IMPACT OF EXTENSION LATERAL DOPING ABRUPTNESS ON DEEP SUBMICRON DEVICE PERFORMANCE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Michael Y. Kwong
August 2002

Copyright by Michael Y. Kwong 2002
All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Robert W. Dutton
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Bruce A. Wooley
(Electrical Engineering)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Peter Griffin
(Electrical Engineering)

Approved for the University Committee on Graduate Studies:

Preface

Device scaling is directly responsible for Moore's law and has enabled tremendous improvements in MOS (Metal-Oxide-Semiconductor) device performance. As device dimensions shrink, the channel resistance decreases, which in turn allows faster circuit operation. Microprocessor chips operating at 2GHz or higher clock speeds are now available. However, as the intrinsic device continues to improve, parasitic components such as the series resistance in the source/drain region start to limit device performance. Understanding and controlling these parasitic components, through proper design of the device, are therefore essential.

TCAD (Technology Computer Aided Design) can be a tremendous tool that could allow us to improve our understanding of various device design parameters on device performance. It allows the conduction of detailed simulation studies whose experimental counterparts would be prohibitively expensive. Furthermore, it allows the probing of various internal quantities that are not available experimentally.

This thesis presents the results of a thorough study, made possible by TCAD tools, of the impact of lateral abruptness and gate-extension overlap on device performance. Lateral abruptness is considered a key device parameter that need to be controlled for deep submicron devices, making this an important parameter to understand.

The study begins by making several simplifying assumptions. The design of modern deep submicron semiconductor device is very complex, making it difficult to understand. Simplifying the device design in the study allows the isolation of the features of interest, and revealed several phenomena that would not have (and have not) been noticed otherwise.

While the results and conclusions from this study are the main focus of this thesis, considerable attention is paid to the methodology, algorithms and software that enabled the successful execution of the study and processing of the simulation results. Many of them can be useful for studying other device parameters. This could be one of the major contributions of this work.

Acknowledgments

The graduate school experience is one of the most amazing, frustrating, depressing, yet ultimately rewarding journeys I have ever undertaken. There are many people whose help and support has been crucial, and whom I would like to take this opportunity to thank.

I would like to thank my advisor, Professor Bob Dutton, for giving me the chance to get a taste of research as an undergraduate, for accepting me into his research group, and for his patient guidance throughout my graduate years. Not only did I learn much about how to do research, I also learnt a great deal about myself in the process, which I truly believe will serve me well for the rest of my life.

I would like to thank Dr. Peter Griffin for the many discussions we had about my research, for the numerous suggestions and feedback, for his detailed comments on this thesis and other papers that I have written, and for all his mentoring efforts.

I would like to thank my associate advisor, Professor Bruce Wooley, for his advice and moral support, as well as for his spending time reviewing my thesis. I would also like to thank Professor Krishna Saraswat for his advice and for being on my defense committee; and Professor Dwight Nishimura for agreeing to chair my defense committee.

I would like to thank the Plummer group, and Professor Jim Plummer in particular, for allowing be to participate in their group meetings, and for offering valuable comments, suggestions and critiques of my work.

I would also like to thank Dr. Dan Connelly and Dr. Michael Duanne for many great discussions, especially for Dan's work on I_{on}-I_{off} as a metric for comparing deep submicron technology, which this thesis utilizes extensively.

Reza: without your encouragements, suggestions and collaboration, I might not have been able to find my "research way". Thank you with all my heart.

Ken, Nathan and Edward: thanks for lending an ear when times were tough. Having group mates such as yourselves makes graduate student life much more tolerable and much less lonely. Chang-hoon: thanks for sharing yours insights and ideas; you have no idea how much I appreciate your help. And to the entire Dutton research group: thanks for helping me grow and learn the past few years.

Fely, Maria, Miho and Dan: thanks for all the wonderful support that you provided us, without which, we would all have taken years longer to finish! Zhiping: thanks for spending so much of your time with me, and for seeing me through the tortuous process of finding the right direction.

Tamara: TA extraordinaire, your enthusiasm is contagious, your passion for teaching is inspiring, your warmth and encouragements I will never forget. May the wild west saloon continue to be a refuge and source of wisdom for generations of students to come.

Mar: thanks for the time you spent with me to generate research ideas. Danny, Dave, Linda: thanks for being there and for all the support through the years! Jef: I feel so blessed to have met you. Thank you for all the discussions, technical or otherwise, and for the support you have given me.

To each and every one that I have met during my time at Stanford: thanks for teaching me so much about myself and life. Hopefully I have been able to return a little bit of the favor, in my own way.

Last but not least, I would like to thank my family for being supportive, understanding, patient, and for having shaped me into who I am.

Contents

Preface iv

Acknowledgments vi

1 Introduction 1

1.1 Device Scaling and Moore's Law 1

1.2 ITRS Roadmap 3

1.3 Methodology of this Work 4

1.4 Scope and Organization 7

2 Challenges for Deep Sub-micron Device Design 9

2.1 Device Design Parameters and Device Scaling 10

2.1.1 Key Device Design Parameters 10

2.1.2 Definition of Threshold Voltage 11

2.1.3 First Order Device Scaling Theory 16

2.2 Second Order Effects in the Deep Sub-micron Regime 16

2.2.1 Short Channel Effects 17

2.2.2 Reverse Short Channel Effects 21

2.2.3 Source/Drain Resistance 22

2.2.4 Quantum Mechanical Effects 25
2.3 Device Design in the Deep Sub-micron Regime 29
2.3.1 Channel Engineering 29
2.3.2 Source/Drain Engineering 31
2.4 Technology Computer Aided Design 34
2.4.1 Traditional Device Design Process 34
2.4.2 Computer Simulations and Device Design 36
2.5 Challenges facing Technology Computer Aided Design 38
2.5.1 Metrology 38
2.5.2 Physical Models 39
2.6 Summary 42

3 Study of Lateral Abruptness 44

3.1 Introduction 44
3.2 Methodology 45
3.2.1 Simulation Details 45
3.2.2 Metric for Comparing Device Technologies 46
3.3 Series Resistance 48
3.4 Threshold Voltage Roll-Off 50
3.4.1 Counter-doping 52
3.4.2 Charge Sharing 54
3.5 On-currents and off-currents 56
3.5.1 Conventional I_{on}-I_{off} characteristics 56
3.5.2 Nominal I_{on}-Nominal I_{off} Plots 58
3.5.3 Supernominal I_{on}-Subnominal I_{off} Plots 62
3.6 Revisiting Key Simulation Assumptions 64

3.6.1 Halo Doping 64
3.6.2 Source/Drain Doping Description 68
3.6.3 Factors affecting Sensitivity to Lateral Abruptness 70
3.7 Summary 73

4 Resistance Calculations 76

4.1 Introduction 76
4.2 Nomenclature 77
4.3 Analytical Resistance Calculations 78
4.4 Vertical Strip Calculation Method 81
4.4.1 Ng and Lynch's Analytical Model 81
4.4.2 Limits of the Method 81
4.5 Calculation based on Equipotential Lines 83
4.5.1 Resistances in the Bulk 83
4.5.2 Contact Resistance 87
4.5.3 Implementation 88
4.6 Calculation Using the Quasi-Fermi Level at the Surface 88
4.7 Results 89
4.7.1 Comparison between Resistance Calculation Methods 89
4.7.2 Resistance Components and Lateral Abruptness 92
4.7.3 Gate-Bias Dependence of Resistance Components 93
4.8 Resistance Calculation versus Resistance Extraction 96
4.8.1 The Shift-and-Ratio Method 96
4.8.2 Shift-And-Ratio versus Physical Resistances 97
4.8.3 Extraction of Gate-Bias Dependent Source/Drain Resistance . 99
4.8.4 Limits of the Shift-And-Ratio method 100

4.8.5 Discussion 104
4.9 Conclusions 105

5 Software Implementation **107**
5.1 Device Simulation Template 108
5.1.1 Grid Generation 111
5.2 Job Farming 114
5.3 I-V Database 117
5.4 Resistance Calculations 119
5.4.1 Design Patterns 119
5.4.2 Mesh Object Class Hierarchy 119
5.4.3 Proxy Design Pattern 123
5.4.4 Builder Design Pattern 124
5.4.5 Iterator Design Pattern 125
5.5 Software Engineering for TCAD Applications 127
5.5.1 Goal of Software Engineering 127
5.5.2 Object-Oriented Design 128
5.5.3 Software Development Process 129
5.6 Summary 130

6 Conclusion **132**
6.1 Contributions 135
6.2 Future Work 136

A Grid Sensitivity Study **138**
A.1 Introduction 138
A.2 Simulation Structure 139

A.3 Resolving Junction Gradient for Deep Sub-micron Device 141
A.4 Grid Dependence of Drain Current and Choice of Physical Models . . 143
A.4.1 Quantum Mechanical Effects: Van Dort Model 145
A.4.2 Grid Sensitivity of Several Mobility Models 146
A.4.3 Polydepletion Effects and Grid Dependence 147
A.5 Grid Dependence of Other Electrical Quantities 149
A.5.1 Grid Dependence of Potential Simulation 151
A.5.2 External Resistance Calculations 152
A.6 Guidelines on Grid Densities 161
A.7 Conclusion 169
A.8 Useful Formulae for Generating Grids 170

B Device Structure Template 172

B.1 Description of Template Files 172
B.2 Partial Code Listings 173

Bibliography 202

List of Tables

1.1 International Technology Roadmap for Semiconductors 1999 Edition. ***italics*** indicate no known solutions. 4

2.1 Useful equations for calculating basic electrical quantities for a transistor. 11

2.2 Constant Field Scaling [84] 17

2.3 Source/drain resistance calculated with several different mobility models for the same device 41

3.1 Parameters for Simulated Devices, chosen according to the 2008 technology node on the 1999 ITRS roadmap 46

3.2 Resistive components for devices with $L_{gate} = 50$ nm and various lateral source/drain abruptness. 49

3.3 (a) I_{on} (in units of $10^{-4}A/\mu m$) for devices with $I_{\text{off}} = 10^{-7}A$ (or $10^{-10}A$ for $L_g = 70nm$) and different abruptness. (b) Percentage deviation of I_{on} from the device with the best case abruptness. Gate-extension overlap = 14 nm. 61

3.4 Percentage deviations of $I_{on,target}$ at $I_{off} = 10^{-7} A/\mu m$ from the devices on the "best case" curve with different abruptnesses. Three different mobility models are explored in the simulations. Oxide thickness is 1.5 nm and channel doping is 4.8×10^{18} for all the devices shown. 71

3.5 Percentage deviation of I_{on} for target I_{off} of $10^{-7} A$ the best case (for a given t_{ox}) for devices with various abruptnesses abruptness. Channel doping is 4.8×10^{18} for all devices shown. Simulated with GMCMOB. Doping profiles S/D1 and S/D2 are as shown in Figure 3.15. 71

3.6 (a) Resistance in the overlap region for devices with two different oxide thickness. (b) Percentage difference from 1.9 nm/dec case. Gate Length is 50 nm. Simulated using GMCMOB. 72

4.1 Resistive components (in $\Omega/\mu m$) calculated using three different methods: quasi-Fermi level at Si/SiO_2 interface ($\phi_n(x)$); integration based on equipotential lines (Equi-ϕ); and integration based on vertical strips (Vertical). Note the error in $R_{overlap}$ for the vertical case. 90

4.2 $R_{overlap}$ (in $\Omega/\mu m$) calculated for devices with different lateral source/drain abruptnesses using three different methods: quasi-Fermi level at Si/SiO_2 interface ($\phi_n(x)$); integration based on equipotential lines (Equi-ϕ); and integration based on vertical strips (Vertical). 92

4.3 Comparing resistive components, in $\Omega/\mu m$, for devices with different lateral doping abruptness in the source/drain extension region. 93

4.4 Effective channel lengths and series resistances extracted using the shift-and-ratio method, for devices with gate length of 50 nm and different lateral source/drain extension abruptness. V_{gs} extraction range used in the shift-and-ratio method is from 0.8 V to 1.5 V. 97

A.1 Simulated I_d for device with $L_g = 0.35\ \mu m$ and lateral junction slope of 30 nm/dec 143
A.2 Drain Current at $V_g = 1.5V$ and $V_d = 0.05V$. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{x3} = 0.4\mathring{A}$ and $h_{y1} = 1.4\mathring{A}$ 145
A.3 Details of the Devices in Figures A.17 to A.30 162

List of Figures

1.1 Device Scaling (a) CPU clock speed and MOS device gate lengths versus year (b) MOS device saturation current versus technology generation (represented by the minimum device length) 2

1.2 Flow chart indicating the approach of this thesis. Using software described in Chapter 5, simulations of numerous MOS devices are performed. The results are processed as in Chapter 4, which provides insight on the analytical resistance calculations reported by Ng, et al [57], the shift-and-ratio method and the scaling limits concerning lateral doping abruptness in the source/drain extension. Grid sensitivity studies are conducted to ensure simulation accuracy. 5

2.1 (a) Threshold voltage of an ideal switch (b) Maximum g_m definition of threshold voltage. 12

2.2 Plot of threshold voltage (V_t) versus channel length (L) for a typical MOS device technology 19

2.3 Charge sharing model for explaining short channel effects (Yau [87]) 19

2.4 Current versus gate voltage plot at low and high drain biases, demonstrating drain induced barrier lowering 20

2.5 The energy band diagram at the source end of a half NMOS device with and without an applied drain bias. 21
2.6 Schematic diagram showing a non-uniform lateral doping distribution that can be used to explain the reverse short channel effects (a) Schematic of the device (b) Doping profile along the line AA' 22
2.7 Intrinsic resistance R_{ch} and extrinsic resistance R_{sd} versus technology generation (indicated by channel length) 24
2.8 Potential well at the Si/SiO_2 interface, charge quantization and band-gap widening 26
2.9 Capacitance-Voltage plot for a device with gate oxide thickness of 2.1 nm modeled with classical models and the quantum mechanical Yu model [16]. Note the substantial difference between the two curves with respect to inversion/accumulation capacitance and threshold voltage. Also shown are models based on effective oxide thickness to fit the accumulation capacitance (2.23 nm) and threshold voltage (2.8 nm) respectively. Both cannot be fitted to at the same time using classical models. 27
2.10 An "ideal" Low-High (Retrograde) Doping Profile, with a channel doping of N_s for the first y_s μm, after which the doping rises to N_a in the substrate 30
2.11 Schematic diagram showing a halo doping (a) Device schematic showing the source/drain extension region (b) Doping profile along the line AA' 31

2.12 Lateral source/drain abruptness requirements according to the ITRS [1], based on threshold voltage roll-off and series resistance considerations respectively. 34
2.13 Technology Design Flow without Simulation 35
2.14 TCAD Based Technology Design Flow 37

3.1 Schematic of a half MOS device, together with the major resistive components 48
3.2 Plot of threshold voltage versus L_g (threshold roll-off characteristics) 51
3.3 Plot of threshold voltage in the linear region versus lateral abruptness of the source/drain extensions for devices with gate lengths ranging from 45 nm to 80 nm 52
3.4 (a) Donor doping profile plots along the $Si-SiO_2$ interface, for extensions with lateral abruptness ranging from 13 nm/dec to 4.5 nm/dec (b) Net doping plots along the $Si-SiO_2$ interface, showing counter-doping clearly 53
3.5 Device Schematic showing two source/drain junctions with different lateral abruptness. Junction 2 have the more abrupt doping. 54
3.6 Plot of DIBL versus lateral source/drain doping abruptness for devices with various gate lengths. DIBL is defined as in equation 2.13 55
3.7 Conventional I_{on}-I_{off} plot for devices with gate lengths ranging from 45 nm to 80 nm, lateral abruptness of the extension from 13 nm/dec to 1.9 nm/dec, and gate-extension overlap of 4 nm (solid symbols), 14 nm and -6 nm (open symbols). Channel doping is 4.8×10^{18}. The mobility model used is LUCMOB. 57

3.8 Nominal I_{on} versus nominal I_{off} plot for devices with gate lengths of (a) 70 nm (b) 50 nm. Lateral abruptness of the extension varies from 13 nm/dec to 1.9 nm/dec, gate-extension overlap for the devices ranges from 14 nm to -6 nm, while channel doping varies from 4.8×10^{18} to 9.0×10^{18}. The mobility model used is LUCMOB. 60

3.9 Plot of supernominal I_{on} versus subnominal I_{off} for devices with nominal gate lengths of (a) 70 nm (b) 50 nm. Gate-extension overlap for the devices ranges from 14 nm to -6 nm (solid symbols represent -1 nm case), while channel doping varies from 4.8×10^{18} to 9.0×10^{18}. The mobility model used is LUCMOB. 63

3.10 Overlap spacer is used as a tunable parameter. Metallurgical channel length (L_{met}) is kept constant for different lateral abruptness by varying overlap spacer length while gate length (L_{gate}) remains constant. . . . 65

3.11 Threshold Voltage and DIBL for devices with fixed halo location (Figure 3.10). The lateral abruptness of the extension junction is varied by controlling the characteristic length of the Gaussian profile. 66

3.12 Overlap spacer is not used as a tunable parameter. L_{met} is kept constant for different lateral abruptness by varying L_{gate}. 67

3.13 Threshold Voltage and DIBL for devices with varying halo location. The lateral abruptness of the extension junction is varied by controlling the characteristic length of the Gaussian profile. 67

3.14 Doping at the surface of devices in (a) Scenario 1 (b) Scenario 2, with various lateral abruptness ($\sigma_{x,ext}$ ranging from 2.6 to 26 nm) 69

3.15 Source/Drain doping profile examined in Table 3.5. Only the 6.5 nm/dec case is shown. Junction is located at x = 0 in this plot. . . . 73

4.1 Typical MOS Device (a) Schematic representation of a half device (b) Major resistive components along the conducting path 78

4.2 (a) Resistances connected in series. $R_{tot} = R_1 + R_2$. (b) Resistances connected in parallel. $R_{tot} = (1/R_1 + 1/R_2)^{-1}$. (c) Resistive block with resistivity of ρ. $R_{tot} = \int_0^L \frac{\rho}{W \cdot H} dl$. (d) Equivalent circuit of microscopic resistances. $\delta R = \frac{\rho \cdot \delta l}{\delta w \cdot \delta h}$. Virtual nodes a, b, c, d correspond to the cross-sections a, b, c, d shown in (c). 79

4.3 Partitioning a device based on vertical strips. 82

4.4 Current flow lines in a typical MOS device. 82

4.5 Partitioning a device based on equipotential lines. Conductance for a resistive strip defined by equipotential lines can be decomposed into microscopic conductances of value $dG = \frac{\sigma(\vec{x})}{d(\vec{x})} d\vec{l}$ connected in parallel . 84

4.6 Equivalent circuit for calculating the resistances of the source/drain contacts 87

4.7 (a) Comparison of R_{sh} calculated using: quasi-Fermi level at Si/SiO_2 interface ($\phi_n(x)$); numerical integration based on equipotential lines (Equi-ϕ); and numerical integration based on vertical strips (Vertical). (b) Expanded plot showing the overlap region (dashed box in (a)), where two-dimensional current flow is important. 91

4.8 Plot of sheet resistance along the channel for various V_{gs}. Lateral doping abruptness in the source/drain extension region is 1.9 nm/dec for this device. The gate extends from -0.025 to 0.025 μm. The metallurgical junctions lie at -0.011 and 0.011 μm respectively. 94

4.9 (a) Schematic Partitioning of resistive strips into bias dependent and independent parts (b) Plot of the contribution of the gate bias dependent part of the sheet conductance versus distance. 95
4.10 Comparing $\hat{R}_{sd}$ extracted through Shift-and-Ratio with the physical resistances calculated using equation 4.21 98
4.11 Flowchart showing how the terms extracted using the new extraction method contributes to the shift-and-ratio extraction results. Notice how the channel term contributes to the extracted source/drain resistance $\hat{R_{sd}}$. 100
4.12 Extracted $\widetilde{R_{sd}}$ term in Ω for the 50 nm device, the 80 nm device and their average. Note the difference between the 3 cases are less than 1% 101
4.13 Shift-and-ratio method applied to equation 4.27, with terms fitted to simulated devices with lateral extension abruptness of 1.9 nm/dec. (a) α factor. (b) Extracted R_{sd} from shift-and-ratio method. (c) R_{sd}^{-} from equation 4.33. (d) R_{sd}^{+} from equation 4.33. All resistance in Ω 102
5.1 Diagram showing the key (a) geometric and (b, c) source/drain doping parameters in the device structure template 110
5.2 Flowchart showing the use of the Python-based parameterized device description scripts. 111
5.3 Schematic showing the grid parameters. h_{x1}, h_{x2} and h_{x3} are the key horizontal, and h_{y1} and h_{y2} the key vertical grid spacing respectively. 112
5.4 Tensor product grid with $h_{x1} = 25nm$, $h_{x2} = 3.7nm$, $h_{x1} = 2.5nm$, $h_{y1} = 0.28nm$ and $h_{y2} = 6.7nm$ 112
5.5 Grid section of length L subdivided into n spaces with grid spacing varying smoothly from h_A to h_B 113

5.6 Grid section of length L with grid spacing varying smoothly from h_A to h_{AB} in n1 steps then to h_B in n2 steps 114
5.7 Farming jobs to a cluster of computers 115
5.8 Class diagram for simulation data storage. This UML [61] diagram shows that DataCV and DataIdVg delegates to and utilizes Simple-Database for performing their functions. 118
5.9 Composition of a mesh from a set of mesh objects 120
5.10 Template for lists of various mesh objects 121
5.11 Class hierarchy for mesh objects used in the resistance calculation code 122
5.12 Proxy Design Pattern for MYKMesh 124
5.13 Builder Design Pattern for TMATifReader. The UML sequence diagram in (b) shows the timing flow during a typical read() call. 126
5.14 Iterator Design Pattern for accessing the boundary edges of regions and electrodes in the mesh 126

A.1 Schematic of Simulated Devices. L_{drawn} = 180 nm. h_{x1}, h_{x2} and h_{x3} are the key horizontal, and h_{y1}, and h_{y2} are the key vertical grid spacing. 140
A.2 Grid for Simulated Device with $h_{x1} = 25nm$, $h_{x2} = 3.7nm$, $h_{x1} = 2.5nm$, $h_{y1} = 0.28nm$ and $h_{y2} = 6.7nm$ 141
A.3 Resolving Junction Gradient (a) Schematic view of device (box indicates region of interest) (b) coarse grid (15 nm) (c) additional grid points around s/d extension junction (d) dense in extension region (e) dense in extension and channel (4 nm) 142
A.4 I_d-V_g simulation for a device with $L_g = 0.35\mu m$ and ideal abrupt junction ($V_{ds} = 0.1V$) 144

A.5 Doping at the surface of the Si substrate for a device with $L_g = 0.35\mu m$ and ideal abrupt junction with various meshes 144
A.6 Drain Current for various values of h_{y1}. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{x3} = .02nm$ and $h_{y2} = 8nm$ 146
A.7 Drain Current for various values of h_{x3}. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{y1} = .28nm$ and $h_{y2} = 8nm$ 147
A.8 Error in the simulation Ion using the indicated physical models and grid with various (a) h_{x3} and (b) h_{y1} 148
A.9 CV simulations for different grid spacing in the polysilicon region close to the $poly/SiO_2$ interface. a. 10 um device. b. 180 nm device 150
A.10 Error in the Potential throughout the substrate of a device with extension abruptness of 3.5 nm/dec 152
A.11 Schematic of Typical HDD Device with Resistive Components in the Source/Drain 153
A.12 Simulated External Resistivity along the channel direction at various V_{gs} biases. $V_{ds} = 0.05V$, $L_{eff} = 0.08\mu m$ and $X_j = 40nm$ 155
A.13 Simulated External Resistivity along the channel direction for devices with various extension junction depth. $V_{gs} = 1.5V$, $V_{ds} = 0.05V$ (a)$x_j = 40nm$ (b)$x_j = 50nm$ (c)$x_j = 65nm$ (d) Peak Error Between the 2 meshes 156
A.14 Simulated External Resistivity along the channel direction for devices with various extension junction depth. $V_{gs} = 1.5V$, $V_{ds} = 1.5V$ (a)$x_j = 30nm$ (b)$x_j = 50nm$ (c)$x_j = 70nm$ (d) Peak Error Between the 2 meshes 157

A.15 Simulated (a) External Resistivity along the channel direction and (b) drain currents for four different meshes. $V_{gs} = 1.5V$, $V_{ds} = 1.5V$ and $x_j = 30nm$ 158
A.16 (a) Threshold roll-off curves and (b) R_{sd} for devices with various gate lengths from an 180 nm technology, simulated using meshes with $h_{x3} =$ 1.6Å and $h_{x3} = 0.8$Å 160
A.17 Error in the simulation of various electrical quantities with CONMOB for device a using grids with various (a) h_{x3} and (b) h_{y1} 162
A.18 Error in the simulation of various electrical quantities with CONMOB and UNIMOB for device a using grids with various (a) h_{x3} and (b) h_{y1} 163
A.19 Error in the simulation of various electrical quantities with GMCMOB and FLDMOB for device a using grids with various (a) h_{x3} and (b) h_{y1} 163
A.20 Error in the simulation of various electrical quantities with GMCMOB, FLDMOB and Van Dort for device a using grids with various (a) h_{x3} and (b) h_{y1} 164
A.21 Error in the simulation of Ion with CONMOB using grids with various (a) h_{x3} and (b) h_{y1} 164
A.22 Error in the simulation of Ion with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1} 165
A.23 Error in the simulation of Ion with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1} 165
A.24 Error in the simulation of Ion with GMCMOB, FLDMOB and Van Dort using grids with various (a) h_{x3} and (b) h_{y1} 166
A.25 Error in the simulation of Rs with CONMOB using grids with various (a) h_{x3} and (b) h_{y1} 166

A.26 Error in the simulation of Rs with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1} 167
A.27 Error in the simulation of Rs with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1} 167
A.28 Error in the simulation of height of Rsh spike with CONMOB using grids with various (a) h_{x3} and (b) h_{y1} 168
A.29 Error in the simulation of height of Rsh spike with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1} 168
A.30 Error in the simulation of height of Rsh spike with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1} 169
A.31 Grid section of length L subdivided into n spaces with grid spacing varying smoothly from h_A to h_B 171
A.32 Grid section of length L with grid spacing varying smoothly from h_A to h_{AB} in n1 steps then to h_B in n2 steps 171

Chapter 1

Introduction

1.1 Device Scaling and Moore's Law

The electronic industry has achieved truly amazing improvements over the past three decades. As Figure 1.1a shows, the maximum clock-speed of the central processing unit (CPU) has increased from 100kHz in 1971 to over 1.4 GHz in 2000, an improvement of over 4 orders of magnitude. Over the same period, the cost of computers actually decreased with the rise of the personal computer[1]. Comparable improvements in price/performance ratio have never before been witnessed in any other products in human history.

These trends resulted in large part from the success of the semiconductor industry in shrinking the transistor. Figure 1.1a shows the decrease of the transistor gate length from 10 μm in 1971 to 0.18 μm in 2000. This is responsible for both the increase in clock-speed observed (through the shortening of signal path), and the continuation of

[1] The original IBM PC started at $1565 in 1981, or $4000 in today's dollars [9]; a 1.5 GHz Pentium IV based computer cost around $3500 in 2000 [28].

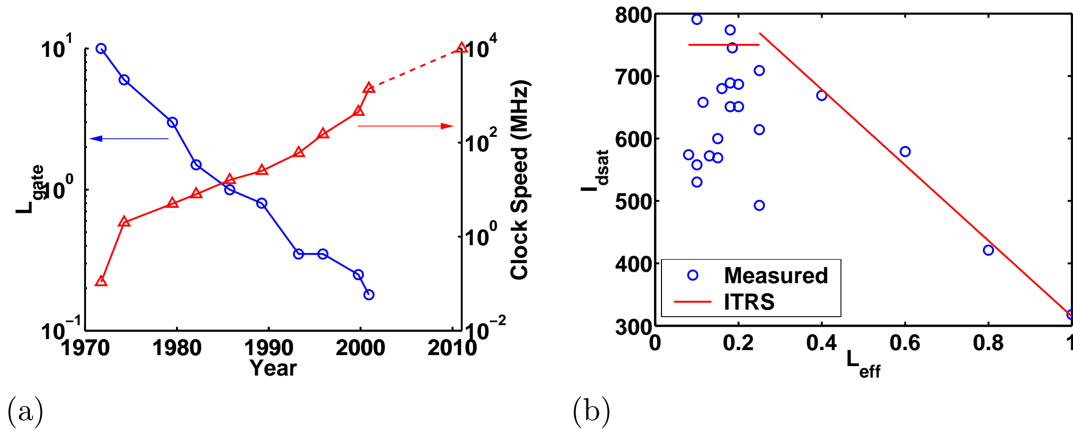


Figure 1.1: Device Scaling (a) CPU clock speed and MOS device gate lengths versus year (b) MOS device saturation current versus technology generation (represented by the minimum device length)

Moore's law [52], which states that the number of transistors per integrated circuit will double every 18 months. Not only does this allow more functionality to be integrated on to a single chip, the same functionality now consumes less silicon real estate, leading to cost savings. It is due to this relentless MOS scaling that CMOS (Complementary Metal-Oxide-Semiconductor) has emerged as the dominant technology for Very Large Scale Integration (VLSI) circuits [73].

At the same time, there are many challenges to continued device scaling. As an example, consider Figure 1.1b which shows the maximum saturated current through a MOS device as the gate length is decreased for successive technology generations. For older device generations[2], the saturated current flowing through the device increases as the gate length decreases. As we shall see in Section 2.1.1, increasing saturated current I_{dsat} is related to increasing circuit performance. Unfortunately, from the 0.25 μm technology onwards, device scaling no longer results in an increase in saturated

[2]i.e. Long gate lengths.

current. This implies that simple device scaling in the deep sub-micron region may not by itself be sufficient to improve performance. Understanding the limits to scaling and how they relate to different device parameters is critical for the design of next generation devices.

1.2 ITRS Roadmap

Throughout this thesis, references will be made to the International Technology Roadmap for Semiconductors (ITRS) [1][3]. This roadmap was created by a group of experts from both academia and industry, and serves several purposes. It specifies key device design parameter values that serve both as predictions and targets[4] for future device technology generations. It also highlights specific challenges that future generations of device scaling will likely face. As a result, it is an important reference document for process and device designers alike.

At the same time, prediction of the future is a tricky business. Many of the values presented in the ITRS result from back of the envelope calculations and extrapolations of current trends, and as such, may be little more than highly intelligent guesses. In this thesis, we shall examine in detail the ITRS predictions regarding a device parameter that has garnered significant attention in literature as a critical limiting factor of device scaling: the lateral doping abruptness of the source/drain extension region. The lateral abruptness values predicted by the ITRS (shown in Table 1.1) are very aggressive and challenging to achieve. Consequently, the validity of these predictions could have significant implications on the proper allocation of research

[3]For instance, Figure 1.1 is based in part on ITRS predictions.

[4]Some might say, self-fulfilling prophecies.

Year	t_{ox} (nm)	Drain Extension X_j (nm)	Lateral Abruptness (nm/decade) low	high
1999	1.9-2.5	42-70	4.8	14
2000	1.9-2.5	36-60	4.1	12
2001	1.5-1.9	30-50	3.4	10
2002	1.5-1.9	25-42	***2.9***	8.5
2003	1.5-1.9	***24-40***	***2.7***	8
2004	1.2-1.5	***20-35***	***2.4***	7
2005	***1.0-1.5***	***20-33***	***2.2***	6.5
2008	***0.8-1.2***	***16-26***	***1.25***	***4.5***
2011	***0.6-0.8***	***11-19***	***0.8***	***3.2***
2014	***0.5-0.6***	***8-13***	***0.5***	***2.2***

Table 1.1: International Technology Roadmap for Semiconductors 1999 Edition. ***italics*** indicate no known solutions.

and development resources for the semiconductor industry.

1.3 Methodology of this Work

Physical experiments involving lateral abruptness are difficult to conduct, due to the challenges presented by multi-dimensional doping metrology (Section 2.5.1). Without accurate measurements of the lateral abruptness of fabricated devices, it is difficult to quantify the exact impact of lateral abruptness on device performance via experimental means.

Figure 1.2 shows the schematic flow of the methodology employed by this thesis. Instead of physical experimentation, TCAD simulations are used. A large number of device designs are simulated. The physical resistance components, threshold voltages and I_{on}-I_{off} characteristics for each device design are extracted and compared to obtain understanding of scaling limits with respect to lateral abruptness. At the same time, the resistance calculations are used to quantify the errors introduced by

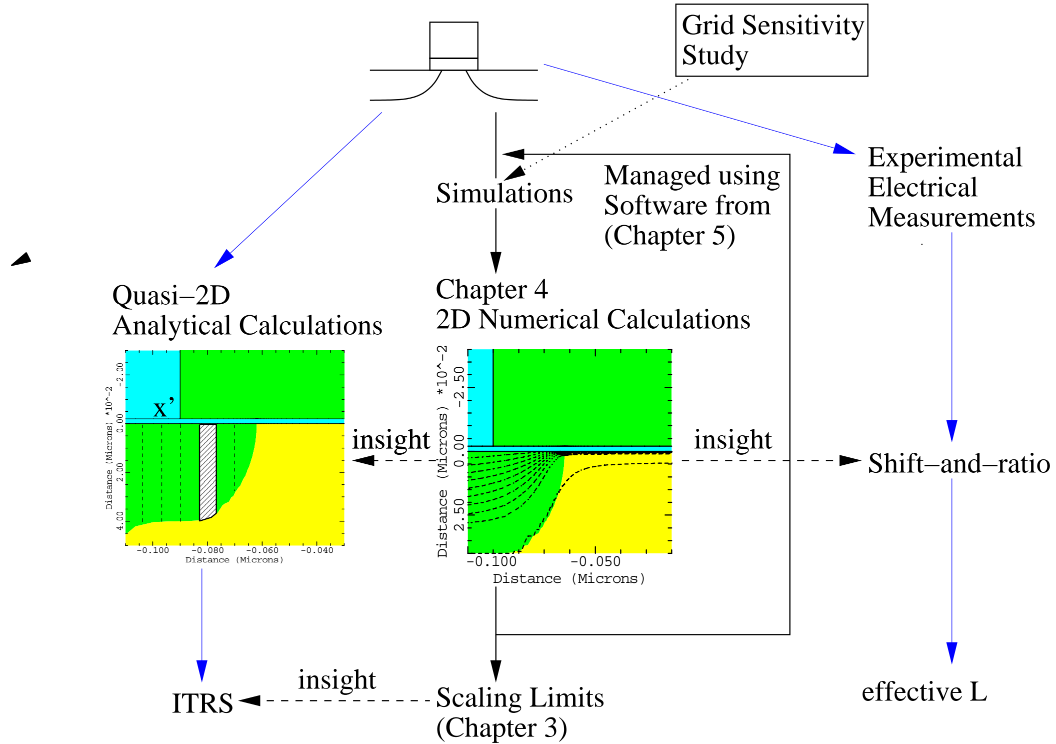


Figure 1.2: Flow chart indicating the approach of this thesis. Using software described in Chapter 5, simulations of numerous MOS devices are performed. The results are processed as in Chapter 4, which provides insight on the analytical resistance calculations reported by Ng, et al [57], the shift-and-ratio method and the scaling limits concerning lateral doping abruptness in the source/drain extension. Grid sensitivity studies are conducted to ensure simulation accuracy.

the resistance calculation method used by the ITRS, as well as better understand the limits of the shift-and-ratio extraction method.

The simulation data is generated and processed using commercial TCAD simulators in conjunction with custom software described in Chapter 5. The custom software post-processes the simulated results to allow easy calculation of resistive components (using the algorithms described in Chapter 4), threshold voltages and various I_{on}-I_{off} plots, through the use of a simple database (Section 5.3). Together

with the device template and the job farming system described in Sections 5.1 and 5.2, these software programs help make the large number of simulations needed for Chapter 3 manageable.

Several issues are worth noting at this point. Firstly, the use of TCAD simulations provides many benefits. It allows complete control of the lateral abruptness and doping profile of the simulated devices. This makes it possible to pin down the impact of lateral abruptness. Moreover, this approach enables a large number of device designs to be examined and compared. A comparable study using silicon wafers would be prohibitive from a time and cost viewpoint. Furthermore, this methodology takes full advantage of the ability of device simulators to calculate internal quantities that can not be obtained experimentally. This in turn enables a rigorous method of calculating channel and source/drain resistance (Chapter 4).

Secondly, an important aspect of any device design study is the metrics used for comparing different devices. In this work, special care is taken to ensure the threshold voltages are consistently defined (Section 2.1.2) and resistance components are properly calculated (Chapter 4). Several commonly used device metrics, including source/drain resistance (Section 3.3), threshold voltage (Section 3.4) and I_{on}-I_{off} curves (Section 3.5), are examined and compared to make sure that the major salient aspects of device performance have been taken into account (Section 3.2.2). This is important in understanding the impact of lateral abruptness.

Thirdly, grid spacing and the physical models chosen can have a dramatic effect on the simulated results. Throughout this thesis, sensitivity studies of the simulation grid are used to determine the appropriate grid spacing to use (Appendix A) for minimizing numerical errors in the simulations. The impact of mobility models is examined by repeating the simulations with different mobility models (Section 3.6.3)

to ensure the central conclusions of the thesis remain consistent.

Last but not least, this work adheres to the KISS (Keep It Simple Stupid) principles. Chapter 3 begins by assuming that the devices have a uniform channel doping, and that the source/drain doping can be described by a simple tensor product of exponential functions in the vertical and horizontal directions (Section 3.2). These assumptions are then revisited and/or relaxed in later sections (Sections 3.6.1 and 3.6.2). This approach allows us to isolate the impact of different aspects of device design on device behavior, and is critical in reaching the proper understanding and conclusions with respect to lateral abruptness.

1.4 Scope and Organization

The remainder of this thesis describes the challenges facing device design in the deep sub-micron region, paying special attention to those that have the most relevance for the rest of this thesis: short channel effects (Section 2.2.1), reverse short channel effects (Section 2.2.2), and source/drain parasitic resistances (Section 2.2.3). Device design considerations such as channel, halo (Section 2.3.1) and source/drain (Section 2.3.2) doping are then presented. In particular, lateral source/drain abruptness (Section 2.3.2) is considered in the context of the ITRS. Chapter 2 concludes with a discussion of the benefits of Technology Computer Aided Design (TCAD) (Section 2.4) and the challenges in its application for deep sub-micron technology (Section 2.5).

Chapter 3 uses device simulations to examine the impact of lateral doping abruptness on device performance from several perspectives: parasitic series resistance (Section 3.3), threshold voltage roll-off (Section 3.4), and three kinds of I_{on}-I_{off} plots

(Section 3.5). These results are contrasted with the conventional wisdom in literature. The impact of halo doping on these results is then examined in Section 3.6.1; several other assumptions used in the simulations are also revisited.

Chapter 4 describes in detail how (series) resistance in an active MOS device should be calculated. A comparison of resistance calculations based on vertical strips (Section 4.4), equipotential lines (Section 4.5) and quasi-Fermi level at the surface (Section 4.6) is presented. The vertical strip method is shown to introduce substantial error in the resistance values. Moreover, it tends to overestimate the benefits of a laterally abrupt junction in reducing series resistance. This has important implications for the ITRS roadmap.

Chapter 5 describes the design and implementation of software for running and managing large number of simulations (Sections 5.1 and 5.2), manipulating and visualizing the simulation results (Section 5.3), and calculating resistances (Section 5.4). This chapter also serves as a case study of how modern software engineering techniques can be applied to the design (Section 5.4) and implementation (Section 5.5) of TCAD software.

Finally, Chapter 6 provides a summary of the contributions, findings and lessons learned as presented in this thesis, and concludes with some possible future work.

We now proceed with a discussion of the challenges facing device design in the deep sub-micron regime.

Chapter 2

Challenges for Deep Sub-micron Device Design

As we noted in Section 1.1, continued device improvements with device scaling have become more difficult. The main reason for this is that many second order effects that could largely be ignored in the past are coming to the fore. These effect in turn must be controlled and minimized using more complicated device designs. Consequently, technology design has become more challenging than ever.

This chapter begins with a description of several key device design parameters and how they are influenced by device scaling as predicted by first order theory (Section 2.1). Several of the most important second order effects are then discussed (Section 2.2). Special attention is paid to short channel and doping related effects. These provide the motivation for much of the research reported in this thesis.

Section 2.3 follows with several doping designs commonly found in deep sub-micron MOSFET devices, and introduces the concept of lateral doping abruptness in the source/drain extension region. Finally, sections 2.4 and 2.5 discuss the promise

and challenges facing Technology Computer Aided Design (TCAD), which is integral to the methodology employed throughout this thesis (Section 1.3).

2.1 Device Design Parameters and Device Scaling

To understand the challenges facing deep sub-micron device design, we must first understand the design parameters available to a device designer and their significance. Section 2.1.1 provides a brief overview of several key parameters. In particular, threshold voltage is a key parameter for both digital and analog applications, and considerable attention is paid to its definition (Section 2.1.2). This will be used in the rest of this thesis to ensure fair and accurate comparisons between device designs. Since second order effects are really deviations from first order trends, it is also useful to review first order theories of how device parameters should change with device scaling, the topic of Section 2.1.3.

2.1.1 Key Device Design Parameters

The goal of device design is to obtain devices with high performance, low power consumption, low cost and high reliability in the field [73]. To simplify the design process, it is useful to distill these requirements into a small number of key design parameters that the designer can focus on.

Device parameters of particular interest include the intrinsic capacitance (C_{ox}), the parasitic capacitances (C_{gsd}) and the on-current (I_{on}), which determine the performance of digital circuits; the worst-case off-current (I_{off}), which is related to the static power consumption of digital circuits; and the transconductance (g_m), intrinsic

$$
\begin{aligned}
C_{ox} &= \epsilon_{ox}/t_{ox} \\
I_{on} &= I_{ds}(V_{ds} = V_{dd}, V_{gs} = V_{dd}) \\
I_{\text{off}} &= I_{ds}(V_{ds} = V_{dd}, V_{gs} = V_{\text{off}}) \\
P_{\text{off}} &= I_{\text{off}} \cdot V_{dd} \\
P_{dynamic} &= f \cdot C_{load} \cdot V_{dd}^2
\end{aligned}
$$

Table 2.1: Useful equations for calculating basic electrical quantities for a transistor.

output resistance (r_0) and unit-current-gain frequency cut-off (f_T), which are important for high performance analog circuits. Table 2.1 shows first order relations for some of these design parameters.

An especially important device parameter is the threshold voltage (V_t). It imposes a lower limit for the supply voltage (V_{dd}), since for adequate current levels, we must have sufficient over-drive $V_g - V_t$ ($= V_{dd} - V_t$). This in turn affects the power consumption of the device. V_t is also related to I_{on} and I_{off}, which influence the device performance and power consumption of digital applications. Moreover, the matching of threshold voltages for different MOS devices is important for the proper operation of many analog circuits. As a result of all this, V_t is often used to monitor the manufacturing process. We will now turn our attention to this important parameter.

2.1.2 Definition of Threshold Voltage

Conceptually, the threshold voltage is simply the applied gate-to-source voltage needed to turn on a MOS device. Figure 2.1a shows the electrical characteristics of an ideal switch. As shown in the Figure, the threshold voltage for such a hypothetical device is easy to define.

A MOS device, however, is not an ideal switch. As a result, arriving at a self-consistent definition of the threshold voltage for real devices is non-trivial. In this

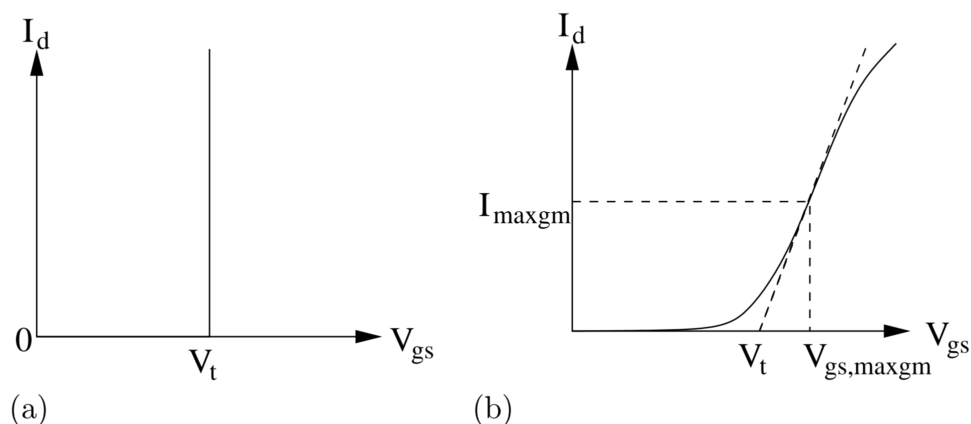


Figure 2.1: (a) Threshold voltage of an ideal switch (b) Maximum g_m definition of threshold voltage.

subsection, two commonly used definitions of threshold voltage, along with their respective limitations, are discussed, followed by a hybrid approach that attempts to solve these problems. The latter is the definition adopted for the rest of the thesis. Finally, the first-order relationships between threshold voltage and the substrate doping for a uniformly doped device is presented. This will be useful for understanding MOSFET doping effects.

Maximum g_m Definition

The threshold voltage can be defined as the horizontal intercept of the tangent to the I_d-V_{gs} curve at the point of maximum g_m (Figure 2.1b) as follows:

$$V_{gs,maxgm} = \underset{V_{gs}}{\text{maximize}} \left. \frac{dI_d}{dV_{gs}} \right|_{V_{ds}} \tag{2.1}$$

$$I_{maxgm} = I_d|_{\substack{V_{ds} \\ V_{gs}=V_{gs,maxgm}}} \tag{2.2}$$

$$V_t = V_{gs,maxgm} - \frac{I_{maxgm}}{\left. \frac{dI_d}{dV_{gs}} \right|_{V_{gs}=V_{gs,maxgm}}} \tag{2.3}$$

This definition does not depend on any arbitrarily chosen parameters, and can therefore be used consistently across different technologies. However, it can be applied only for the linear region (low V_{ds}) and not the saturation region.

Constant I_d Definition

Another widely used definition of the threshold voltage is based on an "appropriately" chosen reference drain current (I_{ref}). The threshold voltage at a given drain voltage is defined as the gate voltage required to produce the reference current for the device concerned.

$$V_t|_{V_d=V_{ds}} = V_{gs}|_{I_d=I_{ref}|_{V_d=V_{ds}}} \tag{2.4}$$

The advantage of the constant current definition is its simplicity. Unlike the maximum g_m definition, there is no need for derivative calculations, which tends to be sensitive to measurement noise. Furthermore, it can be applied for both linear and saturated operation, allowing comparisons of threshold voltage at various bias conditions. However, the appropriate reference current I_{ref} is technology dependent,

making it difficult to use this definition to compare devices from different companies or even over multiple technology generations.

"Critical-Current at Linear-Threshold" Method

In order to overcome these limitations, the "critical-current at linear-threshold" method [91] seeks to combine the maximum g_m definition with the constant current definition.

The linear threshold voltage is defined similarly to the maximum g_m definition.

$$V_{gslin,maxgm} = \underset{V_{gs}}{\text{maximize}} \left. \frac{dI_d}{dV_{gs}} \right|_{V_{ds}=V_{d,lin}} \tag{2.5}$$

$$I_{crit} = I_d \Big|_{\substack{V_{ds}=V_{d,lin} \\ V_{gs}=V_{gslin,maxgm}}} \tag{2.6}$$

$$V_{t,lin} = V_{gslin,maxgm} - \frac{I_{crit}}{\left. \frac{dI_d}{dV_{gs}} \right|_{V_{gs}=V_{gslin,maxgm}}} \tag{2.7}$$

The threshold voltage in the saturation region is then defined as

$$V_{t,sat} = V_{gs} \Big|_{I_d=I_{crit}, V_{ds}=V_{dd}} \tag{2.8}$$

where I_{crit} is the critical current as defined in equation 2.6.

The "critical-current at linear-threshold" method provides a simple, consistent method for defining threshold voltages that works well for both linear and saturated regions of operation and is independent of technology. It can be used for comparing the threshold voltage for devices from different technology designs at various biasing conditions. We shall define threshold voltage using this method for the rest of this thesis.

It is also instructive to examine the theoretical equations governing threshold

voltage for simple device designs. This will be examined in the next subsection.

Threshold Voltage for Devices with a Uniformly Doped Channel

The threshold voltage for a uniformly doped long channel device can be approximated by solving the one-dimensional Poisson's equation [56]. For an n channel device

$$V_t = V_{FB} + 2|\phi_p| + \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_a(2|\phi_p| - V_{BS})} \tag{2.9}$$

For a p channel device

$$V_t = V_{FB} - 2|\phi_n| - \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_d(2|\phi_n| + V_{BS})} \tag{2.10}$$

Here V_{FB} is the flat-band voltage, which for devices with a uniformly doped substrate, is given by

$$V_{FB} = \Phi_{MS} - \frac{Q_f}{C_{ox}} - \frac{1}{C_{ox}}\int_0^{x_{ox}} \frac{x}{x_{ox}}\rho(x)dx \tag{2.11}$$

This in turn is a function of the gate-substrate work-function, as well as the amount of oxide charges present. ϕ_n $(= kT/q \cdot \ln(N_d/n_i))$ and ϕ_p $(= -kT/q \cdot \ln(N_a/n_i))$ represent the built-in potential of the MOS device structure; and N_a and N_d are the acceptor and donor dopant concentration in the substrate respectively.

The key thing to note is that the threshold voltage is strongly dependent on doping concentration. As we increase the doping level, the threshold voltage becomes more positive (negative) for an n-channel (p-channel) device.

2.1.3 First Order Device Scaling Theory

Before proceeding to examine the second order effects important in the deep sub-micron region, it is useful to briefly review the predictions of first order theory. After all, second order effects represent nothing more than a deviation from first order predictions.

In this subsection, we focus on first order scaling laws that summarize different methodologies for scaling device quantities together with their impact on device performance. The intent of these scaling laws is to assist the device designer in understanding and managing the inherent trade-offs in device design.

