A SWITCHED-CAPACITOR ANALYSIS

METAL-OXIDE-SILICON CIRCUIT SIMULATOR

A Dissertation Presented to

The Faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology

Ohio University

In partial Fulfillment

of the Requirement for the Degree

Doctor of Philosophy

by

Ying-Wei Jan

March, 1999

THIS DISSERTATION ENTITLED

“A SWITCHED-CAPACITOR ANALYSIS METAL-OXIDE-SILICON CIRCUIT SIMULATOR”

by Ying-Wei Jan

has been approved

for the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology

Janusz A. Starzyk, Professor

Warren K. Wray, Dean
Fritz J. and Dolores H. Russ
College of Engineering and Technology

Acknowledgments

I would first like to thank School of Electrical Engineering and Computer Science of Ohio University that offered me graduate associateship from September 1994 to August 1995. This financial support had been an essential encouragement in my graduate study, and paid most of the tuition in that period. The tuition would have been a heavy financial burden to me.

I am very grateful to the establishment of my family fund which is provided by Dr. San-Chu'an Jan, M.D., Mr. San-Shen Jan, Mr. Wu-Fu Jan, Dr. Lee M. Chen, Ph.D. and Mrs. Whei-Jen Chen. This family fund has been supporting my living expense in USA since 1991. I am the first one that uses this fund, and very willing to become one of the donors of this fund for the younger members of my family. For my parents: Mr. San-Shen Jan and Ms. Yi-Ching Chang, I wish I can fulfill part of the expectation they have had on me since I was born. I wish I can make them happy by presenting this dissertation. I live on their love and I will always fight for their glory and honor.

I also wish to thank my advisor Professor Janusz A. Starzyk, Ph.D., who has provided the brilliant ideas which are the mainframe of this research, and precious guidance both in academy and life, when I needed it most.

I would also like to thank the members of my graduate committee -- Professor Robert A. Curtis, Ph.D., P.E., Professor Jeffrey J. Giesey, Ph.D., Professor Earle R. Hunt, Ph.D. and Professor Henryk J. Lozykowski, D.Sc., Ph.D. -- for their gracious participation. The priceless knowledge they provided in classes is the important foundation of this research.

I owe special thanks to Professor Liang-Gee Chen, Ph.D., Mr. Kuan-Tao Yu, Mr. George Barnes and Mr. Chi-She Chen. Professor Chen taught me concepts of such contemporary digital computer programming techniques as algorithm analysis, dynamic memory allocation managed by linking lists and pointers, when I was a freshman in National Cheng Kung University, Tainan, Taiwan. Since then, a window of my mind has been opened to a whole new world. Our world is meant by how we look at it, rather than what it actually is. Thanks to Mr. Yu's suggestion and encouragement that I began to learn C++ and MFC programming, which are the essential tools for developing SAMOC. The technical writing support of writing this dissertation is offered by Mr. Barnes. Several hundred-million-bytes of memory modules installed in my computer for SAMOC development, simulation and SPICE simulation are sponsored by Mr. Chen.

I also wish to thank my close friends during these years -- Mr. Cheng-Hu Chang (Ph.D. to be), Mr. Chi-She Chen, Dr. Da-Ming Chiang, Ph.D., Dr. Long-Bin Hsieh, Ph.D., Dr. Po-Lin Huang, Ph.D., Dr. Hewjin Jiau, Ph.D., Mr. Yung-Tsan Jou (Ph.D. to be), Dr. Yen-Ching Lai, Ph.D., Dr. Wei-Yang Lee, M.D., Mr. Chung-Chi Li, Dr. David Lin, D.O., Dr. Heng Ma, Ph.D., Mr. Kerry Dale Schutz, Miss Yu-Lin Shao, Mr. Cheng-Hung Tsai, Dr. Sinko Tsai, Ph.D., Miss Pei-Lien Wu (D.M.A. to be) and Mr. Kuan-Tao Yu -- for the precious friendships, mental supports, being my role models and being the ones I adore. Thanks to their ample knowledge, words of wisdom and brilliant minds, I lean to know, although not entirely, what to do, how to do, what to love, how to love and be courageous, patient and persistent to utilize my limited intellect and talent to finish a work and present this dissertation.

Lastly, I would like to thank Mr. Kerry Dale Schutz for his decent technical writing help for my M.S. thesis, and Dr. Sinko Tsai for his ingenious clustering algorithm, which summarized the computer simulation results of my research. Without their help, I would neither have been able to finish my M.S. thesis nor have had the change to enter the doctoral program in Ohio University. I should have thanked them four years ago, however I just turned out to be reckless at that time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS AND ACRONYMS

ANN	artificial neural network
AVL tree	a binary tree name after Adelson-Velskii and Landis
AWE	asymptotic waveform evaluation
BCR	branch constitutive relations
BJT	bipolar junction transistor
CAD	computer aided design
CMOS	complementary MOS
DC	directed current
DRAM	dynamic random access memory
EDO	extended data out
FET	field effect transistor
GB	giga bytes (10^9 bytes)
I-*V*	current-voltage
KCL	Kirchhoff's current law
KVL	Kirchhoff's voltage law
MB	mega bytes (10^6 bytes)
MFC	Microsoft foundation classes
MHz	mega Hurtz (10^6 Hz)
MNA	modified nodal analysis
MOS	metal-oxide-silicon
MPVD	multi-phase voltage doubler
MS	Microsoft: a software company
NMOS	n-channel MOS
NT	new technology: an operating system from Microsoft
OPAMP	operational amplifier

OS	operating system
PC	personal computer
PMOS	p-channel MOS
psd	power spectral density
PWL	piecewise linear
Q-V	charge-voltage
RCL	resistor-capacitor-inductor
R-C	resistor-capacitor
RISC	reduced instruction set computer
RTL	register-transfer logic
TPVD	two-phase voltage doubler
SAMOC	switched-capacitor analysis of MOS circuit
SC	switched-capacitor
SPICE	simulation program with integrated circuit emphasis
V-I	voltage-current
VLSI	very large scale integrated

Chapter 1

INTRODUCTION

Simulation is one of the most important techniques in designing, improving and optimizing engineering systems. Simulation, based on known physical theories and mathematical models of an engineering system, describes the behavior and estimates the performance of the engineering system before it is practically implemented. Simulation can also help engineers to adjust the controllable parameters of an already built or designed system and change the system's structure in order to improve or optimize its system performance. Consequently, simulations can decrease real world trial and error procedures during the research, design, improvement and optimization stages of development, thus reducing costs and improving productivity. The key to a feasible simulation is the use of adequate mathematical models based on appropriate physical theories, well constructed mathematical representations and efficient numerical methods to solve the mathematical problems. The better the mathematical models, the more likely the engineering problems are translated into a well defined set of mathematical equations. The better the mathematical representations and the more efficient the solution-finding methods, the less effort and computational expense is required at the simulation stages.

Circuits are the most fundamental and essential entities of electrical systems. Circuit analysis is one of the primary tasks of circuit design. Good circuit simulation analyzes the circuit and efficiently estimates the electrical details of the circuit's performance before the circuit is even built. Fast circuit simulation requires a powerful computer, well organized data management, an efficient simulation engine and experienced design engineers. By employing fast circuit analysis, a circuit designer discovers design errors and determine the best circuit organization and parameters in order to satisfy the design specifications in a short period and with reduced effort. One of the most critical missions of circuit simulation is to analyze very large circuits, because the trend of circuit design is to produce smaller, yet more sophisticated systems with low power consumption. The most successful approach to constructing a sophisticated system in a small volume and with low power dissipation is to utilize up-to-date semiconductor technology for very large scale integrated (VLSI) circuits. Due to the increasing complexity of VLSI circuits and new semiconductor circuit fabrication technologies, the study, development and modernization of VLSI circuit simulation techniques has become more and more important.

Circuits, from the most simple examples which can be found in high school level text books to the most complex VLSI circuits, are constructed of interconnected electrical devices. The behavior of circuits can be characterized by the amount of current, which passes through each device and the magnitude of voltage at each interconnection node. In order to translate the electrical design problems to mathematical formulas, the amounts of electric current and the voltage magnitude are represented by numerical values, and mathematical models of different electrical devices are studied and developed. In addition, Kirchhoff's laws associated with the interconnection of electrical devices are applied to formulate the circuit equations. Electrical device models link the practical world with the mathematical representation. Development of these models must either follow established physical theories or be based on the experimental measurements of the electrical device. Kirchhoff's laws are fundamental electrical laws which are derived based on the laws of energy and electric charge conservation (Different methods of formulating circuit equations will be introduced in the later part of this dissertation). Circuit analysis is therefore achieved by formulating and solving the circuit equations.

Circuit simulation depends on the type of circuit design category to be used, and may require DC analysis, sensitivity analysis, time domain analysis, small signal analysis, frequency domain analysis and timing analysis. Circuit simulation can be divided into linear and nonlinear, lumped or distributed, stationary or time varying analyses in which specialized simulators are developed and used depending on the application area. The type of simulation is determined by the presence of linear distributed line elements, time varying components, and so on. The following sections focus on different aspects of circuit simulation.

1.1 Overview

The present research is about circuit simulation techniques and strategies with emphasis on analyzing analog VLSI circuits. In order to verify and evaluate the proposed circuit simulation techniques and strategies, a new circuit simulator “SAMOC” was built in the form of a digital computer program. This work contains the development details of the new simulator: “SAMOC” both in the aspects of circuit analysis and digital computer software implementation techniques. These details underline the data structure for representing a circuit in the computer program, device modeling, circuit partitioning, equation formulation and solution, organization of the event queue and practical implementation of the simulator in C++ computer language. In addition, application examples which illustrate the precision, power and efficiency of the simulator are also presented. In particular benchmark circuits are analyzed to compare this simulator with other programs used in the microelectronic industry.

1.2 Circuit Simulators

Circuit simulators are designed to automatically process mathematical models of the electrical device as well as to formulate and solve circuit equations. By employing dependable circuit simulators and examining the statistical results, circuit designers can focus their efforts more on circuit organization, design and performance optimization and less on circuit analysis.

Modern digital computers (Von Neumann machines) which offer leading edge information and arithmetic processing abilities are the best platforms to implement circuit simulators by means of computer programming. Large memory modules in modern digital computers can store and process massive mathematical representations of device models and device interconnection information. The programmable procedures in modern computers offer a wide variety of processes which can formulate and solve circuit equations according to different circuit simulation algorithms. The robust numerical processing abilities of modern digital computers can be utilized to determine or approximate the numerical solutions of circuit equations in a short time. The study and development of circuit simulators require intimate interaction with device modeling. New device models may trigger new methods of equation formulation and numerical techniques for solving circuit equations.

The development of circuit simulators emphasizes the importance of cooperation between circuit designers and software engineers. Although, a digital computer can manage a larger amount of data and perform much faster arithmetic operations than human brain, it is much weaker in inducing theories or performing symbolic calculus. For this reason, circuit analysis algorithms must be developed in simple and straightforward ways for digital computer applications. Computers may also be slowed down by very complex and massive tasks, but they won’t get tired or complain. In order to exploit this advantage, iterative solution seeking algorithms are invented for computer applications.

1.3 Device Modeling

Electrical device modeling, which represents the characteristics of electrical devices in mathematical or numerical forms, is used to translate electrical problems into mathematical problems and make circuit analysis feasible both for engineers and computers. Circuit simulation techniques must be adaptable to device models. Research in device modeling and simulation techniques should closely interact with one another. Different categories of simulations require different kinds of models for the same device. For example, AC analysis requires AC device models, DC analysis requires DC models and high frequency analysis requires high frequency models. On the other hand, new models developed for new devices or to improve simulation accuracy may induce innovations in simulation techniques. For example, in order to employ nonlinear models, a simulation must involve nonlinear analysis techniques, and in order to handle high frequency interconnect models, partial differential equations may be needed.

Device models can be represented by mathematical equations, combinations of other modeled devices and lookup tables. The representation of a device model for one category of circuit simulation is not unique. The more accurate the model, the more reliable are the obtained simulation results. On the other hand, a complex model which offers precise device behavior descriptions may cost simulation effort and prolong the circuit design period. In some situations, not all internal details of a complex device are important to a simulation; therefore, to the decrease complexity of a simulation, the device is treated like a black box and only the device’s terminal behavior is characterized and used.

In contemporary circuit implementations, most of the devices used are fabricated from semiconductor material, and semiconductor devices are all nonlinear. Chua [1] addresses two approaches to nonlinear device modeling - the physical approach and the black box approach. The modeling of such essential semiconductor devices as *pn* junction diodes and *GaAs FET* (field effect transistor) are physical approaches and are presented in [2]-[6]. Ebers and Moll [7] express a set of equations which describe the relationship of voltage and current for *BJT* (bipolar junction transistors). The Ebers-Moll model is one of the best known *BJT* models. In contemporary VLSI designs, MOS (metal-oxide-silicon) field effect transistors, which consume lower power than *BJT*s, and can be highly integrated, dominate. The MOS model had been introduced with the development of SPICE [8]. The SPICE "level 1" model, which contains less than 10 model parameters and describes the behavior of an MOS transistor by a set of equations, are discussed by Muller and Kamins [9]. Muller and Kamins's equations are quadratic and can be found in most primary semiconductor text books. With the improvement of MOS technology and the discovery of device-characteristic geometry, "level 2", "level 3" and "level 4" (BSIM) [10] models were introduced and employed in commercial simulators such as PSPICE from MicroSim. With the progress in MOS fabrication that has already stepped into submicron technology, BSIM3[1] [11] for SPICE and Level 28[2] for HSPICE [12] models that contain 50 to 100 model parameters are employed to describe some phenomena discovered in deep submicron semiconductor physics. The more model parameters are used, the more

[1] The information about the newest development of BSIM3 can be found in http://www-device.EECS.Berkeley.EDU/~bsim3/

[2] Level 28 model is proprietary and belongs to Meta Software Corp.

complex and difficult the simulation is. Therefore, with the increasing number of MOS transistors in a single die, modern circuit simulation has already become a critical application and has created a challenge to circuit simulator developers, software engineers and computer hardware manufacturers.

1.4 Circuit Equation Formulation

The description of a circuit contains a set of electrical devices, the traits of these devices and the interconnection of these devices. The interconnection of devices creates a specific circuit topology. While the device models translate the characteristics of electrical devices into mathematical forms, the circuit topology determines the formulation of circuit equations. Kirchhoff's current law (KCL) and voltage law (KVL) are the basic electrical laws used to formulate the circuit's equations. As mentioned before, the behavior of a circuit can be characterized by a voltage value in each interconnection node and the current value through each terminal of all included electrical devices. Estimating these voltage and current values is the purpose of the circuit analysis. Among Kirchhoff's laws, KCL indicates that the algebraic sum of currents that flow into an interconnection node of an electrical device in a circuit must be zero. Kirchhoff's current law, KVL indicates that algebraic sum of the voltage drops around any closed loop in a circuit must be zero. Most of the time, KCL involves the nodal formulation and KVL involves mesh formulation.

The interconnection organization of a circuit can be treated as a topology entity which contains nodes (connection points of different devices), branches (devices) and meshes (loops of branches). To describe a linear circuit that contains only linear devices, such as ideal resistors, ideal capacitors and ideal inductors, Bashkow introduced the "*A* matrix" [13] formulation. The "*A* matrix" consists of a current matrix and a voltage matrix. To describe the behavior a circuit that contains energy storage devices, a first-order linear differential equation in a matrix form can be used. The "*A* matrix" circuit network description is also the basic formulation of the state-variable approach [14] to dynamic network analysis.

For systematic procedures which are adequate for computer applications, Branin [15] combined Ohm's and Kirchhoff's laws associated with the principle of superposition, and introduced algorithms for digital computers to formulate network equations. Branin introduced four methods: "the mesh method"; "the node method"; "the cutset method"; and "the mixed method." The circuit analysis program TAP [16]-[19] was created developed from Branin's work in the 1950s. In late the 1960s, the "sparse tableau" formulation was introduced by Hachtel *et al*. [20]. The "sparse tableau" formulation employs very simple and straightforward algorithms adequate to be implemented by a computer programming language. For a circuit network which contains $n+1$ nodes and b branches, a sparse tableau matrix contains n KCL equations, b KVL equations and b BCR (branch constitutive relations) equations. The n KCL equations are applied to the n nodes of the analyzed circuit. The b KVL equations indicate the relationship between branch voltages and node voltages. The b BCR equations indicate the relationship between branch currents and branch voltages. BCR equations are related to properties of electrical devices other than circuit topology. If all devices are linear, then BCR equations can be represented in a matrix form. The matrix size or the number of total equations is much bigger when the sparse tableau formulation is employed. Although the sparse tableau formulation is not designed to produce a concise system matrix, it is sufficient for used in computer-generated formulations.

The sparse tableau formulation was employed in the circuit analysis program ASTAP [21]-[22] developed by IBM. The sparse tableau formulation in general a creates large number of equations and, in most situations, a large matrix with a relatively small number of non-zero elements. Sparse matrix techniques for computer exploits the dynamic memory allocation mechanism and information theories in computer science. Information about sparse matrix manipulation, computer programming algorithms and adjustment for circuit analysis can be found in [23]-[30].

Modified nodal analysis (MNA) introduced by Ho *et al.* [31]-[32] is another circuit formulation algorithm designed for computer programming. MNA as well as nodal analysis creates a matrix stamp (also called an element stencil) for each device. MNA can overcome the normal nodal analysis problems caused by voltage sources, floating sources and inductive elements. MNA can also take good care of controlled sources which are useful in modeling semiconductor and other nonlinear or active devices. Compared to the sparse tableau approach, MNA creates a smaller matrix and uses a smaller number of circuit variables. Many important circuit analysis programs such as CANCER [33] by Nagel and Rohrer, SLIC [34] by Idleman *et al.* and ICD [31]-[32], [35] by Ho *et al.*, employ MNA as the circuit equation formulation algorithm. With the broader usage of semiconductor circuits, one of the most important contemporary circuit simulators SPICE [36] was introduced by Nagel and Pederson in 1973. SPICE evolved techniques of

CANCER, which is the collection of Rohrer's work about circuit analysis which includes circuit formulation, linearization, integration techniques, Gaussian elimination and LU decomposition. SPICE was placed in the public domain and gained attention and recognition from developers and users worldwide. In 1975, SPICE2 [8] was introduced and come to match the popularity of ASTAP developed by IBM. Pederson [37] made a brief comparison between ASTAP, which employs sparse tableau formulation, and SPICE2, which employs MNA. Compared with ASTAP, SPICE2 requires less time to set up, but ASTAP is easier to used for repeated analysis. ASTAP is said to be good for statistical analysis of a circuit, but it takes more effort to perform its sparse tableau formulation.

1.5 DC Solutions

Finding the DC solution of a circuit is a basic element of circuit simulation. DC solution is the key to determine the operating point when analyzing nonlinear circuits. DC solution of a circuit can be determined by formulating and solving the linear and nonlinear circuit equations. Formulating the circuit equations, such as the sparse tableau approach and MNA, was introduced in the previous section. For a circuit that contains only linear devices, the circuit equations are linear. Cramers' rule can be employed for direct solution for a linear system, but only in a nonsingular system matrix. For computer programming, Gaussian elimination is easier to implement than Cramer's rule. One of the best traits of the Gaussian elimination is that the Gaussian elimination can identify reasons for matrix singularity. Most of the time, a singular matrix is caused by isolated nodes which result from open circuits or the presence of floating devices. In some situations, Gaussian elimination can identify and remove isolated nodes and gather information from the well connected nodes. If a matrix equation must be solved several times for different inputs, then LU decomposition for a nonsingular matrix is the best algorithm and can be easily implemented by a computer. Modern simulators may use QR factorization, singular value decomposition and other numerically stable algorithms to effectively solve systems of linear network equations and identify closely related equations.

Besides the direct answer-seeking methods discussed above, some iterative methods, such as the Gauss-Jacobi method and the Gauss-Seidel method, are adequate for computer implementations of very large circuits. These iterative methods can frequently reach a close answer by using relatively few computation steps. The iterative methods are useful for solving circuit networks with limited computer resources, simulating circuits in higher abstraction levels, or in applications where accuracy is not as important as simulation time.

For nonlinear circuits, if the models of nonlinear devices are presented in nonlinear continuous function forms, the Newton-Raphson method is one of the simplest methods used to determine the numerical solution of a nonlinear system. The Newton-Raphson method is an efficient iterative procedure which uses a first order derivative approach to locate the next iteration point. The drawbacks of the Newton-Raphson method are that it may occasionally diverge or oscillate, and it is computationally expensive. The divergence and oscillations are caused by an improper initial guess. The creation and inversion of the Jocobian matrix required by the Newton-Raphson method at each iteration step creates a heavy computational burden, especially for a “big” circuit. Branin and Wang [38] applied Broyden’s method to control the scaling of the correction vector in Newton-Raphson’s iteration and improve the efficiency of the Newton-Raphson method. The Newton-Raphson method is also employed in SPICE [8].

One other way to represent a nonlinear characteristic of electrical devices is to use piecewise linear approximation; that is, to represent a nonlinear curve by several linear segments. As a result, the more segments are that used, the more accurate the approximation and the more complex the analysis. The piecewise linear approximation also gives device modeling more variety for different requirements. For example, in order to reach a more accurate simulation, a model with more segments is applied.

The Katzenelson algorithm [39], is used to solve a piecewise linear circuit. This approach is an iterative approach to determine the DC solution of a piecewise linear circuit, provided that all nonlinear devices are either voltage- or current-controlled. For nonlinear devices which are neither voltage nor current controlled, a parameteric representation of circuit components [40]-[41] was introduced. An ideal diode is one where the devices are neither voltage- nor current-controlled. Tadeusiewicz [42] presented analyzed ideal diode circuits with a proof of existence and uniqueness of the solution by using a parameteric representation.

1.6 Digital Circuits

In the real world, all circuits have continuous-time and arbitrary input and output signals. Circuit designers should prevent input and output exceeding the electrical devices' tolerance in order to avoid damaging the circuits. In addition to physical limitations on upper and lower bounds of the circuit's inputs and outputs, there is a special category of signals with only two signal levels specified. These signals are processed by a special category of circuit design: a digital circuit [43]. With the development of the digital circuits, conventional circuits been called analog circuits. Unlike analog circuits which are designed to have continuous time input and output signals, digital circuits use discrete time signals. That is, the signals are important only at several distinct moments, and at the other instances the signals are discarded. The other trait of digital circuits is that all the signals are digitized. The digitization is realized by dividing possible signal values into finite number of regions, and a signal with a value included in one region is represented by a "digit" or "symbol" which is assigned to that region. The employment of digital circuits moves part of circuit design from continuous time and values to discrete time and digitized values.

Contemporary digital circuit design uses two signal levels and therefore there are only binary symbols in digital circuits. For example, in a 5V system, a voltage signal between 4.5V to 5V can be assigned to represent one symbol, and a voltage signal between 0V to 0.5V can be assigned to represent the other symbol. The remaining part (0.5V - 4.5V) is used to indicate the improper design or infidelity of signals which could be caused by environmental or transmission noise. Most of time, the binary symbols are represented by two digits "0" and "1". Based on Boolean algebra for digital circuits organizations and designs, and technologies of VLSI manufacturing, the digital circuits thrive and are used by electrical engineers to implement the most sophisticated systems such as the newest Von Neumann machines (modern computers).

The major advantage digital circuits is that the information in a binary signal form can be accurately stored and easily reproduced. Furthermore, information in a binary signal form combined with information and coding theories [44] can withstand a much larger error tolerance in signal transmission. The precise restoration, exact reproduction and high transportation tolerance of information are the crucial conditions of building sophisticated systems. In addition, binary signals can be used to represent numerical values both in integer and real number forms. Many arithmetic operations designed for numerical values in binary form - such as increment, decrement, add, subtract, multiply and divide - have been well studied and developed. A digital system can be easily programmed to do a wide variety of jobs. Information processing and transportation, robust arithmetic operations and programmability are traits of digital systems.

In digital systems, the numerical values are represented by a finite number of binary signals (bits) in registers. The truncation error of numerical values caused by finite bits representation are inevitable and can only be decreased by prolonging the registers. That is, using more bits to represent numerical values. The longer the register, the more circuit elements and the more demanding processing speed is required.

Compared to analog circuits, digital circuits need many more circuit elements and more time to process the same amount of information. The larger design area and lower speed are compensated by the high efficiency of modern VLSI technologies which can highly integrate circuit elements and provide high speed capabilities of designed systems.

In the late 1990s, a single chip which contains 9 million MOS transistors working at 450 MHz and has many powerful arithmetic and 3-dimensional graphic functions and sophisticated hierarchy memory management mechanisms is common in the consumer electronic market. In the information storage part, a single chip with less 10 ns access time and 64 millions storage units (bits) is also common in consumer electronic market.

Digital circuits can be designed to set their own initial conditions. The DC analysis of digital circuits is not as important as in analog circuits. The design of digital circuits has moved from circuit level through gate level, register-transfer logic (RTL) level and functional level to system level.

1.7 Analog VLSI and Neural Computing

Digital circuits have become the dominating hardware used to implement information and arithmetic processing systems. People depend more and more on digital systems, but digital systems require very sophisticated programs and powerful digital systems to process some jobs which are quite easy and intuitive for humans such as voice and image recognition. Consequently, some electrical engineers have investigated using existing VLSI technologies to imitate real life neural systems. The imitated life systems are called artificial neural networks (ANN). ANN can be implemented by a digital circuit approach which uses binary signals and memory. Another approach to ANN is the use of analog circuits. Since in ANN application the relative values of signals are much more important than absolute values, it is too expensive and not necessary to prepare long register arrays to store numerical values.

Numerical information in analog circuits can be represented by magnitudes of node voltages or branch currents. Voltage signals are good for low power designs, because a voltage signal can be stored in a single capacitor device instead of being represented by a binary code and stored in a capacitor array or a long register in digital systems. Current signals can be easily added by just wiring two lines; the addition takes no time, while in a digital system a large binary adder is required to do the same job and the process of carry-in and carry-out signals may be very time costing.

Mead [45] used nature of analog circuit organizations - such as current signal duplication, R-C integration and differentiation, algebraic averaging, comparison, and arithmetic functions (addition, subtraction, absolute value, multiplication, exponential, logarithm and square root) - to build a sophisticated system. Fang [46] presented several analog circuit organizations which are useful to build an analog ANN and a design example which is an image processor. The circuit organizations Fang introduced are multipliers, digital to analog converters and winner-take-all circuits. Mead and Fang asserted that the arithmetic operations are natural to the circuits and theoretically take almost light speed to complete.

The difference between digital systems and analog ANNs is that analog ANNs have massive internal interconnections, high parallelism and accept more inputs at a time than do common digital systems. Most of the time an analog ANN does not work at as high frequency as a common digital systems. The final output values caused by changing inputs are more important than the way they evolved from initial conditions (transient analysis). Analog ANNs may still contain millions of MOS transistors and can be implemented by VLSI technologies. The VLSI implementation of an analog circuit is called analog VLSI. Analog VLSI circuits still suffer from a lack of programmability, precise analog memory and noise tolerance which have been overcome by most digital systems. Perhaps a fusion of digital and analog circuits in a system would create a new design methodology of faster and more sophisticated new systems.

1.8 Time Domain Analysis

The major task of time domain analysis is to evaluate the waveform of specified outputs as functions of time. The output waveform is most frequently evaluated with a time varying circuit excitation vector. Time domain waveform evaluation is an important method of analysis in analog circuit design. By inspecting the output waveforms, a circuit designer can verify his designs and estimate the performance of the designed circuit.

For circuits that contain no energy storage devices, the output waveform can be easily estimated by direct transformation of the circuit excitation vector. If the circuit is linear, the output of interest is a linear combination of circuit excitation vector. If the circuit contains nonlinear devices, some nonlinear transformation techniques, or a general purpose method such as Newton-Raphson iteration or the Katzenelson algorithm, are applied to evaluate the output waveform.

If the circuit contains energy storage devices such as capacitors and inductors, then the equations can be presented in the form of differential equations and the initial conditions can be found by evaluating the circuit DC solution. In order to solve differential equations with initial conditions in a digital computer, several one-step implicit integration techniques based on first order Taylor series approximations - such as forward Euler integration, backward Euler integration and trapezoidal integration,- are employed to solve the circuit in time domain numerically. To improve the accuracy of numerical integration, higher order Taylor series approximation techniques were introduced by Brayton [47]. For time domain waveform analysis of nonlinear circuits, the Newton-Raphson method is employed to solve the nonlinear nonlinear equations at each time step. The integration methods for solving differential equations with very sophisticated time step selecting algorithms are employed by such general purpose and accuracy oriented circuit simulators as SPICE, SABER and ASTAP.

For linear circuits, one approach, other than direct numerical integration to evaluate the output waveform of the formulated differential equations, uses Laplace transforms. Laplace transformation can transfer a differential problem into an algebraic problem. This approach determines the transfer function of a circuit in the *s*-domain. The Laplace transform of the output can be obtained by the algebraic product of the Laplace transform of the input signal with the transfer function. The output time domain waveform can be evaluated by the numerical Laplace transform inversion [48]. By applying this approach, the non-numerical-presentable signals, such as Dirac impulse, can be easily addressed.

1.9 Simplified Timing Analysis

Using modern semiconductor technologies, VLSI circuits which contain millions of MOS transistors can be created by a single die to decrease manufacturing costs. In the circuit designer stage, it is too strenuous and not practical to obtain the time domain waveforms by applying the accurate methods which include Newton-Raphson iteration, Jacobian matrix evaluation, implicit numerical integration and direct linear system solving algorithm for every simulation instance. The simulation can be accelerated by either a faster digital computer system, such as a super computer, or by employing simplified simulation techniques which trade accuracy for simulation speed. Practically, not all circuit designers can afford to use super computers. Furthermore, accurate simulation is not always necessary if it costs too much to obtain, especially in the early stage of the design and verification process. There are several approaches to achieve a simplified simulation, including:

1. **Employing simplified device models:** Simplified models for nonlinear devices which help simulators formulated easier-to-solve circuit equations.

2. Partitioning the simulated circuit: The average computational effort of solving an *n* by *n* linear system is O($n^{2.81}$). For simulating a large nonlinear circuit, the situation can be even worse. Finding solutions for several small circuits is easier than for a large circuit. By partitioning the simulated circuit into connected subcircuits, the simulator addresses several smaller circuits one at a time.

3. Employing event-driven simulation techniques: A portion of the simulated circuit is static under some situations, or in some situations the output of that portion is not significant. Event-driven simulation can avoid the redundancy of simulating the not-significant part of a circuit to improve the simulation's efficiency.

4. Using relaxation: The relaxation technique is an iterative method to approximate the solution of a partitioned circuit. Although finding an exact solution by using relaxation technique may take as much or even more computational effort than a direct method, a close solution beneath acceptable tolerance can be swiftly obtained.

5. Using waveform evaluation techniques: The impulse responses of a linear R-C circuit are a linear combinations of the exponential functions of time with different time constants. Waveform evaluation employs the moment matching technique to determine several dominant time constants which can be used to approximate the waveform.

6. Simulating at a higher abstraction level: For a portion of a circuit, if its electrical behavior has been well studied, then this portion of the circuit can be represented by higher abstraction level using an analog hardware description language. The simulator can then focus on the behavior of the other parts of the circuit.

1.9.1 Simplified Device Models

Simplified models are often presented by a piecewise linear or a piecewise constant method to exploit the well-known linear system techniques. In piecewise methods, the nonlinear device characteristics are divided into several linear or constant regions. Linear circuit analysis is employed inside each linear region and the Katzenelson algorithm [39] or a piecewise linear version of Newton-Raphson algorithm like POPCORN [49] is used to handle the nonlinearity between linear regions.

Several simplified MOS transistor models were developed for fast timing MOS circuit simulation. MOTIS [50] uses a two-dimensional table representation of drain to source current (I_{DS}) as a function of MOS transistor's terminal voltages (V_{GS} and V_{DS}). SPECS [51] (Simulation Program for Electronic Circuits and Systems) uses a piecewise constant representation of drain to source current (I_{DS}) as a function of MOS transistor's terminal voltages (V_{GS} and V_{DS}). The area in which I_{DS} has a constant value is a rectangle defined by specified intervals of V_{GS} and V_{DS} as shown in Fig. 1.1. The number of total defined area and size of each constant areas are changeable according to requirement of simulation accuracy.

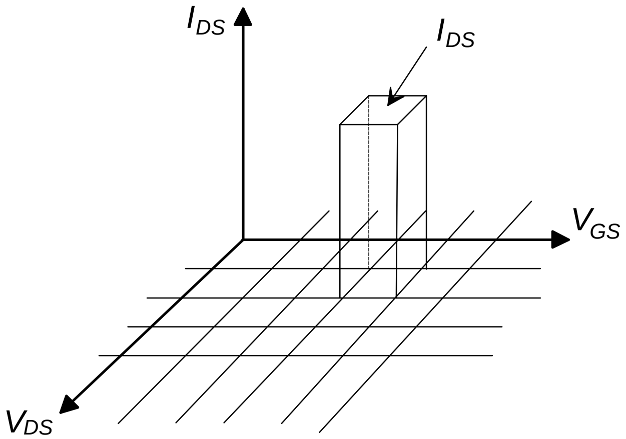


Fig. 1.1 Piecewise constant MOS model used by SPEC.

1.9.2 Circuit Partitioning and Event-Driven Simulation

Circuit partitioning and event-driven simulation are two important techniques used to accelerate the timing simulation. MOTIS [50] was designed to simulate digital circuits which contain only MOS transistors. MOTIS employs a backward Euler formula to determine the change of a logic gate output. This change will propagate to the inputs of gates of the next stage to see if the change causes an event.

SAMSON [52] is an analog and digital circuit simulator which partitions the simulated circuit into subcircuits. Each subcircuit is simulated by using independent time steps from other subcircuits. When solutions of subcircuits reach some predefined pattern, the subcircuits interact with each other. A subcircuit can be "*alert*" when there is a simulation event or become "*dormant*" when there is no simulation event. The computational effort of the simulation can be saved by the appearance of the "*dormant*" subcircuits.

1.9.3 Relaxation Techniques

Relaxation techniques [53]-[54] using Gauss-Jacobi or Gauss-Seidel iterative methods can approximate a close network solution with less effort than direct solution seeking methods. In analyzing a nonlinear circuit or system, relaxation techniques help to avoid using the Newton-Raphson iteration. Timing simulation using relaxation techniques usually combines with circuit partitioning to avoid redundant computation.

1.9.4 Moment Matching and Asymptotic Waveform Evaluation (AWE)

The poles of a transfer function are the time constants of Laplace inversion of an impulse response. The time domain response of a linear system can be obtained by convoluting the input with the impulse response. Moment matching techniques can approximate the dominant poles of the transfer function of a RCL circuit. Asymptotic waveform evaluation [55] (AWE) is the q^{th} order extension of dominant pole approximation.

1.10 SAMOC Simulator

MOS transistors are the most popular devices used in VLSI circuits. Many circuit simulators such as SPICE, ASTAP, SAMSON and Rsim [57] can already analyze MOS circuits with different computational characteristics, such as speed or precision. However, simulating a VLSI circuit containing more than ten thousand MOS transistors at the circuit level has become impractical on current computer systems. Simulation of a VLSI circuit is performed either exactly taking portion by portion of the whole circuit or performed in higher abstraction levels.

In the current circuit design paradigm, digital circuits or analog neural systems are the most sophisticated systems used perform arithmetic operations and process signals and information. The common trait of both digital circuits and analog neural systems is that the DC solution conveys the most desired information about a state of the system with specified inputs. That is, the processed essential information is presented and stored in form of DC currents or voltage levels but not the signal frequency or signal delay time.

However, the existing circuit simulators may spend too much effort in finding "how the signals reach their steady states." Although the existing circuit simulators may provide very likely prediction the system state, the large computational effort used limits the capability and speed of the simulator in analyzing larger circuits.

The main objective of developing SAMOC (Switched-capacitor Analysis of MOs Circuits) simulator is to develop the fastest method of approximating the DC solution of a VLSI circuit according to the time varying circuit excitation. While SPICE and other simulators evaluate the exact output waveform, SAMOC concentrates the computational effort and computing resource to determine the approximate DC solutions in different time intervals as illustrated in Fig. 1.2. By using SAMOC, a circuit designer can determine functional design flaws in a short time before exhaustive precise simulation is performed, therefore, short the circuit design period is achieved.

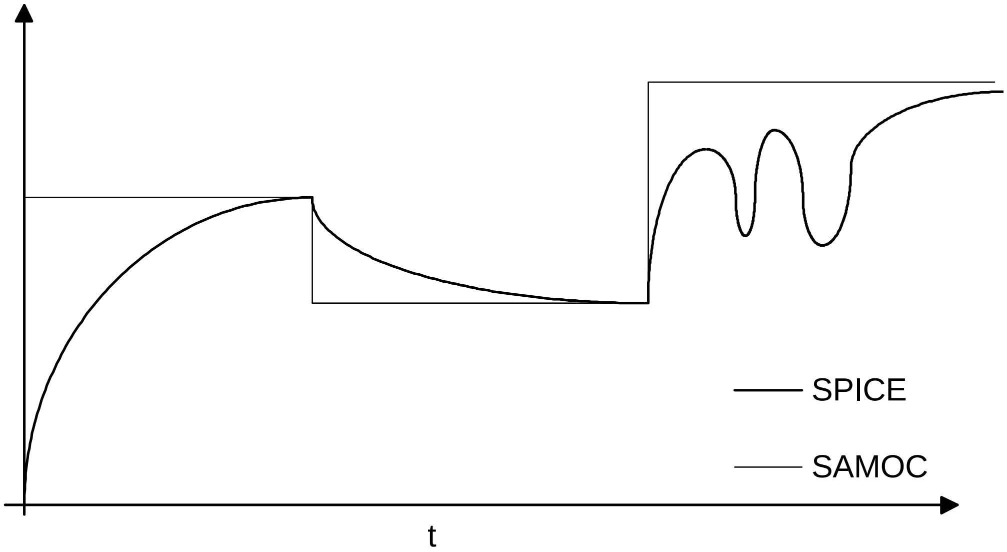


Fig. 1.2 Piecewise constant approximation.

In simulating VLSI circuits, SAMOC employs an extremely simplified MOS transistor piecewise linear model to push the capability of MOS circuit simulator to the limit of a computer system. Time consuming integration procedures mentioned in section 1.8 are not employed. SAMOC finds the final DC solution directly. In order to further accelerate the simulation, SAMOC employs circuit partitioning and event-driven mechanisms, which can eliminate redundant calculations in analyzing a VLSI circuit. Circuit partitioning and event-driven mechanisms are equipped with fast simulators introduced in section 1.9. For circuit equation formulation, SAMOC employs a the modified nodal analysis (MNA) circuit formulation algorithm which can handle the presence of operational amplifiers, ideal switches and controlled sources. MNA is also adequate for the piecewise linear models which are adopted in SAMOC to fit in. Since ideal switches might cause floating nodes and a singular equations matrix, Gaussian elimination is more adequate in SAMOC than LU decomposition for solution seeking. To solve the piecewise linear system, the Katzenelson algorithm is included in SAMOC.

In order to verifying the simulation results at the development stage of the SAMOC simulator, SAMOC was designed to read the SPICE circuit description format. After verifying the SAMOC simulation by using small SPICE format circuits, large SPICE format circuits can be used to compare the simulation capability, precision and required time. SAMOC is written in Microsoft® Visual C++ ® combining Microsoft® foundation classes (MFC) application framework, compiled into a 32 bit application with the Microsoft® Windows 95/98 or Windows NT operating systems. The object-oriented programming language C++ is powerful, versatile and adequate for writing and maintaining large and sophisticated programs. The MFC offers simple ways to build data structures and graphical user interfaces. Low cost IBM PCs are popular in general applications and are becoming the preferable domain for computer aided design (CAD) tools. Both Microsoft® Windows 95/98 and Windows NT operating systems have easy-to-use graphical user interfaces, and offer programmers up to 2GB of addressable memory that allows the simulator to handle large amount of circuit devices and nodal equations. Windows NT, which provides robust, crash proof, multithread, multitasking abilities and supports computers with multiprocessors, is an excellent platform for developing and executing circuit simulation programs as well as other computing resource and power demanding applications.

Details of the development of SAMOC and some SAMOC applications are presented in the following chapters. Chapter 2 presents the piecewise linear device resistive model for semiconductor devices and the circuit equation formulation by filling device stamps to the circuit matrix. Chapter 3 presents the modified Katzenelson algorithm for the piecewise linear model presented in Chapter 2. Chapter 4 presents the simulation techniques used to analyze switched-capacitor networks via capacitor equation formulation and direct solution seeking method. Both Chapters 3 and 4 present simple simulation examples. Chapter 5 presents an example design and simulation of switched-capacitor charge pump circuits using SAMOC. Chapters 6 and 7 present the methods and algorithms of circuit partitioning and event-driven simulation. By applying algorithms in Chapter 6 and 7, SAMOC transfers the traditional device-node circuit simulation problems into block-signal problem. Chapter 8 contains benchmark circuits simulation examples which exploit the circuit partitioning and event-driven simulation mechanisms built into SAMOC. Chapter 9 contains the conclusion.

Chapter 2

DATA STRUCTURE, MODIFIED NODAL ANALYSIS AND PIECEWISE LINEAR SEMICONDUCTOR DEVICE MODELS OF SAMOC

The goal of developing SAMOC is to investigate and test some MOS circuit simulation ideas which can aid in handling thousands or even millions of MOS transistors by using low cost personal computers. Before any circuit analysis subroutines are executed in SAMOC, SAMOC must construct a data structure to handle the whole circuit device information. Because the memory in a computer is limited, the data structure is crucial to the maximum number of devices SAMOC can handle. After the data structure is built, a subroutine will formulate the circuit equations using the information of SAMOC's data structure and device models. Presently, SAMOC formulates the circuit by the method of modified nodal analysis (MNA) and models the nonlinear device by means of a piecewise linear approach. MNA, which is easy for computer programming implementation and can handle controlled devices and ideal switches, is an adequate choice. The piecewise linear model can avoid the complex Newton-Raphson root seeking routine. It is a good approach to simulating large circuits which contain many nonlinear devices. The following sections of this chapter will present the SAMOC data structure, piecewise linear models and MNA.

