Vth3*K2-576*K4*K1^2*Vth4*K3*Vdd-
144*K3^2*Vdd*K1*Vth3*K2+576*K3*K2*Vth2*K4*K1*Vth3+144*K3*K1^2*Vth1^2*K4-
72*In*K1*K3*K2+144*K1*K2*Vth2^2*K4^2-288*Vth1*Vth4*K1^2*K4^2+72*K3^2*K2*Vth3^2*K1-
72*K3*Vth3^2*K1^2*K4+288*K4^2*Vth1*Vdd*K1^2+720*K3*K1^2*Vdd^2*K4)*(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-4*In*K1^2*K3*Vth3-
4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-8*K4*K1^2*Vth4*K3*Vdd^2-
16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-4*In*K1*K3*K2*Vth2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)-
(1152*K3^2*Vdd*K1*Vth3*K2*Vth2^2-1152*K3^2*K1*Vth1*Vth3*K2*Vth2^2-576*K3^2*K2*Vth2^2*Vth3^2*K1-
288*K3^2*Vth3^4*K1^2-288*In^2*K1^2-576*In*K1^2*K3*Vth3^2-
288*K3^2*K2^2*Vth2^4+576*In*K1*K3*K2*Vth2^2+2304*In*K1^2*K3*Vdd*Vth1+1152*In*K1^2*K3*Vdd*Vth3-
1152*In*K1^2*K3*Vth1*Vth3+1152*K3^2*K1^2*Vdd*Vth3^3+2304*K3^2*K1^2*Vdd*Vth1*Vth3^2-
1152*In*K1^2*K3*Vdd^2-1152*K3^2*Vdd^2*K1^2*Vth3^2-1152*In*K1^2*K3*Vth1^2-1152*K3^2*K1^2*Vth1*Vth3^3-

```
1152*K3^2*K1^2*Vth1^2*Vth3^2)*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-
4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-
8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-4*K3*K1^2*Vth1*Vth3*K4-
8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+108*(16*K4*K1^2*Vth4*K3*Vdd*Vth1-
4*In*K1^2*K3*Vth3-4*K3^2*K1^2*Vth3^3+4*K3^2*K2^2*Vth2^3-4*K3^2*K1*Vth3*K2*Vth2^2-
8*K4*K1^2*Vth4*K3*Vdd^2-16*K3*K1^2*Vth1*K4*Vdd^2+4*K3^2*K2*Vth2*Vth3^2*K1+4*K3*K1^2*Vth1*K4*Vth3^2-
4*In*K1*K3*K2*Vth2-8*K3*K1^2*Vdd^2*K4*Vth3-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1^2+8*K3*K1^2*Vdd*Vth1^2*K4+4*Vth1*K4*K1^2*In+8*K3^2*K1^2*Vdd*Vth3^2-
4*In*Vth4*K1^2*K4-8*In*K1^2*K3*Vth1-8*K3^2*K1^2*Vth1*Vth3^2-
8*K3^2*K2*Vth2*Vdd*K1*Vth3+8*K3*K2*Vth2^2*K4*K1*Vth3+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K3*K1^2*Vdd*Vth3*K
4*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K4*K1^2*Vth4*K3*Vdd*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2+8*K3^2*K2*Vth2*K1*Vth
1*Vth3-8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3)^2*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(432*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
432*K3^2*K1*Vth1*Vth3*K2*Vth2^2-216*K3^2*K2*Vth2^2*Vth3^2*K1-108*K3^2*Vth3^4*K1^2-108*In^2*K1^2-
216*In*K1^2*K3*Vth3^2-
108*K3^2*K2^2*Vth2^4+216*In*K1*K3*K2*Vth2^2+864*In*K1^2*K3*Vdd*Vth1+432*In*K1^2*K3*Vdd*Vth3-
432*In*K1^2*K3*Vth1*Vth3+432*K3^2*K1^2*Vdd*Vth3^3+864*K3^2*K1^2*Vdd*Vth1*Vth3^2-432*In*K1^2*K3*Vdd^2-
432*K3^2*Vdd^2*K1^2*Vth3^2-432*In*K1^2*K3*Vth1^2-432*K3^2*K1^2*Vth1*Vth3^3-432*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2+8*(-6*K3^2*K2^2*Vth2^2-4*In*K1^2*K3-4*K3^2*K1^2*Vth3^2+2*K4*K1^2*In-
4*Vth4^2*K1^2*K4^2+4*K4^2*Vdd^2*K1^2-8*K3*K2*Vth2*K4*K1*Vth1+16*K3*K2*Vth2*K4*K1*Vdd-
8*K4*K1*Vth4*K3*K2*Vth2+12*K3*K1^2*Vdd*Vth3*K4-4*K3*K1^2*Vth1*Vth3*K4-
8*K4*K1^2*Vth4*K3*Vth3+6*K3*K2*Vth2^2*K4*K1-
16*K4*K1^2*Vth4*K3*Vth1+24*K3*K1^2*Vdd*K4*Vth1+8*K3^2*K2*Vth2*K1*Vth3-
4*K3^2*K1*Vth1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd+4*K3^2*Vdd*K1*Vth3*K2-16*K3*K2*Vth2*K4*K1*Vth3-
4*K3*K1^2*Vth1^2*K4+2*In*K1*K3*K2-4*K1*K2*Vth2^2*K4^2+8*Vth1*Vth4*K1^2*K4^2-
2*K3^2*K2*Vth3^2*K1+2*K3*Vth3^2*K1^2*K4-8*K4^2*Vth1*Vdd*K1^2-20*K3*K1^2*Vdd^2*K4)^3+12*3^(1/2)*(-(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth... Output
truncated.  Text exceeds maximum line length of 25,000 characters for Command Window display.
