VERILOG-A MODELING OF TRANSISTOR

Zhili Yang

Bachelor of Engineering
Electronic Engineering Major

Department of Electronic Engineering
Macquarie University

April 4, 2016

Supervisor: Dr Oya Sevimli and Prof. Tony Parker

ACKNOWLEDGMENTS

I would like to gratefully and sincerely thank Associate Professor Tony Parker and my department supervisor Oya Sevimli for their guidance, patience, understanding and most importantly, their friendship during my thesis at Macquarie University.

I would like to thank my engineering friends for their help when I needed it the most. Finally, I would like to extend my gratitude to the Department of Engineering for offering me the opportunity to undertake this thesis.

STATEMENT OF CANDIDATE

I, (Zhili Yang), declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: Zhili Yang

Student's Signature: Zhili Yang

Date: 4 April 2016

ABSTRACT

Industry standard transistor models used for computer aided circuit design are not always accurately predicting transistor performance under all circuit conditions, include RF and analog circuit behavior. Transistor model are expected not only circuit performance but also reliability and accuracy. Improved accuracy may need concern access to the internal currents and voltages of the transistor or add extra internal features such as low-frequency noise sources. Circuits that have very stringent accuracy requirements require modeling techniques with higher accuracy. In contrast, Verilog-A enables full control of transistor model and can be integrated with the circuit design software. This document report presents how effective approach to investigate and implement our transistor model in two different environments-AWR and Verilog-A. This is useful for circuit-level design purposes to allow the designer manipulates transistor models have direct access to model equation sets and program structure. A comparison of results between simulation and measurements to transistor model has been illustrated in details.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Circuit simulation allows circuit designers to successfully optimize and validate their design. To simulate a circuit, circuit simulators convert the schematic into a system of linear and/or nonlinear differential equations. In order to achieve a schematic to become a system of equation, each component in the circuit must be represented by a smaller system of linear and/or nonlinear differential equations. The smaller system for a component or block is named as the compact model for component [7].

The availability of accurate, efficient compact models is important to the successful utilization of any circuit simulation software. The need for rapid development and distribution of advanced transistor device models becomes more acute than ever.

Traditionally, Transistor device models that are built-in circuit simulators have been developed using general purpose programming languages, such as C, C++, or Fortran. Therefore, they are targeted specifically to the interface and internal data structures of their host simulator. However, this kind of interfaces have typically been not-portable, not standard and inefficient, which result a time consuming and error-prone endeavor in new model creation.

The rapidly increasing availability and utilization of analog HDLs such as Verilog-A provides the promise of a valid outcome to transistor model development and deployment problem. Over the past several years, Verilog-A has become a leading candidate for compact model development. Recently, the compiled solution of Verilog-A become available, which improve the performance of simulation.

1.1 Motivation

The compact transistor model is at core of circuit-level design of analog and radio frequency integrated circuits(RFICs) enabling the designer to efficiently achieve design goals. They are used for almost all modern electronic design work. Complete and accurate modeling is based on correct design and analysis of electronic internal circuits especially the analogs ones, which allows design to work earlier. Modern circuit are normally very com-

plex. The performance of such circuit is difficult to predict without accurate transistor models.

The motivation behind the works presented in this thesis is importance of compact transistor modeling in Verilog-A.

Industry standard transistor models used for computer aided circuit design. As system become more complex and as devices become more sophisticated, smaller compact models are necessary for describe the behavior of systems and circuits. Therefore, industry standard transistor models are not always suitable for all situations. Improved accuracy may need access to the internal currents and voltages of the transistor or may be required to add extra internal features such as low- frequency noise sources. Verilog-A language is used for integrated with the circuit design software and allows transistor model designer to focus on their area of expertise, which enable full control of the transistor model. The aim of this thesis project is to investigate and implement Verilog-A models for our transistors and compare with measurements.

1.1.1 Challenges

Challenges to designer:

- The compact model developers must be familiar with the Verilog-A computer program.
- Required to familiar with how NI AWRDE operate with visual studio
- The developers require have good understanding about the characteristic of transistor models.
- PDK development traning the way it present, which is not clear.
- Required good undestanding in C, C++ code.

1.2 Thesis Overview

A brief description of the entire project will be summarized here in this section. TThe project plan and deliverables can be found in Appendix B.3 of this document. This thesis project is supervised by Dr Oya Sevimli, senior lecturer in the Department of Engineering, Macquarie University and co-supervised by Prof. Tony Parker. Regular weekly meetings with the academic supervisor are required which ensure the project is in the right track with the set goals for every weeks. Consultation Meetings Attendance Form can be found in Appendix B.2 of this thesis report.

There are 6 chapters in this thesis report. Chapter 1 gives introduce the whole thesis, which includes the main purpose of this thesis project. Chapter 2 gives specific background

information from the literature and discuss the related transistor models. Topics discussed include The physical HBT, FBH HBT model, VBIC model, Noise sources and Low-Frequency Noise Characteristics. It presents the characteristic of transistor and how noise sources effect the behaviour of transistor.

Chapter 3 discuss how implement model wizard in AWR and using Microsoft visual studio to create model. It also discuss what issues we encounter when implement the software and how we solve it. Furthermore, the simulation results will be used to verify our transistor model are accuracy or not.

Chapter 4 will include the procedure of how we create model in NI AWRDE PDK by using NI AWRDE and visual studio

Chapter 5 will include summary about the thesis and demonstrate the contribution. It include a linear resistor in Verilog-A and a nonlinear diode in Verilog-A. Compare the simulation of our model to theoretical solutions. To investiage error of models and how to figure out. Following by a discussion about our project to the future work.

Chapter 6 discusses the possible future work base on the current model in Verilog-A and look at its benefits.

Project Objectives

- Revision of transistor models.
- Familiarize with simulation software such as AWR, visual studio and Verilog-A.
- Verification of transistor models through AWR nonlinear simulation.
- To investigate how noise sources effect the characteristic of transistor model.

1.3 Project Baseline Review

The project was planned to be executed from 1st of March through 24th of June, 2016. Baseline plan was generated with the intention of using only the weekdays, including mid-semester break, for project activities.