2.1 Building Data Structures in SAMOC

In a computer program, the organization of program data and memory is called the data structure. The first consideration in coding for SAMOC is the data structure and the memory management which prepares a working area (memory block) for such device information as connections, type, model and parameters. The data structure and the memory management define the first limitation of how many devices SAMOC can handle. The construction of the data structure in SAMOC is based on a circuit description. The circuit description, which is the core media of the computer aided circuit design, presents all details of a circuit design in an electronic form which is easy to be duplicated, modified, exchanged, archived and distributed. The most popular circuit description is used by SPICE. A typical SPICE circuit description consists of a list of script lines. Each script line represents one device in the circuit. Each script line contains the information about the type of the device, parameters of the device and connecting nodes of the terminals of the device. Fig. 2.1 shows the schematic of a simple MOS circuit. The SPICE description of the circuit is:

```
Vin 1 0 4V
VDD 5 0 5v
M1 5 1 2 5 l=1u w=2u pmos
M2 2 1 0 0 nmos
```

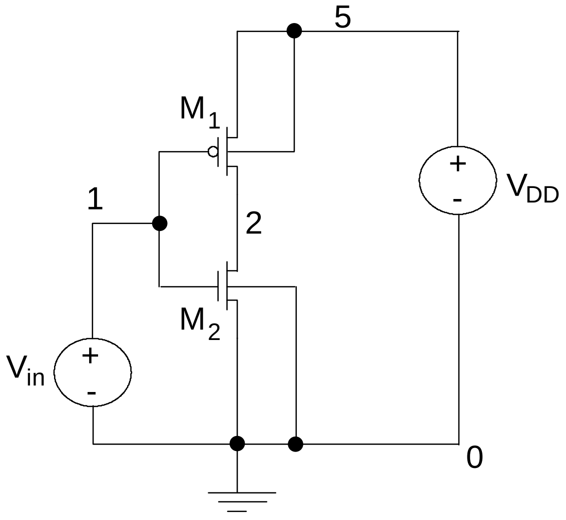


Fig. 2.1 A simple MOS circuit

The circuit shown in Fig. 2.1 contains 4 devices: 2 voltage sources and 2 MOS transistors. The description contains 4 lines. Different devices are recognized by the first character of these lines. In the description

```
Vin 1 0 4V
```

"`V`" means this device is an independent voltage source and "`Vin`" is the device name. There are two terminals of an independent voltage source. The two terminals of "`Vin`" are connected to node "`1`" and node "`0`". Different types of devices have different numbers of terminal connections, different types of parameters and specified device types and models. For example, in the description

```
M1 5 1 2 5 l=1u w=2u pmos
```

"M" indicates the device is a MOS transistor. The MOS transistor contains 4 terminals and the terminals are drain, gate, source and substrate which are connected to node "5", "1", "2" and "5" respectively. In addition, "l=1u w=2u" indicates the geometric information (channel size) of the MOS transistor, and "pmos" indicates that this is a positive type MOS transistor. In order to handle the device information which contains multitype, variable numbers of device terminals and other extra device information such as voltage values, channel sizes or models, the programming language C++, which contains the function of class derivation and inheritance, can handle the complex data type quite well. In C++ programming [61]-[62], the custom defined data type can be implemented by means of using *class*. A *class* is a group of data and functions. The data and the functions are used to represent features of the *class*. For example, a MOS transistor is a *class* and the data of a MOS transistor can be the channel conductance and the connecting nodes. Voltage-current relationships can be described by one of a MOS transistor *class* member functions. A new *class* can be built from nothing or can inherit data from existing *class(es)* and new feature can be added in addition to the features of the existing *class(es)* which the new *class* is derived from. In this sense, the existing *class* is called the base *class* and new *class* is called a derived *class*. The derived *class* inherits selected features from the base *class*. Inheritance can be single inheritance or a multiple inheritance. For a single inheritance, the derived *class* is derived only from one base *class*. In a multiple inheritance, a derived *class* can inherit from many other *classes*. A *class* is used to represent the characteristics of an *object;* an *object* is an instance of its associated *class*. For an example in SAMOC, when SAMOC needs to manipulate a MOS transistor, SAMOC creates an *object* from the MOS transistor *class* which contains information about the specific MOS transistor. That is, a MOS transistor is an *object* and the idea of the MOS transistor is a *class*. Therefore, if SAMOC reads a circuit description which contains 100 MOS transistors, then SAMOC creates 100 objects from the MOS transistor *class* and gives them different attributes according to the connecting nodes, transistor type, geometric information and model of each transistor in the circuit description.

2.1.1 Device Classes

Inheritance in SAMOC is single; the inheritance graph is shown in Fig. 2.2. The 12 supported device types in SAMOC are categorized into 6 classes. In C++ programming custom, all *class* names begin with an uppercase ‘`C`’. The base class is `C2Term,` which represents the simplest circuit devices. `C2Term` contains data about the name of the device, 2 connected terminals and 1 numerical datum. The devices represented by `C2Term` are a capacitor `(C),` an inductor `(L)` and an independent current source `(I).` The class `C2VTerm` is derived from `C2Term`. Similar to `C2Term,` `C2VTerm` is also used to represent devices with 2 terminals, but the devices represented by `C2VTerm` need additional current information in MNA. `C2VTerm` is used to represent a resistor `(R)` and an independent voltage source `(V)`. The class `CDiode,` derived from `C2Term,` is designed for an ideal diode `(D).` In addition to 2 terminal connections and current information, 2 important voltage values, turn on voltage and break down voltage are included in the `CDiode`. *Class* `C4Term` is designed to represent controlled sources such as voltage controlled voltage source `(E),` current controlled current source `(F),` voltage controlled current source `(G)` and voltage controlled switches `(S)`. For the other controlled source, current controlled current source `(H)`, another class `C4HTerm` is derived from `C4Term` for the additional current information needed by current controlled current source.

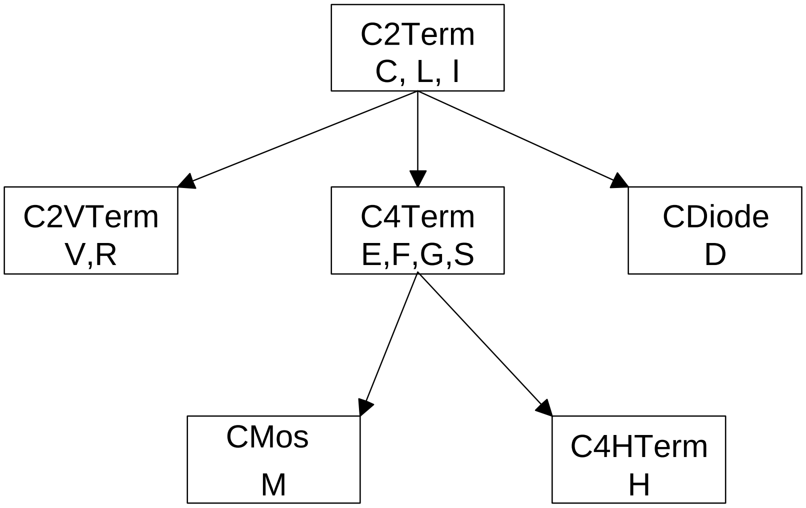


Fig. 2.2 Class inheritance graph of SAMOC

The most important *class*: `CMos` is used to represent and store data about MOS transistors. `CMos` is derived from the `C4Term`. In addition to device name and the 4 terminals' names, CMos contains the geometric information, type of transistor and a pointer to the MOS model. A MOS model is also an *object* of the *class* CMosModel.

C++ programming language has strong type checking. That is, even a pointer (address variable) is constrained to point to a specified type of datum. The pointer used to locate the address of an *object* of the base *class* can be used to locate an *object* of the derived *class* but not vice versa. The main reason for using derived classes instead of defining whole new classes is that the base class C2Term and all the derived classes such as C2VTerm, C4Term, CDiode, C4HTerm and CMos can be all located by the same pointer and exploit usage of *virtual functions*. This simplifies programming, upgrading and maintaining of SAMOC.

Memory management is also handled by functions of C++: new and delete. When SAMOC reads a device description, SAMOC checks the first character of the device description and determines type of the device. Then, SAMOC creates an *object* of the associated *class* according to the type of the device by the function new, and records the address of this object in an address variable, which is also called a pointer. The function new will find a proper space to accommodate the created object and return the address of the object. If there is no memory available, i.e., there are too many devices, an error message will be returned and the program will be terminated. The collection of pointers of all devices is managed by a *class* named "CTypedPtrList", which is one of the Microsoft Foundation Classes (**MFC**). An object of CTypedPtrList is a list of pointers. The list can be arbitrarily long in case the computer system has enough memory or contains less than 2^{32} pointers. **MFC** offers many convenient access functions for

`CTypedPtrList`. Fig. 2.3 shows the SAMOC data structure which represents the device information created for the circuit in Fig. 2.1. There are 4 devices in the circuit shown in Fig. 2.1. Two objects of class `C2VTerm` are created to represent the information of two independent voltage sources, `VDD` and `Vin`. Two *objects* of *class* `CMos` are created for 2 MOS transistors `M1` and `M2`. The pointers of these four *objects* are collected and managed by a list of pointers; the list is an *object* of *class* `CTypedPtrList`. Many further procedures such as forming circuit equations begin with visiting every device objects pointed by this list.

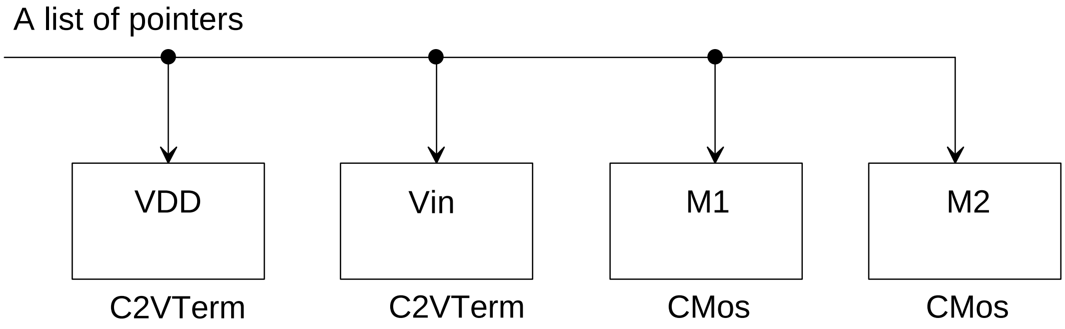


Fig. 2.3 Data structure of the device objects.

2.1.2 Node Class

One very important *class* other than the device *classes* presented in the above section in SAMOC is the node *class*, `CNode`. A node is an interconnection of terminals of different devices. An object of the *class* `CNode` contains a double precision floating point value for node voltage value. The objects of the *class* `CNode` are created at the moment the objects of devices are created. The pointers of `CNode` are also collected and managed by a list of pointers which is also an object of the *class* `CTypedPtrList`.

To speed up data manipulation, details of which will be presented in the following chapters, a node must indicate which devices are connected to it; therefore, an *object* of *class* `CNode` has a list of pointers which contains the addresses of the connected device objects. Unlike the device *class*, which has a definite number of connected nodes, a node can be an interconnection of indefinite number of devices; therefore, a list of device *object* pointers is built at each node *object*. That is, an *object* of `CNode` also contains an *object* of `CTypedPtrList`.

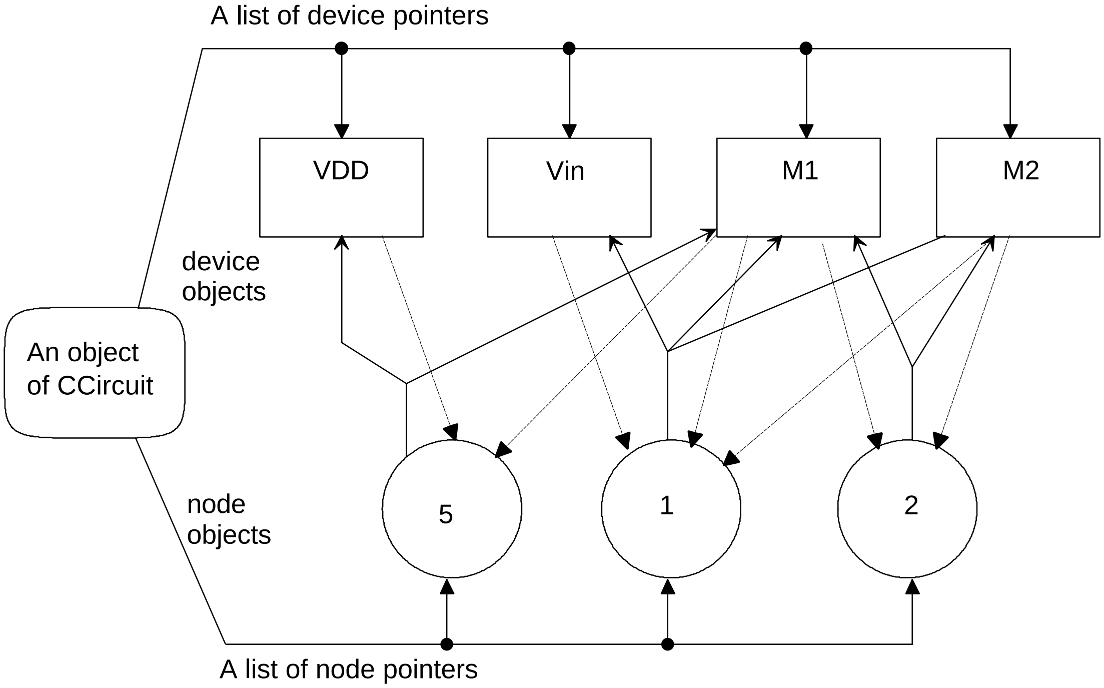


Fig. 2.4 SAMOC nodes and devices data structure of the circuit in Fig. 2.1.

2.1.3 Circuit Class

Fig. 2.4 illustrates the data structure of devices and nodes of the circuit schematic shown in Fig. 2.1. Since C++ is an object oriented programming language, SAMOC also creates a circuit *class* `CCircuit` for each simulated circuit. For each circuit *object*, there are two lists of pointers that are objects of `CTypedPtrList`. One is the device list (list of pointers of device *objects*), and the other is the node list (list of pointers of node *objects*). Each device *object* contains a definite number of pointers that point to its connecting node *objects*, and the node *objects* point back to an indefinite number of device *objects* via a list of pointers to the device *objects*. With the massive pointing structure in SAMOC, the addresses of device *objects* and node *objects* can be obtained by several direct methods that improve the speed of accessing data inside SAMOC. The node ‘0’ is the default ground. Typically, there is no need to generate an *object* for the node ‘0’.

2.2 Modified Nodal Analysis (MNA)

After the data structure for the device and node pointer lists are completed, SAMOC begins to formulate the circuit by means of MNA. The first step in performing MNA is to decide how many equations are needed to analyze the circuit. Typically, the number of equations N can be evaluated from:

$$N = N_{node} + N_R + N_V + N_{diode} + N_{c\text{-}source} + N_H \qquad (2.1)$$

where

N_{node} = number of nodes in the node list

N_{R} = number of resistors in the device list

N_{V} = number of voltage source in the device list

N_{diode} = number of diodes in the device list

$N_{c\text{-}source}$ = number of controlled sources in the device list

N_{H} = number of current controlled current sources in the device list.

By knowing N, SAMOC can ask the operating system to allocate a memory block for space to process the circuit equation

$$\boldsymbol{T}\,\boldsymbol{X} = \boldsymbol{W}, \qquad (2.2)$$

where ***T*** is an N by N circuit matrix, ***X*** is an N by one solution vector and ***W*** is an N by one excitation vector. In SAMOC, all elements in MNA matrix equations are represented by double precision floating point numbers for highly accurate computation. It is very clear that the memory required to solve equation (2.2) is the second limitation of a computer system to simulate a circuit. A large N not only consumes computer memory but also slows down the answer-seeking procedure of equation (2.2), because even in the best conditions, when (2.2) is a purely linear system, solving this N by N linear system the computation complexity is $O(N^3)$ operations. A method for decreasing N will be discussed in Chapter 6.

After the required memory space is obtained, SAMOC begins to formulate the circuit equations by filling the circuit matrix ***T*** with device stamps. The devices are located by visiting all device objects in the device list. Each device object which represents a device in the simulated circuit, and contains such required information as connecting nodes, device parameters and even model information. For linear devices, the device stamps for MNA are presented in Appendix I. For nonlinear devices, SAMOC so far supports ideal diodes, ideal switches and MOS transistors. A nonlinear device is modeled by a piecewise linear approach presented in section 2.3.

2.3 Piecewise Linear Models of Semiconductor Devices for MNA

A piecewise linear approach to nonlinear device modeling eliminates the need for the computationally expensive Newton-Raphson method so that the solution procedure can be accelerated. The nonlinear devices modeled by the piecewise linear MNA supported by SAMOC are ideal switches, ideal diodes and MOS transistors.

2.3.1 Ideal Switch

Rigorously speaking, MNA's ideal switch modeling is natural but not a piecewise linear. The ideal switch model is used to derive the piecewise linear model of an ideal diode. The ideal switch modeling used by SAMOC is presented by Vlach [58]. For an ideal switch connecting nodes denoted *j* and *j'*, the device stamp is

$$\begin{array}{c|ccc|} & j & j' & m+1 \\ \hline j & & & 1 \\ j & & & -1 \\ m+1 & F & -F & F-1 \end{array} \tag{2.3}$$

where *m* +1 is an additional equation dedicated for the ideal switch, while the switch is an open circuit *F=0* and *F=1* while the switch is a short-circuit.

Voltage or current controlled ideal switch is a controllable way to implement a switch in a circuit. SAMOC accepts a voltage controlled ideal switch whose circuit symbol is shown in Fig. 2.5. The input format of the voltage controlled ideal switch is:

```
S[s]_name s1 s2 n1 n2
```

where S or s indicates the type of the device is voltage controlled voltage, _name is the name extension or ID of this voltage controlled ideal switch, `s1` and `s2` are two nodes the switch has to short or to open, and `n1` and `n2` are the two controlling nodes. That is, `If V(n1) > V(n2), then F = 1, else F=0`.

Note, that voltage controlled switch can be described by (2.3) if columns and rows *j* and *j'* are replaced by `n1` and `n2`,

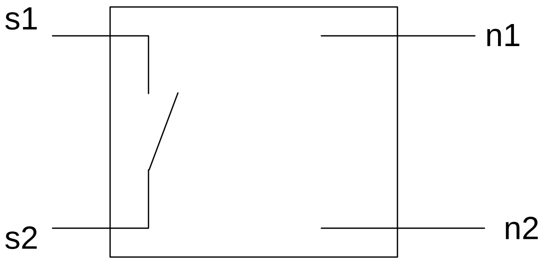


Fig. 2.5 Circuit symbol of a voltage controlled ideal switch

2.3.2 Ideal Diode

The ideal switch that has the voltage current (*V-I*) characteristic shown in Fig. 2.6 is used to obtain a model of an ideal diode. The input format of ideal diode in SAMOC is

```
    D[d]_name n1 n2 E1 E2
```

E_1 is the cut-in voltage and E_2 is the breakdown voltage. SAMOC expects that E_1 and E_2 are numerical values and $E_1 > E_2$. As Fig. 2.6 shows, the ideal diode is neither a voltage controlled nor a current controlled resistive device. SAMOC models the ideal diode by assigning 3 regions in the *V-I* characteristic. Region 1 is the cutoff region. Region 2 is the turned on region and region 3 is the breakdown region. While the operating point is in region 1, the ideal diode is modeled as an open circuit and presented by an ideal switch with $F=0$. While the operating point is in region 2 and 3, the ideal diode is modeled by two segments of independent voltage sources with different voltages.

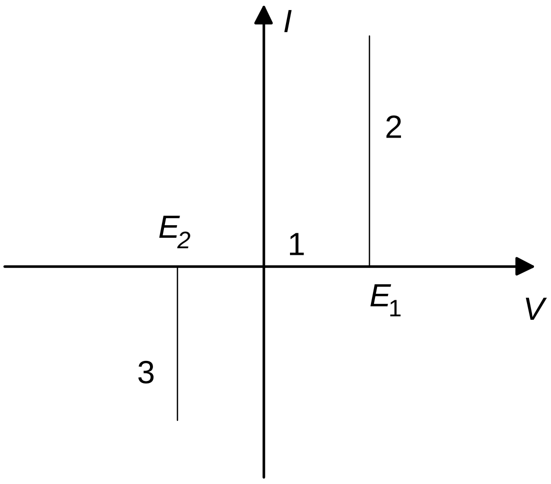


Fig. 2.6 *V-I* characteristic of an ideal diode.

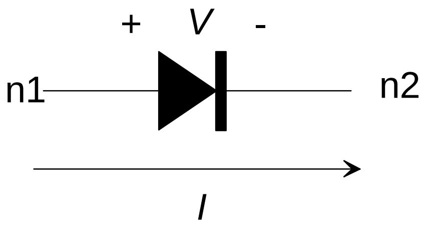


Fig. 2.7 An ideal diode.

The circuit symbol of an ideal diode is shown in Fig. 2.7. The device stamps an ideal diode in region 1, that is $E_2 < V < E_1$, is shown equation (2.4). The (m+1)*th* equation is the additional equation dedicated for this ideal diode.

$$\begin{array}{c|ccc|} & n1 & n2 & m+1 \\ \hline n1 & & & 1 \\ n2 & & & -1 \\ m+1 & 0 & 0 & -1 \end{array} \qquad \begin{array}{c} \text{excitation vector} \\ \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right] \end{array} \tag{2.4}$$

While an ideal diode is in the turned on state, $V > E_1$, the ideal diode is modeled by an independent voltage source with voltage equal to the turned on voltage E_1. The device stamp is shown in equation (2.5).

$$\begin{array}{c|ccc|} & n1 & n2 & m+1 \\ \hline n1 & & & 1 \\ n2 & & & -1 \\ m+1 & 1 & -1 & 0 \end{array} \qquad \begin{array}{c} \text{excitation vector} \\ \left[\begin{array}{c} 0 \\ 0 \\ E_1 \end{array}\right] \end{array} \tag{2.5}$$

While an ideal diode is in the break down state, the device stamp in MNA is shown in equation (2.6).

$$\begin{array}{c|ccc|} & n1 & n2 & m+1 \\ \hline n1 & & & 1 \\ n2 & & & -1 \\ m+1 & 1 & -1 & 0 \end{array} \qquad \begin{array}{c} \text{excitation vector} \\ \begin{bmatrix} 0 \\ 0 \\ E_2 \end{bmatrix} \end{array} \qquad (2.6)$$

2.3.3 MOS Transistor

MOS transistors used to implement majority of contemporary VLSI circuit are the most important devices to SAMOC. MOS transistors can be categorized by two types, PMOS and NMOS, according to the different doping material in the silicon channel. The MOS transistor models were briefly introduced in section 1.2. The simplest model is Muller and Kamins model, also known as the *level-1*, and is the default model in SPICE. In order to obtain a fast simulation, the MOS transistor model used by SAMOC is even simpler than Muller and Kamins model presented in [9]. Muller and Kamins MOS transistor equations are:

Cutoff or subthreshold region:

$$I_{DS} = 0 \qquad V_{GS} \leq V_t. \qquad (2.7)$$

The linear or triode region:

$$I_{DS} = \beta\left[\left(V_{GS} - V_t\right) \times V_{DS} - \frac{V_{DS}^2}{2}\right] \qquad 0 < V_{DS} < V_{GS} - V_t \qquad (2.8)$$

The saturation region:

$$I_{DS} = \beta \left(\frac{(V_{GS} - V_t)^2}{2} \right) \qquad 0 < V_{GS} - V_t < V_{DS} \qquad (2.9)$$

where

β: MOS transistor gain factor

V_t : the threshold voltage.

Muller and Kamins equations describe the electrical behavior of MOS transistors in quadratic polynomial with two variables, V_{GS} and V_{DS}. Solving network equations with Muller and Kamins equations may force computers to utilize the Newton-Raphson method and Jacobian matrix evaluation for each iteration. If the simulated circuit contains many MOS transistors, then the solution seeking procedure may become a heavy burden to the computer system. SAMOC addresses the bottleneck of MOS circuit simulation in MOS modeling by using a piecewise linear resistive approach. The MOS equations used in SAMOC are:

Cutoff region:

$$I_{DS} = \frac{V_{DS}}{R_{\max}} \qquad V_{GS} \le V_t. \qquad (2.10)$$

In this region, SAMOC models an open circuit with an large resistor. The equivalent circuit of a MOS transistor in the cutoff region is illustrated in Fig. 2.8 and the corresponding device stamp is shown equation 2.11.

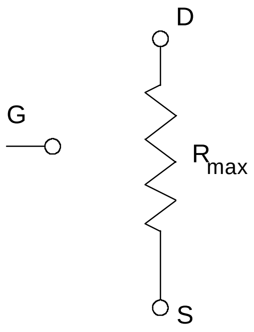


Fig. 2.8 The equivalent circuit of a MOS transistor in the cutoff region.

$$\begin{array}{c c} & \begin{array}{cc} V_D & V_S \end{array} \\ \begin{array}{c} D \\ S \end{array} & \begin{bmatrix} G_{\min} & -G_{\min} \\ -G_{\min} & G_{\min} \end{bmatrix} \end{array} \tag{2.11}$$

Linear region:

$$I_{DS} = \frac{V_{DS}}{R_{\min}} \qquad \begin{cases} V_{GS} > V_t \\ V_{GS} - V_t > \dfrac{V_{DS}(R_{\max} - R_{\min})}{g_m R_{\max} R_{\min}} \end{cases} \tag{2.12}$$

In the linear region, a MOS transistor is modeled by a small resistor. The equivalent circuit of a MOS transistor in the cutoff region is illustrated in Fig. 2.9 and the corresponding device stamp is shown in equation (2.13).

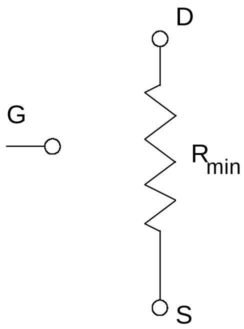


Fig. 2.9 The equivalent circuit of a MOS transistor in the linear region.

$$\begin{array}{c|cc} & V_D & V_S \\ \hline D & G_{\max} & -G_{\max} \\ S & -G_{\max} & G_{\max} \end{array} \qquad (2.13)$$

Saturated region:

$$I_{DS} = g_m V_{GS} + \frac{V_{DS}}{R_{\max}} \qquad \begin{cases} V_{GS} > V_t \\ V_{GS} - V_t < \dfrac{V_{DS}(R_{\max} - R_{\min})}{g_m R_{\max} R_{\min}} \end{cases} \qquad (2.14)$$

In the saturated region, a MOS transistor is modeled by a large transistor R_{max} and a voltage controlled current source with transconductance g_m. The equivalent circuit of a MOS transistor in the saturated region is illustrated in Fig. 2.10 and the corresponding device stamp is shown in equation 2.15.

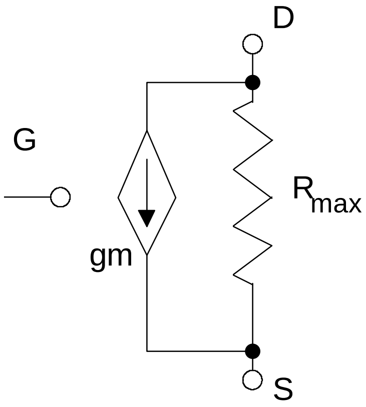


Fig. 2.10 The equivalent circuit of a MOS transistor in the saturated region.

$$\begin{array}{cc} & \begin{array}{ccc} V_D & V_G & V_S \end{array} \\ \begin{array}{c} D \\ S \end{array} & \begin{bmatrix} G_{\min} & g_m & -g_m - G_{\min} \\ -G_{\min} & -g_m & g_m + G_{\min} \end{bmatrix} \end{array} \qquad \begin{array}{c} \text{excitation vector} \\ \begin{bmatrix} g_m V_t \\ -g_m V_t \end{bmatrix} \end{array} \tag{2.15}$$

For approximation,

$$G_{\max} = \beta\left(V_{DD} - V_t\right) \tag{2.16}$$

$$g_m = 0.5 \times G_{\max} \tag{2.17}$$

$$G_{\min} = \frac{G_{\max}}{1000} \tag{2.18}$$

where

β: MOS transistor gain factor

V_t : the threshold voltage.

V_{DD} : the power supply voltage.

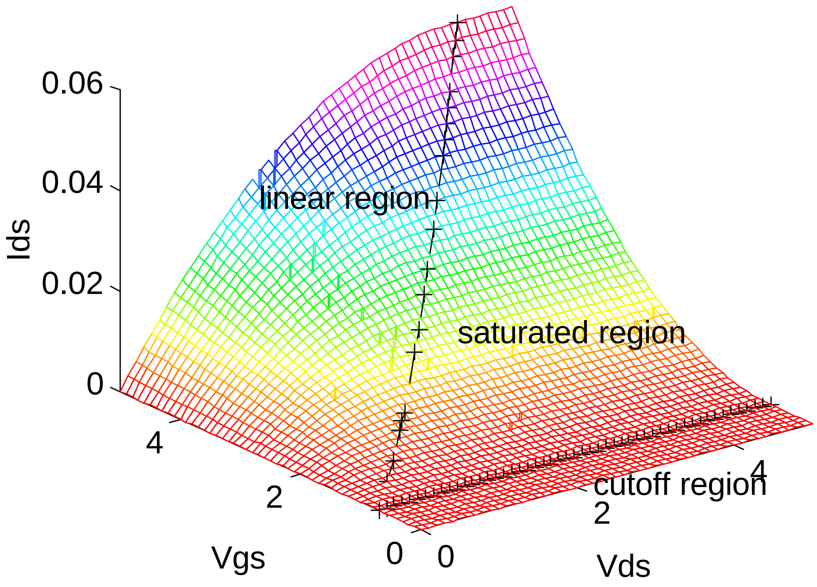


Fig. 2.11 3D illustration of Muller and Kamins MOS model.

Both Muller and Kamins and SAMOC models are expressed by I_{DS} as a function of V_{DS} and V_{GS}. A comparison of Muller and Kamins with SAMOC MOS models can be visualized by 3 dimensional plots of both sets of equations about I_{DS}. Fig. 2.11 shows the 3D visualization of Muller and Kamins MOS model and Fig. 2.12 shows the 3D visualization of the SAMOC model. Lines marked by ‘+’ on the 3D surfaces represent the borderlines of different regions.

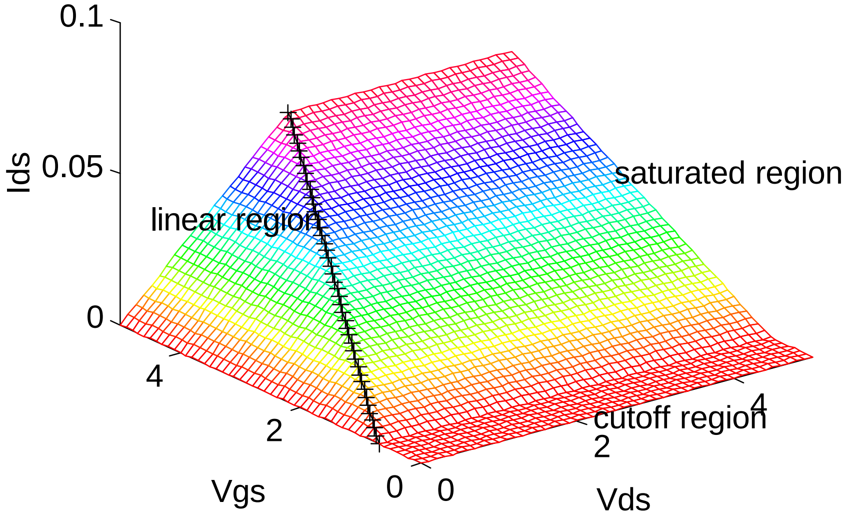


Fig. 2.12 3D illustration of SAMOC MOS model

2.4 Summary

SAMOC uses the dynamic memory allocation technique in C++ to restore and manipulate the indefinite number of devices and nodes in a simulated circuit. The devices in the simulated circuit are represented by circuit classes which are derived in hierarchical fashion in order to simplify the use of pointers. Massive inter-pointing data structure between device objects and node objects provide direct and fast object allocation.

The modified nodal analysis (MNA), which can handle the controlled sources and the ideal switches, and which creates relatively small size of system matrix (for instance as compared to the "sparse tableau approach"), is employed by SAMOC. SAMOC reads the circuit description in SPICE format which consists of a list of electrical devices. Each listed device item contains information about the type and parameters of the device and the symbols or numbers which represent the connecting nodes respectively. SAMOC assigns a number to each interconnection node and creates a space for manipulation of the circuit matrix, the excitation vector and the solution vector. Device stamps are mapped into the circuit matrix and the excitation vector according to the type of the device. The locations of stamps are determined by the assigned numbers of the interconnecting nodes. The nonlinear semiconductor devices, such as the ideal diodes and MOS transistors, are modeled by the piecewise linear approaches in order to achieve a fast circuit simulation mechanism. In the next chapter, the Katzenelson algorithm, which is used to solve piecewise linear systems, will be introduced.

Chapter 3

PIECEWISE LINEAR APPROACHES AND THE KATZENELSON ALGORITHM

In Chapter 2, the piecewise linear approaches to modeling ideal diodes and MOS transistors for MNA were presented. By employing the piecewise linearization technique, the operating area of each piecewise linearized device is divided into several regions and the characteristic of each device appears linear inside every region. The Katzenelson algorithm presented in [39] finds solutions of linearized circuits with voltage or current controlled resistive devices. This chapter presents the Katzenelson algorithm and introduces some minor modifications which can make the Katzenelson algorithm solve the circuit equations in MNA form with the piecewise linear device models presented in Chapter 2.

3.1 The Katzenelson Algorithm

The Katzenelson algorithm was developed for simulating circuits with piecewise linear resistive devices which contain many linear regions. Each region can be modeled by an independent source and a resistive device. For example, in the voltage controlled piecewise linear resistive device shown in Fig. 3.1, the working regions are established by the boundary voltage values .... V_{l-2}, V_{l-1}, V_l, $V_{l+!}$, V_{l+2}.... . In this device, the working region l has the V-I relationship formulated by

$$I = I_l + (V - V_l)\, g_l \tag{3.1}$$

or

$$I = g_l V + I_l - V_l\, g_l \tag{3.2}$$

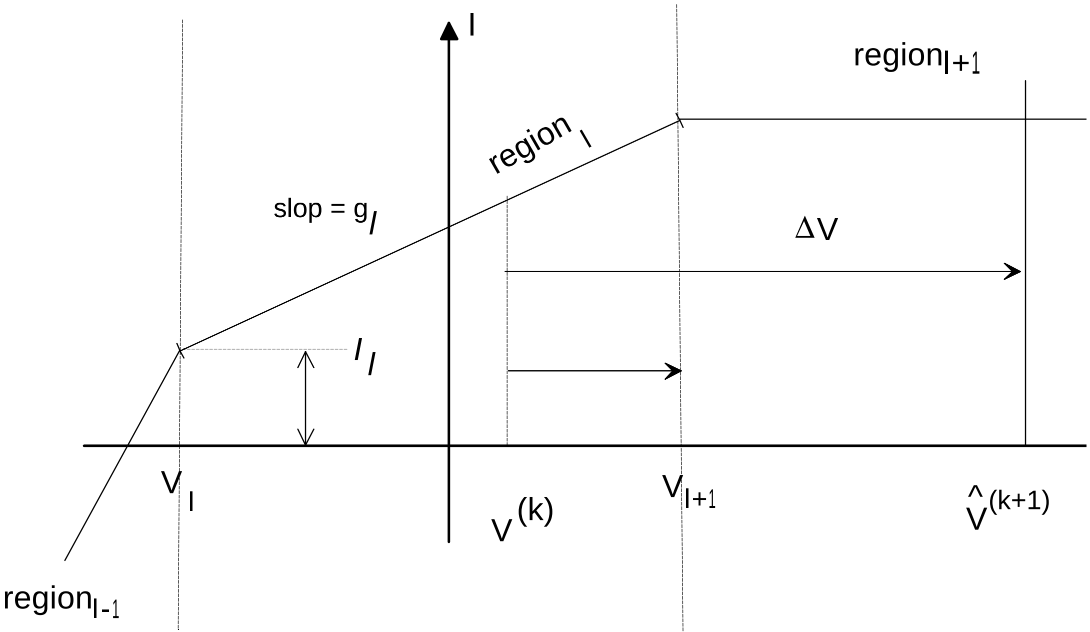


Fig. 3.1 A voltage controlled resistive device.

The Katzenelson algorithm adopts an iterative method. The iterative method begins with an initial guess. The initial guess assumes which region a piecewise linear resistive devices is in. After all piecewise linear resistive devices have their initial regions determined, system equations can be written in linear system form**:**

$$\mathbf{T}_l\, \mathbf{x}_l = \mathbf{w}_l + \mathbf{w} \qquad (3.3)$$

The subscript l indicates the region determined by the initial guess. The matrix $\boldsymbol{T_l}$ is the resistive matrix which contains the resistive information of devices in the assumed regions, e.g. g_l in equation (3.1). $\mathbf{x}_l$ is the solution vector. $\mathbf{w}_l$ is the piecewise linear excitation vector, e.g. I_l - $V_l\, g_l$ in equation (3.2). **w** is the source vector.

The Katzenelson algorithm does not solve $\mathbf{x}_l$ directly as in

$$\mathbf{x}_l = \mathbf{T}_l^{-1}\, (\mathbf{w}_l + \mathbf{w}), \qquad (3.4)$$

since $\mathbf{T}_l$ remains valid only in a limited region. The Katzenelson algorithm adopts an error vector

$$\mathbf{f} = \mathbf{T}_l\, \mathbf{x}_l - \mathbf{w}_l - \mathbf{w}, \qquad (3.5)$$

and the answer is obtained by reducing the error vector **f** to zero**.** For an iterative approach, equation (3.5) can be modified to

$$\mathbf{f}^k = \mathbf{T}_l^k\, \mathbf{x}_l^k - \mathbf{w}_l^k - \mathbf{w}, \qquad (3.6)$$

where the superscript k denotes the kth iteration. That is, $\mathbf{f}^k$ indicates the error in the kth iteration and when k =0, $\mathbf{f}^0$ is the error of the initial guess.

The update of equation (3.6) begins with finding solution of linear system in (3.7).

$$\mathbf{T}^k_l\, \Delta\, \mathbf{x}^k = -\mathbf{f}^k \qquad (3.7)$$

The attempted solution is found by (3.8)

$$\hat{\mathbf{x}}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \quad , \tag{3.8}$$

for which

$$\mathbf{0} = \mathbf{T}_l^k \, \hat{\mathbf{x}}^{k+1} - \mathbf{w}_l^k - \mathbf{w} \quad . \tag{3.9}$$

However, $\hat{\mathbf{x}}^{k+1}$ may not stay inside the linear region *l*. In this case, the update of the solution vector $\Delta\mathbf{x}^k$ can not be fully applied. A scaling factor *t* must be applied to constrain the new solution to lie inside the linear region *l*, reaching the boundary of one or more regions.

$$\mathbf{x}^{k+1} = \mathbf{x}^k + t^k \, \Delta\mathbf{x}^k \tag{3.10}$$

Equation (3.10) shows the scaling factor t^k makes the new solution $\mathbf{x}^{k+1}$ lie on the boundary between two linear regions denoted by *l* and *l*+1. In order to cross the boundary, the matrix and the excitation vector have to be replaced by $\mathbf{T}_{l+1}^{k+1}$ and $\mathbf{w}_{l+1}^{k+1}$. After the replacement, the iteration continues to change the error vector shown in equation (3.6), which can be easily evaluated using equation (3.11) or (3.12)

$$\mathbf{f}^{k+1} = \mathbf{T}_l^k \, \mathbf{x}_l^{k+1} - \mathbf{w}_l^k - \mathbf{w}, \tag{3.11}$$

$$\mathbf{f}^{k+1} = (1- t^k)\, \mathbf{f}^k \, . \tag{3.12}$$

The solution vector is updated using (3.10) until it reaches the scaling factor t^k =1. Equation (3.12) is satisfied, provided that the device is continuous at the region's boundaries.

The purpose of employing the scaling vector is to constrain the updated solution inside the linear region *l*. One example of evaluating t_l is shown by the voltage controlled resistive device whose portion of *V-I* characteristic is illustrated in Fig. 3.1. While at the *k*th iteration, it has the operating point $V^{(k)}$ ($V^{(k)}$ is one element of the solution vector $\mathbf{x}_l^k$) located at linear region *l*. If the attempted amendment calculated by (3.7) is ΔV (ΔV is the corresponding element of $\Delta\mathbf{x}^{\mathbf{k}}$), then the scaling vector contributed by this device at the *k*th iteration is evaluated by:

$$t_l^{\ k} = \frac{V_{l+1} - V^{(k)}}{\Delta V} \quad , \tag{3.13}$$

where V_{l+1} is the boundary between region *l* and *l*+1 as illustrated by Fig. 3.1. t^k must be smaller than 1 to have the effect of iteration constraint. If there is more than one piecewise device, then there will be as many scaling factors. In that case, the smallest t_l will be adopted. For this reason, only one device is allowed to change its working region (state) at an iteration. The solution seeking procedure is finished when the error vector $\mathbf{f}^{\mathbf{k}}\mathbf{=0}$.

3.2 t_{min} in SAMOC

SAMOC employs the Katzenelson algorithm for simulating circuits which contain the piecewise linear devices. The piecewise linear devices supported by SAMOC are the ideal diodes and MOS transistors. Their models were presented in Chapter 2. This section discusses the determination of the initial guess, and the evaluation of the scaling factor contributed by the presentence of ideal diodes and MOS transistors.

3.2.1 Ideal diode

The Ideal diode model in SAMOC has 3 linear regions and is neither a voltage nor a current controlled device. If there is a way to determine the scaling factor t limiting the change inside each of the 3 regions, then employing the Katzenelson algorithm with the SAMOC ideal diode model is still feasible. While in the initial guess stage, an ideal diode is treated as an open switch and the voltage across the ideal diode V_D can be determined. The working area of an ideal diode is $V \in (E_2, E_1)$ and $I \in (-\infty , +\infty)$. The V-I characteristic of an ideal diode is illustrated in Fig. 2.6 and contains the turned on region, the cutoff region and the breakdown region. The determination of which region a diode operating point lies in is made by the voltage across the diode V_D. If V_D is greater than the turned on voltage E_1, then the diode is in the turned on region and it is modeled by an independent voltage source with voltage E_1. If V_D is smaller than the turned on voltage E_1 and greater than the breakdown voltage E_2, then the diode is in the cutoff region and modeled by an open switch. If V_D is smaller than the breakdown voltage E_2, then the diode is in the breakdown region and modeled by an independent voltage source with voltage E_2.

The scaling factor t of the SAMOC ideal diode model which limits the change at the boundary of regions, is state dependent. The formulas to evaluate t are:

1. In state 1, the diode is modeled by an opened ideal switch. That is $I_D = 0$, $\Delta I_D = 0$ and $V_D \in (E_2, E_1)$ and ΔV_D can be any value.

If $V_D + \Delta V_D > E_1$, then $$t = \frac{E_1 - V_D}{\Delta V_D} \quad . \qquad (3.14)$$

2. In state 2, the diode is modeled by an independent voltage source. That is $V_D = E_1$, $\Delta V_D = 0$ and $I_D \geq 0$ and ΔI_D can be any value.

If $I_D + \Delta I_D < 0$, then $$t = -\frac{I_D}{\Delta I_D} \quad . \qquad (3.15)$$

3. In state 3, the diode is modeled by an independent voltage source. That is $V_D = E_2$, $\Delta V_D = 0$ and $I_D \leq 0$ and ΔI_D can be any value.

If $I_D + \Delta I_D > 0$, then $$t = -\frac{I_D}{\Delta I_D} \quad . \qquad (3.16)$$

With scaling factor t evaluated by formulas (3.14-16), 0< t <1 and constrains the changing of the states of an ideal diode. The ideal diode has to stay at the boundaries at least for one iteration and change the device parameters. The boundaries are either $V_D = E_1$, $V_D = E_2$ or $I_D = 0$.

After the initial state is set and the scaling factor t is obtained, there are dynamic state transition rules for the ideal diode during the solution seeking procedure. In this procedure, each piecewise linear device has a chance to change its state. Fig. 3.2 illustrates the state transition rules of an ideal diode. In state 1, the diode is modeled by an opened switch, i.e. $I_D = 0$ and V_D can be of any value. If a diode was previously in state 1 and $|V_D - E_1| < \varepsilon$[3], then the state shifts from 1 to 2. If a diode is previously in state 1 and $|V_D - E_2| < \varepsilon$, then the state shifts from 1 to 3. In state 2, the diode is modeled by an independent voltage source; i.e., $V_D = E_1$ and I_D can be of any value.