```

Second solution:

V1=
```
(K3^2*K1*Vth3*K2+K3*K1^2*Vth3*K4+3*K3*K2*Vth2*K4*K1-K3^2*K2^2*Vth2+2*K4^2*Vdd*K1^2-
K4*K1*Vth4*K3*K2-4*K3*K1^2*Vdd*K4-Vth1*K1^2*K4^2-2*K1*K2*Vth2*K4^2+2*K3*K1^2*Vth1*K4-
K3*K2*K4*K1*Vth1-Vth4*K1^2*K4^2+2*K3*K2*K4*K1*Vdd-2*K3*K2*K4*K1*Vth3+2*K4*K1^2*Vth4*K3)/(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)-1/12/(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*((9*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*((-144*K4^2*Vdd^2*K1^2+144*Vth4^2*K1^2*K4^2-
72*K4*K1^2*In+144*K3^2*K1^2*Vth3^2+144*In*K1^2*K3+216*K3^2*K2^2*Vth2^2-
432*K3*K1^2*Vdd*Vth3*K4+576*K3*K2*Vth2*K4*K1*Vth3+288*K4*K1*Vth4*K3*K2*Vth2-
576*K3*K2*Vth2*K4*K1*Vdd+288*K3*K2*Vth2*K4*K1*Vth1-144*K3^2*Vdd*K1*Vth3*K2-
576*K4*K1^2*Vth4*K3*Vdd+144*K3^2*K1*Vth1*Vth3*K2-288*K3^2*K2*Vth2*K1*Vth3-
864*K3*K1^2*Vdd*K4*Vth1+576*K4*K1^2*Vth4*K3*Vth1-
216*K3*K2*Vth2^2*K4*K1+288*K4*K1^2*Vth4*K3*Vth3+144*K3*K1^2*Vth1*Vth3*K4+144*K1*K2*Vth2^2*K4^2-
72*In*K1*K3*K2+144*K3*K1^2*Vth1^2*K4-288*Vth1*Vth4*K1^2*K4^2-
72*K3*Vth3^2*K1^2*K4+72*K3^2*K2*Vth3^2*K1+720*K3*K1^2*Vdd^2*K4+288*K4^2*Vth1*Vdd*K1^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
```

16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)-
(1152*K3^2*Vdd*K1*Vth3*K2*Vth2^2-1152*K3^2*K1*Vth1*Vth3*K2*Vth2^2-576*K3^2*K2*Vth2^2*Vth3^2*K1-
288*K3^2*Vth3^4*K1^2-288*In^2*K1^2-576*In*K1^2*K3*Vth3^2-
288*K3^2*K2^2*Vth2^4+576*In*K1*K3*K2*Vth2^2+2304*In*K1^2*K3*Vdd*Vth1+1152*In*K1^2*K3*Vdd*Vth3-
1152*In*K1^2*K3*Vth1*Vth3+1152*K3^2*K1^2*Vdd*Vth3^3+2304*K3^2*K1^2*Vdd*Vth1*Vth3^2-
1152*In*K1^2*K3*Vdd^2-1152*K3^2*Vdd^2*K1^2*Vth3^2-1152*In*K1^2*K3*Vth1^2-1152*K3^2*K1^2*Vth1*Vth3^3-
1152*K3^2*K1^2*Vth1^2*Vth3^2)*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+108*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(432*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
432*K3^2*K1*Vth1*Vth3*K2*Vth2^2-216*K3^2*K2*Vth2^2*Vth3^2*K1-108*K3^2*Vth3^4*K1^2-108*In^2*K1^2-
216*In*K1^2*K3*Vth3^2-
108*K3^2*K2^2*Vth2^4+216*In*K1*K3*K2*Vth2^2+864*In*K1^2*K3*Vdd*Vth1+432*In*K1^2*K3*Vdd*Vth3-
432*In*K1^2*K3*Vth1*Vth3+432*K3^2*K1^2*Vdd*Vth3^3+864*K3^2*K1^2*Vdd*Vth1*Vth3^2-432*In*K1^2*K3*Vdd^2-
432*K3^2*Vdd^2*K1^2*Vth3^2-432*In*K1^2*K3*Vth1^2-432*K3^2*K1^2*Vth1*Vth3^3-432*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2+8*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^3+12*3^(1/2)*(-(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*In*K1^2*K3*Vth1)^2*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^2+128*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-

K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2*(4*K4^2*Vdd^2*K1^2-
4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^2+6*(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*In*K1^2*K3*Vth1)^2*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(-768*K3^2*K1*Vth3*K2-
768*K3*K1^2*Vth3*K4-2304*K3*K2*Vth2*K4*K1+768*K3^2*K2^2*Vth2-
1536*K4^2*Vdd*K1^2+768*K4*K1*Vth4*K3*K2+3072*K3*K1^2*Vdd*K4+768*Vth1*K1^2*K4^2+1536*K1*K2*Vth2*K4^2-
1536*K3*K1^2*Vth1*K4+768*K3*K2*K4*K1*Vth1+768*Vth4*K1^2*K4^2-
1536*K3*K2*K4*K1*Vdd+1536*K3*K2*K4*K1*Vth3-1536*K4*K1^2*Vth4*K3)*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*In*K1^2*K3*Vth1)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2-(72*K4^2*Vdd^2*K1^2-
72*Vth4^2*K1^2*K4^2+36*K4*K1^2*In-72*K3^2*K1^2*Vth3^2-72*In*K1^2*K3-
108*K3^2*K2^2*Vth2^2+216*K3*K1^2*Vdd*Vth3*K4-288*K3*K2*Vth2*K4*K1*Vth3-
144*K4*K1*Vth4*K3*K2*Vth2+288*K3*K2*Vth2*K4*K1*Vdd-
144*K3*K2*Vth2*K4*K1*Vth1+72*K3^2*Vdd*K1*Vth3*K2+288*K4*K1^2*Vth4*K3*Vdd-