1.3.1 Time Budget Review

Table 1.1 summarizes the time budget comparison, according to which 42 days of work has been completed within 35 days. The rest days shall be used for composing the project report, therefore no amendments to the time schedule is required.

Although the allotted time for each activity was followed closely, the order of accomplishment varied. Overall, the progress of the report is followed with the project schedule done at start of the semester. Full detail of actual and planned order of accomplishment and duration of each project activity is depicted in Table 1.3.

Estimated Work	80 days
Realized Work	42 days
Percentage Completion	50%

Table 1.1: Time Budget Summary

1.3.2 Financial Budget Review

The project was initially allocated $300. The project didnt require to buy anything except it required AWR and Visual studio software which I have downloaded from DreamSpark and installed in my laptop. Therefore, there is no further purchasing is required, thus the budget surplus may not need.

Chapter 2

Literature Review and Theory

2.1 FBH HBT Model

2.1.1 Introduction

Device models are usually used in circuit simulation for the design of integrated circuits. The Heterojunction Bipolar Transistor(HBT) is based either on the GaAs/AIGaAs or GaInP/GaAs compound semiconductor material. In recent years, heterobipolar transistors can be achieved by using commercial GaAs MMIC technology. Due to their advantages able to operate at high power densities they are the devices suit for power amplifiers, such as in mobile phones. Although technology is mature and industry already has a large amount of HBT-based MMICs, the model development in circuit design lacks behind. The designer has a batch build-in models for GaAs-based FETs in standard circuit simulators, however models for GaAs-based HBTs are scarce. The choice to use neither the simple SPICE-type Gummel-Poon model nor full design with a linear S-parameter based model. None of them are satisfy the requirement.

It leads to deal with extension of the GP model that are required for the description of HBTs, and finally tend to the development of FBH model. The state-of-the-art GaAs-based HBTs have characteristics as below:

- The semi-insulating substrate to avoid parasitic substrate effects that have to be considered in silicon.
- In a sophisticated technology, surface or interface involved problems such as parasitic currents are inappreciable, and even thermal runaway can be restrained by using base feedback or proper emitter or by using thermal shunt technology.

Two effects are necessary to be considered in simulate HBTs. The first one is self-heating, the second one is the current dependence of the transit frequency, effected by high current injection into the collector.

2.1.2 Overview over HBT Model Features

The FBH model consider:

- Differentiation of intrinsic and extrinsic base-collector diode
- Non-ideal base currents (Base-Collector and Base-Emitter)
- Thermal interaction and self-heating (by a thermal port)
- Collector transit time and current-dependence of base-collector capacitance
- Charge differentiation
- Break-down involved with base-emitter and base-collector
- Strengthened noise model
- Scaling with transistor size
- Unclear analytic parameter extraction from measurements

2.1.3 Model topology

Since substrate effects are inappreciable, it leaves a very simple equivalent-circuit topology as shown Figure 2.1. Because of the structure of the HBTs, the total base-collector junction separates into an active part below the emitter, and a parasitic part below the base contacts. Due to current flows nearly vertically from emitter to sub-collector, the parasitic section has an extra PN diode which is reverse biased in usual operation. The section under the emitter is named as active HBT in order to differentiate it from the parasitic BC-diode which conveys no current. But, both of them are portion of the intrinsic HBT, because the parameters are bias dependent. Only the passive embedding network is named as extrinsic HBT.

2.1.4 large-signal Equivalent Circuit

The extrinsic section of the large-signal equivalent circuit is shown in Figure 2.2. It shows the passive feeding structures and contact resistances. Assume all parameters are independent of bias and therefore are same to the small-signal parameters.

The intrinsic large-signal equivalent circuit is shown in Figure 2.3. The extra electrical section, it contains a thermal sub circuit to determine the self-heating of device because of dissipation of power. The current source uses to amplify the base currents in forward and backward operation. Only the current flow through the resistances is amplified, as determined in the figure. The current amplification factors are called BF.
The diode symbols depict bias dependent resistances with diode characteristics. The forward operation case, the collector current is structured by the current source. The

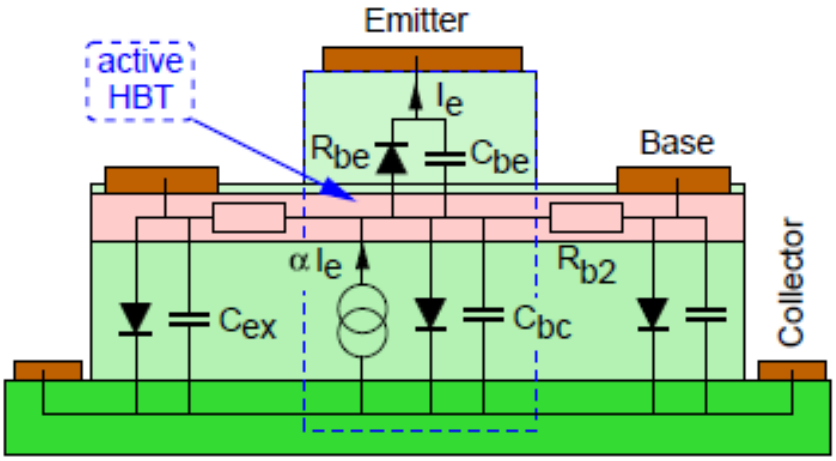


Figure 2.1: Schematic cross section of a single finger HBT [8]

current I_{cf}/BF depicts the ideal base-emitter current which is amplified. The currents I'_b,I''_b, and I'_{bc} depicts the nonideal (parasitic) base currents. The circuit contains bias independent resistors Rbxx, Rbbxx, and Rcxx to generate the possible saturation of these current elements. The base-collector current is divided into two components because of the mesa structure of the HBT: The active HBT include the current I'_{cr} and the parasitic current I''_{cr}. I'_{cr} is effected by the voltage across the parasitic section of diode. Both diodes are split by the resistance of the intrinsic base layer Rb2 that is assumed as a constant value.

Two current sources $I_{av,c}$ and $I_{av,e}$ are used to break down base-collector and base-emitter.

Two charges are used for both junctions, a depletion (Q_{xj}) capacitance and a diffusion (Q_{xD}).

The total dissipated power P_{diss} is supplied to a thermal subcircuit.