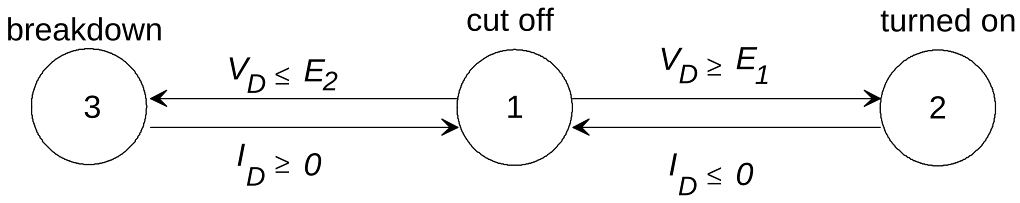


Fig. 3.2 Dynamic state transition diagram of an ideal diode.

If a diode is originally in state 2 and $|I_D - 0| < \varepsilon$, then the state shifts from 2 to 1. In state 3, the diode is also modeled by an independent voltage source; i.e., $V_D = E_2$ and I_D can be of any value. If a diode is originally in state 3 and $|I_D - 0| < \varepsilon$, then the state shifts from 3 to 1. Note that there is no direct transition from state 3 to 2 or from 2 to 3. The scaling factor t is formulated according to different states and constrains the transition.

3.2.2 MOS Transistors

The initial guess of the state of a MOS transistor assumes all MOS transistors are in the cutoff state at the beginning of the iterations. After the gate to source voltage, V_{GS}, and drain to source voltage V_{DS} are evaluated, the state of the MOS transistor can be determined by:

[3] ε is a relatively small numerical value. In SAMOC, $\varepsilon = 10^{-12}$.

If $V_{GS} < V_t$, then the MOS transistor is in state 1 (the cutoff region).

If $V_{GS} \geq V_t$ and V_{GS}-V_t < k V_{DS}, then the MOS transistor is in state 2 (the saturation region).

If $V_{GS} \geq V_t$ and V_{GS}-V_t > k V_{DS}, then the MOS transistor is in state 3 (the linear region).

V_t is the threshold voltage of the MOS transistor and

$$k = \frac{R_{\max} - R_{\min}}{g_m R_{\max} R_{\min}} \quad . \tag{3.17}$$

The SAMOC MOS transistor model is presented in voltage controlled resistive device. The current I_{DS} is a function of V_{DS} and V_{GS}; therefore, the evaluation of the scaling factor t is in the 2-dimensional V_{DS} and V_{GS} plain. The scaling factor t of the SAMOC MOS transistor model, which limits the change at the boundary of regions, is state dependent. The formulas which evaluate t are:

1. In state 1 (the cutoff region), the gate to source voltage, V_{GS}, must be less than V_t for NMOS and greater than V_t for PMOS. The transistor is modeled as a big resistor, $R_{\max}$, between drain and source and ΔV_{GS} can have any value. For NMOS, if $V_{GS} + \Delta\ V_{GS} > V_t$, then

$$t = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.18}$$

For PMOS, if $V_{GS} + \Delta\ V_{GS} < V_t$, then

$$t = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.18a}$$

2. In state 2 (the saturation region), the gate to source voltage $V_{GS} > V_t$ and V_{GS} -V_t< k V_{DS} for NMOS and $V_{GS} < V_t$ and V_{GS} -V_t > k V_{DS} for PMOS. The transistor is modeled as a large resistor, $R_{\max}$, in parallel with a voltage controlled current source with transconductance g_m between drain and source and Δ V_{GS} and Δ V_{DS} can be any value.

For NMOS, if V_{GS} + Δ V_{GS} < V_t, then

$$t_1 = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.19}$$

For PMOS, if V_{GS} + Δ V_{GS} > V_t, then

$$t_1 = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.19a}$$

If V_{GS} + Δ V_{GS} -V_t < k (V_{DS} + Δ V_{DS}) for NMOS or V_{GS} + Δ V_{GS} -V_t > k (V_{DS} + Δ V_{DS}) for PMOS, then

V_{GS} -V_t+ t_2 Δ V_{GS} = k (V_{DS} +t_2 Δ V_{DS}), (3.20)

hence,

$$t_2 = \frac{V_{GS} - V_t - kV_{DS}}{k\Delta V_{DS} - \Delta V_{GS}} \tag{3.21}$$

If conditions in both (3.19) and (3.21) are true, then t *is* the smaller one of t_1 and t_2.

3. In state 3 (the linear region), the gate to source voltage $V_{GS} > V_t$ and V_{GS}-$V_t > k\ V_{DS}$ for NMOS and $V_{GS} < V_t$ and V_{GS}-$V_t < k\ V_{DS}$ for PMOS. The transistor is modeled as a small resistor, $R_{\min}$, between drain and source and $\Delta\ V_{GS}$ and $\Delta\ V_{DS}$ can have any value.

For NMOS, if $V_{GS} + \Delta\ V_{GS} < V_t$, then

$$t_1 = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.22}$$

For PMOS, if $V_{GS} + \Delta\ V_{GS} > V_t$, then

$$t_1 = \frac{V_t - V_{GS}}{\Delta V_{GS}} \quad . \tag{3.22a}$$

If V_{GS} -$V_t + \Delta\ V_{GS} < k\ (V_{DS} + \Delta\ V_{DS})$ for NMOS and V_{GS} -$V_t + \Delta\ V_{GS} > k\ (V_{DS} + \Delta\ V_{DS})$, then

$$V_{GS} \text{ -} V_t + t_2\ \Delta\ V_{GS} = k\ (V_{DS} + t_2\ \Delta\ V_{DS}), \tag{3.23}$$

hence,

$$t_2 = \frac{V_{GS} - V_t - kV_{DS}}{k\Delta V_{DS} - \Delta V_{GS}} \tag{3.24}$$

If conditions in both (3.22) and (3.24) are true, then *t is* the smaller one of t_1 and t_2.

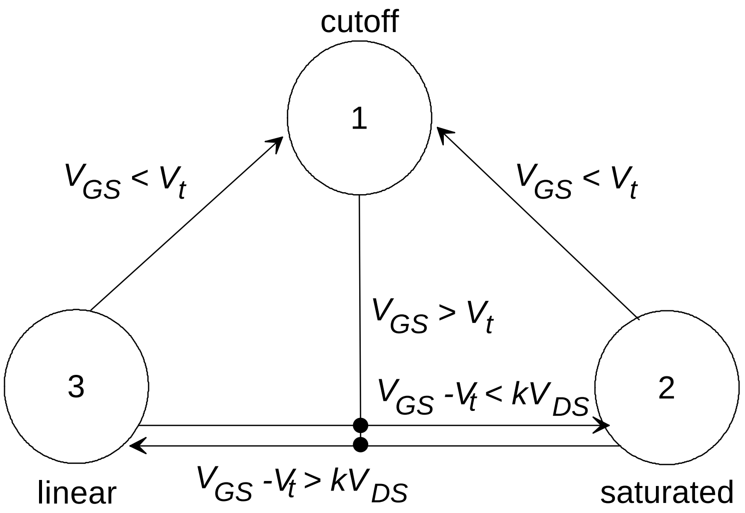


Fig. 3.3 Dynamic state transition diagram of a MOS transistor.

After the scaling factor *t* is obtained by formulas (3.18-24), SAMOC begins to update the states of the MOS transistors. The state transition diagram of the MOS transistors is illustrated in Fig. 3.3. A MOS transistor can switch from any state to any other state. Among the possible transitions, checking V_{GS} has higher priority. That is, if previously a MOS transistor is in state 2 and after an iteration the new V_{GS} is approaching the threshold voltage V_t, $| V_{GS} - V_t | < \varepsilon$, then SAMOC will set the MOS transistor to state 1 without checking the other condition. If $V_{GS} > V_t$ and $| V_{GS} - V_t - k V_{DS} | < \varepsilon$, then the new state will be 3. The same situation happens when the MOS transistor is in state 3. SAMOC checks $| V_{GS} - V_t | < \varepsilon$ to determine whether to shift to state 1 or not and then checks $| V_{GS} - V_t - k V_{DS} | < \varepsilon$ to determine whether to the shift to state 2 or not. On the other hand, if in the previous iteration the MOS transistor was in state 1 and the new V_{GS}

is approaching the threshold V_t, | V_{GS} -V_t | < ε, then SAMOC will compare V_{GS} - V_t and kV_{DS} to determine whether the MOS transistor will shift to state 2 or 3.

3.3 Simulation Examples of SAMOC

With the device models presented in Chapter 2 and the simulation techniques presented in Chapter 3, SAMOC can analyze DC solutions of MOS circuits. In order to test the models and the simulation techniques, some simple circuits are selected to be analyzed by SAMOC.

One of the most important circuit parts in analog CMOS circuit design is the differential amplifier. A differential amplifier is usually used in comparing two voltage signals. Typically, a differential amplifier has two inputs and its output is a function of the difference of two inputs. Most of the time, the output gain is designed to be large to emphasize the difference of two inputs.

One of the best differential amplifiers that can work well inside the allowed range is the wide-range transconductance amplifier presented in [45]. The schematic of the wide-range transconductance amplifier which is constructed by 4 PMOS and 5 NMOS transistors, is shown in Fig. 3.4.

By adding some extra code, it is possible to monitor the state change details while SAMOC determines the DC solution of a circuit. Table 3.1 shows the state changing detail of 9 MOS transistors in the wide-range transconductance amplifier shown in Fig. 3.4 with V+ = 1.05V and V- = 1.00V.

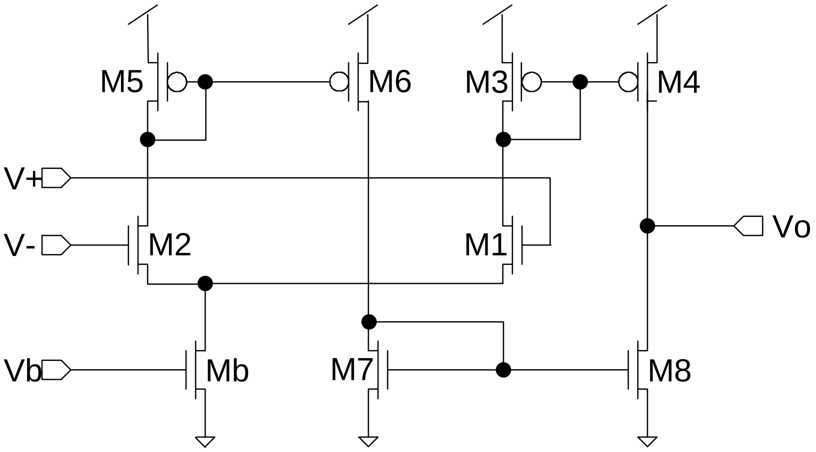


Fig. 3.4 Schematic of the wide-range transconductance amplifier.

The first column of the Table 3.1 is the iteration index. SAMOC's solution seeking algorithm discoveries that the solution in the 7th and 8th steps confirm the convergence of the algorithm. At the first step, all MOS transistors were set to be in state 1 (cutoff region). Step 2 is the preset region and each MOS transistor's state was directly set by the terminal voltages calculated by the states set by step 1. After step 2, the Katzenelson algorithm is applied. From step 2 to step 1, the minimum scaling factor $t_{\min}$ limits the state changes. Only one MOS transistor is allowed to change its state at each iteration step. Sometimes, if two MOS transistors have identical gate and drain, then these two MOS transistors can change states at the same time. Table 3.1 shows from step 2 to 3, M7 and M8 change states at the same step. After step 3, only one MOS transistor changes working region (state) at each iteration step. Fig. 3.5 shows the change of the output voltage (Vo) during the solution seeking procedure of SAMOC. Note that there is no state change from step 7 to 8 in Table 3.1 and there is no numerical value change in the plot presented in Fig. 3.6, either. Vo is low and according to V+ =1.05V and V- =1.00V, the answer is acceptable.

Table 3.1 State transition table

	M5	M6	M3	M4	M2	Mb	M7	M1	M8	remarks
1	1	1	1	1	1	1	1	1	1	initial guess
2	2	2	2	2	1	2	1	1	1	preset
3	2	2	2	2	1	2	2	1	2	M7, M8 changed
4	2	2	2	2	1	3	2	1	2	Mb changed
5	2	2	2	2	2	3	2	1	2	M2 changed
6	2	2	2	2	2	3	2	1	3	M8 changed
7	2	2	2	2	2	3	2	2	3	M1 changed
8	2	2	2	2	2	3	2	2	3	no change, **f** is small, iteration stops

3.4 Comparison with SPICE Simulation via DC Sweep

Fig 3.5 shows a likely acceptable result of the SAMOC simulation. One question may be asked at this point: "How much similar the result is to the SPICE simulation?" This section presents and compares the functional analyses of SAMOC and SPICE simulations. The best way to compare the DC analyses is to perform a linear DC sweep analysis of a circuit by both SAMOC and SPICE simulators. The analyzed circuit is still the wide-range transconductance amplifier illustrated in Fig. 3.4. The DC sweep analyses were performed 5 times with V- =0.5V, 1.5V, 2.5V, 3.5V and 4.5V respectively and V+ values are swept from 0V to 5V. The SPICE simulation was performed by PSPICE® version 8.0 of MicroSim®. Both SAMOC and PSPICE were executed on the same computer with the Microsoft® Windows95® operating system, 64 Megabytes of SDRAM and a Cyrix® 6x86MX® PR200 CPU.

Fig. 3.6 shows the PSPICE DC sweep results obtained by using level-3 model. All PMOS transistors have device geometry specified by w = 6u and l = 2u. All NMOS transistors have w = 2u and l = 2u. The values marked on the lines are the values of V-.

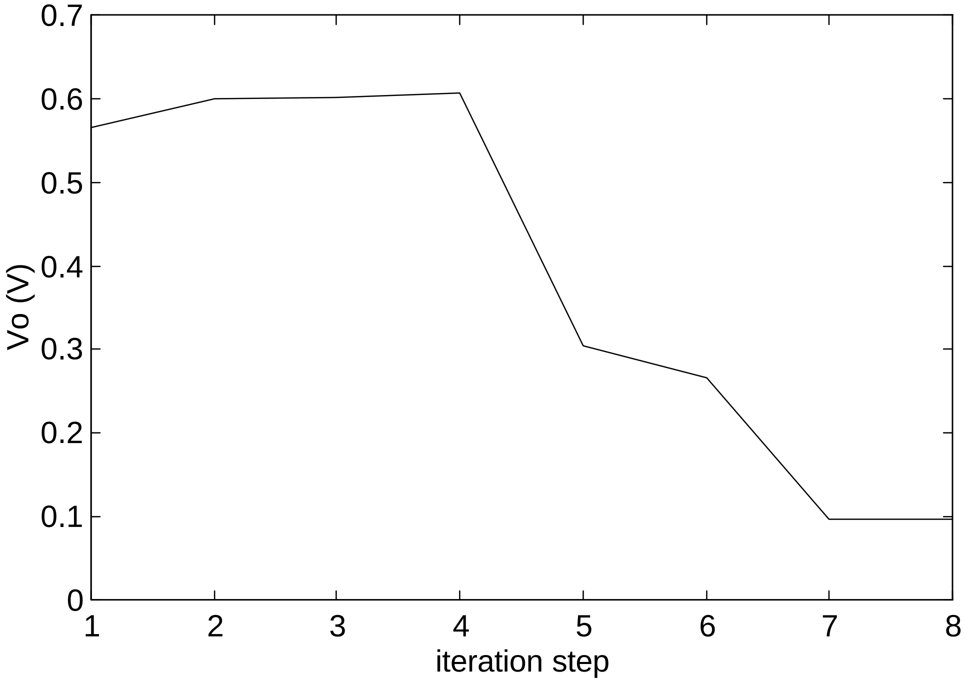


Fig. 3.5 The output voltage as a function of iteration step index.

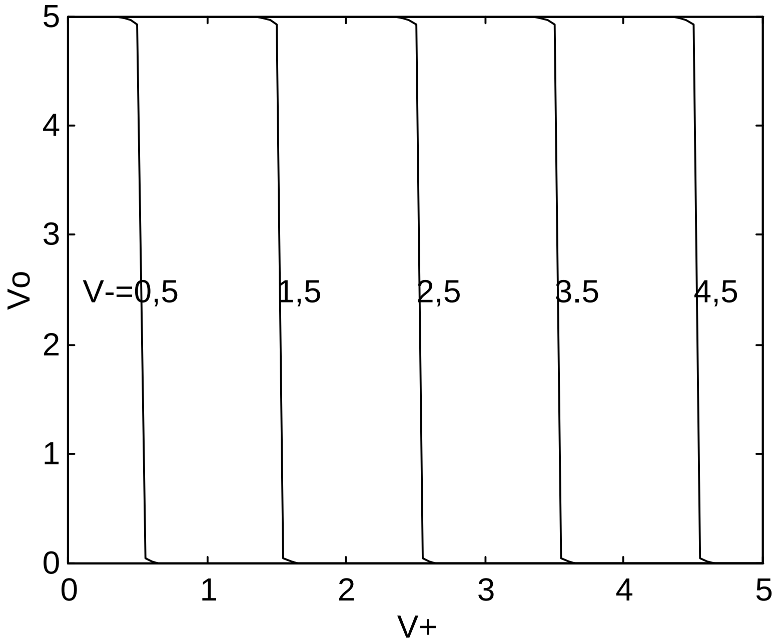


Fig. 3.6 PSPICE DC sweep

Fig. 3.7 shows the DC sweep analyses obtained by SAMOC simulation. Since the sophisticated DC sweep algorithm has not yet been built in SAMOC, for functional verification SAMOC calls the same DC analysis subroutine and changes the V+ values to record the result. Fig. 3.7 shows that the DC sweep results approach the SPICE results. The big difference happens when V- =0.5V and V+ is smaller than 0.7V. The difference may be caused by the SAMOC MOS models does not take subthreshold behavior into account.

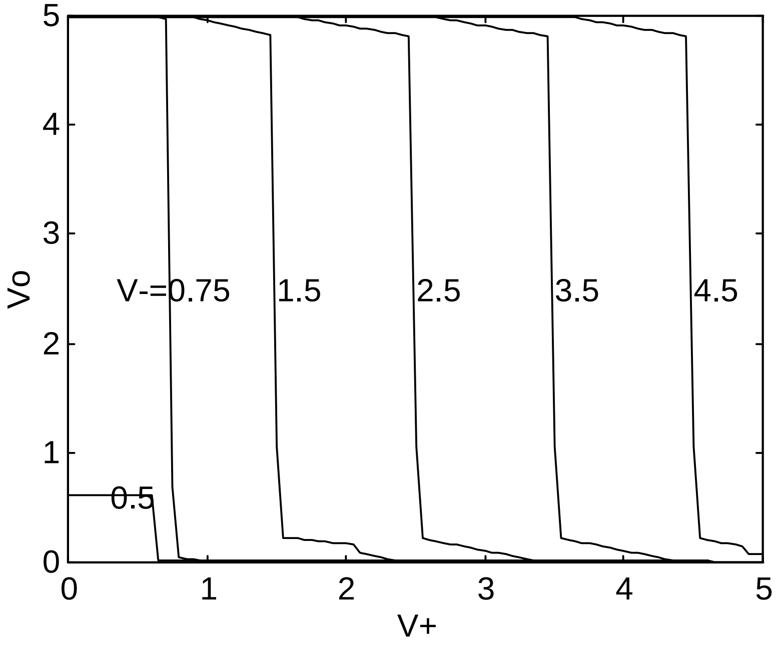


Fig. 3.7 SAMOC DC sweep

3.5 Summary

This chapter presented the modification of the Katzenelson algorithm and piecewise linear approach to MOS and ideal diode circuit simulation. The Katzenelson algorithm adopts an error correcting solution seeking routine. Each piecewise linearized device that needs to change working region would contribute a scaling factor *t* < 1 to limit the error correcting vector. The Katzenelson algorithm chooses the smallest scaling factor *t*, so that most of the time, only one piecewise linearized device is allowed to change its working region (state). After the chosen one changes its state and model parameters, the new circuit equation can change the state of another piecewise linear device. The iteration ends when no piecewise linear device changes states and the error vector **f** becomes very small.

SAMOC supports piecewise linear models for ideal diodes and MOS transistors. There are several main procedures for the solution seeking algorithm:

1. Both categories of piecewise linear devices are set to the cutoff region for determining the initial guess.
2. According to the initial guess, all piecewise linear devices are directly set to the new states.
3. New circuit equations are formulated by following the state information of each piecewise linear device. The desired correcting vector is evaluated.
4. Scaling factor *t* is calculated by the piecewise linear regions' boundary information and the desired correcting vector for each piecewise linear device.
5. The smallest *t* is selected to update the answer.
6. Check if there is any piecewise linear device near the boundary of another working region. If there is, change the working region, then go to 3. If there is none and error vector **f** is small, then a solution vector is found.

In addition to the modification of the Katzenelson algorithm, this chapter also presented analysis details about states (working regions) during the solution seeking procedure of a MOS circuit. Most of the time, only one piecewise linear device's state changes during one iteration thanks to the t_{min} restriction. Functional verification and comparison with SPICE simulation were also presented. In the DC sweep analysis example, SAMOC simulation showed the absence of subthreshold concerning the MOS transistors causes the biggest difference with SPICE simulation.

Chapter 4

SAMOC SIMULATION IN *Q-V* REALM

The SAMOC simulation presented in Chapters 2 and 3 is based on resistive network analysis. The device models focused on the current-voltage (*I*-*V*) relationship and the circuit formulation algorithm, while the modified nodal analysis (MNA) is based on Kirchhoff's current law (KCL). This chapter presents simulations of another category of circuit design - switched-capacitor (SC) networks. SC networks, which are built with CMOS transistors and capacitors, can be highly integrated and fit today's VLSI technologies.

Unlike resistive networks which rely on continuos time current levels to transfer and to transform energy, SC networks rely on charge transfers between capacitors during specified clock periods and remain steady at other times. SC networks hold information in the form of energy stored in capacitors. While SC networks are in steady state and holding the charges between capacitors, they do not consume any energy. SC networks can be built with CMOS transistors and capacitors. SC networks can be looked upon as a design category to implement low power information processing systems or subsystems. The major application of SC networks is in implementing active filters, charge pumps and analog neural network circuit designs.

Since there is no steady current in SC networks, the allowed circuit components in SC networks are ideal capacitors, ideal switches, ideal independent sources, ideal voltage-controlled voltage sources and ideal operational amplifiers (OPAMPs). SAMOC contains a simulation engine for simulating resistive networks in the *I*-*V* realm, which was shown in Chapter 2. The same simulation engine can be used for *Q*-*V* realm simulation, provided some physical theories and proper modification from *I*-*V* realm analysis techniques are considered.

4.1 *Q*-*V* Realm Analysis

Since SC networks do not contain any resistive device to transfer a voltage signal into a current signal, KCL is no longer as useful as in resistive networks. Although KCL can not be used, *conservation of charge* still stands at each node in SC networks. For this reason, analyzing SC networks, one can apply charge conservation equations as the counterpart of KCL equations used in analyzing resistive networks. While the charge is used instead of current, the simulation territory moves from *I*-*V* realm to *Q*-*V* realm.

The most important component in SC networks is the capacitor. A capacitor which has two terminals can transfer a voltage signal into a charge variable. The relationship between charge Q stored inside the capacitor C and the voltage V between two terminals of a linear capacitor is

$$C = \frac{Q}{V} \tag{4.1}$$

A charge Q can be stored in a node incident only to capacitors. The charge Q can be evaluated algebraically. For a node which is only connected to capacitors, such as node a in Fig. 4.1, the charge stored in node a, Q_a, can be evaluated by

$$Q_a = \sum_{i=1}^{n} [(V_a - V_i) \times C_i] \tag{4.2}$$

Voltage V_a may be changed by varying any of the voltages V_i in Fig. 4.1, because Q_a must remain the same according to *conservation of charge* Q_a will remain the same, unless a is connected to a voltage source or the ground with a resistive device.

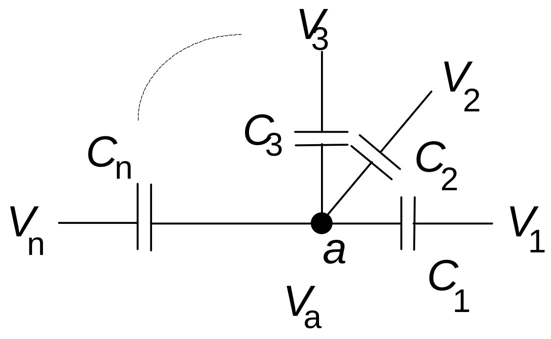


Fig. 4.1 A node "*a*" incident only to capacitors.

In an SC network, a voltage like V_a in Fig. 4.1 can be changed by changes of voltages incident nodes. For example, V_a can be changed by varying any of V_i, $i=1,2,\ldots n$

shown in Fig. 4.1 in order to keep Q_a conserved. The new voltage values of V_i, *i=1,2, .. n* are the controllable variables which are set up by the circuit designers by hard wiring, switching or the outputs of the other part of the circuit in order to store, manipulate or process the circuit data. The new voltages are denoted by V_i', *i= a, 1,2, .. n.* In this scenario, Q_a remains the same according to the *conservation of charge* and an equation based on this law can be used to evaluate V_a':

$$Q_a = \sum_{i=1}^{n}[(V_a - V_i) \times C_i] = \sum_{i=1}^{n}[(V_a' - V'_i) \times C_i] \quad . \tag{4.3}$$

Since V_i, *i=1,2, .. n* and V_a are know and V_i', *i= a, 1,2, .. n* are controllable or can be evaluated by other analyses, the altered voltage V_a' can be evaluated by (4.3).

The other method to change voltages between capacitors like V_a in Fig. 4.1 is to connect two nodes by an ideal switch as shown in Fig. 4.2. Before the switch is closed, Q_a can be evaluated by (4.2) and Q_b can be evaluated by (4.4), which is

$$Q_b = \sum_{i=n+1}^{n+m}[(V_b - V_i) \times C_i] \quad . \tag{4.4}$$

After the ideal switch is closed, $V = V_a = V_b$ and the new equation becomes

$$Q = Q_a + Q_b = \sum_{i=1}^{n+m}[(V - V_i) \times C_i] \quad . \tag{4.5}$$

The new total charge Q can be estimated by (4.2) and (4.4), and the new node voltage can be estimated by (4.5).

The SC network analysis can be performed by formulating charge conservation equations such as (4.2 - 4, 5) at each node. Nodal analysis can be adopted in the SC network simulation. SC network designers usually use two or more phase clocks to control the switches, either assigning voltage to some nodes such as in (4.3) or connecting two nodes such as in (4.5) alternately. SAMOC can formulate SC networks by MNA algorithm and estimate the new voltage value at each node before and after any switching event in the simulated SC network. Since most of the allowed devices in SC network design are linear, the solution vector can be obtained by a direct method such as Gaussian elimination.

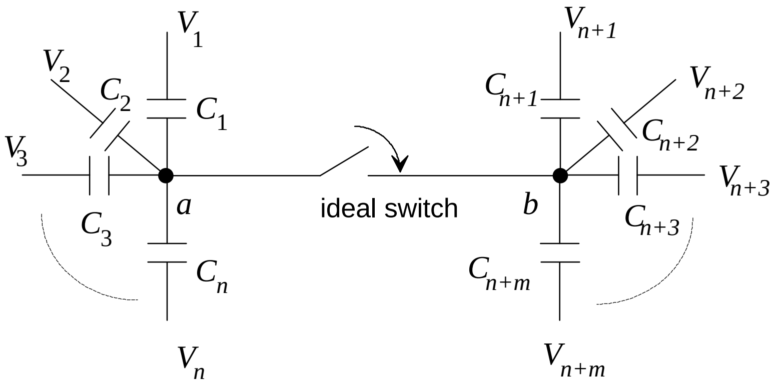


Fig. 4.2 Connecting two nodes via an ideal switch

4.2 MNA for SC Networks and Device Stamps in *Q-V* Analysis

The *Q-V* realm MNA of SC network can written in the matrix form as

$$\boldsymbol{C}\,\boldsymbol{V} = \boldsymbol{W}, \tag{4.6}$$

where ***C*** is the modified capacitive matrix, ***V*** is the solution vector and ***W*** is the excitation vector. Similar to MNA in *I*-*V* realm, MNA *Q*-*V* realm is formulated by filling device stamps into ***C*** according to the assigned nodes, device parameters and device types. The following subsections introduce the device stamps for different devices used to construct SC networks.

4.2.1 Capacitor

A capacitor is a charge storage device. If two terminal voltages of a capacitor C shown in Fig. 4.3 are V_j and $V_{j'}$, then the charges stored at the terminal *j* and *j'* are

$$Q_j = C\,(\,V_j - V_{j'}) \tag{4.7}$$

$$Q_{j'} = C\,(\,V_{j'} - V_j) \tag{4.8}$$

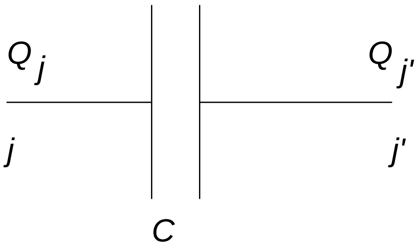


Fig. 4.3 A capacitor *C*.

According to *conservation of charge*, Q_j and $Q_{j'}$ contributed by *C* must remain unchanged no matter the change of V_j or $V_{j'}$. The device stamp of the capacitor *C* is shown in (4.9).

$$\begin{array}{ccc} & j & j' \\ j & C & -C \\ j' & -C & C \end{array} \qquad \overset{\text{excitation vector}}{\begin{bmatrix} Q_j \\ Q_{j'} \end{bmatrix}} , \qquad (4.9)$$

where Q_j and $Q_{j'}$ are evaluated by (4.7-8). The initial values of V_j and $V_{j'}$ can either be assigned to zero or specified by setting the initial condition command `.ic` as in SPICE, for example:

```
.ic V(j) = 5V V(j’) = 2.3V
```

The description format of a capacitor is

`C_name j j’ size_of_capacitor` [in Farads]

4.2.2 Ideal OPAMP

The ideal operational amplifiers (OPAMP) which have infinite input impedance and gain and zero output impedance are not directly supported by SPICE. Ideal OPAMPs are frequently used in SC network design. It is necessary to define an ideal OPAMP as a new device type in SAMOC SC network simulation. The circuit symbol of an ideal OPAMP is illustrated in Fig. 4.4. Circuit designers should pay attention to using the ideal OPAMP, because of the high gain of the amplifier. There must be a capacitor connecting *j’* and *k* to form a negative feedback loop to make the OPAMP work properly.

Vlach and Singhal [58] show the circuit stamps of the OPAMP, expressed by (4.10) with the node notation shown in Fig. 4.4. Besides 3 terminals represented by *j, j’* and *k*, the ideal OPAMP modeling requires one more variable marked by m’ for charge supplied by the OPAMP.

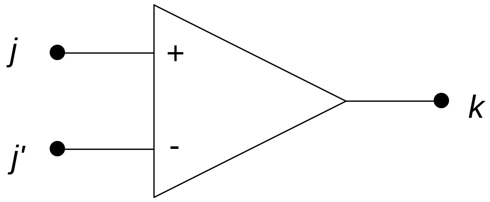


Fig. 4.4 The circuit symbol of an ideal OPAMP

$$\begin{array}{c|cccc|}
 & j & j' & k & m' \\
\hline
j & & & & \\
j' & & & & \\
k & & & & 1 \\
m' & 1 & -1 & &
\end{array} . \qquad (4.10)$$

SAMOC defines a new symbol "O" for OPAMP in *Q-V* realm MNA analysis. The description format for an ideal OPAMP is:

```
O_name j j' k
```

The device class used to represented the OPAMP is named `COpamp` in SAMOC. This class is derived from `C2VTerm` and one additional pointer for the third node is added to the existing *class*.

4.2.3 Ideal Switch

Ideal switches which are used to alter the circuit topology in circuit networks are essential to SC networks for changing voltage values. The model of ideal switches in *I-V* realm analysis with infinite resistance while it is closed and zero resistance while it is open, was presented in section 2.3.1. In *Q*-*V* realm analysis for SC networks, the model is exactly the same. The only difference between the *I*-*V* model and the *Q*-*V* realm model is that the (m+1)th variable in the solution vector represent the current in the *I*-*V* realm and transferred charge in the *Q*-*V* realm.

4.2.3 Ideal Diode

An ideal diode is the only nonlinear device allowed in SC network design. The ideal diode model in the *Q*-*V* realm is different from the one used in the *I*-*V* realm. In the *Q*-*V* realm analysis of SAMOC, the device stamp of each ideal diode is filled in by an ideal switch. In simulation of SC network with ideal diodes, the MNA circuit equations expressed by (4.6) have to be solved twice. At the first analysis, each ideal diode is treated as an open circuit. Then, according to the solution obtained at the first analysis, SAMOC checks the voltages of the two terminals of each ideal diode. If a diode in Fig. 2.7 has $V(\mathrm{n1}) > V(\mathrm{n2})$, then that diode is replaced by a closed circuit, otherwise that diode remains open. If any ideal diode shifted from an open circuit to a closed circuit, the second analysis of (4.6) has to be performed.

4.2.4 Ideal Voltage Source and Ideal Voltage-Controlled Voltage Source

An ideal voltage source in *Q*-*V* realm analysis plays a similar role to the independent voltage source in *I*-*V* realm analysis. An ideal voltage source can directly assign a voltage value to a node regardless how much charge was accumulated in that node. The device stamp is the same as the independent voltage source in *I-V* realm analysis.

The voltage-controlled voltage source is a four terminal device with infinite input impedance and zero output impedance. The OPAMP presented in section 4.2.2 is a special case of voltage-controlled voltage source with output gain equal to infinity. The model of voltage-controlled voltage source in *Q-V* realm analysis is the same as the one which is used in *I-V* realm analysis.

4.3 Simulation Examples

The device stamps presented in section 4.2 can be verified by simulation of simple SC networks. Two simple SC network simulation examples are presented in this section.

4.3.1 SC Integrator

The switched-capacitor integrator illustrated in Fig. 4.5 is one of the simplest SC networks. The transfer function of the integrator is

$$H(s) = \frac{V_{out}}{V_{in}} = -f_c \frac{C_1}{C_2} \frac{1}{s} \tag{4.11}$$

where f_c is the switching frequency.

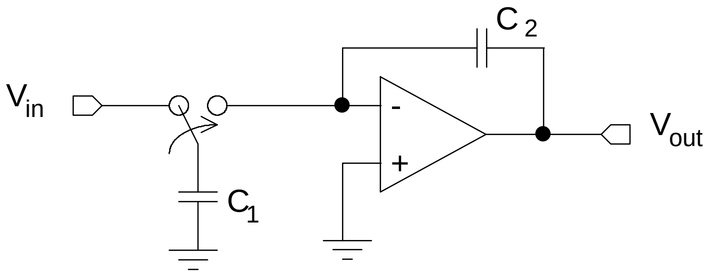
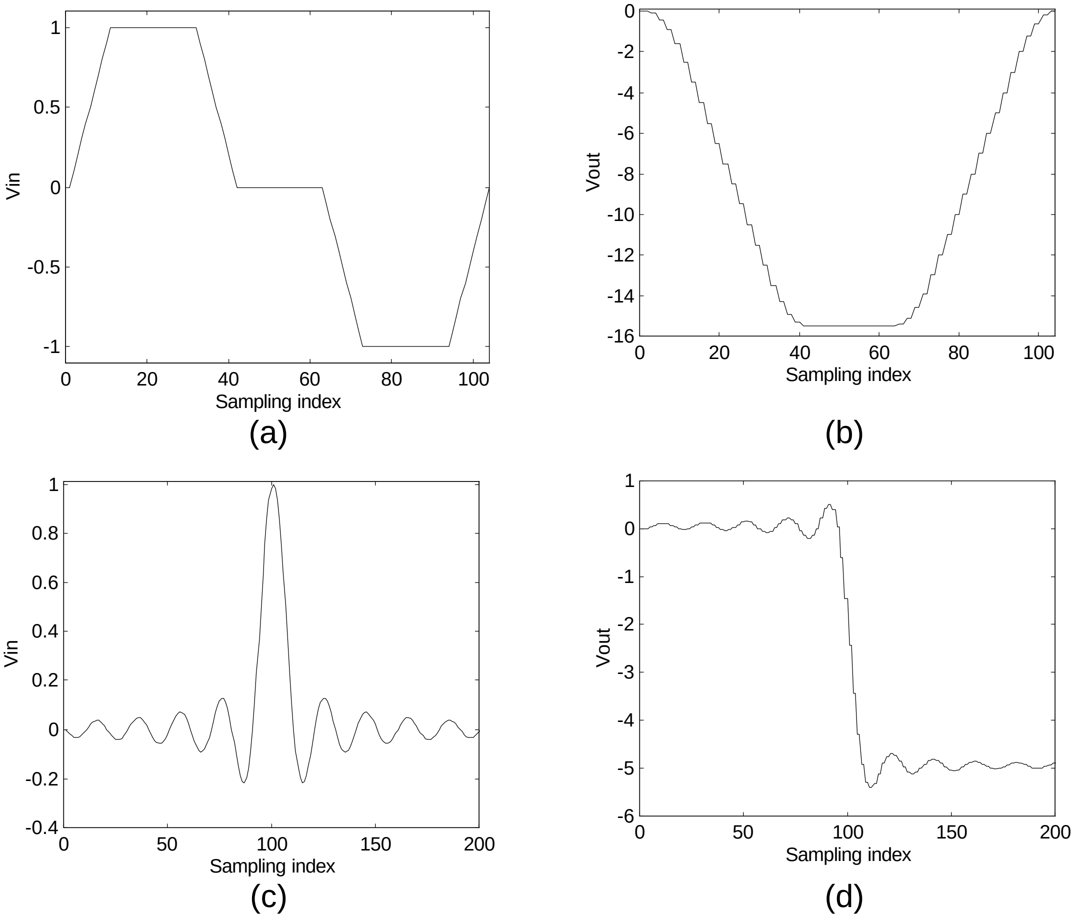


Fig. 4.5 A switch capacitor integrator

SAMOC simulation of an SC network can not give a continuous time signal of V_{in} in Fig. 4.5, since all events in a computer program are discrete in time. Therefore, SAMOC can describe V_{in} as a series of sampled data and obtain sampled data of V_{out}. By way of observing waveforms of V_{in} and V_{out}, it is possible to verify if the integrator works as expected. Fig. 4.6 shows the V_{in} and V_{out} plots of the SAMOC switch-capacitor simulation. Sampled data plotted in Fig. 4.6 (a) and (c) were fed into V_{in} and output results for V_{out} are plotted in Fig. 4.6 (b) and (d) respectively. Regardless of the switching frequency and the capacitor ratio in (4.11), the plots in Fig. 4.6 (b) and (d) look like the negative integration results of the plots in Fig. 4.6 (a) and (c) respectively.



(a)

(b)

(c)

(d)

Fig. 4.6 The V_{in} and V_{out} plots of SC simulation

4.3.2 SC Implementation of a Second-Order Bandpass Filter

A major application of SC networks is to implement filters. Since an SC circuit can be fabricated by CMOS technology in a single chip, SC implementation of filters can dramatically decrease the filter size which was traditionally caused by coil inductors or later by large resistance area in active R-C circuits. SAMOC is not equipped with a frequency domain analysis routine. However, *Matlab* can generate and analyze signals for filter analysis. The estimation of a transfer function of SC filter can be done by feeding a white noise signal and calculating the ratio of power spectral density of the output and input signals. That is, the power spectral density of the input white noise is

$$V_{in}\ (s) = 1 \qquad (4.12)$$

so that the transfer function

$$H(s) = V_{out}(s) \,/\, V_{in}(s) = V_{out}(s). \qquad (4.13)$$

From (4.13), the transfer function of a filter can be obtained by observing the power spectral density of the output signal while a white noise signal is the input. Since this estimation is a statistical result, the larger the sampling space is the more objective the results would be. For this reason, a very long series of random input is used.

SAMOC can neither generate white noise signals nor estimate the power spectral density of signals. Therefore, the signal generation, spectrum estimation and data visualization are finished by using *Matlab*.

A bandpass SC filter containing 8 ideal switches, 4 capacitors and one OPAMP is illustrated in Fig. 4.7. It is a SC realization of an active RC filter with a transfer function provided that $f_c >> f$, where $s = 2\pi f$.

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{-f_c(C_a / C_1)s}{s^2 + sf_c(C_b / C_1 + C_b / C_2) + f_c^{\,2}(C_a / C_1)(C_b / C_2)} \quad . \qquad (4.14)$$

SAMOC simulation of SC networks can be performed by feeding a series of signals into V_{in} and record the output from V_{out}.

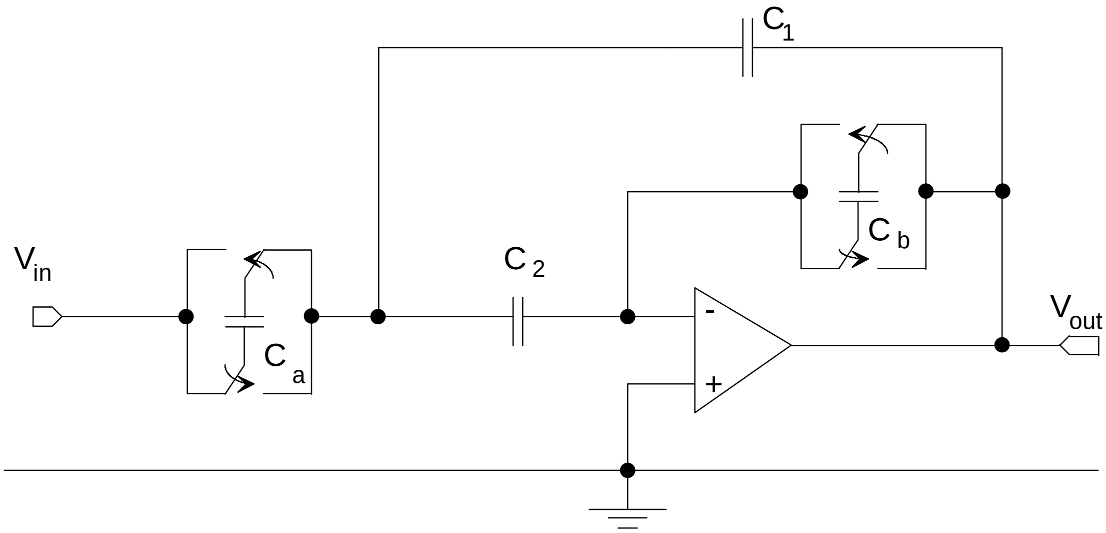
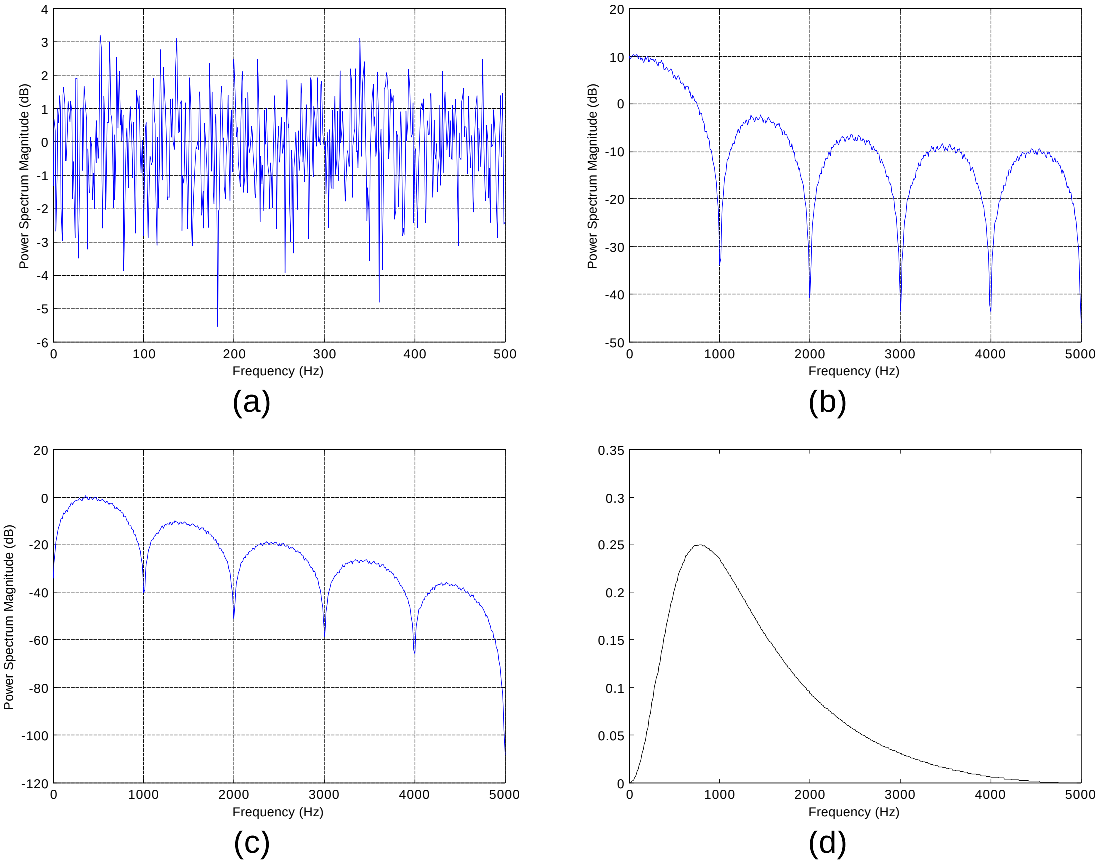


Fig. 4.7 An SC implementation of a second-order bandpass filter.