Constant field MOS scaling theory by Dennard et al [24] is a commonly referenced scaling law. Its main goal is to maintain the electrical integrity of the transistor. As we can see in Table 2.2, under constant field scaling, both gate delay and power consumption decreases as scaling continues. This leads to faster and more efficient semiconductor devices, consistent with Figure 1.1.

However, as discussed at the end of Section 1.1, in reality, the achievable improvements are less than those predicted by basic scaling theories, due to physical and technological limitations. This presents many challenges to the continuation of device scaling, which constitute the main topic of the rest of this chapter.

2.2 Second Order Effects in the Deep Sub-micron Regime

Device design in the deep sub-micron regime is a challenging endeavor. The basic, first-order theories (such as the constant field MOS scaling theory discussed in Section 2.1.3) do not fully describe device behavior. Second-order effects, such as threshold

Parameter	Scaling Model Constant Field
Length	$1/\alpha$
Width	$1/\alpha$
Junction Depth	$1/\alpha$
Gate oxide thickness	$1/\alpha$
Supply Voltage	$1/\alpha$
Substrate Doping	α
Depletion layer thickness	$1/\alpha$
Current	$1/\alpha$
Transconductance	1
Electric Field across gate oxide	1
Load Capacitance	$1/\alpha$
Gate Delay	$1/\alpha$
DC power dissipation	$1/\alpha^2$
Dynamic power dissipation	$1/\alpha^2$
Power-delay product	$1/\alpha^3$

Table 2.2: Constant Field Scaling [84]

variations with channel length due to charge sharing (Section 2.2.1) and doping effects (Section 2.2.2), extrinsic resistances (Section 2.2.3), and quantum mechanical effects (Section 2.2.4), must be taken into account. These second order effects combine to make it difficult for the device designer to have an intuitive grasp of all the trade-offs inherent in device design, complicating the design process.

2.2.1 Short Channel Effects

Equations 2.9 and 2.10 from Section 2.1.2 are based on the solution of Poisson's equation in one dimension. They are appropriate for long channel devices, for which the influence of electric fields emanating from the source/drain regions are much less important than those coming from the gate, since the center of the channel is far from the source/drain region, and the edge of the channel is only a small part of the

device.

This is not the case for devices with short channel lengths. To calculate the threshold voltage in these cases, the full multi-dimensional charge balance must be considered. This can have a significant impact on the threshold values. In practice, short channel effects provide the lower limit of achievable channel lengths for a given technology. Short channel effects tend to lower the threshold voltage for short channel devices. In turn, this leads to larger I_{off} and higher power consumption. Short channel effects also have implications on the matching of device threshold in analog circuits, since the threshold voltage for devices with different channel lengths are no longer the same. Proper modeling of short channel effects is therefore critical for device design.

In this subsection, two different manifestations of short channel effects, threshold voltage roll-off and drain induced barrier lowering, will be examined.

Threshold Voltage Roll-Off due to Short Channel Effects

Consider an n-channel device with a long channel and a target threshold voltage of V_t. It was observed that the threshold voltage of another device with the same technology design but a shorter channel length can be significantly smaller than V_t, as indicated by ΔV_- on Figure 2.2. This is known as threshold voltage roll-off.

This phenomenon can be explained through a charge sharing argument. Unlike the long channel device, in a short channel device, a significant portion of the field lines emanating from the bulk charge terminate in the source and drain regions instead of at the gate. As a result, it is easier for the gate to deplete the channel, lowering the threshold voltage of the device. Using the model given by Yau [87], the NMOS device

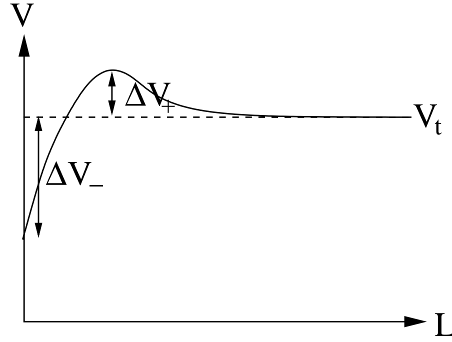


Figure 2.2: Plot of threshold voltage (V_t) versus channel length (L) for a typical MOS device technology

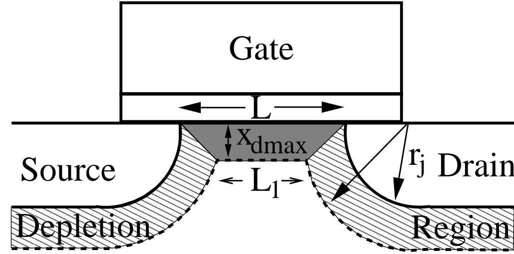


Figure 2.3: Charge sharing model for explaining short channel effects (Yau [87])

threshold voltage is given by

$$V_t = V_{FB} + \frac{\sqrt{-4\epsilon_s q N_a \phi_p}}{C_{ox}} \left[1 - \frac{\mathbf{r_j}}{\mathbf{L}} \left(\sqrt{\mathbf{1} + \frac{\mathbf{2x_{dmax}}}{\mathbf{r_j}}} \right) \right] - 2\phi_p \qquad (2.12)$$

where L is the channel length of the device, r_j is the source/drain junction depth, x_{dmax} is the depletion depth in the channel substrate, V_{FB} is the flat band voltage of the MOS system and ϕ_p is the bulk potential (Figure 2.3). The amount of threshold voltage roll-off is given by the negative term highlighted in **bold** in equation 2.12. Note that increasing junction depth exacerbates short channel effects.

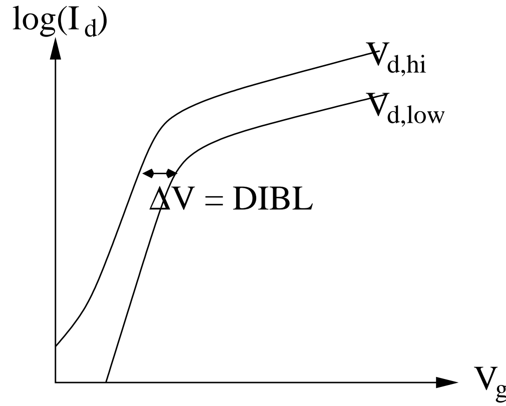


Figure 2.4: Current versus gate voltage plot at low and high drain biases, demonstrating drain induced barrier lowering

Drain Induced Barrier Lowering (DIBL)

For short channel devices, application of a high drain-to-source bias can lower the threshold voltage and increase the off-currents, as shown in Figure 2.4. This is known as drain induced barrier lowering, and is another manifestation of the short channel effects.

Conceptually, drain barrier lowering is caused by the lowering of the potential barrier at the source of a MOSFET due to applied drain bias (Figure 2.5). There are no known closed form expressions for predicting the threshold shift resulting from DIBL. A treatment based on two-dimensional computer simulations can be found in Troutman [78]. The key insight is, like threshold voltage roll-off, DIBL effects increase with increasing junction depth of the source/drain region.

A measure of the severity of DIBL can be defined as follows

$$DIBL = \frac{V_{t,lin} - V_{t,sat}}{V_{dd} - V_{dlin}} \tag{2.13}$$

where V_{dd} is the supply voltage, $V_{d,lin}$ is the linear drain voltage, and $V_{t,lin}$ and $V_{t,sat}$

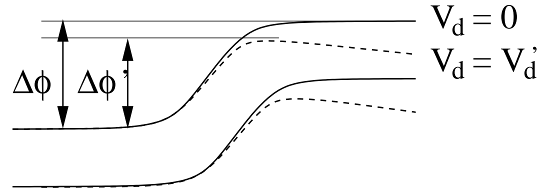


Figure 2.5: The energy band diagram at the source end of a half NMOS device with and without an applied drain bias.

are the threshold voltages in linear and saturated operation respectively, as defined by equations 2.7 and 2.8.

While threshold voltage roll-off is widely used to measure short channel effects, the amount of threshold voltage roll-off is also affected by non-uniform doping effects[1]. Hence, DIBL may be a better indicator of short channel effects than voltage roll-off. As such, it is an important device characteristic that should be monitored.

2.2.2 Reverse Short Channel Effects

Consider again the threshold voltage versus channel length plot for a typical technology (Figure 2.2). Instead of decreasing with channel length as short channel effects would predict (Section 2.2.1), for many device designs, as the channel length is scaled down, the threshold voltage of the n-channel devices may become more positive (indicated by ΔV_+ in Figure 2.2), before eventually coming back down and exhibiting the standard threshold voltage roll-off characteristics. This is known as the reverse short channel effect [58].

One explanation for reverse short channel behavior is presented by Rafferty, et al [64]. The source/drain implants introduce a retrograde profile of point defects at

[1]Such as the reverse short channel effect (Section 2.2.2) and halo doping (Section 2.3.1).

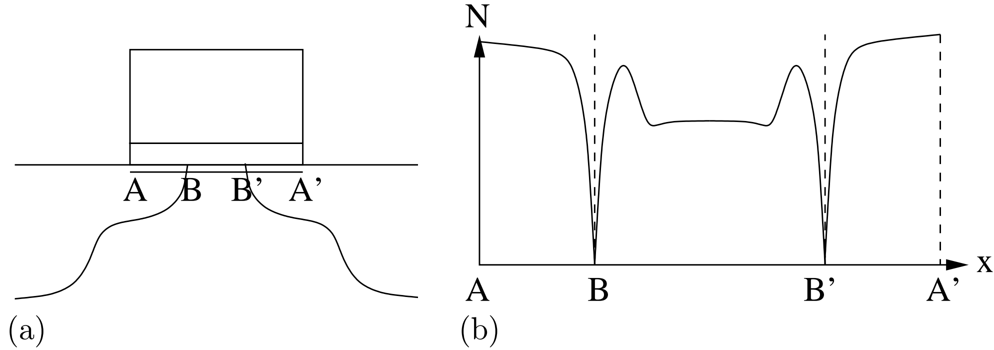


Figure 2.6: Schematic diagram showing a non-uniform lateral doping distribution that can be used to explain the reverse short channel effects (a) Schematic of the device (b) Doping profile along the line AA'

the edge of the channel. This in turn leads to pile up of boron impurity atoms at the surface, increasing the channel doping close to the source/drain region (Figure 2.6b). While the impact of this increase in doping is small for long channel devices, for short channel devices the region with the increased doping is a significant part of the channel. The resultant increase in the effective channel doping with shortening channel length causes the threshold voltage to increase with scaling, until eventually short channel effects take over. This explains the observed roll-up in the threshold voltage in Figure 2.2.

A physical model of the lateral doping distribution responsible for reverse short channel effects can be found in Hanafi, et al [32].

2.2.3 Source/Drain Resistance

Another second order effect results from parasitic device components. Total device resistance can be broken up into an intrinsic and an extrinsic component as follows

$$R_{tot} = R_{chan} + R_{ext} \tag{2.14}$$

The dramatic increases in device performance over the past three decades can be attributed to the decrease of intrinsic channel resistance as the dimensions of a MOS device become smaller. However, as scaling continues, the parasitic resistances have become comparable in value to the intrinsic components and can no longer be ignored.

As we noted in Section 2.2.1, short channel effects are aggravated by increasing the junction depth of the source/drain region. This provides an upper limit on the allowable junction depth. Short channel effects also put a lower limit on the length of the source/drain extension, as the source/drain extension is needed to minimize the impact of the deep source/drain region on short channel effects. At the same time, solid solubility limits the minimum resistivity we could achieve.

As a result of these factors, the extrinsic series resistances of MOS devices do not scale well. The total resistance of the source/drain extension can be estimated using

$$R_{ext} = \frac{\rho L}{A} \tag{2.15}$$

where L is the length of the extension region, and $A = Wx_d$ is the cross-sectional area. Short channel effects and solid solubility limit our ability to tune these parameters and lower the source/drain resistance. As intrinsic device resistance continues to decrease, the parasitic resistances are becoming an increasingly significant part of the total device resistance (Figure 2.7). Their impact on deep sub-micron device performance can no longer be ignored, which further complicates device design.

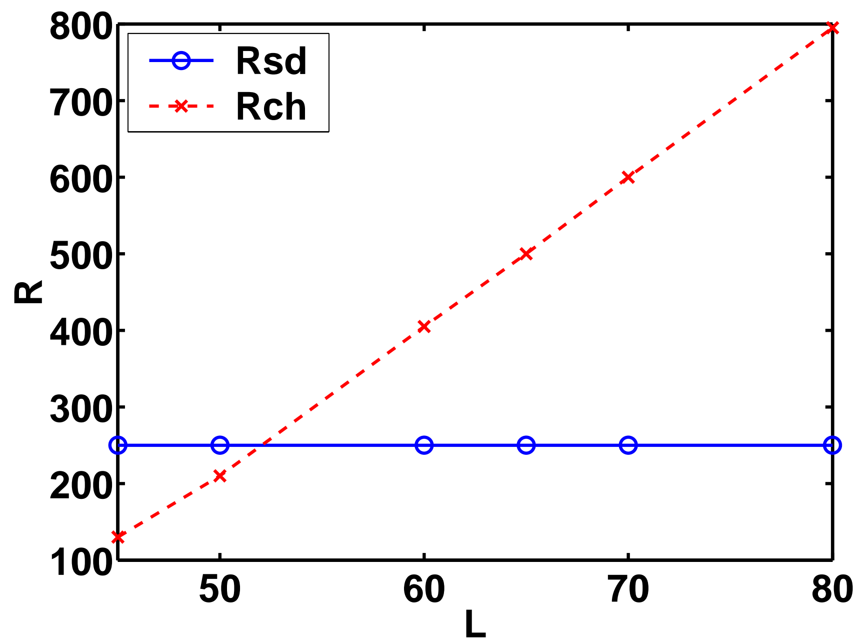


Figure 2.7: Intrinsic resistance R_{ch} and extrinsic resistance R_{sd} versus technology generation (indicated by channel length)

2.2.4 Quantum Mechanical Effects

Making matters worse, classical and semi-classical treatments of solid state physics[2] break down for very small spatial scales. To get an idea of the dimensions involved, consider the Planck-Einstein-de Broglie relations

$$\varepsilon = \hbar\omega \tag{2.16}$$

and the dispersion relation for objects with mass

$$\omega(\vec{k}) = \frac{\hbar k^2}{2m_e} \tag{2.17}$$

The wavelength of an electron is then given by

$$\lambda = \frac{2\pi\hbar}{\sqrt{2m_e\varepsilon}} = 1\,\mathrm{nm} \cdot \sqrt{\frac{1.504eV}{\varepsilon}} \tag{2.18}$$

Hence quantum mechanical effects will be dominant for spatial dimensions on the order of 1 nm [44]. This is comparable to the oxide thicknesses in the deep sub-micron region. As a result, quantum mechanical effects must be considered in modern device design.

Quantum mechanical effects are manifested in two ways in deep sub-micron devices: as charge quantization effects and gate tunneling currents.

[2] Typical of introductory semiconductor courses [56], and still the dominant methodology in device modeling in industry.

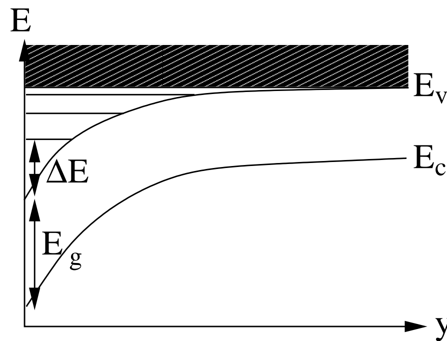


Figure 2.8: Potential well at the Si/SiO_2 interface, charge quantization and band-gap widening

Charge Quantization Effects

For devices with very thin gate oxides (less than 30 Å) and high substrate doping levels, the existence of strong normal electric fields close to the Si/SiO_2 interface, together with the potential barrier presented by the dielectric, creates a potential well for the mobile carriers in the inversion layer. Due to the existence of this potential well, the allowed energy states no longer form a "continuous band", but instead are quantized and discrete. As a consequence, the lowest allowed energy level does not remain aligned with the conduction band edge (shown as ΔV in Figure 2.8). This leads to an effective widening of the band-gap [81].

As a result, the maximum achievable inversion capacitance is reduced, often by 10% or more. At the same time, the threshold voltage of the device becomes larger than classical theory would predict (Figure 2.9). Given the importance of the threshold voltage in determining device performance (as discussed in Section 2.1.2), these effects must be accounted for.

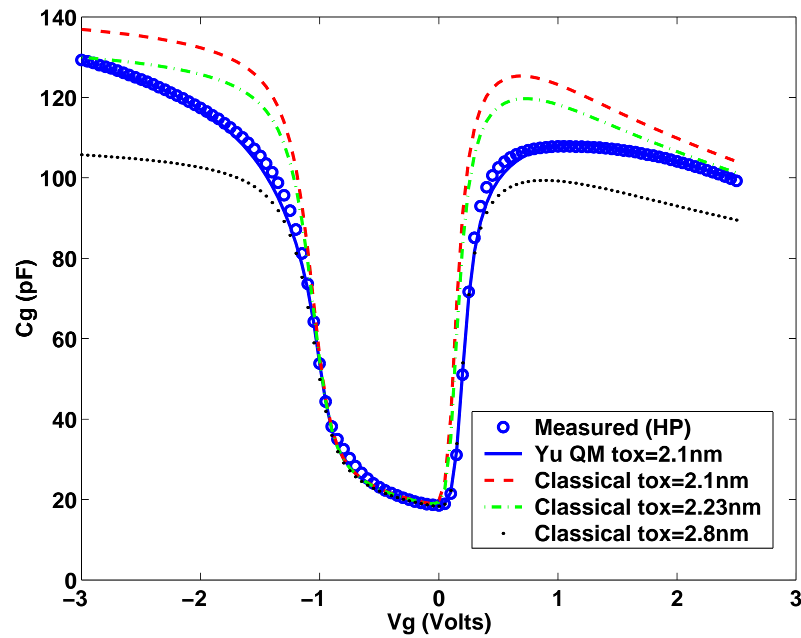


Figure 2.9: Capacitance-Voltage plot for a device with gate oxide thickness of 2.1 nm modeled with classical models and the quantum mechanical Yu model [16]. Note the substantial difference between the two curves with respect to inversion/accumulation capacitance and threshold voltage. Also shown are models based on effective oxide thickness to fit the accumulation capacitance (2.23 nm) and threshold voltage (2.8 nm) respectively. Both cannot be fitted to at the same time using classical models.

Gate Tunneling Current

Furthermore, under quantum theory, there is a finite chance that carriers could tunnel through the oxide barrier and result in non-zero gate current. The classical assumption that gate currents for MOS device are negligible no longer holds in the deep sub-micron regime.

This is especially important for devices with ultra thin gate oxides, due to the exponential dependence of direct tunneling gate current on insulator thickness as follows [18]

$$J_{DT} = C \left(\frac{V_{ox}}{t_{ox}} \right)^2 e^{-Bf(V_{ox},\phi_B)t_{ox}} \tag{2.19}$$

For devices with oxide thicknesses approaching 1 nm, tunneling currents cannot be ignored.

This non-zero gate current has several implications on device operation. The gate current leaks away the inversion charge, leading to anomalous capacitance-voltage curves [19]. This makes the task of estimating the effective electrical oxide thickness more difficult. Moreover, it contributes to the leakage current of the device. Ghani et al [30] estimated that for nitrided oxides, the gate tunneling current would replace the source/drain leakage current as the dominant component of off-currents for devices in the 50 nm technology node and beyond. This in turn imposes a lower limit on the allowable dielectric thickness[3]. The increased leakage current also degrades the operation of dynamic CMOS circuit techniques (such as Domino circuits).

An active area of research is the use of alternative high-permeability (high-K) material as the gate insulator. The goal is to maintain a high level of inversion charge with a thicker gate insulator layer than the SiO_2 case. This minimizes gate currents

[3]Through power considerations

and at the same time allow device scaling to continue [38]. Some common materials being considered include TiO_2 and ZrO_2.

2.3 Device Design in the Deep Sub-micron Regime

As the second order effects described in Section 2.2 become more important for deep sub-micron devices, so does their control through proper device design. Highly complex, non-uniform doping designs using channel engineering (Section 2.3.1) and source/drain engineering (Section 2.3.2) are needed for acceptable device performance in this regime.

2.3.1 Channel Engineering

A good discussion of the need for non-uniform channel doping is given in Taur and Ning [74]. For a MOS device with a uniform doping, the maximum depletion depth and the threshold voltage cannot be designed and controlled independently of each other. A uniform doping level that yields acceptable depletion depth (and thus short channel effects) may lead to undesirable threshold voltages, and vice versa. A non-uniform channel doping provide additional degrees of freedom to allow the device designer to meet the required specifications for the device.

Popular nonuniform doping designs include the super retrograde doping profiles and halo/pocket implants, which will now be discussed.

Super Retrograde Doping

Sun, et al [70] and Shahidi, et al [69] described the use of a low-high (retrograde) doping profile (Figure 2.10) for decoupling the control of short channel effects from

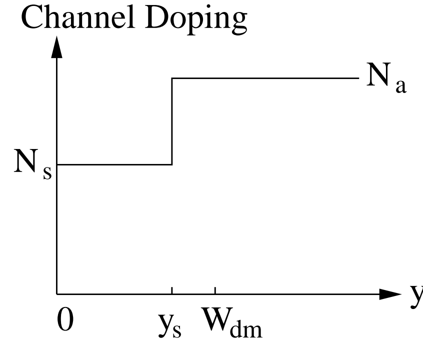


Figure 2.10: An "ideal" Low-High (Retrograde) Doping Profile, with a channel doping of N_s for the first y_s μm, after which the doping rises to N_a in the substrate

the threshold voltage level. The depletion depth is controlled by the substrate doping N_a as follows

$$x_{dmax} = \sqrt{\frac{2\epsilon_s(2|\phi_p| + V_{CB})}{qN_a}} \tag{2.20}$$

The depletion depth in turn determines the severity of short channel effects. At the same time, the surface channel doping N_s allows the threshold voltage to be adjusted more or less independently of the substrate doping.

Halo Doping

The use of halo/pocket implants is another channel engineering technique that can improve short channel effects [37] and thereby control the threshold voltage roll-off [65]. In essence, halo doping mimics the doping profile that led to reverse short channel effects, as discussed in Section 2.2.2. Additional dopants of the same type as the channel doping are implanted into the edge of the channel. Consider an n-channel (p-channel) device. For devices with long channel lengths, the impact of the halo implant is minimal. As the channel length is decreased, the portion of the channel with the increased dopant level becomes more and more significant, leading

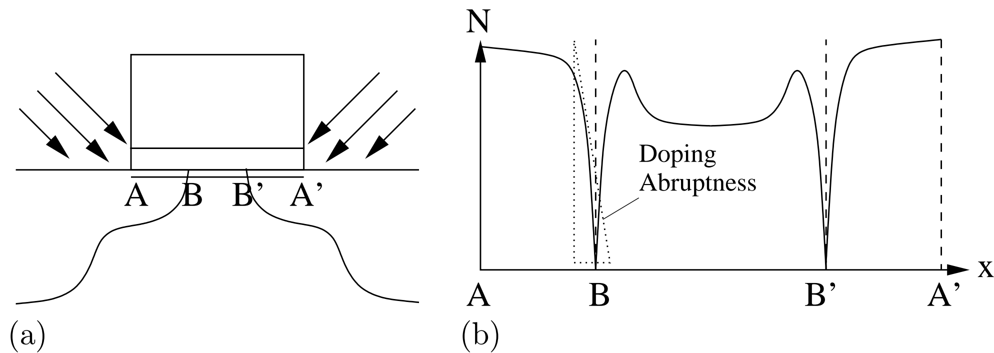


Figure 2.11: Schematic diagram showing a halo doping (a) Device schematic showing the source/drain extension region (b) Doping profile along the line AA'

to an increase in the effective channel doping level with decreasing channel length. This in turn leads to an increase (decrease) in the threshold voltage (equations 2.9 and 2.10), which serves to counteract the decrease (increase) in threshold voltage due to short channel effects. As a result, the threshold voltage roll-off characteristics are improved compared to designs without halo doping.

In Hendriks, et al [36], it is further shown that halo doping can be used to help achieve a reliable device design while minimizing short channel effects. As a result of these benefits, (super-)halo implants has emerged as a dominant technique for controlling short channel effects in the deep sub-micron regime. Examples of modern device designs based on halo doping can be found in Rodder, et al [66] and Taur, et al [75].

2.3.2 Source/Drain Engineering

The channel doping profile is not the only thing important for device performance. The doping profile in the source/drain region can also have a profound impact on both short channel effects (Section 2.2.1) and extrinsic resistances (Section 2.2.3). As

a result, considerable effort has been put into the optimization of the source/drain doping by the semiconductor industry.

This subsection starts with a brief examination of the need for the source/drain extension region in modern device design, followed by a discussion of the available literature on doping abruptness in the source/drain extension region. Much of the rest of this thesis is devoted to better understanding of lateral abruptness and its impact on device performance.

Source/Drain Extension

The shallow source/drain extension is a commonly found feature in modern MOS device designs (Figure 2.11). The shallow source/drain extension was originally motivated by a desire to relieve high electric fields and the resulting breakdown of the device due to hot electron effects [59]; for modern high performance devices however, the source/drain extension is motivated more by the trade-offs between short channel effects and source/drain resistance.

As we saw in Section 2.2.1, short channel effects improve with decreasing junction depths in the source/drain region. However, shallow source/drain junctions would lead to high extrinsic resistances[4], which is detrimental to performance. The source/drain extension presents a shallow effective source/drain junction to the channel region, while at the same time, minimizes the contribution of the source/drain region to extrinsic resistance through the presence of the deep source/drain region.

[4]Recall equation 2.15

Lateral Source/Drain Abruptness

A key parameter in the source/drain extension region that have received considerable attention in literature is the lateral doping abruptness (indicated in Figure 2.11). It is widely believed that an abrupt junction is important for drain current drive [37]. Ng and Lynch [57] showed that the spreading resistance depend on lateral abruptness through a simple analytical calculation. This dependence has since been confirmed through a sheet resistance argument [40], an examination of the quasi-Fermi level in the extension [31], as well as through rigorous resistance calculations based on 2D device simulations [45]. Furthermore, Ghani, et al [30] and Osburn, et al [60] showed that the minimal gate-extension overlap required for proper device operation depends on the lateral abruptness. The amount of overlap is directly related to the parasitic capacitance C_{gsd}, which in turn impacts switching speeds. Finally, lateral abruptness has been shown to affect short channel effects [75]. It is obvious from this discussion that lateral abruptness effects could be very important for device design as technology scaling continues.

Figure 2.12 shows the requirements for lateral source/drain doping abruptness as predicted by the ITRS [1]. There are two sets of numbers: one based on source/drain resistance considerations, calculated as in [57], and another based on short channel effects [75]. The difference between them is substantial, and much of this thesis is motivated by a desire to understand the source of this discrepancy.

Note that traditional wisdom predicts that an abrupt source/drain extension junction improves device performance. On the contrary, for the deep source/drain region, a gradual junction is desired due to junction capacitance considerations [37].

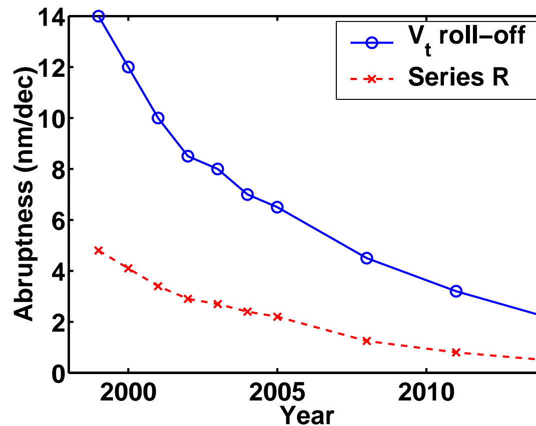


Figure 2.12: Lateral source/drain abruptness requirements according to the ITRS [1], based on threshold voltage roll-off and series resistance considerations respectively.

2.4 Technology Computer Aided Design

The second order effects described in Section 2.2 and the complex device designs needed to control them described in Section 2.3 means that simple first order theory and back of the envelope calculations are no longer sufficient for the design of deep sub-micron devices. Instead, an iterative process is needed.

2.4.1 Traditional Device Design Process

Figure 2.13 shows the typical device design flow. An existing process recipe is modified, through a combination of intuition, experience and/or trial and error, to obtain the processing sequence and the processing conditions required for a new design. These processing steps are then carried out in a fabrication facility to produce actual transistors. Electrical measurements of the manufactured devices are then examined, which in turn provides feedback on possible improvements in the design. This entire process is iterated until the desired specifications are met.

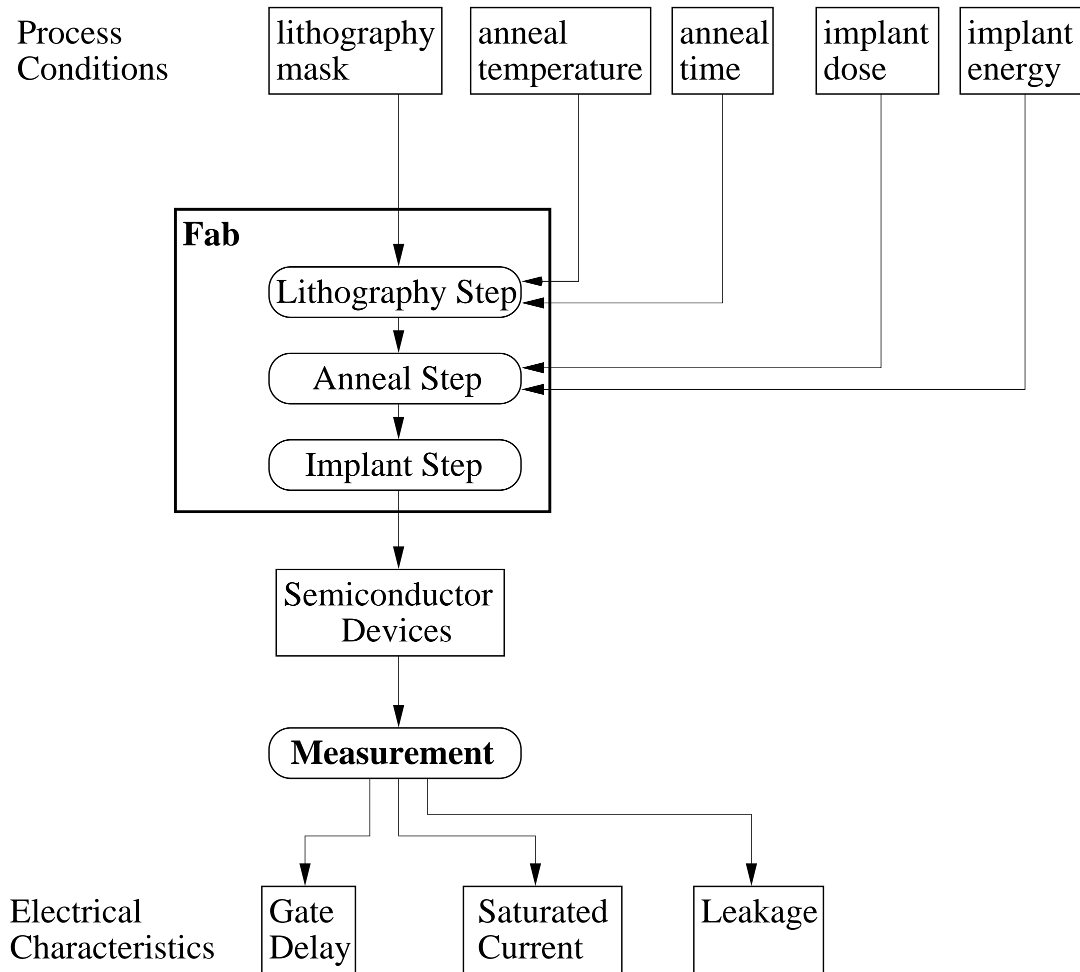


Figure 2.13: Technology Design Flow without Simulation

The problem with this approach is that producing real silicon is expensive, both from a cost and a time viewpoint. Suppose 150 chips[5] could be made from a 300 mm wafer. Assuming a price of $100 per chip, the wafer could lead to sales of $15,000. This is the opportunity cost of using one wafer for experimentation in device design. Furthermore, the turn-around time for passing a wafer through the fabrication process could be measured in days, especially if re-tooling and/or re-calibration of the equipment is needed. These factors makes it very difficult, if not impossible, to conduct a thorough exploration of the design space to obtain an optimal design.

2.4.2 Computer Simulations and Device Design

Technology Computer Aided Design (TCAD) [26] can be a powerful tool for reducing design costs, improving device design productivity and obtaining better device and technology designs (Figure 2.14). While the cost of building a state-of-the art fabrication plant continues to go up, computing power has become a relatively cheap commodity, due to Moore's law and the resulting improvements in price/performance. Instead of going through an expensive and time consuming fabrication process, computer simulations can be used to predict the electrical characteristics of a device design quickly and cheaply. This in turn allows a more thorough investigation of the design space.

TCAD consists of two parts. Process modeling/simulation [34] of the fabrication process, so that physical characteristics such as oxide thickness and doping distribution produced with a given set of processing recipes can be predicted. Device modeling/simulation [7] which can then be used to predict the electrical characteristics of

[5] Assuming a chip area of 3cm^2

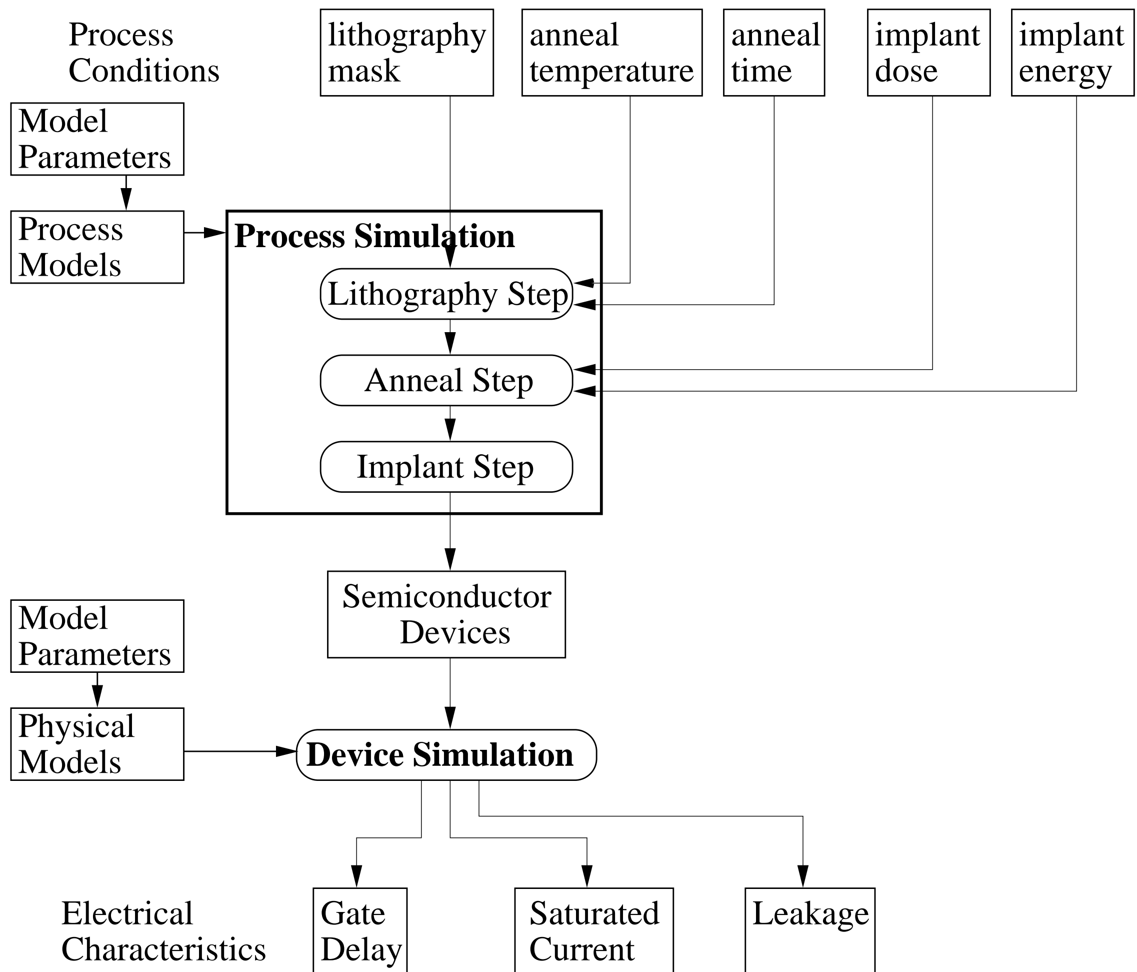


Figure 2.14: TCAD Based Technology Design Flow

the given device structure.

Another important benefit of TCAD is that it can promote physical understanding of how semiconductor devices work. Examination of detailed device operation, such as how the quasi-Fermi level and the spatial carrier (electrons and holes) distribution inside the device varies with biasing conditions, can provide valuable insight into the relationship between a change in process conditions or device design and the resulting impact on device performance. These quantities are often difficult, if not impossible,

to obtain experimentally. In contrast, they are readily available through computer simulations, which directly provides feedback and guidance for device design.

2.5 Challenges facing Technology Computer Aided Design

The previous section presented several reasons for why technology computer aided design has become an indispensable part of semiconductor modeling and design. It is important to note however that accurate TCAD simulations and modeling of physical devices depends critically on calibrated physical models and proper input data[6]. This section examines some of the challenges in obtaining accurate input for calibration (Section 2.5.1) and in selecting the appropriate physical models (Section 2.5.2).

2.5.1 Metrology

Knowledge of the doping profile is important both as a means for calibrating process models, and for providing the starting data for device simulation. Accurate doping metrology of semiconductor devices is a difficult task, however.

According to the ITRS [1], "accurate tools do not exist for dimensional (1D, 2D, or 3D) characterization of sub-100 nm technology." Secondary Ion Mass Spectroscopy (SIMS) [25], the technique used traditionally by the industry for doping characterization, works well only for large area devices. Furthermore, it can only resolve doping variations along one dimension. This is not sufficient for devices in the sub-100 nm

[6]For process simulation, the inputs are the processing conditions. For device simulation, the inputs are the device geometry and doping profiles.

regime, since the full 2D/3D doping distribution is important for the suppression of second order effects, as seen in Section 2.3.

As a result, multi-dimensional doping profile metrology has become an active area of research. Some of the leading metrology candidates include scanning capacitance microscopy (SCM) [89], 2D SIMS [79], inverse modeling [41] [46] [86] and other alternatives [62]. All of these techniques are still in the research phase, and it could be some time before they can be used routinely. Moreover, the achievable resolution of these techniques is still an open question. The lack of a reliable way to measure the doping profiles of semiconductor devices poses a challenge for both accurate device simulation, and accurate calibration of process simulation models.

2.5.2 Physical Models

At the same time, the physical models and the model parameters chosen have a profound impact on simulation results. While a comprehensive treatment of physical models is beyond the scope of this thesis, in this section a brief discussion of the challenges posed by the complex physical models needed for modeling deep sub-micron devices is presented.

Process Models

In theory, process simulations can be used to obtain the doping profiles when accurate metrology is not available. In reality, the difficulty in obtaining accurate multi-dimensional doping profiles of a fabricated device (Section 2.5.1) makes it difficult to obtain well-calibrated models for semiconductor process simulation. This in turn limits the achievable accuracy and reliability of the simulated results. Further complicating matters is the complexity of the processing and the underlying physical

mechanisms.

An example of this complexity involves the interactions between various atomic species [3] and defect structures in the crystal [43] [13] during the dopant diffusion process. Failure to take these effects into account can lead to an underestimation of junction depth and ultimately, an overly optimistic prediction of achievable device performance. Yet the distribution of such defect structures, such as dislocation loops and the profiles of interstitials and vacancies in the device crystal, is often not known. Other examples of model complexity include dopant dose loss (the result of pile-up at the substrate-dielectric interface and diffusion of the dopants into the dielectric [39]); and the electrical activation of chemical dopants in the crystal (in itself a challenging process to model [67] [51]).

The complexity of the processing physics and chemistry and the difficulty in obtaining accurate doping metrology, combine to make process simulation a challenging enterprise. Complicated models with significant number of parameters are required, and in practice, the model parameters are tuned iteratively until the simulated electrical behavior matches the electrical measurements. This is a time consuming, labor intensive and less than fool-proof process in the hands of non-experts.

Mobility Models

Even if the doping profile is known, either through direct doping metrology or process modeling, other challenges to the prediction of device performance remain. An example of these challenges is the evaluation of carrier mobility.

A modern mobility model must take into account surface roughness, scattering due to ionized dopant atoms, mobile carriers, acoustic phonons and optical phonons in the semiconductor crystal. Furthermore, the amount of scattering is dependent on

Mobility Model	R_{sd} (Ω)
UNIMOB [7]	16.7
TFLDMOB [80]	20.9
HPMOB [14]	24.1
LSMMOB [48]	18.1
GMCMOB [55]	28.5
LUCMOB [23]	20.6

Table 2.3: Source/drain resistance calculated with several different mobility models for the same device

dopant and carrier concentrations, as well as the electric fields present in the device. These combine to make the modeling of carrier mobility difficult.

To see the impact that the mobility model can have on device simulations, consider Table 2.3, which shows the source/drain resistance calculated using several different mobility models for the same device. The resulting resistance varies by more than 50%. It is obvious that the proper choice and calibration of mobility models is important for the accurate prediction of device performance.

Quantum Mechanical Effects

Further complicating matters, as discussed in Section 2.2.4, quantum mechanical effects must also be accounted for in deep sub-micron devices.

Quantum mechanical effects can be modeled by solving Schroedinger's equation and Poisson's equation for a device self-consistently [2]. However, this rigorous approach is computationally expensive and difficult to apply in day-to-day device design.

A popular approach for modeling quantum mechanical effects is to use instead the solution of the simplified, one-dimensional problem to motivate a simplified model, then provide fitting parameters to match the measurement data for real devices. Examples include the Hansch model [33], which models charge quantization effects

through the concept of effective density of states; the Van Dort band-gap widening approach [81]; and the hybrid model as described by Choi, et al [16].

The problem with these models is that they are inherently one-dimensional, and as such, do not provide an adequate representation of reality. The fitting parameters may need to be set far away from their theoretical, physical values to match the measured data. Furthermore, they apply only in specific regions in a device, and require arbitrary transitions from the quantum to the classical regimes [81]. These arbitrary transitions, together with their one-dimensional nature, can sometimes cause convergence problems for numerical solutions [35].

The density gradient model [5] is a very promising approach that provides a nice compromise between accuracy and computational efficiency. The density gradient model is a first order solution to the Schroedinger's equation, and has a firm and rigorous theoretical basis[7]. Furthermore, density gradient theory is inherently multi-dimensional. However, for the purposes of this dissertation, a stable implementation of this model is not yet available, and a simple approach based on effective oxide thickness is used instead[8].

2.6 Summary

This thesis is impacted by the challenges presented in this chapter in many ways. The main motivation of the thesis comes from a desire to understand how lateral abruptness in the source/drain extension region relate to the second order effects

[7]Note that density gradient theory is still to some extent a curve fitting model due to the difficulty in modeling the oxide and the gate stack properly [5]

[8]The impact of this modeling choice should be revisited. This is listed as future work in Section 6.2.

described in Section 2.2. At the same time, this work has to deal with the simulation and modeling challenges discussed in Section 2.5.

In this thesis, these modeling challenges are handled in various ways. Attention is paid to ensure proper definitions of the threshold voltage and resistance components are used. Moreover, both short channel effects as well as doping effects are considered. Starting simple, uniform channel doping is examined first, before considering the impact of halo doping, which is important for modern deep sub-micron devices. The use of TCAD simulations allows a thorough study of lateral abruptness that would be difficult from the time, resource and technical perspectives. The lateral abruptness of a parameterized device description is varied directly to avoid the challenges of metrology and process simulations. The impact of mobility models are examined by running the simulations with multiple mobility models to ensure the central conclusions remain consistent with simulation results.

The next chapter examines the findings on the impact of lateral abruptness on device performance. Note that these findings depend on a rigorous algorithm for calculating resistance components as well as supporting software for processing the simulation results. The details of these will be presented in subsequent chapters.

Chapter 3

Study of Lateral Abruptness

3.1 Introduction

As discussed in Section 2.3.2, lateral abruptness in the source/drain extension region is expected to have an important impact on the source/drain resistance, short channel effects and drive current of a MOS device. At the same time, differences between the two sets of lateral doping abruptness requirements predicted by the ITRS [1] (Figure 2.12) are substantial. It is therefore important to re-examine the basis for these numbers and to reconcile their differences.

This chapter presents the results of a detailed simulation study of lateral doping abruptness, gate-extension overlap length and their overall impact on device performance. The methodology used and the important issue of proper metrics for comparing device technologies is first discussed in Section 3.2. Next the impact of lateral source/drain abruptness is examined from several viewpoints: series resistance (Section 3.3), threshold voltage roll-off (Section 3.4) and I_{on}-I_{off} characteristics (Section 3.5). Finally, the impact of halo doping, the assumption that the source/drain

extension doping profile can be described as a tensor product, and the impact of mobility models, oxide thickness and detailed doping design on the impact of lateral abruptness are examined in Section 3.6.

3.2 Methodology

3.2.1 Simulation Details

Device simulation is well suited to the study of lateral abruptness of the source/drain extension, since it allows precise control of the doping profiles of the devices, which are difficult to ascertain experimentally. The device parameters, shown in Table 3.1, are chosen based on a 50 nm L_{gate} device according to the ITRS roadmap [1], corresponding to the 2008 technology node. The two-dimensional doping profile of the extension is assumed to be well-described by a function of the form

$$N(x, y) = N \cdot f_y(y) \cdot f_x(x) \tag{3.1}$$

To simplify the definition of "lateral abruptness", exponentially varying doping is assumed at the junctions. To simplify the simulations, and to isolate the effects of the lateral abruptness, a uniform channel doping is used, and the vertical doping profile of the extension (away from the tip) is held constant[1] as the lateral abruptness is varied. The lateral abruptness is modified by changing the lateral to vertical ratio from 0.3 to 2.0, causing the lateral abruptness to vary from 1.9 nm/dec to 13.0 nm/dec. The Lucent mobility model [23] is used in the simulations[2].

[1] An exponentially varying doping of 6.5 nm/dec is used as the vertical doping profile in the source/drain extension

[2] Note the choice of mobility models can have a significant impact on the simulated values.

V_{dd} (V)	0.9
Lgate (nm)	50
Poly Doping (cm^{-3})	5.4×10^{20}
$t_{ox,eff}$ (nm)	1.5
Extension X_j (nm)	20
S/D Extension Peak Doping (cm^{-3})	2.6×10^{20}
Substrate Doping (cm^{-3})	7.5×10^{18}
Sidewall Spacer (nm)	40
Contact Size (nm)	80
Contact Resistivity ($\Omega - cm^2$)	5×10^{-8}
Target V_t (V)	0.35

Table 3.1: Parameters for Simulated Devices, chosen according to the 2008 technology node on the 1999 ITRS roadmap

Note that in reality, super-halo profiles [75] rather than uniform channel doping are employed in deep sub-micron devices, and the source/drain implant resemble a Gaussian rather than an exponential function. Nevertheless, the above simplifications are useful for gaining physical insight. (We will revisit this issue in Section 3.6.)

3.2.2 Metric for Comparing Device Technologies

An important issue for comparing device technologies is the choice of a proper metric. The ITRS uses series resistance and threshold voltage roll-off to compare devices with different lateral abruptness. However, neither of these by itself is a good metric for comparing device technologies. Series resistance does not take I_{off} into account, while threshold voltage does not reflect the impact of series resistance[3], which can be important for I_{on}. Both these quantities only provide a partial picture of the overall device performance. This is the fundamental reason behind the discrepancy between

[3]Note depending on the extraction method, series resistance could affect the extracted threshold voltage

the two sets of ITRS abruptness numbers (Section 3.1).

I_{on}-I_{off} curves provide an alternative, widely used way for comparing device technologies. They reflect both power consumption (leakage) and device performance (circuit delay) and thus present a more complete picture of digital device performance. Connelly and Foisy [20], however, raised two criticisms of the conventional I_{on}-I_{off} curve. First, it assumes that gate length is a free parameter that can be used to tune device performance (for instance, to achieve off-current targets). This is not consistent with lithography-limited processes: for a typical technology design, the target gate length is the starting point; other device parameters (such as the channel doping) are then chosen to maximize the drive current while satisfying constraints such as maximum I_{off}. Therefore, Connelly and Foisy suggest using channel doping instead of channel length as the free parameter to obtain the nominal I_{on}-I_{off} plots.

The second criticism is that the conventional I_{on}-I_{off} curve fails to account for the impact of process variations in gate length on circuit performance. In Connelly and Foisy [20], it is shown through a statistical argument that the average leakage of the nominal device can be estimated by considering the leakage of a sub-nominal device that is 0.78 standard deviations shorter than the nominal case. At the same time, the delay of a chain of devices with nominal gate length can be estimated by considering the delay of a super-nominal device that is 1.6 standard deviations longer than nominal. Accordingly, plotting the "super-nominal" I_{on} versus the "sub-nominal" I_{off} gives a better prediction of actual circuit performance than either the conventional or the nominal I_{on}-I_{off} curves. This can also be seen from the fact that while halo/pocket implants improve the device performance overall, this is not reflected in the conventional I_{on}-I_{off} curve [65]. On the other hand, the advantage of the halo implant is obvious from the super-nominal I_{on} versus the sub-nominal I_{off}

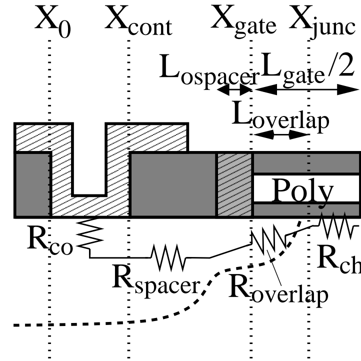


Figure 3.1: Schematic of a half MOS device, together with the major resistive components

plots [20].