The voltage $\triangle T_j$ is equal to the temperature rise at the emitter junction because of self-heating of the device.

2.1.5 Thermal Effect

Because of the large thermal resistivity of GaAs, thermal effect containing self-heating are very important in GaAs-based HBTs and must be taken into account when determine the large signal performance of the device. The equivalent circuit denoted in Figure 2.3, the temperature increase at the emitter junction can be figure out. The emitter junction temperature replace the device temperature is used since the temperature distribution is no uniform through the device and the most meaningful effect in temperature rise on

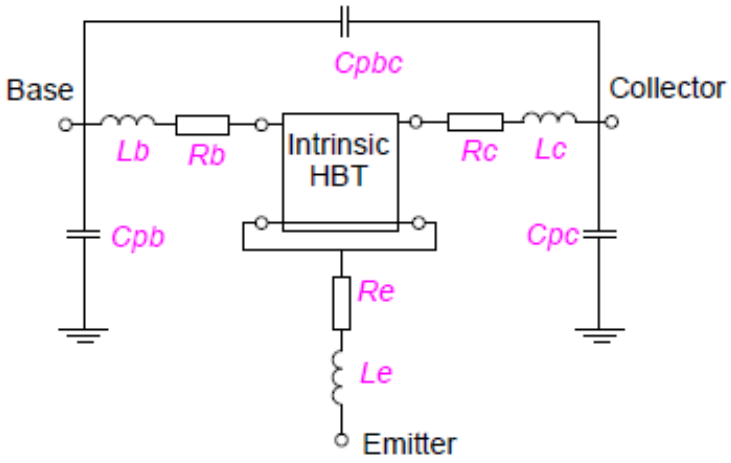


Figure 2.2: Extrinsic section of equivalent circuit. The model parameters are denoted in oblique magenta letters []

device performance happen at the emitter junction. The R_{th} in the circuit stands for thermal resistance corresponding to the junction temperature increase. Normally, R_{th} is a constant value in order to simplify the circuit. (A New Large Signal HBT Model)

2.1.6 Importance of HBT modeling

Heterojunction Bipolar Transistor(HBTs) offer substantially great power gain cut-off frequencies. In GaAs/AlGaAs or GaAs/GaInP devices, the wide bandgap emitter allows large base doping without significant charge enter into emitter, giving low base resistance even the device with small base widths.

Physically significant HBT models provide the ability no only to predict and to design a practical circuit with good accuracy, but also to estimate the device parameters how they affecting the circuit performance. This could achieve the optimization of device design with respect to a particular application and the evaluation of the manufacturing process [9].

2.1.7 Challenge of achieve HBT modeling

The problem of how to construct a precise model for GaAs/AlGaAs or GaAs/GaInP HBT can becomes a prominent issue.

An accurate device model that allows the reliable for a wide range of operating biases and signal frequencies is need for the design of high performance microwave circuits. Recently, existing bipolar transistor models used in most commercial circuit simulators, which are based on the Gummel-Poon model. It does not consider several effects which are important for predict the performance of HBT. Especially, the self-heating effect has been

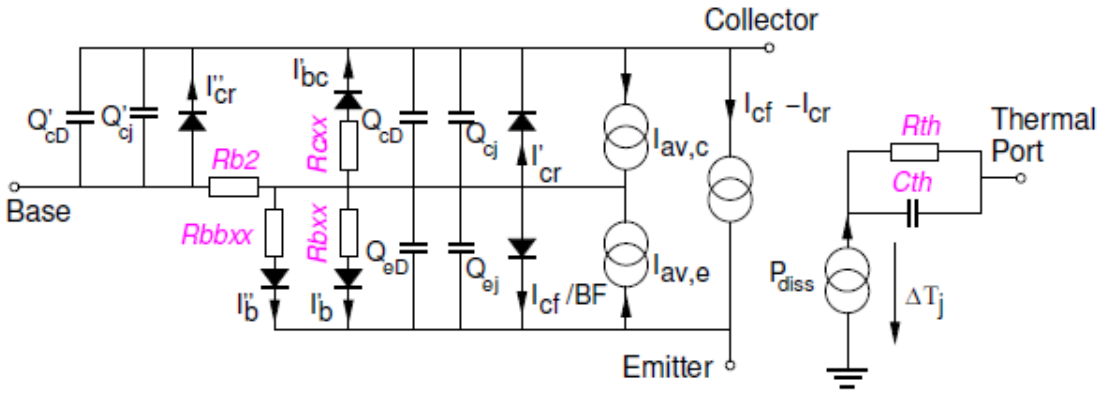


Figure 2.3: Intrinsic section of large-signal equivalent circuit. The model parameters are denoted in oblique magenta letters []

recognized as critical factor in determine the operation of HBT at high power dissipations, because of the low thermal conductivity of GaAs. To design circuits in which thermal effects are critical, dependence of junction depletion capacitance and transit time on collector current, and variation of effective electron barrier in an emitter heterojunction [12]. The thermal and electrical behavior of the circuit must be modeled and solved consistently, which can be complicated. No only this, but also a circuit model for GaAs/AlGaAs HBT is required in nearly all software and simulators to effectively regenerate the physical properties of the device in any new design. Modeling of GaAs/AlGaAs HBT can be very tedious task due to its complexity of device geometry, long process of linear and non-linear model parameter extraction and non-linear properties of the device.

2.2 VBIC Model

2.2.1 Compare Gummel-Poon model to VBIC model

Since the early 1970s, the SPICE Gummel-Poon model has become the workhorse for silicon bipolar junction transistor modeling. The construction of the Gummel-Poon(GP) model has consideration in the pivotal physical parameters and mechanisms that allow to control BJT behavior in intuitive and consistent manner. However, the approximation that underlie the Gummel-Poon model ignores the effects, which are critical for sufficiently accurate modeling of BJTs. [4] Such as:

- Parasitic transistor, self-heating, avalanche multiplication is not modeled.
- Collector resistance modulation is not take into account
- Simplified Early effect model is not accurate

The VBIC model for bipolar transistors is an extension of the Gummel-Poon model for Si bipolar transistors. The VBIC models has been improved base on the Gummel-Pool model in the follow aspects:

- Improved temperature
- Improved Early effect on transit time
- Parasitic substrate
- Parasitic transistors
- Parasitic fixed (oxide) capacitance
- Quasi-saturation modeling
- Avalanche multiplication
- Decoupled base current from collect current
- Electro thermal (self-heating effect) modeling [10]

2.2.2 Review of VBIC model

Before talk about the VBIC model, we give a brief review about the SGP model. It is a three-terminal model such as emitter, base, and collector terminals and contains three current sources I_{CC},I_{bc} and I_{be}, two capacitances related with the charges Q_{bc} and Q_{be} stored between the base and collector terminals and between the base and emitter terminals, respectively, and four series resistances.