A series of signals contains 10,000 random values created by the *Matlab* function `randn()` are fed into V_{in} and the output V_{out} evaluated by SAMOC is recorded and sent back to *Matlab*. The capacitors C_a and C_b are set to be 1n Farad, and capacitors C_1 and C_2 are set to be 4n Farad. The power spectral densities of both signals are estimated by the `psd()` function included in the *Matlab* signal processing toolbox. By assuming the sampling frequency of input signal is 1 kHz. The power spectral density (psd) of the input signal is plotted in Fig. 4.8 (a). The clock frequency is set to be 10 kHz to have a 10x oversampling. Fig. 4.8 (b) illustrates the psd of the input signal after 10x oversampling which contains 100,000 sampled data. These 100,000 sampled data were sent to the SC based bandpass filter. The psd of the filter output by SAMOC simulation is illustrated by Fig. 4.8(c). The power transfer function, $H(e^{j\omega})H^*(e^{j\omega})$, of the bandpass filter can were estimated by the ratio of power spectral densities of two signals, and is plotted in Fig. 4.8 (d).



(a)

(b)

(c)

(d)

Fig. 4.8 Power spectral densities of input and output signals, and the power transfer function.

By filling in selected filter parameters, the RC filter transfer function of the filter is

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{-10000s}{s^2 + 5000s + 6250000} \quad , \qquad (4.15)$$

and it can be visualized by `bode()` routine in *Matlab*, if $s = j\omega = j\,2\pi f$. On the other hand, the absolute value of SC filter transfer function, $|H(e^{j2\pi f})|$, with clock frequency f_c = 10 kHz can be obtained by taking square root of the values plotted in Fig. 4.8(d). Fig. 4.9 illustrates the $|H(2\pi f)|$ and $|H(e^{j2\pi f})|$. Note that, according to the SC filter approximation, $|H(e^{j2\pi f})|$ will be more similar to $|H(2\pi f)|$ by using larger clock frequency f_c, which also implies that oversampling rate will be higher.

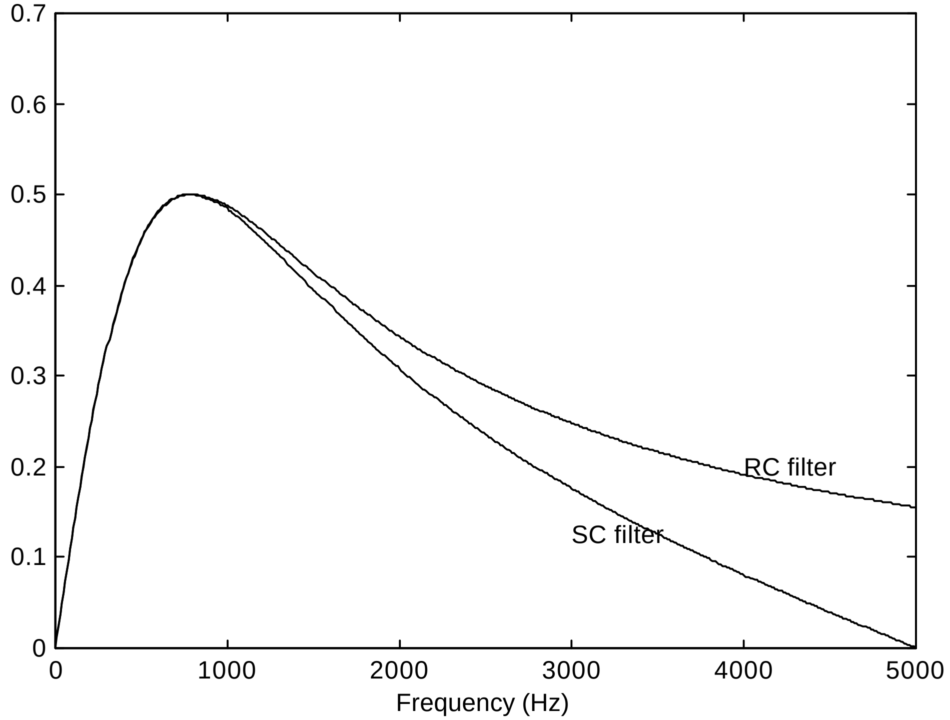


Fig. 4.9 Transfer functions of the second order bandpass filters.

4.4 Summary

Q-V realm analysis of SC networks is presented in this chapter. In designing SC networks, only capacitors, ideal diodes, OPAMP, ideal capacitors, ideal voltage sources and ideal voltage controlled voltage sources are allowed. There is no steady state current in *Q-V* realm analysis. By assuming ideal charge transfer between capacitors through ideal switches, MNA is quite adequate to perform *Q-V* realm analysis as well as *I-V* analysis. Since most of allowed devices are linear, the analysis in *Q-V* realm MNA can be done by direct method.

A new type of device, OPAMP, which is not directly supported by SPICE, is introduced in SAMOC simulation. The device stamp of an OPAMP, which was originally introduced by Vlach and Singhal for *I-V* realm analysis, is also valid in *Q-V* realm analysis.

The device stamps used in *I-V* realm analysis and in *Q-V* realm analysis are almost the same. A capacitor in *Q-V* realm analysis works virtually similar to a conductor in *I-V* realm analysis. The difference is that in *Q-V* realm analysis, the calculation of accumulated charge has to be done and added into the excitation vector of the circuit equations. As a result, an SC network is a circuit in which a previous state influences the next state. An ideal diode requires an additional solution of *Q-V* realm equations in order to decide the open/closed condition of every ideal diode.

Two simple SC network simulation examples are presented to verify the validity of the presented device stamps. One is an SC integrator and the input/output waveforms show the validity of *Q-V* realm device stamps for MNA. The other is an SC bandpass filter. In order to estimate the transfer function of the SC filter, an input signal consisted of a series of random numbers generated by *Matlab* is used as a test input signal of the filter. The output signal obtained from simulation is sent back to *Matlab*. The power transfer function of the example filter is estimated from the ratio of power spectral densities of output and input signals. The magnitude of the transfer function as a function of signal frequency is obtained by taking square root of the power transfer function. Although SAMOC does not include the power spectral density estimation subroutines, with the combination usage of *Matlab* signal processing tool box, the transfer function of the SC based filter can be verified via time series signal processing. The differences between a SC filter with a specified switching frequency and RC filter can be visualized. Two examples presented in this chapter were used to verify the device models and the *Q*-*V* realm circuit formulation techniques. The next chapter will introduce another application of SAMOC *Q*-*V* realm analysis: charge pump circuits.

Chapter 5

SAMOC APPLICATIONS I: DC-DC

SWITCHED-CAPACITOR BASED CHARGE PUMP CIRCUIT SIMULATIONS

A DC-DC charge pump circuit provides a DC voltage that is higher than the DC voltage of the power supply or a voltage of reverse polarity. In many applications - such as Power IC, continuous time filters, and EEPROM - voltages higher than the power supplies are frequently required. Increased voltage levels are obtained in a charge pump as a result of transferring charges to a capacitive load and do not involve amplifiers or transformers. For that reason, a charge pump is a device of choice in semiconductor technology where normal range of operating voltages is limited. Charge pumps usually operate at a high frequency level in order to increase their output power within a reasonable size of total capacitance used for charge transfer. This operating frequency may be adjusted by compensating for changes in the power requirements and saving the energy delivered to the charge pump.

5.1 SC Based Charge Pump Designs

Among many approaches to the charge pump design, switched-capacitor circuits charge pumps are very popular because they can be implemented on a single chip or on the same chip together with other components of an integrated system. There are three basic organizations of switched-capacitor (SC) DC-DC charge pumps: Dickson [63], Makowski [64] and Starzyk charge pumps. Three of the charge pump organizations push and store charge inside arrays of capacitors in order to reach a high voltage output. The Dickson charge pump requires a series of diodes. Makowski and Starzyk charge pumps are purely switched capacitor based designs which contain only capacitors and switches. All of them require two inverted and non-overlapped clocks to work.

According to different values of desired outputs, charge pumps can be cascaded to have higher voltage gain. That is, different numbers of cascaded stages of charge pump have different voltage gains. The voltage gain is a function of the number of stages. For Dickson pumps, the voltage gain is a linear function of number of stages. For an *n*-stage Dickson pump, the voltage gain A_v can be evaluated by (5.1).

$$A_v = n + 1 \tag{5.1}$$

For example, a 7-stage Dickson charge pump has voltage gain equal to 8. The schematic of a 7-stage Dickson charge pump with voltage gain equal to 8 is shown in Fig. 4.1. The charge pump contains two switches, 8 diodes and 7 capacitors marked by C and the capacitive load of the charge pump is modeled by a capacitor C_{load}. For each additional stage of Dickson charge pump, an ideal diode and a capacitor are needed.

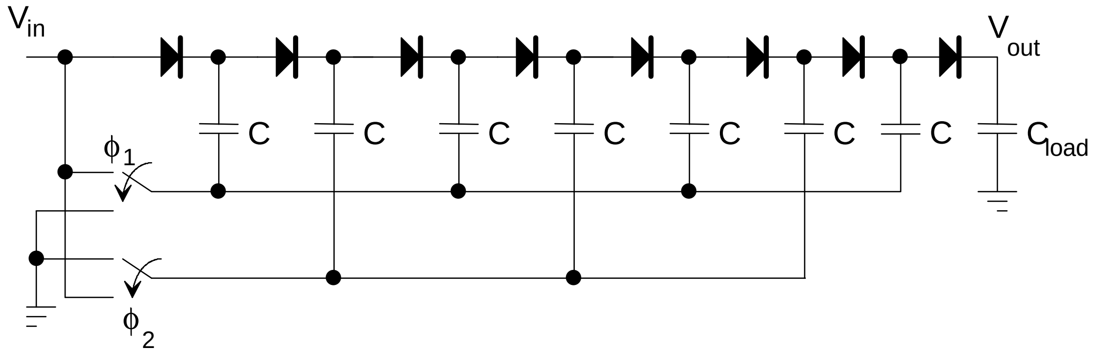


Fig. 5.1 An 8x Dickson charge pump

The voltage gain of an *n*-stage Makowski charge pump is the (*n*+1)*th* Fibonacci number. The Fibonacci sequence used is : 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233 ... . For instance, a 4-stage Makowski charge pump has a voltage gain equal to 8. The maximum transferred power of a charge pump design is influenced by the size of its capacitor. In addition, the number of required capacitors indicates the total required design area. The number of required capacitors is a direct index of the charge pumps' manufacturing costs. In all SC based charge pumps which require two phase clocks, Makowski's charge pump can reach the highest voltage gain with the least number of required capacitors. Fig. 5.2 illustrates the schematic of a 4-stage Makowski charge pump with input V_{in}, output V_{out}, where the capacitive load of the charge pump is modeled by C_{load}. The voltage gain of the displayed Makowski charge pump is equal to 8.

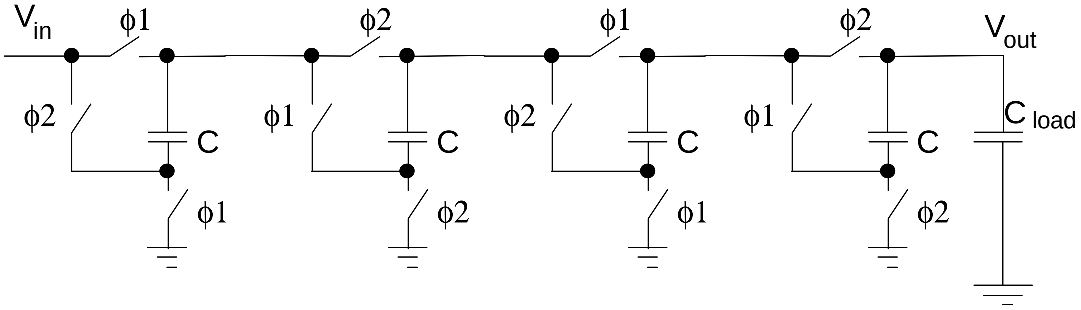


Fig. 5.2 An 8x Makowski charge pump

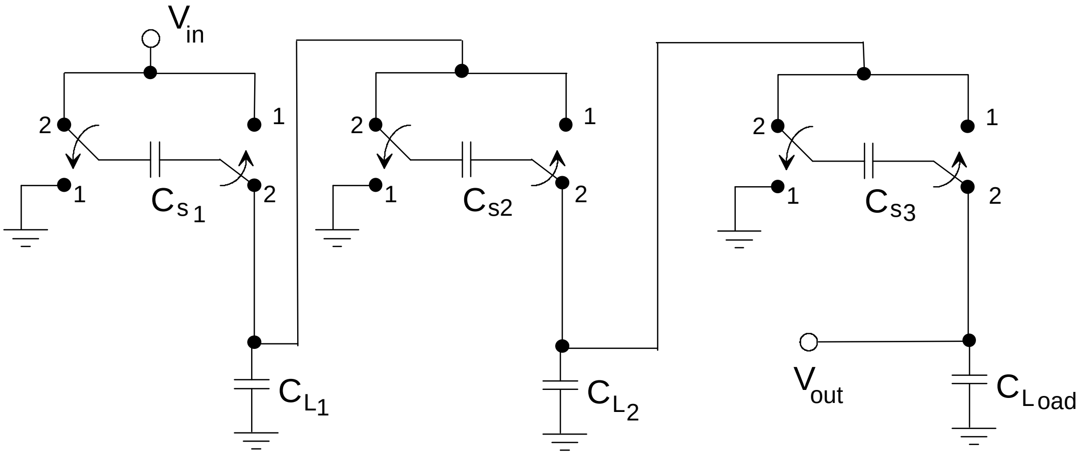


Fig. 5.3 An 8x TPVD Starzyk charge pump

Starzyk charge pumps are constructed by switched-capacitor voltage doublers. Each voltage doubler doubles the input voltage. By cascading *n* voltage doublers, the voltage gain is 2^n. For instance, a 3-stage Starzyk charge pump has a voltage gain equal to 8.

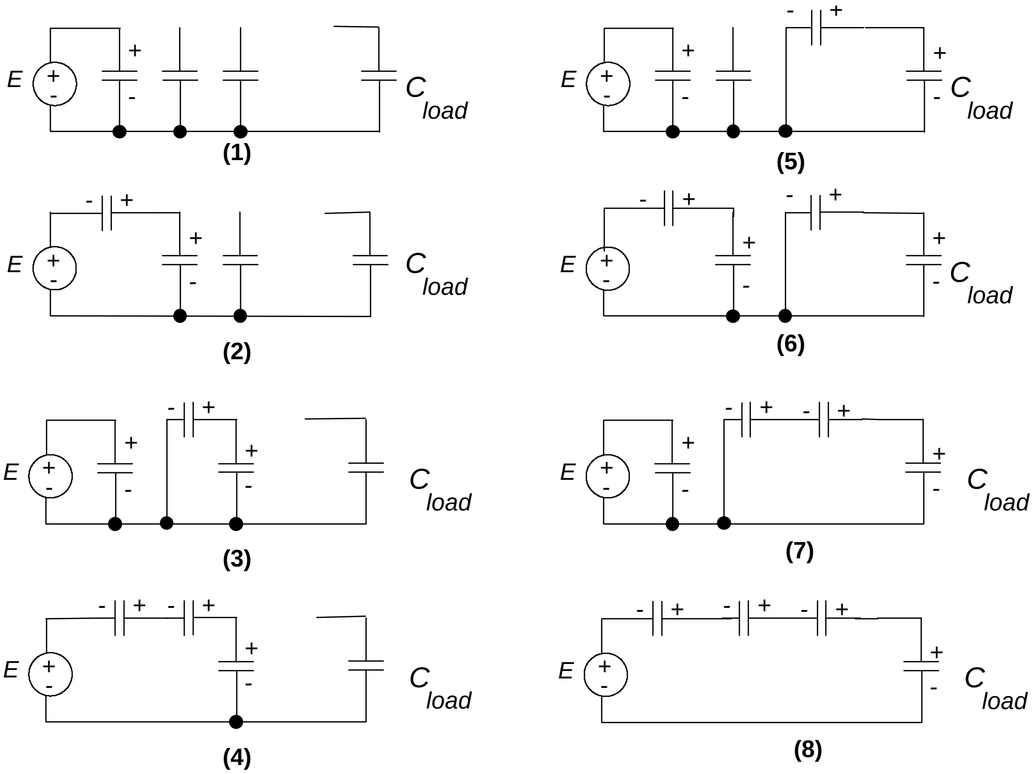


Fig. 5.4 An 8x MPVD Starzyk charge pump

There are two categories of Starzyk charge pump, one is based on the two phase voltage doublers (TPVD) and the other is based on the multiphase voltage doublers (MPVD). Fig. 5.3 shows the schematic of a 3-stage Starzyk TPVD charge pump. The voltage gain is 8. TPVD charge pump contains two way switches. While clock 1 is on, the switches connect to nodes marked by 1 in Fig. 5.3. While clock 2 is on, the switches connect to nodes marked by 2 in Fig. 5.3. The MPVD charge pump shown in Fig. 5.4

also has a voltage gain equal to 8. The schematic of the MPVD charge pump is the same as TPVD, but without C_{L1} and C_{L2}. The presented MPVD charge pump has 8 states. The transition of states must follow the order indicated on Fig. 5.4. The MPVD can be implemented by a finite state machine.

5.2 Time Domain Analysis

Three of four charge pump organizations, which contain diodes or ideal switches are controlled by two inverted clocks, were presented in section 5.2. It is exhausting to derive a symbolic input/output relationship in a mathematical format, especially when ideal diodes are involved in Dickson charge pump. The best way to estimate the behavior of charge pump circuits and determine the optimal circuit parameters is to use a computer simulator to analyze the circuits. The *Q*-*V* realm simulation engine of SAMOC can simulate switched-capacitor networks which contain three organizations of the charge pumps. The analysis of SC networks in SAMOC assumes that only the ideal switches and ideal capacitors are used in the simulated SC networks. That is, the switches do not dissipate charge which is caused by open circuit leakage current or the closed circuit resistance. In addition, it is assumed that all charge transfers are at the speed of light. The time domain analysis of an SC charge pump is accomplished by analyzing switch events one after another.

5.2.1 Comparing with SPICE simulation

SPICE simulation of the TPVD charge pump was performed in order to compare it with SAMOC for accuracy and computing efficiency. Since SPICE neither supports ideal floating capacitors nor ideal switches, additional resistors are added in order to make the SPICE simulation feasible. A 10^{10} Ω leakage resistor is added in parallel with each floating C_s capacitor in TPVD charge pump. The turned-on resistance (R_{on}) of all of the voltage control switches is 1 Ω, and the turned-off resistance (R_{off}) is 10^7 Ω. In SAMOC simulation, there is no leakage resistor of capacitor, and ideal switches have $R_{on} = 0$ Ω and $R_{off} = \infty$ Ω. Fig. 5.5 shows both the SAMOC and SPICE simulations. The clock period is 40ns. SPICE and SAMOC show similar results at the beginning of the simulation. The leakage resistance of capacitors and turned-off resistance of switches resulted in the pump reaching lower voltage value in SPICE simulation. The SPICE simulation was performed by MicroSim PSPICE® which works in Microsoft® Windows 95, the same platform as SAMOC. The two simulators were run on the same computer with the same operating system for comparison of timing efficiency. SPICE took 121.3 sec to simulate a three stage voltage doubler, while SAMOC required only 1.3 sec. This time difference, which indicated the high performance of SAMOC program, grows significantly larger if SPICE is run using full models of MOS switches. Full SPICE analysis with parasitic values extracted from a layout of designed pump has to be performed at the design stage of a silicon charge pump. However, this analysis is extremely costly to do for the type of investigation conducted in this work. For this reason, analysis using SAMOC is fully justified and facilitates study of the fundamental properties of the proposed pumps.

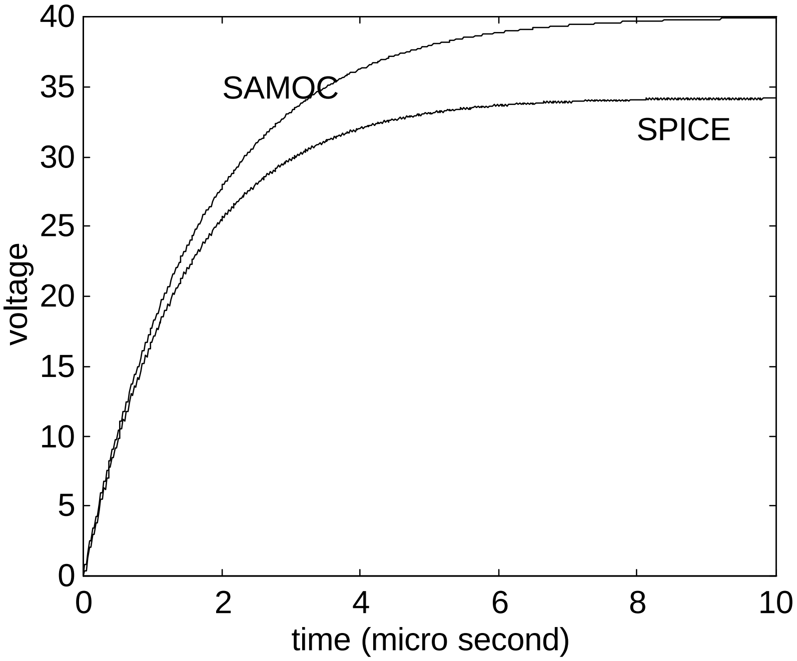


Fig. 5.5 Comparison of SPICE and SAMOC simulation

5.2.2 The SAMOC Simulation of SC Charge Pumps

The time domain SAMOC simulation results of the output waveforms of the compared charge pumps as a function of number of iterations are illustrated in Fig. 5.6. The fastest is the Dickson charge pump which needs about 100 switching events to reach 99% of the 8x output. The TPVD charge pump is slower and requires more than 400 iterations to reach 99% of the final voltage gain. The output of MPVD is updated after 8 iterations and its rise time is bit shorter than that of TPVD. The Makowski charge pump needs about 250 iterations to reach 99% of the final voltage gain. These large numbers of iterations translate directly into the number of required clock cycles and are indicative of the energy transfer efficiency of the analyzed circuits. However, they should not be of a significant concern to most of the applications for a charge pump implemented in a modern IC technology, where a maximum frequency of operation may reach a level of several hundred megahertz.

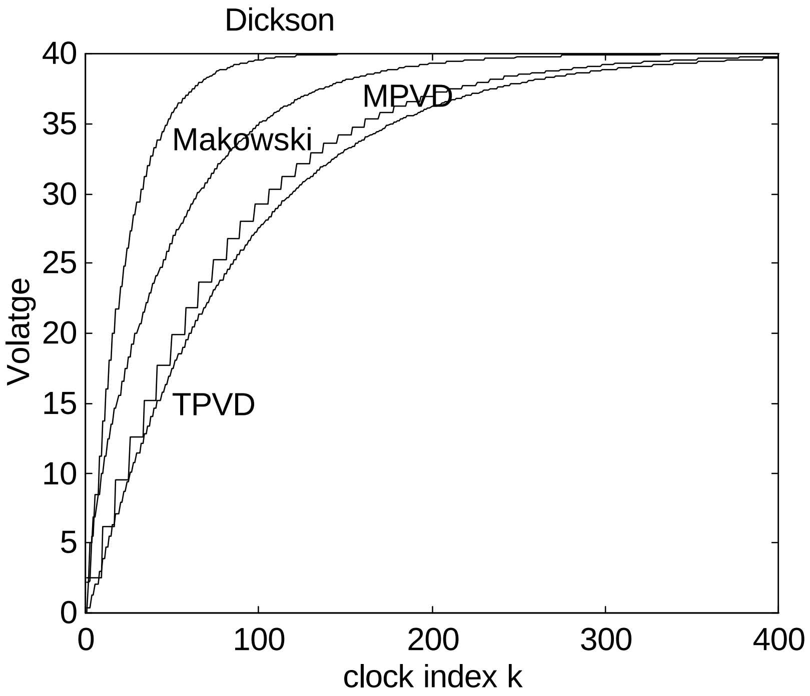


Fig. 5.6 Time domain SAMOC simulation result.

5.3 Consumed Power Analysis Based on a Resistive Load

Although the resistor is not an allowed device in the Q-V realm simulation engine of SAMOC, the analysis of the power consumed by output resistance load of a charge pump is very important to charge pump analysis and design. The SAMOC's *Q-V* realm simulation engine can be modified to analyze the effect of only one resistor in the simulated SC network.

Fig. 5.7 shows the equivalent circuit of a charge pump whose output is connected to a resistive load, and the resistive load is modeled by R_{load}. A resistor is a power dissipation device which drains electrical charge in the form of a current from a high voltage node to ground as shown in Fig. 5.7. C_{eq} is the equivalent capacitor of the charge pump measured at its output terminals. Since the switches of charge pump alter the topologies of the whole charge pump circuit, C_{eq} is not a constant but remains unchanged between two switching events. The electrical charge Q_R consumed by R_{load} during two switching events can be evaluated by:

$$Q_R = V_o C_{eq} (1 - \exp(-\frac{T}{R_{load} C_{eq}})) \quad , \tag{5.2}$$

where T is the time elapsed between two switching events and V_o is the output voltage without the resistive load R_{load}.

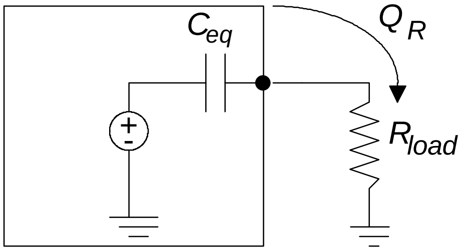


Fig. 5.7 The equivalent circuit of a charge pump circuit with a resistance load.

After evaluating Q_R we can modify the Q-V realm equations as follows:

$$\boldsymbol{CV} = \boldsymbol{Q} - Q_R\, \boldsymbol{d}, \tag{5.3}$$

where $\boldsymbol{d}$ is the selection vector, $V_{out} = \boldsymbol{d}^t \boldsymbol{V} = V_R$. That is, the load resistor in the *Q-V* realm analysis is treated as a voltage dependent charge drain which removes electric charge from the equivalent capacitance C_{eq}. The power P_R dissipated by R_{load} (delivered by the charge pump) can be estimated by

$$P_R = V_R I_R = \frac{V_R^2}{R_{load}} \quad . \tag{5.4}$$

The equivalent capacitance C_{eq} is estimated by putting a dummy voltage source V_d = 0V in the output (see Fig. 5.8 (a)). The electric charge Q_d, goes through the dummy voltage source and can be obtained from simulation of the shorted charge pump formulating the modified nodal-like equations. Then the equivalent capacitance C_{eq} can be estimated using

$$C_{eq} = \frac{Q_d}{V_o} \quad , \tag{5.5}$$

where V_o is the open circuit output voltage which is shown in Fig. 5.8 (b) and used in (5.5).

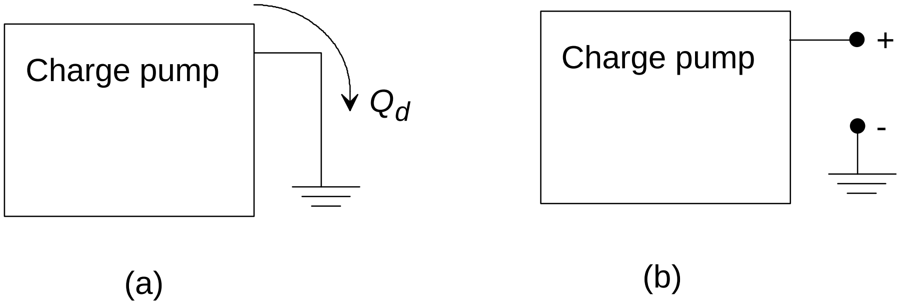


Fig. 5.8 Illustration of the closed circuit charge Q_d (a), and the open circuit output voltage V_O (b).

The resistive load analysis requires 2 Q-V realm analyses in each clock phase instance. The first one is to evaluate the equivalent capacitance C_{eq} by using formula (5.5), and the second one is to calculate the effect caused by removing Q_R from the output of the charge pump by using formula (5.2). In order to understand the effect of the resistive load on the output voltage and the amount of the output power delivered, a TPVD charge pump with 4 cascaded voltage doublers was used for resistive load analysis with different values of the load resistance, R_{load}. The capacitors C_L and C_S are 100pF, and the clock period T is 40ns with power supply 5V. The load resistances used were 2kΩ, 20kΩ, 200kΩ and infinity. Fig. 5.9 shows the simulation results for the first 1500 switching events.

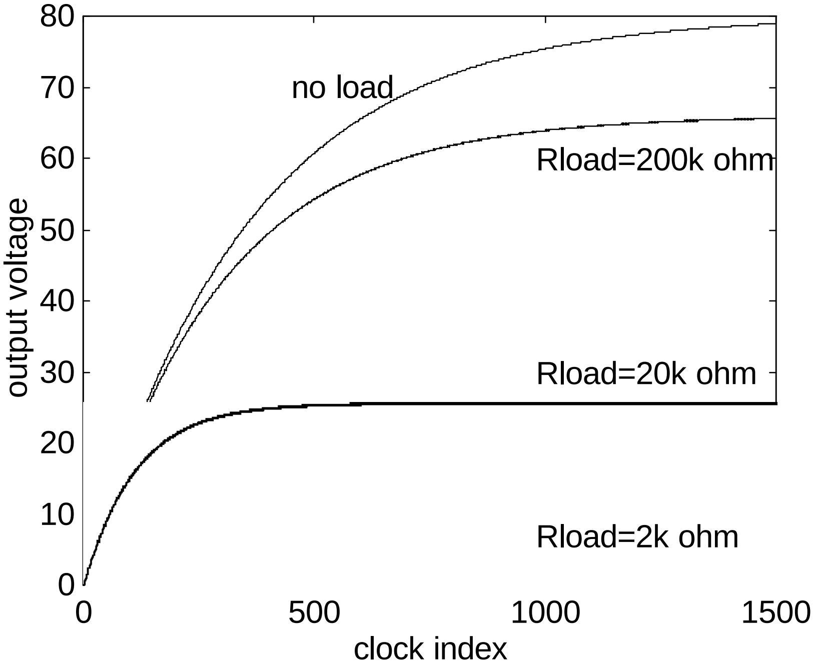


Fig. 5.9 The 4-stage (16x) TPVD charge pump output voltage with different values of R_{load}.

Inspecting the plot, it is obvious that the charge pump can no longer supply the 16x output, while the R_{load} is present. The load resistor R_{load} drains the electric charge supplied by the charge pump. The loss of electric charge decreases the output voltage of the charge pump. The smaller the R_{load} , the more electric charge, Q_R, is drained by R_{load} during the clock period, and the lower the output voltage is.

The resistive load analysis can be used to estimate the output power of a charge pump according to different values of R_{load} and output voltages. To obtain the power and output voltage characteristics of a charge pump, R_{load} values from 10 Ω to 1 mega Ω were used in simulation.

The output power of charge pumps is estimated in SAMOC by finding the maximum output voltage and then using (5.4) to evaluate the output power. Since the time domain output of charge pump is a monotonely increasing function and with a converged final value, SAMOC detects the increasing percentage of the output and approximates the converged maximum output V_R . Fig. 5.10-12 show the SAMOC simulation results of the output power as a function of the load resistance R_{load}. Fig. 5.10 shows that increasing number of stages of a TPVD charge pump can increase the output voltage but can not increase the output power. Fig. 5.11 shows that the output power is a function of capacitors inside the charge pump. The larger the capacitors used, the greater power a charge pump can deliver. Fig. 5.12 shows the same simulation applied to different kinds of charge pump designs. The analyzed four charge pump designs have the same size of individual capacitors and have the same voltage gain equal to 8. Analyzing the plots, we see that a Dickson pump can deliver the largest mount of power and an MPVD delivers the least. A more comprehensive comparison of the four charge pumps is shown in Table 5.1.

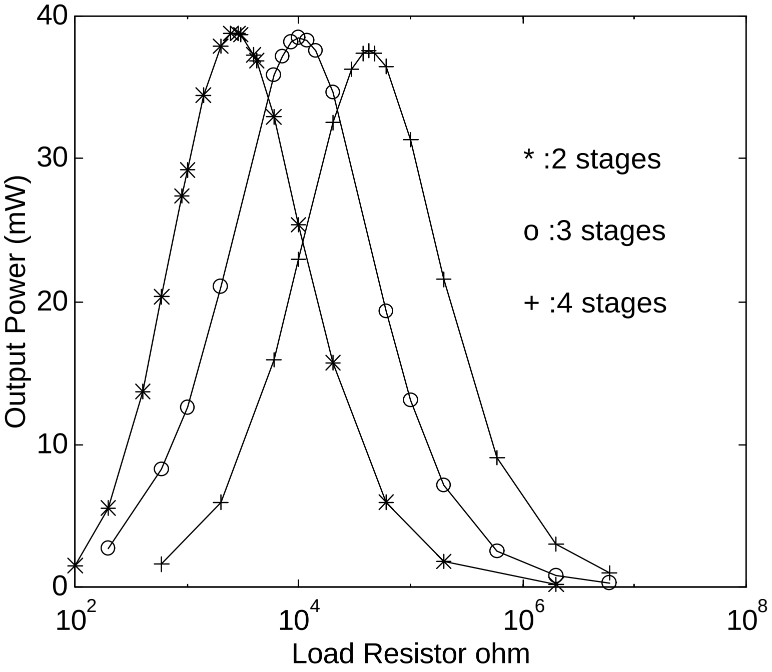


Fig. 5.10 The output power as a function R_{load} for TPVD charge pumps with different number of stages.

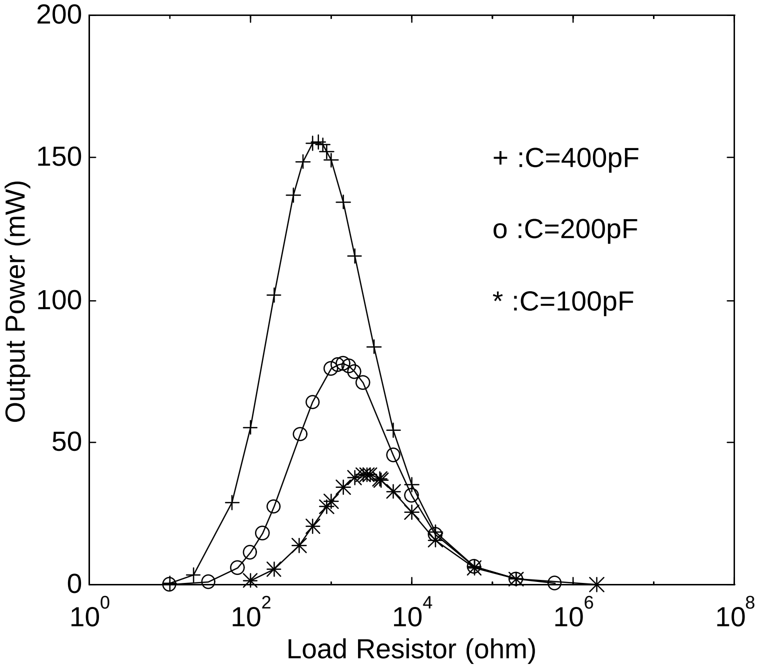


Fig. 5.11 Output power of 2-stage TPVD charge pumps as a function of load resistance and pump capacitance.

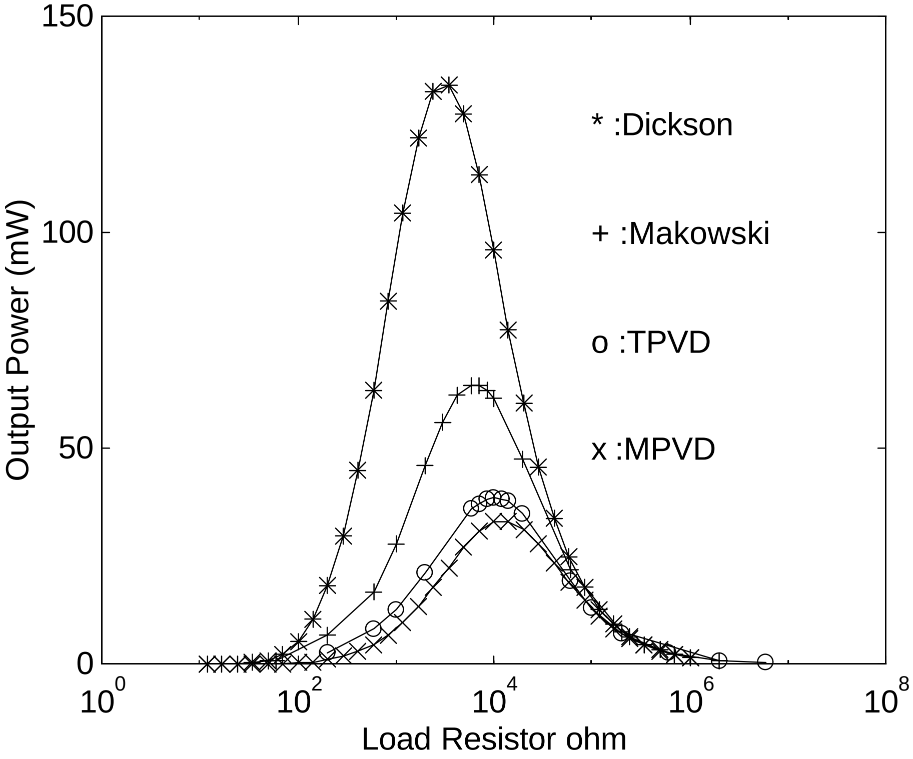


Fig. 5.12 The output power as a function of R_{load}.

Table 5.1: Required number of devices for charge pumps with the same voltage gain A_V = 8.

	TPVD	Dickson	Makowski	MPVD
n (number of stages)	3	7	4	3
switches	12	0	12	12
floating capacitors	3	7	4	3
grounded capacitors	2	0	0	0
diodes	0	8	0	0

5.4 Summary

SAMOC's *Q*-*V* realm simulation engine can be used to analyze the charge pump circuits as well as other switched-capacitor circuits shown in Chapter 4. The comparison between SPICE and SAMOC simulations is presented. SAMOC is more than 10 times faster, based on the assumption of ideal charge transferring between capacitors and idealized models of devices.

SAMOC's simulation of charge pumps in time domains can make a compare rising times. The resistive load analysis of charge pumps became feasible after a minor modification of *Q*-*V* realm simulation engine in SAMOC, although resistor is not an allowed device in SC network designs. The consumed power is estimated by the R-C model and the equivalent capacitor is evaluated by the ratio of closed circuit electrical charge and the open circuit voltage. The consumed power is a function of switch frequency, capacitors' size and the load resistor. The maximum power a charge pump can deliver while the output voltage is half of the maximum output voltage; i.e. ,

$$V_{Pmax}=V_{(Rload=0)} / 2 \qquad (5.6)$$

This interesting result resembles a similar maximum power transfer condition in a linear resistive circuit (voltage divider).

Chapter 6

RESISTIVE-CAPACITIVE NETWORK ANALYSIS, AND CIRCUIT PARTITIONING

In SAMOC, there are two simulation engines for DC analysis. One is the resistive network analyzer which is based on *I-V* realm analysis. The other is the switched-capacitor (SC) network analyzer which is based on *Q-V* realm analysis. The resistive network analyzer presented in Chapters 2 and 3 contains linear and piecewise nonlinear device models and a solution seeking procedure based on the Katzenelson algorithm. The SC network analyzer presented in Chapter 4 is based on *conservation of charge.* This chapter combines resistive and switched-capacitor networks analyses.

In SAMOC, both resistive and SC network analyses employ the MNA circuit formulation approach. The number of circuit equations for MNA, as shown in (2.1), depends mainly on the number of nodes and devices in the simulated circuit. In contemporary VLSI designs, circuits with sophisticated functions usually contain thousands or millions of nodes and devices. Analyzing contemporary VLSI circuits require large numbers of circuit equations to estimate circuit behavior. Solving large numbers of circuit equations by computer demands large memory to store and process the equations. In addition, even applying the piecewise linear approach presented in Chapter 3, the computational complexity of solving n linear equations is O($n^{2.81}$) [65]. In order to extend the ability of circuit simulation for a resource limited computer system, many new methods of managing and manipulating circuit equations are introduced. In the new method employed by SAMOC, instead of formulating the MNA equations of the whole circuit, only a portion or a block of the circuit equations are formulated and solved at a time. By employing this method, much less memory during the circuit simulation is required. In addition, the computational complexity for solving circuit equations in matrix form is dramatically decreased. If analyzing a circuit needs n equations, and that circuit can be divided into several portions which needs n_1, n_2 .. ,where $n = n_1 + n_2 + ...$, equations respectively to solve then

$$O(n^{2.81}) \geq O(n_1^{2.81}) + O(n_2^{2.81}) + ... \qquad (6.1)$$

In order to analyze a circuit portion by portion, circuit partitioning is necessary. After a circuit is partitioned into several blocks, the whole circuit can be analyzed block by block.