72*K3^2*K1*Vth1*Vth3*K2+144*K3^2*K2*Vth2*K1*Vth3+432*K3*K1^2*Vdd*K4*Vth1-
288*K4*K1^2*Vth4*K3*Vth1+108*K3*K2*Vth2^2*K4*K1-144*K4*K1^2*Vth4*K3*Vth3-72*K3*K1^2*Vth1*Vth3*K4-
72*K1*K2*Vth2^2*K4^2+36*In*K1*K3*K2-
72*K3*K1^2*Vth1^2*K4+144*Vth1*Vth4*K1^2*K4^2+36*K3*Vth3^2*K1^2*K4-36*K3^2*K2*Vth3^2*K1-
360*K3*K1^2*Vdd^2*K4-144*K4^2*Vth1*Vdd*K1^2)*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-

```
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^3*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)-(72*K4^2*Vdd^2*K1^2-
72*Vth4^2*K1^2*K4^2+36*K4*K1^2*In-72*K3^2*K1^2*Vth3^2-72*In*K1^2*K3-
108*K3^2*K2^2*Vth2^2+216*K3*K1^2*Vdd*Vth3*K4-288*K3*K2*Vth2*K4*K1*Vth3-
144*K4*K1*Vth4*K3*K2*Vth2+288*K3*K2*Vth2*K4*K1*Vdd-
144*K3*K2*Vth2*K4*K1*Vth1+72*K3^2*Vdd*K1*Vth3*K2+288*K4*K1^2*Vth4*K3*Vdd-
72*K3^2*K1*Vth1*Vth3*K2+144*K3^2*K2*Vth2*K1*Vth3+432*K3*K1^2*Vdd*K4*Vth1-
288*K4*K1^2*Vth4*K3*Vth1+108*K3*K2*Vth2^2*K4*K1-144*K4*K1^2*Vth4*K3*Vth3-72*K3*K1^2*Vth1*Vth3*K4-
72*K1*K2*Vth2^2*K4^2+36*In*K1*K3*K2-
72*K3*K1^2*Vth1^2*K4+144*Vth1*Vth4*K1^2*K4^2+36*K3*Vth3^2*K1^2*K4-36*K3^2*K2*Vth3^2*K1-
360*K3*K1^2*Vdd^2*K4-144*K4^2*Vth1*Vdd*K1^2)*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^3*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*In*K1^2*K3*Vth1)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)-
(576*K3^2*Vdd*K1*Vth3*K2*Vth2^2-576*K3^2*K1*Vth1*Vth3*K2*Vth2^2-288*K3^2*K2*Vth2^2*Vth3^2*K1-
144*K3^2*Vth3^4*K1^2-144*In^2*K1^2-288*In*K1^2*K3*Vth3^2-
144*K3^2*K2^2*Vth2^4+288*In*K1*K3*K2*Vth2^2+1152*In*K1^2*K3*Vdd*Vth1+576*In*K1^2*K3*Vdd*Vth3-
576*In*K1^2*K3*Vth1*Vth3+576*K3^2*K1^2*Vdd*Vth3^3+1152*K3^2*K1^2*Vdd*Vth1*Vth3^2-576*In*K1^2*K3*Vdd^2-
576*K3^2*Vdd^2*K1^2*Vth3^2-576*In*K1^2*K3*Vth1^2-576*K3^2*K1^2*Vth1*Vth3^3-
576*K3^2*K1^2*Vth1^2*Vth3^2)*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^2-
144*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-
K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-
4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-
```

12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2-(64*K3^2*Vdd*K1*Vth3*K2*Vth2^2-64*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
32*K3^2*K2*Vth2^2*Vth3^2*K1-16*K3^2*Vth3^4*K1^2-16*In^2*K1^2-32*In*K1^2*K3*Vth3^2-
16*K3^2*K2^2*Vth2^4+32*In*K1*K3*K2*Vth2^2+128*In*K1^2*K3*Vdd*Vth1+64*In*K1^2*K3*Vdd*Vth3-
64*In*K1^2*K3*Vth1*Vth3+64*K3^2*K1^2*Vdd*Vth3^3+128*K3^2*K1^2*Vdd*Vth1*Vth3^2-64*In*K1^2*K3*Vdd^2-
64*K3^2*Vdd^2*K1^2*Vth3^2-64*In*K1^2*K3*Vth1^2-64*K3^2*K1^2*Vth1*Vth3^3-64*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(4*K4^2*Vdd^2*K1^2-
4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^4+4*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(4*K4^2*Vdd^2*K1^2-
4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^3+(16*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
16*K3^2*K1*Vth1*Vth3*K2*Vth2^2-8*K3^2*K2*Vth2^2*Vth3^2*K1-4*K3^2*Vth3^4*K1^2-4*In^2*K1^2-
8*In*K1^2*K3*Vth3^2-
4*K3^2*K2^2*Vth2^4+8*In*K1*K3*K2*Vth2^2+32*In*K1^2*K3*Vdd*Vth1+16*In*K1^2*K3*Vdd*Vth3-
16*In*K1^2*K3*Vth1*Vth3+16*K3^2*K1^2*Vdd*Vth3^3+32*K3^2*K1^2*Vdd*Vth1*Vth3^2-16*In*K1^2*K3*Vdd^2-
16*K3^2*Vdd^2*K1^2*Vth3^2-16*In*K1^2*K3*Vth1^2-16*K3^2*K1^2*Vth1*Vth3^3-16*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^3-256*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^3*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^3+27*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^4*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)^2+27*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-