Basic on Figure 2.4 presents the equivalent circuit if the new VBIC model. Different with the conventional SGP model, which is three terminals, the VBIC has four terminal model including the base, emitter, collector, and substrate expressed by the letters b, e, c and s, respectively, and the current entering into these terminals are I_b,I_e,I_c andI_s. The other node of VBIC model are extrinsic base bx ,intrinsic base bi, parasitic base bp, intrinsic emitter ei, intrinsic collector ci, and extrinsic collector cx [2].

2.2.3 VBIC model more suitable than Gummel-Poon model for HBT modeling

The VBIC bipolar junction transistor model is a candidate for GaAs HBT modeling for a couple of reasons:

- Self-heating and excess-phase effects is included in VBIC model
- It offers a base current model that is a more accurate approximation to the actual device than the one which is used in the conventional Gummel-Poon(G-P) model

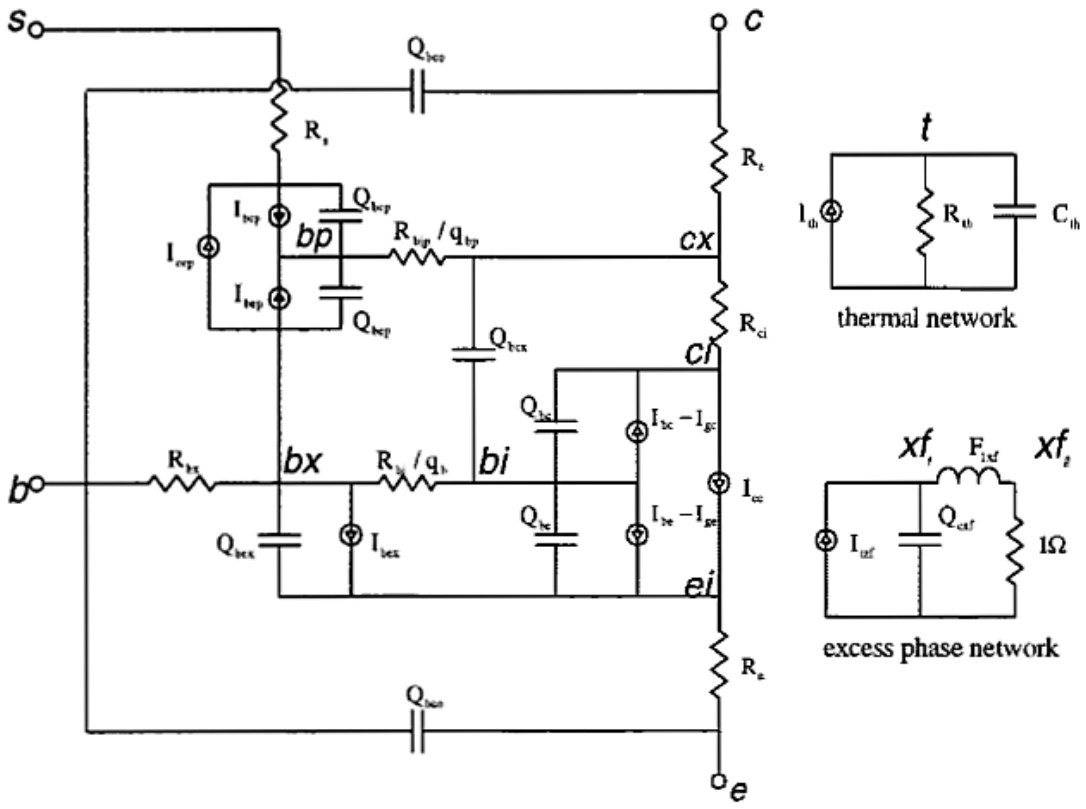


Figure 2.4: Equivalent circuit for the VBIC bipolar transistor model [2]

- The base current and the collector current are not relevant through a constant current gain as in the G-P situation
- Quasi-saturation behavior in VBIC model
- It is available in commercial circuit simulators

2.2.4 Issues of VBIC model for HBT modeling

Currently, Existing industry models such as Gummel-Poon(G-P) model and VBIC model are not sufficient to characterize the GaAs/AlGaAs HBT device. VBIC is not perfectly suitable for GaAs/AlGaAs HBT devices because:

- The ideality factors of base-collector and base-emitter current components in HBT is variation with temperature, which is not modeled by VBIC model [3].
- The physical effects involve with high current effects in VBIC model dose not match the GaAs/AlGaAs HBT device.

- The thermal resistance of HBTs is usually to be temperature dependent, which is not modeled by VBIC model [3].
- Relatively high operation current densities make HBTs very sensitive to the base push-out (Kirk) effects, which are required more accurate modeling of transistor time and the current-dependent collector capacitance [3].
- The transistor time frequency is not right when VBIC model is adopted to the GaAs/AlGaAs HBT device.
- The relatively lightly doped collector of GaAs/AlGaAs HBTs make the mobile charge modulation, which is required to be take into account in collector capacitance [11]
- Collector punch-through has a no negligible effect on the capacitance [11]

2.2.5 Enhance VBIC model for HBT model

In order to perfectly match the the GaAs/AlGaAs HBT device, a modified VBIC models required in improve with transport current, transits-time and collector capacitance. And also the extraction and optimization of VBIC model parameters are required to be developed. Therefore, the advance VBIC model keeps all of the features of the existing VBIC model while consist of new features with a couple of additional model parameters that are capable of solve the previously effects.