6.1 *I-V* Realm Nodes and *Q-V* Realm Nodes

The purpose of DC analysis of a circuit is to estimate the DC voltage of each node and MNA equations are formulated on each nodes. In SAMOC DC analysis, there are two types of nodes. There are *I-V* realm nodes and *Q-V* realm nodes. *I-V* realm nodes connect with resistive devices. A modified nodal equation based on KCL is formulated according to each *I-V* realm node and all resistive devices are connected with that node. *Q-V* realm nodes, on the other hand, are nodes connected only with capacitors, ideal switches, ideal diodes, ideal OPAMPs, and dependent and independent voltage sources. A modified nodal equation based on *conservation of charge* is formulated for each *Q-V* realm node and all allowed devices connected with that node. In SAMOC analysis, two types of nodes arouse two types of circuit formulation methods demanded by different analysis procedures. It is necessary to distinguish I-V realm nodes from *Q-V* nodes for computer circuit equation generation.

In order to mark the type of node in SAMOC simulation, a Boolean variable named `m_bIsAQVNode` is added into the *class* `CNode`. In SAMOC, all *objects* of *class* `CNode` have their `m_bIsAQVNode` set to be *TRUE* at the initialization stage of the reading circuit descriptions. Each node has a list of pointers (see Fig. 2.3) which can locate all connected devices incident to that node. The variable `m_bIsAQVNode` is set to be *FALSE*, if

1. The node is connected with a resistive device, such as resistor or the first or the third node of a MOS transistor.
2. The node is connected with a current source, such as an independent current source, the third or the forth node of current-controlled-current-source, or the third or the forth node of a voltage-controlled-current-source.

3. The node is connected to another node by a short circuit, such as the first or the second node of a current-controlled-voltage-source or the first or the second node of a current-controlled-current-source,

After the Boolean variable `m_bIsAQVNode` in each node is set, SAMOC can formulate different nodal equations either with the *I*-*V* model or with the *Q*-*V* model according to the node type.

6.2 Circuit Partitioning to *I*-*V* Realm and *Q*-*V* Realm Circuit Blocks

The purpose of partitioning a circuit in simulation is to decrease the required memory and computational effort for processing and storing circuit equations. Since there are *I*-*V* realm simulation and the *Q*-*V* realm simulation engines in the SAMOC, it is feasible to partition the simulated circuit into several blocks, and some blocks can be analyzed using the *I*-*V* realm engine while different blocks can be analyzed using the *Q*-*V* realm engine. The *I*-*V* realm engine may employ the Katzenelson algorithm for interactive solution seeking method, and the Q-V realm engine employs a direct method. The separation of *I*-*V* and *Q*-*V* realm blocks also avoids the interactive solution seeking method being employed to analyze a *Q*-*V* realm block. To realize this approach, *I*-*V* realm blocks are considered to contain only *I*-*V* realm devices, and the *Q*-*V* realm blocks contain only *Q*-*V* realm devices. While analyzing a circuit block, an area of memory in a computer system is allocated by SAMOC for restoring and solving circuit equations, the known voltage values are obtained from the main node list of the circuit. After either *I*-*V* or *Q*-*V* realm analysis, the new voltage values are written back to the node list and the memory allocated for analyzing circuit equations is released back to the computer system for better efficient usage of the computing resources. According to (6.1), the more blocks a circuit is partitioned into, the less computational effort is demanded. By simulating a circuit one block at a time, a resource-limited computer system can analyze a relatively large circuit.

6.2.1 *I-V* Realm Blocks

An *I-V* realm block is constructed by a group of circuit devices which can be analyzed by KCL and MNA independently from other parts of the simulated circuit. Since the *I-V* realm analysis is based on nodal analysis and KCL, an *I-V* realm block defined in SAMOC is a group of interconnected nodes and devices for which the only terminals with nonzero current are the ground and voltage source terminals. That is, if there is any resistive link(s) between two nodes in a circuit other than voltage sources and ground nodes, then these two nodes and the resistive link(s) should be in the same *I-V* realm block.

An *I-V* realm block is usually surrounded by voltage sources, ground connections and capacitors. Fig. 6.1 shows an idea of partitioning a circuit into three *I-V* blocks. The capacitors which link *I-V* block together are treated as open circuits in DC analysis, since there is no DC current passing through a capacitor. Note that the nodes connected with voltage sources or ground can be shared by many *I-V* realm blocks, because there is no KCL equation written at ground nor at the output of the voltage sources.

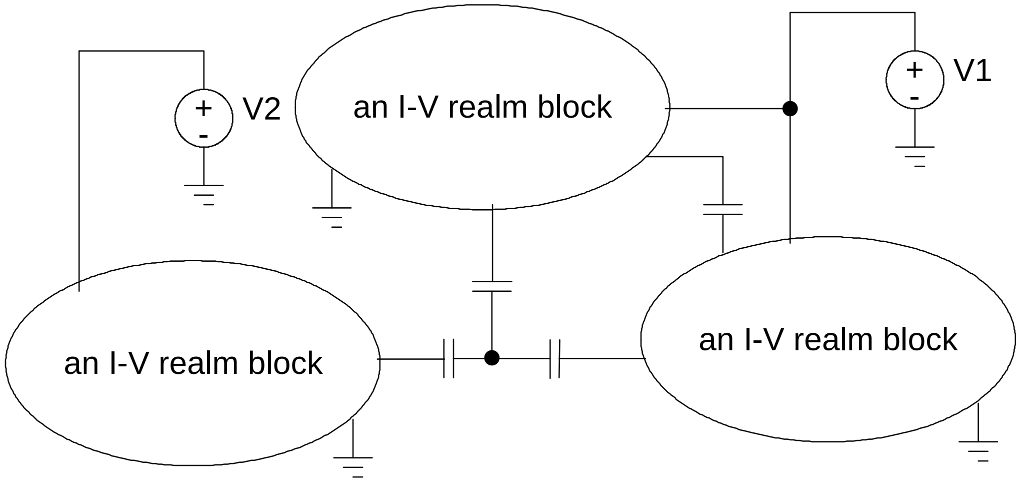


Fig. 6.1 A simulated circuit is partitioned into 3 *I-V* realm blocks.

6.2.2 Shareable Devices and Shareable Nodes

In MNA, there is no KCL equation written in a node which connects to a voltage source or the ground. The voltage value of this type of node is determined before analysis procedure is applied and the amount of current flows through this node can have any value. The nodes connecting a voltage source can be divided into many nodes, as shown in Fig. 6.2 and this voltage source separation will not influence the MNA applied. This simple resistive network can be divided into two *I-V* realm blocks by tearing node “1” and “0” into two nodes. In this scenario, the independent voltage source `V1`, node “1” and node “0” can belong to both *I-V* realm blocks. SAMOC sets nodes of this kind as shared nodes and the independent voltage sources as shareable devices, so that they can be included in several different *I-V* realm blocks. A Boolean variable `m_bIsASharableNode` is added to the *class* `CNode` to represent this shareable characteristic. The setting of the

Boolean variable m_bIsASharableNode in programming SAMOC is done by searching the device list. Nodes connected with an independent or outputs of a controlled voltage source have their m_bIsASharableNode set to *TRUE*. Otherwise, the m_bIsASharableNode of a node is set to *FALSE*. For the example shown in Fig. 6.2 (a). Nodes "1" and "4" and "0" have their m_bIsASharableNode set to *TRUE*. The m_bIsASharableNode of node "2" and "4" are set to *FALSE*.

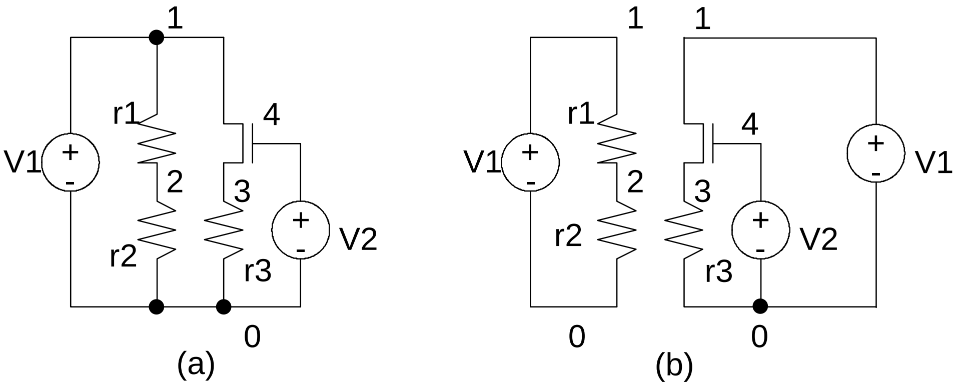


Fig. 6.2 An example of sharing node 1, 0 and V1 in circuit partitioning.

6.2.3 *I-V* Realm Devices

I-V realm devices which are used to construct *I-V* realm blocks are resistive devices, short circuits and current sources. The resistive devices supported by SAMOC are resistors or MOS transistors as shown in Fig. 6.3. A resistive device is constructed by a resistive link and two *I-V* connecting nodes. For a resistor shown in Fig. 6.3 (a), the resistor itself is a resistive link and the two *I-V* connecting nodes are "1" and "2" . For

MOS transistor shown in Fig. 6.3 (b), the channel between source and drain is a resistive link and the two *I-V* connecting nodes are “1” and “3” as shown in Fig. 6.3 (c) .

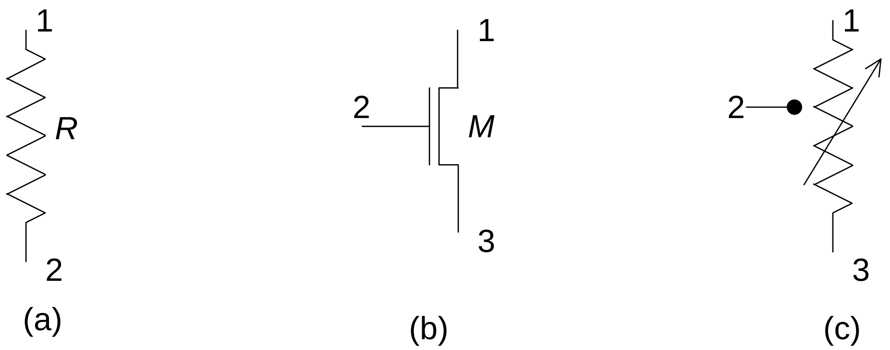


Fig. 6.3 Resistive devices.

The current devices and short circuits supported by SAMOC are an independent current source (Fig. 6.4 (a)), a voltage controlled current source (Fig. 6.4 (b)), a current controlled voltage source (Fig. 6.4 (c)) and a current controlled current source (Fig. 6.4 (d)). Similar to resistive devices, the current devices also have two *I-V* connecting nodes. For an independent current source shown in Fig. 6.4 (a), the two *I-V* connecting nodes are “1” and “2”. For a voltage controlled current source shown in Fig. 6.4 (b), the two *I-V* connecting nodes are “3” and “4”. For a current controlled voltage source shown in Fig. 6.4 (c), the two *I-V* connecting nodes: “1” and “2”. For a current controlled current source shown in Fig. 6.4 (d), there are two pairs of *I-V* connecting nodes are “1” and “2” as well as “3” and “4”. The placement of the *I-V* connecting nodes of a current controlled current source in different blocks will be taken cared by the *I-V* block extraction process described in section 6.2.5.

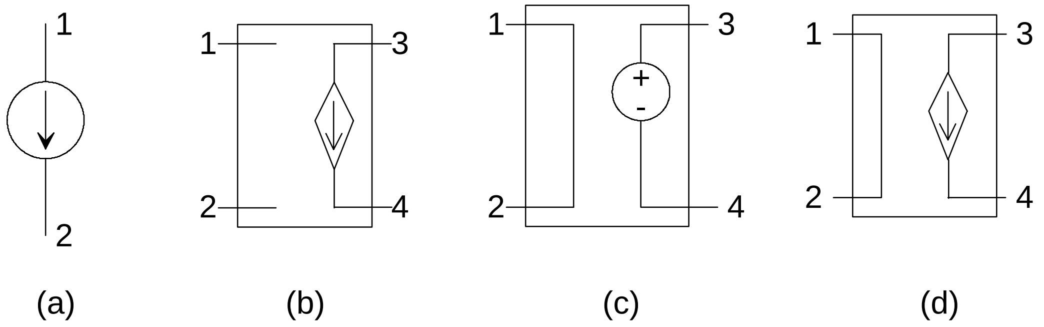


Fig. 6.4 Current devices.

The other devices which can also be included in a *I-V* realm block are ideal diodes, switches, voltage controlled voltage sources, ideal voltage sources and OPAMPs. These devices are allowed to appear in both *I-V* or Q-V realm blocks. If they are connected with *I-V* realm devices, as in the example on Fig. 6.5 (a), they are I-V realm devices. If they are connected with *Q-V* realm devices such as the example in Fig. 6.5 (b), they are *Q-V* realm devices. If they are connected with both *I-V* and *Q-V* realm devices, such as the example in Fig. 6.5 (c), they are *Q-V* realm devices, since in *I-V* realm DC analysis the effect caused by capacitors is discarded. The diode in Fig. 6.5 (c) does not effect the resistive network in the left part of the circuit. But the diode will influence the *Q-V* realm analysis in the right part of the circuit. In such case, the *I-V* realm analysis has to be performed first before the *Q-V* realm analysis. The ordering of interconnected circuit blocks for simulation will be presented section 6.2.8.

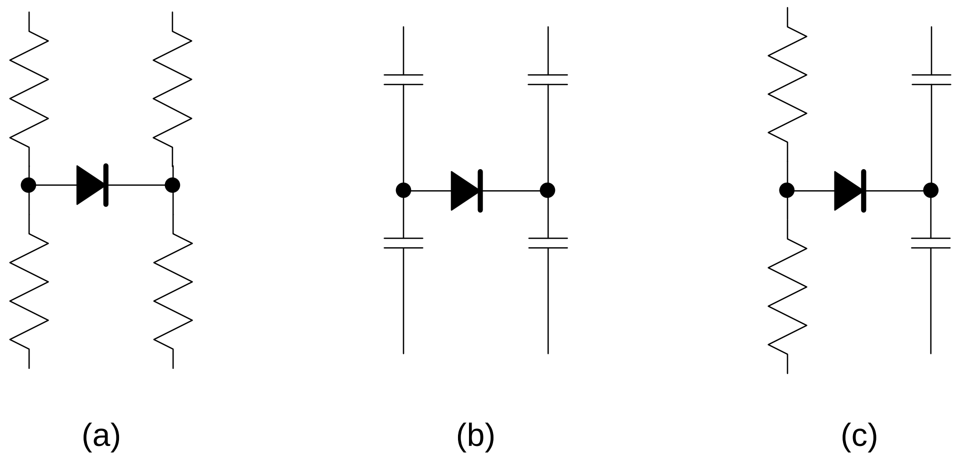


Fig. 6.5 (a) A diode link between two *I-V* realm nodes, (b) a diode link between two *Q-V* realm nodes, (c) a diode link between an I-V realm node and a *Q-V* realm node**.**

6.2.4 *I-V* Realm Block Extraction

I-V realm blocks are constructed by interconnected *I-V* realm devices via *I-V* realm nodes. The *I-V* realm block extraction begins with finding a non-included two-terminal *I-V* realm device, (usually a resistor, a MOS transistor or an independent current source). *I-V* realm devices of this kind usually connect with two *I-V* realm nodes and these two nodes connect with other *I-V* realm devices. Then these *I-V* realm devices are connected with other *I-V* realm nodes. The *I-V* realm block extraction procedure is constructed by including new *I-V* realm devices connected to its *I-V* realm nodes, and including new *I-V* realm nodes connected included *I-V* realm devices alternately. The included nodes and devices must be marked to prevent infinity loops. The extraction procedure does not include new *I-V* realm devices from shareable nodes. The extraction procedure stops when all connected devices are included and marked. The voltage source then will be added if a circuit block contains both nodes of the voltage source. For a simple example, in the circuit in Fig. 6.2 (a), the extraction begins at including the resistor r1 and steps of action are illustrated in table 6.1.

Table 6.1 An *I-V* block extraction example.

step	new device	new nodes	devices and nodes in the *I-V* realm block	remark
1	r1	1, 2	r1 1,2	node 1 is shareable
2	r2 (from node 2)	2, 0	r1, r2 1 ,2 ,0	node 0 is shareable
3	V1		V1, r1, r2 1, 2 ,0	This block contains node 1 and 0. V1 is included

The software implementation of *I-V* realm block extraction can exploit the massive interconnected data structure of SAMOC, which are presented in Chapter 2. As illustrated in Fig. 2.3, it is easy to locate the connecting nodes from a device and from a node to locate its connected device. The programming technique used in SAMOC for *I-V* realm extraction is recursive. The subroutine which searches and includes a device incident to a node is applied to each new included device and its nodes. Each subroutine ends when for all its processed device, their nodes are

1. *I-V* realm nodes whose all interconnected devices are included, and
2. shareable nodes

Next section presents more detail about including controlled source, voltage source, switches, and OPAMPs, into an *I-V* realm block.

6.2.5 Including Controlled Sources and MOS Transistors into an I-V Realm Block

The initialization of an *I-V* realm block does not begin with a controlled source, since a controlled source can belong to different blocks. For example, in Fig. 6.4(d), there is no current from node “1” to node “4”. An *I-V* realm block which contains nodes “1” and “2” can be analyzed independently from the *I-V* block which contains nodes “3” and “4”. An *I-V* realm device, except the initial one, is included in an *I-V* realm block through a node connection with an already included device. The including procedure depends on the device type and connecting node. Among them, controlled sources are more complex then other type of devices, since any two nodes of a controlled source could be a short circuit, an open circuit, current source or voltage source.

For MOS transistors, as shown in Fig. 6.4 (b-c), there is no DC current connection between gate to source or gate to drain. Gate can be treated as open circuit and a separable node. In *I-V* realm block extraction, different rules are applied to different types of devices and different terminals (nodes). The comprehensive descriptions of devices inclusion are summarized in Table 6.2.

Table 6.2 The descriptions of including devices into an I-V realm circuit block.

circuit device	SPICE symbol	connection node number	action	remarks	include to the *I-V* block?
resistors	R	1	include node “2”	resistive link between “1” and “2”	Yes
		2	include node “1”		
independent current source	I	1	include node “2”	a current source connect “1” and “2”	Yes
		2	include node “1”		
capacitor	C	1	do nothing	it’s an open circuit in I-V realm analysis	No
		2	do nothing		
independent voltage source	V	1	do nothing	will be added by another procedure	Yes
		2	do nothing		
voltage controlled current source Fig. 6.4 (b)	G	1	do nothing	it’s an open circuit	No
		2	do nothing	it’s an open circuit	
		3	include node “4”	a current source connect “3” and “4”	Yes
		4	include node “3”		
current controlled voltage source Fig. 6.4 (c)	H	1	include node “2”	a short circuit connect “1” and “2”	Yes
		2	include node “1”		
		3	do nothing	will be added by another procedure	
		4	do nothing		

Table 6.2 (Continued)

current controlled current source Fig. 6.4 (d)	F	1	include node “2”	a short circuit connect “1” and “2”	Yes
		2	include node “1”		
		3	include node “4”	a current source connect “3” and “4”	
		4	include node “3”		
MOS transistor Fig. 6.4 (b)	M	1	include node “3”	a resistive link between “1” and “3”	Yes
		2	do nothing	it’s an open node	No
		3	include node “1”	a resistive link between “1” and “3”	Yes
		4	do nothing	not used by SAMOC	No
voltage controlled voltage source	E	1	do nothing	open circuit between “1” and “2”	No
		2	do nothing		
		3	do nothing	will be added by another procedure	Yes
		4	do nothing		
voltage controlled ideal switch	S	1	include node “2”	a possible short circuit between these two nodes	Yes
		2	include node “1”		
		3	do nothing	they are open circuits	No
		4			
ideal diode	D	1	include node “2”	a possible short circuit between these two nodes	Yes
		2	include node “1”		
OPAMP	O (not in SPICE)	1	do nothing	will need another procedure	Yes
		2			
		3			

6.2.6 Adding Voltage Sources into An *I-V* Realm Block

Four types of voltage sources are supported by SAMOC: independent voltage source, voltage controlled voltage source, current controlled current source and OPAMP. Because there is no KCL equation written at the terminals of these voltage sources and these sources are shareable device as illustrated in Fig. 6.2, including voltage source into a circuit block requires another procedure.

An independent voltage source is a two-terminal device which connects two nodes. It is added into an *I-V* block if that block contains both nodes. The voltage controlled voltage source is a four-terminal device and connects with four nodes. Inside the device, it is an open circuit between the first and second nodes and there is no equation for these two nodes. Hence, the first and second nodes can be discarded. It is a voltage source between the third and forth nodes. The voltage controlled voltage source is added into an *I-V* block, if that block contains the third and forth nodes.

Similar to a voltage controlled voltage source, a current controlled voltage source is a four-terminal device. A current controlled voltage source has a short circuit between its first and second nodes and a voltage source between its third and forth nodes. In *I-V* realm block extraction, a current controlled voltage source is divided into two parts and can belong to different *I-V* realm blocks. The *I-V* realm block which contains the first and second nodes has the short circuit part. The *I-V* realm blocks which contain the third and forth nodes have the voltage source part. Note that similar to other voltage sources, the voltage source part of a current controlled source can belong to many *I-V* blocks.

OPAMP is a special condition of the voltage controlled voltage source type of device. An OPAMP connects to three nodes. Similar to a voltage controlled voltage source, an OPAMP is added into the *I-V* block which contains the third node of the OPAMP.

6.2.7 Data Structure for I-V Realm Block Partitioning

The *objects* representing *I-V* realm blocks in SAMOC are of the *class* of `CCircuitBlock`. The organization of `CCircuitBlock` illustrated in Fig. 6.6 is very similar to `CCircuit` presented in Chapter 2. `CCircuitBlock` contains two main lists of pointers. One is the device list which can locate the devices included in the *I-V* realm block. The other is the node list which is used to locate nodes in the *I-V* realm block. In analyzing the *I-V* realm block, the known voltage and current values or initial conditions can be obtained from the node list of *object* of `CCircuit`. The device list can be used to generate the MNA models. One pointer which can locate the address of an *object* of `CCircuitBlock` is added to the device *class* (`C2Term`) and the node *class* (`CNode`). Therefore, in future data manipulation, it takes a little effort to know which circuit block a device or a node belongs to. As illustrated in Fig. 6.6 each node and each device has a pointer which locate its circuit block. Adding this variable helps the speed of circuit block extraction because this variable helps avoiding infinity loops. Another function of this variable is that it accelerates the setting the block simulation order which is presented in the section 6.2.8.

A node, except a shareable node, can only be included in one circuit block. For the shareable nodes, the embedded pointer to circuit block is useless. Nodes of all controlled sources, voltage controlled switches, and OPAMPs can belong to different circuit blocks. For voltage controlled current sources, voltage controlled voltage sources, voltage switches, and OPAMPs, the controlling nodes are open circuits and there is no equation for controlling nodes. Hence, for controlled current sources, voltage controlled voltage sources, voltage controlled switches, and OPAMPs, their circuit block pointers locate the circuit block which contains their output nodes. For current controlled voltage sources and current controlled current sources, there are circuit equations for both controlling and controlled nodes. The controlled (output) nodes of a current controlled voltage source are shareable and can belong to many circuit blocks. Some procedures are required to finish the work, such as adding voltage sources procedure presented in section 6.2.6. For current controlled current sources, KCL is applied to both controlling and controlled parts of the device. One further pointer to a circuit block is added for this device. Therefore, all device *objects* except the current controlled current source *objects* in SAMOC have a simple pointer to a circuit block. The current controlled current source *objects* have two pointers to circuit blocks. A circuit block list (list of pointers to circuit blocks) is added into `CCircuit` *class* for managing all extracted circuit blocks.

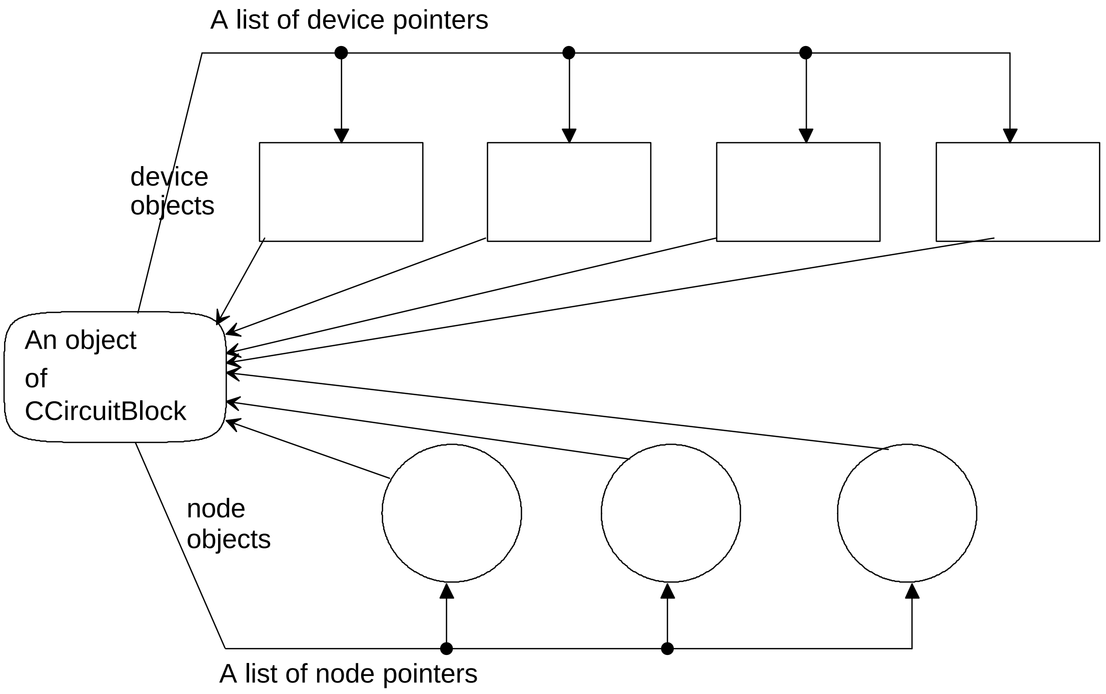


Fig. 6.6 Data structure organization of a circuit block.

6.2.8 Posterior-Prior Relationships Between Circuit Blocks for Analysis

All controlled devices have controlling node(s) and controlled nodes. The device models or parameters of the controlled parts are determined by the controlling parts of the device. In the circuit analysis presented in Chapter 3, SAMOC creates all circuit equation in MNA and KCL algorithms and finds solutions at one linear or piecewise linear system. All variables of controlled devices are formulated and solved at the same time.

In the circuit block analysis, the controlling parts and the controlled parts of a device may not belong to the same circuit block. In this scenario, the circuit block(s) with the controlling parts must be analyzed prior to the circuit block(s) with the controlled parts.

The controlled devices supported by SAMOC are voltage controlled current sources, voltage controlled voltage sources, current controlled current sources, current controlled voltage sources, voltage controlled switches and MOS transistors. For voltage controlled current sources, voltage controlled voltage sources, current controlled current sources and current controlled voltage sources, the circuit blocks containing the first and second nodes have to be analyzed prior to the circuit blocks containing the third and fourth nodes. For voltage controlled switches, circuit blocks containing the third and fourth nodes have to be analyzed prior to the circuit blocks containing the first and second nodes. For a MOS transistor, the circuit block containing the second node has to be analyzed prior to the circuit blocks containing the first and third nodes.

The posterior-prior relationship setting of circuit blocks exploits the data structure presented in Chapter 2 and section 6.2.7. As illustrated in Fig. 2.3, for all controlled sources, it is easy to locate nodes. Fig. 6.6 shows that each node has a pointer to the circuit block which contains this node. Therefore, it does not cost much effort to know if a controlled device is connected to two different circuit blocks. For a controlled device that connects two different circuit blocks, the posterior-prior relationship of the two circuit blocks for analysis is easy to be found and set. For the example illustrated in Fig. 6.7, from a transistor Mx shown in Fig. 6.7(a), it is easy to know that the node “2” of Mx belongs to a different circuit block (block a) than nodes “1” and “3” (block b) do. A direction link to block b, implemented by a pointer, is added to block a to mark that analyzing of block a has to be prior to analyzing block b, as shown in Fig. 6.7 (b).

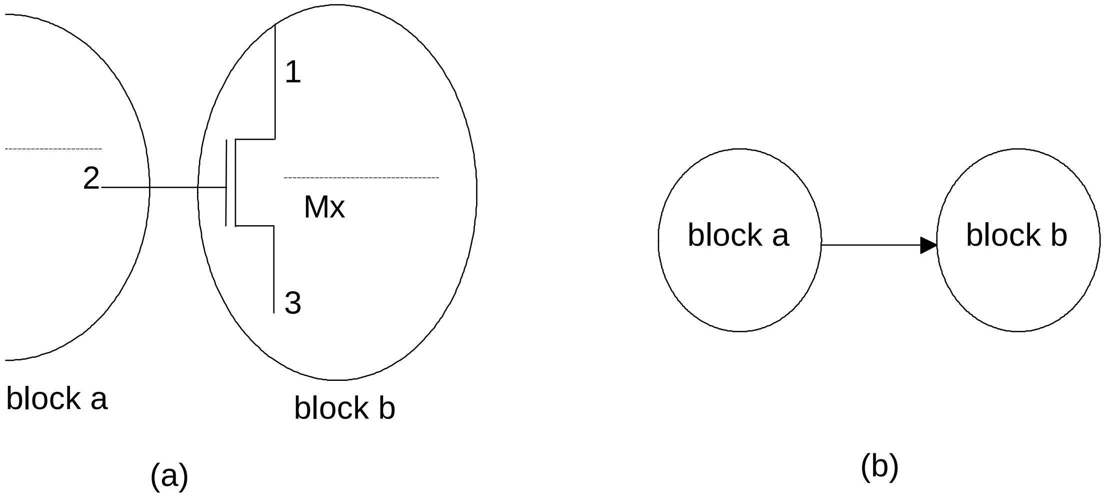


Fig. 6.7 An example of posterior-prior setting of two circuit blocks.

For the example illustrated in Fig. 6.8, both block “a” and “b” must be analyzed prior to block “f” and block “f” has to be analyzed prior to block “g” and “d”. For the complex prior-posterior relationships, two circuit block lists (lists of pointers to circuit blocks) are added to `CCircuitBlock` *class*. One circuit block list is added to locate the circuit blocks which have to be analyzed prior to “*this*” circuit block. The other circuit block list locates the circuit blocks which have to be analyzed posterior to “*this*” circuit block. The massive interconnection between circuit blocks for the prior-posterior relationships speeds up the event-driven analysis which is presented in Chapter 7. Fig. 6.8, representing a circuit by circuit block and signal between blocks, is call a block-signal diagram of a circuit. In computer programming, the block-signal diagram is represented by a directed graph. Many algorithms invented for manipulating a directed graph direct such as topology sorting and least cost path can be used to handle the block-signal diagram tasks.

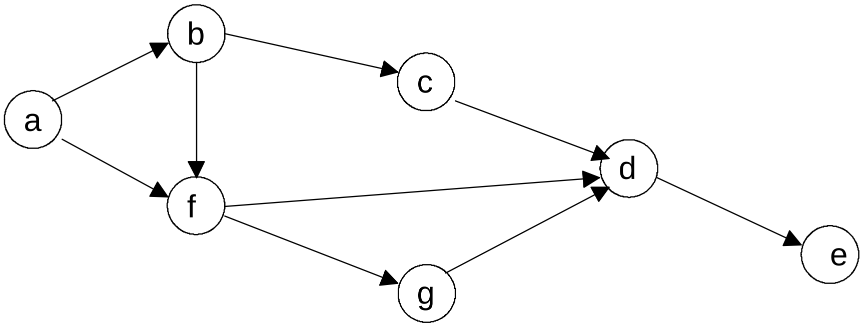


Fig. 6.8 A block-signal diagram of a circuit which contains 7 circuit blocks.

6.3 Device Stamps Modification for Analyzing I-V Realm Blocks

In order to analyze a circuit block, the device stamps for MNA presented in Chapter 2 have to be modified, because all nodes of some devices may not belong to the same circuit block. The devices whose stamps have to be modified for block analysis are MOS transistors, voltage controlled voltage sources, voltage controlled current sources, current controlled voltage sources and current controlled current sources.

6.3.1 MOS Transistors (M)

The MOS transistor model in SAMOC has three regions. The device stamps are illustrated by (2.11), (2.13) and (2.15). For block analysis, if the gate of the MOS transistor belongs to the same block as the drain and the source belong to, then the device model used for the block is the same as presented in Chapter 2. If the gate is not in the same circuit block with the drain and the source, then the circuit containing the gate has to be analyzed prior to the one containing drain and source. After V_g is obtained, the state of the MOS transistor can be determined. If the MOS transistor is in the cutoff region, then (2.11) is filled in to the circuit matrix of circuit block which contains drain and source of the transistor. If the MOS transistor is in the linear region, then (2.13) is filled in to the circuit matrix of circuit block which contains drain and source of the transistor. If the MOS transistor is in the saturation region, then (2.15) is modified to

$$\begin{array}{c|cc|c|} & V_D & V_S & \text{excitation vector} \\ D & G_{\min} & -G_{\min} - g_m & g_m(V_t - V_g) \\ S & -G_{\min} & G_{\min} + g_m & -g_m(V_t - V_g) \end{array} \tag{6.2}$$

and filled in to the circuit matrix of the circuit block which contains drain and source of the transistor.

6.3.2 Voltage Controlled Voltage (E) and Current Sources (G)

The two controlling nodes of voltage controlled voltage and current sources can be any two nodes in the circuit. There are three kinds of possible situations for a voltage controlled source whose controlling node(s) is (are) not in the same circuit block as the their output nodes. For the situation shown in Fig. 6.9 (a), the circuit blocks a and b which contain controlling nodes of the controlled device, 1 and 2, have to be analyzed prior to the block c which contains the output of the controlled sources. For the situation shown in Fig. 6.9 (b), the circuit block a which contains both controlling nodes of the controlled device, 1 and 2, must be analyzed prior to the block c which contains the output of the controlled sources. In situations presented in Fig. 6.9 (a) and (b), the voltages of two controlling nodes, V(1) and V(2), are estimated prior to analyzing the circuit block containing the controlled source. While analyzing the circuit block containing the controlled source, the controlled source is treated as an independent source with the output value as a function of V(1) and V(2).

For the situation illustrated in Fig. 6.9 (c), the circuit block "c" contains a voltage controlled source and one of its two controlling node, "1", is included in another circuit block "a". The circuit block "a" has to be analyzed prior to "c". The device stamps of the voltage controlled sources in this case as follows:

1. For the voltage controlled current source with node "1" belonging to other circuit block, V_1 must be estimated prior to analyzing the block that contains this voltage controlled current source. The device stamp for the voltage controlled current source is:

$$\begin{array}{ccc} & V_2 & \text{excitation vector} \\ 3 & \left[\begin{array}{c} -g \\ g \end{array}\right] & \left[\begin{array}{c} -g \times V_1 \\ g \times V_1 \end{array}\right] \\ 4 & & \end{array} \qquad (6.3)$$

2. For the voltage controlled current source with node "2" belonging to other circuit block, V_2 must be estimated prior to analyzing the block that contains this voltage controlled current source. The device stamp for the voltage controlled current source is:

$$\begin{array}{c|c|} & V_1 \\ 3 & g \\ 4 & -g \end{array} \qquad \text{excitation vector} \begin{array}{|c|} \\ g\times V_2 \\ -g\times V_2 \end{array} \qquad (6.4)$$

3. For the voltage controlled voltage source with node “1” belonging to other circuit block. *V₁* must be estimated prior to analyzing the block that contains this voltage controlled voltage source. The device stamp for the voltage controlled current source is:

$$\begin{array}{c|cccc|} & V_2 & V_3 & V_4 & I \\ 2 & & & & \\ 3 & & & & 1 \\ 4 & & & & -1 \\ m+1 & \mathrm{m} & 1 & -1 & \end{array} \qquad \text{excitation vector} \begin{array}{|c|} \\ \\ \\ \\ \mathrm{m}\times V_1 \end{array} \qquad (6.4)$$

4. For the voltage controlled voltage source with node “2” belonging to other circuit, and *V₂* must be estimated prior to analyzing the block that contains this voltage controlled voltage source. The device stamp for the voltage controlled current source is:

$$\begin{array}{c|cccc|} & V_1 & V_3 & V_4 & I \\ 1 & & & & \\ 3 & & & & 1 \\ 4 & & & & -1 \\ m+1 & -\mathrm{m} & 1 & -1 & \end{array} \qquad \text{excitation vector} \begin{array}{|c|} \\ \\ \\ \\ -\mathrm{m}\times V_2 \end{array} \qquad (6.5)$$

For the situation in Fig. 6.9 (d), where all four terminals of a voltage controlled source belong to the same circuit block, the device stamp remains the same as presented in Appendix I.

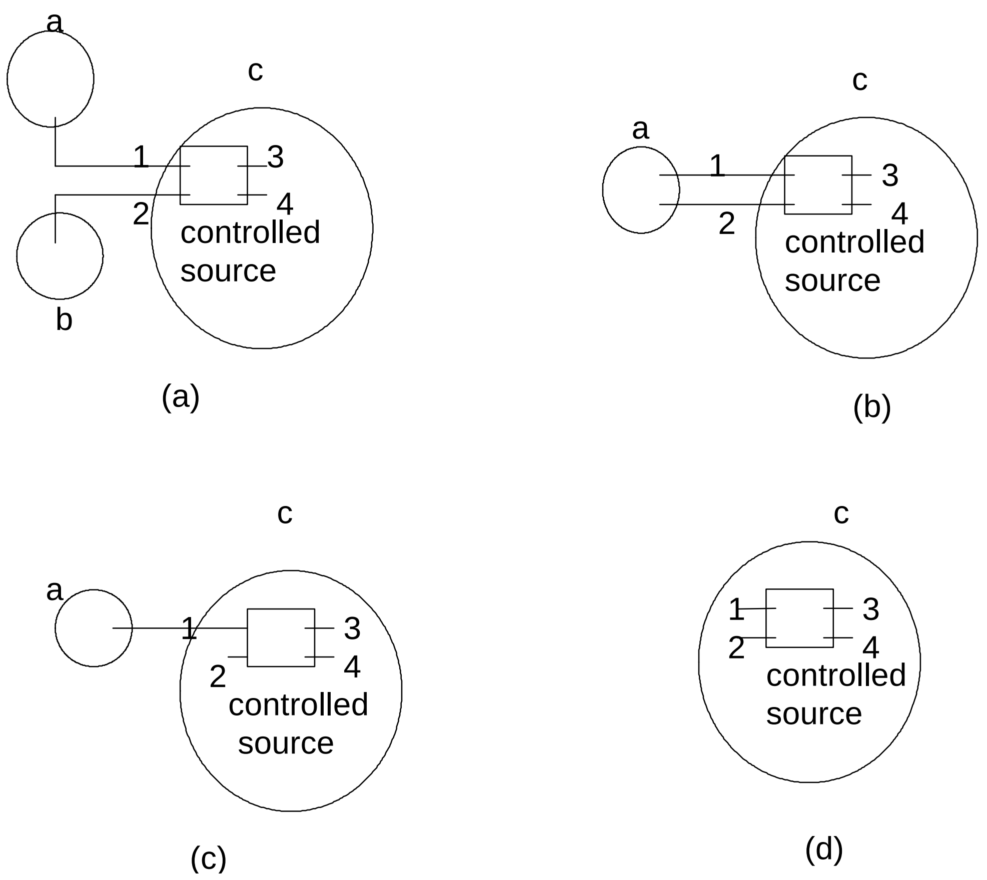


Fig. 6.9 A voltage controlled source with its controlling node in many circuit blocks.

6.3.3 Current controlled voltage source (H)

For a current controlled voltage source, there is only one situation where its controlling nodes and controlled nodes do not belong to the same circuit block. Fig. 6.10 illustrates this situation where controlling nodes (1 and 2) belong to block a, and the controlled nodes (3 and 4) belong to block b.

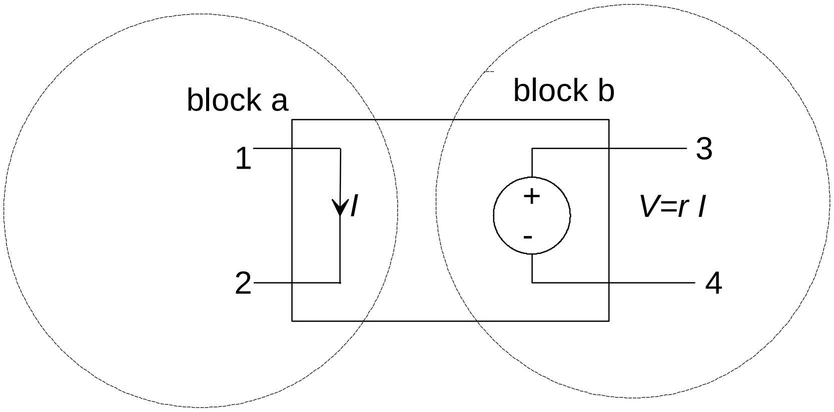


Fig. 6.10 A current controlled voltage source whose controlling nodes and controlled nodes do not belong to the same circuit block.

In block a, the current controlled voltage source is a short circuit. Block a must be analyzed before block b in order to obtain the magnitude of current *I* in block a. The circuit stamp for the current controlled voltage source in block a is:

$$\begin{array}{c|ccc|} & V_1 & V_2 & I \\ 1 & & & 1 \\ 2 & & & -1 \\ m+1 & 1 & -1 & \end{array} \tag{6.6}$$

The controlled nodes of the current controlled voltage source in block b are treated as an independent voltage source. Since the current *I* has been obtained by analyzing block a and *r* is the parameter provided by the current controlled voltage source, the device stamp for the controlled part of the current controlled voltage source is:

$$\begin{array}{cccc} & V_3 & V_4 & I \\ 3 & & & 1 \\ 4 & & & -1 \\ m+1 & 1 & -1 & \end{array} \qquad \begin{array}{c} \text{excitation vector} \\ \\ \\ r \times I \end{array} \tag{6.7}$$

6.3.4 Current controlled current source (F)

For a current controlled current source, there is only one situation where its controlling nodes and controlled nodes do not belong to the same circuit block. Fig. 6.11 illustrates this situation where controlling nodes (1 and 2) belong to block a, and the controlled nodes (3 and 4) belong to block b.

In block a, the current controlled current source is a short circuit. Block a has to be analyzed before block b in order to obtain the magnitude of current *I* in block a. The circuit stamp for the current controlled current source in block a is:

$$\begin{array}{cccc} & V_1 & V_2 & I \\ 1 & & & 1 \\ 2 & & & -1 \\ m+1 & 1 & -1 & \end{array} \tag{6.8}$$

The controlled nodes of the current controlled current source in block b are treated as an independent current source. Since the current *I* has been obtained by analyzing block a and α is the parameter provided by the current controlled current source, the device stamp for the controlled part of the current controlled source is:

$$\begin{array}{cc} & \text{excitation vector} \\ 3 & \left[\begin{array}{c} -a \times I \\ a \times I \end{array}\right] \\ 4 & \end{array} \qquad (6.9)$$

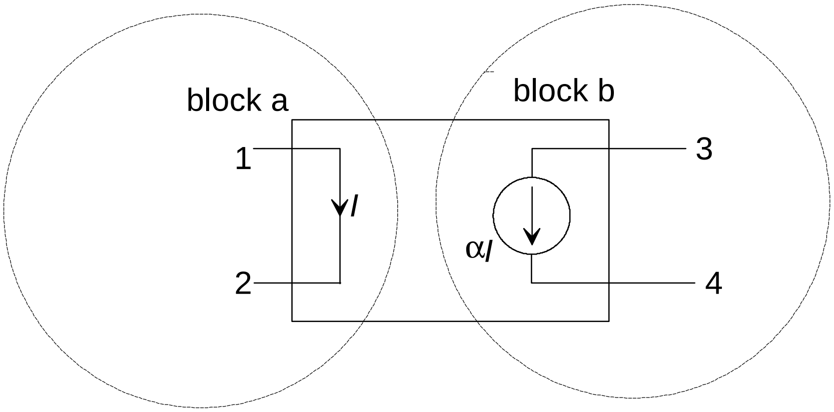


Fig. 6.11 A current controlled current source whose controlling nodes and controlled nodes do not belong to the same circuit block.