The impact of lateral abruptness will now be examined from these different viewpoints: series resistance, threshold voltage roll-off, conventional I_{on}-I_{off}, nominal I_{on}-I_{off} and super-nominal I_{on} versus sub-nominal I_{off} plots. While the first two metrics are incomplete, they provide valuable insight into device operation. The I_{on}-I_{off} plots allow us to draw several conclusions regarding the effect of lateral abruptness and gate-extension overlap length. At the same time, this simulation study provides an excellent example and comparison of the various I_{on}-I_{off} plots.

3.3 Series Resistance

Figure 3.1 shows a schematic diagram of one-half of a typical MOS device and the major resistive components in it. Note the presence of the "overlap spacer" (of length $L_{ospacer}$), which is used to tune the gate-extension overlap length ($L_{overlap}$).

abruptness (nm/dec)	R_{co}	R_{spacer}	R_{ov}	R_{chan}	I_{on} (μm)	I_{off} (nm)
1.9	135	27	64	199	630	51.0
3.3	135	27	73	213	572	14.5
4.5	135	28	81	220	538	6.9
6.5	135	29	92	222	513	4.0
9.8	135	32	108	214	505	3.5
13.0	135	47	117	201	512	4.8

Table 3.2: Resistive components for devices with L_{gate} = 50 nm and various lateral source/drain abruptness.

Table 3.2 shows the resistive components calculated using

$$R_{sh}(x) = \frac{d\phi(x)}{dx} \Big/ I_{ds} \tag{3.2}$$

$$\begin{aligned} R_{cont,s} &= \frac{\phi_{x=x_{cont,s}} - V_s}{I_{ds}} \\ R_{cont,d} &= \frac{V_d - \phi_{x=x_{cont,d}}}{I_{ds}} \end{aligned} \tag{3.3}$$

where ϕ is the quasi-Fermi level obtained from device simulation[4]. The metallurgical overlap between the gate and the source/drain extension is kept constant at 14 nm.

Table 3.2 shows the resistive components of devices with gate length of 50 nm and various lateral abruptness. The overlap resistance improves from 117Ω to 64Ω as the lateral source/drain abruptness increases from 13.0 nm/dec to 1.9 nm/dec[5]. A corresponding increase in the on-currents can be expected as the lateral source/drain

[4] ϕ_n is used for NMOS device. Note these calculations agree with the rigorous two-dimensional resistance calculations described in [45].

[5] Note that for devices with very gradual junctions, the lateral slope extends beyond the overlap region, causing an increase in R_{spacer} as well, further degrading device performance.

abruptness is increased. This is confirmed by Table 3.2[6].

The problem with using source/drain resistance for comparing devices with different abruptness is that off-current is ignored. As Table 3.2 demonstrates, when the lateral abruptness of the junction is increased from 6.5 nm/dec to 1.9 nm/dec, the on-currents improve by 22.7% while the off-currents degrade by over an order of magnitude. Using source/drain resistance as the sole criterion in comparing these devices ignores this trade-off[7]. This in turn will lead to an overestimation of the benefits of having an abrupt junction. It is therefore concluded that the set of abruptness requirement numbers on the ITRS roadmap [1] based on the series resistance calculations (as developed by Ng and Lynch [57]) in Figure 2.12 are overly stringent.

We will now examine the second metric used by the ITRS: threshold voltage roll-off.

3.4 Threshold Voltage Roll-Off

Taur [75] [72] showed that very gradual lateral junctions in the source/drain extension regions could degrade the threshold voltage roll-off of a given technology. The impact of lateral abruptness on threshold voltage roll-off will now be examined in greater detail.

Figure 3.2 shows the threshold roll-off characteristics of simulated devices (with device parameters as shown in Table 3.1) with lateral doping abruptness of 1.9 nm/dec, 6.5 nm/dec and 13 nm/dec. The other device parameters are as described in Table

[6]The behavior of the devices with the most gradual slopes is affected by the threshold voltage roll-off, a fact that will be examined further in Section 3.4

[7]Much of the discussion in literature on lateral abruptness, such ad Hori and Mizuno [37] ignores this trade-off as well.

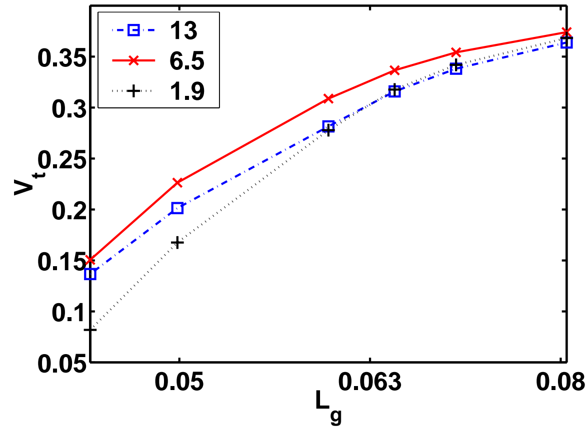


Figure 3.2: Plot of threshold voltage versus L_g (threshold roll-off characteristics)

3.1. The maximum g_m definition (Section 2.1.2) is used to define the linear region threshold voltage

$$V_{t,lin} = V_t(V_{ds} = V_{d,lin}) \tag{3.4}$$

The excessive threshold roll-off observed in the sub-50 nm devices is due to the use of uniform channel doping (no halo doping). For practical devices with these device dimensions, a carefully designed, highly non-uniform doping profile is needed. Nevertheless, the qualitative trends based on uniform doping provide important insight[8].

It is clear from Figure 3.2 that threshold voltage roll-off is degraded by lateral source/drain junction gradients that are too gradual ($>$ 6.5 nm/dec), consistent with Crabbe [22] and Taur [75], as well as those that are too abrupt ($<$ 3.3 nm/dec), which is not consistent with existing literature on the topic.

Figure 3.3 shows an alternative view of the data. By plotting the threshold voltage in the linear region against lateral source/drain abruptness for devices with the same gate lengths, it is again obvious that the threshold voltage roll-off is degraded by

[8]The issue of halo doping will be revisited in Section 3.6.1

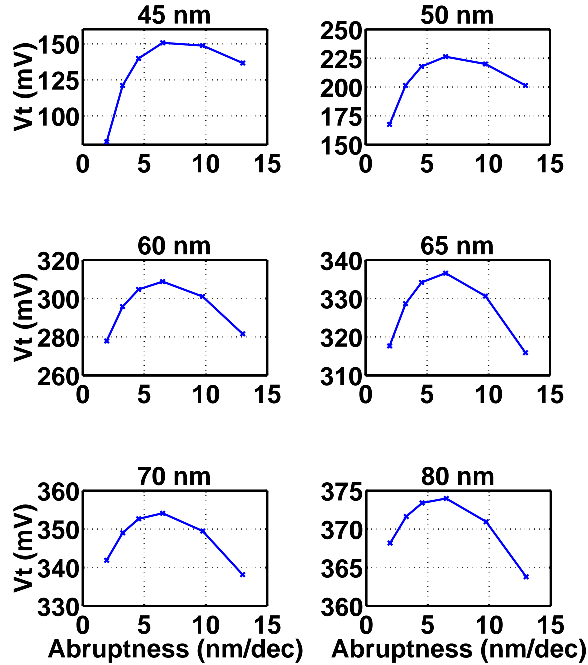


Figure 3.3: Plot of threshold voltage in the linear region versus lateral abruptness of the source/drain extensions for devices with gate lengths ranging from 45 nm to 80 nm

both source/drain extension junctions that are too gradual laterally and those that are too abrupt. This behavior can be explained by considering two competing effects: counter-doping and charge sharing.

3.4.1 Counter-doping

The degradation of threshold roll-off caused by very gradual junctions is due to counter-doping of the channel by the tails of the source/drain gradients [75] [72]. Figure 3.4a shows four different donor doping profiles in the source/drain extension, representing lateral source/drain abruptness of 13 nm/dec down to 4.5 nm/dec. Figure 3.4b shows the net doping that results from adding these donor profiles to the

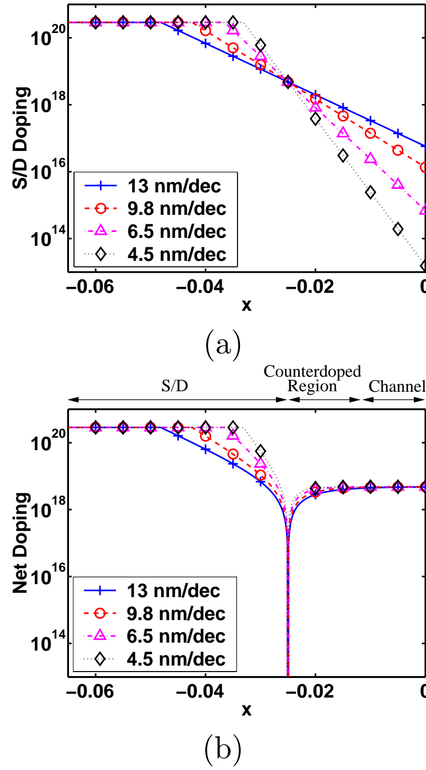


Figure 3.4: (a) Donor doping profile plots along the $Si-SiO_2$ interface, for extensions with lateral abruptness ranging from 13 nm/dec to 4.5 nm/dec (b) Net doping plots along the $Si-SiO_2$ interface, showing counter-doping clearly

uniform channel doping of $4.8 \times 10^{18} cm^{-3}$. The tail of these source/drain donor profiles (of finite lateral abruptness) extends into the channel and counter-dopes the edge of the channel. This in turn causes the threshold voltage to drop by lowering the net channel doping. The more gradual the junction gradient, the more severe counter-doping is, leading to increased degradation in the threshold voltage roll-off.

As Taur showed in [72], this counter-doping effect is analogous to the halo effect [75], with the existence of a non-uniform lateral doping leading to shifts in the device threshold voltages.

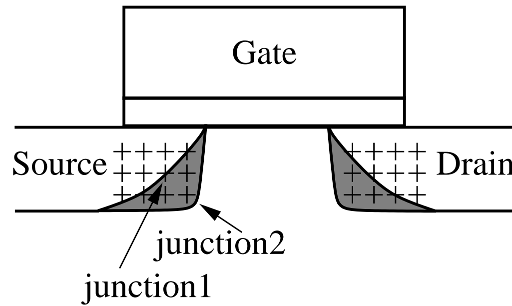


Figure 3.5: Device Schematic showing two source/drain junctions with different lateral abruptness. Junction 2 have the more abrupt doping.

3.4.2 Charge Sharing

To understand how junctions with very abrupt lateral gradients can degrade threshold voltage roll-off, recall from Section 2.2.1 that increasing the junction depth of the source/drain region exacerbates short channel effects.

Figure 3.5 shows schematically two source/drain junctions with different lateral abruptness. Assuming the vertical doping abruptness is finite and equation 3.1 holds, increasing lateral junction abruptness of the source/drain extension will lead to a more box-shaped profile (as depicted by "junction 2" in the figure). The dopant in the shaded area of the more abrupt device will lead to an increase in the "effective junction depth" seen from the channel. This will in turn aggravate short channel effects, leading to the degradation in threshold roll-off for the very abrupt junctions shown in Figures 3.2 and 3.3.

Drain induced barrier lowering (DIBL) is another manifestation of the short channel effects. Figure 3.6 shows a plot of the amount of drain induced barrier lowering for devices with lateral abruptness ranging from 1.9 nm/dec to 13 nm/dec and various gate lengths. The threshold voltage at high drain bias is defined using the "critical-current at linear threshold" approach as described in Section 2.1.2. DIBL is defined

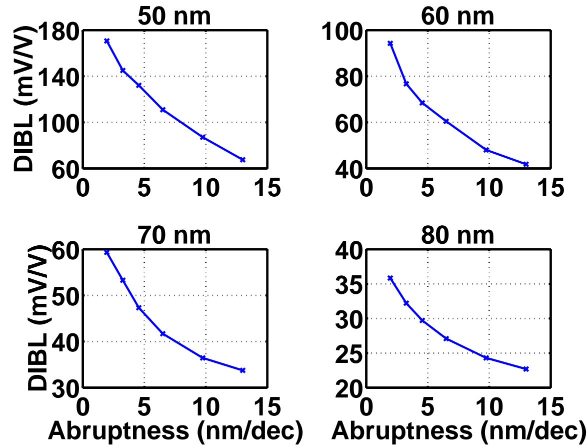


Figure 3.6: Plot of DIBL versus lateral source/drain doping abruptness for devices with various gate lengths. DIBL is defined as in equation 2.13

as described in Section 2.2.1. Note that the linear drain voltage V_{dlin} is taken to be 0.05V. It is apparent from Figure 3.6 that the more abrupt junctions exhibit more severe drain induced barrier lowering effects. This can be explained by noting that an increase in (effective) junction depth also degrades DIBL (Section 2.2.1).

In summary, there are two problems with the use of threshold voltage roll-off as a metric by the ITRS[1]. Firstly, as mentioned in Section 3.2.2, threshold voltage roll-off is only a partial measure of device performance and does not take series resistance into account. Secondly, the assumption that increasing abruptness always improves threshold voltage roll-off is not correct. Therefore neither series resistance nor threshold voltage roll-off is an appropriate device performance metric.

We now proceed to examine the use of I_{on}-I_{off} plots for comparing device technologies with different lateral abruptness. They take into account both threshold voltage and series resistance and as such provide a more complete picture of device performance.

3.5 On-currents and off-currents

The conventional I_{on}-$I_{\rm off}$ plots, which are widely used in industry, will be examined first. The refinements discussed in Section 3.2.2 is then applied to examine the impact of lateral abruptness on device performance.

3.5.1 Conventional I_{on}-$I_{\rm off}$ characteristics

Figure 3.7 shows the conventional I_{on}-$I_{\rm off}$ plot for different device designs. The free parameter along each curve is the gate length of the devices and ranges from 45 nm to 80nm. Note that in the conventional I_{on}-$I_{\rm off}$ plot, the target off-current is achieved by adjusting the channel length while keeping all other device parameters constant. As noted in Section 3.2.2, this may not be consistent with a typical technology design scenario. To generate the different device designs, gate-extension overlap length is varied from 14 nm to -6 nm, while the lateral gradient of the source/drain extension doping is varied from 1.9 nm/dec to 13 nm/dec. For the same target off-current, drive current improves with increasing lateral abruptness.[9] As noted in Section 3.3 however, the improvement is significantly less than would be expected from considering source/drain resistance or on-current alone. The improvement in I_{on} by increasing abruptness from 13 nm/dec to 1.9 nm/dec for devices with a constant $I_{\rm off}$ value of 100 nA is less than 5% (Table 3.3b), a substantially smaller improvement than the 22.7% improvement in I_{on} observed in Table 3.2.[10]

Note also that for devices with sufficient gate-extension overlap, as the gate-extension overlap length is decreased, the devices follow approximately the same

[9] Increasing lateral abruptness shifts curve to the right.
[10] Where no attempt is made to keep $I_{\rm off}$ constant.

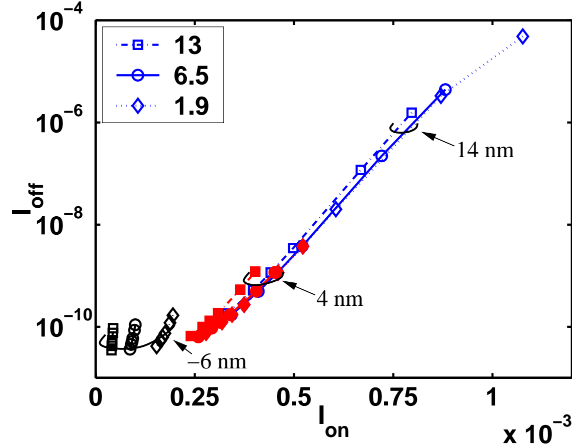


Figure 3.7: Conventional I_{on}-I_{off} plot for devices with gate lengths ranging from 45 nm to 80 nm, lateral abruptness of the extension from 13 nm/dec to 1.9 nm/dec, and gate-extension overlap of 4 nm (solid symbols), 14 nm and -6 nm (open symbols). Channel doping is 4.8×10^{18}. The mobility model used is LUCMOB.

trend line. In this regime, decreasing the gate-extension overlap length while keeping the gate length constant simply serves to increase the metallurgical channel length. Eventually, as the gate-extension overlap is decreased further, the I_{on}-I_{off} performance starts to degrade, due to degradation of the coupling between the channel and the source/drain regions. This is especially pronounced in devices that have relatively gradual lateral abruptness, suggesting that devices with abrupt junctions are better able to tolerate small gate-extension overlap. This is consistent with the results reported by Osburn, et al [60].

Thompson, et al [77], states that a significant, minimum overlap is needed for satisfactory I_{dsat}. While at first glance this may seem inconsistent from the results presented previously, it is important to note that the overlap length of a real device is difficult to determine from experimental data. In the paper, the overlap length of each device is estimated from the point where the curve of C_{miller} versus offset spacer

width “flattens out”. In reality, this transitional point is not clear-cut. As a result, there is considerable uncertainty in the minimum overlap requirement estimate. This is the key to reconciling those results with this thesis.

3.5.2 Nominal I_{on}-Nominal I_{off} Plots

The nominal I_{on}-nominal I_{off} plot is obtained by using channel doping rather than gate length as the free parameter. Figure 3.8 shows the nominal I_{on}-I_{off} plots for devices with gate lengths of 70 nm and 50 nm. As the channel doping is varied, care is taken to maintain the same junction depth and sheet resistance in the source/drain regions by modifying the source/drain doping profiles. (Note that in this case the results do not change substantially if the same source/drain profile is used instead for all channel dopings.) Device designs with various lateral abruptness and gate-extension overlap length are included. Figure 3.8a shows that for the 70 nm devices, performance improves with increasing overlap. This is the result of both the improved source/drain-to-channel coupling and the shorter channel lengths implied by increasing overlap for a constant gate length. Note that devices with shorter channel lengths tend to have higher performance (this is the reason for device scaling).

The behavior of the 50 nm devices is somewhat different (Figure 3.8b). While performance still improves with increasing overlap for devices with substantial “underlap” (due to improved source/drain-to-channel coupling), increasing the gate-extension overlap for devices with sufficient overlap no longer yields improvements in device performance. For these ultra short devices, with the chosen device design parameters, short channel effects are severe and they tend to dominate. As a result, no further improvements in device performance can be expected from scaling down

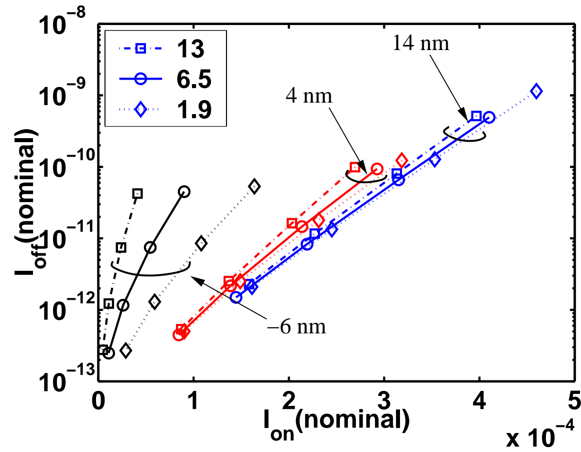
the channel. Using the channel doping as the free parameter in establishing the I_{on}-I_{off} plots presents a better picture of the limits of channel length scaling for a given technology[11] than the conventional plots.

Table 3.3a shows the I_{on} obtained for devices with various lateral source/drain abruptness and a specific target I_{off} value. In the "conventional case" (Figure 3.7), devices with channel doping of $4.8 \times 10^{18} cm^{-3}$ are considered, and the target I_{off} of 100 nA is obtained by tuning the gate length of the devices. For the "nominal 50 nm" case (Figure 3.8b), the target I_{off} of 100 nA is obtained by tuning the channel doping instead. Similarly, for the "nominal 70 nm" case, the target I_{off} value of 100 pA for the 70 nm case is obtained by tuning the channel doping. Since the target I_{off} values are not the same, it is not meaningful to compare the I_{on} obtained here for the 50 nm and 70 nm devices. Instead, the focus is on the changes in I_{on} due to varying junction abruptness.

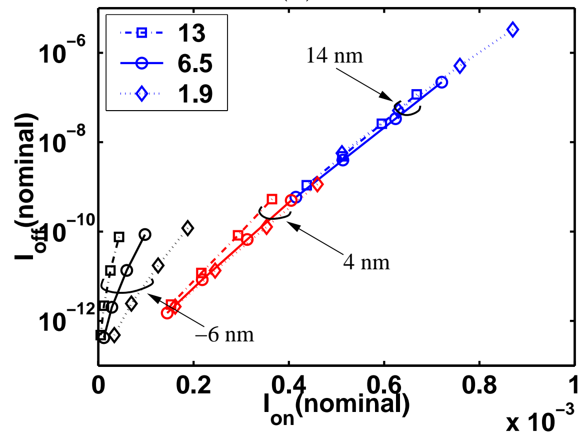
For both the conventional case and the nominal 70 nm devices, increasing lateral abruptness of the source/drain extension doping leads to an improvement in device performance. The nominal 50 nm devices on the other hand show an optimum abruptness value of 6.5 nm/dec. For these ultra short devices, short channel effects dominate. The degradation in off-currents due to the lowering of threshold voltages (Section 3.4) from ultra abrupt junctions more than compensates for any improvements in on-currents.

---

[11] With given oxide thickness, junction depth and other key design parameters



(a)



(b)

Figure 3.8: Nominal I_{on} versus nominal I_{off} plot for devices with gate lengths of (a) 70 nm (b) 50 nm. Lateral abruptness of the extension varies from 13 nm/dec to 1.9 nm/dec, gate-extension overlap for the devices ranges from 14 nm to -6 nm, while channel doping varies from 4.8×10^{18} to 9.0×10^{18}. The mobility model used is LUCMOB.

Abruptness	Conv.	Nominal 50 nm	Nominal 70 nm
13 nm/dec	6.605	6.605	3.23
9.8 nm/dec	6.746	6.743	3.30
6.5 nm/dec	6.821	6.802	3.35
4.5 nm/dec	6.852	6.794	3.38
3.3 nm/dec	6.871	6.755	3.39
1.9 nm/dec	6.887	6.676	3.42

(a)

Abruptness	Conv.	Nominal 50 nm	Nominal 70 nm
13 nm/dec	-4.09%	-2.89%	-5.40%
9.8 nm/dec	-2.05%	-0.87%	-3.40%
6.5 nm/dec	-0.96%	0.00%	-1.94%
4.5 nm/dec	-0.51%	-0.12%	-1.15%
3.3 nm/dec	-0.24%	-0.69%	-0.61%
1.9 nm/dec	0.00%	-1.85%	0.00%

(b)

Table 3.3: (a) I_{on} (in units of $10^{-4}A/\mu m$) for devices with $I_{\text{off}} = 10^{-7}A$ (or $10^{-10}A$ for $L_g = 70nm$) and different abruptness. (b) Percentage deviation of I_{on} from the device with the best case abruptness. Gate-extension overlap = 14 nm.

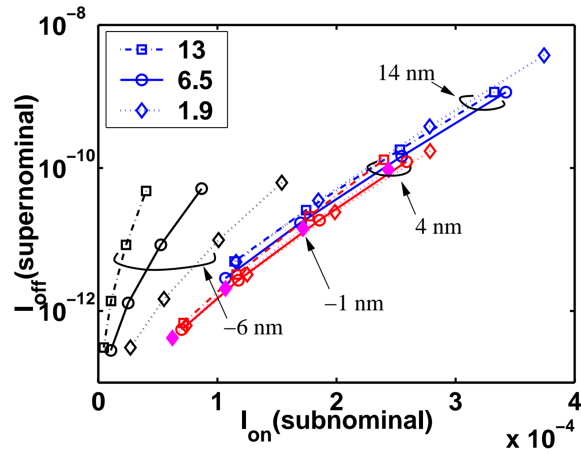
3.5.3 Supernominal I_{on}-Subnominal I_{off} Plots

As discussed in Section 3.2.2, statistical variations of gate length can be incorporated into the I_{on}-I_{off} plots by considering the I_{on} of the supernominal device and the I_{off} of the subnominal device. This turns out to have a major impact on device design.

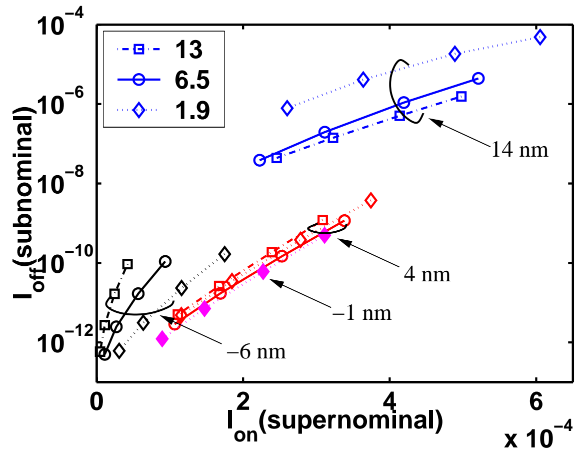
Figure 3.9 shows the super-nominal I_{on} versus sub-nominal I_{off} curves for the same set of device designs used in earlier discussions. The sub-nominal device is assumed to have a gate length that is 5 nm shorter than the nominal device, while the super-nominal device is assumed to have a gate length that is 10 nm longer than the nominal case.[12]

Gate-extension overlap length is shown to have a significant impact on device performance. For each gate length and device design, there exists an optimal overlap under this metric. A major effect of taking process variations into account is to highlight the impact of short channel effects, which in turn causes the apparent "optimal" channel length to increase. For devices with gate length of 50 nm, very abrupt junctions with a slight "underlap" have the best performance. The severity of the short channel effects for these devices means the increase in source/drain resistance and the degradation in source/drain-to-channel coupling due to the underlap are more than compensated for by the lowering of short channel effects due to the increase in metallurgical channel length. The 70 nm devices experience much less severe short channel effects. A small overlap of approximately 4 nm is desirable in this case. These results suggest that an "overlap spacer", which allows tuning of the extension overlap, would be useful for device design.

[12] These values are appropriate for an earlier generation of devices described by Connelly and Foisy [20].



(a)



(b)

Figure 3.9: Plot of supernominal I_{on} versus subnominal I_{off} for devices with nominal gate lengths of (a) 70 nm (b) 50 nm. Gate-extension overlap for the devices ranges from 14 nm to -6 nm (solid symbols represent -1 nm case), while channel doping varies from 4.8×10^{18} to 9.0×10^{18}. The mobility model used is LUCMOB.

It is also shown that the impact of lateral abruptness on device performance depends on the amount gate-extension overlap. An abrupt junction is desirable for devices with insufficient overlap (for instance the -6 nm case), due to the resulting decrease in the source/drain resistance. On other hand, for devices with substantial overlap (such as the 14 nm case), a more abrupt junction actually hurts device performance, due to the degradation of charge sharing effects. Note that this is in contrast with the results in Figures 3.7 and 3.8, which does not accentuate short channel effects to the same degree.

3.6 Revisiting Key Simulation Assumptions

3.6.1 Halo Doping

Uniform channel doping is assumed for the devices considered thus far. However, for deep sub-micron devices, a halo/pocket implant is needed to maintain acceptable short channel behavior [75]. The impact of introducing halo doping on understanding lateral abruptness effects will now be examined. In particular, the focus will be on the threshold roll-off characteristics of devices with different lateral abruptness (analogous to Section 3.4).

The source/drain doping are described by equation 3.1, with f_y and f_x defined as follows (similar to [8])

$$f_y(y) = e^{\left(\frac{y-y_{peak}}{\sigma_{y,ext}}\right)^2} \tag{3.5}$$

$$f_x(y) = \begin{cases} e^{\left(\frac{x-x_{peak}}{\sigma_{x,ext}}\right)^2} & : \quad x > x_{peak} \\ 1 & : \quad x \leq x_{peak} \end{cases} \tag{3.6}$$

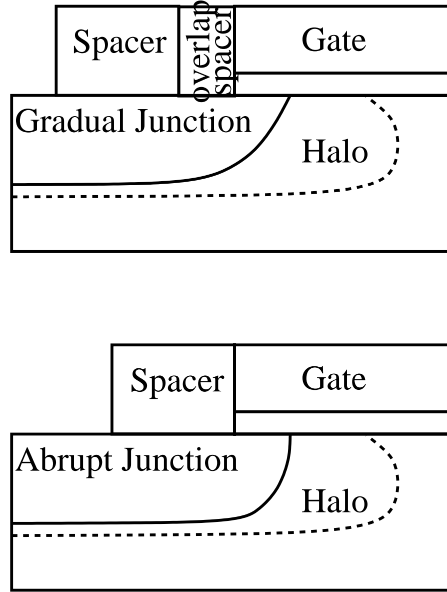


Figure 3.10: Overlap spacer is used as a tunable parameter. Metallurgical channel length (L_{met}) is kept constant for different lateral abruptness by varying overlap spacer length while gate length (L_{gate}) remains constant.

The basic source/drain parameters are taken from the Well-Tempered MOSFET[6]. The metallurgical channel length is kept constant at 22 nm, and $\sigma_{y,ext}$ is 17 nm for all devices. The gate is used as the mask for the halo implant, so the halo location is fixed relative to the gate edge.

Two scenarios are considered. In the first (Figure 3.10), the gate length is kept constant at 50 nm. As a result, the location of the halo is fixed. A constant metallurgical channel length for devices with different lateral abruptness is maintained by adjusting the overlap spacer. In the second scenario (Figure 3.12), the overlap spacer is not used as a tunable parameter. A constant metallurgical channel length is achieved by changing the gate length. As a result, the location of the halo varies for the different devices. Note that this second scenario is the typical scenario considered in literature in the context of lateral abruptness effects.

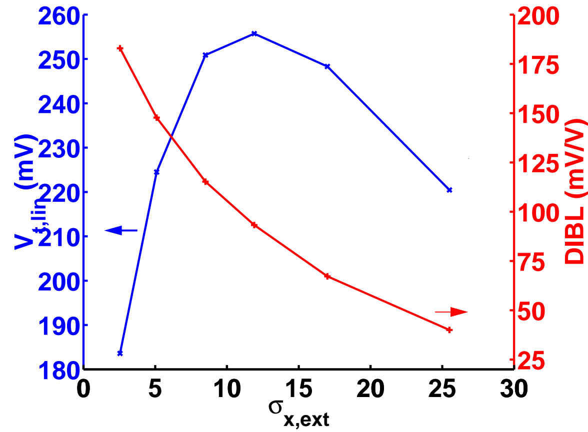


Figure 3.11: Threshold Voltage and DIBL for devices with fixed halo location (Figure 3.10). The lateral abruptness of the extension junction is varied by controlling the characteristic length of the Gaussian profile.

Figure 3.11 shows the threshold voltage and DIBL for devices in the first scenario. The threshold voltage roll-off is degraded by both lateral doping gradients that are too abrupt or too gradual, a result of the interplay between counter doping and charge sharing effects, similar to Section 3.4. The degradation of DIBL effects as the lateral gradient is increased is obvious, indicating that charge sharing effects degrades with increasing junction gradient.

Figure 3.13 shows the threshold voltage and DIBL for devices in the second scenario. Contrary to the above, the threshold voltage roll-off improves monotonically with increasing junction abruptness (decreasing $\sigma_{x,ext}$). This is consistent with the results in [72], and forms the basis for the claim that short channel effects improve with increasing abruptness. Contradicting this claim, DIBL effects are clearly degraded as the lateral gradient increases (Figure 3.11b). This indicates that charge sharing effects still degrades with increasing lateral junction abruptness.

The reason for the continual improvements in threshold roll-off as abruptness

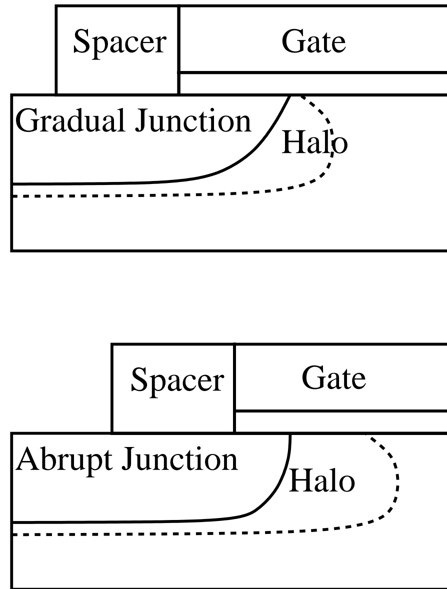


Figure 3.12: Overlap spacer is not used as a tunable parameter. L_{met} is kept constant for different lateral abruptness by varying L_{gate}.

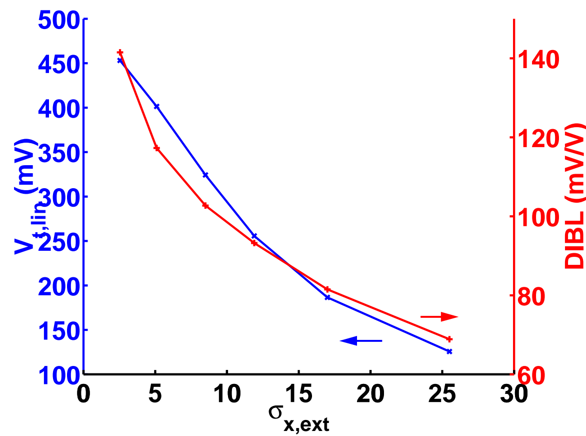


Figure 3.13: Threshold Voltage and DIBL for devices with varying halo location. The lateral abruptness of the extension junction is varied by controlling the characteristic length of the Gaussian profile.

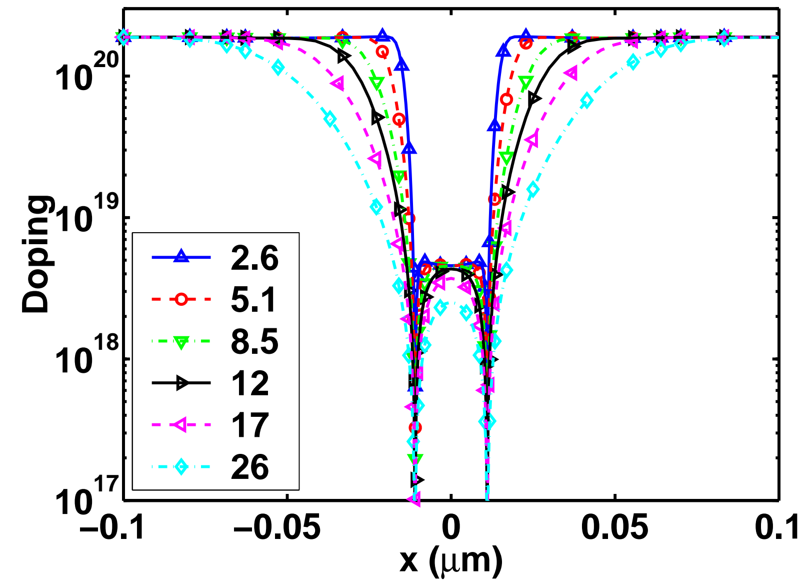
increases in this scenario is the changing halo location. Figures 3.14a and b shows the net doping along the surface of the devices for the two scenarios discussed. Under scenario two, as we increase the lateral abruptness of the extension, the gate length has to be decreased to maintain a constant metallurgical channel length (which is correlated to the effective channel length), causing the halo to move closer to the center. This in turn leads to an increase in the doping at the channel center. The resulting increase in the threshold voltage masks the degradation in charge sharing effects, resulting in the threshold voltage plot of Figure 3.13.

This explains why the threshold voltage behavior described in Section 3.4 was not observed in previous publications.

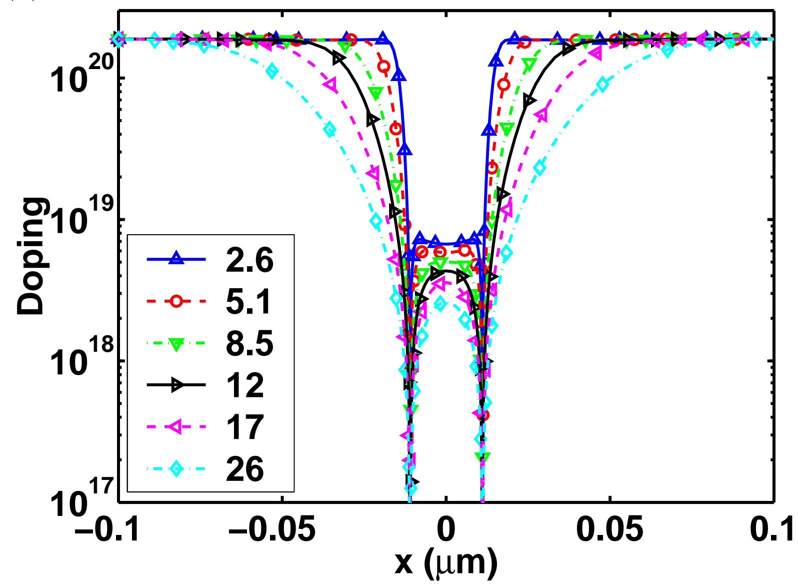
3.6.2 Source/Drain Doping Description

One of the assumptions of this work is that doping in the source/drain extension region can be described by a tensor product as in equation 3.1. This is a reasonable approximation for doping profiles resulting from ion implantation and diffusion. Under this assumption, increasing lateral abruptness would lead to a more box-shaped profile, which eventually would degrade threshold roll-off characteristics, as described in Section 3.4.2.

However, it is not known if equation 3.1 is appropriate for doping profiles resulting from techniques such as laser annealing, a leading candidate for creating these ultra abrupt junctions [71] [27]. In fact, due to the difficulty in estimating the full two-dimensional doping, not much is known about the lateral profile resulting from these techniques. Privitera, et al [62] suggest that laser annealing could result in a box-shaped profile. Hence using laser annealing as the means to obtain an abrupt lateral doping gradient may also couple an abrupt junction with a box-shaped profile, which



(a)



(b)

Figure 3.14: Doping at the surface of devices in (a) Scenario 1 (b) Scenario 2, with various lateral abruptness ($\sigma_{x,ext}$ ranging from 2.6 to 26 nm)

would cause charge sharing effects to be degraded. This in turn would lessen the overall benefit of a laterally abrupt junction, as we saw in Section 3.5.

Note that increased lateral abruptness is not the only benefit of laser annealing. Laser annealing also increases the vertical abruptness of the junction, increases the activated doping concentration, and allows the construction of ultra shallow junctions [71]. While lateral abruptness by itself does not seem to confer much performance benefit, all these factors combine to offer (potentially) improved device performance for laser annealed devices [8].

3.6.3 Factors affecting Sensitivity to Lateral Abruptness

Mobility Models

As demonstrated in Section 2.5.2, proper choice and calibration of mobility models are very important for predictive TCAD. In this section, we briefly examine the impact the mobility models has on the simulation results. Table 3.4 shows the I_{on} obtained after tuning the gate length to achieve off currents of 100nA/μm, for devices with lateral source/drain extension abruptness ranging from 13 nm/dec to 1.9 nm/dec and simulated using the Universal mobility model [7], Lucent mobility model [23] and the Generalized mobility curve model [55]. It is obvious that the sensitivity of the drive current to gradient abruptness depends greatly on the mobility model chosen.

A complete examination of mobility models is outside the scope of this thesis. (See Mujtaba [54] and Mudanai, et al [53] for discussions of accumulation layer mobility modeling.) Nevertheless, it is important to note that while the trends discussed in this chapter are not expected to change with model choice, the actual numbers presented should be taken as indicative of scaling trends, not as absolute target values. Ultimately, for quantitatively predictive simulations, the mobility model must be

Abruptness	UNIMOB [7]	LUCMOB [23]	GMCMOB [55]
13 nm/dec	-3.85%	-4.09%	-12.80%
9.8 nm/dec	-1.59%	-2.05%	-8.23%
6.5 nm/dec	-0.49%	-0.96%	-4.54%
4.5 nm/dec	-0.13%	-0.51%	-2.58%
3.3 nm/dec	0.00%	-0.24%	-1.30%
1.9 nm/dec	-0.03%	0.00%	0.00%

Table 3.4: Percentage deviations of $I_{on,target}$ at $I_{off} = 10^{-7}A/\mu m$ from the devices on the "best case" curve with different abruptnesses. Three different mobility models are explored in the simulations. Oxide thickness is 1.5 nm and channel doping is 4.8×10^{18} for all the devices shown.

Abruptness	$t_{ox} = 1.5nm$ S/D 1	$t_{ox} = 2.0nm$ S/D 1	$t_{ox} = 2.0nm$ S/D 2
13 nm/dec	-12.80%	-12.79%	-15.77%
9.8 nm/dec	-8.23%	-8.59%	-10.95%
6.5 nm/dec	-4.54%	-4.98%	-6.54%
4.5 nm/dec	-2.58%	-3.21%	-4.19%
3.3 nm/dec	-1.30%	-2.06%	-2.61%
1.9 nm/dec	0.00%	0.00%	0.00%

Table 3.5: Percentage deviation of I_{on} for target I_{off} of $10^{-7}A$ the best case (for a given t_{ox}) for devices with various abruptnesses abruptness. Channel doping is 4.8×10^{18} for all devices shown. Simulated with GMCMOB. Doping profiles S/D1 and S/D2 are as shown in Figure 3.15.

calibrated to the given technology.

Gate Oxide Thickness and Source/Drain Doping Design

Tables 3.5 and 3.6 show the impact of gate oxide thickness and source/drain doping design on the sensitivity of I_{on} (at target off-currents of 100nA/μm) to lateral abruptness in the lateral source/drain extension region.

The increase in dependence of I_{on} on the junction abruptness with increasing oxide thickness for relatively abrupt junctions can be understood by considering Figure 3.1b.

Abruptness	$t_{ox} = 1.5nm$	$t_{ox} = 2.0nm$
13 nm/dec	96.8	129.8
9.8 nm/dec	91.9	118.9
6.5 nm/dec	70.0	97.7
4.5 nm/dec	67.8	81.4
3.3 nm/dec	59.8	69.9
1.9 nm/dec	51.7	58.3

(a)

Abruptness	$t_{ox} = 1.5nm$	$t_{ox} = 2.0nm$
13 nm/dec	87.2%	122.6%
9.8 nm/dec	77.8%	103.9%
6.5 nm/dec	35.4%	67.6%
4.5 nm/dec	31.1%	39.6%
3.3 nm/dec	15.7%	19.9%
1.9 nm/dec	0.0%	0.0%

(b)

Table 3.6: (a) Resistance in the overlap region for devices with two different oxide thickness. (b) Percentage difference from 1.9 nm/dec case. Gate Length is 50 nm. Simulated using GMCMOB.

As we increase the oxide thickness, the accumulation layer charge decreases, leading to an increase in the accumulation resistance components. This in turn causes current spreading to take place closer to the metallurgical junction. As a result the spreading resistance plays a more prominent role in determining the resistance in the overlap region. Since the spreading resistance depends strongly on the lateral abruptness, the net result is an increase in dependence of the device characteristics on source/drain abruptness.

For more gradual junctions, this effect is less significant. As the junction gradient extends beyond the gate edge, the resistance under the spacer also increases as the junction gradient decreases. The resistance in the spacer region does not have a strong dependence on the gate oxide thickness. As a result, the gate oxide thickness no longer has a strong influence on the sensitivity of device characteristics to junction abruptness.

Also shown in Table 3.5 is the effect of source/drain doping details on the sensitivity of device characteristics to lateral source/drain abruptness. Figure 3.15 shows the doping profiles in the source/drain extension region for the devices referred to in the

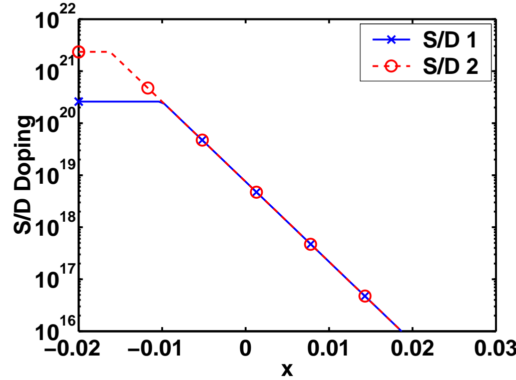


Figure 3.15: Source/Drain doping profile examined in Table 3.5. Only the 6.5 nm/dec case is shown. Junction is located at x = 0 in this plot.

table. The sensitivity to source/drain extension junction gradient is higher for the second source/drain design since the gradient extends further into the source/drain region, increasing the portion of the device over which the junction gradient has influence.

3.7 Summary

Proper choice of performance metric is important for comparing the different device designs. Series resistance by itself is not a good criterion for judging device performance, since it considers only on-currents and ignores off-currents. Similarly, threshold roll-off does not take into account the impact of series resistance. To reconcile the two sets of ITRS abruptness requirements, both of these effects must be considered.

For this reason, I_{on}-I_{off} curves are useful in comparing different technologies. Two refinements of the conventional I_{on}-I_{off} plot are described in this chapter. Use of the channel doping as the free parameter (instead of channel length as in the conventional

I_{on}-I_{off} plots) better match the typical technology design scenario and better demonstrate the limits of device scaling. At the same time, channel length variations can be taken into account through consideration of supernominal I_{on} versus subnominal I_{off}.

A thorough simulation study of the impact of lateral source/drain abruptness and the gate-extension overlap length on device performance are presented, from the viewpoint of series resistance, threshold voltage roll-off and three different kinds of I_{on}-I_{off} plots.

Lateral source/drain abruptness is shown to have a dramatic impact on the series resistance of modern MOS devices. This is significant since external resistances make up a larger fraction of the total resistance of MOS devices as device scaling continues.

At the same time, for source/drain extensions whose doping profile is well described by a tensor product of two one-dimensional functions[13], it is shown that the threshold roll-off characteristics are degraded by a lateral extension junction that is too gradual or too abrupt, and there is an optimal abruptness for a given extension junction depth design. Two competing factors are at work: for very gradual junction gradients, counter-doping of the channel by the junction tails causes the threshold voltage of small dimension device to decrease; for very abrupt junction gradients, the resulting increase in effective junction depth aggravates charge sharing effects. Note that the presence of halo doping could confound and mask this effect.

Furthermore, the gate-extension overlap length is shown to have significant impact on device performance, suggesting the use of an overlap spacer would be beneficial for device optimization. Moreover, the impact of lateral abruptness on I_{on}-I_{off} depends

[13]Such as a Gaussian in the vertical direction and an error function in the horizontal direction.

on the gate-extension overlap length. For devices with substantial underlap, increasing abruptness improves performance; for devices with sufficient overlap, increasing abruptness hurts performance.

In summary, increasing lateral source/drain abruptness lowers series resistance, which tend to improve the drive current. However, for very abrupt junctions, this improvement is mitigated by the degradation in leakage currents due to the worsening of short channel effects. The net improvement in I_{on}-I_{off} is less than would be expected if series resistance alone is considered. This must be taken into account in deciding if an abrupt lateral junction should be employed in the design of deep sub-micron devices.

Chapter 4

Resistance Calculations

4.1 Introduction

In Chapter 3, considerable attention was paid to the impact of lateral abruptness on series resistance. This is because series resistance has become a significant limiting factor for device performance in the deep sub-micron region. While the intrinsic channel resistance improves with scaling, the parasitic series resistances in the source, drain and contact regions do not scale well and are becoming a significant part of the total resistance. Furthermore, as presented in Chapter 3, source/drain resistance is closely linked to the lateral abruptness in the doping profiles. Understanding the factors controlling these resistive components in the source/drain region is therefore very important.

Three different methods for calculating the physical resistance components in a MOS device are examined in this chapter. After a brief review of nomenclature in Section 4.2 and basic resistance calculations in Section 4.3, resistance calculation

strategies based on partitioning the source/drain region into vertical strips are examined in Section 4.4. The analytical treatment of spreading resistance presented in Ng and Lynch [57] and cited by the ITRS [1] in formulating the requirements for lateral extension abruptness, is used as an example of such strategies. In Section 4.5, a more rigorous method for calculating resistive components is developed. This method uses equipotential lines obtained through device simulation, and correctly accounts for the two dimensional nature of current flow in the source/drain regions. In Section 4.6, the sheet resistivity equation as given in Taur and Ning [74] is shown to be equivalent to a special case of the generalized method from Section 4.5. In Section 4.7, the results obtained for the above three methods are compared. Resistance calculations based on vertical strips are shown to produce substantial errors in the spreading resistance. Finally, in Section 4.8, we compare the physical resistances calculated using these numerical methods with the extracted values from the shift-and-ratio method. A new extraction technique is presented and used to better understand the behavior of the shift-and-ratio method and to explain the observed discrepancies. The values obtained from extraction methods such as shift-and-ratio are shown to be meaningful only in the context of the assumptions implicitly made in the extraction.

4.2 Nomenclature

Figure 4.1 shows a schematic diagram of one-half of a typical MOS device, together with the major resistive components. The total resistance along the conducting path is made up of the intrinsic channel resistance, and the parasitic series resistance in the source and drain. The extrinsic series resistance is in turn made up of several

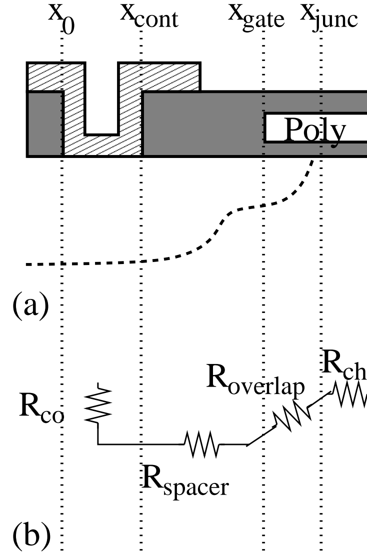


Figure 4.1: Typical MOS Device (a) Schematic representation of a half device (b) Major resistive components along the conducting path

components. The total resistance is given by

$$R_{tot} = R_{ext} + R_{ch} = R_{co} + R_{spacer} + R_{overlap} + R_{ch} \tag{4.1}$$

where R_{co} is the contact resistance for the silicide, R_{spacer} is the resistance of the region under the source/drain spacer, and $R_{overlap}$ is the resistance in the extension overlap region under the gate.