2.3 Noise Sources

2.3.1 Noise Phenomena

There is various style of noise phenomena in electronic devices which result in a fluctuation of the signals. The most significant styles are thermal noise, shot noise, diffusion noise, generation-recombination noise, and flicker noise $(1/f)$. The amount of influence of one noise phenomena to the whole noise not only due to the device and its characteristic of operation but also on the operation conditions which is included temperature, bias point, and frequency. A detail of description of noise phenomena is presented in the following parts.

2.3.2 Thermal noise

Thermal noise is produced in any physical resistance that indicates dissipation when current is flowed through it. Thermal noise is a type of electronic noise effected by the random thermal motion of the charge carriers producing microscope current fluentuations. It was formulated first by Nyquist and Johnson in 1926, and is named as Johnson-Nyquist noise. Thermal noise is directly affected by the absolute temperature T and is not related

to applied voltage or current. In a resistor, thermal noise is called as white Gaussian noise. A series voltage generator or shunt current used to model thermal noise, which is defined as

$$\overline{v^2} = 4kTR \triangle f$$

or

$$\overline{i^2} = 4kT\frac{1}{R} \triangle f$$

Where R refers to circuit resistance, k refers to Boltmanns constant. △f refers to bandwidth in hertz and T stands for absolute temperature, which is at the room temperature approximate 300°K.

2.3.3 Shot noise

Different to thermal noise, shot noise is always depending on a direct current flow. It involves in diode and transistors. Shot noise is determined by fluctuation in current that is effected by carriers flowing through the potential barrier at the junction. Such as the emitter-base or base-collector junction in transistor. The noise current generated by shot noise can be defined as:

$$\overline{i^2} = 2qI_D \triangle f$$

Where q refers to the electronic charge which is equal to $1.6 * 10^{-19}$ C, I_D is the bias current.

The shot noise is also called as white noise and it is proportional to the bias current.

2.3.4 Diffusion Noise

Diffusion noise happens in semiconductors with electron density gradient d_n/d_x where the thermally activated electrons move from places with a high electron concertation to places with a low concentration. The noise is effected by collisions of the moving carriers with the lattice.

2.3.5 Flicker noise ($1/f$ noise)

Flicker noise refers to a physical property in all active devices. It has an enormous influence on circuit. Flicker noise is a low- frequency noise. It is also called as $1/f$ noise because noise is produced with a slope inversely proportional to the frequency. There is not lower frequency limit for the $1/f$ noise and the overall noise power can be infinite. Flicker noise can be expressed in term of voltage noise spectrum S_V or resistance noise spectrum S_R or current noise spectrum S_I or conductance noise spectrum S_G, and these spectra are

normally standardized. Such as S_V/V^2, S_R/R^2, S_G/G^2. In 1969, Hooge suggested that the fluctuations in conductivity will cause $1/f$ noise. Flicker noise is normally defined as

$$\overline{i^2} = K_f \frac{I_{DC}^{AF}}{f^{f^{FE}}} \triangle f$$

Where K_f refers to level fitting, A_f refers to the current dependency factor, and f_{FE} refers to the frequency dependent power factor.

2.3.6 Generation-Recombination noise

Generation-Recombination (G-R) noise is one style of electrical noise produced by fluctuations in the amount of free carriers in semiconductor materials.

2.4 Low-Frequency Noise Characteristics

2.4.1 Low-Frequency Noise in Nonlinear Systems

There are two dominant noises at low frequency which are flicker noise and G-R noise. Because of up conversion effects, low-frequency noise can play a crucial role in HBT model. Traditionally, bipolar large-signal models only consider low frequency noise produced at the base-emitter junction. The low-frequency noise behaviour of HBTs is important. To find out where the noise sources are located and which of these sources dominate, it is require to consider the dependences of the noise behaviour on transistor geometry, bias point, temperature and frequency.

2.4.2 HBT Noise Model

The large-signal model employed is the FBH model developed for GaAs-based HBTs. The equivalent circuit containing noise sources is indicated in Figure 2.5. All resistances R_b, R_{b2}, R_e, R_c shows thermal noise at the actual junction temperature T_j. Shot noise is contained by two correlated noise sources. The current I_{be} is drive the base-emitter shot noise. The current $I_f = \beta_f * I_{be}$ is drive the collector-emitter shot noise.

In Figure 2.6, all resistances generate thermal noise, $< |i_b|^2 >$ and $< |i_c|^2 >$ present shot noise. The sources at the base-emitter junction $< |i_n * f_b|^2 >$, and at the emitter resistance $< |v_n * f_e|^2 >$ generate low-frequency noise. Additionally, the model considers the second noise source which represent the Hooge noise contribution of the bulk emitter.

In the equivalent circuit of combination for both intrinsic and extrinsic of HBT, Figure 2.7, the location of this noise is defined as “base-emitter noise source”. In case of HBT models, the noise is effected by current and frequency.

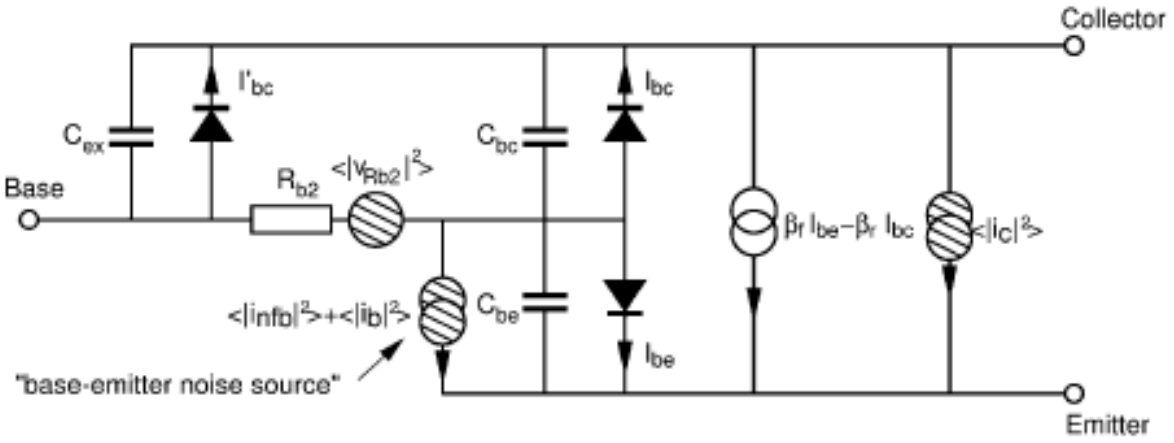


Figure 2.5: Intrinsic large-signal of HBT and noise equivalent circuit []

In Figure 2.7, the base and emitter resistances R_b, R_{b2}, and R_e are very low value in modern HBTs in order to achieve very low collector noise currents. Therefore, the base-emitter noise source is almost short circuited. The Oppositely, the noise for high-impedance source is not affected, because the emitter branch is almost connected to an open circuit which result the voltage noise source does not drive any current. The emitter noise source is mainly determined by the collector noise current.