4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)^2*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^4+4*(-4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^3*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^3+(-
320*K3^2*K1*Vth3*K2-320*K3*K1^2*Vth3*K4-960*K3*K2*Vth2*K4*K1+320*K3^2*K2^2*Vth2-
640*K4^2*Vdd*K1^2+320*K4*K1*Vth4*K3*K2+1280*K3*K1^2*Vdd*K4+320*Vth1*K1^2*K4^2+640*K1*K2*Vth2*K4^2-
640*K3*K1^2*Vth1*K4+320*K3*K2*K4*K1*Vth1+320*Vth4*K1^2*K4^2-
640*K3*K2*K4*K1*Vdd+640*K3*K2*K4*K1*Vth3-640*K4*K1^2*Vth4*K3)*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-
8*In*K1^2*K3*Vth1)*(4*K3^2*Vdd*K1*Vth3*K2*Vth2^2-4*K3^2*K1*Vth1*Vth3*K2*Vth2^2-
2*K3^2*K2*Vth2^2*Vth3^2*K1-K3^2*Vth3^4*K1^2-In^2*K1^2-2*In*K1^2*K3*Vth3^2-
K3^2*K2^2*Vth2^4+2*In*K1*K3*K2*Vth2^2+8*In*K1^2*K3*Vdd*Vth1+4*In*K1^2*K3*Vdd*Vth3-
4*In*K1^2*K3*Vth1*Vth3+4*K3^2*K1^2*Vdd*Vth3^3+8*K3^2*K1^2*Vdd*Vth1*Vth3^2-4*In*K1^2*K3*Vdd^2-
4*K3^2*Vdd^2*K1^2*Vth3^2-4*In*K1^2*K3*Vth1^2-4*K3^2*K1^2*Vth1*Vth3^3-4*K3^2*K1^2*Vth1^2*Vth3^2)*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)*(4*K4^2*Vdd^2*K1^2-
4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-4*In*K1^2*K3-
6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^2)^(1/2))^(1/3)-(96*K4^2*Vdd^2*K1^2-
96*Vth4^2*K1^2*K4^2+48*K4*K1^2*In-96*K3^2*K1^2*Vth3^2-96*In*K1^2*K3-
144*K3^2*K2^2*Vth2^2+288*K3*K1^2*Vdd*Vth3*K4-384*K3*K2*Vth2*K4*K1*Vth3-
192*K4*K1*Vth4*K3*K2*Vth2+384*K3*K2*Vth2*K4*K1*Vdd-
192*K3*K2*Vth2*K4*K1*Vth1+96*K3^2*Vdd*K1*Vth3*K2+384*K4*K1^2*Vth4*K3*Vdd-
96*K3^2*K1*Vth1*Vth3*K2+192*K3^2*K2*Vth2*K1*Vth3+576*K3*K1^2*Vdd*K4*Vth1-
384*K4*K1^2*Vth4*K3*Vth1+144*K3*K2*Vth2^2*K4*K1-192*K4*K1^2*Vth4*K3*Vth3-96*K3*K1^2*Vth1*Vth3*K4-
96*K1*K2*Vth2^2*K4^2+48*In*K1*K3*K2-
96*K3*K1^2*Vth1^2*K4+192*Vth1*Vth4*K1^2*K4^2+48*K3*Vth3^2*K1^2*K4-48*K3^2*K2*Vth3^2*K1-
480*K3*K1^2*Vdd^2*K4-192*K4^2*Vth1*Vdd*K1^2)*(-K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-
4*K3*K1^2*K4)*((-144*K4^2*Vdd^2*K1^2+144*Vth4^2*K1^2*K4^2-
72*K4*K1^2*In+144*K3^2*K1^2*Vth3^2+144*In*K1^2*K3+216*K3^2*K2^2*Vth2^2-
432*K3*K1^2*Vdd*Vth3*K4+576*K3*K2*Vth2*K4*K1*Vth3+288*K4*K1*Vth4*K3*K2*Vth2-
576*K3*K2*Vth2*K4*K1*Vdd+288*K3*K2*Vth2*K4*K1*Vth1-144*K3^2*Vdd*K1*Vth3*K2-
576*K4*K1^2*Vth4*K3*Vdd+144*K3^2*K1*Vth1*Vth3*K2-288*K3^2*K2*Vth2*K1*Vth3-
864*K3*K1^2*Vdd*K4*Vth1+576*K4*K1^2*Vth4*K3*Vth1-
216*K3*K2*Vth2^2*K4*K1+288*K4*K1^2*Vth4*K3*Vth3+144*K3*K1^2*Vth1*Vth3*K4+144*K1*K2*Vth2^2*K4^2-
72*In*K1*K3*K2+144*K3*K1^2*Vth1^2*K4-288*Vth1*Vth4*K1^2*K4^2-
72*K3*Vth3^2*K1^2*K4+72*K3^2*K2*Vth3^2*K1+720*K3*K1^2*Vdd^2*K4+288*K4^2*Vth1*Vdd*K1^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)*(8*K3*K2*Vth2^2*K4*K1*Vth3-4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-
4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-

```
8*K3*K1^2*Vdd^2*K4*Vth3-4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-
16*K3*K1^2*Vth1*K4*Vdd^2-8*K4*K1^2*Vth4*K3*Vdd^2-
4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)-
(1152*K3^2*Vdd*K1*Vth3*K2*Vth2^2-1152*K3^2*K1*Vth1*Vth3*K2*Vth2^2-576*K3^2*K2*Vth2^2*Vth3^2*K1-
288*K3^2*Vth3^4*K1^2-288*In^2*K1^2-576*In*K1^2*K3*Vth3^2-
288*K3^2*K2^2*Vth2^4+576*In*K1*K3*K2*Vth2^2+2304*In*K1^2*K3*Vdd*Vth1+1152*In*K1^2*K3*Vdd*Vth3-
1152*In*K1^2*K3*Vth1*Vth3+1152*K3^2*K1^2*Vdd*Vth3^3+2304*K3^2*K1^2*Vdd*Vth1*Vth3^2-
1152*In*K1^2*K3*Vdd^2-1152*K3^2*Vdd^2*K1^2*Vth3^2-1152*In*K1^2*K3*Vth1^2-1152*K3^2*K1^2*Vth1*Vth3^3-
1152*K3^2*K1^2*Vth1^2*Vth3^2)*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)*(-K3^2*K2^2+3*K4^2*K1^2-
4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+108*(8*K3*K2*Vth2^2*K4*K1*Vth3-
4*In*Vth4*K1^2*K4+4*K3^2*K2^2*Vth2^3-4*K3^2*K1^2*Vth3^3+8*K3*K1^2*Vdd*Vth1^2*K4-