4.3 Analytical Resistance Calculations

Analytical models such as the one presented by Ng and Lynch [57] can provide a great deal of qualitative insight into device behavior. In order to better understand these analytical models, the basic equations for resistance calculations are reviewed in this section.

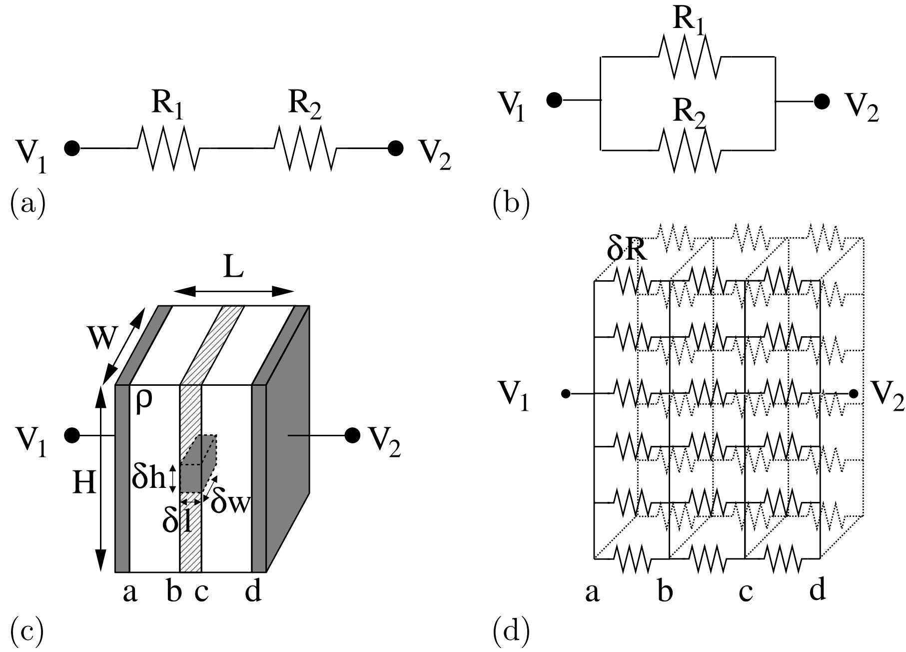


Figure 4.2: (a) Resistances connected in series. $R_{tot} = R_1 + R_2$. (b) Resistances connected in parallel. $R_{tot} = (1/R_1 + 1/R_2)^{-1}$. (c) Resistive block with resistivity of ρ. $R_{tot} = \int_0^L \frac{\rho}{W \cdot H} dl$. (d) Equivalent circuit of microscopic resistances. $\delta R = \frac{\rho \cdot \delta l}{\delta w \cdot \delta h}$. Virtual nodes a, b, c, d correspond to the cross-sections a, b, c, d shown in (c).

Figures 4.2a and 4.2b show resistances connected in series and in parallel respectively. It is important to note that resistances connected in parallel share the same potential at their end-points (they are connected to the same node). The derivation of the equations governing the parallel combination of resistances depends on this fact.

Figure 4.2c shows a uniform block of material with resistivity ρ. We know from the definition of resistivity for a uniform block that

$$R_{tot} = \frac{\rho L}{WH} \tag{4.2}$$

It is instructive to consider an alternative viewpoint. Conceptually, the block in

Figure 4.2c can be divided into slabs perpendicular to the current flow (shaded strip) which can in turn be divided into a large number of smaller blocks. Assuming these blocks are sufficiently small and that resistivity is continuous in space, the resistance of each small block can be calculated using

$$R_{i,j,k} = \frac{\rho(\vec{x})\delta l}{\delta h \cdot \delta w} \tag{4.3}$$

The resistance of a given slab is then given by combining the constituent blocks in parallel

$$R_{slab,k} = \left(\sum_i \sum_j R_{i,j,k}^{-1} \right)^{-1} \tag{4.4}$$

$$= \frac{\rho \cdot \delta l}{W \cdot H} \text{ for uniform } \rho \tag{4.5}$$

Summing the resistances for all the slabs then gives

$$R_{tot} = \sum_k R_{slab,k} \tag{4.6}$$

$$= \frac{\rho L}{W \cdot H} \text{ for uniform } \rho \tag{4.7}$$

which is the same as Equation 4.2. The complete equivalent resistive network is shown in Figure 4.2d. Note that the end faces of each slab correspond to a single (virtual) node on the equivalent circuit. This is valid since each of the faces is an equipotential surface.

4.4 Vertical Strip Calculation Method

4.4.1 Ng and Lynch's Analytical Model

Ng and Lynch [57] present an analytical treatment of the resistive components in the source/drain region. Such analytical models can provide a great deal of qualitative insight into device behavior. However, the approximations needed to arrive at an analytical solution limit the achievable accuracy. Given that the Ng and Lynch approach forms part of the basis for the ITRS [1] requirements for lateral source/drain abruptness, it is important to examine the modeling assumptions carefully.

Ng and Lynch attempt to account for current spreading by partitioning the source/drain region into vertical strips as indicated in Figure 4.3. Integrating the resistivity along these vertical strips and summing the resulting resistances in series gives[1,2]

$$R_{sp} = \int \frac{\rho(x')dx'}{(\tan 1)x'W} - ... \tag{4.8}$$

where W is the width of the device. Note that the ellipsis in equation 4.8 represents a correction term resulting from the way R_{sp} is defined in Ng and Lynch, which we ignore here.

4.4.2 Limits of the Method

Resistance of the vertical strips can be calculated as given in equation 4.8 only if both of the vertical side-walls of each strip are equipotential surfaces. Figures 4.4 and 4.5 show the current flow lines and the equi-quasi-Fermi potential lines in a

[1] Note that Ng and Lynch assumes that current spreads at an angle of 1 radian.
[2] x' is the distance from the edge of the channel to a point in the source/drain region along the Si/SiO_2 interface, as defined in [57].

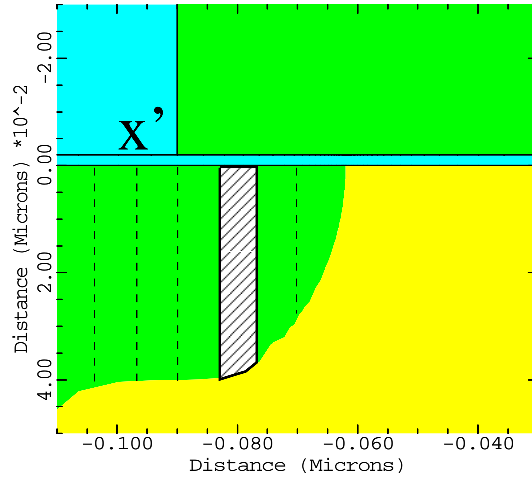


Figure 4.3: Partitioning a device based on vertical strips.

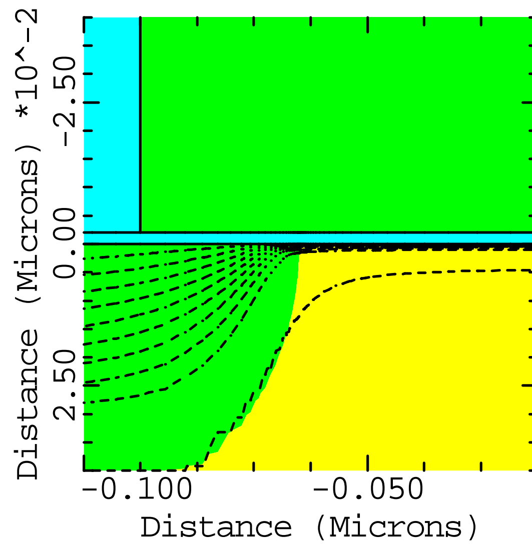


Figure 4.4: Current flow lines in a typical MOS device.

typical MOS device obtained from device simulation. It can be observed that due to current spreading in the source/drain extension regions, the equipotential lines are not vertical and the current flow is not parallel to the surface. As a result, resistance calculations based on vertical strips are subject to significant errors in these regions, as we discussed further in Section 4.7.1.

4.5 Calculation based on Equipotential Lines

Device simulation allows us to solve the electrostatic and current equations rigorously for devices with arbitrary doping profiles. A method for calculating resistance components from simulation results is now developed. The non-uniformity in resistivity due to local variations in carrier concentrations and mobility as well as the multi-dimensional nature of current flow are accurately accounted for.

4.5.1 Resistances in the Bulk

When current flow is not one-dimensional, resistances must be combined in accordance with the equipotential lines. The region of interest is first partitioned into thin resistive strips delineated by the equipotential lines (striped area in Figure 4.5). These strips are in turn partitioned into microscopic resistance elements (with conductance dG) connected in parallel (shaded area in Figure 4.5). The resistance of the strip defined by the equipotential lines, indicated as potential ϕ_1 and ϕ_2 can, then be

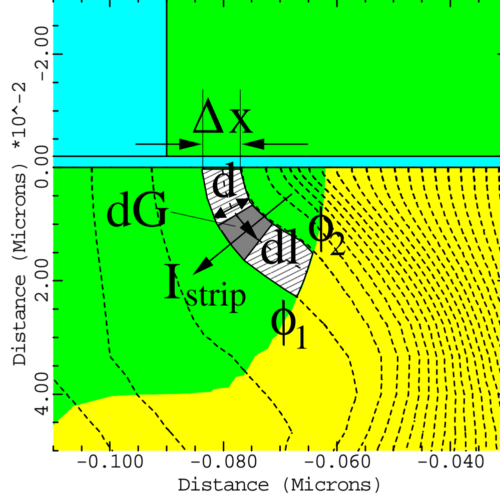


Figure 4.5: Partitioning a device based on equipotential lines. Conductance for a resistive strip defined by equipotential lines can be decomposed into microscopic conductances of value $dG = \frac{\sigma(\vec{x})}{d(\vec{x})} d\vec{l}$ connected in parallel

calculated as follows:

$$\frac{1}{R_{\phi_1,\phi_2}} = \int dG = \int \frac{\sigma(\vec{x})}{d(\vec{x})} W \cdot d\vec{l} = \frac{W}{\phi_2 - \phi_1} \int \sigma(\vec{x}) \vec{\nabla}\phi(\vec{x}) \cdot d\vec{l} \tag{4.9}$$

where $\vec{x}$ is the vector representing a point along the path integral, $d(\vec{x}) = (\phi_2 - \phi_1)/\vec{\nabla}\phi(\vec{x})$ is the width of the strip at point $\vec{x}$ along the strip, and $\sigma = 1/\rho$.[3] Since current flow in a MOS device is confined to a finite layer close to the surface, the limits of integration for each strip are bounded.

[3]Calculated as described in the following subsection.

The total resistance of the region can then be calculated by summing these resistances in series

$$R = \sum_{i=0}^{n-1} R_{\phi_i,\phi_{i+1}} \tag{4.10}$$

Calculation of Local Resistivity from Device Simulation

The local resistivity of a (semiconductor) material is typically defined as

$$\rho(\vec{x}) \equiv \frac{\mathcal{E}(\vec{x})}{J(\vec{x})} = \frac{1}{q(\mu_n(\vec{x})n(\vec{x}) + \mu_p(\vec{x})p(\vec{x}))} \tag{4.11}$$

It is not immediately obvious that equation 4.11, which is based purely on carrier drift, is appropriate for calculating resistance a real semiconductor device. In resistance calculations, we are ultimately interested in obtaining a measure of the difficulty of external biases to cause current flow. Since the total current in a device involves a delicate balance between the drift and diffusion components, and since even at zero bias, huge drift and diffusion currents could exist in the device junctions, it is obvious that both drift and diffusion must somehow be taken into account.

We can generalize the notion of resistivity as the ratio between the current density and an appropriate driving force, very much analogous to equation 4.11. The current components are given by

$$\vec{J}_n(\vec{x}) = q\mu_n(\vec{x})n(\vec{x})\vec{\nabla}\phi_n(\vec{x}) \tag{4.12}$$

$$\vec{J}_p(\vec{x}) = q\mu_p(\vec{x})p(\vec{x})\vec{\nabla}\phi_p(\vec{x}) \tag{4.13}$$

Given that externally applied biases directly modulate the quasi-Fermi levels at the contacts, this, together with equations 4.12 and 4.13 above, suggest that the gradient

of quasi-Fermi levels can serve as a more meaningful driving force in the resistivity definition. We therefore **define**

$$\rho_n(\vec{x}) \equiv \frac{\vec{\nabla}\phi_n(\vec{x})}{\vec{J}_n(\vec{x})} = \frac{1}{q\mu_n(\vec{x})n(\vec{x})} \tag{4.14}$$

$$\rho_p(\vec{x}) \equiv \frac{\vec{\nabla}\phi_p(\vec{x})}{\vec{J}_p(\vec{x})} = \frac{1}{q\mu_p(\vec{x})p(\vec{x})} \tag{4.15}$$

The total resistance is then given by

$$R_{tot} = \left(\frac{1}{R_n} + \frac{1}{R_p}\right)^{-1} \tag{4.16}$$

where R_n and R_p can be calculated from ρ_n and ρ_p respectively using equation 4.9 from Section 4.5.1.

For a MOS device, one of the carrier terms will dominate. Consider an NMOS device. R_{tot} reduces to R_n, allowing us to define

$$\rho(\vec{x}) \equiv \rho_n(\vec{x}) = \frac{1}{q\mu_n(\vec{x})n(\vec{x})} \tag{4.17}$$

This is equivalent to equation 4.11 with electrons dominating. In essence, since drift current dominates in a MOS device, resistivity as defined using the quasi-fermi levels is the same as the standard resistivity definition based on drift. Hence for a MOS device, we could use either equation 4.11 or equation 4.14/4.15 for resistivity calculations, whichever is more convenient.

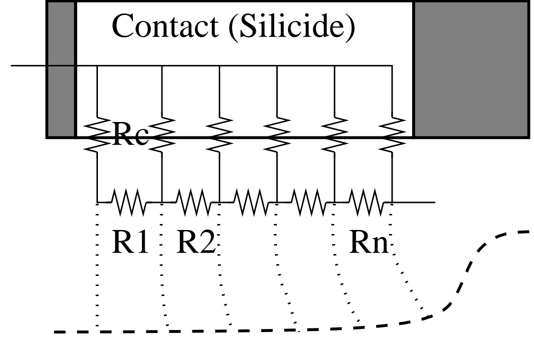


Figure 4.6: Equivalent circuit for calculating the resistances of the source/drain contacts

4.5.2 Contact Resistance

Figure 4.6 shows a distributed resistive network used for calculating the resistive contribution of the diffused contact region. The total contact resistance is obtained by recursively applying the following

$$R_{\Sigma,1} = R_{c,1} + R_1 \tag{4.18}$$

$$R_{\Sigma,i} = \left(\frac{1}{R_{\Sigma,i-1}} + \frac{1}{R_{c,i}} \right)^{-1} + R_i \qquad \mathrm{i} = 1 \ldots \mathrm{n} \tag{4.19}$$

$$R_{co} = \left(\frac{1}{R_{\Sigma,n}} + \frac{1}{R_{c,n+1}} \right)^{-1} \tag{4.20}$$

where R_i is the resistance of the i-th strip (calculated using equation 4.9), $R_{c,i} = \rho_c \cdot l_i \cdot W$ is the contact resistance associated with that strip, ρ_c is the contact resistivity between the silicide and the source/drain region, l_i is the distance spanned by the i-th strip at the surface of the device and W is the width of the device.

4.5.3 Implementation

The resistance calculation is applied as a post-processing step following device simulation[4]; solution files stored in TIF format [7] are used as input. The equipotential lines are generated using the marching triangles algorithm [49] on an unstructured mesh. The resistance components of the device are then obtained from equations 4.9, 4.10 and 4.20.

4.6 Calculation Using the Quasi-Fermi Level at the Surface

A third method for calculating resistance inside a MOS device uses an incremental form of Ohm's law [74] [54] [17]. For an NMOS device

$$R_{sh}(x) = \frac{d\phi_n(x)}{dx} \Bigg/ \frac{I_{ds}}{W} \tag{4.21}$$

where x is the horizontal coordinate along the Si/SiO_2 interface, ϕ_n is the electron quasi-Fermi level at the Si/SiO_2 interface obtained from device simulation, I_{ds} is the total current and W is the width of the device.

Equation 4.21 involves only quantities at the Si/SiO_2 interface and seems at first glance to be a crude, one-dimensional approximation. In fact, [74] suggests that it is valid only when current flow is largely parallel to the x-direction and the equipotential contours are perpendicular to the Si/SiO_2 interface. This is not the case; it will now be shown that equation 4.21 is appropriate even where two-dimensional current flow

[4]Avant! Medici [7] is used in this case

is important.

Consider a resistive strip similar to the one in Figure 4.5, defined to contain all the drain current flowing through the device ($I_{strip} = I_{ds,tot}$). The total resistance of the strip is given by Ohm's law

$$R_{\phi_1,\phi_2} = \frac{\phi_2 - \phi_1}{I_{ds}} \tag{4.22}$$

where ϕ_1 and ϕ_2 are the values of the quasi-Fermi level for the equipotential lines bounding the strip, and I_{ds} is the total drain current. This should give the same result as equation 4.9. Dividing both sides of equation 4.22 by Δx (the width of the strip at the Si/SiO_2 interface), and taking the limit as $\Delta x \rightarrow 0$, equation 4.22 becomes identical to equation 4.21. Thus for the purposes of calculating the resistance of regions containing all the drain current flowing in the device, equation 4.21 is equivalent to equation 4.9.

Note however that the more general expressions of equations 4.9 and 4.10 allow calculation of resistances for arbitrarily defined regions, which is not possible using equation 4.21.

4.7 Results

4.7.1 Comparison between Resistance Calculation Methods

Figure 4.7 shows a comparison of the sheet resistance calculated using the three methods described in this chapter: the calculation based on quasi-Fermi levels at the Si/SiO_2 interface ($\phi_n(x)$, see Section 4.6); numerical integration based on equipotential lines (Equi-ϕ, see Section 4.5); and numerical integration based on vertical strips

Method	R_{co}	R_{spacer}	$R_{overlap}$	R_{chan}
III $\phi_n(x)$	65.8	35	**62.1**	428
II Equi-ϕ	66.2	36.1	**61.7**	432
I Vertical	65.6	33.1	**44.4**	434

Table 4.1: Resistive components (in $\Omega/\mu m$) calculated using three different methods: quasi-Fermi level at Si/SiO_2 interface ($\phi_n(x)$); integration based on equipotential lines (Equi-ϕ); and integration based on vertical strips (Vertical). Note the error in $R_{overlap}$ for the vertical case.

(Vertical, see Figure 4.3 in Section 4.4). Figure 4.7b shows an expanded plot that emphasizes the gate-extension overlap region (indicated by the dashed box in Figure 4.7a).

As we discussed in Section 4.6, the R_{sh} calculated using the two quasi-Fermi level based methods match well, except in the contact regions[5]. At the same time, it is clear from Figure 4.7b that the vertical strip method significantly underestimates the sheet resistivity in the extension region, where current spreading is important. An error of 27% in the calculated $R_{overlap}$ results from using this method, as shown in Table 4.1. This is significant since the approach of Ng and Lynch [57], used for deriving the ITRS roadmap numbers for source/drain abruptness, computes the spreading resistance based on vertical strips (Section 4.4.1). Note that the impact of the resistance calculation method on the other resistive components are small, since current spreading is significant only in the overlap region.

Table 4.2 shows a comparison of $R_{overlap}$ calculated using the three methods for devices with different lateral doping abruptness in the source/drain extension region. In all cases, the resistances calculated using quasi-Fermi potential at the Si/SiO_2

[5]Equation 4.21 assumes a constant I_{ds} throughout the device, which is not true in the contact regions

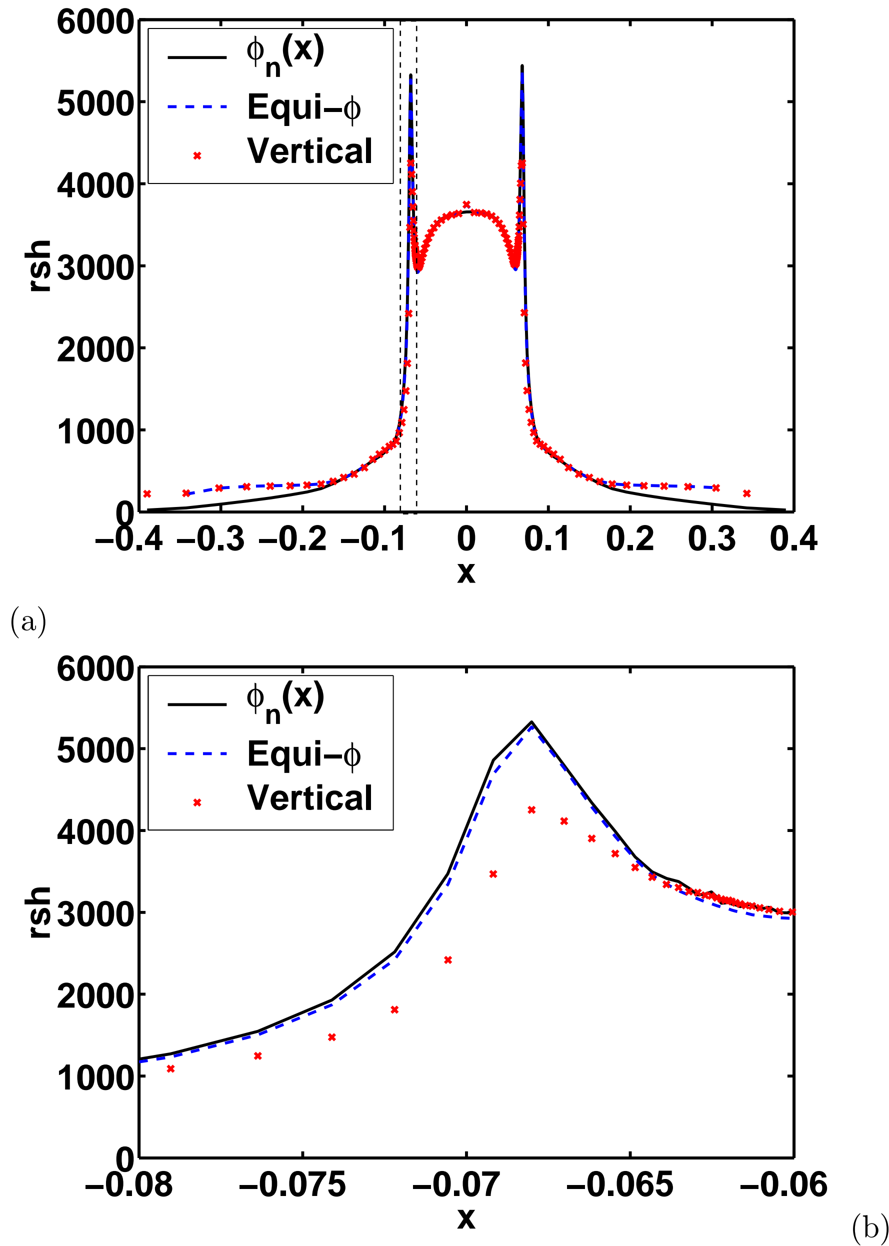


Figure 4.7: (a) Comparison of R_{sh} calculated using: quasi-Fermi level at Si/SiO$_2$ interface ($\phi_n(x)$); numerical integration based on equipotential lines (Equi-ϕ); and numerical integration based on vertical strips (Vertical). (b) Expanded plot showing the overlap region (dashed box in (a)), where two-dimensional current flow is important.

Abruptness (nm/dec) / Method	10.0	7.5	5.0	3.5	2.5
III $\phi_n(x)$	92.3	81.6	69.7	62.1	56.8
II Equi-ϕ	92.5	81.6	69.5	61.7	56.4
I Vertical	85.0	71.0	55.1	44.4	37.0
% diff between I and II	8.1%	13.0%	20.7%	28.0%	53.3%

Table 4.2: $R_{overlap}$ (in $\Omega/\mu m$) calculated for devices with different lateral source/drain abruptnesses using three different methods: quasi-Fermi level at Si/SiO_2 interface ($\phi_n(x)$); integration based on equipotential lines (Equi-ϕ); and integration based on vertical strips (Vertical).

interface match those calculated using numerical integration based on equipotential lines as before. At the same time, the error incurred by resistance calculations based on vertical strips increases with increasing junction abruptness. This can be understood by noting that for more abrupt junctions, current spreading occurs closer to the metallurgical junction and affects a larger fraction of the gate-extension overlap region. From these results, we conclude that resistance calculations based on vertical strips overestimate the benefit of increasing lateral source/drain abruptness on series resistance.

4.7.2 Resistance Components and Lateral Abruptness

Table 4.3 shows the resistive components calculated using equation 4.21 for a different set of devices with various lateral source/drain extension abruptness. A 30% decrease in $R_{overlap}$ is observed as the lateral abruptness varies from 6.5nm/dec to 1.9nm/dec. The qualitative trend is consistent with the predictions of Ng and Lynch [57]. Note that $R_{overlap}$ is the only resistive component with a strong dependence on lateral

Abruptness	R_{co}	R_{spacer}	$R_{overlap}$	R_{chan}	V_{th}
1.9 nm/dec	135.1	26.86	63.86	198.5	0.35
3.3 nm/dec	135.1	27.1	72.7	212.7	0.38
4.5 nm/dec	135.1	27.5	80.62	220.2	0.39
6.5 nm/dec	135.1	28.58	91.88	222.4	0.39

Table 4.3: Comparing resistive components, in $\Omega/\mu m$, for devices with different lateral doping abruptness in the source/drain extension region.

source/drain extension abruptness.

4.7.3 Gate-Bias Dependence of Resistance Components

Figure 4.8 shows the sheet resistance of a device with lateral source/drain extension abruptness of 1.9 nm/dec at various applied gate biases. The gate extends from -0.025 to +0.025 μm, while the metallurgical junctions lie at -0.011 and +0.011 μm respectively (only the left half of the channel is shown). It is clear that the channel and the extension resistance have a strong gate-bias dependence. This is due to gate control of the inversion and accumulation layers.

Figure 4.9a shows schematically the partitioning of the resistive strips into a bias dependent and a bias independent part. The bias dependent part is defined here to be the region in which the resistivity varies by more than 1% as the gate bias varies from 0.4V to V_{dd}. Using equation 4.9, we can calculate the gate-bias dependent part of the conductance for each equipotential strip as follows

$$G_{\phi_1,\phi_2,dept} = \frac{W}{\phi_2 - \phi_1} \int_{\mathcal{L}_{dept}} \sigma(\vec{x})\vec{\nabla}\phi(\vec{x}) \cdot d\vec{l} \qquad (4.23)$$

where $\mathcal{L}_{dept}$ is the gate bias dependent part of the strip. Figure 4.9b shows the calculated gate-bias dependent resistance as a fraction of the total strip resistance.

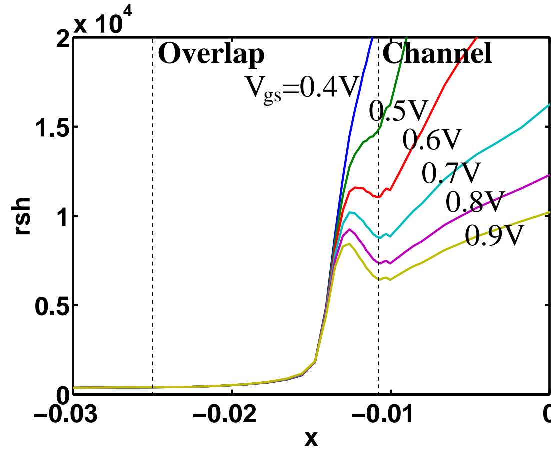
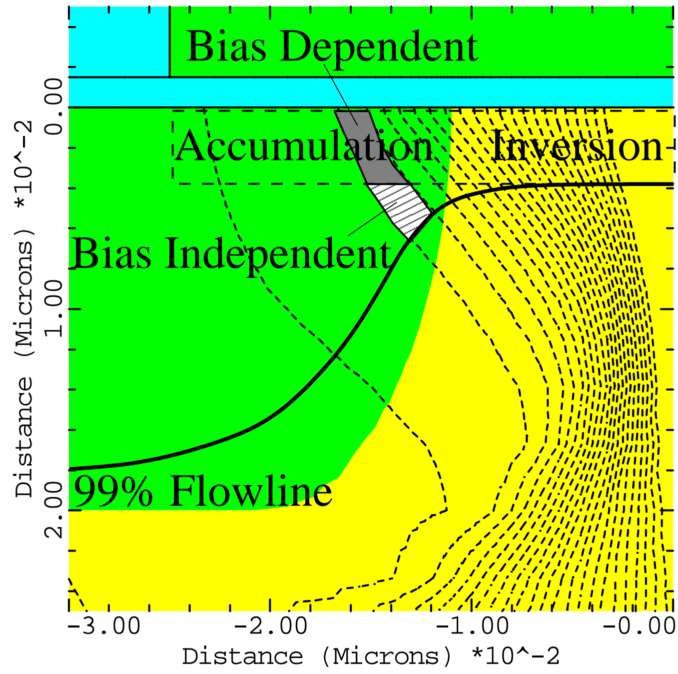


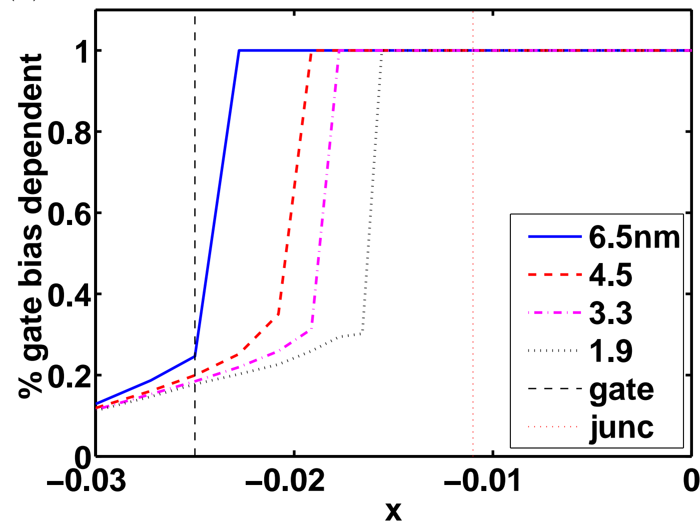
Figure 4.8: Plot of sheet resistance along the channel for various V_{gs}. Lateral doping abruptness in the source/drain extension region is 1.9 nm/dec for this device. The gate extends from -0.025 to 0.025 μm. The metallurgical junctions lie at -0.011 and 0.011 μm respectively.

In the channel, as well as in the part of the source/drain extension close to the channel, the resistance of the entire strip depends on the applied gate voltage. This is consistent with Figure 4.8. Moving further away from the channel, current spreading begins to siphon current away from the accumulation layer. As a result, a smaller fraction of the strip remains controlled by the gate. Note that the current spreading takes place sooner for more abrupt doping profiles, as anticipated by the discussion in Section 4.7.1.

In conclusion, equation 4.21 provides a simple but accurate way of calculating sheet resistances. On the other hand, while equations 4.9 and 4.10 are more involved, they can be applied to more general geometries and provide more detailed information about the device.



(a)



(b)

Figure 4.9: (a) Schematic Partitioning of resistive strips into bias dependent and independent parts (b) Plot of the contribution of the gate bias dependent part of the sheet conductance versus distance.

4.8 Resistance Calculation versus Resistance Extraction

While the resistance calculations described in Section 4.5 provide valuable insight for device design, they cannot be applied directly to actual devices as spatial variation of quasi-fermi level cannot be measured directly. To handle experimental data, extraction techniques such as the shift-and-ratio method [76] [72] are essential. Correlating the physical resistances (as calculated by equation 4.9 or 4.21) with the values obtained by these extraction techniques is not straightforward, however. In this section, the shift-and-ratio method is applied to the simulated data for several devices. The results are compared to the physical resistances calculated using equation 4.21. A new extraction method is then presented and used to understand the observed discrepancies. Finally, the impact of the basic assumptions of the shift-and-ratio method on resistance extraction is discussed.

4.8.1 The Shift-and-Ratio Method

The shift-and-ratio method is based on the following equation [72]

$$R^i_{tot}(V_g) = \hat{R}_{sd} + \hat{L}^i_{\text{eff}} \hat{f}_{ch}(V_g - V^0_t - \delta^i) \tag{4.24}$$

A major assumption of equation 4.24 is that the total device resistance can be partitioned into a gate-bias independent part ($\hat{R}_{sd}$), and a gate-bias dependent part that is directly proportional to channel length. The latter component is assumed to have the same basic functional V_g dependence for devices of all channel lengths.[6]

[6]Threshold voltages are allowed to differ for devices with different channel lengths.

Lat Abruptness	L_{eff} (nm)	R_{sd} (Ω)
1.9 nm/dec	20.9	153.9
3.3 nm/dec	21.6	164.1
4.5 nm/dec	21.9	176.0
6.5 nm/dec	22.5	178.5

Table 4.4: Effective channel lengths and series resistances extracted using the shift-and-ratio method, for devices with gate length of 50 nm and different lateral source/drain extension abruptness. V_{gs} extraction range used in the shift-and-ratio method is from 0.8 V to 1.5 V.

To obtain the threshold shift δ^i, the derivatives of the total resistance for the long channel and short channel devices are shifted with respect to one another until their ratio is approximately independent of gate bias. The effective channel length and source/drain resistance of the short channel device are then calculated from [76] [72]

$$\frac{\hat{L}^0_{\text{eff}}}{\hat{L}^i_{\text{eff}}} = r_{\delta min} \equiv \frac{dR^0_{tot}(V_g)}{dV_g} \Bigg/ \frac{dR^i_{tot}(V_g - \delta^i)}{dV_g} \tag{4.25}$$

$$\hat{R}_{sd} = \frac{r_{\delta min} R^i_{tot}(V_g - \delta^i) - R^0_{tot}(V_g)}{r_{\delta min} - 1} \tag{4.26}$$

Note that the gate bias range used in the extraction could have substantial influence on the extracted results [72].

4.8.2 Shift-And-Ratio versus Physical Resistances

The $\hat{L}_{\text{eff}}$ and $\hat{R}_{sd}$ values extracted from the shift-and-ratio method, for devices with 50 nm gate length and various lateral source/drain extension abruptness, are shown in Table 4.4 and Figure 4.10. Compare this with the gate-bias independent part[7] of the

[7]Resistances are assumed to be gate-bias independent where $r_{sh}(V_g = 0.4V)$ and $r_{sh}(V_g = V_{dd})$ differ by less than 1%.

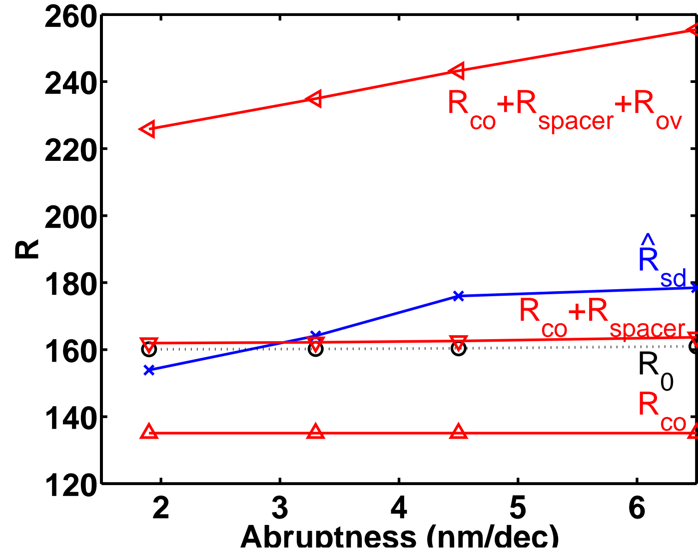


Figure 4.10: Comparing $\hat{R}_{sd}$ extracted through Shift-and-Ratio with the physical resistances calculated using equation 4.21

physical source/drain resistance R_0[8] shown in Figure 4.10. While the $\hat{R}_{sd}$ extracted using the shift-and-ratio method strongly depends on lateral source/drain abruptness, the gate-bias independent part of the source/drain resistance R_0 is essentially independent of lateral abruptness. Given that R_{ov} is the only resistive component that has a strong dependence on lateral abruptness, it seems likely that the $\hat{R}_{sd}$ extracted by the shift-and-ratio method actually incorporates a portion of R_{ov}, which is gate-bias dependent. This violates the assumptions of equation 4.24.

Note that while $\hat{R}_{sd}$ should be larger than R_0 in general[9], R_0 is actually larger than the extracted $\hat{R}_{sd}$ for the 1.9 nm/dec case. Note also that the extracted $\hat{L}_{\text{eff}}$ of devices with very abrupt junctions are smaller than the metallurgical channel length

[8] Calculated from the quasi-Fermi level

[9] $\hat{R}_{sd}$ is the gate bias independent resistance, plus part of R_{ov}

of 22 nm.[10] These discrepancies result from the violation of the assumptions of the method.

4.8.3 Extraction of Gate-Bias Dependent Source/Drain Resistance

A new extraction method which relaxes the assumption that the source/drain resistance be gate-bias independent is now presented. This will help in studying the behavior of the shift-and-ratio method in more detail.

Due to the accumulation layer, the resistance in the gate-extension overlap region is gate-bias dependent. However this bias dependence is different from that of the channel due to differences in the conduction mechanisms.[11] Accordingly, allowing the source/drain resistance to vary with gate bias, equation 4.24 can be modified as follows:

$$R^i_{tot}(V_g) = \tilde{R}_{sd}(V_g) + \tilde{L}^i_{eff}\tilde{f}_{ch}(V_g - V^0_t - \delta^i) \tag{4.27}$$

The terms in equation 4.27 can be extracted by considering a long channel device and two other devices of different gate lengths. Assuming the long channel device is dominated by the channel resistance, we can extract the channel resistance per unit length as follows

$$\tilde{f}_{ch}(V_g - V^0_t) \approx \frac{R^0_{tot}(V_g)}{\tilde{L}^0_{eff}} \approx \frac{R^0_{tot}(V_g)}{L^0_{gate}} \tag{4.28}$$

[10] Devices with $\hat{L}_{\text{eff}}$ smaller than L_{met} are also observed in [72].

[11] Accumulation layer and current spreading in the overlap region versus inversion layer in the channel.

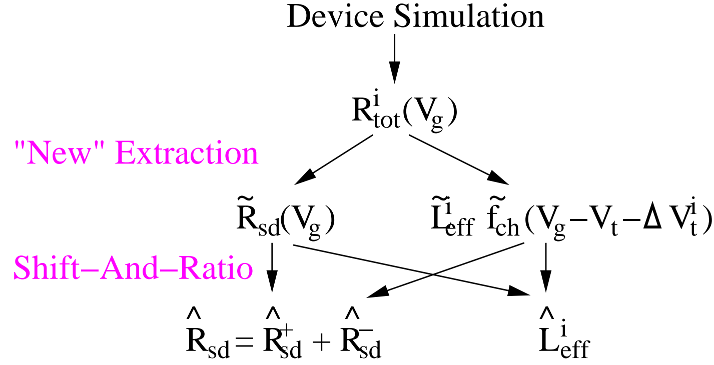


Figure 4.11: Flowchart showing how the terms extracted using the new extraction method contributes to the shift-and-ratio extraction results. Notice how the channel term contributes to the extracted source/drain resistance $\hat{R_{sd}}$.

For the remaining devices, we can write

$$\tilde{R}^a_{sd}(V_g) = R^a_{tot}(V_g) - (L^a_{gate} - \Delta L)\cdot\tilde{f}_{ch}(V_g - V^0_t - \delta V^a_t) \tag{4.29}$$

$$\tilde{R}^b_{sd}(V_g) = R^b_{tot}(V_g) - (L^b_{gate} - \Delta L)\cdot\tilde{f}_{ch}(V_g - V^0_t - \delta V^b_t) \tag{4.30}$$

Now $\tilde{R}^a_{sd}(V_g)$ should equal $\tilde{R}^b_{sd}(V_g)$. Hence ΔL, δV^a_t and δV^b_t can be extracted by minimizing (for instance, through simulated annealing)

$$E = \sum_i (\tilde{R}^a_{sd}(V^i_g) - \tilde{R}^b_{sd}(V^i_g))^2 \tag{4.31}$$

4.8.4 Limits of the Shift-And-Ratio method

We can now examine the impact of the assumptions of the shift-and-ratio method on the extracted results by following the flow-chart in Figure 4.11. We begin by fitting the model represented by equation 4.27 to the simulated data. Applying the procedure described in Section 4.8.3 to simulated devices with gate lengths of 5 μm,

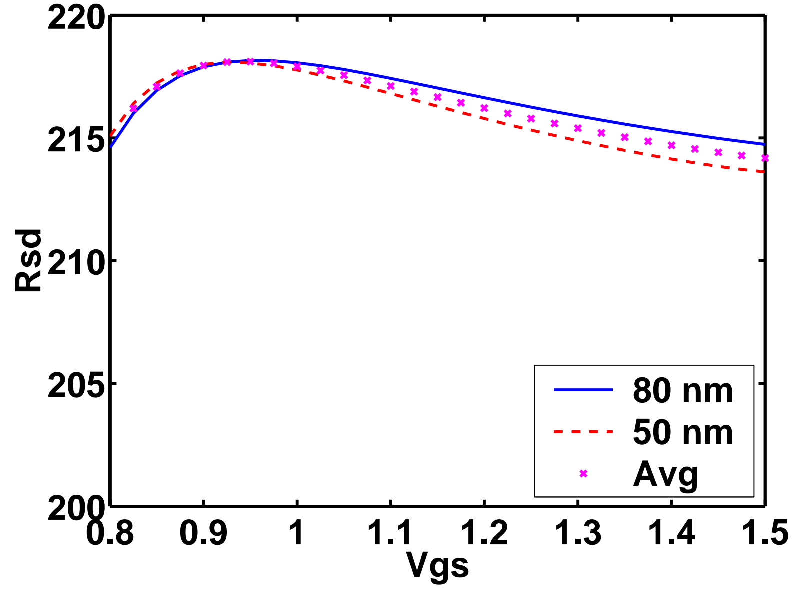


Figure 4.12: Extracted $\widetilde{R_{sd}}$ term in Ω for the 50 nm device, the 80 nm device and their average. Note the difference between the 3 cases are less than 1%

80 nm and 50 nm yields an optimal fit at $\Delta L = 0.0356$, $\delta V_t^a = 0.131$ and $\delta V_t^b = 0.036$. The extracted R_{sd} versus gate bias is shown in Figure 4.12. Note that the extracted R_{sd} for the 50 nm and the 80 nm devices differ by less than 1%, suggesting that equation 4.27 is a good model for these devices.

Applying the shift-and-ratio method to the model represented by equation 4.27 then allows the separation of extracted resistance into its constituent components. Equation 4.25 becomes

$$r_{\delta min} = \frac{L^0}{\tilde{L}^i} \frac{\frac{\tilde{R}'_{sd}(V_g)}{L^0} + f'_{ch}(V_g - V_t^0)}{\frac{\tilde{R}'_{sd}(V_g-\delta)}{\tilde{L}^i} + f'_{ch}(V_g - V_t^i - \delta)} = \left(\frac{L^0}{L^i}\right) \cdot \alpha \tag{4.32}$$

Meanwhile, equation 4.26 becomes

$$\hat{R}_{sd} = \hat{R}^+_{sd} + \hat{R}^-_{sd} \tag{4.33}$$

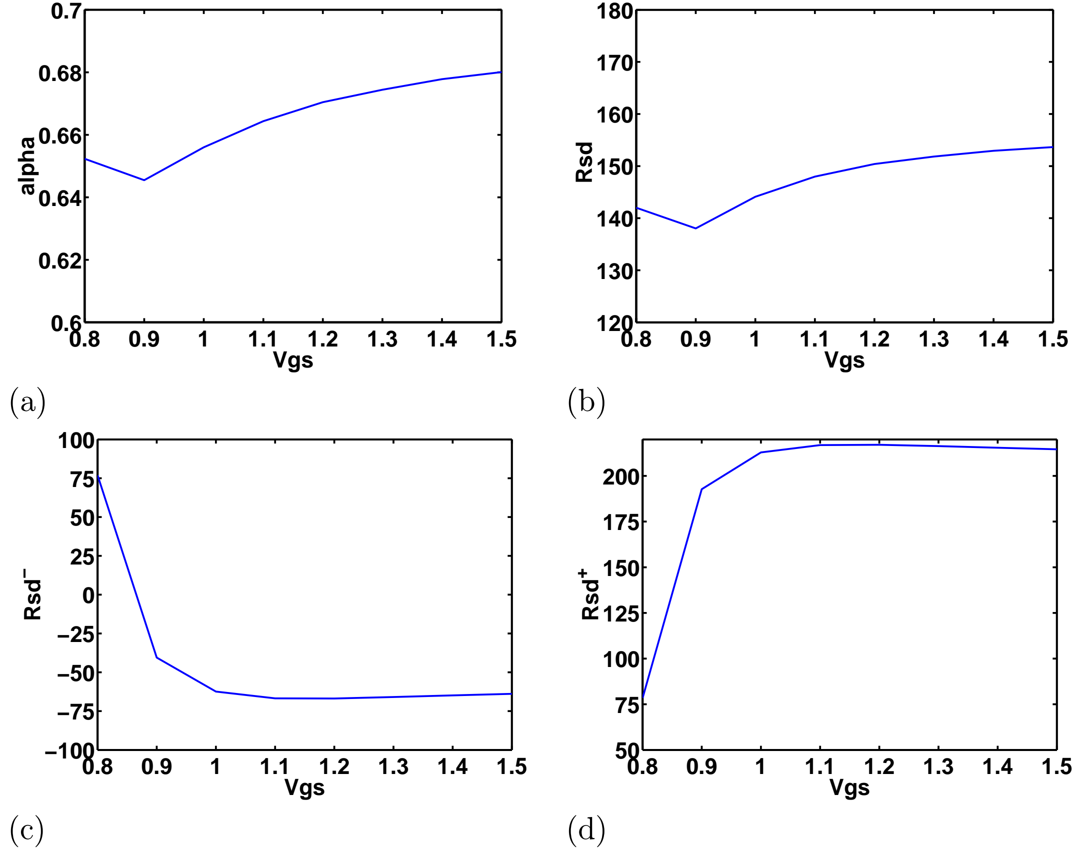


Figure 4.13: Shift-and-ratio method applied to equation 4.27, with terms fitted to simulated devices with lateral extension abruptness of 1.9 nm/dec. (a) α factor. (b) Extracted R_{sd} from shift-and-ratio method. (c) R_{sd}^{-} from equation 4.33. (d) R_{sd}^{+} from equation 4.33. All resistance in Ω

where

$$\hat{R}_{sd}^{+} = \frac{\frac{\tilde{L}^0}{\tilde{L}^i}\alpha\frac{\tilde{R}_{sd}(V_g-\delta)}{\tilde{R}_{sd}(V_g)} - 1}{\frac{\tilde{L}^0}{\tilde{L}^i}\alpha - 1}\tilde{R}_{sd}(V_g) \tag{4.34}$$

$$\hat{R}_{sd}^{-} = \frac{\alpha - 1}{\frac{\tilde{L}^0}{\tilde{L}^i}\alpha - 1}\tilde{L}^0\tilde{f}_{ch}(V_g - V_t^0) \tag{4.35}$$

Figure 4.13 shows the result of applying the shift-and-ratio method to the model

represented by equation 4.27 for the example. $\hat{R}_{sd}^{+}$ (Figure 4.13d) approximately equals the total physical resistance ($R_{co}+R_{spacer}+R_{ov}$). However, $\hat{R}_{sd}$ as extracted by the shift-and-ratio method (Figure 4.13b) does not equal the total physical resistance, the desired value. This is because the extension is influenced by the gate and α can be smaller than unity

$$\alpha \approx \frac{1}{1+\frac{R_{sd1}f'_{sd}(V_g-\delta)}{L^i f'_{ch}(V_g-V_t^i-\delta)}} \tag{4.36}$$

This in turn causes $\hat{R}_{sd}^{-}$ to become negative (Figure 4.13c). In fact, the value of this negative term is sufficiently large to cause the extracted R_{sd} to be smaller than R_0, as we observed in Section 4.8.1. Note that the negative resistance component is simply a result of violating the assumptions of the extraction method, and has no physical meaning.

Further comments on the extracted L_{eff} are in order. Firstly, equation 4.32 reduces to $\frac{L^0}{L^i}$ when $V_t^0 = V_t^i + \delta$ and either the extension resistance is small or substantially less dependent on gate bias than the channel resistance. However, as we noted before, the extension is measurably influenced by the gate and α can be smaller than unity. This is especially pronounced for gradual junctions. As a result, L_{eff}^i increases for more gradual junctions, as was noted in Table 4.4.

Secondly, in Section 4.8.1, it was observed that the extracted L_{eff} could be smaller than L_{met} for devices with abrupt junctions. The key thing to note is that the difference between the two terms in equation 4.27 lies in the fact that the latter is proportional to L. Due to counterdoping and short channel effects[12], the edge of the channel behaves differently from the center region, and is not scaled with channel length. The extraction routine cannot distinguish between this and the source/drain

[12]The following applies also in the presence of halo and reverse short channel effects.

resistance, thus the former is lumped together with the latter. As a result, the "base" channel length L is actually shorter than L_{met}. The extracted device length L (as defined by equation 4.27) for the 50 nm device in Section 4.8.3 is 14 nm, smaller than the metallurgical channel length of 22 nm. At the same time, a small α factor of about 0.67 is obtained (Figure 4.13a). The L_{eff} of 20.9 nm in Table 4.4, extracted using the shift-and-ratio method, then result from dividing 14 nm by the factor of 0.67.