2.4.3 How the low-frequency noise sources are implemented

- Low-pass noise sources

The modelling interface of the circuit simulator provide low-frequency noise sources.
In Figure 2.8, the noise level is effected by a function of current f (I), and the output of noise signal is low pass filtered.

- Cyclostationary noise sources

In Figure 2.9, a function of current f (I) is multiplied with a constant low-frequency noise. A sub circuit is implemented by using the instantaneous current when calculating the noise source through the simulator in order to get a cyclostationary source.

- Partly correlated cyclostationary noise sources

In Figure 2.10, the implementation is quite similar to the cyclostationary source, however it need to separated noise sources, and high-pass and low-pass filtering of the current.

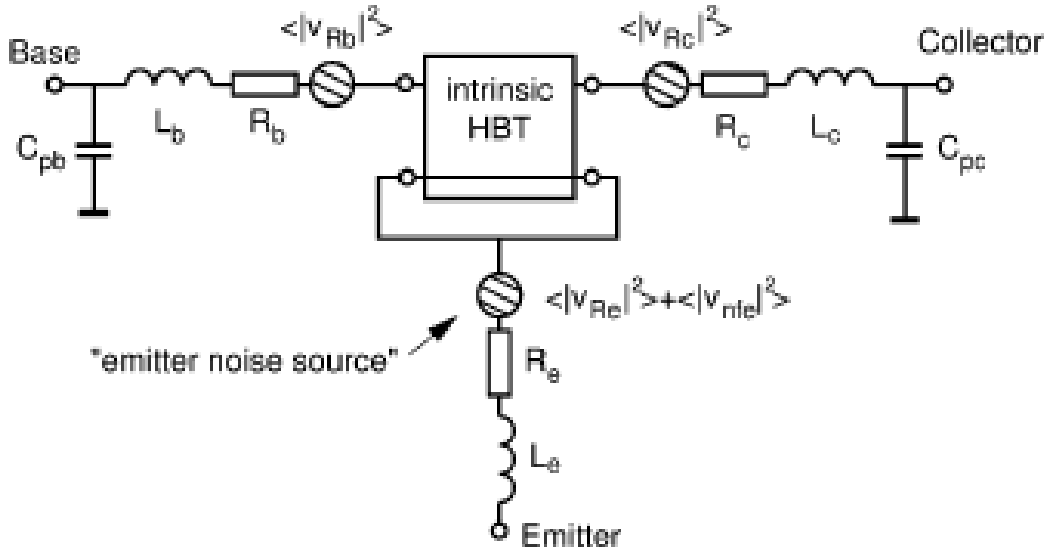


Figure 2.6: Extrinsic large-signal of HBT and noise equivalent circuit []

2.5 Verilog-A

Verilog-A is a procedural language, with constructs similar to C and other languages. It offers simple constructs to describe the behavior of model for the simulator program. The program language allows circuit designers to create modules (behavioral blocks in a circuit) and define the characteristic of modules by specifying the mathematical relationships between the currents and voltages terminals of the modules. Verilog-A was recently enhanced to offer greater support for compact modeling. The NI AWR Design Environment provides a complete software solution, which promotes the compact model development. Due to its ease of use, efficient system design and powerful circuit simulation. Verilog-A can be used directly in circuit simulators - as well as parameter extraction of NI AWR Design Environment, which enable to handing measured data.

2.5.1 Verilog-A Modules

A module is a unit of Verilog-A code that is used to describe a component. In Verilog-A, the built-in primitives depict common circuit components, such as capacitors, inductors, resistors and semiconductor devices. A module that simplify refers to other modules is known as a structural model. In Verilog-A, components are constructed by using branches and nodes. A node is defined as a point where endpoints of braches may connect, and a branch is defined as a single path between two terminal nodes. Nodes and braches are constrained by Kirchhoffs laws, which is represented as below:

- The flow enter into a node must always sum to zero.
- The potential is the same everywhere on a node