8*K4*K1^2*Vth4*K3*Vth1^2-4*K4*K1^2*Vth4*K3*Vth3^2-8*K3*K1^2*Vdd^2*K4*Vth3-
4*In*K1*K3*K2*Vth2+4*K3*K1^2*Vth1*K4*Vth3^2+4*K3^2*K2*Vth2*Vth3^2*K1-16*K3*K1^2*Vth1*K4*Vdd^2-
8*K4*K1^2*Vth4*K3*Vdd^2-4*K3^2*K1*Vth3*K2*Vth2^2+8*K3^2*K1^2*Vdd*Vth3^2+4*Vth1*K4*K1^2*In-
4*In*K1^2*K3*Vth3+8*In*K1^2*K3*Vdd+8*K1^2*Vdd^3*K4*K3-8*K3^2*K1^2*Vth1*Vth3^2-
8*K4*K1^2*Vth4*K3*Vth1*Vth3+8*K3*K1^2*Vdd*Vth3*K4*Vth1+4*K3*K2*Vth2^2*K4*K1*Vth1+8*K4*K1^2*Vth4*K3*V
dd*Vth3-8*K3^2*K2*Vth2*Vdd*K1*Vth3+16*K4*K1^2*Vth4*K3*Vdd*Vth1-
8*K3*K2*Vth2^2*K4*K1*Vdd+8*K3^2*K2*Vth2*K1*Vth1*Vth3+4*K4*K1*Vth4*K3*K2*Vth2^2-8*In*K1^2*K3*Vth1)^2*(-
K3^2*K2^2+3*K4^2*K1^2-4*K1*K2*K4^2+6*K3*K2*K4*K1-4*K3*K1^2*K4)+(432*K3^2*Vdd*K1*Vth3*K2*Vth2^2-
432*K3^2*K1*Vth1*Vth3*K2*Vth2^2-216*K3^2*K2*Vth2^2*Vth3^2*K1-108*K3^2*Vth3^4*K1^2-108*In^2*K1^2-
216*In*K1^2*K3*Vth3^2-
108*K3^2*K2^2*Vth2^4+216*In*K1*K3*K2*Vth2^2+864*In*K1^2*K3*Vdd*Vth1+432*In*K1^2*K3*Vdd*Vth3-
432*In*K1^2*K3*Vth1*Vth3+432*K3^2*K1^2*Vdd*Vth3^3+864*K3^2*K1^2*Vdd*Vth1*Vth3^2-432*In*K1^2*K3*Vdd^2-
432*K3^2*Vdd^2*K1^2*Vth3^2-432*In*K1^2*K3*Vth1^2-432*K3^2*K1^2*Vth1*Vth3^3-432*K3^2*K1^2*Vth1^2*Vth3^2)*(-
4*K3^2*K1*Vth3*K2-4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*Vth4*K3)^2+8*(4*K4^2*Vdd^2*K1^2-4*Vth4^2*K1^2*K4^2+2*K4*K1^2*In-4*K3^2*K1^2*Vth3^2-
4*In*K1^2*K3-6*K3^2*K2^2*Vth2^2+12*K3*K1^2*Vdd*Vth3*K4-16*K3*K2*Vth2*K4*K1*Vth3-
8*K4*K1*Vth4*K3*K2*Vth2+16*K3*K2*Vth2*K4*K1*Vdd-
8*K3*K2*Vth2*K4*K1*Vth1+4*K3^2*Vdd*K1*Vth3*K2+16*K4*K1^2*Vth4*K3*Vdd-
4*K3^2*K1*Vth1*Vth3*K2+8*K3^2*K2*Vth2*K1*Vth3+24*K3*K1^2*Vdd*K4*Vth1-
16*K4*K1^2*Vth4*K3*Vth1+6*K3*K2*Vth2^2*K4*K1-8*K4*K1^2*Vth4*K3*Vth3-4*K3*K1^2*Vth1*Vth3*K4-
4*K1*K2*Vth2^2*K4^2+2*In*K1*K3*K2-4*K3*K1^2*Vth1^2*K4+8*Vth1*Vth4*K1^2*K4^2+2*K3*Vth3^2*K1^2*K4-
2*K3^2*K2*Vth3^2*K1-20*K3*K1^2*Vdd^2*K4-8*K4^2*Vth1*Vdd*K1^2)^3+12*3^(1/2)*(-(-4*K3^2*K1*Vth3*K2-
4*K3*K1^2*Vth3*K4-12*K3*K2*Vth2*K4*K1+4*K3^2*K2^2*Vth2-
8*K4^2*Vdd*K1^2+4*K4*K1*Vth4*K3*K2+16*K3*K1^2*Vdd*K4+4*Vth1*K1^2*K4^2+8*K1*K2*Vth2*K4^2-
8*K3*K1^2*Vth1*K4+4*K3*K2*K4*K1*Vth1+4*Vth4*K1^2*K4^2-8*K3*K2*K4*K1*Vdd+8*K3*K2*K4*K1*Vth3-
8*K4*K1^2*V... Output truncated.  Text exceeds maximum line length of 25,000 characters for Command Window display.
```

APPENDIX B

The matlab program used in Chapter V:

MATLAB CODE FOR $I_{critical}$:

```
function V=numerical_SRAM_left(x1o,x2o,Ico)
clc;
tol=1e-6; k=0;
%  x1o=0;x2o=1;Ico=0;
   x1=x1o; x2=x2o; In1=Ico; V_pre=[x1o;x2o;Ico]; error=1;
   while (error>tol)
   %for n=1:10
      k=k+1
      V=(V_pre-J(x1,x2)^(-1)*[f(x1,x2)+(1/10.2e-15)*In1;g(x1,x2);df_x1(x1,x2)*dg_x2(x1,x2)-
df_x2(x1,x2)*dg_x1(x1,x2)])
      error=norm(V-V_pre,2)
      V_pre=V;
      x1=V(1);x2=V(2);In1=V(3);
   end
Vdd=1*(1);
Vth2=0.18*(1); Vth4 = 0.18*(1);
Vth1=0.18*(1); Vth3 = 0.18*(1);
 plot(V(1),V(2),'o'),axis([0 Vdd 0 Vdd]),xlabel('V1'),ylabel('V2'),title('CASE I Bifurcation point');
   hold on
%% plot regions
increment = 0.01;
x=0:increment:Vdd;
if (Vth1>Vth4)
   plot (x,x-Vth4,'g')
   plot (x,x-Vth1,'g')
else
   plot (x,x-Vth1,'g')
   plot (x,x-Vth4,'g')
end
if (Vth2>Vth3)
   plot (x,x+Vth3,'g')
```

```
    plot (x,x+Vth2,'g')
else
    plot (x,x+Vth2,'g')
    plot (x,x+Vth3,'g')
end
plot (x,Vth2*ones(1,(Vdd-0)/increment+1),'m')
plot (x,(Vdd-Vth1)*ones(1,(Vdd-0)/increment+1),'m')
plot (Vth4*ones(1,(Vdd-0)/increment+1),x,'m')
plot ((Vdd-Vth3)*ones(1,(Vdd-0)/increment+1),x,'m')
hold off
end
%% J matrix
function J_matrix=J(x1o,x2o)
dh_x1=df_x1(x1o,x2o)*d2g_x1_x2(x1o,x2o)+d2f_x1_x1(x1o,x2o)*dg_x2(x1o,x2o)-
df_x2(x1o,x2o)*d2g_x1_x1(x1o,x2o)-d2f_x1_x2(x1o,x2o)*dg_x1(x1o,x2o);