4.8.5 Discussion

Extraction results obtained when the assumptions for the extraction method are violated will lose physical meaning. Violation of these assumptions for deep submicron devices become more likely due to the complexity of the device physics. This does not by any means invalidate shift-and-ratio and other extraction methods, and they remain tremendously useful. However, conclusions about the physical device, such as the metallurgical channel length and lateral doping profile [10], obtained from these extraction methods have to treated with caution [4]. All major assumptions of the methods have to be examined for their impact, and relaxed if necessary.[13] Alternatively, one can treat the extraction method as the *definition* of electrical quantities, to be used only in a context that is consistent with the extraction method [90], and rely instead on physical techniques such as those described in [79] [82] [85] for obtaining the physical quantities.

The extraction procedure described in Section 4.8.3 relaxes the assumption that

[13]For example, the extraction method described in Section 4.8.3. Alternatively, inverse modeling techniques [42] [46], with proper choice of physical models, may be used.

the source/drain resistance be gate-bias independent, as imposed by the shift-and-ratio method, and could in theory be applied to experimental data. However, the requirement for measurement of devices with three different channel lengths will make it vulnerable to statistical process variations [11].[14] The presentation and use of equation 4.27 in this chapter is intended solely as a vehicle for error analysis and interpretation of the observed trends.

4.9 Conclusions

A rigorous method for calculating bulk and contact resistances using device simulation that properly takes into account the two dimensional nature of current flow in scaled MOS devices is presented. This is compared with two other resistance calculation strategies. Resistance calculations based on partitioning the device into vertical strips without regard to the potential distribution are shown to yield substantial errors where current spreading is important, and can result in an overestimation of the benefits of exploiting abrupt junctions. At the same time, the sheet resistance equation based on the quasi-Fermi level at the Si/SiO_2 interface is shown to be identical to a special case of the generalized method.

The shift-and-ratio method is applied to simulated data and the results are correlated with the physical resistances of the device calculated using the methods described in this chapter. Two of the assumptions of the shift-and-ratio method are violated for deep submicron devices: 1) that the source/drain resistance is gate-bias independent 2) that the channel resistance is directly proportional to channel length.

[14] This is not a concern for simiulated data and for understanding the operation of the shift-and-ratio method.

As a result, the effective channel length can be shorter than the metallurgical channel length and the extracted source/drain resistance can be smaller than the physical values. A new extraction method that relaxes some of these assumptions is presented.

Ultimately, the extracted value from the shift-and-ratio method makes sense only within the context of the assumptions on which the method is based. It may be best to treat the extracted quantities as electrical values that may be useful even if they do not correlate fully with the physical quantities, and rely on more physical techniques for extracting physical parameters such as the lateral doping profile and metallurgical channel length.

Chapter 5

Software Implementation

The study of lateral abruptness reported in this thesis has several challenges. A large number of devices have to be simulated, studied and compared. All these simulations must be managed. Moreover, current TCAD simulation tools do not provide rigorous calculations of resistive components in the device, which as seen in Chapter 4 is important for understanding lateral abruptness. It was therefore necessary to develop custom software to overcome these challenges.

The design and implementation of TCAD software does not always get the attention that it deserves, since TCAD software is primarily a means to an end, an enabler to facilitate better physical understanding and, ultimately, better design of devices. The result is often software programs that are less robust, less flexible, and ultimately, less useful than they otherwise might be.

At the same time, software engineering as a discipline has advanced significantly in the past two decades. Object-oriented design and analysis [12], version control, black box/white box testing, and formal software development processes with an emphasis on quality control all have contributed to improvements in software robustness,

maintainability, extensibility and usability. While research efforts such as ALAMODE [88] and PROPHET [63] point the way to how next generation TCAD applications can be built, the current crop of TCAD simulators have their roots in the 1970s and 1980s, without the benefits of these new software engineering methodology. While device and process engineers have become experts in working around the limitations of the software, better design and implementation of TCAD software would ultimately provide a boon to their productivity.

For this work, considerable effort has been expended to design and implement software to overcome the limitations of existing TCAD simulators. Along the way, software engineering techniques are employed where feasible to enhance the quality of the software. This chapter begins with an examination of the software approach used by this thesis to solve various simulation issues and the design decisions encountered in the software implementation (Sections 5.1, 5.3, 5.2, and 5.4), and concludes with a discussion of how software engineering can be applied to the development of TCAD applications (Section 5.5). The Unified Modeling Language (UML) [61] is used throughout to help illustrate graphically the design of the code.

5.1 Device Simulation Template

A parameterized device description provides a template for easy generation of devices with different designs. A good parameterization allows control of all the device parameters of interest and is useful for device optimization and exploration of the design space. Figure 5.1a shows a schematic diagram of the key geometric parameters in the device, such as the spacer length (L_{spacer}), the overlap spacer length ($L_{overlap}$), the gate length (L_{gate}) and the oxide thickness (t_{ox}), according to the parameterization

adopted in this thesis. This parameterization is straightforward, except possibly for the addition of the overlap spacer length, which allows the gate-extension overlap to be tuned and was important for the investigation in Section 3.5. Figures 5.1b and c shows the parameterization of the doping in the source/drain region, which is based on tensor-products as presented in equation 3.1. An application of this parameterized description can be found in Chapter 3.

While most of the commercial TCAD simulators provide their own proprietary scripting languages for creating a device, their capabilities tend to be rather limiting. To allow maximum flexibility and control in this thesis, instead of using the built-in scripting languages directly, a set of Python [50] scripts are used to generate the files needed for the simulations[1]. Figure 5.2 shows how these scripts fit into the device design process.

Many of the parameters in the device description form subsets that belong together logically. Consider the parameters shown in Listing 5.1. They are related as follows

$$L_{spacer,total} = L_{spacer} + L_{overlap} \tag{5.1}$$

To maximize flexibility and ease of use, the user can specify any two out of the three parameters in the set, and the scripts will automatically calculate the third. Using this feature, the designer can make comparisons of sets of devices with either the same total spacer lengths or the same overlap lengths with equal ease.

Listing 5.1: Partial Listing of the Device Structure Parameters

[1]Python is an open-source, dynamic, extensible and object-oriented scripting language that has been gaining popularity. Python runs on top of a virtual machine, and is inherently less efficient than a compiled language such as C/C++. However, given that the computation time is dominated by the simulations themselves and is impacted only weakly by the overhead of managing the simulations, this is not a major concern.

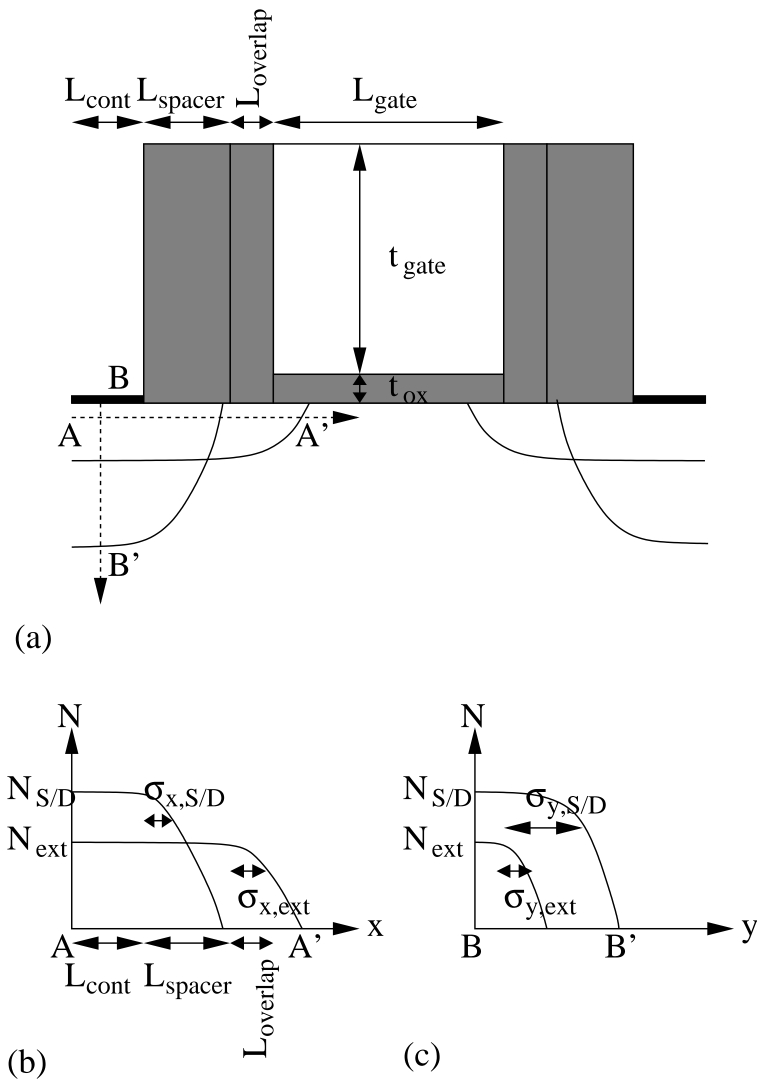


Figure 5.1: Diagram showing the key (a) geometric and (b, c) source/drain doping parameters in the device structure template

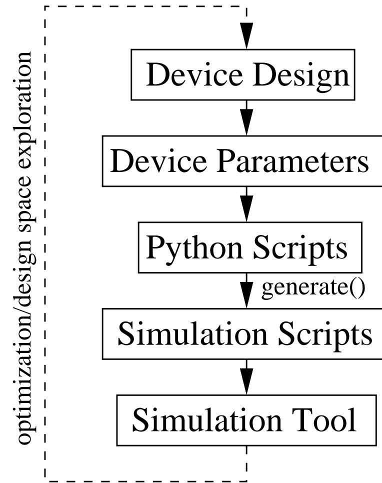


Figure 5.2: Flowchart showing the use of the Python-based parameterized device description scripts.

```
# Spacer
self.spacerMaterial = "oxide" # "oxide" or "nitride"
self.totalSpacerLength = 0.06 # specify 2 of the 3 lengths
self.basicSpacerLength = []
self.overlapSpacerLength = 0.0
```

5.1.1 Grid Generation

For simplicity and control, a tensor product grid is used for the simulations in this thesis. Figure 5.3 shows the key grid spacing parameters. Parameters h_{x1}, h_{x2} and h_{x3} represent the horizontal grid spacing far away from the gate, at the drawn gate edge, and at the metallurgical junction of the source/drain extension respectively. Parameters h_{y1} and h_{y2} are the vertical grid spacings at the Si/SiO_2 interface and the source/drain metallurgical junction respectively. Figure 5.4 shows such a tensor product grid with a specific instance of the grid parameter set.

In order to minimize local truncation error in finite difference calculations during

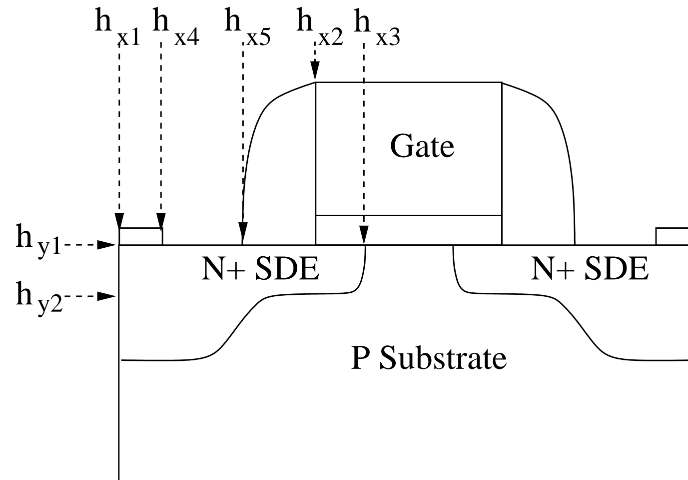


Figure 5.3: Schematic showing the grid parameters. h_{x1}, h_{x2} and h_{x3} are the key horizontal, and h_{y1} and h_{y2} the key vertical grid spacing respectively.

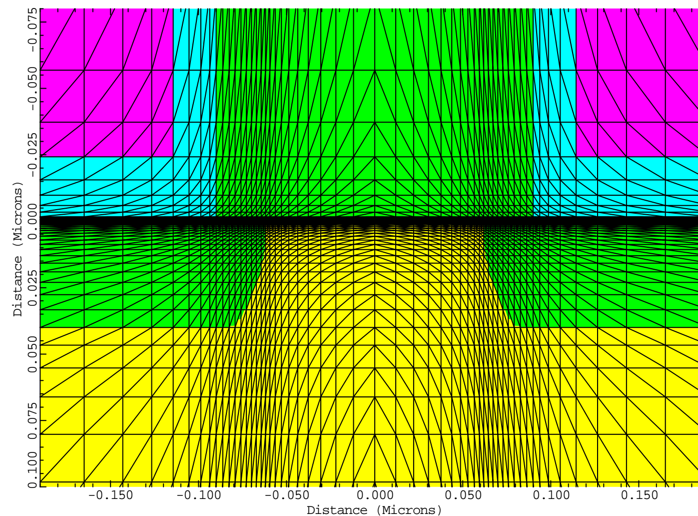


Figure 5.4: Tensor product grid with $h_{x1} = 25nm$, $h_{x2} = 3.7nm$, $h_{x1} = 2.5nm$, $h_{y1} = 0.28nm$ and $h_{y2} = 6.7nm$

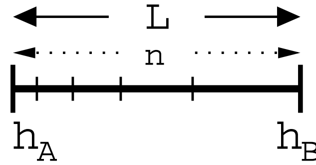


Figure 5.5: Grid section of length L subdivided into n spaces with grid spacing varying smoothly from h_A to h_B

the TCAD simulations, smooth grid spacing transitions are maintained in the channel and source/drain extension regions as much as possible. A simple way of achieving this for a tensor product mesh is to make sure that the size of adjacent cells differ by no more than a constant factor, resulting in grid spacings that form a geometric sequence.

Consider a grid section as depicted in Figure 5.5. We know from the geometric series formula

$$h_B = h_A \cdot r^{n-1} \tag{5.2}$$

$$L = h_A \frac{r^n - 1}{r - 1} \tag{5.3}$$

where r is the ratio between the adjacent grid spaces. From these relationships, we obtain

$$h_B = \frac{h_A + (r-1) \cdot L}{r} \tag{5.4}$$

$$n = \frac{\log(\frac{L}{h_A}(r-1)+1)}{\log(r)} \tag{5.5}$$

These equations are useful when we need to enforce a dense grid spacing at one end of the grid section, while the other end should be “sparse”.

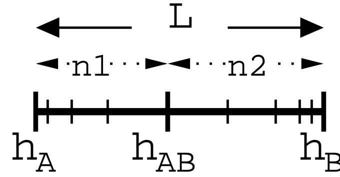


Figure 5.6: Grid section of length L with grid spacing varying smoothly from h_A to h_{AB} in n1 steps then to h_B in n2 steps

Similarly, for the grid section depicted in Figure 5.6, we have

$$h_{AB} = \frac{h_A + h_B + (r-1) \cdot L}{2r} \tag{5.6}$$

This equation is useful, for instance, when we want the grid section to have a dense grid at both ends and a sparse grid in the middle.

The equations described in this section have been implemented in and are used by the Python scripts in generating the simulation grid for the parameterized device.

5.2 Job Farming

The large number of simulations that must be run to fully explore the device design space[2] exacts heavy requirements on available computational resources. In order to keep computation times to manageable levels, a simple job farming system is implemented using the programming language Python [50] and exploiting standard Unix system functions[3]. Figure 5.7 shows a block diagram representing this job farming system.

[2]Simulation variables explored in this thesis include lateral abruptness, gate-extension overlap, channel doping concentration, channel length and mobility models.

[3]For instance, `ssh` for remote command executions, `fork()` and `pipe()` for spawning processes that allow concurrent execution and monitoring of multiple remote commands.

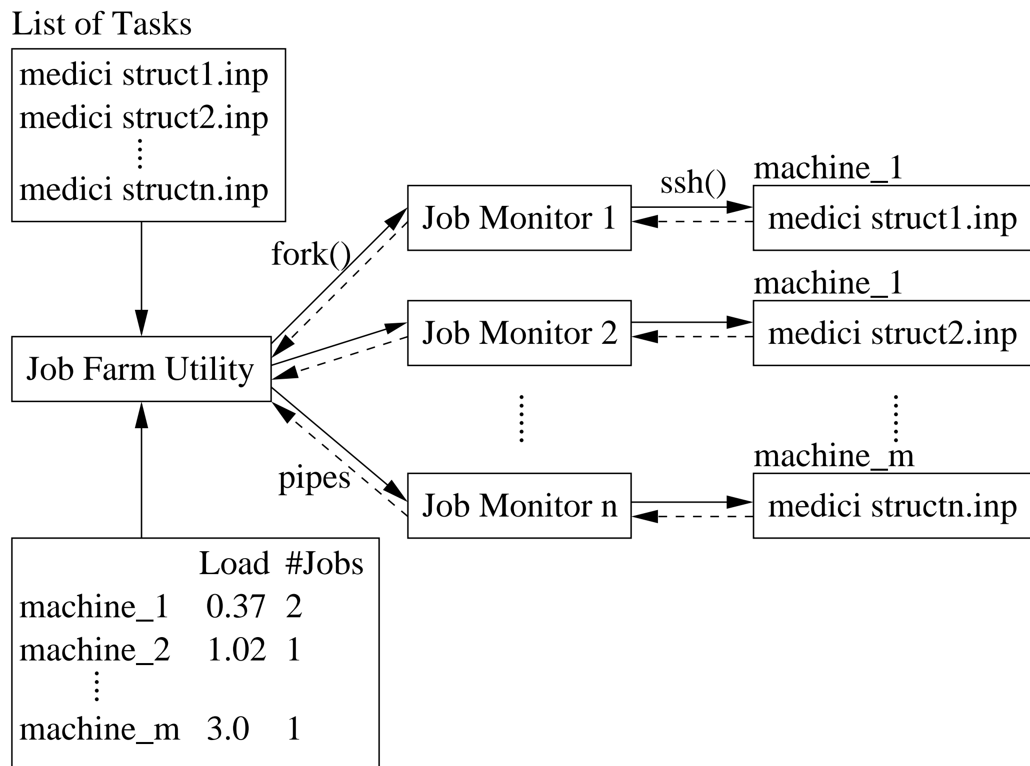


Figure 5.7: Farming jobs to a cluster of computers

When the user or a client application submit a list of tasks to be run, the job farming system begins by querying the load on all known machines in the cluster[4]. Simple load balancing is performed using a performance metric that takes into account the current load, the number of available processors and their performance rating for each machine.

$$\text{Expected Performance} = \frac{\text{Num Processors} \cdot \text{SPEC mark per processor}}{1 + \text{Machine Load}} \tag{5.7}$$

Using this metric, the submitted jobs are sent to the machines in order of highest expected performance.

A race condition exists due to the lag between when a process starts and when the new process is reflected in the system load measure. If multiple task lists are submitted before the machine loads can be updated, some machines become overloaded disproportionately. This is handled by adding a constraint which ensures the number of dispatched jobs per processor does not differ by more than a preset limit and thus evens out the number of active jobs started by the job farming system on all processors[5]. The current implementation keeps track of the number of dispatched jobs on each machine in a file. Access to this information by different processes is synchronized by using a file link as a lock[6].

When a new job is submitted, the job farming daemon begins by forking off a new child process. This child process is then responsible for starting the new job on the appropriate remote machine within the cluster, according to the load balancing

[4] Using the Unix utility `w`.

[5] A future enhancement would be to take into account the time the last job was started on each processor to determine if the load reading is trustworthy.

[6] Note that a file link can be used as locks due to the fact that the creation of a file link is an atomic operation for the NFS filesystem.

algorithm; for monitoring the progress of this job; and for returning its output to the parent process through UNIX pipes. Statistics such as the time taken for the completion of the task are also calculated.

Note that proper error handling is critical for the robustness of a job farming system. The parent process examines the output and restarts the job on a different machine in the event of error (for instance, if the remote machine goes down during the execution of the command).

5.3 I-V Database

The job farming system described in the previous section allows large numbers of simulations to be run. This in turn generates large volumes of data. To facilitate processing of the simulated data, a simple database was implemented in Matlab[7].

The simulated data are stored in database tables, implemented using Matlab arrays in the class `SimpleDatabase`. The key to the database is the `selectData` function. It allows the user to select only those rows of data that satisfy some general filtering criterion. For instance, I_d-V_g (or I_d-V_d) curves can be easily extracted from the complete set of I-V data by selecting rows with the same V_d (V_g). Multiple selects can be cascaded to further refine the resulting data set.

The `SimpleDatabase` class can be used to store general data of any nature. The `DataCV` and `DataIdVg` classes uses and specializes the `SimpleDatabase` class for handling storage of CV and I-V data (Figure 5.8). They provide routines for calculating

[7]Due to its interpreted nature, Matlab is not the most efficient way to implement databases. However, it does provide a full featured visualization environment for free.

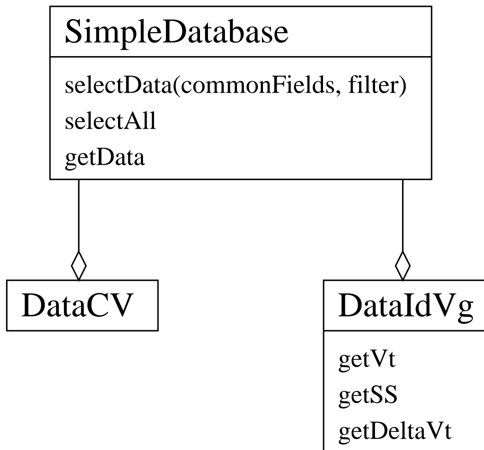


Figure 5.8: Class diagram for simulation data storage. This UML [61] diagram shows that DataCV and DataIdVg delegates to and utilizes SimpleDatabase for performing their functions.

the threshold voltage, sub-threshold slope and the difference in threshold voltages between two simulation conditions or between different device designs[8]. These routines have been used in Chapter 3 to manipulate, examine and visualize simulation results from various viewpoints.

We now proceed to examine the design principles and choices behind the implementation of the resistance calculation methodology[9] discussed in Chapter 4.

---

[8]The routines work by issuing the proper selectData statements, and post-processing the resulting data set behind the scenes afterwards.

[9]The program is implemented in C++. For a good reference on the C++ programming language, see Lippman, et al [47]

5.4 Resistance Calculations

5.4.1 Design Patterns

Object-orientation has emerged in the past decade as a dominant software analysis and design principle, as evidenced by the runaway success of object-oriented programming languages such as C++ and Java. Object-oriented programming promises software designs and implementation that are easier to maintain, more flexible, more extensible, and more amenable to future reuse through data encapsulation and class inheritance.

Design patterns [21] [29] extend these benefits further by capturing the essence of software designs that have proven useful in solving specific design issues. They specify the situations and conditions when a particular software design may be applicable, and provide a general template of how these software designs can be applied. This in turn encourages the adoption and use of good software architectures. Just as important, they help ensure that the code is readable and maintainable by providing common concepts and a common language for communicating software designs.

The rest of this section describes the design patterns used in the design and implementation of the resistance calculation code.

5.4.2 Mesh Object Class Hierarchy

The mesh is an important part of many numerical computation software. Before the physical quantities (or fields) can be solved for, the object of interest must be discretized, typically into points and elements. The physical equations are then solved. To make things even more complicated, for semiconductor applications, discontinuous fields, such as the electric field at the interface of two types of dielectric materials,

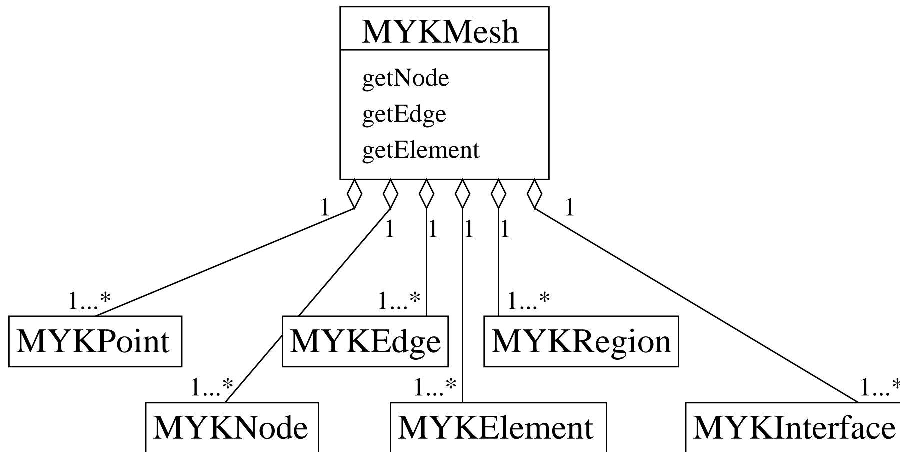


Figure 5.9: Composition of a mesh from a set of mesh objects

must be allowed for.

In the design of the resistance calculation code, the data associated with the mesh is managed by a set of mesh objects. Figure 5.9 shows how the overall mesh, represented by `MYKMesh`, is composed of mesh objects of various types, such as `MYKPoint`.

At the same time, the mesh objects participate in various relationships as indicated in Figure 5.11. Every node corresponds to a point. A point, in turn, could correspond to any number of coincident nodes (one for each region), each of which can have a distinct field value. This allows for discontinuous fields across region (material) boundaries. An edge is defined by its two end-points, while an element is defined by the bordering edges[10]. Along the same lines, a region is composed of an arbitrary number of elements.

These relationships are managed by having the objects participating in each relationship maintain references to each other. Lists are kept in the appropriate mesh objects to handle any many-to-one relationships. To allow for maximum code reuse,

[10]Three edges for a triangular element, four for a rectangular element.

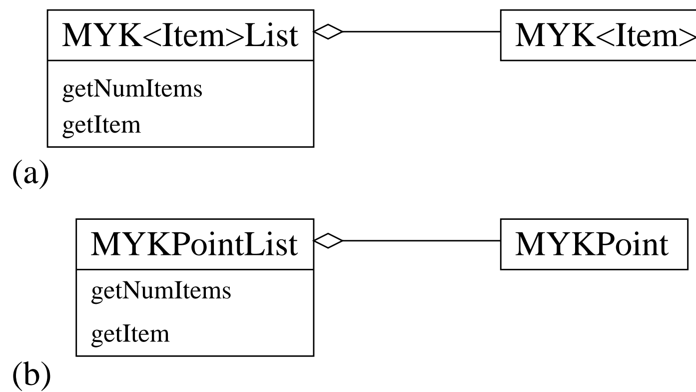


Figure 5.10: Template for lists of various mesh objects

these lists are created by using a class template, shown in Figure 5.10a. Figure 5.10b shows the instantiation of such a list for `MYKPoint` objects. The advantage of using templates is that it ensures a consistent API (Application Programming Interface) for all lists utilized by the application. This approach also provides a single point of maintainence: all changes and bug fixes to the template will propagate to all instantiated lists automatically.

Another point to note is that these mesh objects form a proper class hierarchy, as illustrated by the UML diagram in Figure 5.11. All mesh objects inherit from the superclass, `MYKObject`. This superclass defines shared attributes and behaviors. For instance, all mesh objects have a unique ID and maintain a reference to the mesh that contains the grid objects, instead of implementing these separately, they only need to be implemented once in the superclass. Another example is the inheritance of `MYKTriangle` and `MYKRectangle` from `MYKElement`. `MYKElement` defines the method `marchToNextEdge`, which all elements are required to implement[11]. This method

[11] In other words `marchToNextEdge` is required by the interface of `MYKElement`.

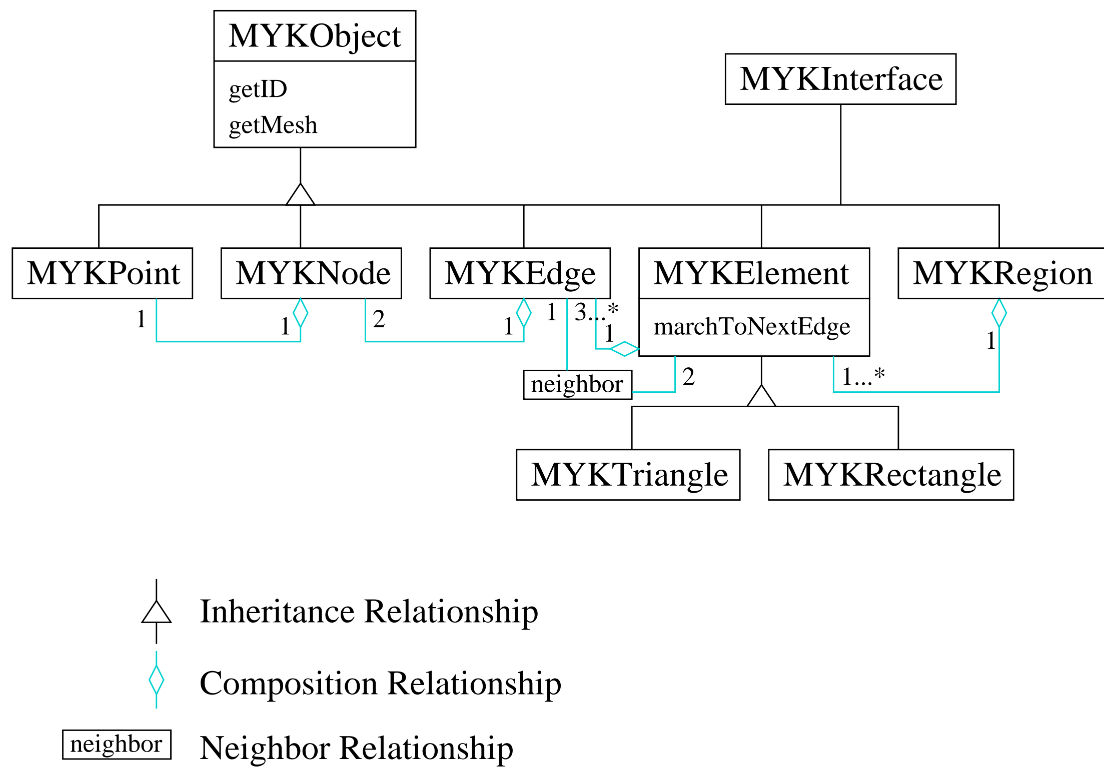


Figure 5.11: Class hierarchy for mesh objects used in the resistance calculation code

represents the portion of the marching element algorithm (Section 4.5.3) that is dependent on the element type. The general marching element algorithm, implemented by the method `extractContours`, can work with multiple element types by simply calling the polymorphic method. The runtime then takes care of dispatching to the proper routine. Note that other element types could be added simply by inheriting from `MYKElement` and implementing the required methods[12].

5.4.3 Proxy Design Pattern

While the class hierarchy shown in Figure 5.11 contains all the information needed for managing the mesh and the discretization, to minimize memory requirements, information about the relationships are stored in only one, not both, of the participating objects. As a result, the ease of information access is not symmetric. For instance, while the end-points of a given edge are stored directly in the edge object and can be obtained through a simple API call[13], the inverse operation (finding all the edges that are connected to a given point) involves iterating through all edges, which is inefficient for large meshes[14].

To strike a balance between performance and memory utilization, the proxy design pattern is utilized. Figure 5.12 shows two mapping classes, `Point2EdgeMap` and `Point2ElementMap` . These mapping classes allow reconstruction of relationship information not stored directly in the basic mesh objects. When a method such as `findEdgesWithPoint` is called on the object `MYKMesh`, an appropriate mapping object is created dynamically the first time it is needed. The edges in the mesh are

[12]Such as `marchToNextEdge`.
[13]An O(1) operation.
[14]An O(total number of edges) operation.

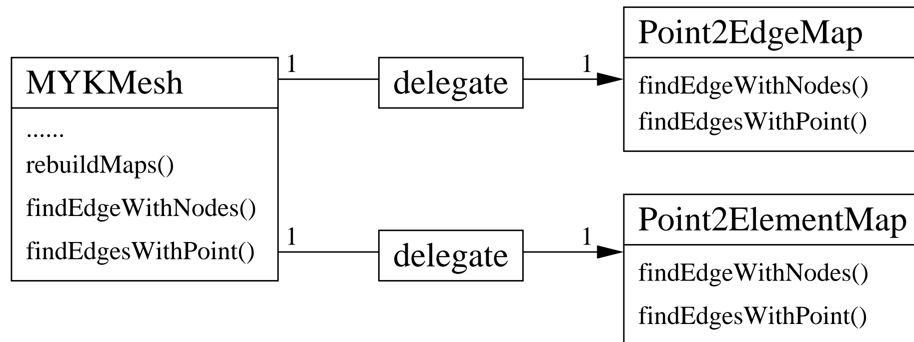


Figure 5.12: Proxy Design Pattern for `MYKMesh`

traversed to generate the map from the points to their adjacent edges. Information can then be retrieved by delegating to the helper mapping object. A reference to this map object is stored in the `MYKMesh` object to allow for efficient access of the reconstructed information in future calls to the method.

Stale information in the mapping objects can be avoided by updating these objects immediately when the mesh is modified (such as when more grid points are added to the mesh). Note that the mapping objects can be deleted to free up memory, and then regenerated as necessary.

5.4.4 Builder Design Pattern

The mesh objects shown in Figure 5.11 are used to store the mesh data during the runtime of the program. They cannot be used for permanent storage on the file system[15]. Explicit conversion of the mesh object hierarchy to and from the appropriate file format(s) is necessary.

For this thesis, the file format of choice is the TMA TIF (Technology Interchange

[15] Unlike Java, C++ does not support automatic serialization of objects in memory to and from a file.

Format)[7]. To isolate the details of reading and writing different file formats from the setup and construction of the mesh objects, a simplified version of the builder design pattern (Figure 5.13) is employed in the implementation. The main advantage of this approach is the easy extension of the code to other file formats[16]. Additional reader classes could share the top level flow for building the mesh objects, while the file format specific code remains isolated inside the `processXXX` methods.

5.4.5 Iterator Design Pattern

One final design pattern we shall look at is the iterator pattern. This provides a uniform interface for processing objects in a collection, regardless of how the collection is implemented[17]. This is put to effective use in the class `TMATifReader`. In the mesh objects, the edges defining the boundaries of regions and those defining electrodes are managed separately by `MYKRegion` and `MYKElectrode`. However, in the TIF file format, these boundary edges are stored in a single list, regardless of whether they pertain to a region, or an electrode. In converting the mesh objects to a TIF file, an iterator (Figure 5.14) is used to encapsulate the fact that the edges come from different types of objects. This allows the client code to iterate over all edges without regard to whether they come from regions or electrodes. The code can thus focus on the main logic without worrying about the complexity of various end conditions encountered during the iteration.

[16]For use with other simulation tools from other TCAD vendors.
[17]Arrays, linked list, hash table, etc.

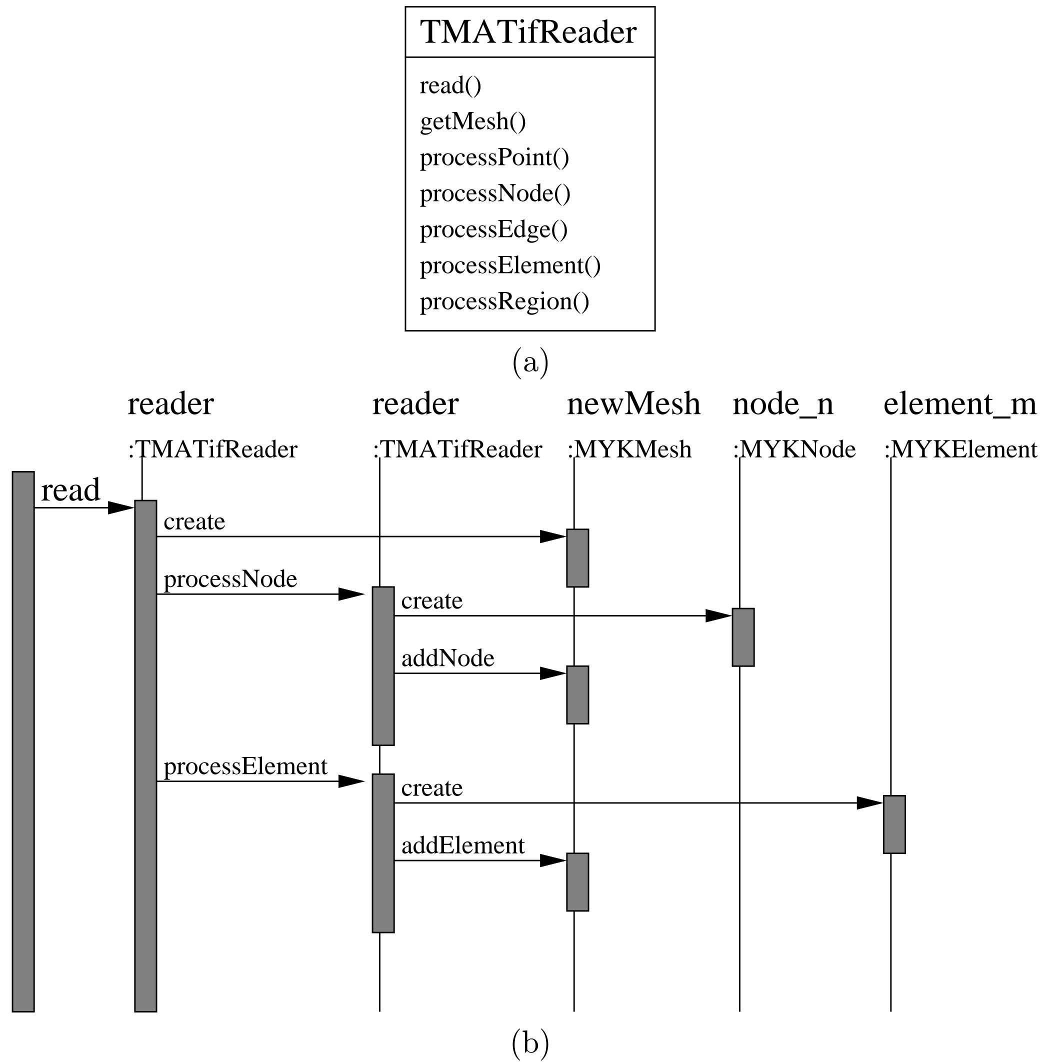


Figure 5.13: Builder Design Pattern for `TMATifReader`. The UML sequence diagram in (b) shows the timing flow during a typical `read()` call.

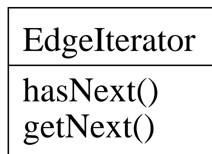


Figure 5.14: Iterator Design Pattern for accessing the boundary edges of regions and electrodes in the mesh

5.5 Software Engineering for TCAD Applications

5.5.1 Goal of Software Engineering

Software engineering aims to produce software code that is robust, flexible, extensible, and easy to maintain and use. Robust software should not crash, and should produce correct simulation results for all possible input conditions.

Flexible, extensible software allows it to be used in situations that may not have been envisioned by the original developers. It may even provide a full featured scripting language to allow the user to extend the functionality without rewriting the source code. Since the development of a full featured scripting language is non-trivial[18], and standard, full-featured scripting languages that are designed to be extensible exist[19], a modern tool should leverage and use these scripting languages as much as possible[20].

Maintainable software minimizes the cost of future code/design changes by anticipating them. Code implementing different functionalities are isolated from one another, through well designed interfaces, making it easier to introduce new functionality without introducing more bugs. Different components can be modified independently of each other. Collaboration and transfer of knowledge is also made easier as well with a clean design.

The key to achieving these goals lies in proper design and implementation of the code using a properly managed software development process[21].

[18]One-off, proprietary scripting languages, used in many of the commercial TCAD tools, tend to be limited and non-standard.

[19]TCL (Tool Control Language), Python and Perl being a few prominent examples.

[20]ALAMODE [88] is a good example of such a modern TCAD tool.

[21]Management can be done by developers exercising discipline, or through a more formal process.

5.5.2 Object-Oriented Design

As we discussed in section 5.4.1, through encapsulation, inheritance and polymorphism, good object-oriented design helps ensure software that is flexible, extensible and maintainable. Design patterns are formal templates describing well-known object-oriented designs that have proven to be useful in practice.

Encapsulation isolates the implementation of each subsystem from each other. Interactions between these subsystems happen through well defined interfaces. The implementation and functionality of each subsystem can hence evolve independently of each other without impacting the rest of the code, as long as the interfaces stay the same[22]. For example, it is not important whether the list is maintained in the `MYKNodeList` class (Figure 5.10) as a linked list or a dynamic array, as long as the predefined interface is respected. This allows the programmer to change the implementation in the future[23] without major impact to the rest of the code.

Inheritance promote code reuse through inheritance of functionality by children classes from their parents. All the classes shown in Figure 5.11 inherit from the class `MYKObject`, and therefore all of them inherit the concept of object id and the mesh they belong to. By localizing and sharing code, inheritance helps enhance maintainability. Any bug fixes and code changes to the code in the parent classes will be reflected automatically in the children classes. Compare this with the cut-and-paste approach, whereby the same code changes need to be duplicated in all the children classes, a time consuming and error prone process.

Polymorphism means that a method can be applied to objects of more than one type, with the outcome of the method dependent on the target object type. In

[22]Actually, some changes, such as the addition of methods to the interfaces, are allowed.
[23]For example, due to scalability or performance considerations.

many strongly typed languages[24], polymorphism is closely tied with inheritance. A child class inheriting from a parent is considered **to be** a parent (since it inherits and implements the same programming interface), and can be used anywhere the parent is allowed. Consider the `MYKElement` subtree in Figure 5.11. Client code designed with `MYKElement` in mind will automatically be able to handle `MYKTriangle` and `MYKRectangle` classes. Additional element types can be introduced simply by inheriting from `MYKElement`, and overriding any methods as necessary to implement element specific behavior.

5.5.3 Software Development Process

A good software design is not sufficient by itself. It must be implemented properly. A properly managed software development process can help tremendously in ensuring code that is robust and maintainable.

Proper and thorough testing of the code is by far the most important tool in ensuring code quality and robustness. Regression tests whereby a range of test cases with good coverage of possible input values are run. Any deviation of the test results from the expected results could indicate a bug that has been introduced by code changes.

Version control is another tool that, especially when combined with regression tests, could help ensure code quality[25]. By identifying the changes between a working version with a non-working one, the bug-containing code can be isolated, speeding up the debugging process. Version control can also facilitate collaboration by managing

[24]Such as C++ and Java.
[25]Examples of version control systems include sccs, rcs, and cvs.

the access and modifications of the same set of code files[26].

Other techniques that can assist in the development of robust, maintainable code include the establishment of standard coding styles for the development team, standards for comments and documentation of code, and informal or formal code reviews whereby the design and implementation is discussed and critiqued by other members of the team to ensure good, consistent standards are followed.

5.6 Summary

Several functionalities and features, such as rigorous resistance calculation, and flexible manipulation and post-processing of simulation data, are missing from commercial TCAD software. In this work, custom software is developed to perform these functions. As much as possible, the software developed for this thesis follows the process described in this chapter. Object-oriented design techniques and design patterns are employed to ensure extensible and maintainable code, with the designs in turn formalized using the Unified Modeling Language (UML). Version control (CVS) is employed for all code. Test cases are set up and regression tests are run regularly to ensure program correctness where appropriate. Comments of code paths and any tricky logic is provided in the code itself throughout.

The advent of TCAD software has brought about a revolution in semiconductor design. Yet many of the software tools in active use today are beginning to show their age. Proper use of modern software engineering techniques will allow increased productivity in the semiconductor community, both by making it easier for software

[26]Either through locking of files being modified, thereby allowing only one person to edit a file at any time (sccs); or through differencing and merging utilities that allows changes to the same files to be reconciled (cvs).

vendors to incorporate new features and models, and by providing more robust, easier to use software to device designers, who can then apply these new features in their design activities.

Chapter 6

Conclusion

There have been many advances by the semiconductor industry in the past three decades. Minimum feature sizes have been reduced by more than 2 orders of magnitude, while the clock speed increased by more than 4 orders of magnitude in the same time frame (Figure 1.1). However, as device dimensions continue to shrink, second order effects such as parasitic short channel effects (Section 2.2.1), reverse short channel effects (Section 2.2.2), source/drain resistances (Section 2.2.3) and quantum mechanical effects (Section 2.2.4) are becoming more and more important. In order to control these second order effects, the device designs needed for deep sub-micron devices are becoming more complex as well, involving complicated doping distributions such as super retrograde channel doping (Section 2.3.1), halo doping (Section 2.3.1), and source/drain extension (Section 2.3.2). There is a natural tendency to dismiss simple device designs such as a uniform channel doping as uninteresting and to focus instead on the optimization of devices with the full complexity of a modern super halo structure, given that it is known a simple doping design will not produce sufficient device performance in the deep sub-micron regime.

However, it is exactly due to the complexity of the modern device that a simpler doping design can be useful, even critical for providing insight. It is difficult to have an intuitive grasp of how different parameters in a complicated halo design impact different aspects of the device performance. In fact, the parameters could confound the effects of one another, making it difficult to truly understand the import of any particular parameter.

The misconception regarding the impact of lateral source/drain doping abruptness on device performance (Section 2.3.2) is due in part to the complexity of device designs used in deep sub-micron devices. The different degrees of interaction between the source and drain halo doping can be confounded with the effects of lateral abruptness (Section 3.6.1). This is revealed only by examining devices with uniform channel doping with various lateral source/drain extension abruptness, which isolates the effects of doping abruptness.

Another important observation of this work is the importance of choosing a proper metric for comparing the performance of different device designs. Focusing only on either series resistance (Section 3.3) or threshold voltage (Section 3.4) ignores important aspects of the overall device performance and could lead to a skewed picture of the impact of lateral abruptness. In contrast, I_{on}-I_{off} plots takes into account both drive- and leakage-currents to give a more complete picture of digital circuit performance achievable using a particular device design. The conventional I_{on}-I_{off} (Section 3.5.1) can be further improved by using doping rather than gate length as the design parameter (Section 3.5.2), and by using supernominal-I_{on} and subnominal-I_{off} (Section 3.5.3) instead of the nominal device values.

Using the supernominal-I_{on} versus subnominal-I_{off} plot, it is shown that the effect of lateral abruptness on device performance depends on the amount of gate-extension

overlap present. For devices with substantial underlap, increasing lateral abruptness improves performance by enhancing the coupling between the source/drain and channel regions (Section 3.3). For devices with sufficient overlap, however, increasing lateral abruptness could hurt performance, due to the degradation of charge sharing effects (Section 3.4.2). Increasing lateral abruptness therefore does not automatically lead to improved device performance, contrary to many results presented in literature.

Furthermore, it is shown that series resistance calculations depend greatly on the methodology chosen. The vertical strip calculation method (Section 4.4), on which the ITRS roadmap numbers are based, introduces over 25% error in the calculated value (Section 4.7.1). Moreover, the error introduced by this method increases with increasing lateral abruptness, in a way that tend to overestimate the benefits of a laterally abrupt source/drain extension.

Overall, the benefits of junction abruptness for device performance are much less than have been predicted, especially if lateral abruptness can only be achieved via expensive means. It is also noted that the amount of gate-extension overlap can have a significant impact on device performance, and it is useful to include gate-extension overlap as a device design parameter for optimization. Devices with a small overlap or underlap have optimal performance when supernominal-I_{on} subnominal-I_{off} curves are used as the metric (Section 3.5.3).

Considerable attention has been paid in this work to software engineering. While physical insight and workable device designs are the ultimate goals, by paying attention to the means to these ends, we could maximize productivity by producing software that is robust, extensible, maintainable and reusable. The extra effort expended in software design and practicing a structured TCAD software development process has been repaid many times over with the effort saved, certainly in this work

and hopefully by future generations of technology designers.

6.1 Contributions

1. In this thesis, various performance metric (series resistance, threshold voltage roll-off and 3 different types of I_{on}-I_{off} curves) and their appropriateness for the study of lateral abruptness are examined.

2. A detailed study of lateral abruptness and its impact on threshold voltage roll-off is presented, showing the threshold voltage roll-off does not improve monotonically with increasing lateral abruptness, contrary to what the existing literature reports.

3. Better understanding of the interaction of lateral abruptness and gate-extension overlap length is achieved. The gate-extension overlap is shown to have a dramatic impact on both device performance and the impact of lateral abruptness on device performance.

4. A rigorous method for calculating resistances in an active semiconductor device that fully takes into account the multi-dimensional spreading of current from the channel to the source/drain region has been developed.

5. Errors resulting the vertical strip calculation method for resistances are quantified. The vertical strip calculation method is shown to overestimate the benefit of lateral abruptness on device performance.

6. Better understanding of the impact of assumptions inherent in the Shift-and-Ratio method (for instance that the source/drain resistances are independent of

gate bias) is reached. Furthermore, the error that results when the assumptions are violated is quantified.

7. A new V_g-dependent series resistance extraction method that relaxes some of the assumptions of the Shift-and-Ratio method is presented.

6.2 Future Work

1. One way of extending this work would be to study of the impact of quantum mechanical effects on the conclusions of this thesis (through the use of the density gradient model).

   Quantum mechanical effects are taken into account very crudely in this thesis through the effective oxide approach. However, classical models predict the peak of carrier distribution in a MOS device is at the Si/SiO_2 interface. In reality, the peak of the carrier distribution is pushed away from the oxide into the substrate by quantization effects. This could affect current spreading, and should result in the shift in the optimal lateral abruptness

   (a) For this study, it would be necessary to extend the resistance calculation software to work with other simulation tools and file formats, especially PROPHET. This will allow the use of the use of more advanced physical models, such as the density gradient model, in the study.

2. A detailed study of the optimization of halo and source/drain doping on electrical characteristics would be timely and important.

   Fabrication of halo devices involve complicated processing steps and material chemistry. To simplify the study, one possible approach, is to again take a

simplified, parameterized representation of the doping, and vary the relevant parameters directly, in order to get a better understanding of the way device performance varies with different doping levels and placements.

3. Further development and examination of V_g-dependent R_{sd} extraction technique for halo devices is needed to verify the viability of this technique for real devices.

   A key issue that remains to be verified is whether the new technique would handle halo devices better than the shift-and-ratio method, which is known to have problems (even yielding non-physical parameters) with strongly doped halo devices.