6.4 *Q-V* Realm Blocks

A *Q-V* realm block is constructed by a group of circuit devices which can be analyzed independently from other parts of the simulated circuit by *conservation of charge* and MNA from other part of the simulated circuit. In SAMOC, a *Q-V* realm block contains only *Q-V* realm devices which are capacitors, ideal switches, ideal diodes, voltage controlled voltage sources and OPAMPs. The simulation engine of *Q-V* realm blocks is the switched-capacitor simulator presented in Chapter 4.

Similar to analyzing procedure of *I-V* realm blocks, the *Q-V* realm simulator reads the voltage data from the main node list that belongs to the *object* of the `CCircuit` *class*, and after the *Q*-*V* realm block analyzing, the new voltage values are written back to the main node list.

6.4.1 *Q*-*V* Realm Block Extraction

The extraction of a *Q*-*V* realm block is very similar to the extraction of an *I*-*V* realm block. The extraction begins with finding a capacitor with a least one terminal connecting with a *Q*-*V* realm node. If a capacitor is connected with two *I*-*V* nodes, as illustrated in Fig. 6.12, then there is no conservation of charge equation for that capacitor. That capacitor can be discarded in DC analysis, because it is treated as an open circuit. A *Q*-*V* realm node always connects with at least two *Q*-*V* realm devices. The *Q*-*V* realm node is included into the *Q*-*V* realm block and then the *Q*-*V* realm devices are included. The extraction procedure is constructed by including *Q*-*V* realm devices and nodes alternately. The location of connected devices from nodes and connected nodes from devices can be efficiently performed in large circuit using data structure in SAMOC which was presented in Chapter 2.

The inclusion of controlled sources into a *Q*-*V* realm block is much simpler than their inclusion into an *I*-*V* realm block. The only allowed controlled source in a *Q*-*V* realm block is a voltage controlled voltage source. In MNA of a *Q*-*V* node, the characteristic between two controlling nodes of a voltage controlled voltage source is an open circuit, and the characteristic between two controlled nodes is a voltage source.

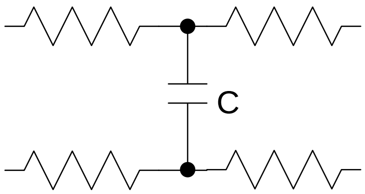


Fig. 6.12 A capacitor C connects with two *I-V* realm nodes.

During the device inclusion step of the *Q-V* block extraction, a device is included into the block via a connecting node. The process of including a device into a circuit block not only depends on the type of included device but also on the node of the device. Table 6.3 illustrates the comprehensive description of including a device into a *Q-V* block via a connection node.

Table 6.3 The rules for inclusion of devices into a *Q-V* realm block.

device type	SPICE symbol	connection node of the device	action	remarks	include device into *Q-V* block?
capacitor	C	1	include node “2”	capacitive link between “1” and “2”	Yes
		2	include node “1”		
independent voltage source	V	1		will need another procedure to add voltages	Yes
		2			

Table 6.3 (Continued)

voltage controlled voltage source	E	1	do nothing	open circuit between these 2 nodes	Yes
		2			
		3	do nothing	needs another procedure to add voltages	
		4			
current controlled voltage source	H	1	do nothing and send an error message	a short circuit with DC current shouldn’t take place in a *Q*-*V* realm block	No
		2			
		3	do nothing	needs another procedure to add voltages	Yes
		4			
MOS transistor	M	1	do nothing	shouldn’t happen, it’s an *I*-*V* realm node	no
		2	create a C_{gs}, C_{gd} and C_{gb}	parasitic capacitors	yes
		3	do nothing	shouldn’t happen, it’s an *I*-*V* realm node	no
		4	do nothing	shouldn’t happen, it’s an *I*-*V* realm node	no
Ideal diode	D	1	include node “2”	a possible short circuit between these two nodes	yes
		2	include node “1”		

Table 6.3 (Continued)

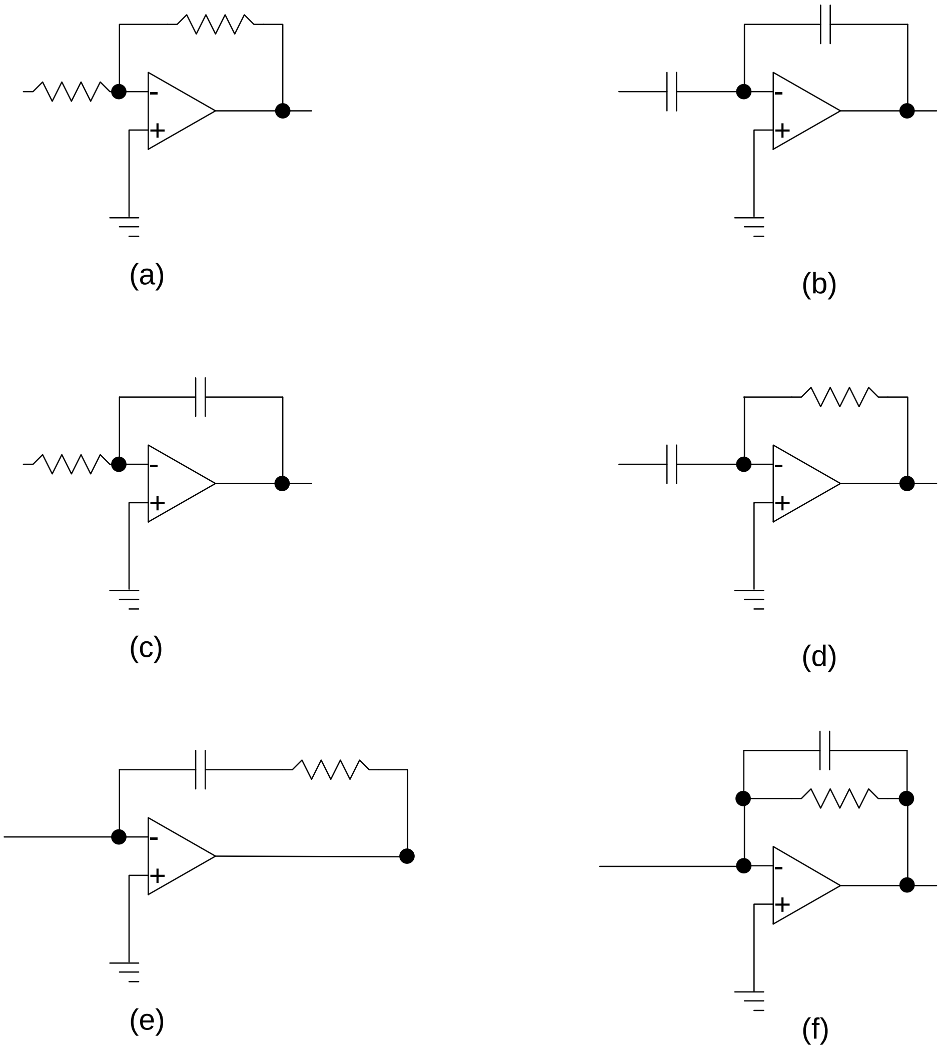
Ideal switch	S	1	include node “2”	a possible short circuit between these two nodes	yes
		2	include node “1”		
		3	do nothing	inside the device, it’s an open circuit between these nodes	no
		4			
OPAMP	O (not in SPICE)	1	do nothing	will need another procedure to decide	
		2			
		3			

6.4.2 Including Voltage Sources and OPAMPs into a *Q-V* Realm Block

Adding voltage source and OPAMPs into a *Q-V* realm block requires an additional procedure, because output nodes of voltage sources and OPAMPs are shareable nodes and can belong to many different circuit blocks which may be *I-V* realm or *Q-V* realm blocks.

Adding an independent voltage source into a *Q-V* realm block is done by checking if this *Q-V* node has both nodes connected to the two terminals of the independent voltage source. Adding a voltage controlled or current voltage source in to a *Q-V* realm block is done by checking if this *Q-V* node has both nodes which connect to the two controlled nodes of the controlled source. Note that the controlling nodes of a current controlled voltage source can not be included into a *Q-V* realm block, because there is no DC current in any *Q-V* realm block.

An ideal OPAMP is a special case of a voltage controlled voltage source and it is not supported in SPICE. An ideal OPAMP requires a feedback link. There are several possible feedback link organizations, as illustrated in Fig. 6.13. If the feedback link with an OPAMP is as shown in Fig. 6.13 (a), then they can be included in an *I-V* realm block.



(a) (b) (c) (d) (e) (f)

Fig. 6.13 Feedback link and OPAMPs.

If the feedback link with an OPAMP is as shown in Fig. 6.13 (b), then it can be included in a *Q-V* realm block. For situations illustrated in Figs. 6.13 (c- f), neither *I-V* realm nor *Q-V* realm simulation engine can find out the DC solution. These cases will require a frequency domain analysis or integration techniques. Note that this discussion on feedback links with OPAMPs is greatly simplified. In general circuits, feedback links with OPAMPs can be very complex. SAMOC will try to organize *I-V* and *Q-V* realm block no matter how complicated the feedback loop is. Therefore, if the feedback loops extends beyond a single block, the algorithm provides a solution which converges to a true value only after several iterations.

6.4.3 Analyzing *Q-V* Realm Blocks Incident with *I-V* Realm Blocks

The analysis of a *Q-V* realm block employs the switched-capacitor simulation engine presented in Chapter 4. To analyze a *Q-V* realm block which has connection to other *I-V* realm blocks, such as illustrated in Fig. 6.14 (a), some pseudo voltage sources V1 and V2 must be added to the *Q-V* realm block, as illustrated in Fig. 6.14 (b). Because the values of V1 and V2 are determined by the *I-V* realm block, the *I-V* realm block has to be analyzed prior to the Q-V realm block. The procedure to add a pseudo voltage sources is:

1. Pseudo voltage sources are only applied to *Q-V* realm blocks while that *Q-V* realm block is analyzed. That is, pseudo voltage sources are tentative entities in SAMOC simulation.
2. A pseudo voltage source is added to a *Q-V* realm device which has one node connected to a *Q-V* realm node and the other connected to an *I-V* realm node. The

added pseudo voltage source connects to the *I-V* realm node of the *Q-V* realm device as shown in Fig. 6.14(b).

3. When a pseudo voltage source is added, a posterior-prior relationship between a *Q-V* realm block and an *I-V* realm block needs to be set.

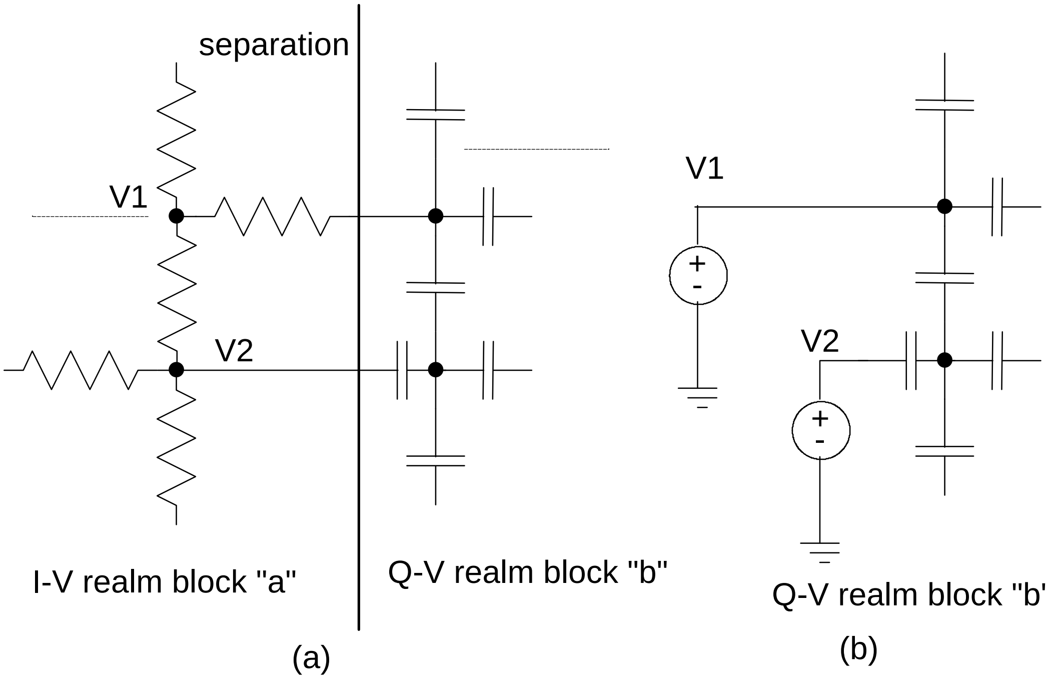


Fig. 6.14 Illustration of an interface between an *I-V* realm block and a *Q-V* realm block and pseudo voltage sources V1 and V2.

6.4.4 The *Q-V* realm Block Class

In C++ implementation of SAMOC, the *Q-V* realm block *class* that is used to represent *Q-V* realm blocks is very similar to the *I-V* realm *class*. A *Q-V* realm block contains a list of devices and a list of nodes. The lists are constructed by pointers which locate the address of device and the node *objects*. In addition to the device and the node lists, each block has two circuit block lists. These two lists, which contain pointers to other circuit blocks, are used to represent the posterior-prior relationships between blocks. One list locates the blocks which this block has to be analyzed prior to. The other list locates the blocks posterior to which this block must be analyzed.

6.5 Summary

The KVL circuit formulation and *conservation of charge* are used in SAMOC. The former is applied to *I*-*V* realm nodes of a circuit and the latter is applied to *Q*-*V* realm nodes. All nodes in a circuit are *Q*-*V* realm nodes unless they are connected with an *I*-*V* realm device, since an *I*-*V* realm device will inject or extract current into or from the node it connects. There is no steady current in *Q*-*V* realm analysis.

To improve the simulation efficiency, a circuit can be partitioned into several blocks and the circuit simulation can be achieved by analyzing circuit block individually instead of whole simulated circuit. The massive interconnecting data structure of SAMOC makes circuit partitioning of a circuit feasible and even easier. The circuit partitioning is realized by circuit block extraction and a circuit block is a portion of the simulated circuit which can be formulated and solved independently from other part of the simulated circuit. The basic process of circuit block extraction is to include a device from a node and then searches all devices incident with the other node(s) of the included device. The whole circuit block extraction consists of including devices and nodes alternately. Different types of devices have different rules of inclusion into a circuit block. There are two kinds of circuit blocks: *I-V* realm and *Q-V* realm. An *I-V* realm block is a resistive network and a *Q-V* realm block is a capacitive network.

The nodes of many kinds of devices such as controlled sources and MOS transistors can belong to different circuit blocks. In this situation, a posterior-prior relationship between two circuit block has to be established and represented by a directed graph. The direct graph of a circuit presenting the circuit blocks and signals between blocks is called the block-signal diagram. By this means, analyzing a circuit block becomes a very important mechanism in SAMOC. MNA is used in formulating circuit block as well as formulating the whole circuit as discussed in Chapter 2. For devices whose nodes can belong to different circuit block, some modification of the device stamps presented in Chapter 2 are set and presented in this chapter.

In addition to a device whose nodes may belong to different blocks, if a *Q-V* realm block connects with *I-V* realm block(s), then the other type of posterior-prior relationship is also created. In this situation, pseudo voltage sources have to be added to the *Q-V* realm nodes while the *Q-V* realm block is analyzed. After all posterior-prior relationships between circuit blocks are settled, a directed graph called block-signal diagram is created to represent and store the information. Accordingly, the device-node problem in circuit analysis is transfer into a block-signal problem. Chapter 7 will presents the simulation method based on the block-signal diagram.

Chapter 7

ANALYSIS OF INDIVIDUAL CIRCUIT BLOCK AND CIRCUIT SIMULATION VIA BLOCK-SIGNAL DIAGRAM

In addition to requiring less memory, analyzing a circuit individually by block can decrease the number of piecewise linear regions and accelerate the convergence of the Katzenelson algorithm. With the mechanisms of circuit block extraction and settlement of posterior-prior relationship between blocks, SAMOC transfers the device-node problem of circuit simulation into a block-signal problem. The directed block graph, called the block-signal diagram, represents the topology of a simulated circuit in a higher abstraction level. In the directed block graph, the vertices are circuit blocks and the edges are the interactive signals between blocks. The signal delay can be restored as a form of the weight of an edge. This chapter presents simulation schemes and ideas about analyzing a circuit from its block-signal diagram.

7.1 Analysis of an Individual Circuit Block

The basic process of simulation via a block-signal diagram is the analysis of individual block. There are two types of circuit blocks: an *I-V* realm block and a *Q-V* realm block. In the C++ implementation of SAMOC, two *classes*, which are categorized as the solving *classes*, are designed to handle and manage the analysis tasks for two types of circuit blocks. The tasks defined in the solving *classes* are memory acquisition, circuit equation formulation and solving the equations. In a resources-limited computer system, an *object* of a solving *class* is created when analyzing a block is required, and the created *object* is destroyed after the block is analyzed. By destroying an *object* of a solving *class* after block analysis, the computer system can recycle and reuse the computer’s resources efficiently.

7.1.1 Solving *Class* for *I-V* realm block analysis

In SAMOC, there is a *class* named `CSolveIVCB` (*class* for solving an *I-V* realm block) which is designed to execute and manage the equation generating and solving procedures and memory management tasks. An *object* of `CSolveIVCB` is created when an *I-V* realm circuit block is to be analyzed. After the analysis, the created *object* is destroyed and the memory is released back to the system. The tasks which need to be done during the life period of an object of the `CSolveIVCB` are:

1. **Read the circuit block's device data and node voltages:** Each circuit block has a device list and a node list. The device list and the node list offer the circuit block topology information. Furthermore, the node list contains the initial conditions before the circuit block is analyzed. The initial conditions are used as the initial guess of the iterative algorithm.
2. **Estimate the required memory for MNA analysis:** The required memory for solving the *I-V* block consists of $N^2 + 2N$ double precision words. N^2 is for the modified nodal equation matrix and $2N$ is for the source (excitation) vector and the solution vector. N is estimated using (2.1), but the devices and nodes are limited to those included in the analyzed circuit block only.
3. **Formulate the circuit equations by filling in the device stamps:** The MNA analysis presented in Chapters 2 and 3 is applicable to the circuit blocks as well as to the whole circuit. Software implementation of using the device stamps exploits the virtual functions of the C++ language. Virtual functions also make the coding of program brief and each virtual function can be updated independently.
4. **Determine the solution of the MNA matrix:** Since the piecewise linear approach is employed to model MOS transistors and ideal diodes, the Katzenelson algorithm is employed to determine the solution. Virtual function techniques in C++ programming are used to find the scaling factor for each

piecewise linear approached device. The smallest scaling factor, t_{min}, is chosen by comparing scaling factors of all piecewise linear devices. t_{min} is essential to the Katzenelson algorithm.

5. **Update the solution:** The data in the solution vector are written back to the main node list of th**e** circuit via the node list owned by the circuit block.

6. **Release the acquired memory and destroy the solving *object*:** If the *object* of `CSolveIVCB` *class* is created in stack, then the *object* will be automatically destroyed while the program runs out of scope. If the *object* of `CSolveIVCB` *class* is created in heap by command `"new"`, then the *object* have to be destroyed by the command `"delete"` and free the memory. For a resource-limited computer system, creating the *object* of `CSolveIVCB` *class* in stack can save memory. For a speed-demanding occasion, creating the *object* of `CSolveIVCB` *class* in heap can save time.

7.1.2 Solving *Class* for *Q-V* realm block analysis

A *class* named `CSolveQVCB` (*class* for solving a *Q-V* circuit block) is designed to solve Q-V realm blocks. The `CSolveQVCB` *class* is very similar to `CSolveIVCB` presented above. The main differences are:

1. A *Q-V* realm block is a memory unit and collecting initial conditions (most of time the nodes' voltage values) is the essential step in formulating *Q-V* equations.

2. Pseudo voltage sources have to be added if the *Q-V* realm block connects with *I-V* realm blocks. Since the purpose of adding pseudo voltage sources is for solution seeking only, the added pseudo voltage sources are destroyed along with the *object* of the `CSolveQVCB` *class*.
3. There is no piecewise linear device in a *Q-V* realm block. The solution can be obtained by direct method instead of the Katzenelson algorithm in *I-V* block analysis.

In SAMOC, a "simulation queue" is created to handle and manage the sequence of block analysis. The queue is implemented by a list of pointers and the pointers locate the addresses of circuit blocks. The order in the queue is determined by the rank numbers of the circuit blocks which are presented in section 7.4. Fig. 7.1 illustrates a simulation queue created for the circuit block graph presented in Fig. 6.8. The block analysis proceeds from the head to the tail of the queue. To analyze a block (for example, circuit block "a" in Fig. 7.1) a solving *object* of `CSolveIVCB` or `CSolveQVCB` *class* is created to process the block's device and node lists, read voltage values from the main node list, acquire memory, formulate the circuit equations of the circuit block, solve the circuit equations, update data back to the main node list and release the acquired memory. After that, the solving class is destroyed and the same procedure takes place at the next circuit block "b". The DC analysis of the whole simulated circuit is finish while the last circuit block of the simulation queue is analyzed.

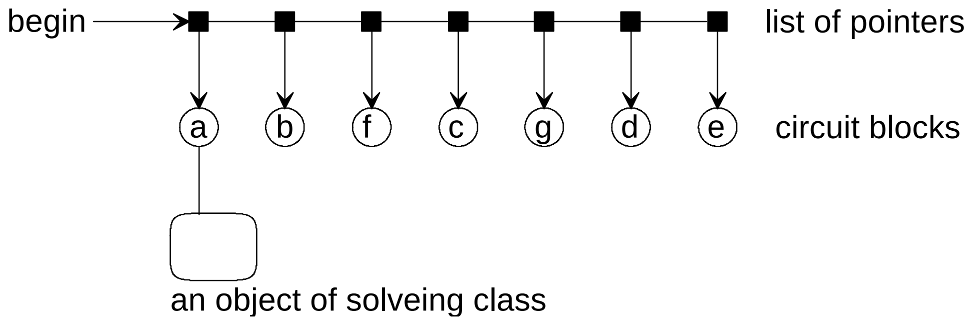


Fig. 7.1 A simulation queue.

7.2 MOS-Capacitor Modeling for *Q-V* Realm Analysis

The most important device of a *Q-V* realm block is a capacitive device. A capacitive device can either be directly specified by a circuit description as a discrete capacitor or embedded in a MOS transistor as a parasitic capacitor. Since MOS transistors are the most important devices in SAMOC simulation, the modeling of parasitic capacitances of MOS transistors is very important for the Q-V realm analysis of a circuit.

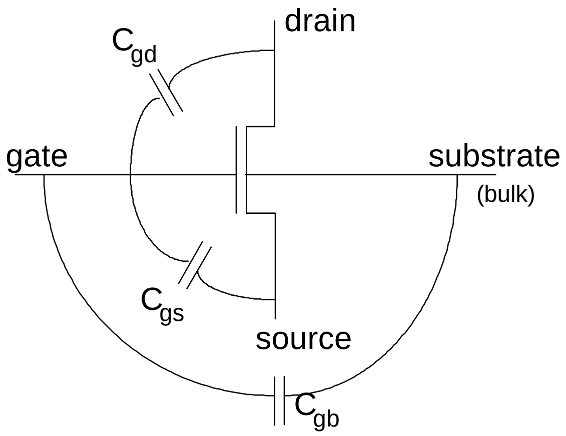


Fig. 7.2 Parasite capacitance of a MOS transistor.

Fig. 7.2 illustrates the parasitic capacitance model of a MOS transistor. The major capacitances in this model are the gate-to-drain capacitor, C_{gd}, gate-to-source capacitor,

C_{gs} and gate-to-substrate capacitor, C_{gb}. In [72] the values of these capacitors depend on the states of the MOS transistor and are illustrated in table 7.1, where

$\varepsilon = \varepsilon_0\ \varepsilon_{SiO2}$

ε_0 = permittivity of free space

ε_{SiO2} = dielectric constant of SiO_2

A = gate area

t_{ox} = thickness of the SiO_2 layer under the gate.

Table 7.1 Gate capacitors of a MOS transistor.

capacitor	state of the MOS transistor		
	cutoff	linear	saturated
C_{gb}	$\frac{eA}{t_{ox}}$	0	0
C_{gs}	0	$\frac{eA}{2t_{ox}}$	$\frac{2eA}{3t_{ox}}$
C_{gd}	0	$\frac{eA}{2t_{ox}}$	0

The gate of a MOS transistor is usually connected with a resistive network composed of drains or sources of other MOS transistors. If the gate of a MOS transistor connects with an *I-V* realm node, then the gate capacitors can be discarded in the DC analysis of the entire circuit. If the gate of a MOS transistor is connected to a *Q-V* realm node, as illustrated in Fig. 7.3 (a), then the gate capacitors, C_{gb}, C_{gd} and C_{gs}, will participate in the *Q-V* realm analysis for this node as illustrated in Fig. 7.3(b).

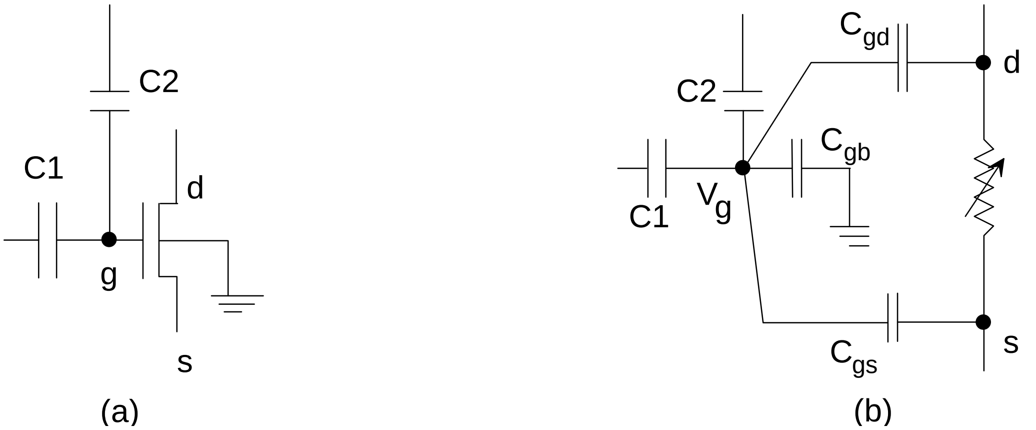


Fig. 7.3 A MOS transistor with its gate connecting to a Q-V realm node.

Since the gate voltage V_g in Fig. 7.3(b) is influenced by the values of gate capacitors, C_{gb}, C_{gd} and C_{gs}, the state of the MOS transistor is determined by the gate-to-source voltage V_{gs}. The values of the gate capacitors C_{gb}, C_{gd} and C_{gs}, are different at different states of the MOS transistor. For this reason, the *Q-V* realm node g must be analyzed together with the *I-V* nodes d and s. In SAMOC circuit partitioning the circuit will be separated into several *I-V* and *Q-V* realm blocks. For a situation like the one shown in Fig. 7.3 (a), two or more blocks will be analyzed using the same set of equations. That is, SAMOC creates a current equation for every *I-V* realm node and a charge equation for every *Q-V* realm node. In this scenario, the gate capacitor of a MOS transistor is state dependent. The modeling of gate capacitors is a piecewise constant as shown in Table 7.1, and the Katzenelson algorithm is applied with a limit on the correcting vector both in *I-V* and *Q-V* realm analyses simultaneously.

7.3 Mixed Mode Solving Class and Modification of Simulation Queue

As discussed in Section 7.2, the MOS capacitors DC analysis may include more than one block at a time, and the group of circuit blocks must contain both *Q-V* realm and *I-V* realm blocks. In this situation, a new *class* has to be designed to handle the multi-block analysis. In addition, the ranking order of circuit blocks which is set may be changed according to the multi-block analysis at one simulation instance. The analysis ranking change would result in a rescheduling of block analysis, and hence a modification of the simulation queue is necessary.

7.3.1 Mixed Mode Multi-Block Solving *Class*

In order to analyze more than one circuit block which are both *I-V* and *Q-V* realm, a new *class* is designed. The processes induces reading data from the main node list (belongs to an *object* of the *class* `CCircuit`), acquiring memory, formulating circuit equations, applying the Katzenelson algorithm, etc, which are exactly the same as which are defined by `CSolveIVCB`. In C++ programming implementation, the new class, `CSolveMixModMCB` shown in Fig. 7.4, can be derived from the existing *class*, `CSolveIVCB.` One list of circuit blocks is added to `CSolveMixModMCB` in order to locate the circuit blocks which need to be analyzed together.

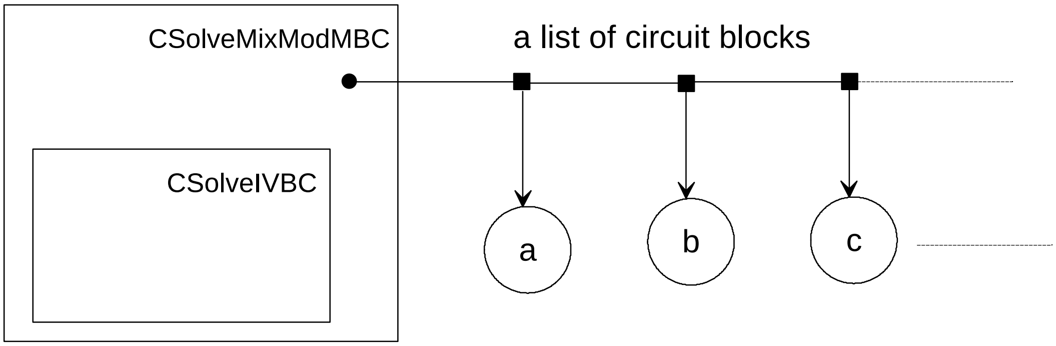


Fig. 7.4 A *class* designed for analyzing multi-block is derived from `CSolveIVCB`

In addition to a list of circuit blocks, new lists of devices and nodes are added to make the procedure defined in `CSolveIVCB` process the same data structures. The added list of devices is the summation of the lists of the circuit blocks which are located by the list of circuit block of the `CSolveMixModMCB`. Besides augmenting the new device list by summing up device lists of the linked circuit blocks, some devices, such as a shareable device, might appear in more than one circuit block. In this situation, a procedure illustrated in Fig. 7.5 is added to remove the redundancy. After the new device lists is built, the new node list can be built according to the new device list. Since all devices contain node lists, the building of the new node list does not cause much effort. Fig. 7.5 illustratesa simple example of building a new device list from two device lists. Fig. 7.5(a) shows two duplicated device lists of two circuit blocks. Fig. 7.5 (b) shows how the new list is created by adding one list to the other list and removing such repeated devices as “e” and “a”. Fig. 7.5 (c) shows the new device list which is ready for circuit block analysis.

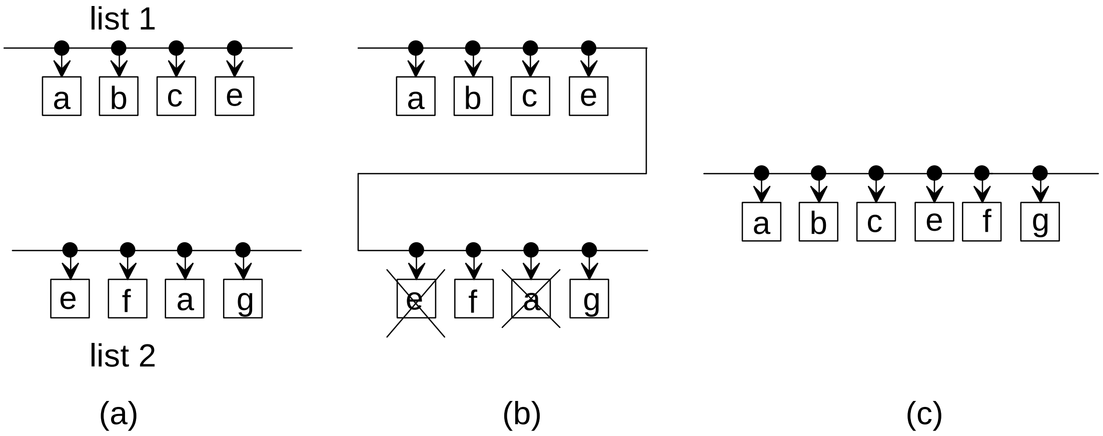


Fig. 7.5 Building of a new device list from two device lists.

7.3.2 Modification of the Simulation Queue

The multi-block analysis which uses more than one block at a simulation time instance, will change the established posterior-prior relationship. Consider an example, a circuit with block graph shown in Fig. 6.8. If a Q-V realm node in the block “g” connects to the gate of a MOS transistor in the I-V block “d”, then one voltage of Q-V realm block “g” influences the state of a MOS transistor in I-V block “d”, and the state of the MOS transistor influences the gate capacitors value in the Q-V realm block “g”, (Fig. 7.6). Hence, circuit blocks “g” and “d” should be analyzed together, and a multi-block analysis has to be performed.

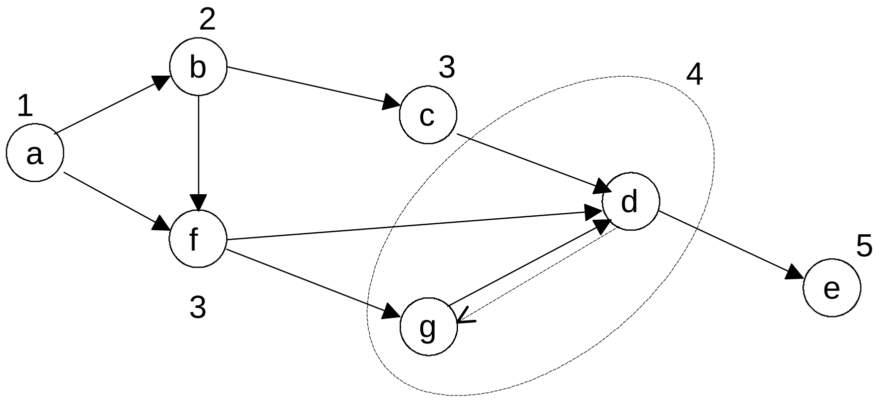


Fig. 7.6 A multi-block analysis of two circuit blocks “g” and “d”.

The multi-block analysis not only requires the creation of a solving *object* of `CSolveMixModMCB` which would take care of the analysis task, but also influences the simulation queue. Fig. 7.7 illustrates the modification caused by the multi-block analysis of blocks “g” and “d”.

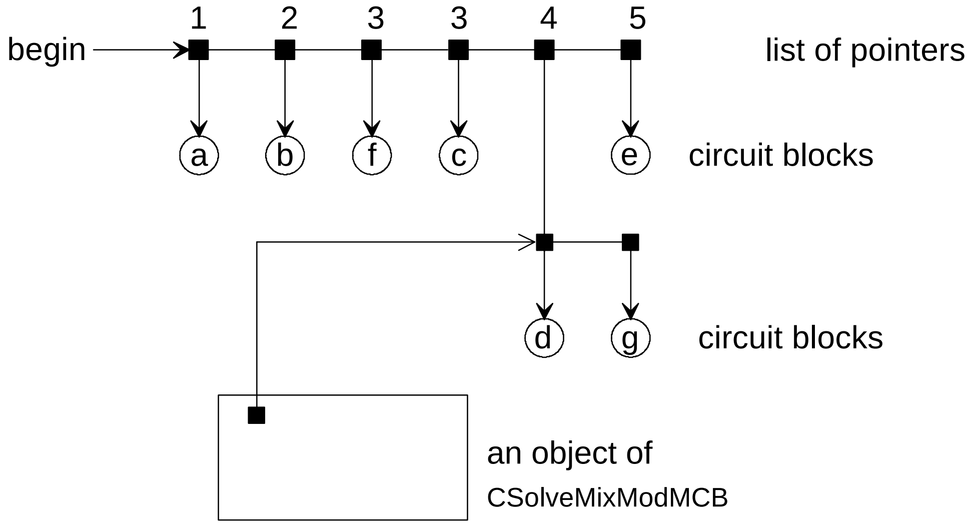


Fig. 7.7 The modification of the simulation queue.

The multi-block analysis is not limited to interacting circuit blocks. In an example illustrated in Fig. 7.8, blocks “b” and “g” are interacting but the multi-block analysis must also involve block “f” which was scheduled to be analyzed between “b” and “g”.

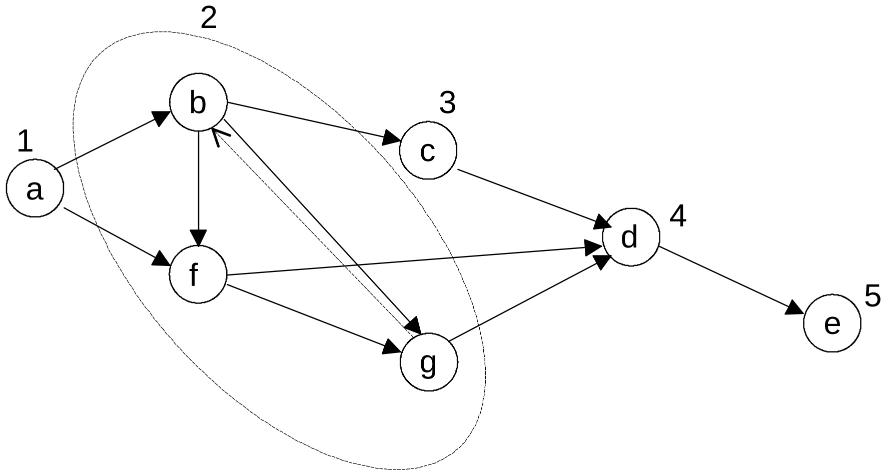


Fig. 7.8 A multi-block analysis of three circuit blocks “b”, “f” and “g”.

7.4 Simulating a Circuit with an Acyclic Block-Signal Diagram

Finding the operating state of every semiconductor device in the simulated circuit is the first step of circuit simulation. One direct and simple way to find out these operating states is to analyze all circuit blocks of the circuit. There are several approaches to setting the order of blocks to be analyzed. For a circuit whose block-signal diagram is acyclic, the most economic method which costs least computation effort is to set the order number for each block, and analyze blocks from a block with lower order numbers to a block with higher number. A block’s order number is the topology depth of that block in the block-signal diagram.

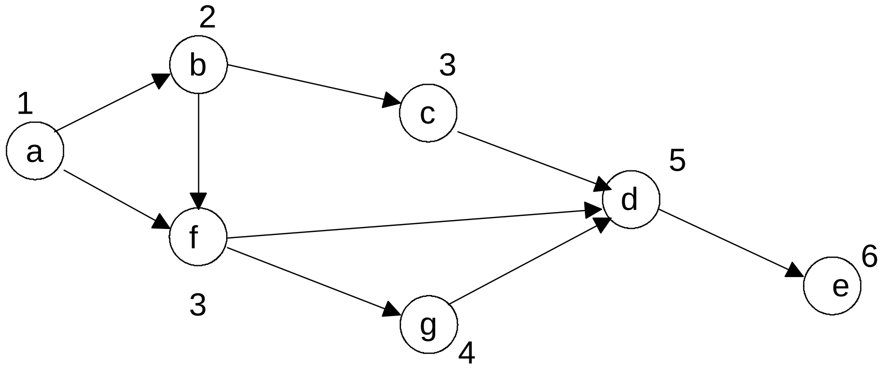


Fig. 7.9 Order numbers of blocks in the acyclic block-signal diagram.

7.4.1 Setting the Order Number of Block in the Block-Signal Diagram

Setting of the order number in SAMOC begins with setting all rank numbers to "1". SAMOC then chooses one block and adds 1 to the number of all the blocks that the one has to be analyzed prior to. The same procedure is applied to each block that has its rank number changed. The procedure ends by checking if the rank number of all blocks are larger than the blocks they are posterior to. The simulation queue can be built on the order number. The sequence in the queue is arranged by the blocks order number and is the analyzing order of the block. For the block-signal diagram shown in Fig. 7.9, the simulation queue is illustrated by Fig. 7.1 and the analyzing order can be `{a -> b -> f -> c ->g -> d -> e }` or `{a -> b -> c -> f ->g -> d -> e }`. DC analysis is attained by analyzing all blocks with the sequence set by the simulation queue.

7.4.2 The Activated Blocks

After the DC analysis is performed and the operating points of all nonlinear devices are found, SAMOC is ready for the time varying excitation vector. The time varying excitation vector could directly influence the solution vector which is obtained by previous analysis and change the states of semiconductor devices of a block. In this situation, the block, whose solution vector and device states are about to change, is called an activated block. In SAMOC, a new *object* of solving *class* is created to estimate the new solution vector. The new solution vector compares with the old one before it is written back to the main node list. If the new solution vector is different than the old one, the activated block fires at its posterior blocks. An example is illustrated in Fig. 7.10. Block "b" is the activated block. If there is any change in "b's" solution vector, "b" fires at block "f" and "c". Block "f" and "c" are the fired blocks.

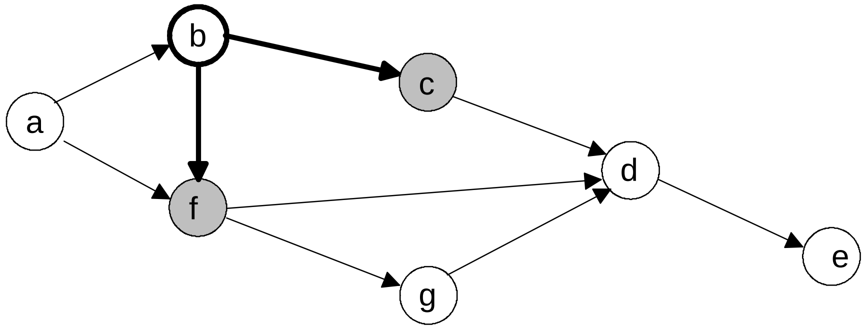


Fig. 7.10 A block fires at two posterior blocks.

7.4.3 From a Fired Block to an Activated Block and Definition of Events

A fired block is a block which has at least one prior block having a new solution vector. A fired block may not necessarily become an activated block, since the change of the prior block may not really influence the fired block. A simulation event happens when a fired block becomes an activated block. The occurrence of an simulation event is decided by the linking device between the fired block and the prior block which fired at it. Three types of device would cause posterior-prior relationship between blocks and they are controlled sources, voltage controlled switches and MOS transistors.

1. Controlled sources: The nodes of a controlled source can belong to more than one block. According to the setup rule of posterior-prior relationship between blocks, the controlling node(s) belong to the prior block which is also the activated block and the controlled nodes belong to the fired block. If a fired block links an activated block with a controlled source and the activated block has a changed solution vector, then the fired block becomes activated and a simulation event is induced. Fig. 7.11 illustrates a posterior-prior relationship between 5 blocks induced by a voltage controlled voltage source (VCVS). Blocks “a” and “b” contain two controlling nodes of the VCVS. Blocks “c”, ”d” and “e” have the output node of the VCVS, since output nodes of the VCVS are sharable nodes. the posterior-prior relationship between these 5 blocks are: Block “a” is prior to “c”, “d” and “e”. Block “b” is prior to “c”, “d” and “e”. There is no posterior-prior relationship between “a” and “b”. If “a” is activated and has a changed solution vector, “a” fires at “c”, “d” and “e” and “c”, “d” and “e” are also activated. The same scenario is also applicable to “b” as to “a”.