dh_x2=df_x1(x1o,x2o)*d2g_x2_x2(x1o,x2o)+d2f_x1_x2(x1o,x2o)*dg_x2(x1o,x2o)-
df_x2(x1o,x2o)*d2g_x1_x2(x1o,x2o)-d2f_x2_x2(x1o,x2o)*dg_x1(x1o,x2o);
J_matrix=[df_x1(x1o,x2o),df_x2(x1o,x2o),1/10.2e-15;
          dg_x1(x1o,x2o),dg_x2(x1o,x2o),0;
          dh_x1,dh_x2,0];
end
%% second order differentiation
function d2fx1x1=d2f_x1_x1(x1o, x2o)
 delt=(1e-8)*100;
% d2fx1x1=(df_x1(x1o+delt,x2o)-df_x1(x1o,x2o))/delt;
 d2fx1x1=(f(x1o+delt,x2o)-2*f(x1o,x2o)+f(x1o-delt,x2o))/delt^2;
end
function d2fx2x2=d2f_x2_x2(x1o, x2o)
 delt=(1e-8)*100;
 %d2fx2x2=(df_x2(x1o,x2o+delt)-df_x2(x1o,x2o))/delt;
 d2fx2x2=(f(x1o,x2o+delt)-2*f(x1o,x2o)+f(x1o,x2o-delt))/delt^2;
end
function d2fx1x2=d2f_x1_x2(x1o, x2o)
 delt=(1e-8)*100;
 %d2fx1x2=(df_x1(x1o,x2o+delt)-df_x1(x1o,x2o))/delt;
```

```
 d2fx1x2=(f(x1o+delt,x2o+delt)-f(x1o+delt,x2o-delt)-f(x1o-delt,x2o+delt)+f(x1o-delt,x2o-
delt))/(4*delt^2);
end
function d2gx1x1=d2g_x1_x1(x1o, x2o)
 delt=(1e-8)*100;
% d2gx1x1=(df_x1(x1o+delt,x2o)-df_x1(x1o,x2o))/delt;
 d2gx1x1=(g(x1o+delt,x2o)-2*g(x1o,x2o)+g(x1o-delt,x2o))/delt^2;
end
function d2gx2x2=d2g_x2_x2(x1o, x2o)
 delt=(1e-8)*100;
% d2gx2x2=(df_x2(x1o,x2o+delt)-df_x2(x1o,x2o))/delt;
 d2gx2x2=(g(x1o,x2o+delt)-2*g(x1o,x2o)+g(x1o,x2o-delt))/delt^2;
end
function d2gx1x2=d2g_x1_x2(x1o, x2o)
 delt=(1e-8)*100;
% d2gx1x2=(df_x1(x1o,x2o+delt)-df_x1(x1o,x2o))/delt;
 d2gx1x2=(g(x1o+delt,x2o+delt)-g(x1o+delt,x2o-delt)-g(x1o-delt,x2o+delt)+g(x1o-delt,x2o-
delt))/(4*delt^2);
end
%% first order differentiation
function dfx1=df_x1(x1o, x2o)
delt=1e-8;
dfx1=(f(x1o+delt,x2o)-f(x1o,x2o))/delt;
end
function dfx2=df_x2(x1o, x2o)
delt=1e-8;
dfx2=(f(x1o,x2o+delt)-f(x1o,x2o))/delt;
end
function dgx1=dg_x1(x1o, x2o)
delt=1e-8;
dgx1=(g(x1o+delt,x2o)-g(x1o,x2o))/delt;
end
function dgx2=dg_x2(x1o, x2o)
delt=1e-8;
dgx2=(g(x1o,x2o+delt)-g(x1o,x2o))/delt;
```

```
end
%% SRAM functions
function f_=f(x1,x2)
  [f_]=func(x1,x2);
end
function g_=g(x1,x2)
  [f_,g_]=func(x1,x2);
end
function [f,g]=func(x1,x2)
A=100;
C1=10.2e-15;
C2=C1;
Vdd=1*(1);
Vth2=0.18*(1); Vth4 = 0.18*(1);
Vth1=0.18*(1); Vth3 = 0.18*(1);
Kp1=(0.0005859472212)*(1); Kp3=(0.0005859472212)*(1);
Kn2=(0.00123172235)*(1);   Kn4=(0.00123172235)*(1);
f= (Kp1/(C1*A^2))*(log(1+exp(A*(Vdd-x2-(Vth1))))^2 - log(1+exp(A*(x1-x2-(Vth1))))^2) -
(Kn2/(C1*A^2))*(log(1+exp(A*(x2-(Vth2))))^2 - log(1+exp(A*(x2-x1-(Vth2))))^2); % + In1/C1;
g= (Kp3/(C2*A^2))*(log(1+exp(A*(Vdd-x1-(Vth3))))^2 - log(1+exp(A*(x2-x1-(Vth3))))^2) -
(Kn4/(C2*A^2))*(log(1+exp(A*(x1-(Vth4))))^2 - log(1+exp(A*(x1-x2-(Vth4))))^2); % + In2/C2;
end

function V=numerical_SRAM_right(x1o,x2o,Ico)
%clc;
format long
tol=1e-6;k=0;
%  x1o=0;x2o=1;Ico=0;
    x1=x1o; x2=x2o; In2=Ico; V_pre=[x1o;x2o;Ico]; error=1;
    while (error>tol)
    %for n=1:10
        k=k+1
        V=(V_pre-J(x1,x2)^(-1)*[f(x1,x2);g(x1,x2)+(1/10.2e-15)*In2;df_x1(x1,x2)*dg_x2(x1,x2)-
df_x2(x1,x2)*dg_x1(x1,x2)])
        error=norm(V-V_pre,2)
```

```
        V_pre=V;
        x1=V(1);x2=V(2);In2=V(3);
    end
    Vdd=1*(1);
Vth2=0.18*(1); Vth4 = 0.18*(1);
Vth1=0.18*(1); Vth3 = 0.18*(1);
 plot(V(1),V(2),'o'),axis([0 Vdd 0 Vdd]),xlabel('V1'),ylabel('V2'),title('CASE I Bifurcation point');
    hold on
%% plot regions
increment = 0.01;
x=0:increment:Vdd;
if (Vth1>Vth4)
    plot (x,x-Vth4,'g')
    plot (x,x-Vth1,'g')
else
    plot (x,x-Vth1,'g')
    plot (x,x-Vth4,'g')
end
if (Vth2>Vth3)
    plot (x,x+Vth3,'g')
    plot (x,x+Vth2,'g')
else
    plot (x,x+Vth2,'g')
    plot (x,x+Vth3,'g')
end
plot (x,Vth2*ones(1,(Vdd-0)/increment+1),'m')
plot (x,(Vdd-Vth1)*ones(1,(Vdd-0)/increment+1),'m')
plot (Vth4*ones(1,(Vdd-0)/increment+1),x,'m')