Appendix A

Grid Sensitivity Study

A.1 Introduction

The second order physical effects described in Chapter 2 pose substantial numerical challenges in the modeling of deep sub-micron devices. Physical models, involving highly non-linear terms as well as stiffly coupled equations that will tighten requirements on both the grid and solver technologies, are needed for modeling quantization effects. Inclusion of the poly depletion effect requires physical simulation of the polysilicon gate region. In the bulk, sufficient grid density is needed for calculation of the series resistance, and to resolve the lateral abruptness of source/drain junctions. All these impose constraints on the minimum grid that can be used.

At the same time, we must avoid excessively dense grids which lead to unreasonable simulation times. This is especially important for device optimization and inverse modeling applications [86] where large number of simulations are required. Understanding and control of the grid requirements for modeling deep sub-micron devices continues to grow in importance.

While various automatic grid generation [68] [15] and grid refinement techniques exist, it is still useful to understand the sensitivity of simulation to changes in grid densities in different part of the structure. This is especially true since some of the commercially available gridding tools do not work perfectly, and could lead to excessive grid, obtuse triangles, and convergence problems. Moreover, some of the physical models used in industry are designed to work with regular grids and may not work well with the unstructured grid that some of these tools generate.

This appendix presents grid requirement guidelines for the modeling of deep submicron devices, based on a detailed study of MOS devices targeted at the 2001 technology node. Section A.3 examines the issue of selecting an appropriate grid to properly resolve a junction with given doping gradient. Section A.4 examines the dependence of grid sensitivity of drain current on the physical models chosen. Section A.5 examines the grid sensitivity of other electrical quantities such as series resistance calculations. Finally, Section A.6 presents some guidelines for selecting the appropriate grid densities under different conditions, and suggests that error versus grid density plots should be an integral part of the documentation of new models/model implementations.

A.2 Simulation Structure

For simplicity and control, a tensor product grid is used in this thesis to explore the grid sensitivity of drain current. Figure A.1 shows the key grid spacing parameters, which are varied systematically. Their impact on the simulated currents are then examined. In order to minimize local truncation error in the finite difference calculation, smooth grid spacing transitions are maintained in the channel and source/drain

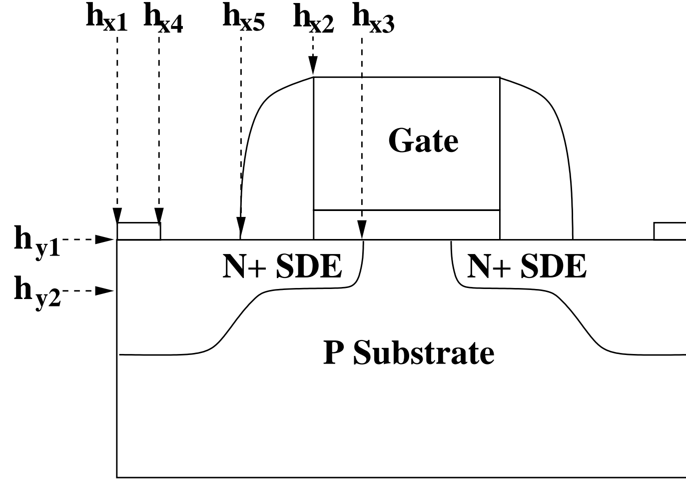


Figure A.1: Schematic of Simulated Devices. $L_{drawn} = 180$ nm. h_{x1}, h_{x2} and h_{x3} are the key horizontal, and h_{y1}, and h_{y2} are the key vertical grid spacing.

extension as much as possible (Appendix A.8). Parameters h_{x1}, h_{x2} and h_{x3} represent the grid spacing far away from the gate, at the drawn gate edge, and at the metallurgical junction of the source/drain extension respectively. Parameters h_{y1}, and h_{y2} are the grid spacings at the Si/SiO_2 interface and the source/drain metallurgical junction respectively. Figure A.2 shows an example grid generated with a particular set of parameters.

Most of the devices used for this study have lateral source/drain extension abruptness of 1.8, 3.5, 5 and 7 nm/decade. The gate oxide thickness vary from 1.8 to 1.0 nm, while the drawn gate length vary from 180 to 70 nm, corresponding to the 2001, 2004 and 2008 technology node on the ITRS [1]. Simulation results of these devices are used in the discussion throughout the thesis, and summarized in Section A.6.

We now begin by examining the grid requirements for resolving the doping gradient in abrupt junctions for modern devices.

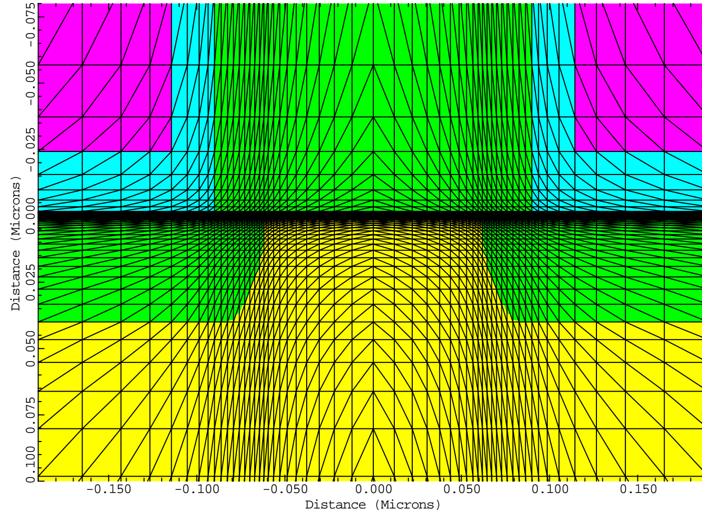


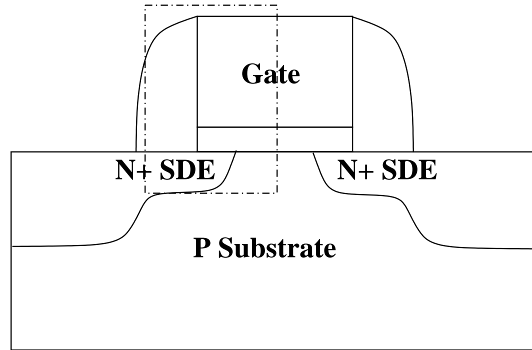
Figure A.2: Grid for Simulated Device with $h_{x1} = 25nm$, $h_{x2} = 3.7nm$, $h_{x1} = 2.5nm$, $h_{y1} = 0.28nm$ and $h_{y2} = 6.7nm$

A.3 Resolving Junction Gradient for Deep Submicron Device

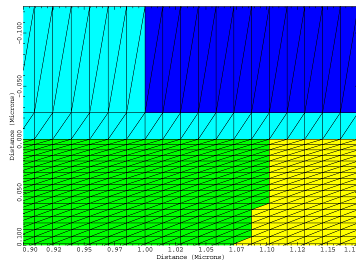
As we mentioned before, the abruptness of lateral gradients in the source/drain extension regions is an important parameter determining device performance for deep sub-micron devices [30]. In order to resolve the difference between devices with different source/drain abruptness, a sufficiently fine grid is required.

For the boxed region of the MOS device shown in Figure A.3a, a set of four meshes shown in Figure A.3b-e are considered. Table A.1 shows the simulation results for a $0.35\mu m$ device with lateral junction slope of 30 nm/dec with these four meshes. The variation in the simulated I_d due to the differences in the mesh is under 2%.

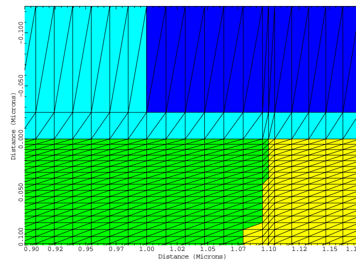
The mesh requirement is more stringent for a device with a more abrupt junction. As shown in Figure A.4, the simulated I_d for the device with an ideal abrupt



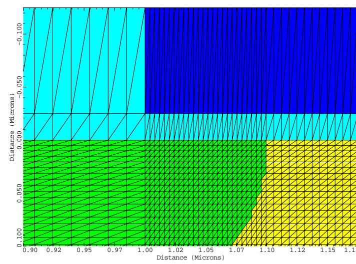
(a)



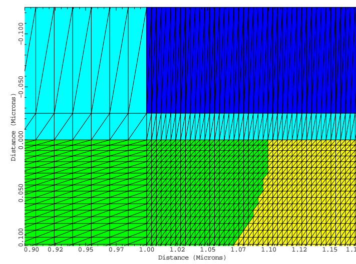
(b) Mesh i



(c) Mesh ii



(d) Mesh iii



(e) Mesh iv

Figure A.3: Resolving Junction Gradient (a) Schematic view of device (box indicates region of interest) (b) coarse grid (15 nm) (c) additional grid points around s/d extension junction (d) dense in extension region (e) dense in extension and channel (4 nm)

Table A.1: Simulated I_d for device with $L_g = 0.35\ \mu m$ and lateral junction slope of 30 nm/dec

Mesh	I_d	% diff wrt Dense
(i)	5.531e-5	1.4
(ii)	5.502e-5	1.9
(iii)	5.613e-5	0.05
(iv)	5.610e-5	0

junction using the coarse mesh is 15% smaller than the I_d observed using the other three meshes. The doping plot at the Si/SiO_2 interface (Figure A.5) shows that the coarse mesh is not sufficient to resolve the lateral abruptness of source/drain extension junction. As a result, the junction looks more gradual than intended, leading to a large simulation error. Adding only a few more grid points right around the junction (mesh ii) was sufficient to resolve the abrupt junction and thus recover the correct simulated I_d.

A.4 Grid Dependence of Drain Current and Choice of Physical Models

The optimal grid required for simulation of deep sub-micron devices is related not only to the doping profiles, but also to the choice of physical models. As we discussed before, scaling in device dimensions necessitates the incorporation of more complicated models, introducing additional nonlinearities, to account for poly depletion, quantization effects as well as bias dependence of carrier mobility. These models impose more stringent constraints on the grid needed for convergence and accuracy. It is therefore meaningful to talk about grid sensitivities in the context of specific physical

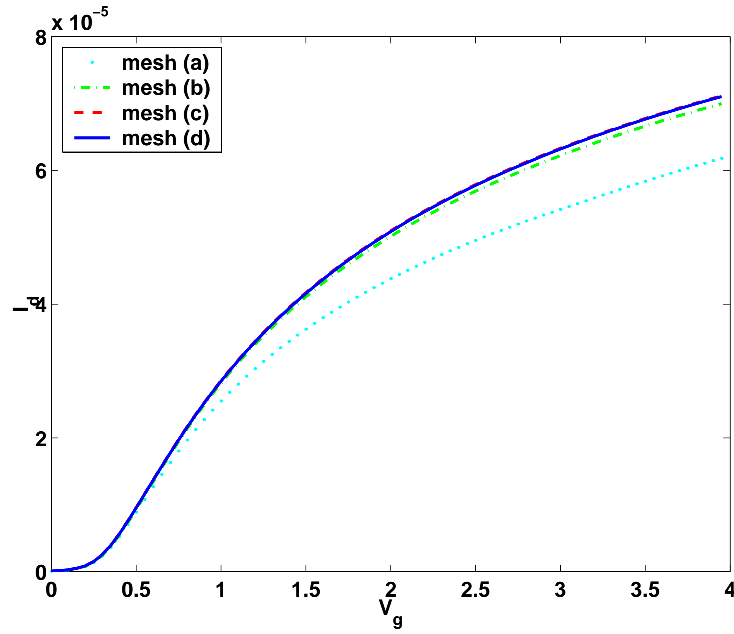


Figure A.4: I_d-V_g simulation for a device with $L_g = 0.35\mu m$ and ideal abrupt junction ($V_{ds} = 0.1V$)

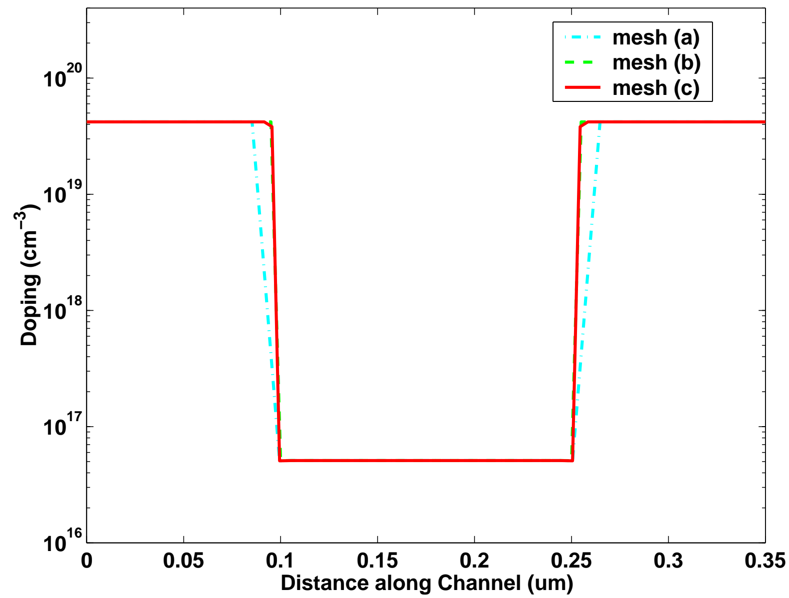


Figure A.5: Doping at the surface of the Si substrate for a device with $L_g = 0.35\mu m$ and ideal abrupt junction with various meshes

lat ext slope	h_{y2} 8 nm	2 nm	% Difference
3.5 nm/dec	3.5489e-5	3.5179e-5	0.9%
13 nm/dec	3.4266e-5	3.4300e-5	0.09%

Table A.2: Drain Current at $V_g = 1.5V$ and $V_d = 0.05V$. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{x3} = 0.4\mathring{A}$ and $h_{y1} = 1.4\mathring{A}$

models.

Section A.4.2 explores the grid sensitivity of several typical mobility models. Section A.4.1 examines the grid sensitivity due to the Van Dort Model for modeling charge quantization in the channel. Finally, Section A.4.3 explores the grid sensitivity of modeling polydepletion effects.

A.4.1 Quantum Mechanical Effects: Van Dort Model

At the same time, more accurate physical models are needed to simulate scaling effects beyond the current technology node. One example is the charge quantization effects due to both decreased gate dielectric thickness and increased substrate doping. A phenomenological model, such as the one proposed by Van Dort [81], provides a simple, empirical way of modeling charge quantization in the inversion layer. However, non-linear terms introduced by such models tend to degrade convergence properties of the simulator, as well as to increase the sensitivity of the results to grid spacing. We'll now examine the grid sensitivity due to the Van Dort model.

As expected, the simulated drain currents are insensitive to changes in grid density away from the gate edge (h_{x1}), at the gate edge (h_{x2}), and the bottom of the source/drain extension junction (h_{y2}). (For example, Table A.2 shows the impact of varying h_{y2} on I_{drain}.)

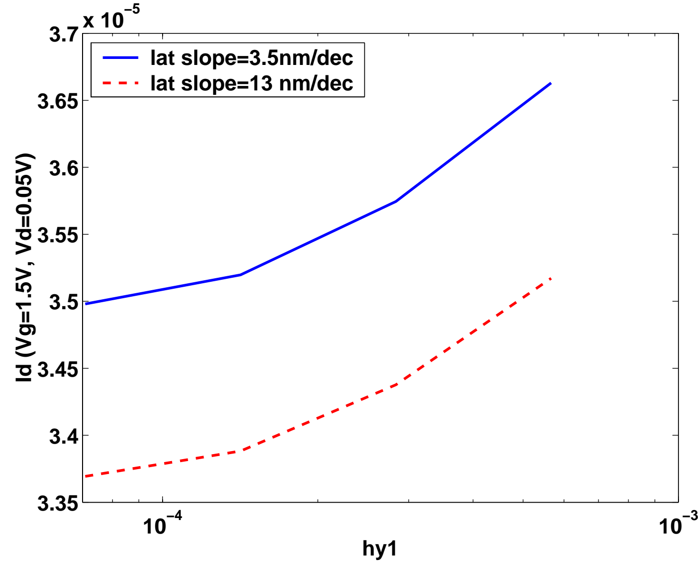


Figure A.6: Drain Current for various values of h_{y1}. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{x3} = .02nm$ and $h_{y2} = 8nm$

The current is however strongly affected by the vertical grid spacing close to the the Si/SiO_2 interface (Figure A.6), and the horizontal grid spacing at the source/drain extension junction under the gate (Figure A.7). From these figures it can be observed that a lateral grid density of 0.4Å around the junction, and a vertical density of 1.4Å close to the Si/SiO2 interface are needed to achieve an accuracy of 1% in the simulated drain current. Note that the grid requirements is much higher when using the Van Dort model.

A.4.2 Grid Sensitivity of Several Mobility Models

The mobility model used in the simulation could also have a significant influence on the grid sensitivity. CONMOB is a simple concentration dependent mobility model. UNIMOB stands for the universal mobility model which is similar to the enhanced surface mobility model reported by Watt [83]. GMCMOB stands for the Generalized

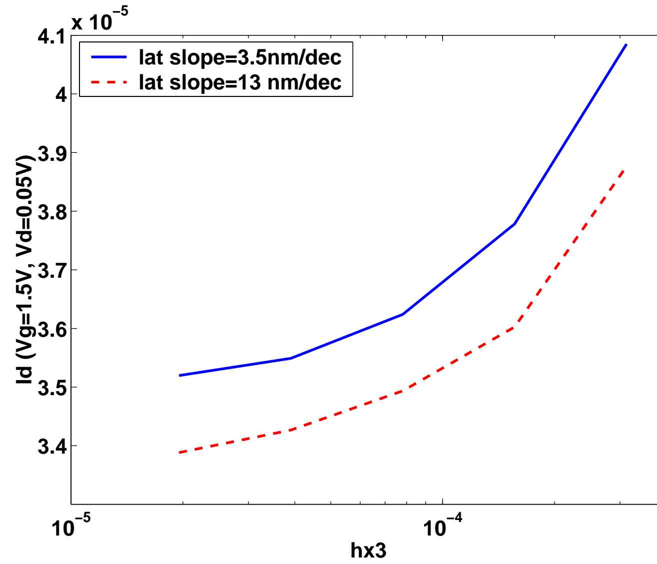


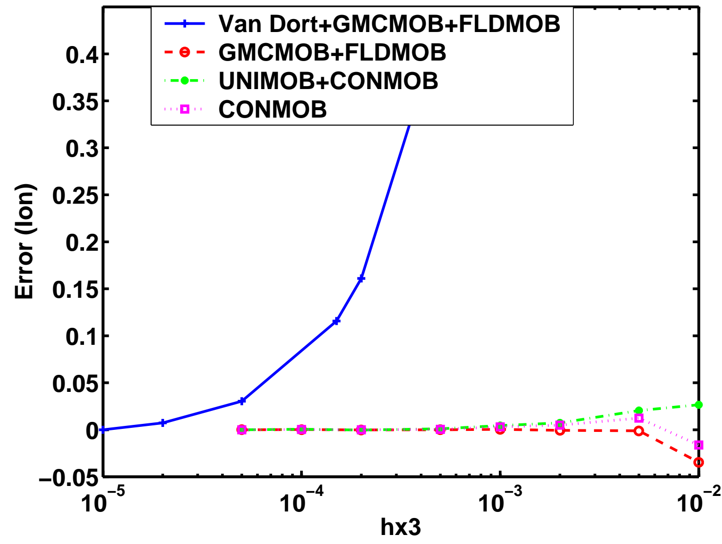
Figure A.7: Drain Current for various values of h_{x3}. $h_{x1} = 25nm$, $h_{x2} = 4nm$, $h_{y1} = .28nm$ and $h_{y2} = 8nm$

Mobility model reported by Mujtaba [55]. As expected, the grid requirements with respect to h_{x3} increases as more complicated physical models (UNIMOB, GMCMOB, Van Dort versus CONMOB) are used. Interestingly, the sensitivity of Ion to changes in h_{y1} is less for GMCMOB than for CONMOB.

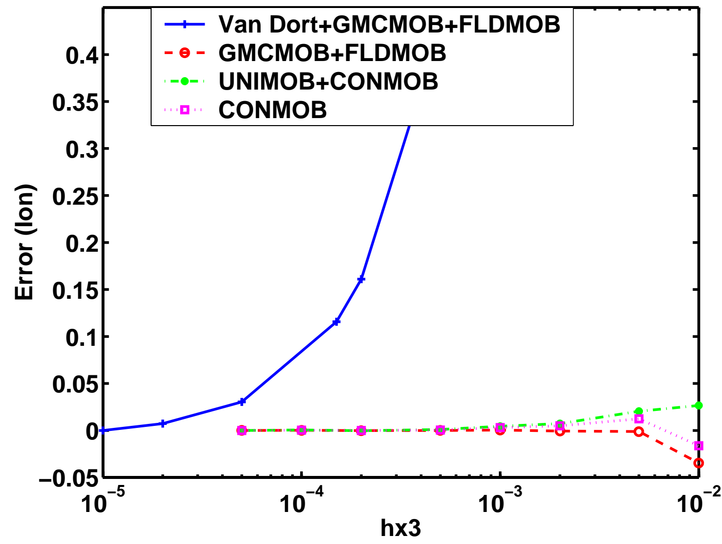
A.4.3 Polydepletion Effects and Grid Dependence

Instead of treating the gate as an equipotential body, modeling of poly depletion effects requires the solution of the semiconductor equations in the polysilicon gate region. Just as the grid spacing close to the Si/SiO_2 interface is critical for accurate simulation of the channel behavior, grid spacing in the polysilicon gate close to the interface will impact simulations where poly depletion is important.

Figure A.9 shows the CV simulation results with poly depletion taken into account for several gridded structures with different spacing in the polysilicon region. Note



(a)



(b)

Figure A.8: Error in the simulation Ion using the indicated physical models and grid with various (a) h_{x3} and (b) h_{y1}

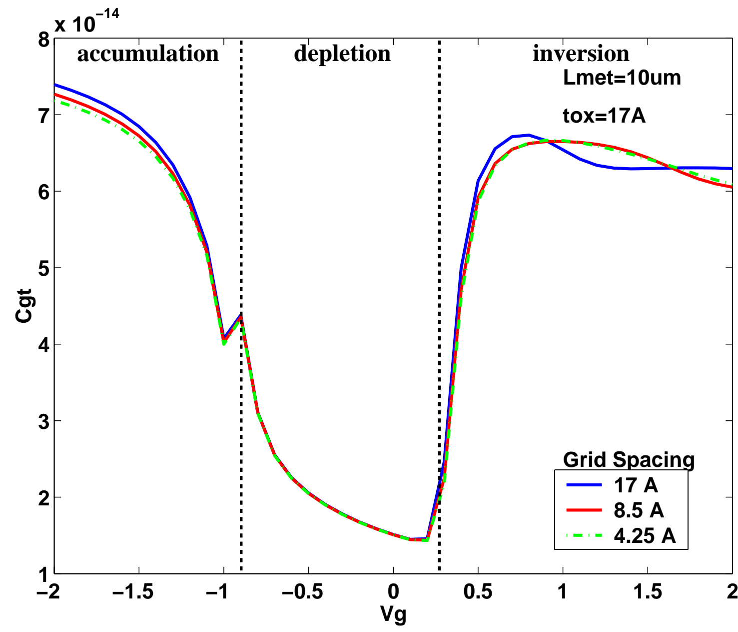
the discontinuity at flat band is due to the use of the Van Dort model [81] [16] for modeling quantization effects.

As expected, the grid spacing in the poly has significant impact on the shape of the CV curve in the inversion region, where the poly depletion effects are strongest. Somewhat surprisingly, the grid spacing in the poly silicon gate also has a significant impact on the accumulation capacitance and the depletion capacitance for the short channel device (Figure A.9b). The accumulation capacitance for the 17Å grid is 3% greater than that for the 2.1Å grid. This will translate to approximately the same degree of error for t_{ox} extraction applications.

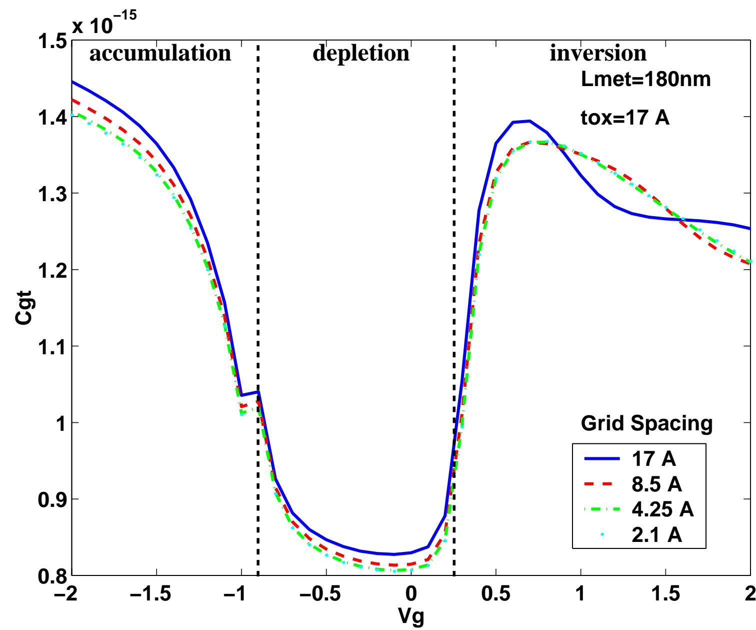
A.5 Grid Dependence of Other Electrical Quantities

As we saw in Section A.3 and A.4, simulation drain current for deep sub-micron devices has a strong grid dependence. This could be exacerbated if we are interested in investigating other key device design parameters. External resistance, effective channel length and threshold voltage are examples of key device parameters that are important in device design. As we shall see, while these parameters are closely related to drive current, the grid sensitivity for calculating these quantities to a specified level of accuracy can be quite different from one another. As a result, the optimal grid will depend on which physical quantities we want to calculate.

It is therefore important to examine the grid sensitivity of all the desired quantities directly in determining the appropriate mesh to use. It is not sufficient to study the grid sensitivity of one particular quantity and assume it is representative of other quantities of interest.



(a)



(b)

Figure A.9: CV simulations for different grid spacing in the polysilicon region close to the $poly/SiO_2$ interface. a. 10 um device. b. 180 nm device

Section A.5.1 examines the grid dependence of potential calculations. Section A.5.2 examines the grid dependence of applying the Shift and Ratio method (useful for extracting the effective channel length, source/drain resistance and threshold voltage) to simulated data. Section A.5.2 looks at the grid dependence of a different method of calculating external resistance.

A.5.1 Grid Dependence of Potential Simulation

Electrical potential is a critical quantity in understanding device operation. It is important to simulate potential to sufficient accuracy. Therefore, the local error in the potential calculation is often used to determine the amount of grid refinement necessary.

Figure A.10 shows the local potential error resulting from the simulation of an NMOS device from Table A.2. The grid densities are given by $h_{x3} = 0.4\text{Å}$ and $h_{y1} = 1.4\text{Å}$, $h_{y1} = 8nm$. A reference dense mesh is created by splitting each element in half in both the x and the y direction. The potential error is then calculated by comparing the simulated potential on the original mesh with the potential at the corresponding node on the reference mesh.

$$\delta\psi(x,y) = \psi(x_i,y_i)|_{x_i=x,y_i=y} - \psi(x_{i'},y_{i'})|_{x_{i'}=x,y_{i'}=y} \qquad \text{(A.1)}$$

We can see that the error in the potential is highest close to the source/drain junction. However, as we saw in Table A.2, I_{drain} is not sensitive to the grid density of the junction away from the channel. This is because the drain current is determined mostly by the channel and the source/drain extension close to the channel. The influence of the source/drain far from the channel on current flow is weak. A regrid

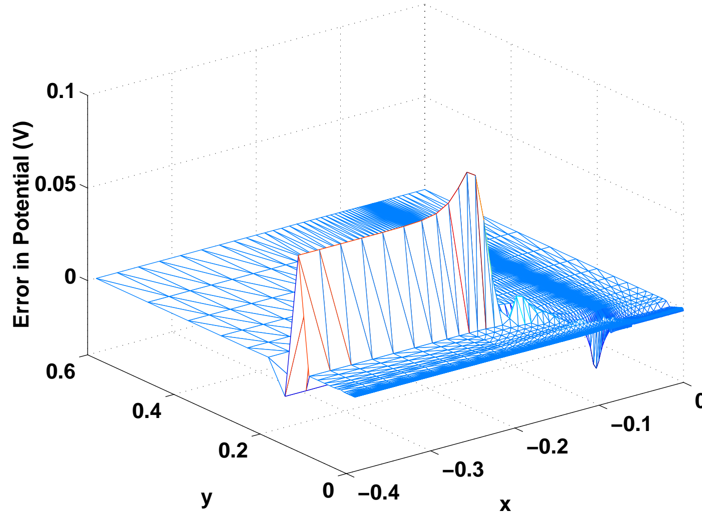


Figure A.10: Error in the Potential throughout the substrate of a device with extension abruptness of 3.5 nm/dec

conducted according to potential would produce a grid that could resolve the potential variation across the junction well, but that may not be optimal if drain currents is our ultimate concern.

A.5.2 External Resistance Calculations

It is well known that intrinsic current drive increases as the channel length is reduced. This implies that the channel resistance scales with the channel length of the device. On the other hand, the external resistance (due to contact resistance R_{co} and the resistance in the source/drain region) does not scale with channel length. In fact, the source/drain extension junction depth x_j must be scaled to control short channel effects which in turn cause the source/drain extension to increase. As a result, source/drain resistance becomes an increasing fraction of the total device resistance for each new technology node.

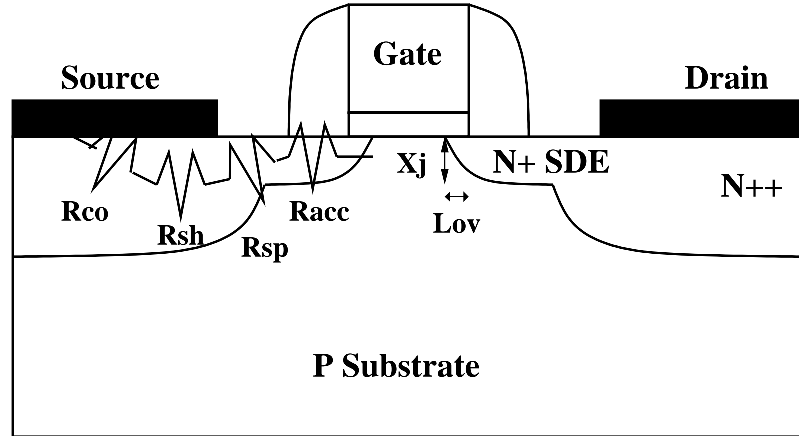


Figure A.11: Schematic of Typical HDD Device with Resistive Components in the Source/Drain

It is therefore important to be able to examine and calculate the external resistance. We now examine the grid sensitivity of the calculation of external resistance from the Shift and Ratio Method (Section A.5.2) and and the quasi-Fermi level (Section A.5.2).

Grid Sensitivity of External Resistance Calculations using Quasi-Fermi Level

Figure A.11 shows a schematic view of a typical MOS device, with emphasis on the source/drain extension regions. Also shown in the figure are two critical parameters that govern device performance: the extension junction depth X_j and overlap length L_{ov} [17].

It is assumed that current flow is largely parallel to the x-direction. The sheet resistivity can then be calculated from simulation results using an incremental form

of Ohm's law [54] [74]

$$I_{ds} = \int_0^\infty J(y)dy = \frac{d}{dx}\phi_n(x)\int_0^\infty qn(x,y)\mu(x,y)dy = \frac{\frac{d}{dx}\phi_n(x)}{R_{sh}(x)} \tag{A.2}$$

$$R_{sh}(x) = \frac{\frac{d}{dx}\phi_n(x)}{I_{ds}} \tag{A.3}$$

where x and y are the coordinates parallel and perpendicular to the Si/SiO_2 interface respectively, and ϕ_n is the electron quasi-Fermi level obtained from the two-dimensional device simulations. I_{ds} is the total current and is independent of x due to current continuity along the channel.

In order to properly model the accumulation layer in the extension region, the unified mobility model as proposed by Mujtaba [55] is used in the simulations [17].

Figure A.12 shows the results of the resistivity calculations for two different meshes at various gate biases with drain bias set to 0.05V. The coarse mesh has a spacing of approximately 5 nm along the channel. The dense mesh has 3 times more grid in both the vertical and the horizontal directions compared to the coarse mesh.

Figure A.13 focuses on the simulated resistivity for high gate bias case ($V_{gs} = 1.5V$), with drain bias held at 0.05 V. The relative error between the 2 meshes increases from 9% for the 65 nm to 10.4% for the 40 nm, indicating an increase in the grid density requirements as the extension junction depth scales.

The grid sensitivity becomes even more significant as drain bias increases. As shown in Figure A.14), with $V_{gs} = 1.5V$ and $V_{ds} = 1.5V$, the relative error between the two meshes increases from 11% for the 70 nm device to 33% for the 30 nm device.

Figure A.15a shows simulated resistivity for the same device under identical bias conditions as in Figure A.14, focusing on the drain extension region. To further understand the grid requirements for the resistivity simulations, included in the figure

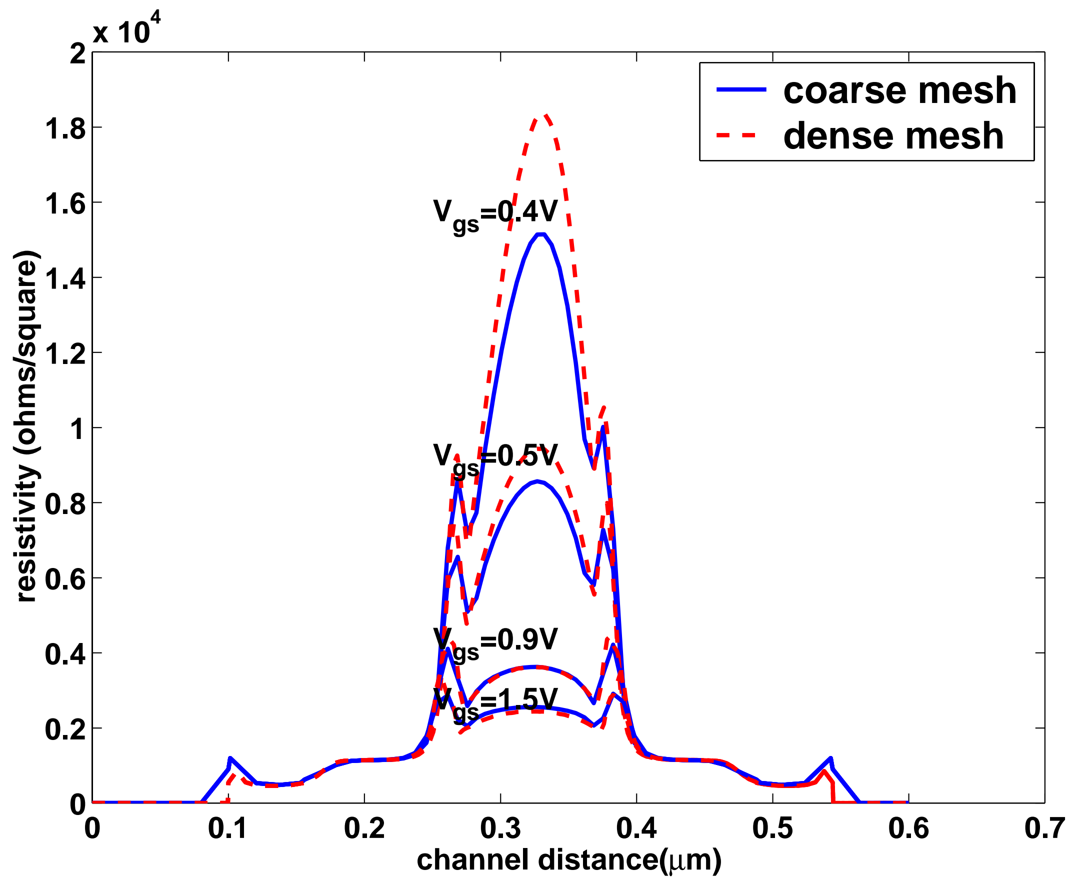
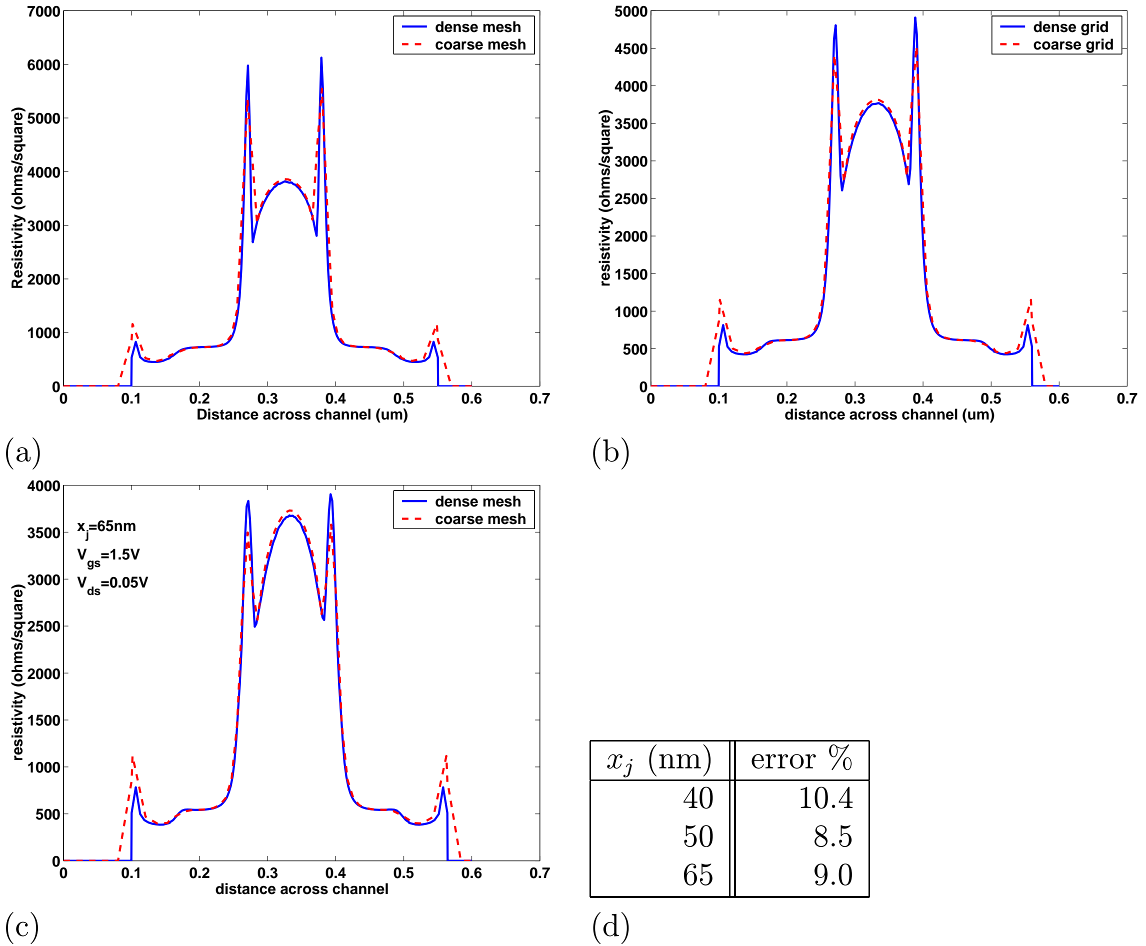
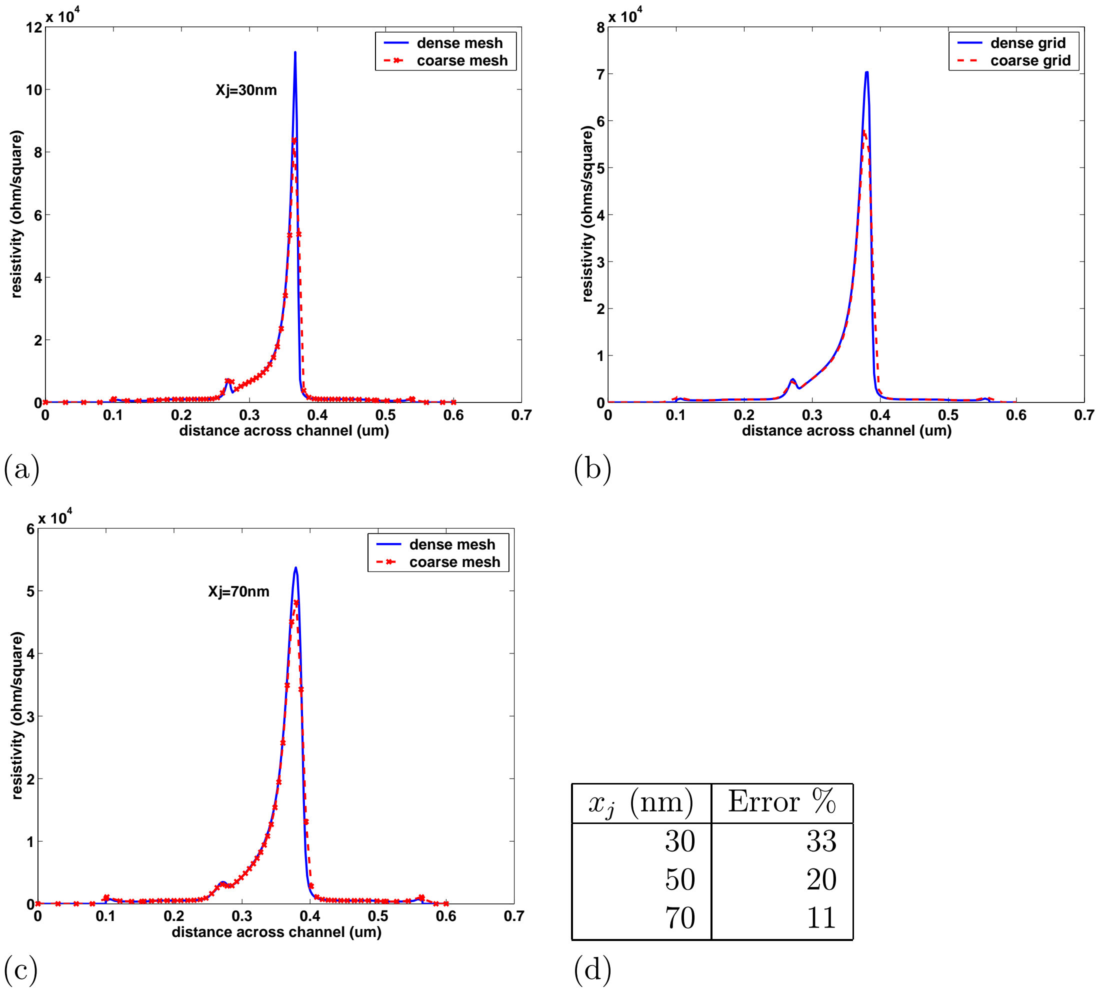


Figure A.12: Simulated External Resistivity along the channel direction at various V_{gs} biases. $V_{ds} = 0.05V$, $L_{eff} = 0.08\mu m$ and $X_j = 40nm$



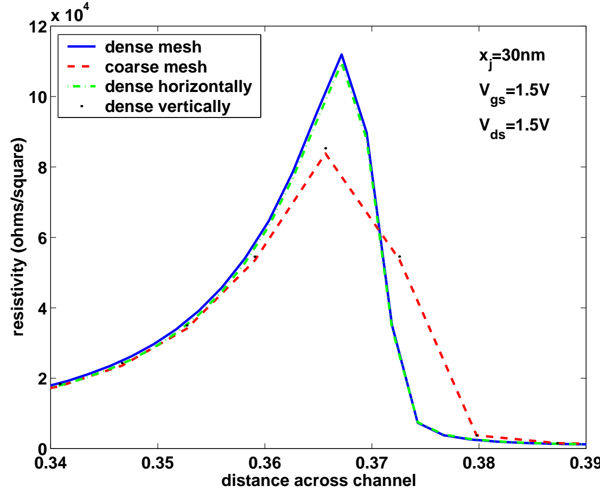
x_j (nm)	error %
40	10.4
50	8.5
65	9.0

Figure A.13: Simulated External Resistivity along the channel direction for devices with various extension junction depth. $V_{gs} = 1.5V$, $V_{ds} = 0.05V$ (a)$x_j = 40nm$ (b)$x_j = 50nm$ (c)$x_j = 65nm$ (d) Peak Error Between the 2 meshes



x_j (nm)	Error %
30	33
50	20
70	11

Figure A.14: Simulated External Resistivity along the channel direction for devices with various extension junction depth. $V_{gs} = 1.5V$, $V_{ds} = 1.5V$ (a)$x_j = 30nm$ (b)$x_j = 50nm$ (c)$x_j = 70nm$ (d) Peak Error Between the 2 meshes



(a)

Mesh	I_{drain}
Coarse	5.6697e-4
Dense Horizontally	5.7547e-4
Dense Vertically	5.6179e-4
Dense	5.6604e-4

(b)

Figure A.15: Simulated (a) External Resistivity along the channel direction and (b) drain currents for four different meshes. $V_{gs} = 1.5V$, $V_{ds} = 1.5V$ and $x_j = 30nm$

are the coarse and the dense mesh as before, together with a mesh that is dense horizontally and another that is dense vertically. We can see that the difference between the dense and the coarse mesh is due mainly to the mesh density in the lateral direction: refining the vertical mesh density causes the resistivity to change by 2%, with the horizontal mesh accounting for the remaining 31%. This can be understood by considering the fact that the resistivity is calculated as the derivative of the quasi-Fermi level in the horizontal direction. A fine horizontal grid is therefore needed to resolve the resistivity, especially in the high drain bias case.

The table in Figure A.15(b) shows the simulated drain currents using the different meshes for the same device. The error in simulated drain current for the different meshes is no larger than 1.7%, much smaller than the 33% error observed in the resistivity calculation. Hence while the coarse mesh is not acceptable for resistivity studies, it will do fine for simulating drain currents. Obviously, the grid requirement

for a specified level of simulation accuracy depends on the exact quantity that we are interested in.

Grid Sensitivity of Shift and Ratio Method

The technique of "Shift and Ratio" allows the extraction of effective channel length, threshold voltage roll-off and source/drain resistance from I-V characteristics [72]. The key is to find the amount of shift δ needed for the S^i vs. V_g curves for a long channel device and a device with unknown channel length such that their ratio r remains constant, independent of gate voltage, where the quantities S^i, r, and δ are defined as follows

$$r(\delta, V_g) \equiv \frac{S^0(V_g)}{S^i(V_g - \delta)} \tag{A.4}$$

$$S^i(V_g) \equiv \frac{R^i_{tot}}{dV_g} \tag{A.5}$$

with

$$R^i_{tot}(V_g) = R_{sd} + L^i_{eff} f(V_g - V^i_t) \tag{A.6}$$

The voltage roll-off is then given by

$$\Delta V_t = \delta \tag{A.7}$$

while the channel length can be solved from

$$L^i_{eff} = \frac{L^0_{eff}}{< r >_{\delta min}} \tag{A.8}$$

To examine this widely used technique in the context of device modeling, refer again to the devices simulated in Section A.4.1. Figure A.16 shows the V_t and R_{sd} vs.

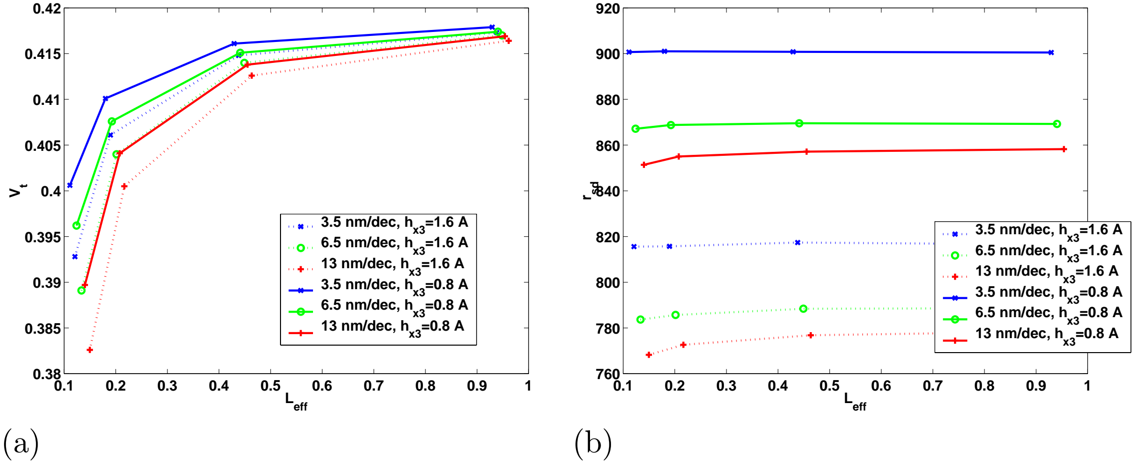


Figure A.16: (a) Threshold roll-off curves and (b) R_{sd} for devices with various gate lengths from an 180 nm technology, simulated using meshes with $h_{x3} = 1.6$Å and $h_{x3} = 0.8$Å

L_{eff}, extracted using the Shift and Ratio method for two different meshes. Notice the difference in the V_t roll-off as well as in the extracted R_{sd} due to the lateral abruptness is masked by sensitivity to the grid. We must therefore make sure that comparable meshes (as defined by the simulation error) must be used in simulations comparing devices with different lateral abruptness using the Shift and Ratio technique.

Note also that while the error between the drain currents at an applied bias of $V_g = 1.5V$ and $V_d = 0.05V$, simulated using the two meshes is 4%, the extracted R_{sd} on the other hand differs by more than 9%; and while the threshold voltages extracted from the two meshes differ by only 1.5%, the threshold voltage roll-off (ΔV_g) differs by almost 42%. This once again shows the different grid sensitivities of different electrical quantities.

A.6 Guidelines on Grid Densities

Choosing the appropriate grid density is important in achieving sufficient accuracy while keeping simulation times at a minimum. Choosing a grid that is too coarse will lead to errors in the simulation results. To make matters worse, the resulting error will depend on the dimensions and doping details of the device being simulated. Comparisons between these devices using too coarse a grid could thus lead to misleading results. On the other hand, choosing a grid that is finer than needed will lead to excessive computational requirements.