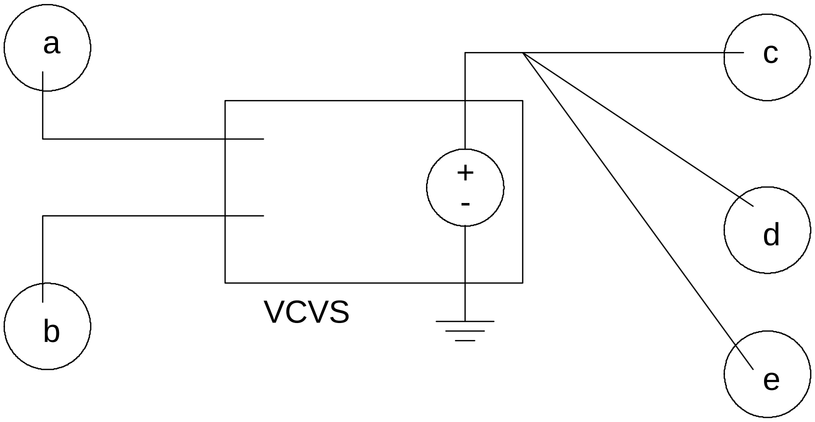


Fig. 7.11 A posterior-prior relationship between 5 blocks.

2. Voltage controlled switches: For a voltage controlled switch, the controlling nodes can belong to the activated block(s) and the switch belongs to the fired block. If a fired block links an activated block with a controlled source and the activated block has a changed solution vector, then the fired block becomes activated only when the changed solution vector changes the switch’s state. If the changed solution vector does not turn on a previously open switch or turn off a previously closed switch, then there is no simulation event and the fired block does not become an activated block. Fig. 7.12 illustrates a posterior-prior relationship between two blocks “a” and “b” and the relationship is induced by a voltage controlled switch, Sx. Block “a” has the controlling nodes, V1 and V2, and “b” has the switch. In simulation order, block “a” is prior to “b”. If block “a” has a changed solution vector, then “a” fires at “b”. Block “b” becomes activated only when the change of V1 and V2 cause the switching state of Sx.

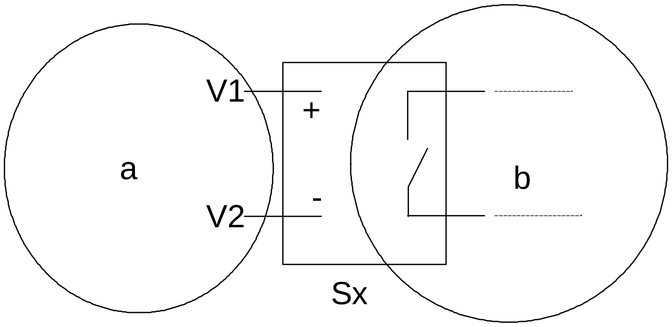


Fig. 7.12 A posterior-prior relationship induced by a voltage controlled switch.

3. MOS transistors: A MOS transistor has three nodes and three linear regions. If a MOS transistor causes a posterior-prior relationship between blocks, then the gate must belong to the prior block and the source and drain belong to the posterior block. That is: the prior (activated) block has V_g and the posterior (fired) block has V_d and V_s. According to the piecewise linear model presented in Chapter 2, while the MOS transistor is in the saturated state, a voltage controlled current source, whose controlling node is the gate, is used for device characteristic representation. Therefore, if the MOS transistor is in the saturated state and V_g is changed because the prior block is activated, then the posterior block is fired and activated. If the MOS transistor is in cutoff state or linear state, then the I_{ds}-V_{ds} characteristic is modeled by a resistor only. The fired (posterior) block is activated only when the change in V_g causes the state change of the MOS transistor.

An interface between I-V and Q-V realm blocks also induces posterior-prior relationship blocks. The I-V realm block is the prior and the Q-V realm block is the posterior. In this situation, if the prior block is activated, the posterior block is also activated.

7.4.4 Computer Implementation of Activated and Fired Block

Two Boolean variables, `m_bActivated` and `m_bFired` are added in to the circuit block class `CCircuitBlock`. Before the DC analysis subroutine is applied to the simulated circuit, `m_bActivated`s of all blocks are set to *TRUE* and `m_bFired` are set to *FALSE*. `m_bActivated` of a block is set to *FALSE* after the analysis. Before the analysis results are written back to the main node list, SAMOC compares the new solutions with the old one. If there is any difference between the old and new results, then all posterior blocks are fired and this operating is executed by setting the `m_bFired` of each posterior block *TRUE*.

For a fired blocks with `m_bFired` *TRUE*, if its `m_bActivated` has been *TRUE*, then there is no further action in this part, since a block can be activated by many causes and an activated block won't turn to an inactivated. with a block analysis If its `m_bActivated` is *FALSE*, then a procedure following the description in section 7.4.3 is executed to decide the fired block becomes activated and set the new values of `m_bActivated`. `m_bFired` is set back to *FALSE* after the decision making.

The fired-activated process may cause a chain reaction inside the block-signal diagram. The chain may be broken by an activated block that does not fire at its posterior blocks or a fired block that does not become activated.

The circuit analysis according to time varying excitation is accomplished by analyzing all activated block with the order set by event queue. One variation in excitation vector requires one walk through the simulation queue and analysis of each activated block.

7.5 Timing Analysis of a Circuit with a Cyclic Block-Signal Diagram

For all circuits with feedback loops, block-signal diagrams are cyclic. For a cyclic block-signal diagram, the algorithm used to set the order numbers of blocks for an acyclic block-signal diagram will result in an infinite loop. Building a simulation queue according to the ordered blocks becomes unfeasible. One simple approach to simulate a circuit with a cyclic block-signal diagram is to assume that there is a signal delay between every two blocks in the block-signal diagram. By this assumption, an integer number called topological depth is assigned to every block in the diagram. A block must have larger topological depth than its prior blocks. The rules of setting the topological depths of blocks are:

1. For blocks without any prior block, their topological depth is 0.
2. For blocks which have time varying device such as time varying voltage source, their topological depth is 1.

3. The topological depths of the other blocks are the shortest directed path from any block whose topological depth is equal to 1.

4. If there are still any undetermined blocks, their topological depths are set to 1.

Fig. 7.13 illustrates an example of setting the topological depths of blocks. In Fig. 7.13, block “A” does not have any prior block, thus its topological depth is 0. Blocks which have 0 topological depth represents block without any time varying signals. It also implies that they are static blocks and they only need to be analyzed once. Blocks which contain (or are directed fired by) time varying devices, such as blocks “B”, “C” and “D”, are the sources of time varying signals in the block-signal diagram. Other blocks which accept signals from blocks “B”, “C” and “D” have their topological depths equal to the shortest distance from blocks “B”, “C” and “D”. Blocks such as “M”, “N”, “O”, “P” and “Q” have no path from level 1 blocks. Although the time varying signals will never propagate to these blocks, they can possibly create events to influence the simulated circuit. For a block of this kind, the topological depths is set to 1 in order to make it influence as many blocks as it can. The topological depths of its posterior blocks is not measured by the shortest directed path from it, because the simulation discussed here is driven by time varying input signals.

A simulation queue built based on the topological depths of blocks is illustrated on Fig. 7.14. At each time instance of the timing simulation, all blocks with topological depths equal to 1 will be analyzed only if the time varying input signals cause any event, and only if the block analysis is necessary. If the new analyzed results are different from the ones at the node list, then the analyzed blocks fire at their posterior blocks. After all blocks with topological depth equal to 1 are analyzed, the same process will be applied to all blocks with topological depth equal to 2 and so on.

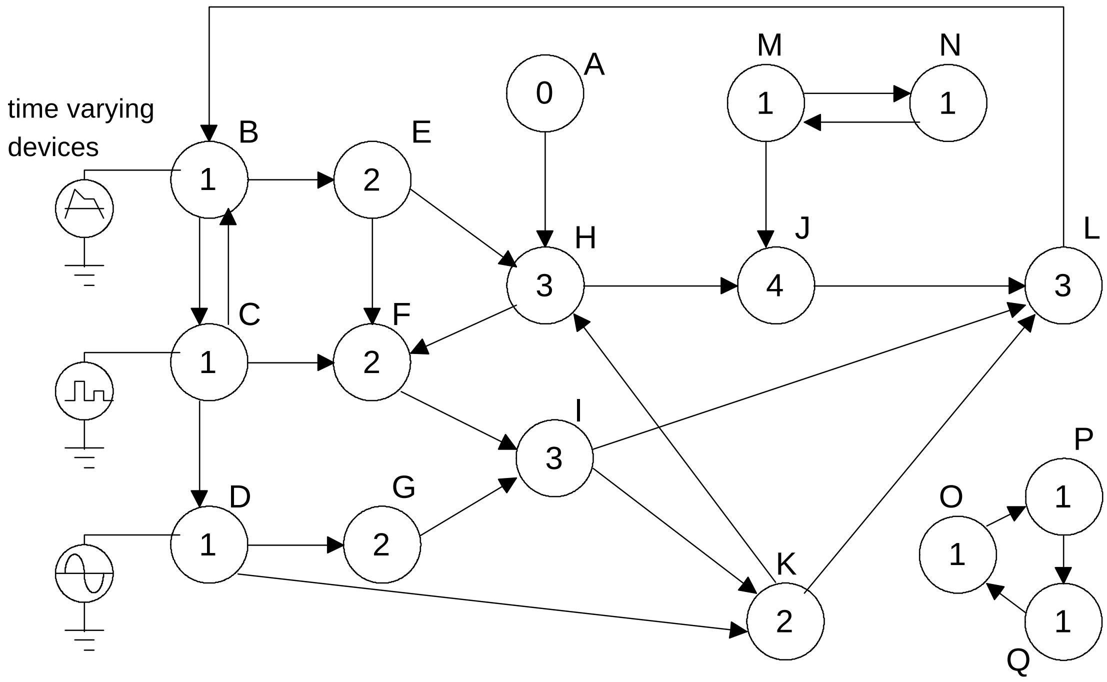


Fig. 7.13 The topological depth setting of a block-signal diagram.

The topological depth of a block implies the virtual time index of when the time varying signal will propagate to that block. A block might fire at other blocks with smaller topological depth. A possible effect of this firing will be taken in to account at the next iteration, since all signal transferring have the same delay time.

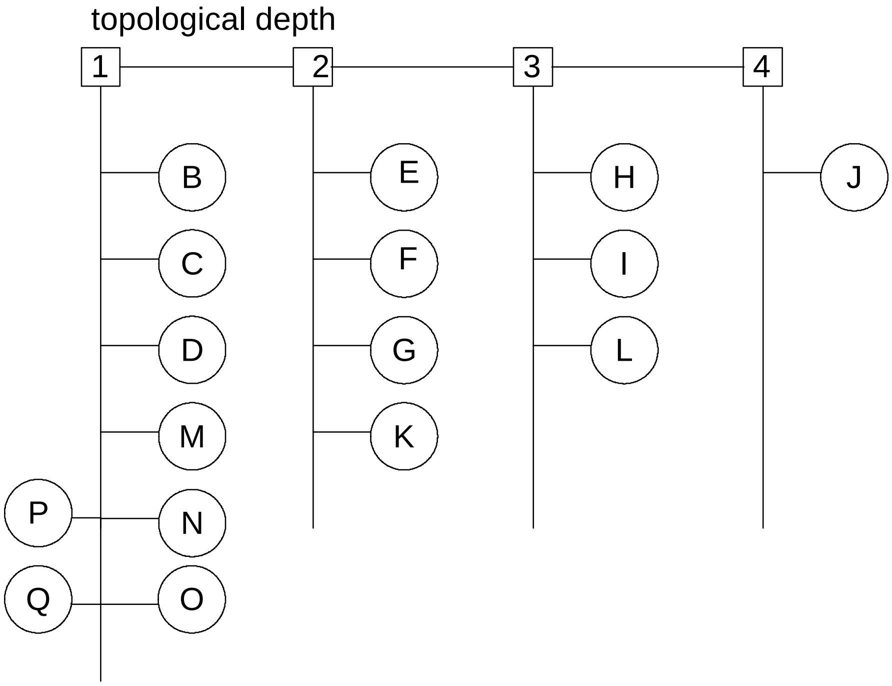


Fig. 7.14 The simulation queue.

7.6 Summary

After the block-signal diagram is built by the methods presented in Chapter 6, SAMOC analyzes the simulated circuit on the block-signal level instead of the device-node level. Analyzing individual blocks becomes an important and fundamental mechanism for block-signal analysis. There are two types of blocks: *I*-*V* and *Q*-*V* realm blocks. SAMOC employs MNA to both types of blocks for circuit equation formulation. Solving an *I*-*V* realm block requires the Katzenelson algorithm, which needs a subroutine to update the equation set and to control the iteration loops. On the other hand, a linear method can be applied to solve *Q*-*V* realm blocks, which means that only one circuit formulation subroutine is required. In some cases, pseudo voltage sources are added to a *Q-V* realm block which is driven by *I-V* realm blocks. In C++ implementation, two solving *classes* according to different kinds of solving method, are designed to do the memory acquiring, circuit formulation and numerical solution-seeking tasks.

For each analyzed block, an *object* of the solving *class* is created to do the block analysis task. After the analysis, the created *object* of the solving *class* is destroyed and hence the system resource acquired by the *object* of the solving *class* is returned back to the system for another *object* of the solving *class*. *Object*s of the solving *class* can be created and destroyed during the run time of SAMOC. The solving *class*, therefore, is a run-time *class*. In MFC or other advanced C++ class libraries, run-time *class* can be handled by multithread programming which can be transplanted to and exploit the advantage of parallel computing in a multiprocessor environment effortlessly.

An *object* of the solving *class* reads the voltage data from the main node list as the initial conditions. After the circuit block is analyzed, the new solutions are written back to the main node list before the *object* of the solving *class* is destroyed. In data structure design of SAMOC, the main node list is the interactive media between circuit blocks.

Although in DC analysis of a circuit, a capacitor is treated as an open circuit and will not effect the results of *I-V* analysis, the nodes of *Q-V* realm blocks may be the controlling nodes of a voltage controlled sources or may be connected to gates of MOS transistors. For this reason, extracting and analyzing *Q-V* realm blocks are important in DC analysis.

If the gate of a MOS transistor in an *I-V* realm block is driven by a *Q-V* realm block via a capacitive link, multi-block analysis becomes inevitable. Multi-block analysis not only contains mixed modes (*Q-V* realm and *I-V* realm) formulation, but also stimulates a topological change in block-signal diagram. The existence of multi-block analysis reduces simulation efficiency in circuit analysis by block method. In C++ implementation, a new class which is derived from solving classes of *I-V* realm block is designed to do the mixed mode multi-block analysis.

In simulating a circuit from its block-signal diagram, a simulation queue is built from the posterior-prior relationship between blocks to manage the block simulation. A block is activated if its input nodes have a changed input signal or its device state was changed by any of its prior blocks. An *object* of solving *class* is created to execute the block analysis tasks for each activated block. Before the new solutions of the block are written back to the main node list, SAMOC checks if any solution changed. If any of them did, then the analyzed block fires at all its inactivated posterior blocks. SAMOC then checks all fired blocks, if they are activated by the change in the main node list.

DC analysis requires all blocks in the block-signal diagram to be analyzed. For a block-signal diagram without feedback loops, a simulation queue can be easily created according to the order number of each block in the diagram. The order number of each block in the block-signal diagram is the same as the depth of the vertex in an acyclic directed graph. The DC analysis is therefore attained by analyzing all blocks in the block-signal diagram with the order managed by the simulation queue.

To handle the dynamic circuit analysis, which is caused by time varying excitations, SAMOC adds two Boolean variables, thus suppressing insignificant block analysis. One variable indicates if the block is activated and the other indicates if the block is fired upon, since not all fired blocks will become activated. In dynamic analysis, SAMOC only analyzes the activated blocks. A block is activated because its inputs are changed or any of its MOS transistors or controlled switches have a changed state. An analyzed block fires at its inactive posterior blocks if the analyzed block caused a change in the main node list. A fired block becomes active if any of its device changes its state and the change is induced by the change of data in the main node list.

For a circuit whose block-signal diagram has feedback loops, the block-signal diagram is cyclic and it costs more computation effort to create and maintain a simulation queue. An approach which can be easily implemented to analyze a circuit with a cyclic block-signal diagram is to assume all signals between blocks have the same delay time and assign a topological depth to each block. The smaller the topological depth, the earlier the time varying signals propagate to the block. That means the block should be analyzed prior to the blocks having larger topological depth. By applying this method, the time step should be small enough to take care of the signals of internal loops. This approach can be applied to both acyclic and cyclic block-signal diagrams.

Chapter 8

SAMOC APPLICATIONS II: EVENT DRIVEN TIMING SIMULATION EFFICIENCY ANALYSIS AND BENCHMARK CIRCUIT SIMULATION

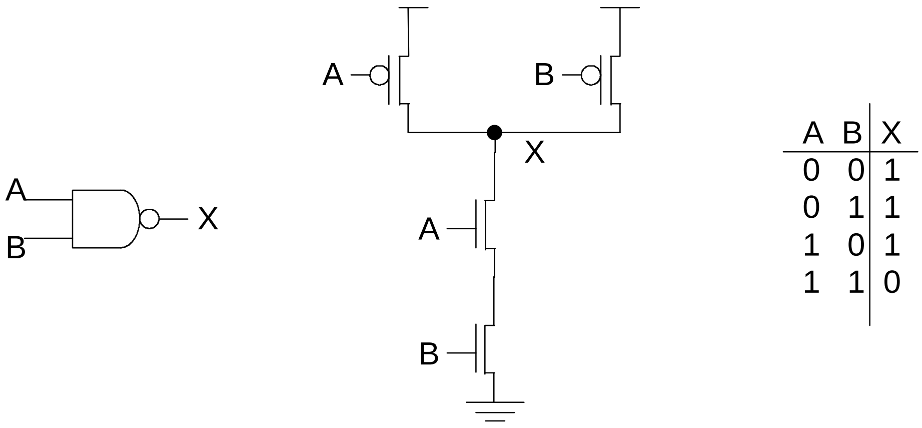
This chapter presents the results of benchmark circuits simulation performed by the SAMOC program and efficiency improvement evaluation of the event-driven timing simulation. Simulation results are presented in the following order: small CMOS sequential logic circuits with 40 to 150 MOS transistors; MSI CMOS logic circuits with 200 to 300 MOS transistors; VLSI benchmark circuits with over 10,000 MOS transistors; small analog circuits with 10 to 50 MOS transistors; and large analog circuits.

8.1 Small CMOS Sequential Logic Circuits

Although functional simulation of logic circuits can be practiced at higher abstraction levels such as at the gate level and register transfer level, timing analysis of sequential logic circuits at the circuit level is a very good way to validate the simulator. Simulation of small sequential circuit is the first step in verifying the simulation mechanisms built in SAMOC. These include validation of the MOS model for timing analysis, circuit block analysis and the timing analysis scheme.

8.1.1 CMOS NAND gate

CMOS NAND gates can be used in the construction of any logic circuit. Most of the logic circuits simulated presented in this chapter are built on CMOS NAND gates. Fig. 8.1 illustrates a CMOS implementation of a NAND gate, where Fig. 8.1 (a) shows the logic symbol of a NAND gate in a logic circuit, Fig. 8.1 (b) shows the CMOS schematic of the NAND gate and Fig. 8.1 (c) shows the truth table. In the truth table, symbol 1 represents high voltage and symbol 0 represents low voltage. In the circuit level simulation presented in this chapter, 5 voltage logic is used.



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(a) (b) (c)

Fig. 8.1 CMOS NAND gate.

8.1.2 Positive Edge-Triggered D Flip-flop

D flip-flop shown by Fig. 8.2 is an essential device to construct data registers for large logic systems and it is constructed of 3 negative R-S flip-flops. Simulating the timing behavior of a positive edge-triggered *D* flip-flop exemplifies to examining SAMOC's cyclic block-signal diagram analysis.

SAMOC partitions the positive edge-triggered *D* flip-flop, which contains 29 devices and 20 nodes, into 6 blocks each containing one NAND gate. In the simulation queue, gates N2, N3 and N4 have a topological depth equal to 1 and gates N1, N5 and N6 have a topological depth equal to 2. The depth of the whole circuit is 2. Fig. 8.3 illustrates the timing simulation results of the positive edge-triggered *D* flip-flop as a function of the simulation time. There are 200 simulation instances in the results shown in Fig. 8.3. The first plot is the clock signal. The second plot is the input signal D. The third plot is the output Q of the positive edge-triggered *D* flip-flop; the plot shows that the simulation results are as expected of a correctly designed flip-flop. The fourth plots shows the number of blocks (n) which were analyzed during each simulation instance. If there is no event-driven simulation, then all 6 blocks should be analyzed during each simulation instance. The summation of n is 128, which means, on average, 0.64 blocks were analyzed at each simulation instance. Event driven simulation for timing analysis of the positive edge-triggered *D* flip-flop saves about 89.33% of the block analysis.

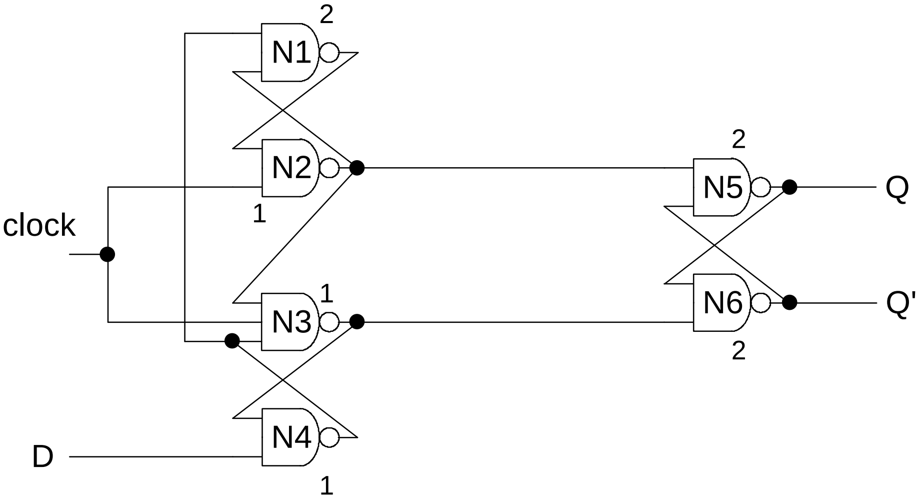


Fig. 8.2 Positive edge-triggered *D* flip-flop.

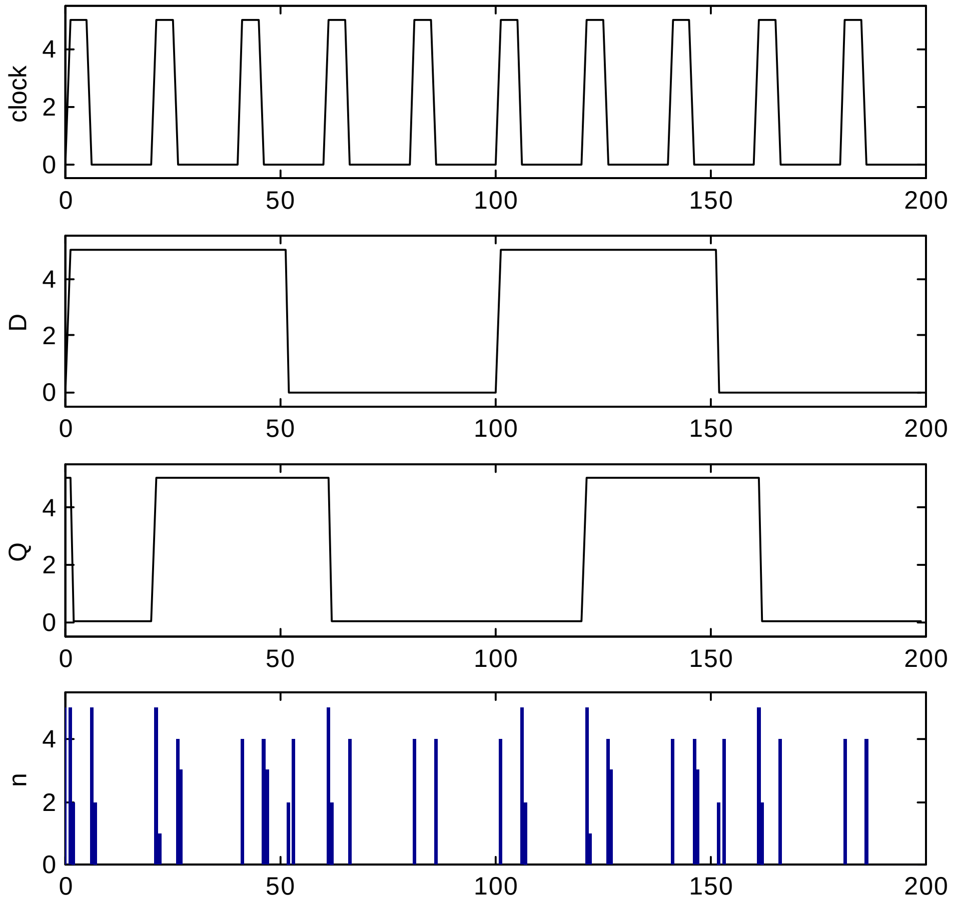


Fig. 8.3 The timing simulation results of the positive edge-triggered *D* flip-flop.

8.1.3 Master-Slave *J-K* Flip-flop

J-K flip-flops and *D* flip-flops are useful in the construction of large logic systems. Fig. 8.4 shows a master-slave *J-K* flip-flop which was implemented by 8 CMOS NAND gates. The SAMOC simulation shows that the master-slave *J-K* flip-flop contains 40 devices, 27 nodes and 8 blocks. The topological depth of the master-slave *J-K* flip-flop is 3. The topological depth of each block is marked in Fig. 8.5.

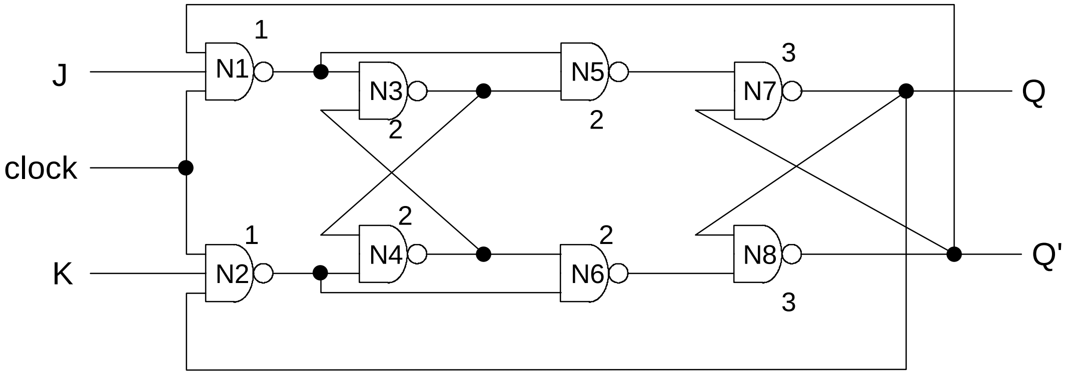


Fig. 8.5 Master-Slave *J-K* flip-flop

SAMOC timing analysis results are shown in Fig. 8.6. Similar to the results shown in Fig. 8.4, there are 200 simulation instances in the presented results. The functional verification of the master-slave *J-K* flip-flop can be practiced by observing the fourth plot of Fig. 8.6. The fifth plot of Fig. 8.6 shows the computational effort as a function of the simulation instance. There were 173 blocks analyzed during the 200 simulation instances. This means, on average, 0.865 block among 8 blocks is analyzed in each simulation instance. The event-driven simulation shows that 89.19% of block analysis was saved compared to simulation without event-driven approach.

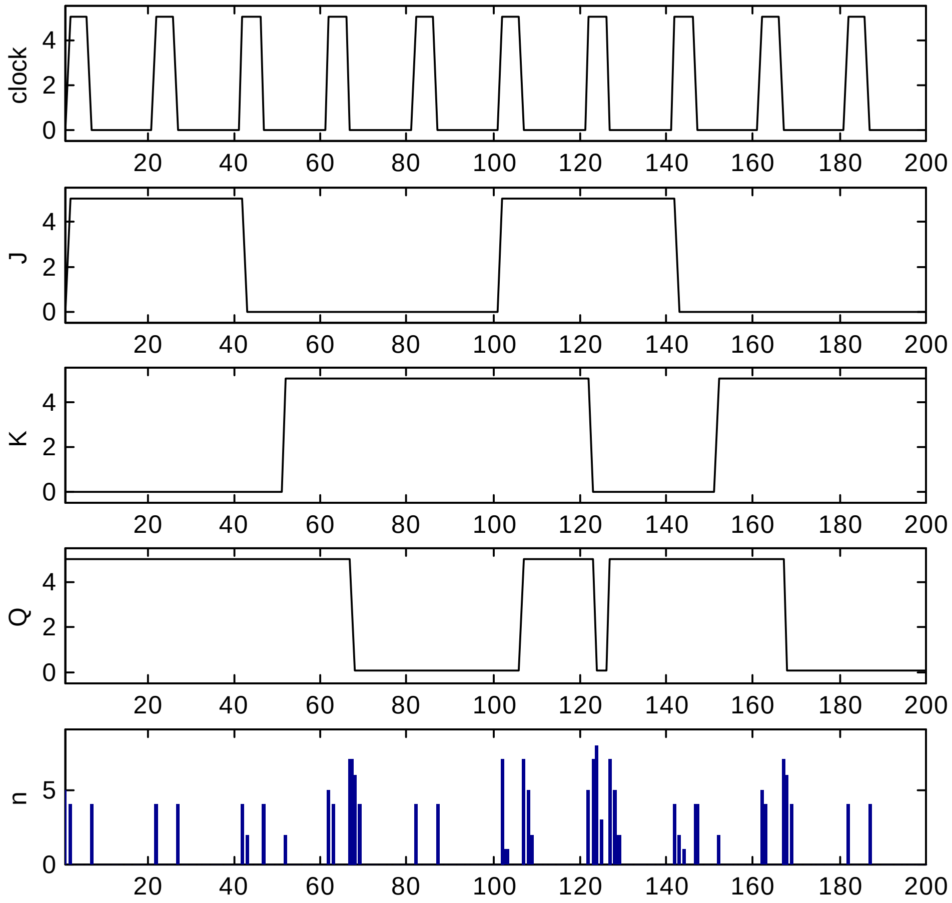


Fig. 8.6 The SAMOC timing simulation results of the master-slave *J-K* flip-flop.

8.1.4 4-bit Binary Ripple Counter

When analyzing circuits with larger topological depth, event driven simulation can avoid even larger percentage of non-significant block analysis. A simulation event caused by time varying signals may not always propagate to the end of the circuit. Especially in a circuit with larger topological depth, the simulation events are more likely blocked in the middle of the signal path and can not propagate through the circuit. Fig. 8.7 shows a 4-bit ripple counter constructed of four master-slave *J-K* flip-flops presented in section 8.1.3. The 4-bit counter contains 146 devices (2 voltage sources and 144 MOS transistors), 77 nodes and 32 blocks. The sole time varying source is the clock signal. The topological depth for SAMOC simulation of the counter is 12.

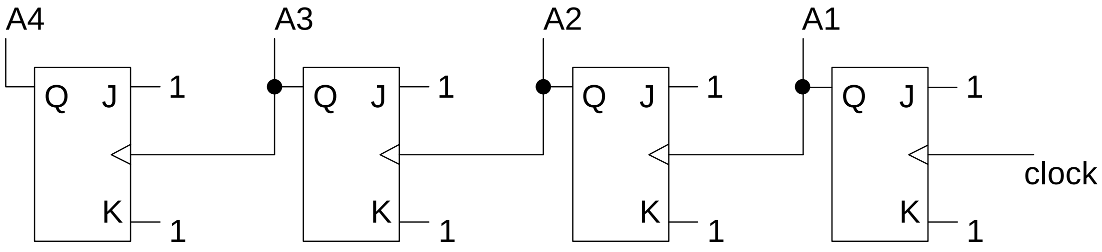


Fig. 8.7 4-bit binary ripple counter.

The SAMOC timing simulation results of the 4-bit binary ripple counter are illustrated in Fig. 8.9. There are 400 simulation instances in the presented results. The plots of signals (clock, A1, A2, A3 and A4) confirms the validation of the SAMOC simulation. The sixth plot in Fig. 8.9 is the number of triggered blocks as a function of the simulation instance. Observing the sixth plot, it is clear that during the timing simulation, most blocks remain static and the number of analyzed blocks increased when the output values of signals A1, A2, A3 or A4 are changed. Quantitatively, 961 blocks were triggered and analyzed during the 400 simulation instances, which means, on average, 2.4 of 32 blocks were analyzed at each simulation instance. The event-driven simulation mechanism saved about 92.49% of block analysis than without event-driven simulation approach.

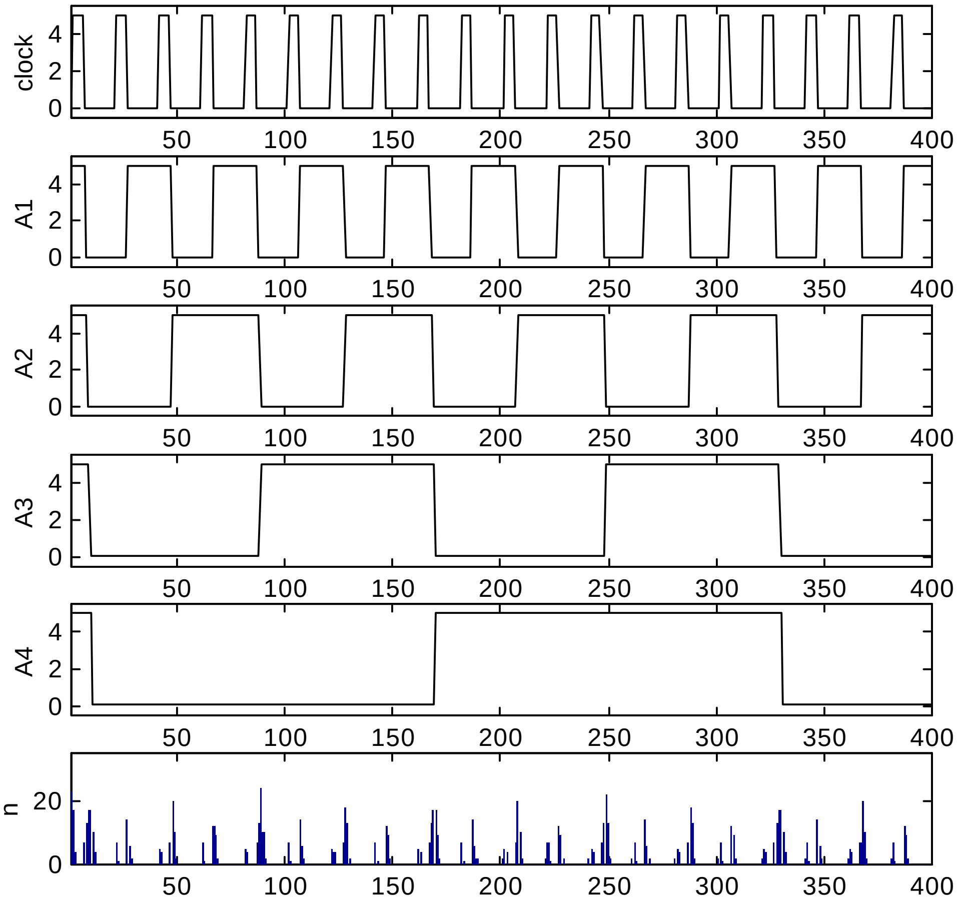


Fig. 8.9 The SAMOC timing simulation results of the 4-bit ripple counter.

8.2 Benchmark Circuit Simulation

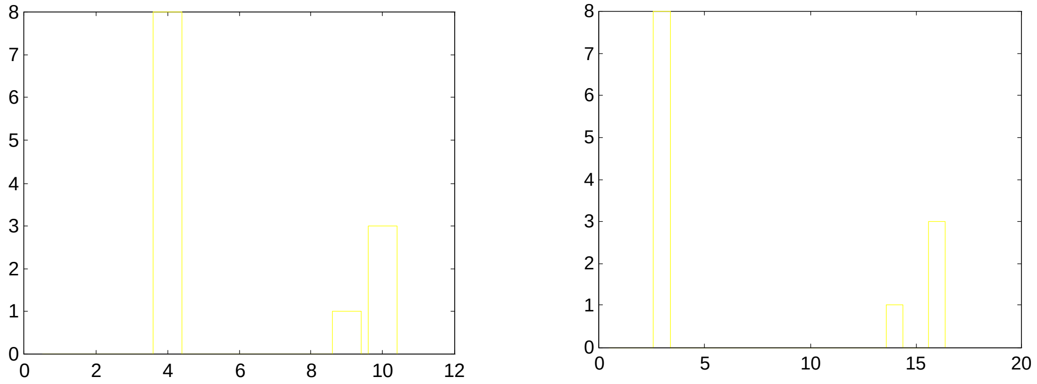
A number of benchmark circuits can be found in Internet[4]. Several types of benchmark circuits such as BJT (bipolar junction transistor) circuits and MOS circuits in SPICE netlist format can be downloaded from the web-site. Most of the MOS circuit

[4] The URL (universal resource location) of the benchmark circuits is http://www.cbl.ncsu.edu/CBL_Docs/csim90.html

netlists contain time domain analysis (transient analysis) commands, which can be performed by SAMOC. SAMOC timing analysis of three circuits downloaded from the web-site is presented in subsections 8.2.1, 8.2.2 and 8.2.3. SAMOC simulation was performed on IBM compatible personal computer running Windows NT with single Cyrix 6x86MX PR 200 processor and 64 MB SDRAM. The presented simulation results include partitioning information, required computing resources (CPU time and required memory) and a number of analyzed blocks at each simulation instance.

8.2.1 voter25

The circuit “voter25” contains 74 MOS transistors. SAMOC detects that “voter25” contains 82 elements and 52 nodes. That is, without circuit partitioning, 52 circuit equations are required to analyze the circuit. SAMOC partitions the circuit into 12 blocks and histograms of numbers of nodes and elements are shown in Fig. 8.10. There are 8 blocks containing 4 nodes 3 elements, 1 block containing 9 nodes and 14 elements and 3 blocks containing 10 nodes and 16 elements. The topological depth of the whole circuit is 2. SAMOC program and data structure of “voter25” occupy 3100k bytes of computer memory. Transient analysis specified in the circuit description is performed from $t = 0$ to 10^{-6} sec with time step 10^{-8} sec. The 100 simulation instances took 3 sec of CPU time to finish.



(a) (b)

Fig. 8.10 Partitioning results: histograms of numbers of nodes and elements.

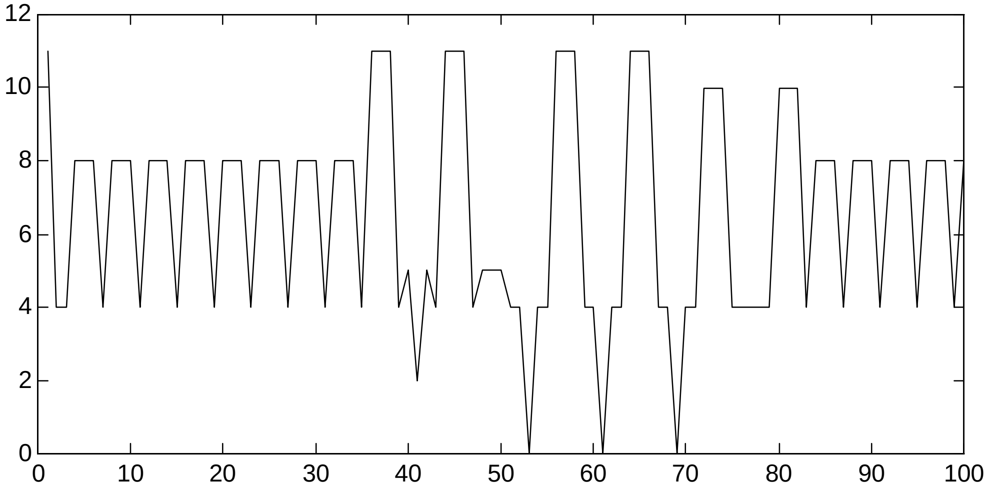
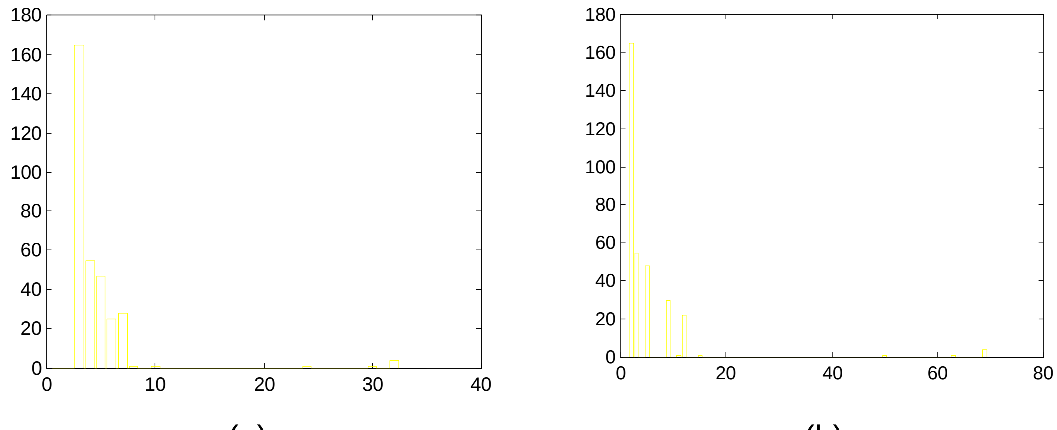


Fig. 8.11 Number of analyzed blocks as a function of simulation instance.

The number of analyzed blocks at each simulation instance is illustrated in Fig. 8.11. In the worst case, 11 of 12 blocks were analyzed at a simulation instance. At 3 simulation instances, because of the event-driven simulation mechanism, none of the blocks was required to be analyzed. During the 100 simulation instances, 666 blocks were analyzed; i.e., on average, 6.66 blocks were analyzed at each simulation instance.

8.2.2 sqrt

The circuit “sqrt” is a larger one which contains 1118 MOS transistors and 1022 capacitors. SAMOC detects that “sqrt” contains 2219 elements and 525 nodes. The partitioning routine in SAMOC extracted 328 blocks from “sqrt”. Histograms of numbers of nodes and elements are shown in Fig. 8.12. There are 3 main blocks and each one them contains 32 nodes 69 elements. The topological depth of whole circuit is 4. SAMOC program and data structure of “sqrt” occupy 4176k bytes of computer memory. Transient analysis specified in the circuit description begins from `t = 0` to $9x10^{-7}$ sec with time step 10^{-10} sec. These 9,000 simulation instances took 39 min and 6 sec of CPU time to finish. The number analyzed block at each simulation instance is illustrated in Fig. 8.13. The largest amount of analyzed blocks for a simulation instance is 322. The minimum number of analyzed blocks for a simulation instance is 168. Totally 1,538,959 blocks were analyzed during the 9,000 simulation instances. On average, 171 blocks were analyzed at each simulation instance.



(a) (b)

Fig. 8.12 Partitioning results: histograms of numbers of nodes and elements.