plot ((Vdd-Vth3)*ones(1,(Vdd-0)/increment+1),x,'m')
hold off
end
%% J matrix
function J_matrix=J(x1o,x2o)
dh_x1=df_x1(x1o,x2o)*d2g_x1_x2(x1o,x2o)+d2f_x1_x1(x1o,x2o)*dg_x2(x1o,x2o)-
df_x2(x1o,x2o)*d2g_x1_x1(x1o,x2o)-d2f_x1_x2(x1o,x2o)*dg_x1(x1o,x2o);
```

```
dh_x2=df_x1(x1o,x2o)*d2g_x2_x2(x1o,x2o)+d2f_x1_x2(x1o,x2o)*dg_x2(x1o,x2o)-
df_x2(x1o,x2o)*d2g_x1_x2(x1o,x2o)-d2f_x2_x2(x1o,x2o)*dg_x1(x1o,x2o);
J_matrix=[df_x1(x1o,x2o),df_x2(x1o,x2o),0;
        dg_x1(x1o,x2o),dg_x2(x1o,x2o),1/10.2e-15;
        dh_x1,dh_x2,0];
end
%% second order differentiation
function d2fx1x1=d2f_x1_x1(x1o, x2o)
 delt=(1e-8)*100;
% d2fx1x1=(df_x1(x1o+delt,x2o)-df_x1(x1o,x2o))/delt;
 d2fx1x1=(f(x1o+delt,x2o)-2*f(x1o,x2o)+f(x1o-delt,x2o))/delt^2;
end
function d2fx2x2=d2f_x2_x2(x1o, x2o)
 delt=(1e-8)*100;
 %d2fx2x2=(df_x2(x1o,x2o+delt)-df_x2(x1o,x2o))/delt;
 d2fx2x2=(f(x1o,x2o+delt)-2*f(x1o,x2o)+f(x1o,x2o-delt))/delt^2;
end
function d2fx1x2=d2f_x1_x2(x1o, x2o)
 delt=(1e-8)*100;
 %d2fx1x2=(df_x1(x1o,x2o+delt)-df_x1(x1o,x2o))/delt;
 d2fx1x2=(f(x1o+delt,x2o+delt)-f(x1o+delt,x2o-delt)-f(x1o-delt,x2o+delt)+f(x1o-delt,x2o-
delt))/(4*delt^2);
end
function d2gx1x1=d2g_x1_x1(x1o, x2o)
 delt=(1e-8)*100;
% d2gx1x1=(df_x1(x1o+delt,x2o)-df_x1(x1o,x2o))/delt;
 d2gx1x1=(g(x1o+delt,x2o)-2*g(x1o,x2o)+g(x1o-delt,x2o))/delt^2;
end
function d2gx2x2=d2g_x2_x2(x1o, x2o)
 delt=(1e-8)*100;
% d2gx2x2=(df_x2(x1o,x2o+delt)-df_x2(x1o,x2o))/delt;
 d2gx2x2=(g(x1o,x2o+delt)-2*g(x1o,x2o)+g(x1o,x2o-delt))/delt^2;
end
function d2gx1x2=d2g_x1_x2(x1o, x2o)
 delt=(1e-8)*100;
```

```
% d2gx1x2=(df_x1(x1o,x2o+delt)-df_x1(x1o,x2o))/delt;
 d2gx1x2=(g(x1o+delt,x2o+delt)-g(x1o+delt,x2o-delt)-g(x1o-delt,x2o+delt)+g(x1o-delt,x2o-
delt))/(4*delt^2);
end
%% first order differentiation
function dfx1=df_x1(x1o, x2o)
delt=1e-8;
dfx1=(f(x1o+delt,x2o)-f(x1o,x2o))/delt;
end
function dfx2=df_x2(x1o, x2o)
delt=1e-8;
dfx2=(f(x1o,x2o+delt)-f(x1o,x2o))/delt;
end
function dgx1=dg_x1(x1o, x2o)
delt=1e-8;
dgx1=(g(x1o+delt,x2o)-g(x1o,x2o))/delt;
end
function dgx2=dg_x2(x1o, x2o)
delt=1e-8;
dgx2=(g(x1o,x2o+delt)-g(x1o,x2o))/delt;
end
%% SRAM functions
function f_=f(x1,x2)
  [f_]=func(x1,x2);
end
function g_=g(x1,x2)
  [f_,g_]=func(x1,x2);
end
function [f,g]=func(x1,x2)
A=100;
C1=10.2e-15;
C2=C1;
Vdd=1*(1);
Vth2=0.18*(1); Vth4 = 0.18*(1);
Vth1=0.18*(1); Vth3 = 0.18*(1);
```

```
Kp1=(0.0005859472212)*(1); Kp3=(0.0005859472212)*(1);
Kn2=(0.00123172235)*(1);   Kn4=(0.00123172235)*(1);
f= (Kp1/(C1*A^2))*(log(1+exp(A*(Vdd-x2-(Vth1))))^2 - log(1+exp(A*(x1-x2-(Vth1))))^2) -
(Kn2/(C1*A^2))*(log(1+exp(A*(x2-(Vth2))))^2 - log(1+exp(A*(x2-x1-(Vth2))))^2); % + In1/C1;
g= (Kp3/(C2*A^2))*(log(1+exp(A*(Vdd-x1-(Vth3))))^2 - log(1+exp(A*(x2-x1-(Vth3))))^2) -
(Kn4/(C2*A^2))*(log(1+exp(A*(x1-(Vth4))))^2 - log(1+exp(A*(x1-x2-(Vth4))))^2); % + In2/C2;
end
```

MATLAB CODE FOR $T_{critical}$:

```
clc
clear all;
syms x1 x2 Inoise1 Inoise2 Vth1 Vth2 Vth3 Vth4 Vb1 Vb2 Vb3 Vb4
%%%%%%%%%%%%%
A=100;
C1=10e-15*(1); C2=10e-15;
Vdd=1*(1);
n=0;
m=0;
Kp1=5.859472212e-4*(1-n);
Kn2=1.23172235e-3*(1+n);
Kp3=5.859472212e-4*(1+n);
Kn4=1.23172235e-3*(1-n);
Vth1=0.18*(1-m);
Vth2=0.18*(1-m);
Vth3=0.18*(1+m);
Vth4=0.18*(1+m);
isright=1;
f1 = (Kp1/(C1*A^2))*(log(1+exp(A*(Vdd-x2-(Vth1))))^2 - log(1+exp(A*(x1-x2-(Vth1))))^2) -
(Kn2/(C1*A^2))*(log(1+exp(A*(x2-(Vth2))))^2 - log(1+exp(A*(x2-x1-(Vth2))))^2) ;
f2 = (Kp3/(C2*A^2))*(log(1+exp(A*(Vdd-x1-(Vth3))))^2 - log(1+exp(A*(x2-x1-(Vth3))))^2) -
(Kn4/(C2*A^2))*(log(1+exp(A*(x1-(Vth4))))^2 - log(1+exp(A*(x1-x2-(Vth4))))^2) ;
%Jacobian
J=([diff(f1,x1),diff(f1,x2);diff(f2,x1),diff(f2,x2)]);
J_1=inv(J);