Unfortunately, the optimal grid needed in TCAD simulations to achieve a given accuracy is a complicated function of the doping profile (Section A.3), the chosen physical models (Section A.4), the bias conditions and the physical quantities we are interested in studying (Section A.5).

Typically, the initial grid is refined based on doping information and/or the potential and electric field of the initial bias. However, as shown in Section A.5.1, the grid dependence of local potential is not the same as the grid dependence of the terminal currents and other electrical quantities we may really be interested in. Ideally, we would like to relate the local grid to the expected error in the electrical quantities (such as currents, capacitance) we are interested in directly, but a priori error estimate of that nature is a difficult task.

In this appendix, instead of trying to come up with a new error estimate, we have simply conducted a series of simulations for devices of different doping profiles, dimensions, using different physical models, and calculated the error of various simulated quantities compared to those obtained through simulation on a fine mesh (when the simulated value has stabilized). The simulation structure used was shown in Section A.2. Parameters h_{x3} and h_{y1} are varied, and the grid densities changes

Device	Lgate (um)	Tox(nm)	Abrutpness (nm/dec)	Nsub	Xj,ext (um)
a	0.18	1.8	5	1.5e18	0.040
b	0.18	1.8	1.8	1.5e18	0.040
c	0.18	1.0	1.8	5.0e18	0.021
d	0.07	1.0	1.8	5.0e18	0.021

Table A.3: Details of the Devices in Figures A.17 to A.30

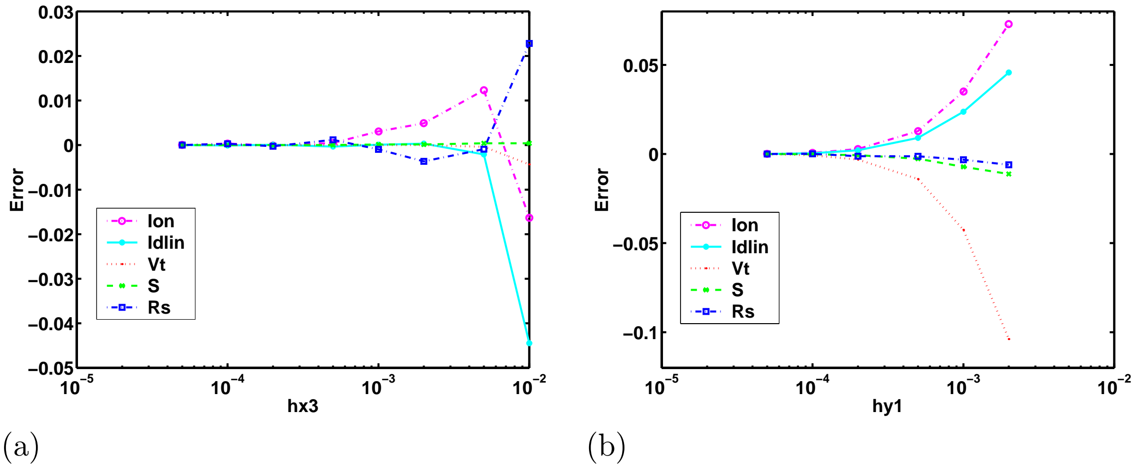


Figure A.17: Error in the simulation of various electrical quantities with CONMOB for device a using grids with various (a) h_{x3} and (b) h_{y1}

geometrically as specified in Appendix A.8. Four devices, relevant for the 2001 and 2008 technology node on the ITRS [1], are considered as shown in Table A.3. The results is summarized in Figures A.17 through A.30.

The user can refer to these error plots to obtain the grid densities needed to achieve a given accuracy in the electrical quantities of interest. This can serve as the starting point for further simulations and TCAD-based computational studies.

Given the importance of grid on TCAD simulations, and the strong dependence of grid sensitivities on the model choice and model implementation, error versus grid plots for a given model under various conditions (as in Figures A.17 through A.30)

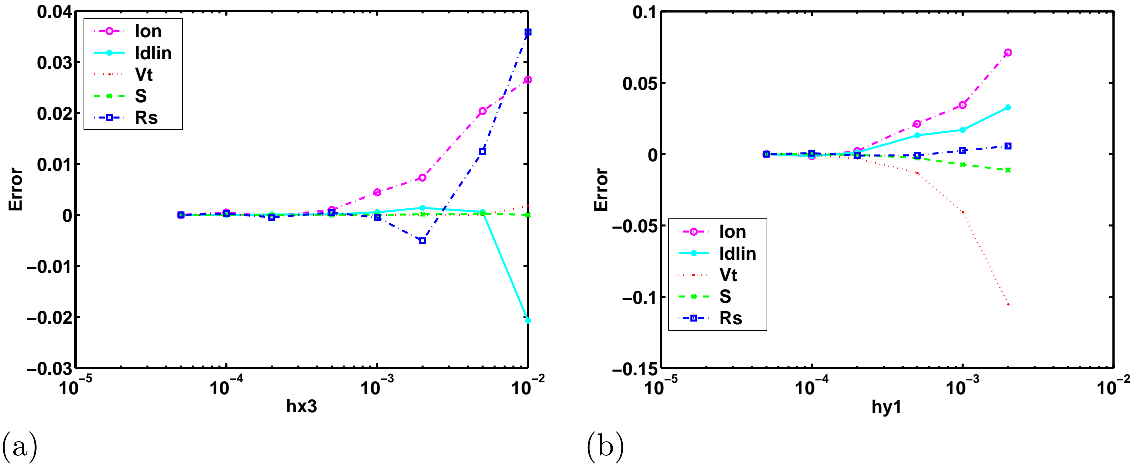


Figure A.18: Error in the simulation of various electrical quantities with CONMOB and UNIMOB for device a using grids with various (a) h_{x3} and (b) h_{y1}

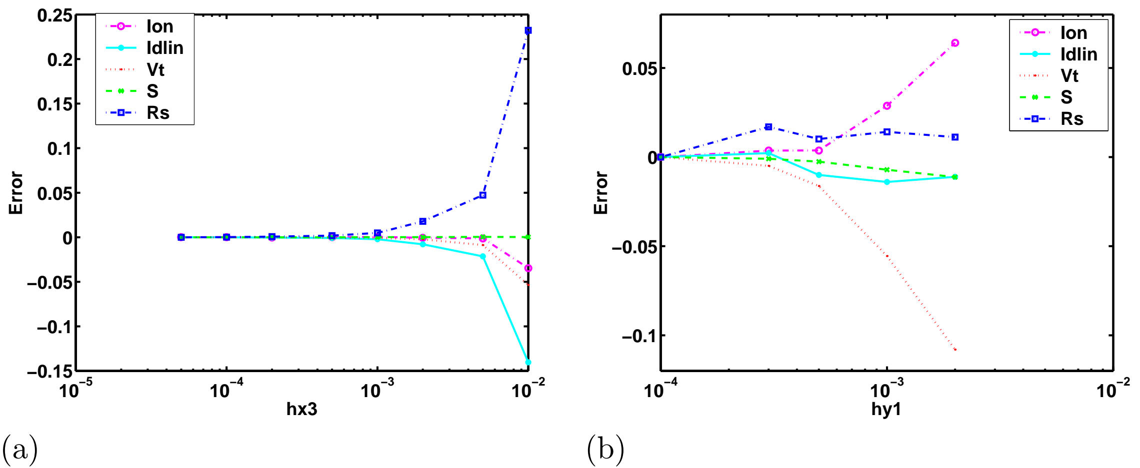
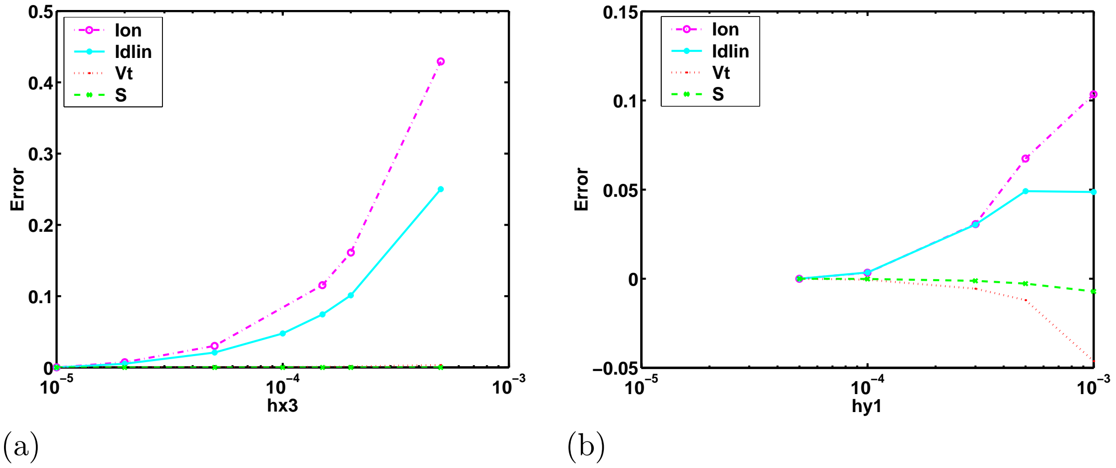
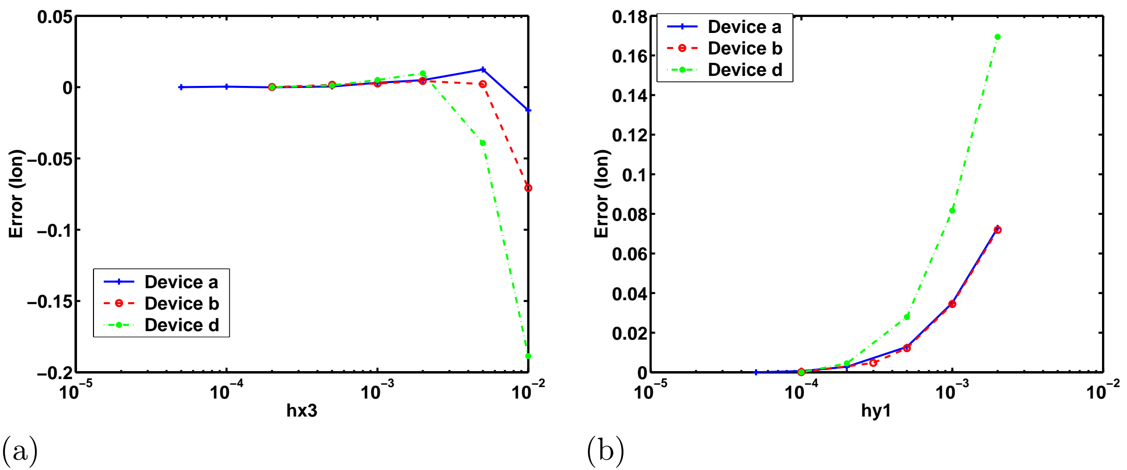


Figure A.19: Error in the simulation of various electrical quantities with GMCMOB and FLDMOB for device a using grids with various (a) h_{x3} and (b) h_{y1}



(a) (b)

Figure A.20: Error in the simulation of various electrical quantities with GMCMOB, FLDMOB and Van Dort for device a using grids with various (a) h_{x3} and (b) h_{y1}



(a) (b)

Figure A.21: Error in the simulation of Ion with CONMOB using grids with various (a) h_{x3} and (b) h_{y1}

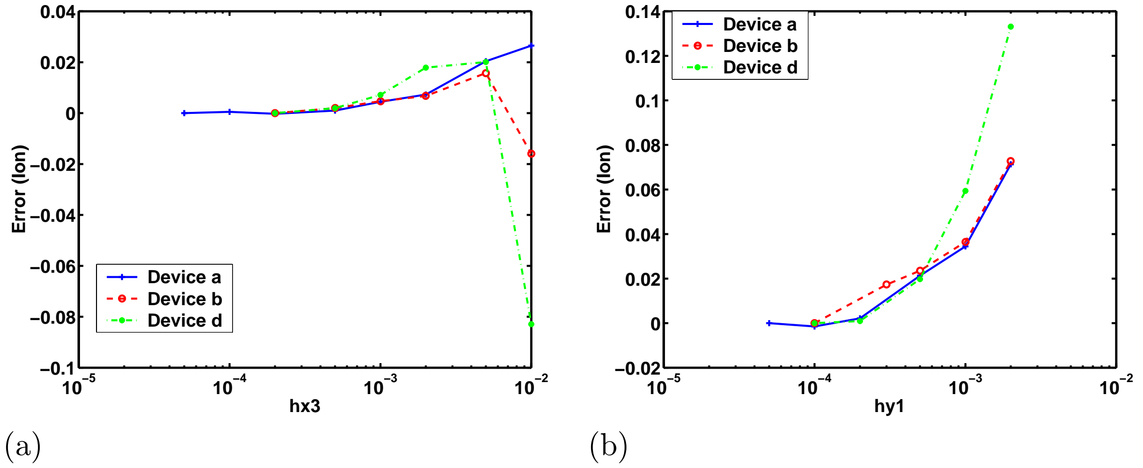


Figure A.22: Error in the simulation of Ion with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1}

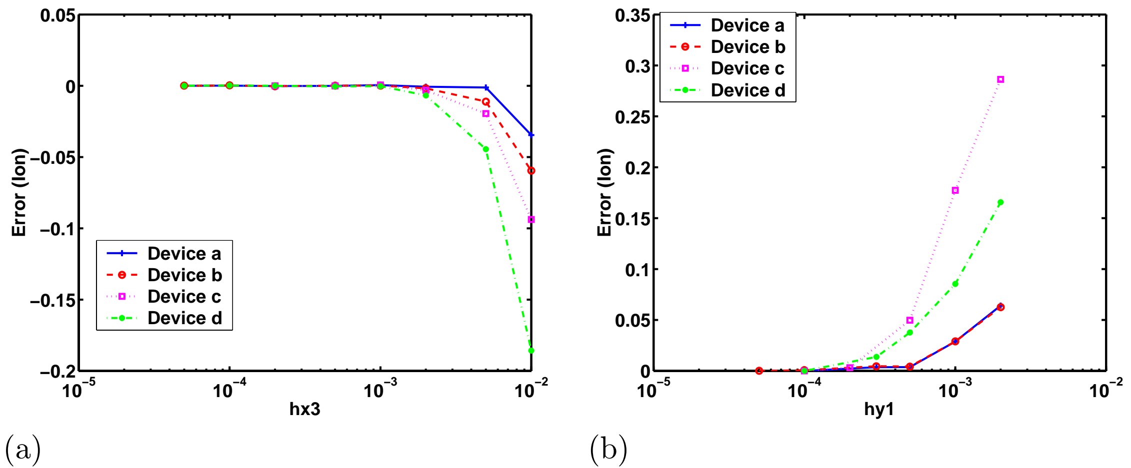


Figure A.23: Error in the simulation of Ion with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1}

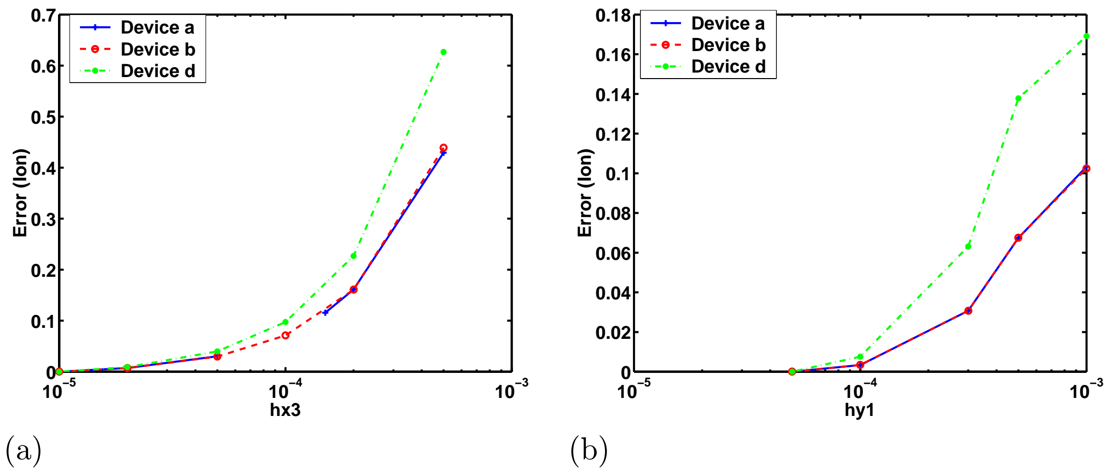


Figure A.24: Error in the simulation of Ion with GMCMOB, FLDMOB and Van Dort using grids with various (a) h_{x3} and (b) h_{y1}

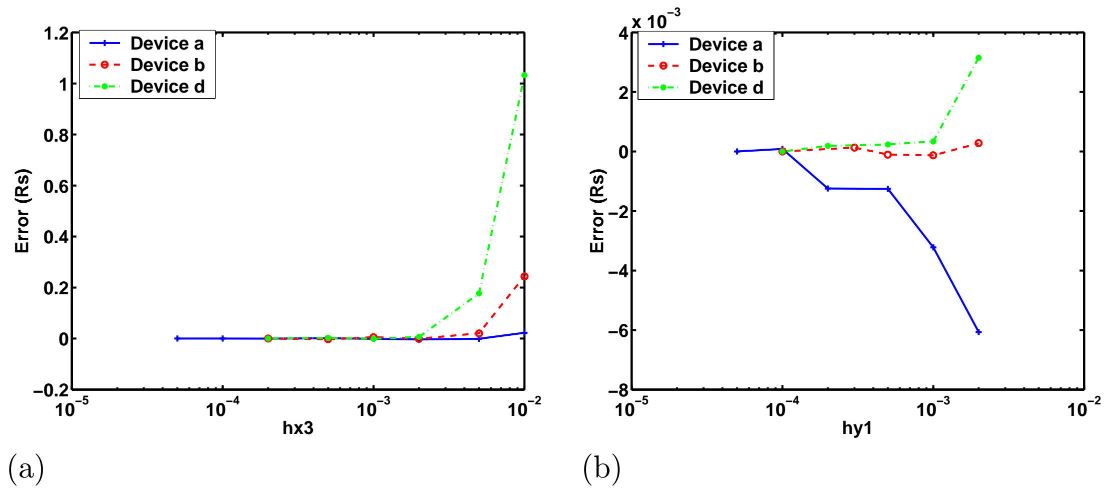


Figure A.25: Error in the simulation of Rs with CONMOB using grids with various (a) h_{x3} and (b) h_{y1}

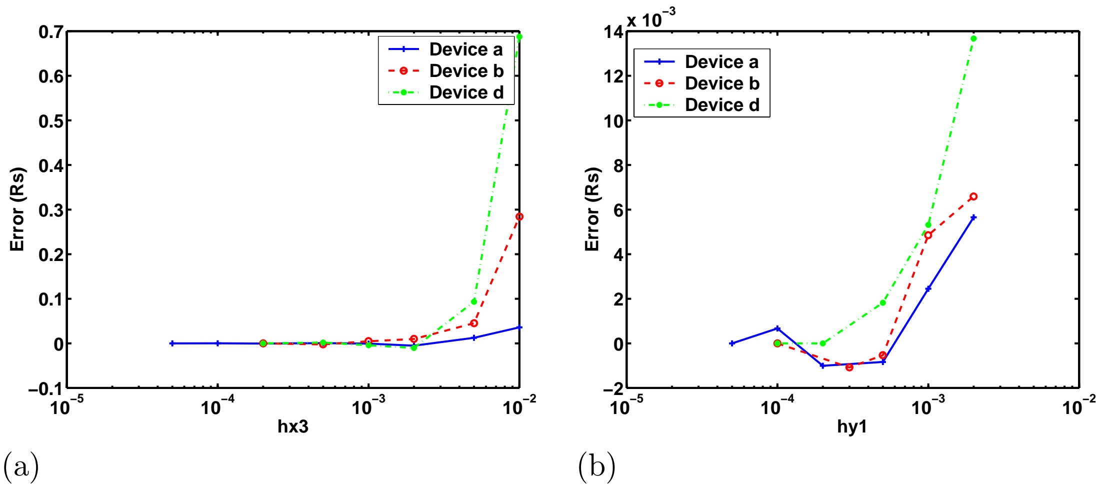


Figure A.26: Error in the simulation of Rs with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1}

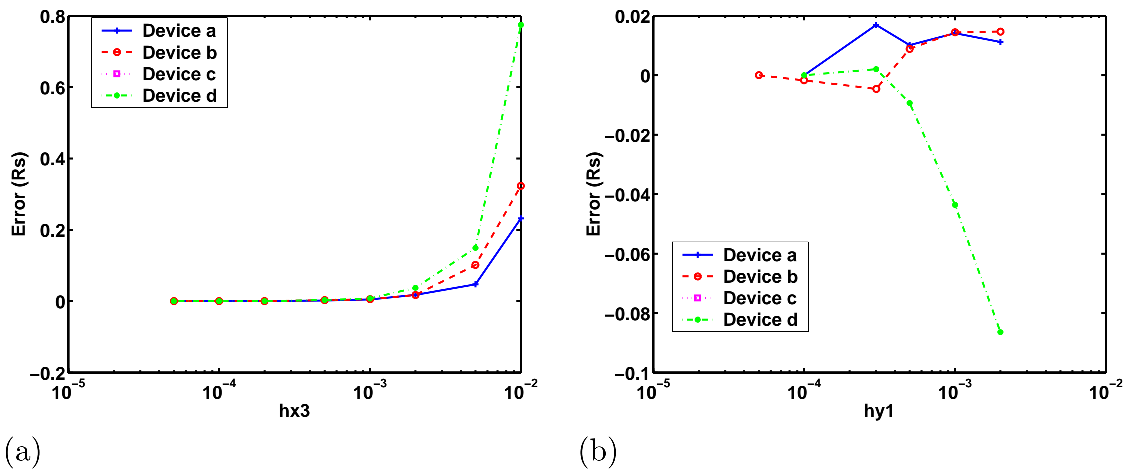


Figure A.27: Error in the simulation of Rs with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1}

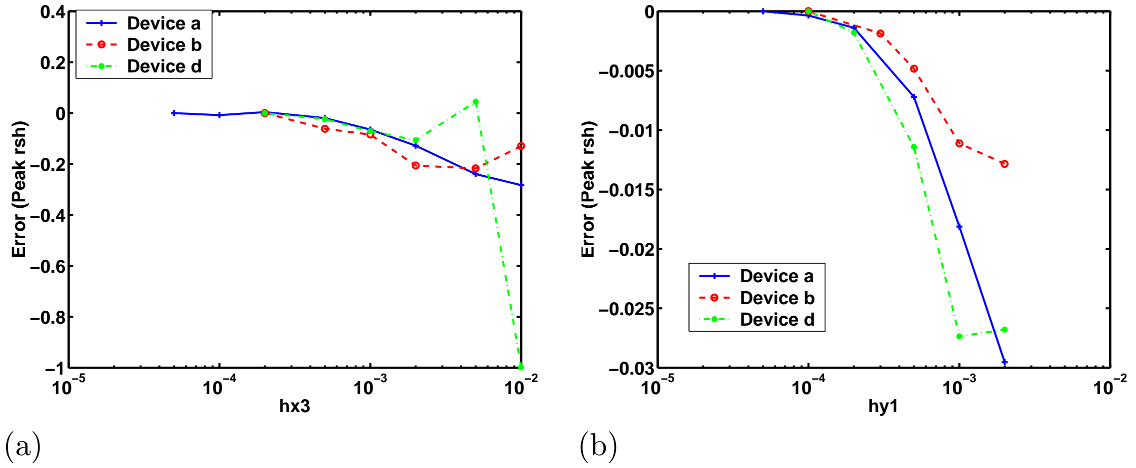


Figure A.28: Error in the simulation of height of Rsh spike with CONMOB using grids with various (a) h_{x3} and (b) h_{y1}

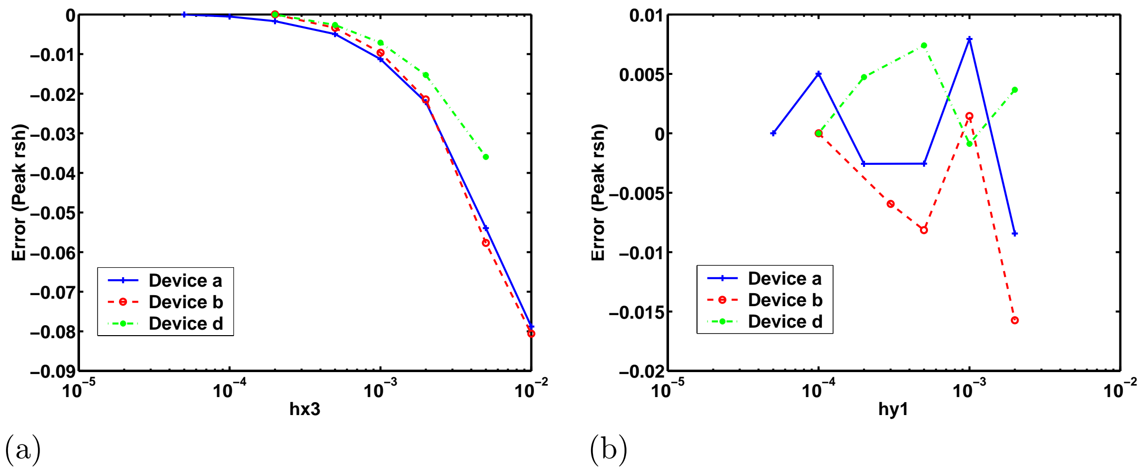


Figure A.29: Error in the simulation of height of Rsh spike with CONMOB and UNIMOB using grids with various (a) h_{x3} and (b) h_{y1}

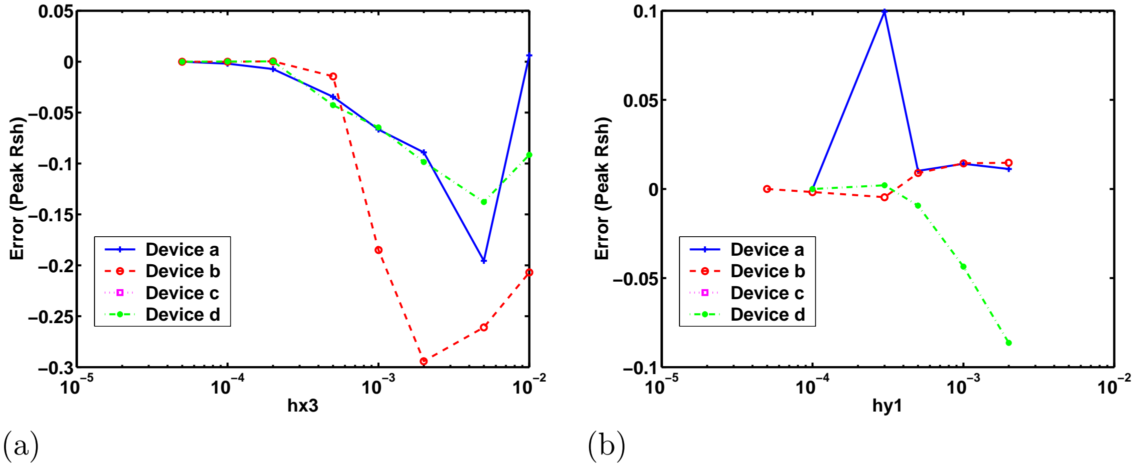


Figure A.30: Error in the simulation of height of Rsh spike with GMCMOB and FLDMOB using grids with various (a) h_{x3} and (b) h_{y1}

would be useful for the user both in choosing the model (and understanding the trade-off between model complexity, accuracy and approximate simulation time) and in establishing a starting grid for future explorations. As such, the grid sensitivity of any model implementation should be considered an intrinsic and important property of the implementation and should be provided in the appropriate place in the documentation.

A.7 Conclusion

The need for abrupt source/drain extension in deep sub-micron devices, and the importance of polydepletion and quantization effects all lead to more stringent grid requirements. Sufficient grid density is needed to resolve the doping at the abrupt junction, and to handle the non-linearities inherent in the advanced physical models needed in simulating deep sub-micron devices.

This appendix quantified these requirements through detailed simulation of devices targeted at the 2001 and 2008 technology node on the ITRS. Error plots for various electrical quantities obtained through device simulation using various model choice and grid densities, for devices with various doping slopes and oxide thickness are provided as a starting point for further simulations.

It was also shown that the grid requirements depend critically on the quantities of interest. The “optimal” grid is only meaningful in the context of simulating of a particular set of electrical quantities. As such, any grid guidelines and gridding methodology must be followed and chosen with care to ensure proper modeling of a modern deep sub-micron device.

A.8 Useful Formulae for Generating Grids

In order to minimize local truncation error, it is desirable to have a smooth transition in the grid spacing. A simple way of achieving that in a tensor product mesh is to make sure that the size of adjacent cells differ by no more than a constant factor, resulting in a geometric sequence in grid spacing.

Consider the section of a grid depicted in Figure A.31. We know from the geometric series formula

$$h_B = h_A \cdot r^{n-1} \tag{A.9}$$

$$L = h_A \frac{r^n - 1}{r - 1} \tag{A.10}$$

where r is the ratio between the adjacent grid spaces. From these relationships, we obtain

$$\boxed{h_B = \frac{h_A + (r-1) \cdot L}{r}} \tag{A.11}$$

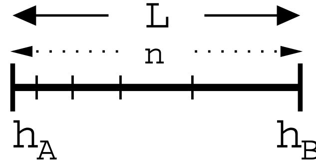


Figure A.31: Grid section of length L subdivided into n spaces with grid spacing varying smoothly from h_A to h_B

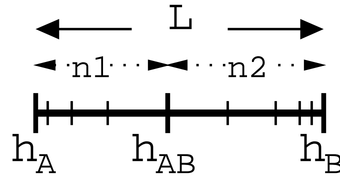


Figure A.32: Grid section of length L with grid spacing varying smoothly from h_A to h_{AB} in n1 steps then to h_B in n2 steps

$$\boxed{n = \frac{\log(\frac{L}{h_A}(r-1)+1)}{\log(r)}} \tag{A.12}$$

These equations are useful when we need to enforce a dense grid spacing at one end of the grid section, while the other end should be "sparse".

Similarly, for the grid section depicted in Figure A.32, we have

$$\boxed{h_{AB} = \frac{h_A+h_B+(r-1)\cdot L}{2r}} \tag{A.13}$$

This is useful when we want the grid section to have a dense grid at the two ends and a sparse grid in the middle.

Appendix B

Device Structure Template

B.1 Description of Template Files

The template code can be found at http://www-tcad.stanford.edu/∼yiupun

ParameterObjects.py Methods for retrieving the device and simulation parameters used to generate the simulation scripts. Parameters not explicitly specified will be calculated from other device parameters where possible.

DeviceStructTemplate.py Uses device simulators to generate the device structure file for further simulations.

ContactAttributesTemplate.py Generates simulation statements for the electrical contact.

IVSimulationTemplate.py Utility methods used for generating simulation statements for various IV ramping scenarios, as well as for post-processing the simulation results for extracting current, quasi-Fermi level and resistance information.

TIFfromTS4MED.py Uses process simulators to generate the device structure file for further simulations. Could be more realistic that devices generated from DeviceStructTemplate.py.

struct_params.py Collection of device parameters. Customized for each device to be simulated.

B.2 Partial Code Listings

Listing B.1: DeviceStructTemplate.py

```
# Generate Medici file for creating device structure for further
# simulations.

import os.path
import math
import FilePreprocessor

class DeviceStructTemplate(FilePreprocessor.FileTemplate):
  # METHOD __init__
  # PARAMETERS:
  #   simCmdFileName - name of the simulation command deck to
  #     generate
  #   device - a DeviceStructParameters object
  #   grid - a GridParameters object
  #   tifOutFilename - name of the TIF file for storing the device
  #     structure
  def __init__(self, simCmdFileName, device, grid,
               tifOutFilename = 'struct.tif'):
    FilePreprocessor.FileTemplate.__init__(self, simCmdFileName)
    self.tifOutFile = tifOutFilename
    self.device = device
    self.grid = grid
    self.summary = '␣summary'

  def writeCmdFile(self):
    self.p('$␣----------------------------------------------------')
```

```
    self.p('$␣␣␣␣␣␣␣␣␣MEDICI␣FILE␣FOR␣creating␣n-MOSFET␣structure')
    self.p('$␣----------------------------------------------------')
    self.p('')
    self.p('MESH␣␣␣␣␣␣out.file="' + self.tifOutFile
           + '"␣␣␣␣␣␣␣␣␣␣␣tif')
    self.p('')
    self.writeXMeshCmds()
    self.p('')
    self.writeYMeshCmds()
    self.p('')
    self.writeEliminateCmds()
    self.p('')
    self.writeRegionCmds()
    self.p('')
    self.writeElectrodeCmds()
    self.p('')
    self.writeDopingCmds()
    self.p('')
    self.p('$␣****************************************************')
    self.p('CALL␣␣␣␣␣␣file=contact.med')

  def writeXMeshCmds(self):
    self.resetMeshCmds()
    self.p('$␣*****␣X.MESH␣*****')
    self.p('$␣source␣contact')
    self.writeMeshCmd(self.device.getXMin(),
                      self.device.getSourceContactCenterX(), [],
                      self.grid.getBdyXDensity(),
                      self.grid.getContactCenterXDensity(), [],
                      'x')
    self.writeMeshCmd([], self.device.getSourceContactGapX(), [],
                      self.grid.getContactCenterXDensity(),
                      self.grid.getContactGapXDensity(), [], 'x')

    # No Gap if getDistanceFromCont2Spacer() <= 0.0
    if (self.device.getDistanceFromCont2Spacer() > 0.0):
      self.p('$␣-␣gap␣between␣contact␣and␣spacer␣on␣source␣side')
      self.writeMeshCmd([], self.device.getSourceGapSpacerX(),
                        self.grid.getGapInteriorL2(),
                        self.grid.getContactGapXDensity(),
```

```
                    self.grid.getGapSpacerXDensity(),
                    self.grid.getGapInteriorXDensity(), 'x')

    self.p('$␣Spacer␣and␣Source␣Extension')
    if self.device.hasOverlap():
       self.p('$␣-␣Spacer␣on␣source␣side')
       self.writeMeshCmd([], self.device.getSourceSpacerGateX(),
                    self.grid.getSpacerInteriorL2(),
                    self.grid.getGapSpacerXDensity(),
                    self.grid.getSpacerGateXDensity(),
                    self.grid.getSpacerInteriorXDensity(),
                    'x')
    else:
       self.p('$␣-␣Spacer␣on␣source␣side')
       self.writeMeshCmd([], self.device.getSourceMetJuncX(),
                    self.grid.getSpacerInteriorL2(),
                    self.grid.getGapSpacerXDensity(),
                    self.grid.getMetJuncXDensity(),
                    self.grid.getSpacerInteriorXDensity(),
                    'x')
       self.p('$␣-␣gate/source␣extension')
       self.writeMeshCmd([], self.device.getSourceSpacerGateX(),
                    [], self.grid.getMetJuncXDensity(),
                    self.grid.getSpacerGateXDensity(), [],
                    'x')
    self.p('$␣gate␣section')
    if self.device.hasOverlap():
       self.p('$␣-␣gate/source␣extension␣overlap')
       self.writeMeshCmd([], self.device.getSourceMetJuncX(), [],
                    self.grid.getSpacerGateXDensity(),
                    self.grid.getMetJuncXDensity(), [], 'x')
       self.p('$␣-␣channel')
       self.writeMeshCmd([], self.device.getChannelCenterX(), [],
                    self.grid.getMetJuncXDensity(),
                    self.grid.getChannelCenterXDensity(), [],
                    'x')
    else:
       self.p('$␣-␣channel')
       self.writeMeshCmd([], self.device.getChannelCenterX(), [],
                    self.grid.getSpacerGateXDensity(),
```

```
                    self.grid.getChannelCenterXDensity(), [],
                    'x')
if self.device.isHalfStructure():
   return
# The following will only run if device is a full transistor
if self.device.hasOverlap():
   self.writeMeshCmd([], self.device.getDrainMetJuncX(), [],
                    self.grid.getChannelCenterXDensity(),
                    self.grid.getMetJuncXDensity(), [], 'x')
   self.p('$␣-␣gate/drain␣extension␣overlap')
   self.writeMeshCmd([], self.device.getDrainSpacerGateX(), [],
                    self.grid.getMetJuncXDensity(),
                    self.grid.getSpacerGateXDensity(), [],
                    'x')
else:
   self.writeMeshCmd([], self.device.getDrainSpacerGateX(), [],
                    self.grid.getChannelCenterXDensity(),
                    self.grid.getSpacerGateXDensity(), [],
                    'x')
self.p('$␣Spacer␣and␣Drain␣Extension')
if self.device.hasOverlap():
   self.p('$␣-␣Spacer␣on␣Drain␣Side')
   self.writeMeshCmd([], self.device.getDrainGapSpacerX(),
                    self.grid.getSpacerInteriorL1(),
                    self.grid.getSpacerGateXDensity(),
                    self.grid.getGapSpacerXDensity(),
                    self.grid.getSpacerInteriorXDensity(),
                    'x')
else: # Underlap
   self.p('$␣-␣gate/drain␣extension')
   self.writeMeshCmd([], self.device.getDrainMetJuncX(), [],
                    self.grid.getSpacerGateXDensity(),
                    self.grid.getMetJuncXDensity(), [], 'x')
   self.p('$␣-␣Spacer␣on␣Drain␣Side')
   self.writeMeshCmd([], self.device.getDrainGapSpacerX(),
                    self.grid.getSpacerInteriorL1(),
                    self.grid.getMetJuncXDensity(),
                    self.grid.getGapSpacerXDensity(),
                    self.grid.getSpacerInteriorXDensity(),
                    'x')
```

```
      self.p('$␣drain␣contact')
      if (self.device.getDistanceFromCont2Spacer() > 0.0):
         self.p('$␣-␣gap␣between␣contact␣and␣spacer␣on␣drain␣side')
         self.writeMeshCmd([], self.device.getDrainContactGapX(),
                           self.grid.getGapInteriorL1(),
                           self.grid.getGapSpacerXDensity(),
                           self.grid.getContactGapXDensity(),
                           self.grid.getGapInteriorXDensity(), 'x')

      self.writeMeshCmd([], self.device.getDrainContactCenterX(), [],
                        self.grid.getContactGapXDensity(),
                        self.grid.getContactCenterXDensity(), [],
                        'x')
      self.writeMeshCmd([], self.device.getXMax(), [],
                        self.grid.getContactCenterXDensity(),
                        self.grid.getBdyXDensity(), [], 'x')

   def writeYMeshCmds(self):
      self.resetMeshCmds()
      self.p('$␣*****␣Y.MESH␣*****')
      self.p('$␣-␣poly␣gate')
      self.writeMeshCmd(self.device.getYMin(),
                        self.device.getGateOxideY(),
                        [], self.grid.getTopGateYDensity(),
                        self.grid.getGateOxideYDensity(), [], 'y')
      self.p('$␣-␣oxide')
      self.writeMeshCmd([], self.device.getOxideSubstrateY(), [],
                        self.device.getOxideThickness()/4, [], [],
                        'y')
      self.p('$␣-␣substrate')
      self.writeMeshCmd([], self.device.getConductionEdgeY(), [],
                        self.grid.getOxideSubstrateYDensity(),
                        self.grid.getCondEdgeYDensity(), [], 'y')
      self.writeMeshCmd([], self.device.getExtJuncY(), [],
                        self.grid.getCondEdgeYDensity(),
                        self.grid.getExtJuncYDensity(), [], 'y')
      self.writeMeshCmd([], self.device.getDepletionEdgeY(), [],
                        self.grid.getExtJuncYDensity(),
                        self.grid.getDepletionEdgeYDensity(), [],
```

```
                  'y')
   self.writeMeshCmd([], self.device.getYMax(), [],
                  self.grid.getDepletionEdgeYDensity(),
                  self.device.getYMax()/5, [], 'y')

# METHOD resetMeshCmds
# DESCRIPTION: should be called before all X.MESH statements,
#   then again before all Y.MESH statements
def resetMeshCmds(self):
   self.gridSectionMin = []

# METHOD writeMeshCmd
# DESCRIPTION: generate a mesh statement with the specified
# PARAMETERS:
#   pmin - the minimum coordinate of the grid section;
#     if missing, minimum coordinate in entire structure is
#     assumed
#   pmax - the maximum coordinate of the grid section;
#     if missing, maximum coordinate in entire structure is
#     assumed
#   h1 - grid density on the minimum side
#   h2 - grid density on the maximum side;
#     if missing, constant grid density is assumed
#   h3 - grid density in the interior (optional)
#   xOrY - 'x' or 'y' indicating the type of mesh statement
#     to use
def writeMeshCmd(self, pmin, pmax, interior, h1, h2, h3, xOrY):
   if (((h3 == []) and (interior != []))
       or ((h3 != []) and (interior == []))):
      raise ValueError('h1␣and␣h3␣must␣be␣specified␣together')
   if (xOrY == 'x'):
      meshCmd = 'X.MESH'
      minTerm = '␣x.min='
      maxTerm = '␣x.max='
   else:
      meshCmd = 'Y.MESH'
      minTerm = '␣y.min='
      maxTerm = '␣y.max='
   cmdLine = meshCmd
   if (pmin != []):
```

```
    cmdLine = cmdLine + minTerm + str(pmin)
elif (self.gridSectionMin != []):
    pmin = self.gridSectionMin
else:
    pmin = self.device.getXMin()
if (h3 == []):
    if (pmax != []):
        cmdLine = cmdLine + maxTerm + str(pmax)
        self.gridSectionMin = pmax
    else:
        pmax = self.device.getXMax()
if (h2 == []):
    # Only h1 specified
    # Constant grid density in the grid section
    self.p(cmdLine + '␣min.spac=' + str(h1/10))
    self.p('+␣h1=' + str(h1) + self.summary)
elif (h1 + h2 < pmax - pmin) and (h3 == []):
    # h1 and h2 specified
    # Grid that varies geometrically from one end to the next
    self.p(cmdLine + '␣min.spac=' + str(min(h1, h2)/10))
    self.p('+␣h1=' + str(h1) + '␣h2=' + str(h2) + self.summary)
elif (h1 + h2 < pmax - pmin):
    # h1, h2, and h3 specified
    # Grid that varies geometrically from one end to the next
    # self.p(cmdLine + ' min.spac=' + str(min(h1, h2, h3)/10))
    # self.p('+ h1=' + str(h1) + ' h2=' + str(h2)
    # + ' h3=' + str(h3) + self.summary)
    # Medici bug, can't use h3
    self.p(cmdLine + maxTerm + str(pmin+interior)
          + '␣min.spac=' + str(min(h1, h3)/10))
    self.p('+␣h1=' + str(h1) + '␣h2=' + str(h3) + self.summary)
    cmdLine = meshCmd
    if (pmax != []):
        cmdLine = cmdLine + maxTerm + str(pmax)
    self.p(cmdLine + '␣min.spac=' + str(min(h2, h3)/10))
    self.p('+␣h1=' + str(h3) + '␣h2=' + str(h2) + self.summary)
else:
    self.p(cmdLine + '␣n.space=1␣min.spac='
          + str((pmax-pmin)/10))
```

```
def writeEliminateCmds(self):
   if self.grid.hasEliminate():
      # Eliminate y grid lines in Gate
      self.p('$␣Eliminate␣y␣grid␣lines␣in␣Gate')
      y = self.grid.getGateEliminateStartY()
      for i in range(self.grid.getNumGateEliminateY()):
         if (y < self.device.getYMin()):
            break
         yDensity = self.grid.getGridDensityY(y)
         (x1, x2) = self.findEliminateXRange(
            self.device.getSourceMetJuncX(), yDensity)
         (x3, x4) = self.findEliminateXRange(
            self.device.getDrainMetJuncX(), yDensity)
         # Clip eliminate region a little away from gate edge
         if ((x1 != [])
             and (x1 < self.device.getSourceSpacerGateX()
                  + 2 * self.grid.getSpacerGateXDensity())):
            x1 = (self.device.getSourceSpacerGateX()
                  + 2* self.grid.getSpacerGateXDensity())
         if ((x4 != [])
             and (x4 > self.device.getDrainSpacerGateX()
                  - 2 * self.grid.getSpacerGateXDensity())):
            x4 = (self.device.getDrainSpacerGateX()
                  - 2* self.grid.getSpacerGateXDensity())
         if ((x1 == []) or (x2 == []) or (x3 == []) or (x4 == [])
             or (x1 >= x2) or (x3 >= x4)):
            print 'Invalid␣eliminate␣statement␣at␣y␣=', y
         elif (x2 >= x3):
            self.p('ELIMINATE␣column␣x.min=' + str(x1)
                   + '␣x.max=' + str(x4) + '␣y.max=' + str(y))
         else:
            self.p('ELIMINATE␣column␣x.min=' + str(x1)
                   + '␣x.max=' + str(x2) + '␣y.max=' + str(y))
            self.p('ELIMINATE␣column␣x.min=' + str(x3)
                   + '␣x.max=' + str(x4) + '␣y.max=' + str(y))
         # NOTE: Arbitrary factor of 3 is used as increment
         y = y - 3 * self.grid.getGridDensityY(y)
      # Eliminate y grid lines in Substate
      self.p('$␣Eliminate␣y␣grid␣lines␣in␣Substrate')
      y = self.grid.getSubstrateEliminateStartY()
```

```
        for i in range(self.grid.getNumSubstrateEliminateY()):
            if (y > self.device.getYMax()):
                break
            yDensity = self.grid.getGridDensityY(y)
            (x1, x2) = self.findEliminateXRange(
                self.device.getSourceMetJuncX(), yDensity)
            (x3, x4) = self.findEliminateXRange(
                self.device.getDrainMetJuncX(), yDensity)
            if (x1 == []) or (x2 == []) or (x3 == []) or (x4 == []):
                print 'Invalid␣eliminate␣statement␣at␣', y
            elif (x2 >= x3):
                self.p('ELIMINATE␣column␣x.min=' + str(x1)
                          + '␣x.max=' + str(x4) + '␣y.min=' + str(y))
            else:
                self.p('ELIMINATE␣column␣x.min=' + str(x1)
                          + '␣x.max=' + str(x2) + '␣y.min=' + str(y))
                self.p('ELIMINATE␣column␣x.min=' + str(x3)
                          + '␣x.max=' + str(x4) + '␣y.min=' + str(y))
            # NOTE: Arbitrary factor of 3 is used as increment
            y = y + 3 * self.grid.getGridDensityY(y)
        # Eliminate y in Oxide
        self.p('$␣Eliminate␣grid␣lines␣in␣Oxide')
        self.p('ELIMINATE␣column␣x.max='
                  + str(self.device.getSourceGapSpacerX())
                  + '␣y.max='
                  + str(self.device.getGateOxideY()
                          - 3 * self.grid.getGateOxideYDensity()))
        self.p('ELIMINATE␣row␣x.max='
                  + str(self.device.getSourceGapSpacerX())
                  + '␣y.max='
                  + str(self.device.getGateOxideY()
                          - 3 * self.grid.getGateOxideYDensity()))
        self.p('ELIMINATE␣column␣x.min='
                  + str(self.device.getDrainGapSpacerX())
                  + '␣y.max=' + str(self.device.getGateOxideY()))
        self.p('ELIMINATE␣row␣x.min='
                  + str(self.device.getDrainGapSpacerX())
                  + '␣y.max=' + str(self.device.getGateOxideY()))

# METHOD: findEliminateXRange
```

```
# RETURNS: (x1, x2) indicating the range of x coordinates to
#   eliminate
def findEliminateXRange(self, x, ydensity):
   factor = 1
   x1 = x
   x2 = x
   xleft = x1
   xright = x2
   while ((xleft >= self.device.getXMin())
          and (self.grid.getGridDensityX(xleft)
               < factor * ydensity)):
      x1 = xleft
      # NOTE: Arbitrary factor of 3 is used as increment
      xleft = xleft - 3 * self.grid.getGridDensityX(xleft)
   while ((xright <= self.device.getXMax())
          and (self.grid.getGridDensityX(xright)
               < factor * ydensity)):
      x2 = xright
      # NOTE: Arbitrary factor of 3 is used as increment
      xright = xright + 3 * self.grid.getGridDensityX(xright)
   if (x1 < x) and (x2 > x):
      return (x1, x2)
   else:
      return ([], [])

def writeRegionCmds(self):
   self.p('$␣*****␣Region␣Definition*****')
   self.p('$␣-␣gate')
   self.writeRegionCmd('polygate', 'polysili',
                       self.device.getSourceSpacerGateX(),
                       self.device.getDrainSpacerGateX(),
                       self.device.getYMin(),
                       self.device.getGateOxideY())
   self.p('$␣-␣LTO')
   if (self.device.getLTOThickness() > 0.0):
      self.writeRegionCmd('lto', 'oxide', self.device.getXMin(),
                          self.device.getSourceSpacerGate(),
                          self.device.getGateOxideY()
                          -self.device.getLTOThickness(),
                          self.device.getGateOxideY())
```

```
        self.writeRegionCmd('lto', 'oxide',
                            self.device.getSourceSpacerGateX()
                            -self.device.getLTOThickness(),
                            self.device.getSourceSpacerGateX(),
                            self.device.getYMin(),
                            self.device.getGateOxideY())
        self.writeRegionCmd('lto', 'oxide',
                            self.device.getDrainSpacerGateX(),
                            self.device.getXMax(),
                            self.device.getGateOxideY()
                            -self.device.getLTOThickness(),
                            self.device.getGateOxideY())
        self.writeRegionCmd('lto', 'oxide',
                            self.device.getDrainSpacerGateX(),
                            self.device.getDrainSpacerGateX()
                            +self.device.getLTOThickness(),
                            self.device.getYMin(),
                            self.device.getGateOxideY())
    self.writeRegionCmd('lto', 'oxide', self.device.getXMin(),
                        self.device.getSourceGapSpacerX(),
                        self.device.getYMin(),
                        self.device.getGateOxideY()
                        -self.device.getLTOThickness())
    self.writeRegionCmd('lto', 'oxide',
                        self.device.getDrainGapSpacerX(),
                        self.device.getXMax(),
                        self.device.getYMin(),
                        self.device.getGateOxideY()
                        -self.device.getLTOThickness())
    self.p('$␣-␣spacer')
    self.writeRegionCmd(self.device.getSpacerMaterial(),
                        self.device.getSpacerMaterial(),
                        self.device.getSourceGapSpacerX(),
                        self.device.getSourceSpacerGateX()
                        -self.device.getLTOThickness(),
                        self.device.getYMin(),
                        self.device.getGateOxideY()
                        -self.device.getLTOThickness())
    self.writeRegionCmd(self.device.getSpacerMaterial(),
                        self.device.getSpacerMaterial(),
```

```
                      self.device.getDrainSpacerGateX()
                      +self.device.getLTOThickness(),
                      self.device.getDrainGapSpacerX(),
                      self.device.getYMin(),
                      self.device.getGateOxideY()
                      -self.device.getLTOThickness())
   self.p('$␣-␣gate␣oxide')
   self.writeRegionCmd('gateox', 'oxide', [], [],
                      self.device.getGateOxideY(),
                      self.device.getOxideSubstrateY())
   self.p('$␣-␣substrate')
   self.writeRegionCmd('silicon', 'silicon', [], [],
                      self.device.getOxideSubstrateY(),
                      self.device.getYMax())

# METHOD writeRegionCmd
# DESCRIPTION: generate region command
# PARAMETERS:
#   name, material - name and material type of the region
#   xmin, xmax, ymin, ymax - rectangular area defining the region;
#     if missing, the respective boundary of the device is assumed
# NOTE: if xmin and xmax are greater than channelCenterX,
#   and this is a half structure, region will be skipped;
#   if xmax alone is greater than channelCenterX, will be clipped
#   to channelCenterX
def writeRegionCmd(self, name, material, xmin, xmax, ymin, ymax):
   if (self.device.isHalfStructure() and
       (xmin != []) and (xmin > self.getChannelCenterX())):
      return
   cmdLine = 'REGION␣name=' + name + '␣' + material
   if (xmin != []):
      cmdLine = cmdLine + '␣x.min=' + str(xmin)
   if (xmax != []):
      if ((self.device.isHalfStructure())
          and (xmax > self.device.getChannelCenterX())):
         # Clip xmax
         cmdLine = (cmdLine + '␣x.max='
                    + str(self.device.getChannelCenterX()))
      else:
         cmdLine = cmdLine + '␣x.max=' + str(xmax)
```

```
  if (ymin != []):
    cmdLine = cmdLine + '␣y.min=' + str(ymin)
  if (ymax != []):
    cmdLine = cmdLine + '␣y.max=' + str(ymax)
  self.p(cmdLine)

def writeElectrodeCmds(self):
  self.p('$␣*****␣Electrodes␣*****')
  self.writeElectrodeCmd('source',
                [], self.device.getSourceContactGapX(),
                self.device.getOxideSubstrateY(),
                self.device.getOxideSubstrateY())
  self.writeElectrodeCmd('drain',
                self.device.getDrainContactGapX(), [],
                self.device.getOxideSubstrateY(),
                self.device.getOxideSubstrateY())
  if self.device.hasPolyGate():
    self.writeElectrodeCmd(
     'gate', self.device.getSourceSpacerGateX(),
     self.device.getDrainSpacerGateX(),
     self.device.getYMin(), self.device.getYMin())
  elif self.device.hasMetalGate():
    self.writeElectrodeCmd(
     'gate', self.device.getSourceSpacerGateX(),
     self.device.getDrainSpacerGateX(),
     self.device.getYMin(), self.device.getGateOxideY())
  self.p('ELECTRODE␣name=back␣bottom')

# METHOD writeElectrodeCmd
# DESCRIPTION: generate electrode command
# PARAMETERS:
#  name - name and material type of the electrode
#  xmin, xmax, ymin, ymax - rectangular area defining the
#    electrode; if missing, the respective boundary of the
#    device is assumed
# NOTE: if xmin and xmax are greater than channelCenterX,
#  and this is a half structure, region will be skipped;
#  if xmax alone is greater than channelCenterX, will be
#  clipped to channelCenterX
def writeElectrodeCmd(self, name, xmin, xmax, ymin, ymax):
```

```
    if (self.device.isHalfStructure() and
         (xmin != []) and (xmin > self.getChannelCenterX())):
        return
    cmdLine = 'ELECTRODE␣name=' + name
    if (xmin != []):
        cmdLine = cmdLine + '␣x.min=' + str(xmin)
    if (xmax != []):
        if ((self.device.isHalfStructure())
             and (xmax > self.device.getChannelCenterX())):
            # Clip xmax
            cmdLine = (cmdLine + '␣x.max='
                       + str(self.device.getChannelCenterX()))
        else:
            cmdLine = cmdLine + '␣x.max=' + str(xmax)
    if (ymin != []):
        cmdLine = cmdLine + '␣y.min=' + str(ymin)
    if (ymax != []):
        cmdLine = cmdLine + '␣y.max=' + str(ymax)
    self.p(cmdLine)

def writeDopingCmds(self):
    self.p('$␣*****␣Doping␣Profiles␣*****')
    self.writePolyDopingCmds()
    self.p('')
    self.writeChanDopingCmds()
    self.p('')
    self.writeHaloDopingCmds()
    self.p('')
    self.writeSDExtDopingCmds()
    self.p('')
    self.writeSDContDopingCmds()

# METHOD writePolyDopingCmds
# DESCRIPTION: generate the profile commands responsible for
#   the poly doping
def writePolyDopingCmds(self):
    if self.device.hasPolyGate():
        self.p('$␣-␣poly␣doping␣for␣' + self.device.getDeviceType()
               + '␣gate')
        if (self.device.isNMOS()):
```

```
            self.p('PROFILE␣region=polygate␣n.type␣uniform␣conc='
                   + str(self.device.getPolyDoping()))
         elif (self.device.isPMOS()):
            self.p('PROFILE␣region=polygate␣p.type␣uniform␣conc='
                   + str(self.device.getPolyDoping()))

   # METHOD writeChanDopingCmds
   # DESCRIPTION: generate the profile commands responsible for
   #   the channel doping
   def writeChanDopingCmds(self):
      self.p('$␣-␣channel/substrate␣doping')
      if self.device.isNMOS():
         self.p('PROFILE␣region=silicon␣p.type␣uniform␣conc='
                + str(self.device.getSubstrateDoping()))
      elif (self.device.isPMOS()):
         self.p('PROFILE␣region=silicon␣n.type␣uniform␣conc='
                + str(self.device.getSubstrateDoping()))

   # METHOD writeHaloDopingCmds
   # DESCRIPTION: generate profile commands responsible for
   #   the halo doping
   def writeHaloDopingCmds(self):
      if self.device.hasHalo():
         self.p('$␣halo')
         if self.device.isNMOS():
            dopantType = 'p.type'
         elif self.device.isPMOS():
            dopantType = 'n.type'
         self.p('PROFILE␣region=silicon␣' + dopantType + '␣n.peak='
                + str(self.device.getHaloPeak()))
         self.p('+␣x.min=' + str(self.device.getXMin())
                + '␣x.max=' + str(self.device.getSourceHaloX())
                + '␣y.min=' + str(self.device.getHaloY())
                + '␣y.max=' + str(self.device.getHaloY()))
         self.p('+␣y.char=' + str(self.device.getHaloSigmaY())
                + '␣x.char=' + str(self.device.getHaloSigmaX()))
         if not(self.device.isHalfStructure()):
            self.p('PROFILE␣region=silicon␣' + dopantType + '␣n.peak='
                   + str(self.device.getHaloPeak()))
            self.p('+␣x.min=' + str(self.device.getDrainHaloX())
```

```
                + '␣x.max=' + str(self.device.getXMax())
                + '␣y.min=' + str(self.device.getHaloY())
                + '␣y.max=' + str(self.device.getHaloY()))
         self.p('+␣y.char=' + str(self.device.getHaloSigmaY())
                + '␣x.char=' + str(self.device.getHaloSigmaX()))

   # METHOD writeSDExtDopingCmds
   # DESCRIPTION: generate the profile commands responsible for
   #   the doping for the source/drain extension region
   def writeSDExtDopingCmds(self):
      self.p('$␣-␣source/drain␣extension')
      if self.device.isNMOS():
         dopantType = 'n'
      elif self.device.isPMOS():
         dopantType = 'p'
      if (self.device.getExtDopingDefnMethod() == 'file'):
         self.p('PROFILE␣region=silicon␣' + dopantType
                + '.col=2␣in.file='
                + self.device.getExtDopingFile())
         self.p('+␣1d.ascii␣xy.rat=' + str(self.device.getExtLatD())
                + '␣x.min=' + str(self.device.getXMin())
                + '␣x.max='
                + str(self.device.getSourceSpacerGateX()
                    - self.device.getOverlapSpacerLength()))
         if not(self.device.isHalfStructure()):
            self.p('PROFILE␣region=silicon␣' + dopantType
                   + '.col=2␣in.file='
                   + self.device.getExtDopingFile())
            self.p('+␣1d.ascii␣xy.rat='
                   + str(self.device.getExtLatD())
                   + '␣x.min='
                   + str(self.device.getDrainSpacerGateX()
                       + self.device.getOverlapSpacerLength())
                   + '␣x.max=' + str(self.device.getXMax()))
      elif (self.device.getExtDopingDefnMethod() == 'gauss'):
         self.p('PROFILE␣region=silicon␣x.min='
                + str(self.device.getXMin())
                + '␣x.max='
                + str(self.device.getSourceSpacerGateX()
                    - self.device.getOverlapSpacerLength()
```

```
                + self.device.getExtPeakXDelta()))
        self.p('+␣y.min=' + str(self.device.getExtPeakYMin())
              + '␣y.max=' + str(self.device.getExtPeakYMax())
              + '␣y.char=' + str(self.device.getExtGaussSigmaY())
              + '␣x.char=' + str(self.device.getExtGaussSigmaX()))
        self.p('+␣' + dopantType + '.type␣n.peak='
              + str(self.device.getExtGaussPeak()))
        if not(self.device.isHalfStructure()):
           self.p('PROFILE␣region=silicon␣x.min='
                 + str(self.device.getDrainSpacerGateX()
                      + self.device.getOverlapSpacerLength()
                      - self.device.getExtPeakXDelta())
                 + '␣x.max=' + str(self.device.getXMax()))
           self.p('+␣y.min=' + str(self.device.getExtPeakYMin())
                 + '␣y.max=' + str(self.device.getExtPeakYMax())
                 + '␣y.char='
                 + str(self.device.getExtGaussSigmaY())
                 + '␣x.char='
                 + str(self.device.getExtGaussSigmaX()))
           self.p('+␣' + dopantType + '.type␣n.peak='
                 + str(self.device.getExtGaussPeak()))

  # METHOD writeSDContDopingCmds
  # DESCRIPTION: generate the profile commands responsible for
  #   the doping in the deep source/drain region
  def writeSDContDopingCmds(self):
     if self.device.hasContactDoping():
        self.p('$␣S/D␣Contact')
        if self.device.isNMOS():
           dopantType = 'n'
        elif self.device.isPMOS():
           dopantType = 'p'
        self.p('PROFILE␣region=silicon␣' + dopantType
              + '.type␣␣␣␣␣␣␣␣n.peak='
              + str(self.device.getContactPeak()))
        self.p('+␣x.min=' + str(self.device.getXMin()) + '␣x.max='
              + str(self.device.getSourceGapSpacerX())
              + '␣y.min=' + str(self.device.getContactY())
              + '␣y.max=' + str(self.device.getContactY()))
        self.p('+␣xy.rat=' + str(self.device.getContactLatD())
```

```
            + '␣y.junct='
            + str(self.device.getContactJunctionDepth()))
        if not(self.device.isHalfStructure()):
            self.p('PROFILE␣region=silicon␣' + dopantType
                + '.type␣n.peak='
                + str(self.device.getContactPeak()))
            self.p('+␣x.min=' + str(self.device.getDrainGapSpacerX())
                + '␣x.max=' + str(self.device.getXMax())
                + '␣y.min=' + str(self.device.getContactY())
                + '␣y.max=' + str(self.device.getContactY()))
            self.p('+␣xy.rat=' + str(self.device.getContactLatD())
                + '␣y.junct='
                + str(self.device.getContactJunctionDepth()))
```