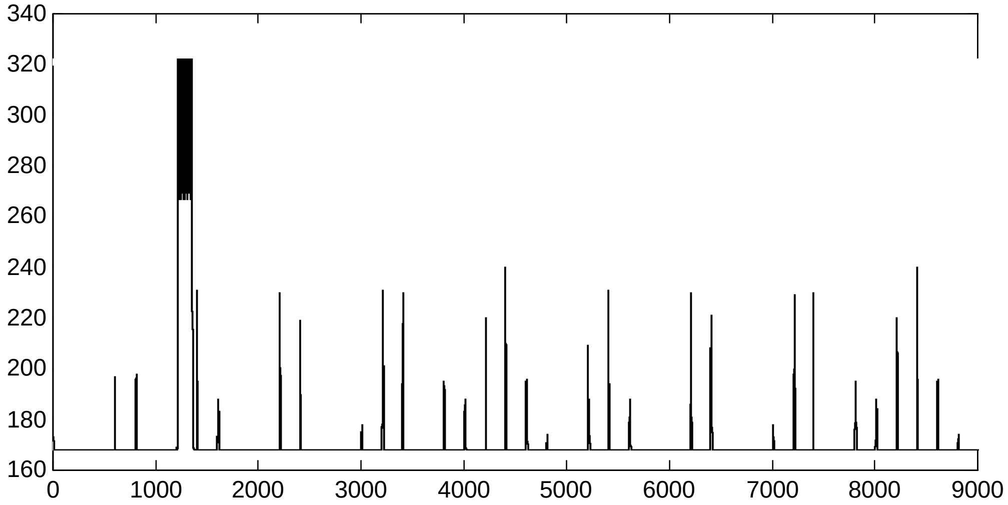


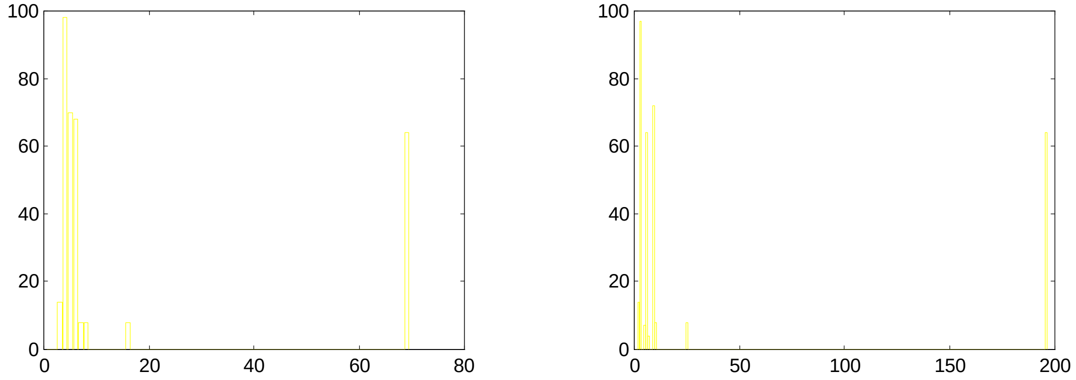
Fig. 8.13 Number of analyzed blocks as a function of simulation instance.

8.2.3 ram2k

“ram2k” contains 13880 MOS transistor and 156 capacitors. SAMOC partitions “ram2k” into 338 blocks with total 4873 nodal equations. The data structure created for all information of the “ram2k” circuit occupies 12,552k bytes of computer memory. The histograms of number of nodes and number of elements are shown in Fig. 8.4 (a) and (b). The histograms shows that the most of the devices are partitioned in 64 main blocks. Each block contains 70 nodes and 196 devices.

The topological depth of “ram2k” is 3 and the specified transient analysis is from 0 to `600ns` with `1ns` time step. SAMOC analyzes the 600 simulation instances. The total processing time in this machine is 49 min. The number of analyzed blocks is a function of the 600 simulation instances and is illustrated in Fig. 8.5. The minimum number of blocks which were analyzed in a simulation instance is 13 and the maximum number of blocks is

306. There are totally 20797 blocks analyzed during the 600 simulation instances. On average, 34.66 of 338 blocks were analyzed in each simulation instance. The circuit partitioning and event driven simulation mechanisms built in SAMOC saved 89.75% of block simulation.



(a) (b)

Fig. 8.14 Partitioning results: histograms of numbers of nodes and elements.

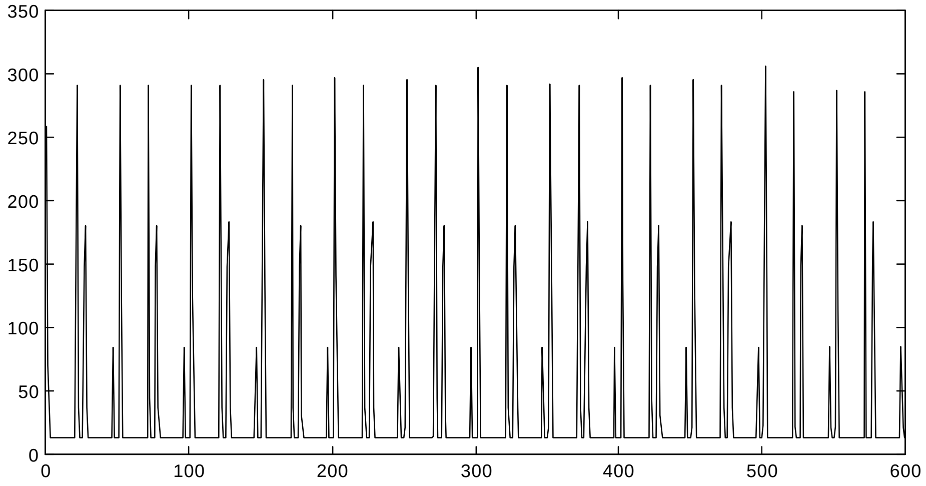


Fig. 8.15 Number of simulated blocks as a function of simulation instance.

8.3 Summary

Section 8.1 presents SAMOC simulation results of some common CMOS based logic circuits. Functional verification of SAMOC simulation such as circuit partitioning and event-driven simulation can be performed by analyzing the presented logic circuits. The programming and debugging stages of developing SAMOC are realized by analyzing the circuits presented in section 8.1 and some other simple gates or differential pair circuits.

The circuit simulation results presented in Section 8.2 are examples of how SAMOC handles large simulated circuits. The information about functions of the benchmark circuits are not provided by the web-site, where the benchmark circuits are downloaded from. The verification of the simulation results can not be performed, since HSPICE, SPICE2 and SPICE3 failed to analyze most of the benchmark circuits. However, the performed simulation is helpful for the future research of large MOS circuit analysis by presenting the circuit partitioning results and event-driven timing simulation information of the benchmark circuits.

SAMOC can not analyze all downloaded benchmark circuits. Even after circuit partitioning, some circuits may still contain blocks with more than 1,600 nodes and 2000 MOS transistors. The computer used for simulation needs more than 1 hour to solve a 1,600 by 1,600 matrix by using Gaussian elimination method. The computational complexity of the iterative Katzenelson algorithm is O(n), where n is the number of piecewise linear devices. Hence, a block containing 1,600 nodes and 2,000 MOS transistors will require solving a 1,600 by 1,600 matrix 2,000 times and the estimated time will be 2,000 hours for a block analyzed. In order to perform hundreds or thousands of simulation instances presented in Section 8.2, it will requires months or years to finish the simulation. It is definitely impractical and of no doubt to say: SAMOC failed the simulation. Although the simulation results can be obtained by using better and faster machine such as a super computer, depending on brutal force will make SAMOC no different from SPICE, and much worse than SPICE in precision. In addition, a huge block of such size is frequently activated, since it is the major portion of the simulated circuit. Event-driven simulation becomes nearly useless in this scenario. Therefore, from the simulation examples, the sizes of the simulated blocks and solving linear systems are the bottlenecks of the SAMOC simulation. Advanced circuit partitioning according to some bolder assumption such as adaptive partitioning from the transmission gates could be a solution.

Chapter 9

CONCLUSION

Contemporary semiconductor technologies and VLSI circuit design paradigms allow engineers to design very sophisticated systems and implement them in finger-nail-size silicon flakes. These sophisticated systems are capable of executing fast and complex arithmetic operations, performing signal processing, and extracting information from data bases. Most of the processed data and information are stored in a form of DC voltages, charges or currents.

DC solutions and other circuit responses can be evaluated theoretically by combining KCL, KVL and device models. This theoretical evaluation is very crucial in circuit design. Generally, the major tasks of the theoretical evaluation are formulation of the circuit equations and solution of defined equations to predict the simulation results.

Computer programs designed to automatically formulate and solve circuit equations are called circuit simulators. Circuit simulators read circuit descriptions, which contain device characteristics and circuit topology, and formulate the circuit equations by physical laws such as KCL, KVL and *conservation of charge.* After the circuit equations are prepared, simulators can automatically analyze circuits by the defined methods. The simulation can predict the circuit's behavior and gather statistics.

With the rapid growth of the complexity of VLSI circuit design, using general purpose circuit simulators such as SPICE, SABER and ASTAP for VLSI circuit simulation has become impractical. The required computer resource grows quadraticly with the number of nodes in the simulated circuits, and the consumed time for simulating a circuit is proportional to the cube of the number of nodes. In additions, calculating complex and precise device models is a considerable burden to computer systems. Hence, simulation of VLSI circuits has moved either to higher abstraction levels or device and waveform simplification.

The presented circuit simulation techniques and strategies in this research aimed at improving analog simulation performance in VLSI MOS circuit with ideal capacitors. The major application domain for the presented approach is large neural network simulation, in which computation is performed using voltage and charges, rather than voltages and currents. Such networks can be designed to dissipate very little energy as no DC current will be used. In addition the network should process the information locally and forward it to other computational centers. This will create a signal flow very much like in a digital circuit with very little or no feedback. Only some computational centers will be activated at a time which will naturally benefit from circuit partitioning and event-driven simulation.

9.1 Research Summary

The circuit simulator, SAMOC, developed in this research, is a software implementation of the proposed circuit simulation techniques and strategies. SAMOC employs piecewise constant approximation for time domain analysis of VLSI circuit by using comparatively less computer resources. In device modeling, SAMOC employs extremely simplified piecewise linear models for MOS transistors and diodes. In circuit formulation, SAMOC employs the modified nodal analysis (MNA) method, which is adequate to model the behavior of switches and controlled sources. The Katzenelson algorithm and the Gaussian elimination method are employed to solve the piecewise linear system.

There are two simulation engines built into SAMOC. One analyzes resistive networks and the other is for switched-capacitor (SC) networks. SAMOC analyzes a SC network by depending on the switching events, evaluating initial conditions and formulating algebraic charge equations. By using ideal switch model and ideal OPAMP model, SAMOC can provide very fast simulation results which are good for the functional verification for SC network designs. Simulation examples such as an integrator and a filter are presented. SAMOC also aids in designing and analyzing transferred power of SC based charge pump circuits.

In order to analyze VLSI circuits, SAMOC is equipped with circuit partitioning and event-driven simulation mechanisms. Circuit partitioning is made possible by defining shareable nodes, sharable devices, by separating controlled sources and by separating MOS transistors from gates to their drains and sources. The circuit blocks are extracted and the prior-posterior relationships between circuit blocks are also set. Circuit partitioning builds a block-signal diagram as a form of a directed graph for the simulated circuit. By analyzing each block individually, the employed Katzenelson algorithm converges easily because there are far less piecewise linear devices. In addition, determining solution of small linear systems costs less computational effort. Hence, analyzing a big circuit block by block costs less than analyzing the entire circuit.

Event driven simulation is realized by creating a simulation queue of the partitioned blocks. The order of block in the simulation queue is sorted by the topological depth of each block which assures that each block is analyzed before its posterior blocks and after its prior blocks. After a block is analyzed, it fires at all its posterior blocks. A procedure then checks all the fired blocks to determine if the fired blocks become activated and require to be analyzed. The firing and becoming active process is the "event" defined in SAMOC simulation. Note that: not all the fired blocks would become active, therefore, some insignificant block analyses can be avoided.

Timing simulation of SAMOC exploits the circuit partitioning and event driven simulation mechanisms. In timing simulation, the topological depths of blocks are redefined by assuming all signals transferred between blocks have the same delay time. Blocks without any prior block have the smallest topological depth '0' and highest simulation priority. Blocks of this kind will never be influenced during the timing simulation and only need to be analyzed once at the beginning of the timing simulation. The circuit blocks which contain or are directly influenced by time varying excitations (inputs) have second lowest topological depth ‘1’. The time step between two simulation instances is defined by SAMOC’s users. For every time step, the blocks with topological depth ‘1’ are analyzed first, then block with topological depth ‘2’, ‘3’ and so on until the process finishes analyzing ones with the largest topological depth.

From the simulation results, the circuit partitioning and event driven simulation do help SAMOC to do timing simulation of some circuits which failed simulators such as HSPICE, SPICE3 and SPICE2. The computational efficiency improvement induced by employing circuit partitioning and event driven simulation essentially depends on the topological depth of the simulated circuits. In the presented simulation example, the computational efficiency improvement can be up to 92%. That is, on average, only 8% of blocks are required to be analyzed during each simulation instance. More benchmark circuit simulation comparison is presented in Appendix II.

9.2 Software Implementation Summary

From the software implementation point of view, SAMOC is written in C++ programming language and follows the paradigm of object oriented design. Programs coded in C++ can be easily compiled to executable binary files. Some sophisticated C++ compilers can produce very efficient machine code, and can even optimize the code for a specified computer hardware configuration. Because of the standard format of C++, transplanting C++ programs across platforms (MS Windows to UNIX) and hardware (PC to RISC Workstation) costs little effort. The object oriented design is so far the best software technology available for developing, upgrading, and maintaining sophisticated programs.

The *class* hierarchy in C++ is ideal for representing devices with 2 to 4 terminals and with different characteristics. The *virtual function* techniques which accompany the *class* hierarchy in C++ make device processing such as partitioning circuits and creating matrix for MNA easy to implement, revise, reuse, extend and upgrade.

The *list of pointers* data structure built in Microsoft Foundation Class (MFC) is the principal data structure for accessing and managing the device, node and block data. MFC also provides embedded mechanisms to manage memory and prevent the memory leaks.

9.3 Future Works

Future development of SAMOC shows promise for further improvement in its performance and size of analyzed networks. Sections 9.3.1 and 9.3.2 list several selected improvements for future work.

9.3.1 Simulation Techniques Part

The most needed feature in SAMOC is delay estimation for intra- and inter-blocks. Delay estimation has been one of the most significant topics in research of circuit simulation techniques. Currently, the most ingenious method is the asymptotic waveform evaluation approach, AWE [55]. AWE begins with the frequency domain analysis and uses q^{th} order moment matching techniques to find out q most influencing time constants to synthesize the time domain waveform of a linear circuit.

Finding an efficient way to employ and modify AWE for piecewise linear models used in SAMOC can be a challenging research topic, although AWE may not be the only approach. For instance, a method based on Padé approximation [71] shows better approximation accuracy with similar computational effort. If the signal delay inside or between blocks can be evaluated, then the topological depth setting presented in Chapter 7 can be replaced by real delay time instead of currently used unit delay. Hence, SAMOC is capable of handling the signal racing problem. Furthermore, the block delay estimation can aid in developing adaptive simulation time step selection, which will improve both precision and efficiency of SAMOC simulation.

9.3.2 Software Technique Part

Netlist Error Detection

SAMOC is not equipped with a sophisticated error checking routine which is built into SPICE. On the other hand, the ideal switch model used by SAMOC would eventually cause existence of open circuits and floating nodes during circuit simulation. To overcome this problem a simple linear system solving algorithm that can discard the singular part of a linear system is built in SAMOC. SAMOC users will not be able to notice design error if the block MNA matrix is singular. For this reason, the connectivity error detection could be added to the existing programs.

The *AVL* Tree

The dynamic data structure which can provide fastest data search, access and sorting is the *AVL* tree (name after Adelson-Velskii and Landis) [65] [72]. The *AVL* tree is a sorted and perfectly balanced tree, which is much better for random access of huge amount of device and node information than linear list currently used by SAMOC. For a data structure with N elements, the average access effort for using linear list is $N/2$, while the *AVL* tree needs $\log_2(N)/2$. Although setting up an *AVL* tree costs a lot more than a linear list, and somehow adding an additional element to an *AVL* tree would cause a considerable rearrangement of the *AVL* tree, the *AVL* tree can significantly speed up the SAMOC simulation in random data access while the simulated circuit is huge and the number of simulation instances is large.

The *AVL* tree can be implemented in C++ by employing *template* techniques as in a similar way MFC implements linear list. *AVL* is not supported by the current version of MFC.

Multithread Programming

Multithread can be explained by multiprocessing of a single program. In multiprocessing environment, each process has its own set of instructions, data and system resources (e.g., memory or files). All threads of a multithread program can share the same set of data and system resource. A common example of a multithread program is a word processor, which allows users to edit and print a document during the same period, while the editing-thread and the print-thread belong to the same program (word processor) and share the same set of data (document). Computer system can slice and distribute CPU time to several threads. The block analysis of SAMOC can exploit the multithread programming mechanism that allows more than one block to be analyzed at the same time; that is, creating one thread for simulating each block. In a single CPU computer system, there may be no advantage of using multithread for SAMOC. But for a computer with multiprocessors and running the NT operating system, NT can automatically distribute the job of different threads to different CPUs. This will speed up the SAMOC simulation, especially in analyzing massively parallel interconnected circuits. MFC supports multithread programming and provides required function for multithread programming such as synchronizing data to make multithreading and event-driven simulation safe.

Hash

SAMOC uses device names and node names in the circuit description to identify devices and nodes in the simulated circuits. These names are strings of characters and string identification in SAMOC employs the member functions of the *class* `CString` of MFC. In some cases, users could named the devices by using meaningful words which could be very long and verbose. Furthermore, SAMOC generates devices and nodes in subcircuit by catenating device name in the subcircuit definition with the subcircuit identification in the circuit description. For simulating a circuit with heretical subcircuit definition, the generated names would be very long that contain 100 - 200 characters to identify devices and nodes which would seriously slow down the device and node identification. The identification of devices and nodes is frequently required in block extracting (circuit partitioning), MNA equation formulating and determining simulation events. If there is a hash table (look-up table) which simplify the device and node identification, then SAMOC simulation can be accelerated.

BIBLIOGRAPHY

[1] L. O. Chua, “Nonlinear circuits,” *IEEE Transactions on Circuits and Systems,* vol. CAS-31, no. 1, pp. 69-87 January 1984.

[2] L. O. Chua, and C-W Tseng, “A memristive circuit model for P-N junction diodes,” *Int. J. Circuit Theory Applications* , vol. 2, no. 4, pp. 367-289, December 1974.

[3] J. W. Gannett and L. O. Chua, “A nonlinear circuit model for IMPATT diodes,” *IEEE Transactions on Circuits and Systems,* vol. cas-25, no. 1, pp. 299-308 May 1978.

[4] L. O. Chua and Y. W. Sing, “A nonlinear lumped circuit model for Gunn diodes,” *Int. J. Circuit Theory Applications* , vol. 6, pp. 375-708, October 1978.

[5] L. O. Chua and Y. W. Sing, “Nonlinear lumped circuit model for SCR,” *IEE J. Electronic Circuit and Systems,* vol. 3, no. 1, pp. 5-14, January 1979.

[6] L. O. Chua and Y. W. Sing, “Nonlinear lumped circuit model for GaAs FET,” *IEEE Transactions Electron Devices,* vol. ED-30, pp. 825-833, July 1983.

[7] J. J. Ebers and J. L. Moll, “Large-Signal behavior of junction transistors” *Proceedings of the IRE 42,* pp. 1761-72, December 1954.

[8] L. W. Nagel, **SPICE2: A computer program to simulate semiconductor circuits**, Electron. Res. Lab., University of California, Berkeley, Rep., ERL-M520, May 1975.

[9] R. S. Muller and T. I. Kamins, **Device Electronics for Integrated Circuits,** Wiley, 1977.

[10] **The Design Center: Circuit Analysis Reference Manual,** MicroSim Corporation, 20 Fairbanks, Irvine, CA 92718.

[11] Y. Cheng, M. Chan, K. Hui, M.C. Jeng, Z.H. Liu, J.H. Huang, Kai Chen, P. K. Ko and C. Hu, **BSIM3v3 Manual (Final Version),** Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, December 1996.

[12] **HSPICE User’s Manual,** Meta-Software, 50 Curtner Ave. Suite 16, Cambell, CA. 95008.

[13] T. R. Bashkow, “The *A* matrix, new network description,” *IRE Transactions on Circuit Theory,* vol. CT-4, pp. 117-119, 1957.

[14] E. S. Kuh and R. A. Rohrer, “The state-variable approach to network analysis,” *Proceedings of IEEE,* vol. 53, no. 11, pp. 672-686, July 1965.

[15] F. H. Branin, Jr. “Computer methods of network analysis,” *Proceedings of IEEE,* vol. 55, no. 11, pp. 1787-1801, November 1967.

[16] F. H. Branin, Jr., “ The relation between Kron’s method and the classic method of network analysis,” in *IRE WESCON Conv. Rec.,* pt. 2, pp. 3-28, 1959.

[17] F. H. Branin, Jr., “DC analysis portion of PETAP-A program for analyzing transistor switching circuits,” IBM Development Lab., Poughkeepsie, NY, Tech Rep. 00.701, 1960.

[18] F. H. Branin, Jr., “DC and transient analysis of networks using a digital computer,” in *IRE Int. Conv. Rec.,* pt. 2, pp. 236-256, March 1962.

[19] N. G. Brooks and H. S. Long, “A program for computing the transient response of transistor switching circuits PETAP,” IBM Development Lab., Poughkeepsie, NY, Tech Rep. 00.700, 1959.

[20] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, “The sparse tableau approach to network analysis and design,” *IEEE Transactions on Circuit Theory,* vol. CT-18, pp. 101-113, January 1971.

[21] “ASTAP -- Advanced statistical analysis program,” IBM Program Product Document SH20-1118-0, IBM Data Processing Div., White Plains, NY, 1973.

[22] W. T. Weeks *et al.,* “Algorithms for ASTAP -- A network analysis program,” *IEEE Transactions on Circuit Theory,* vol. CT-20, pp. 628-634, November 1973.

[23] I. S. Duff, “A survey of sparse matrix research,” *Proceedings of IEEE,* pp. 500-535, April 1977.

[24] S. C. Eisenstat *et al.,* **Yale Sparse Matrix Package. Part 1 and 2,** Research Reports No. 112 and 114, Yale University, Dep. Computer Sciences, New Heaven CT.

[25] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman, “Algorithm and data structure for sparse sysmetric Gaussian elimination,” *SIAM Journal of Scientific and Statistical Computing,* pp. 225-237, June 1981.

[26] R. D. Berry, “An optimal ordering of electronic circuit equations for a sparse matrix solution,” *IEEE Transactions on Circuit Theory,* vol. CT-18, pp. 40-50, January 1971.

[27] W. F. Tinney and J. W. “Direct solution of sparse network equations by optimally ordered triangular factorization,” *Proceedings of IEEE,* vol. 55, no. 11, pp. 1801-1809, November 1967.

[28] O. Wing and J. Huang, “SCAP -- A sparse matrix analysis program,” *Proceedings of the ISCAS,* pp. 213-215, 1975.

[29] A. Geoge and J. Liu, **Computer Solution of Large Sparse Positive Definite**, Prentice Hall Inc., Eaglewood Cliffs, NJ, 1981.

[30] E. Rothburg and A. Gupta, “Efficient sparse matrix factorization on high performance workstation -- exploit the memory hierarchy,” *ACM Transactions on Mathematical Software,* vol. 17, no. 3, pp. 313-334, September 1991.

[31] C. W. Ho, A. E. Ruehli, P. A. Brennan, and D. A. Zein, “Interactive circuit analysis and design using APL,“ *Proceedings 1975 ISCAS,* pp. 216-219.

[32] C. W. Ho, A. E. Ruehli, and P. A, Brennan, “The modified nodal approach to network analysis,” *IEEE Transactions on Circuits and Systems,* vol. CAS-25, pp. 504-509, June 1975.

[33] L. W. Nagel and R. A. Roher, “Computer analysis of nonlinear circuits, excluding Radiation (CANCER),” *IEEE Journal of Solid State Circuits,* vol. SC-6, pp. 162-182, August 1971.

[34] T. E. Idleman, F. S. Jenkins, W. J. McCalla, and D. O. Pederson, “SLIC- A simulator for linear integrated circuits,” *IEEE Journal of Solid State Circuits,* vol. SC-6, pp. 188-204, August 1971.

[35] D. A. Zein, C. W. Ho, and A. J. Gruodis, “A new interactive circuit design program,” in *Proceedings IEEE Int. Symposium Circuit and System,* Huston, TX, pp. 913-917, 1980.

[36] L. W. Nagel and D. O. Pederson, ”Simulation program with integrated circuit emphasis,” in *Proceedings 16th Midwest Symposium Circuit Theory,* Waterloo, Canada, April 1973.

[37] D. O. Pederson, “A historical review of circuit simulation,” *IEEE Transactions on Circuits and Systems,* vol. CAS-31, no. 1, pp. 103-111 January 1984.

[38] F. H. Branin and H. H. Wang, “A fast reliable iteration method for dc analysis of nonlinear networks”, *Proceedings of IEEE*, vol. 55, no. 11, pp. 1819-1826, November 1967.

[39] J. Katzenelson, “An algorithm for solving nonlinear resistor networks,” *The Bell System Technical Journal*, vol. 44 pp. 1605-1620, October 1965.

[40] L. O. Chua and N. N. Wang, “A new approach to overcome the overflow problem in computer aided analysis of nonlinear resistive circuits,” *Int. J. Circuit Theory Applicat.*, vol. 3, pp. 261-284, 1975.

[41] P. Adorjan, “The $p = x + f(x)$ transformation: A new approach to the analysis of diode-transistor networks with exponential characteristics,” *Int. J. Circuit Theory Applicat.*, vol. 9, pp. 482-488, 1981.

[42] M. Tadeusiewicz, “DC analysis of circuits with idealized diodes considering reverse bias breakdown phenomenon,” *IEEE Transcactions on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 44, no. 4, pp. 312-326, April 1997.

[43] C. H. Roth, Jr., **Fundamentals of Logic Design,** 4th edition, West Publishing Company, 1992.

[44] S. W. Golpmb, R. E. Peile and R. A. Scholtz, **Basic Concepts in Information Theory and Coding, The Adventures of Secret Agent 0011,** Plenum Publish Corp., New York, 1994.

[45] C. Mead**, Analog VLSI and Neural Systems,** Addison Wesley 1989.

[46] X. Fang, **Small area, low power, mixed-mode circuits for hybrid neural network applications**, Ph.D. dissertation, Ohio University, November 1994.

[47] R. K. Brayton, F. G. Gustavson and G. D. Hachtel, “A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formula,” *Proceedings of the IEEE*, vol. 60, no 1, pp. 98-108, January, 1972.

[48] K. Singhal and J. Vlach, “Computation of time domain response by numerical inversion of the Laplace transform,” *Journal of the Franklin Institute*, vol. 299, no. 2, pp. 109-126, 1975.

[49] S. Topçu, O. Ocali, A. Atalar, and M. A. Tan ``A novel algorithm for DC analysis of piecewise-linear circuits: Popcorn," *IEEE Trans. on Circuits and Systems--Part I: Fundamental Theory and Applications*, vol. 41, pp. 553--556, August 1994.

[50] B. R. Chawla, H. K. Gummel, and P. Kozah, "MOTIS - an MOS timing simulator," *IEEE Transcactions on Circuits and Systems*, vol. CAS-22, pp. 901-909, December 1975.

[51] C. Visweswariah and R. A. Rohrer, "Piecewise approximation circuit simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-10(7), pp. 861-870, July 1991.

[52] K. A. Sakallah and S. W. Director, "SAMSON2: An event driven VLSI circuit simulator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4 (4), pp. 668-684, July 1985.

[53] E. Lelarasamee, A. E. Ruehli and A. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-1, no. 3, pp. 131-145, July 1982.

[54] J. K. White and A. Sangiovanni-Vincentelli, **Relaxation Techniques for the Simulation of VLSI Circuits,** Kluwer Academic Publisher, 1987.

[55] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design*, vol-9, no. 4, pp. 352-366, April 1990.

[56] R. Bryant, "A switch level model and simulator for MOS digital systems," *IEEE Transactions on Computers*, vol. C-33 pp. 160-177, February 1984.

[57] R. Kao and M. Horowitz, "Timing analysis for piecewise linear Rsim," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 13, no. 12, pp. 1498-1512, December 1994.

[58] J. Vlach, and K. Singhal, **Computer Methods for Circuit Analysis and Design,** Van Nostrand Reinhold, New York 1994.

[59] L. Pillage, R. A. Rohrer and C. Visweswariah, **Electronic Circuit and System Simulation Methods,** McGraw-Hill, New York 1994.

[60] F. J. Hill and G. R. Peterson, **Computer Aided Logical Design with Emphasis on VLSI, 4th edition,** John Wiley & Sons, Inc., New York 1993.

[61] B. Stroustrup, **The C++ Programming Language 2nd Edition,** Addison- Wesley Publishing Company, 1991.

[62] S. B. Lippman, **C++ Primer,** Addison-Wesley Publishing Company, 1991.

[63] J. K. Dickson, “On-chip high voltage generation in NMOS integrated circuits using an improved voltage multiplier technique,” *IEEE J. Solid-State Circuits*, vol., SC-11, pp. 374-378, June 1976.

[64] M. S. Makowski, "Realizability Conditions and Bounds on Synthesis of Switched -Capacitor DC-DC Voltage Multiplier Circuits”, *IEEE Trans. on Circuits and Systems -1: Fundamental Theory and Applications*, vol. 44 no. 8. , pp. 684-691 August 1997.

[65] E. Horowitz and S. Sahni, **Fundamentals of Computer Algorithms,** Computer Science Press, Potomac, Md. :, c1978**.**

[66] **Microsoft Visual C++: User’s Guide,** Microsoft Press, 1996.

[67] **Microsoft Visual C++: Microsoft Foundation Class Library Reference Part One,** Microsoft Press, 1996.

[68] **Microsoft Visual C++: Microsoft Foundation Class Library Reference Part Two,** Microsoft Press, 1996.

[69] **Microsoft Visual C++: Programming with MFC,** Microsoft Press, 1996.

[70] R. L. Burden and J. D. Faires, **Numerical Analysis, 5th Edition,** PWS-KENT Publishing Company, Boston, 1993.

[71] P. Feldman and R. W. Freund, “Efficient linear circuit analysis by Padé approximation the Lanches process,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 12, pp. 639-649, December 1995.

[72] M. A. Weiss, **Algorithms, Data Structures and Problem Solving with C++,** Addison-Wesley Publishing Company, INC, 1996.

[73] N. H. E. Weste and K. Eshraghian, **Principle of CMOS VLSI: A Systems Perspective, 2nd Ed.,** Addison Wesley, 1992.

[74] M. S. Ghausi, **Electronic Devices and Circuits, Discrete and Integrated,** Van Nostrand Reinhold, New York, 1985.

APPENDIX I

SUPPORTED DEVICES IN SAMOC

The device input format supported by SAMOC is very similar to the SPICE format. The supported devices of SAMOC, are categorized and listed at the Tables I.i and I.ii.

I.1 Device Input format

Voltage Sources

There are three types of voltage sources:

1. Independent voltage source :

```
V_name j j’ voltage_value
```

2. Voltage controlled voltage source

```
E_name k k’ j j’ voltage_gain
```

3. Current controlled voltage source

H_name k k’ j j’ transimpedance

For independent voltage sources, SAMOC supports time varying input method modeling which inside PWL (piecewise linear), PULSE (period pulse) and SIN (sinusoid). They have the same format as in the PSPICE input.

Current Source

There are three types of current sources:

1. Independent current source :

```
I_name j j’ current_value
```

2. Voltage controlled current source

```
G_name k k’ j j’ transconductance_value
```

3. Current controlled current source

```
F_name k k’ j j’ current_gain
```

Voltage Controlled Ideal Switches

```
S_name k k’ j j’
```

MOS Transistors

```
M_name d g s b model_name[5] [l=1u][6] [w=1u][7]
```

[5] currently the supported model is PWL and use pmos and nmos to distinguish the type.

Ideal Diode

`D_name j j’ turnedon_voltage`[volt] `breakdown_voltage` [volt]

Capacitor

`C_name j j’ capacitance`[pico (10^{-12}) Farad]

Resistor

`R_name j j’ resistance`[ohm]

I.2 Device Symbols and Stamps

The presented device stamps are used only when all terminals of a device are partitioned into the same block. The device stamp of a device whose terminals are in different blocks is presented in Chapter 6. The data presented in Tables I.i and I.ii are the same as which presented in [58].

[6] the default value is 1 μ m
[7] the default value is 1 μ m

Table I.i The 2-terminal circuit devices supported in SAMOC.

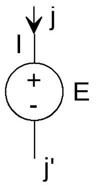
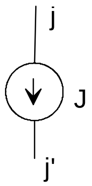
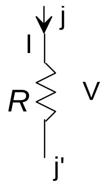
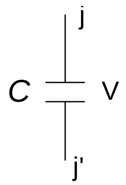
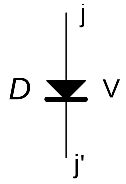
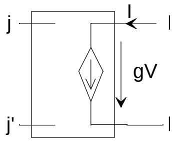
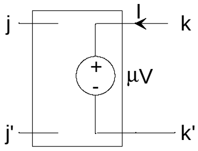
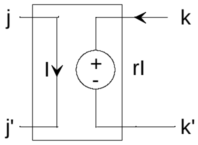
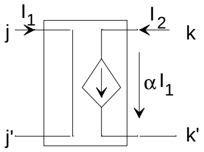
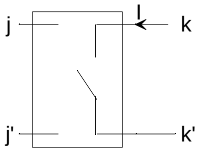
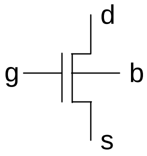
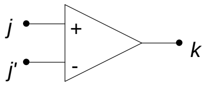
element	circuit symbol	matrix	equations	SAMOC class
voltage source (V, v)		$\begin{array}{cc} & \begin{array}{ccc} V_j & V_{j'} & I \end{array} \\ \begin{array}{c} j \\ j' \\ m+1 \end{array} & \begin{pmatrix} & & 1 \\ & & -1 \\ 1 & -1 & \end{pmatrix} \end{array}$ excitation vector $\begin{pmatrix} \\ \\ E \end{pmatrix}$	V_j - $V_{j'}$ = E I_j = I $I_{j'}$ = -I	C2VTerm
current source (I ,i)		$\begin{array}{c} j \\ j' \end{array} \begin{pmatrix} -J \\ J \end{pmatrix}$ excitaiton vector	I_j = J $I_{j'}$ = -J	C2Term
resistor (R, r)		$\begin{array}{cc} & \begin{array}{ccc} V_j & V_{j'} & I \end{array} \\ \begin{array}{c} j \\ j' \\ m+1 \end{array} & \begin{pmatrix} & & 1 \\ & & -1 \\ 1 & -1 & -R \end{pmatrix} \end{array}$	V_j - $V_{j'}$ - R I =0 I_j - $I_{j'}$ I	C2VTerm
capacitor (C, c)		$\begin{array}{cc} & \begin{array}{cc} V_j & V_{j'} \end{array} \\ \begin{array}{c} j \\ j' \end{array} & \begin{pmatrix} C & -C \\ -C & C \end{pmatrix} \end{array}$ excitation vector $\begin{pmatrix} Q_j \\ Q_{j'} \end{pmatrix}$	$C(V_j$ $-V_{j'})$ = Q_j $C(V_{j'}$ $-V_j)$ = $Q_{j'}$	C2Term
ideal diode (D, d)		$\begin{array}{cc} & \begin{array}{ccc} V_j & V_{j'} & I_D \end{array} \\ \begin{array}{c} j \\ j' \\ m+1 \end{array} & \begin{pmatrix} & & 1 \\ & & -1 \\ F & -F & F-1 \end{pmatrix} \end{array}$	$V_j < V_{j'}$ F=0 $V_j > V_{j'}$ F=1 PWL approach	CDiode

Table I.ii The multi-terminal circuit devices supported in SAMOC.

element	circuit symbol	matrix	equations	SAMOC class
voltage controlled current source (G, g)		$$\begin{array}{c\|cc\|} & V_j & V_{j'} \\ k & g & -g \\ k' & -g & g \end{array}$$	$I_j = 0$ $I_{j'} = 0$ $I_k = g\ (V_j - V_{j'})$ $I_{k'} = -\ g\ (V_j\ V_{j'})$	C4Term
voltage controlled voltage source (E, e)		$$\begin{array}{c\|ccccc\|} & V_j & V_{j'} & V_k & V_{k'} & I \\ j & & & & & \\ j' & & & & & \\ k & & & & & 1 \\ k' & & & & & -1 \\ m+1 & -m & m & 1 & -1 & \end{array}$$	$-\mu\ V_j + \mu\ V_{j'} +$ $V_k - V_{k'} = 0$ $I_k = I$ $I_{k'} = -I$	C4Term
current controlled voltage source (H, h)		$$\begin{array}{c\|cccccc\|} & V_j & V_{j'} & V_k & V_{k'} & I_1 & I_2 \\ j & & & & & 1 & \\ j' & & & & & -1 & \\ k & & & & & & 1 \\ k' & & & & & & -1 \\ m+1 & 1 & -1 & & & & \\ m+2 & & & 1 & -1 & r & \end{array}$$	$V_j - V_{j'} = 0$ $V_k - V_{k'} - r\ I1 =$ 0 $I_j = -\ I_{j'} = I1$ $I_k = -I_{k'} = I2$	C4HTerm
current controlled current source (F, f)		$$\begin{array}{c\|ccccc\|} & V_j & V_{j'} & V_k & V_{k'} & I \\ j & & & & & 1 \\ j' & & & & & -1 \\ k & & & & & a \\ k' & & & & & -a \\ m+1 & 1 & -1 & & & \end{array}$$	$V_j - V_{j'} = 0$ $I_j = -I_{j'} = I$ $I_k = -I_{k'} = \alpha\ I$	C4Term

voltage controlled ideal switch (S, s)		$$\begin{array}{c\|ccc\|} & V_k & V_{k'} & I_s \\ k & & & 1 \\ k' & & & -1 \\ m+1 & F & -F & F-1 \end{array}$$	open circuit
			F=0
			short circuit
			F=1
				C4Term

MOS transistor (M, m)		discussed in Chapter 2		CMos
Ideal OPAMP (O, o)		$$\begin{array}{c\|cccc\|} & j & j' & k & m' \\ j & & & & \\ j' & & & & \\ k & & & & 1 \\ m' & 1 & -1 & & \end{array}$$	$V_j = V_{j'}$	COpamp

APPENDIX II

BENCHMARK CIRCUIT SIMULATION AND COMPARISON

The presented benchmark circuit simulation was perform on an IBM PC compatible computer with a Cyrix® 6x86MX PR200 CPU running at 166 MHz (66 x 2.5), and a Number Nine® Imagine 128-II PCI graphic card with 4 MB of video RAM. The computer was equipped with 512 MB of EDO (extended data out) DRAM. Two operating systems, MS® Windows NT 4.0 with service pack 4 and Redhat® Linux 5.2 with kernel 2.0.34, were installed in the used computer. The SPICE simulation was performed in Linux by using Berkeley SPICE3f5 which can be found in the Internet[8]. SAMOC simulation of the same set of benchmark circuits was performed in the same

[8] One place to download SPICE3f5 from is http://uiarchive.cso.uiuc.edu

computer with the same configuration as for the SPICE simulation while the operating system was switched from Linux to MS Windows NT.

Table II.1 shows the SPICE simulation results. Each simulation is time domain transient analysis. SPICE3f5 did not finished the simulation of benchmark circuits "sqrt" and "ram2k". The simulation processes ended because of not enough system memory.

Table II.i SPICE simulation results of benchmark circuits.

Circuit	MOSFETs	Capacitors	Resistors	Time span (sec)	Time step (sec)	simulation instances	CPU time (sec)
ab_integ	31	22	3	4.0E-01	1.0E-04	4000	1.38
ab_opamp	31	24	4	2.0E-04	1.0E-07	2000	1.69
cram	60	42	0	8.0E-07	1.0E-09	800	15.36
mux8	64	29	0	3.0E-05	1.0E-07	300	226.19
toronto	58	33	0	2.0E-06	1.0E-08	200	4.72
counter	220	0	0	1.0E-07	5.0E-10	200	28.43
b330	330	0	0	1.2E-06	6.0E-10	2000	failed[9]
voter25	74	0	0	1.0E-06	1.0E-08	100	4580[10]
sqrt	1118	1022	0	9.0E-07	1.0E-09	900	25314.3[11]
ram2k	13880	156	0	6.0E-07	1.0E-09	600	23269.1[12]

SAMOC simulation of the same set of benchmark circuits is presented in Table II.2. Table II.2 also shows the partitioning information and topological depth. The required CPU time information is not the as same as which in Chapter 8. It is because the change of computer hardware configuration and SAMOC version. System memory was

[9] time step too small fail.

[10] finished but required 350 MB of system memory.

[11] out of memory fail.

[12] out of memory fail.

switched from 64MB SDRAM in Chapter 8 to 512MB EDO RAM. SAMOC version used for benchmark analysis was switched from debug mode in Chapter 8 to released mode.

Table II.ii SAMOC simulation results of benchmark circuits[13].

Circuit	elements	nodes	blocks	TP[14]	CPU time (sec)
ab_integ	62	36	10	5	3
ab_opamp	62	36	10	5	10
cram	114	45	18	1	9
mux8	105	43	15	6	73
toronto	102	37	36	1	5
counter	223	97	30	1	6
b330	363	179	90	1	96
voter25	82	52	12	2	1
sqrt	2,219	525	328	4	1,116
ram2k	14,095	4,873	338	3	1,869

[13] The simulation results shown in this table are different from the results in Chapter 8. The difference is caused by different configuration of compiler option and computer hardware.

[14] TP = topological depth

Jan, Ying-Wei, Ph.D. March, 1999
Electrical Engineering

<u>A Switched-Capacitor Analysis Metal-Oxide-Silicon Circuit Simulator</u> (209 pp.)

Director of Dissertation: Janusz A. Starzyk, Ph.D.

This research presents such new circuit simulation strategies and techniques as piecewise device modeling, circuit partitioning and event-driven analysis for fast simulation of analog MOS-based VLSI circuits. An automatic circuit formulation algorithm: based on the modified nodal analysis, and numerical analysis techniques such as Katzenelson algorithm and Gaussian elimination were combined with the proposed methods to form a complete solution for fast analog VLSI time domain simulation.

In order to verify and evaluate the proposed circuit simulation techniques and its partition strategies, a new circuit simulator, SAMOC, was built in the form of a digital computer program. This work presents details of building this circuit simulator with its data structure for circuit representation, and circuit block analysis scheduling for event-driven simulation. Both are specially designed for analyzing large scale MOS circuits on resource limited computer systems.

Simulations of several well-known small analog and digital circuits were presented for functional verification of SAMOC. A set of benchmark circuits which contain large amount of MOS transistors were used to evaluate the simulation efficiency improvement of the proposed methods comparing with standard industrial SPICE simulation. The simulation results indicate that SAMOC can exploit the latency of circuits, speed up the circuit simulation, and analyze such large circuits, which SPICE3f5 failed to do.

Approved: ____