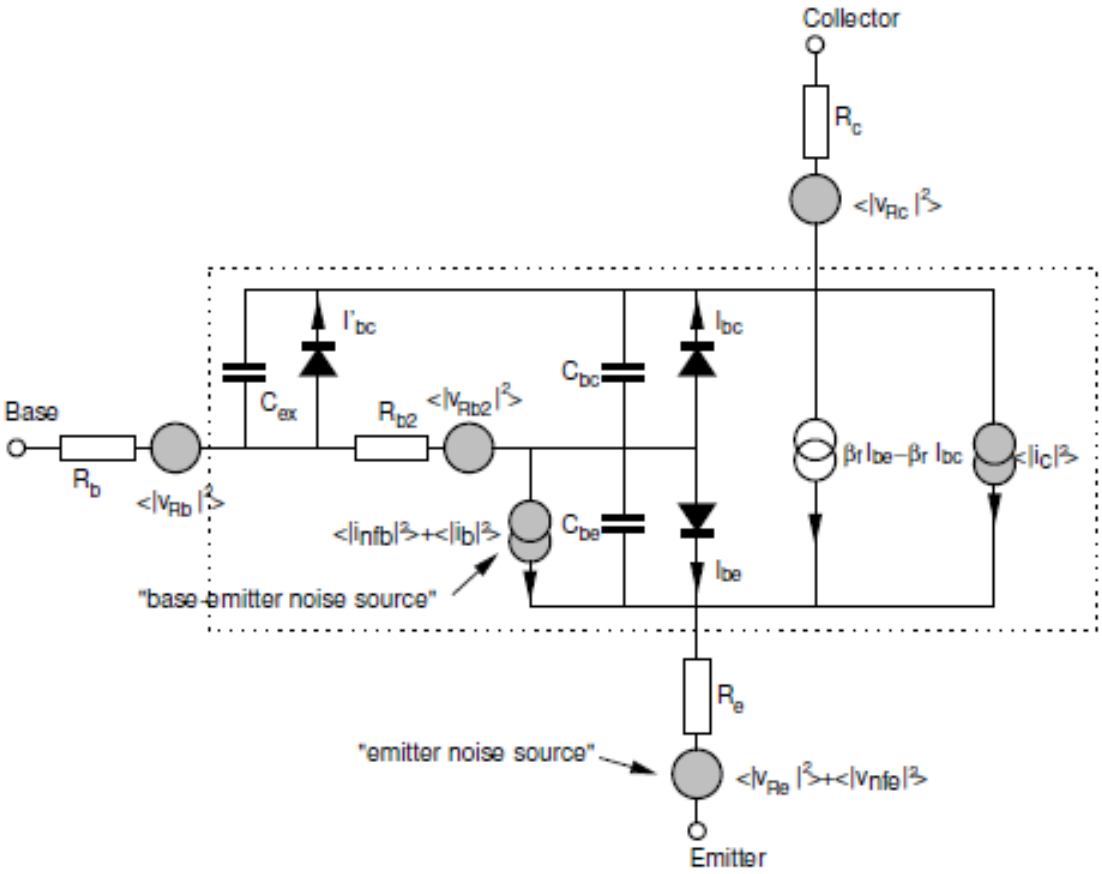


Figure 2.7: Large-signal and noise HBT equivalent circuit []

- The potential on a branch is equal to the difference potential between two nodes, which are connected.
- The flow enter into one terminal of a branch is always equal to the flow out of other terminal [1].

2.5.2 Advantages of Verilog-A

The main reason for preferring Verilog-A for compact modeling rather than other general-purpose programming languages is free the model developer from the burden of dealing with the simulator interface.

Generally, the modeling user interface of simulator in Brand A is quite difference to other modeling user interface of simulator in using and definition of variables. As a result of this, is very hard to implement the same model in different simulators without extra programing errors and compiling warnings, at the first trial. With Verilog-A, new models

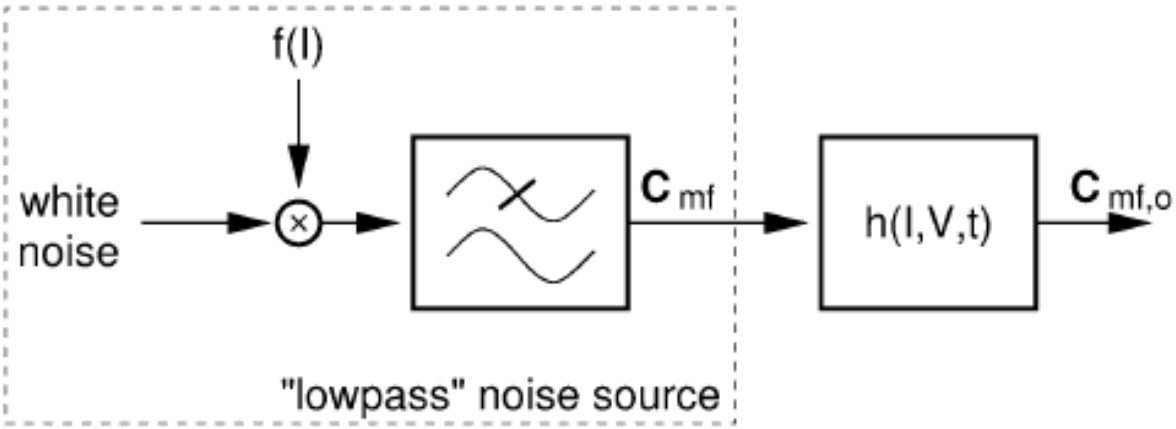


Figure 2.8: Schematics illustrating the behaviour of low-frequency noise in large-signal excitation (low-pass noise) []

can be added nearly as quickly as the equations can be defined. Furthermore, it can maintain control of the intellectual performance of the model within various simulators.

Verilog-A also offers a powerful system for defining model parameters. The declaration statement contains the default value and can specify the range of effective values. The default value may also define as a function of other parameters, which is very useful to determine the relationship between input signal and output signal. As a result, Verilog-A is an extremely efficient program language for writing compact models [4].

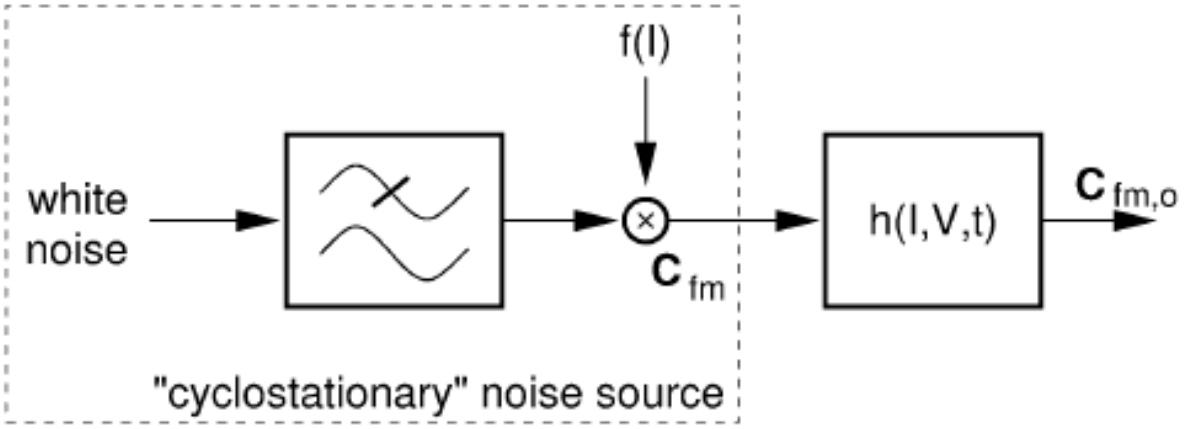


Figure 2.9: Schematics illustrating the behaviour of low-frequency noise in large-signal excitation (cyclostationary noise) []