%find saddle point
epsilon=1e-4;
x1=Vdd/2;x2=Vdd/2; xm=[x1;x2];
while norm(eval([f1;f2]))>epsilon
   xm=xm-eval(J_1)*eval([f1;f2]);
   x1=xm(1,1);x2=xm(2,1);
end
figure(3)
plot (xm(1),xm(2),'o'),axis([0 Vdd 0 Vdd]), grid,xlabel('x1'),ylabel('x2')  %plot saddle ponit
```

```
title('SRAM')
hold on
%conditions
tol_up=1e-2;
tol_bt=1e-3;
step=1e-4;
stepdw=50;
stepup=100;
%top part
x1=xm(1)+step; %initial (+) purturbation
x2=xm(2)+step;
k=1;
plot(x1,x2,'k')
while (x1<Vdd || x2<Vdd)
x1_next=x1+step*-1*eval(f1);
x2_next=x2+step*-1*eval(f2);
error=sqrt((x1_next-x1)^2+(x2_next-x2)^2);
   while (error>tol_up || error<tol_bt)
    if error>tol_up
       step=step/stepdw;
    else
        step=step*stepup;
    end
    x1_next=x1+step*-1*eval(f1);
    x2_next=x2+step*-1*eval(f2);
    error=sqrt((x1_next-x1)^2+(x2_next-x2)^2);
   end
 x1=x1_next; x2=x2_next;
 x1_top(k)=x1; x2_top(k)=x2;
 k=k+1;
end
plot(x1_top,x2_top,'k')
coef=polyfit(x1_top,x2_top,3);
poly=@(x) coef(1)*x^3+coef(2)*x^2+coef(3)*x+coef(4);
syms x1 x2
```

```
Ic=4.96038283934e-4;
if(isright)
    f1 = (Kp1/(C1*A^2))*(log(1+exp(A*(Vdd-x2-(Vth1))))^2 - log(1+exp(A*(x1-x2-(Vth1))))^2) -
(Kn2/(C1*A^2))*(log(1+exp(A*(x2-(Vth2))))^2 - log(1+exp(A*(x2-x1-(Vth2))))^2) ;
    f2 = (Kp3/(C2*A^2))*(log(1+exp(A*(Vdd-x1-(Vth3))))^2 - log(1+exp(A*(x2-x1-(Vth3))))^2) -
(Kn4/(C2*A^2))*(log(1+exp(A*(x1-(Vth4))))^2 - log(1+exp(A*(x1-x2-(Vth4))))^2)+Ic/C2 ;
else
    f1 = (Kp1/(C1*A^2))*(log(1+exp(A*(Vdd-x2-(Vth1))))^2 - log(1+exp(A*(x1-x2-(Vth1))))^2) -
(Kn2/(C1*A^2))*(log(1+exp(A*(x2-(Vth2))))^2 - log(1+exp(A*(x2-x1-(Vth2))))^2)+Ic/C1 ;
    f2 = (Kp3/(C2*A^2))*(log(1+exp(A*(Vdd-x1-(Vth3))))^2 - log(1+exp(A*(x2-x1-(Vth3))))^2) -
(Kn4/(C2*A^2))*(log(1+exp(A*(x1-(Vth4))))^2 - log(1+exp(A*(x1-x2-(Vth4))))^2) ;
end
%% plotting
x1o=Vdd;x2o=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0;
x1=x1o;
x2=x2o;
%pre-set initials
k=1; %index
x1t(k)=x1o;x2t(k)=x2o;
tt(k)=t; Inoise(k)=eval(Inoise2);
%pre-set conditions
tol_up=1e-2;
tol_bt=1e-3;
step_pre=1e-14;
step=step_pre;
stepdw=30;
stepup=100;
stop=0;
cont=1;
while (cont)
    t_pre=t;
    t=t_pre+step;
    x1_next=x1+step*eval(f1);
```

```
    x2_next=x2+step*eval(f2);
    error=sqrt((x1_next-x1)^2+(x2_next-x2)^2);
    while (error>tol_up  || error<tol_bt)
        if error>tol_up
            step=step/stepdw;
        else
            step=step*stepup;
        end
        t=t_pre+step;
        x1_next=x1+step*eval(f1);
        x2_next=x2+step*eval(f2);
        error=sqrt((x1_next-x1)^2+(x2_next-x2)^2);
    end
    x1=x1_next;
    x2=x2_next;
    k=k+1; %the following is putting stuff into array
    tt(k)=t;
    Inoise(k)=eval(Inoise2);
    x1t(k)=x1; x2t(k)=x2; step=step_pre;%reset to initial step sizE
   if (x2>poly(x1))
       cont=0;
   end
end
plot (x1t,x2t,'-')
hold off
TIMEdiag = figure ('Name','Time Diagram','Numbertitle','off','Position',[0 25 550 400]);
plot(tt,x1t,'b'), grid, xlabel('time'), ylabel('x1 and x2');
hold on
plot(tt,x2t,'r')
hold off
tc=tt(length(tt))
```

VITA

Yenpo Ho was born in Taipei, Taiwan. He came to the United States of America in 1998. In 2002, he received his B.S. degree in Electrical Engineering from the University of Arizona. He received his M.S. in Electrical Engineering at Texas A&M University at College Station, in May 2008. His research interest is in the area of electronic circuits and control systems. He is a member of Eta Kappa Nu and Tau Beta Pi honor society.

Mr. Ho can be reached at 301 Wisenbaker Research Center (WERC) room 320H, College Station, TX 77843-3126. His email is ypho@neo.tamu.edu.