Listing B.2: ParameterObjects.py

```
# Abstract classes for accessing Structure and Simulation Parameters

import types
import math

......

# CLASS DeviceStructParameters
# DESCRIPTION: parameters governing the structure to be simulated
# USER PARAMETERS:
# halfStruct = 1 if half structure o be simulated (e.g. CV); 0 else
# deviceType = "nmos" or "pmos"
# gateType = "poly" or "metal"
# (if poly gate) polyDoping = doping for the polysilicon region
# substrateDoping = background doping in the silicon substrate
# haloPeak = peak doping of halo, or 0.0 to turn off halo
# (if halo) haloSigmaY = characteristic length in the y direction
#           haloSigmaX = characteristic length in the x direction
#           haloY = y coordinate of center of gaussian
#           haloOffset = offset of halo from source/drain extension
# extDopingDefn = how extension doping will be defined "file" or
#           "gauss" extLatD = lateral characteristic length/vertical
#           one
# (if file) extDopingFile = name of file containing doping
# (if gauss)extGaussPeak = peak doping of extension gaussian
```

```
#   (either)extGaussSigmaX = characteristic length in the x direction
#   (or)    extGaussSigmaY = characteristic length in the y direction
#   (either)extPeakXDelta = extra length of the "peak doping" region
#            beyond overlap spacer edge
#   (or)    extPeakYDelta = depth of the "peak doping" region in
#            extension
# contPeak = peak doping of contact gaussian; 0.0 to turn off
# (if cont) contLatD = lateral characteristic length/vertical one
#           contY = center of contact gaussian
#           contJuncDepth = contact junction depth
# contResistivity = contact resistivity
# ltoThickness = thickness of low temperature oxide (0.0 if none)
# gateThickness = thickness of gate region
# tox = thickness of dielectric/oxide
# (either) lmet = metallurgical channel length
#       or lgate = gate length
# (2 of)  self.SDLength = total distance from edge of device to gate
#         self.distFromCont2Spacer = distance from contact to spacer
#         self.contL = length of contact
# contactCenterXOffset = offset from the edge of device of the "grid
#   center" of the s/d contact
# spacerMaterial = "oxide" or "nitride"
# (2 of)  self.totalSpacerLength = total length of spacer
#         self.basicSpacerLength = basic spacer
#         self.overlapSpacerLength = overlap spacer (affects overlap
#           length)
# conductionThickness = expected thickness of conduction layer.
#   allows better control of grid.
# extJuncDepth = controls where the extjunc grid is
#   NOTE: makes sure this is consistent with doping
# depletionWidth = expected width of the depletion region for the
#   S/D allows better control of grid
# ymax = y coordinate of the edge of device
class AbstractDeviceStructParameters:
    # METHOD: isHalfStructure
    # RETURNS: returns 1 if half structure is desired (e.g. for
    #   CV sim)
    def isHalfStructure(self):
        return self.halfStruct
```

```
......

# METHOD getGateLength
# RETURNS: the gate length of the device
def getGateLength(self):
    if (self.lgate != []):
        return getPositiveNumber(self.lgate, "lgate")
    elif (self.lmet != []):
        return (getPositiveNumber(self.lmet, "lmet")
                + 2 * self.getExtOverlapLength())
    else:
        raise AttributeError(
            "At␣least␣one␣of␣lmet␣or␣lgate␣must␣be␣specified")

# METHOD getMetallurgicalLength
# RETURNS: the metallurgical channel length of the device
def getMetallurgicalLength(self):
    if (self.lmet != []):
        return getPositiveNumber(self.lmet, "lmet")
    elif (self.lgate != []):
        return (getPositiveNumber(self.lgate, "lgate")
                - 2*self.getExtOverlapLength())
    else:
        raise AttributeError(
            "At␣least␣one␣of␣lmet␣or␣lgate␣must␣be␣specified")

# METHOD getExtOverlapLength
# RETURNS: the overlap length between the source extension and
#   the gate
# NOTE: could be negative
def getExtOverlapLength(self):
    if (self.lmet == []) or (self.lgate == []):
        return (self.getExtJuncDisplacement()
                - self.getOverlapSpacerLength())
    else:
        return (self.lgate - self.lmet) / 2.0

......

# METHOD getBasicSpacerLength
```

```
    # RETURNS: the basic spacer for controlling the location
    # NOTE: totalSpacerLength = basicSpacerLength
    #                         + overlapSpacerLength
    def getBasicSpacerLength(self):
        if ((self.totalSpacerLength == [])
            or (self.overlapSpacerLength == [])):
            retVal = getPositiveNumber(self.basicSpacerLength,
                                       "basicSpacerLength")
        elif (self.basicSpacerLength == []):
            retVal = (getPositiveNumber(self.totalSpacerLength,
                                        "totalSpacerLength")
                      - self.overlapSpacerLength)
        else:
            if not(isPositiveNumber(self.totalSpacerLength) and
                   isPositiveNumber(self.basicSpacerLength)):
                raise AttributeError(
                    "The␣spacer␣lengths␣must␣be␣numbers␣>␣0.0")
            elif not(equals(self.basicSpacerLength
                           +self.overlapSpacerLength,
                           self.totalSpacerLength)):
                raise AttributeError("inconsistent␣spacer␣lengths")
            else:
                retVal = self.basicSpacerLength
        if (retVal <= 0.0):
            raise AttributeError("basicSpacerLength␣must␣be␣>␣0.0")
        return retVal

    # METHOD getTotalSpacerLength
    # RETURNS: the total spacer length
    # NOTE: totalSpacerLength = basicSpacerLength
    #   + overlapSpacerLength
    def getTotalSpacerLength(self):
        if (self.totalSpacerLength == []):
            retVal = (getPositiveNumber(self.basicSpacerLength,
                                        "basicSpacerLength")
                      + self.overlapSpacerLength)
        elif ((self.basicSpacerLength == [])
              or (self.overlapSpacerLength == [])):
            retVal = getPositiveNumber(self.totalSpacerLength,
                                       "totalSpacerLength")
```

```
    else:
        if not(isPositiveNumber(self.totalSpacerLength) and
               isPositiveNumber(self.basicSpacerLength)):
            raise AttributeError(
              "totalSpacerLength␣and␣basicSpacerLength"
              + "␣must␣be␣>␣0.0")
        elif not(equals(self.basicSpacerLength
                        + self.overlapSpacerLength,
                        self.totalSpacerLength)):
            raise AttributeError("inconsistent␣spacer␣lengths")
        else:
            retVal = self.totalSpacerLength
    if (retVal <= 0.0):
        raise AttributeError("totalSpacerLength␣must␣be␣>␣0.0")
    return retVal

# METHOD getOverlapSpacerLength
# RETURNS: the overlap spacer length. this allows control of
#   the overlap.
# NOTE: totalSpacerLength = basicSpacerLength
#                           + overlapSpacerLength
def getOverlapSpacerLength(self):
    if ((self.basicSpacerLength == [])
        or (self.totalSpacerLength == [])):
        retVal = self.overlapSpacerLength
    elif ((self.overlapSpacerLength == [])):
        retVal = (getPositiveNumber(self.totalSpacerLength,
                                   "totalSpacerLength")
                  - getPositiveNumber(self.basicSpacerLength,
                                     "basicSpacerLength"))
    else:
        if not(isPositiveNumber(self.totalSpacerLength) and
               isPositiveNumber(self.basicSpacerLength)):
            raise AttributeError(
              "totalSpacerLength␣and␣basicSpacerLength␣must␣be"
              + "␣numbers␣>␣0.0")
        elif not(equals(self.basicSpacerLength
                        + self.overlapSpacerLength,
                        self.totalSpacerLength)):
            raise AttributeError("inconsistent␣spacer␣lengths")
```

```
        else:
            retVal = self.overlapSpacerLength
    return retVal

# totalSDLength = contactL + distFromCont2Spacer
#                 + TotalSpacerLength

# METHOD getTotalSDLength
# RETURNS: total length of the source/drain region
#   (from edge of device to gate edge)
def getTotalSDLength(self):
    if ((self.distFromCont2Spacer == []) or (self.contL == [])):
        return getPositiveNumber(self.SDLength, "SDLength")
    elif (self.SDLength == []):
        return (self.getTotalSpacerLength() +
               getPositiveNumberOrZero(self.distFromCont2Spacer,
                                    "distFromCont2Spacer") +
               getPositiveNumber(self.contL, "contL"))
    else:
        if not(isPositiveNumber(self.SDLength) and
              isPositiveNumber(self.contL) and
              isPositiveNumberOrZero(self.distFromCont2Spacer)):
            raise AttributeError(
              "SDLength␣and␣contL␣must␣be␣numbers␣>␣0.0;"
              + "␣distFromCont2Spacer␣must␣be␣a␣number␣>=␣0.0")
        elif not(equals(self.SDLength,
                      self.contL + self.distFromCont2Spacer
                      + self.getTotalSpacerLength())):
            raise AttributeError(
              "Inconsistent␣contact/spacer␣lengths")
        else:
            return self.SDLength

......

# METHOD getContactGapXOffset
# RETURNS: the x coordinate of the boundary between contact
#   and the gap between contact and spacer (relative to channel
#   center)
def getContactGapXOffset(self):
```

```
        return (self.getGateLength()/2.0 + self.getTotalSpacerLength()
                + self.getDistanceFromCont2Spacer())

    # METHOD getGapSpacerXOffset
    # RETURNS: the x coordinate of the boundary between
    #   the gap between contact and spacer and the spacer
    #   (relative to channel center)
    def getGapSpacerXOffset(self):
        return (self.getGateLength()/2.0
                + self.getTotalSpacerLength())

    # METHOD getSpacerGateXOffset
    # RETURNS: the x coordinate of the boundary between spacer and
    #   gate (relative to channel center)
    def getSpacerGateXOffset(self):
        return self.getGateLength()/2.0

    ......

    # METHOD getXMin
    # RETURNS: minimum x coordinate of the device
    def getXMin(self):
        return (- self.getGateLength()/2 - self.getTotalSDLength()
                + self.getChannelCenterX())

    # METHOD getSourceContactCenterX
    # RETURNS: x coodinate of the "grid center" of the source contact
    def getSourceContactCenterX(self):
        return (self.getXMin() + self.getContactCenterXOffset())

    ......

    # METHOD getYMin
    # RETURNS: minimum y coordinate of the device
    def getYMin(self):
        return (self.getOxideSubstrateY() - self.getOxideThickness()
                - self.getGateThickness())

    # METHOD getGateToxY
    # RETURNS: the y coorindate of the boundary between the Gate and
```

```
    #  the dielectric
    def getGateOxideY(self):
        return self.getOxideSubstrateY() - self.getOxideThickness()

    ......

# FUNCTION getGridDensityH2
# RETURNS: grid density of one side of the grid section
#   given the grid density at the other end
# PARAMETERS:
#   density0 - given grid density
#   ratio - desired geometric ratio
#   sectionL - size of the grid section
def getGridDensityH2(density0, ratio, sectionSize, tolerance = 1e-5):
    if not(isPositiveNumber(density0) and isPositiveNumber(ratio)
           and isPositiveNumber(sectionSize)):
        raise ValueError(
          "density0,␣ratio␣and␣sectionL␣must␣be␣numbers␣>␣0.0")
    elif (sectionSize > (1+ratio)*density0):
        # 2 grid spaces with size density0 and ratio*density0
        # could fit in section
        return (density0 + (ratio-1)*sectionSize)/ratio
    elif (sectionSize > (1+1/ratio)*density0):
        # 2 grid spaces with size density0 and density0/ratio
        # could fit in section (only releveant if ratio > 1.0).
        # Fit 2 grid space in section
        return (1-tolerance)*(sectionSize-density0)
    else:
        # Fit 1 grid space in section
        return sectionSize

# FUNCTION getGridDensityH3
# RETURNS: grid density at the interior of a grid section
#   given the grid density at both ends
# PARAMETERS:
#   H1, H2 - grid density at the 2 ends of the grid section
#   r1, r2 - desired geometric ratio to and from the interior point
#   L - size of the grid section
def getGridDensityH3(h1, h2, r1, r2, L, tolerance = 1e-5):
    if not(isPositiveNumber(h1) and isPositiveNumber(h2)
```

```
          and isPositiveNumber(r1) and isPositiveNumber(r2)
          and isPositiveNumber(L)):
        raise ValueError("h1,␣h2,␣r1,␣r2␣and␣L␣must␣be␣numbers␣>␣0.0")
    else:
        retVal = ((L + h1 / (r1 - 1) - h2 * r2 / (r2 - 1))
                  / (r1 / (r1 - 1) - 1 / (r2 - 1)))
        if (not(isPositiveNumber(retVal))):
            raise ValueError("h3␣must␣be␣>␣0.0!")
        return retVal

# CLASS GridParameters
# DESCRIPTION: parameters governing the grid density for simulation
#   of device
# USER PARAMETERS:
# (required) metJuncXDensity = grid density at the extension
#              junction
# (either) channelCenterXDensity = grid density at center of channel
#          spacerGateXDensity = grid density at bdy of spacer and
#            gate
#          spacerInteriorXDensity = grid density at the interior of
#            spacer region
#          gapSpacerXDensity = grid density at bdy of gap and spacer
#          gapInteriorXDensity = grid density at the interior of gap
#            region
#          contactGapXDensity = grid density at bdy of contact and
#            gap
#          bdyXDensity = grid density at the edge of the device
#       or ratioChannel = transition from extension junc to center
#            of channel
#          ratioSG2MJ = transition from extension junc to gate edge
#          ratioSpacer1 = transition from gate edge to spacer center
#          ratioSpacer2 = transition from spacer center to spacer
#            edge
#          ratioCG2GS1 = transition from spacer edge to gap center
#          ratioCG2GS2 = transition from gap center to contact
#          ratioBdy2CG = transition from contact to boundary
# (required) topYDensity = grid density at the oxide/silicon
#            interface
# (either) condEdgeYDensity = grid density at the edge of conduction
#          extJuncYDensity = grid density at the extension junction
```

```
#          deplEdgeYDensity = grid density at the depletion edge
#          botYDensity = grid density at bottom of device
#       or ratioOS2CE = transition from surface to edge of
#            conduction
#          ratioCE2EJ = transition from channel edge to extension
#            junction
#          ratioEJ2DE = transition from extension junction to
#            depletion edge
#          ratioDE2Bot = transition from depletion edge to bottom
#            of device
class AbstractGridParameters:
    def __init__(self, device):
        self.device = device
        self.initDefaultParameters()

    # METHOD initDefaultParameters
    # DESCRIPTION: contains user parameters
    def initDefaultParameters(self):
        raise AttributeError('Unimplemented␣Error')

    # FUNCTION getChannelCenterXDensity
    # RETURNS: x grid density at the center of the channel
    # NOTE: can be specified by user, or will be calculated
    #   automatically
    def getChannelCenterXDensity(self):
        if (self.channelCenterXDensity == []):
            if (self.device.hasUnderlap()):
                # Gate Edge lies between Metallurgical junction and
                # Channel Center
                return getGridDensityH2(
                  self.getSpacerGateXDensity(),
                  getPositiveNumber(self.ratioChannel,
                                    "ratioChannel"),
                  self.device.getGateLength()/2.0)
            else:
                # Metallurgical junction lies between (in the
                # inclusive sense) Gate Edge and Channel Center
                return getGridDensityH2(
                  self.getMetJuncXDensity(),
                  getPositiveNumber(self.ratioChannel,
```

```
                                "ratioChannel"),
                self.device.getMetallurgicalLength()
                  /2.0)
    else:
        return getPositiveNumber(self.channelCenterXDensity,
                                 "channelCenterXDensity")

......

# METHOD getTopGateYDensity(self)
# RETURNS: y grid density at the top of the poly gate
def getTopGateYDensity(self):
    return self.device.getGateThickness()/3

# METHOD getGateOxideYDensity(self)
# RETURNS: y grid density at the poly gate/oxide interface
def getGateOxideYDensity(self):
    return self.device.getOxideThickness()/4

......

# METHOD logInterpolate
# RETURNS: interpolated value at the indicated point y
# PARAMETERS
#   val0, val1 - values at point 0 and point 1 of segment
#   deltay - y - y0
#   totaly - y1 - y0
def logInterpolate(self, val0, val1, deltay, toty):
    return math.exp(math.log(val0)
                  + (math.log(val1) - math.log(val0))
                    * deltay / toty)

# METHOD findGeometricRatio
# RETURNS: ratio for a geometric series that sum to l and
#   scale from a to b
# PARAMETER:
#   l - sum of the geometric series
#   a, b - starting and ending term
def findGeometricRatio(self, l, a, b):
    return (l - a) / (l - b)
```

```
# METHOD findGeometricTerm
# RETURNS: the value of the term when the partial sum of
#   the geometric series is y
# PARAMETER:
#   y - partial sum of the geometric series
#   l - total sum of the geometric series
#   a, b - starting and ending term
def findGeometricTerm(self, y, l, a, b):
    r = self.findGeometricRatio(l, a, b)
    return (y * (r-1) + a) / (math.sqrt(r))

......
```

Bibliography

[1] International technology roadmap for semiconductors 1999 edition front end processes final draft, October 1999.

[2] A. Abramo, A. Cardin, L. Selmi, and E. Sangiorgi. Two-dimensional quantum mechanical simulation of charge distribution in silicon MOSFETs. *IEEE Transactions of Electron Devices*, 47(10):1858–63, October 2000.

[3] A. Agarwal, D. J. Eaglesham, H.-J. Gossmann, L. Pelaz, S. B. Herner, D. C. Jacobson, T. E. Haynes, Y. Erokhin, and R. Simonton. Boron-enhanced-diffusion of boron: the limiting factofr for ultra-shallow junctions. In *International Electron Devices Meeting. Technical Digest*, pages 467–70, San Francisco, CA, USA, December 1997.

[4] K. Ahmed, I. De, C. Osburn, J. Wortman, and J. Hauser. Limitations of the modified shift-and-ratio technique for extraction of the bias dependence of l_{eff} and r_{sd} of LDD MOSFET's. *IEEE Transactions of Electron Devices*, 47(4):891–3, April 2000.

[5] M. G. Ancona, Z. Yu, R. W. Dutton, P. J. Vande Voorde, M. Cao, and D. Vook. Density-gradient analysis of tunneling in MOS structures with ultra-thin oxides. In *Simulation of Semiconductor Processes and Devices, 1999*, pages 235–8,

Tokyo, Japan, September 1999.

[6] D. A. Antoniadis, I. J. Djomehri, K. M. Jackson, and S. Miller. Well-tempered bulk-Si NMOSFET device home page. http://www-mtl.mit.edu/Well/, Aug 1999.

[7] Avant! Corporation. *Medici. Two-Dimensional Device Simulation Program. User's Manual*, July 2000.

[8] V. Axelrad, A. Al-Bayati, B. Adibi, and P. Carey. A simulation study of cmos performance improvement by laser annealed source/drain extension profiles. In *Conference on Ion Implantation Technology, 2000.*, pages 239–42, September 2000.

[9] M. Bellis. The IBM PC - history. http://inventors.about.com/library/weekly/aa031599.htm.

[10] S. Biesemans, M. Hendriks, S. Kubicek, and K. De Meyer. Accurate determination of channel length, series resistance and junction doping profile for MOSFET optimisation in deep submicron technologies. In *1996 Symposium on VLSI Technology. Digest of Technical Papers*, pages 166–7, Honolulu, HI, USA, June 1996.

[11] S. Biesemans, M. Hendriks, S. Kubicek, and K. De Meyer. Practical accuracy analysis of some existing effective channel length and series resistance extraction methods for MOSFET's. *IEEE Transactions on Electron Devices*, 45(6):1310–6, June 1998.

[12] G. Booch. *Object-Oriented Analysis and Design with Applications.* Cummings Pub. Co., Redwood City, CA, USA, 1994.

[13] I. Bork and W. Molzer. Appropriate initial damage conditions for "three-stream" point defect diffusion models. In *Simulation of Semiconductor Processes and Devices*, pages 175–8, Seattle, WA, USA, September 2000.

[14] K. M. Cham, S.-Y. Oh, D. Chin, and J. L. Moll. *Computer-Aided Design and VLSI Device Development*, page 100 ff. Kluwer Academic Publishers, Hingham, MA, USA, 1986.

[15] T. Chen, D. W. Yergeau, and R. W. Dutton. Efficient 3d mesh adaptation in diffusion simulation. In *Proceedings of International Conference on Simulation of Semiconductor Processes and Devices*, pages 171–2, Tokyo, September 1996.

[16] C. H. Choi, J. S. Goo, T. Y. Oh, Z. Yu, R. W. Dutton, A. Bayoumi, M. Cao, P. Vande Voorde, D. Vook, and C. H. Diaz. MOS c-v characterization of ultra-thin gate oxide thickness (1.3-1.8 nm). *IEEE Electron Device Letters*, 20(6):292–4, June 1999.

[17] C. H. Choi, J. S. Goo, Z. Yu, and R. Dutton. Shallow source/drain extension effects on external resistance in sub-0.1 μm MOSFET's. *IEEE Transactions on Electron Devices*, 47(3):655–8, March 2000.

[18] C. H. Choi, K. H. Oh, J. S. Goo, Z. Yu, and R. W. Dutton. Direct tunneling current model for circuit simulation. In *International Electron Devices Meeting, 1999. Technical Digest.*, pages 735–8, Washington, DC, USA, December 1999.

[19] C. H. Choi, Y. Wu, J. S. Goo, Z. Yu, and R. W. Dutton. Capacitance reconstruction from measured C-V in high leakage, nitride/oxide MOS. *IEEE Transactions on Electron Devices*, 47(10):1843–50, October 2000.

[20] D. Connelly and M. Foisy. Improved device technology evaluation and optimization. In *Simulation of Semiconductor Processes and Devices, 2000. SISPAD 2000. 2000 International Conference on*, pages 155–8, Seattle, WA, USA, September 2000.

[21] J. O. Coplien. *Advanced C++ - Programming Styles and Idioms*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1992.

[22] E. Crabbé, R. Logan, J. Snare, P. Agnello, and J. Sun. Anomalous short-channel effects in 0.1 μm MOSFETs. In *International Electron Devices Meeting. Technical Digest*, pages 571–4, New York, NY, USA, December 1996.

[23] M. N. Darwish, J. L. Lentz, M. R. Pinto, P. M. Zeitzoff, T. J. Krutsick, and H. H. Vuong. An improved electron and hole mobility model for general purpose device simulation. *IEEE Transactions on Electron Devices*, 44(9):1529–38, September 1997.

[24] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, sc-9(5), October 1974.

[25] H. Doi. Development of secondary ion mass spectrometer. i. development of the AES-IMA device and is application to solid surface. *Mass Spectroscopy*, (4):325–49, Dec 1977.

[26] R. W. Dutton and Z. Yu. *Technology CAD – Computer Simulation of IC Processes and Devices*. Kluwer Academic Publishers, Norwell MA, USA, 1993.

[27] S.B. Felch, D.F. Downey, E.A. Arevalo, S. Talwar, C. Gelatos, and Y. Wang. Sub-melt laser annealing followed by low-temperature RTP for minimized diffusion. In *Conference on Ion Implantation Technology, 2000.*, pages 167–70, Sep 2000.

[28] B. Fletcher. Computers and internet – pentium IV launch. http://www1.sympatico.ca/news/Specials/2000/11/21-pentium4.html, November 2000.

[29] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1995.

[30] T. Ghani, K. Mistry, P. Packan, S. Thompson, M. Stettler, S. Tyagi, and M. Bohr. Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors. In *2000 Symposium on VLSI Technology Digest of Technical Papers*, pages 174–5, 2000.

[31] K. Goto, M. Kase, Y. Momiyama, H. Kurata, T. Tanaka, M. Deura, Y. Sanbonsugi, and T. Sugii. A study of ultra shallow junction and tilted channel implantation for high performance 0.1 μm pMOSFETs. In *International Electron Devices Meeting 1998. Technical Digest*, pages 631–4, San Francisco, USA, December 1998.

[32] H. I. Hanafi, W. P. Noble, R. S. Bass, K. Varahramyan, Y. Lii, and A. J. Dally. A model for anomalous short-channel behavior in submicron MOSFET's. *IEEE Electron Device Letters*, 14(12):575–7, December 1993.

[33] W. Hansch, Th. Vogelsang, R. Kircher, and M. Orlowski. Carrier transport near the Si/SiO_2 interface of a MOSFET. *Solid-State Electronics*, 32(10):839–49, October 1989.

[34] S. E. Hansen and M. D. Deal. *SUPREM-IV.GS Two Dimesional Process Simulation for Silicon and Gallium Arsenide*. Integrated Circuits Laboratory, Stanford University, Stanford, CA, USA, 1993.

[35] S. A. Hareland, M. Manassian, W.-K. Shih, S. Jallepalli, H. Wang, G. L. Chindalore, Jr. A. F. Tasch, and C. M. Maziar. Computationally efficient models for quantization effects in MOS electron and hole accumulation layers. 45(7):1487–93, July 1998.

[36] M. Hendriks, G. Badenes, and L. Deferm. Halo doping for good performance and reliability in 0.25 μm CMOS technology. In *ESSDERC '96 - 1996 26th European Solid State Device Research Conference*, pages 515–8, Bologna, Italy, September 1996.

[37] A. Hori and B. Mizuno. CMOS device technology toward 50 nm region - performance and drain architecture. In *Electron Devices Meeting, 1999. IEDM Technical Digest. International*, pages 641–4, Washington, DC, USA, December 1999.

[38] A. Inani, R. V. Rao, B. Cheng, and J. Woo. Gate stack architecture analysis and channel engineering in deep sub-micron MOSFETs. *Japanese Journal of Applied Physics, Part 1 (Regular Papers, Short Notes and Review Papers)*, 38(4b):2266–2271, April 1999.

[39] R. Kasnavi, Y. Sun, R. Mo, P. Pianetta, P. B. Griffin, and J. D. Plummer. Characterization of arsenic dose loss at Si/SiO_2 interface. *Journal of Applied Physics*, 87(4):2255–60, February 2000.

[40] P. Keys, H.-J. Gossmann, K.K. Ng, and C.S. Rafferty. Series resistance limits for 0.05 μm MOSFETs. *Superlattices and Microstructures*, 27(2-3):125–36, 2000.

[41] N. Khalil and J. Faricelli. Inverse modeling profile determination: Implementation issues and recent results. In *Proceedings of International Conference on Simulation of Semiconductor Processes and Devices*, pages 19–22, Tokyo, Japan, September 1996.

[42] N. Khalil, J. Faricelli, D. Bell, and S. Selberherr. The extraction of two-dimensional MOS transistor doping via inverse modeling. *IEEE Electron Device Letters*, 16(1):16–19, January 1995.

[43] R. Kim, T. Aoki, T. Hirose, Y. Furuta, S. Hayashi, T. Shano, and K. Taniguchi. Modeling of arsenic transient enhanced diffusion and background boron segregation in low-energy As^+ implanted Si. In *International Electron Devices Meeting. Technical Digest*, pages 523–6, San Francisco, CA, USA, December 2000.

[44] H. Kroemer. *Quantum Mechanics for Engineering, Materials Science, and Applied Physics*. Prentice Hall, New Jersey, 1994.

[45] M. Y. Kwong, C.-H. Choi, R. Kasnavi, P. Griffin, and R. W. Dutton. Series resistance calculation for source/drain extension region with 2-d device simulation. *IEEE Transactions on Electron Devices*, 49(7):1219–26, July 2002.

[46] Z. K. Lee, M. B. McIlrath, and D. A. Antoniadis. Two-dimensional doping profile characterization of MOSFET's by inverse modeling using I-V characteristics in

the subthreshold region. *IEEE Transactions on Electron Devices*, 46(8):1640–1649, August 1999.

[47] S. B. Lippman and J. Lajoie. *C++ Primer*. Addison Wesley Longman Inc., Reading, Massachusetts, USA, 1998.

[48] C. Lombardi, S. Manzini, A. Saporito, and M. Vanzi. A physically based mobiliy model for numerical simulation of nonplanar devices. *IEEE Transactions on Computer-Aided Design*, 7(11):1164–71, November 1988.

[49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(3):163–9, July 1987.

[50] M. Lutz. *Programming Python*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996.

[51] A. Mokhberi, P. B. Griffin, and J. D. Plummer. Kinetics of boron activation. In *Simulation of Semiconductor Processes and Devices, 2000*, pages 163–6, Seattle, WA, USA, September 2000.

[52] G. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[53] S. Mudanai, G. L. Chindalore, W.-K. Shih, H. Wang, Q. Ouyang, Jr. A. F. Tasch, C. M. Maziar, and S. K. Banerjee. *IEEE Transactions on Electron Devices*, 46(8):1749–59, August 1999.

[54] S. A. Mujtaba. *Advanced Mobility Models for Design and Simulation of Deep Submicrometer MOSFETs*. PhD thesis, Stanford University, December 1995.

[55] S. A. Mujtaba, S. Takagi, and R. W. Dutton. Accurate modeling of coulombic scattering, and its impact on scaled MOSFETs. In *Symposium on VLSI Technology. Digest of Technical Papers.*, pages 99–100, Tokyo, Japan, June 1995.

[56] R. S. Muller and T. I. Kamins. *Device Electronics for Integrated Circuits*. John Wiley and Sons, Inc., New York, 1986.

[57] K. K. Ng and W. T. Lynch. Analysis of the gate-voltage-dependent series resistance of MOSFET's. *IEEE Transcations on Electron Devices*, 33(7):965–972, July 1986.

[58] M. Nishida and H. Onodera. An anomalous increase of threshold voltages with shortening the channel lengths for deeply boron-implanted n-channel MOSFETs. *IEEE Transactions on Electron Devices*, ED-28(9):1101–1103, September 1981.

[59] S. Ogura, C. F. Codella, N. Rovedo, J. F. Shepard, and J. Riseman. A half-micron MOSFET using double-implanted LDD. In *International Electron Devices Meeting. Technical Digest.*, pages 718–21, San Francisco, CA, USA, December 1982.

[60] C. M. Osburn, I. De, K. F. Yee, and A. Srivastava. Design and integration considerations for end-of-the roadmap ultrashallow junctions. *Journal of Vacuum Science & Technology B (Microelectronics and Nanometer Structures)*, 18(1):338–45, Jan-Feb 2000.

[61] M. Page-Jones. *Fundamentals of Object-Oriented Design in UML*. Dorset House Publishing, New York, New York, USA, 2000.

[62] V. Privitera, C. Spinella, G. Fortunato, and L. Mariucci. Two-dimensional delineation of ultrashallow junctions obtained by ion implantation and excimer laser annealing. *Applied Physics Letters*, 77(4):552–4, Jul 2000.

[63] C. S. Rafferty and R. K. Smith. *Solving Partial Differential Equations with the PROPHET simulator*. Lucent Technologies, December 1997. http://www-tcad.stanford.edu/ prophet/math.pdf.

[64] C. S. Rafferty, H.-H. Vuong, S. A. Eshraghi, M. D. Giles, M. R. Pinto, and S. J. Hillenius. Explanation of reverse short channel effect by defect gradients. In *International Electron Devices Meeting. Technical Digest*, pages 311–2, USA, December 1993.

[65] M. Rodder, A. Amerasekera, S. Aur, and I. C. Chen. A study of design/process dependence of 0.25μm gate length CMOS for improved performance and reliability. In *International Electron Devices Meeting. Technical Digest*, pages 71–4, San Francisco, USA, December 1994.

[66] M. Rodder, S. Hattangady, N. Yu, W. Shiau, and P. Nicollian. A 1.2V, 0.1 μm gate length CMOS technology: Design and process issues. In *International Electron Devices Meeting. Technical Digest*, pages 623–6, San Francisco, USA, December 1998.

[67] P. M. Rousseau, S. W. Crowder, P. B. Griffin, and J. D. Plummer. Arsenic deactivation enhanced diffusion and the reverse short-channel effect. *IEEE Electron Device Letters*, 18(2):42–4, February 1997.

[68] Z.H. Sahul, E.W. McKenna, and R.W Dutton. Grid evolution for oxidation simulation using a quadtree based grid generator. In *Proceedings of International Workshop on Numerical Modeling of processes and Devices for Integrated Circuits: NUPAD V*, Honolulu, HI, USA, June 1994.

[69] G. G. Shahidi, D. A. Antoniadis, and H. I. Smith. Indium channel implants for improved MOSFET behavior at the 100 nm channel length regime. *IEEE Transactions on Electron Devices*, 36(11):2605, November 1989.

[70] J. Y.-C. Sun, Y. Taur, R. H. Dennard, and S. P. Klepner. Submicrometer-channel CMOS for low-temperature operation. *IEEE Transactions on Electron Devices*, 34(1):19–27, January 1987.

[71] S. Talwar, G. Verma, and K.H. Weiner. Ultra-shallow, abrupt, and highly-activated junctions by low-energy ion implantation and laser annealing. In J. Matsuo, G. Takaoka, and I. Yamada, editors, *International Conference on Ion Implantation Technology Proceedings, 1998*, volume 2, pages 1171–4, Kyoto, Japan, Jun 1998. Piscataway, NJ, USA: IEEE.

[72] Y. Taur. MOSFET channel length: Extraction and interpretation. *IEEE Transactions on Electron Devices*, 47(1):160–170, January 2000.

[73] Y. Taur, D. A. Buchanan, W. Chen, D. J. Frank, K. E. Ismail, S.-H. Lo, G. A. Sai-Halasz, R. G. Viswanathan, H.-J. C. Wann, S. J. Wind, and H.-S. Wong. CMOS scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, April 1997.

[74] Y. Taur and T. H. Ning. *Fundamentals of Modern VLSI Devices.* Cambridge University Press, 1998.

[75] Y. Taur, C. H. Wann, and D. J. Frank. 25 nm CMOS design considerations. In *International Electron Devices Meeting 1998. Technical Digest*, pages 789–792, Piscataway, NJ, USA, December 1998.

[76] Y. Taur, D. S. Zicherman, D. R. Lombardi, P. J. Restle, C. H. Hsu, H. I Hanafi, M. R. Wordeman, B. Davari, and G. G. Shahidi. A new 'shift and ratio' method for MOSFET channel-length extraction. *IEEE Electron Device Letters*, 13(5):267–269, May 1992.

[77] S. Thompson, P. Packan, T. Ghani, M. Stettler, M. Alavi, I. Post, S. Tyagi, S. Ahmed, S. Yang, and M. Bohr. Source/drain extension scaling for 0.1 μm and below channel length MOSFETs. In *1998 Symposium on VLSI Technology Digest of Technical Papers*, pages 132–133, 1998.

[78] R. R. Troutman. VLSI limitations from drain-induced barrier lowering. *IEEE Transactions on Electron Devices*, 26(4):461–9, April 1979.

[79] V. A. Ukraintsev, S. T. Walsh, S. P. Ashburn, C. F. Machala, H. Edwards, J. T. Gray, S. Joshi, D. Woodall, and M.-C. Chang. Two-dimensional dopant characterization using SIMS, SCS and TSUPREM4. In *International Electron Devices Meeting 1999. Technical Digest*, pages 349–52, Washington, DC, USA, December 1999.

[80] Jr. V. M. Agostinelli, H. Shin, and Jr. A. F. Tasch. A comprehensive model for inversion layer hole mobility for simulation of submicrometer MOSFET's. *IEEE Transactions of Electron Devices*, 38(1):151–9, January 1991.

[81] M. J. van Dort, P. H. Woerlee, and A. J. Walker. A simple model for quantisation effects in heavily-doped silicon MOSFETs at inversion conditions. *Solid-State Electronics*, 37(3):411–14, March 1994.

[82] W. Vandervorst, T. Clarysse, N. Duhayon, P. Eyben, T. Hantschel, M. Xu, T. Janssens, H. De Witte, T. Conard, J. Deleu, and G. Badenes. Ultra shallow

junction profiling. In *International Electron Devices Meeting. Technical Digest*, pages 429–32, San Francisco, CA, USA, December 2000.

[83] J. T. Watt. Improved surface mobility model in PISCES. In *Computer-Aided Design of IC Fabrication Processes*, Stanford University, August 1988.

[84] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design - A Systems Perspective*, pages 250–3. Addison-Wesley Publishing Company, 1993.

[85] P. De Wolf, R. Stephenson, S. Biesemans, Ph. Jansen, G. Badenes, K. De Meyer, and W. Vandervorst. Direct measurement of leff and channel profile in MOSFETs using 2-D carrier profiling techniques. In *International Electron Devices Meeting 1998. Technical Digest*, pages 559–62, San Francisco, CA, USA, December 1998.

[86] S. Yamaguchi and H. Goto. Inverse modeling-a promising approach to know what is made and what should be made. In *Proceedings of 1998 Asia and South Pacific Design Automation Conference*, pages 117–21, Yokohama, February 1998.

[87] L. D. Yau. A simple theory to predict the threshold voltage of short-channel IGFET's. *Solid-State Electronics*, 17(10):1059–63, October 1974.

[88] D. W. Yergeau. *A Dial-An-Operator Approach to Simulation of Impurity Diffusion in Semiconductors*. PhD thesis, Stanford University, 1999. http://www-tcad.stanford.edu/tcad/pubs/theses/yergeau.pdf.

[89] G. M. Yu, P. B. Griffin, and J. D. Plummer. Validation of two-dimensional impland and diffusion profiles using novel scanning capacitance microscope sample preparation and deconvolution techniques. In *Electron Devices Meeting, 1998. IEDM Technical Digest. International*, pages 717–20, San Francisco, CA, USA, December 1998.

[90] X. Zhou, K. Y. Lim, and D. Lim. A new "critical-current at linear-threshold" method for direct extraction of deep-submicron MOSFEF effective channel length. *IEEE Transactions on Electron Devices*, 46(7):1492–4, July 1999.

[91] X. Zhou, K. Y. Lim, and D. Lim. A simple and unambiguous definition of threshold voltage and its implication in deep-submicron MOS device modeling. *IEEE Transactions on Electron Devices*, 46(4):807–9, April 1999.