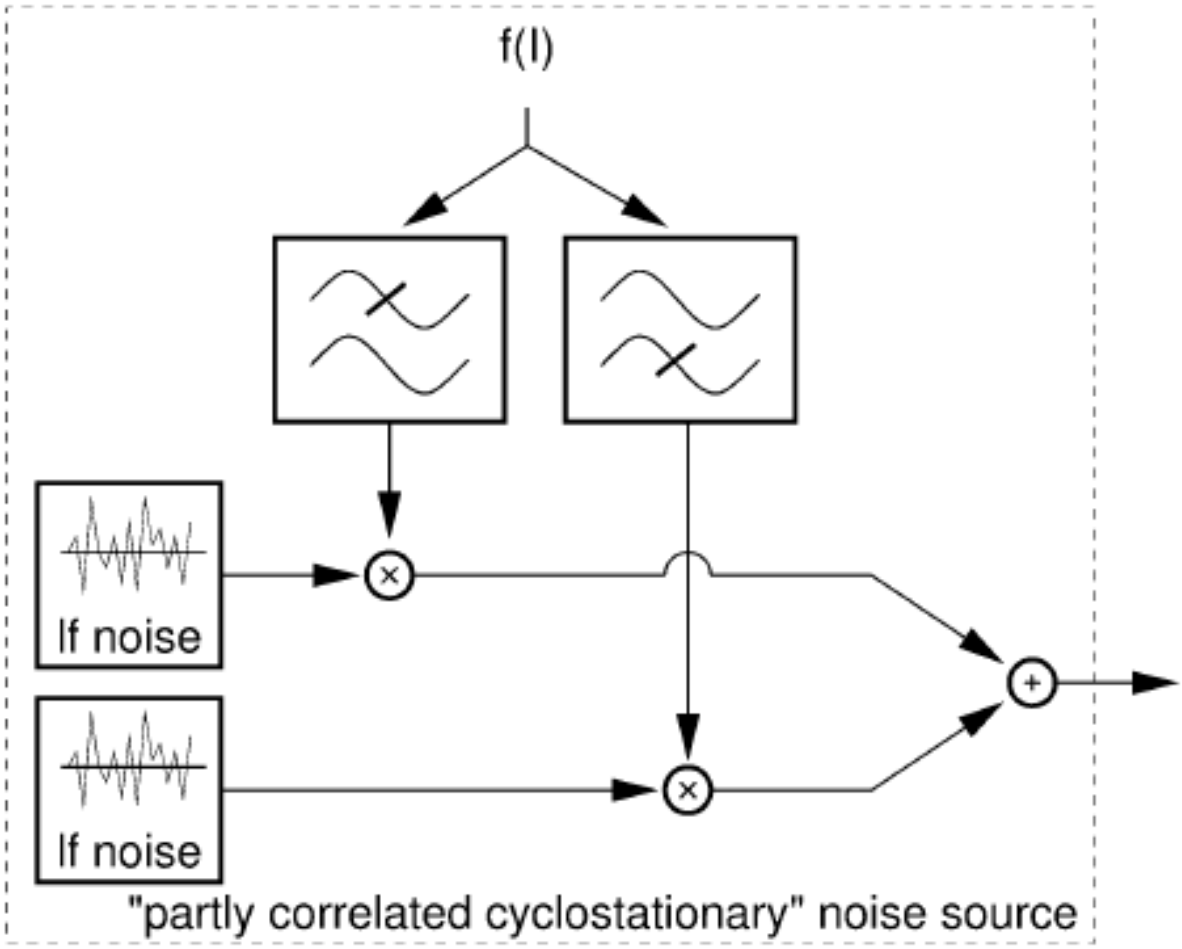


Figure 2.10: Schematic indicating implementation of the partly correlated cyclostationary noise source []

Chapter 3

PDK Development Training

3.1 PDK Overview

A process Design Kit is a complete set of files used within the semiconductor industry to model a fabrication process as the design tools, which are critical for design an integrated circuit. The PDK is identify by the foundry defining a certain technology variation for their process. The PDK can be used for design, simulate, draw and verify design, which improving design more efficiency, reducing the design cycle times and ensuring quality [6]. Basic on Figure 3.1 explains the detailed information of PDK and how it operation. The elements of a PDK are Device Symbols Device Symbols & View, Model, Parameterized Cell(PCell), Technology File, PV Rule and Documentation. These elements are required for each design state.

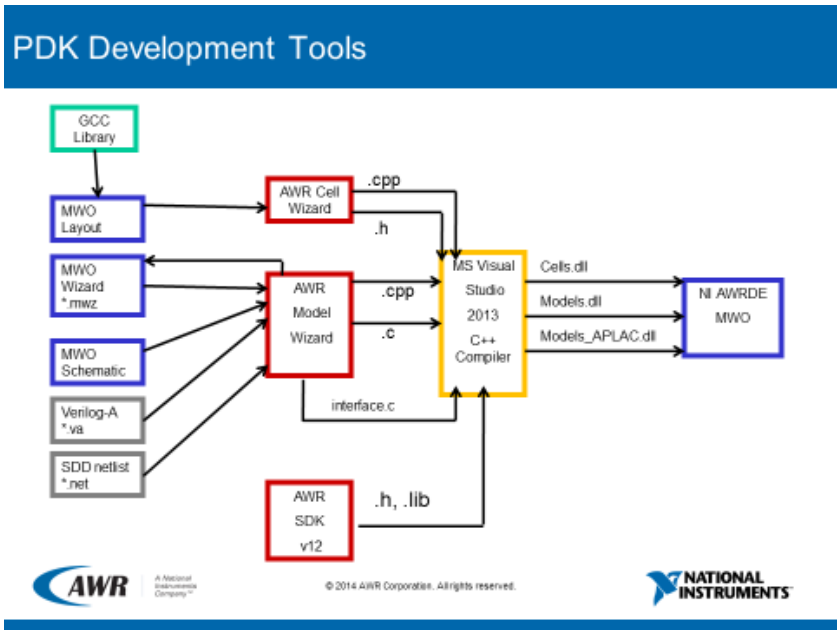


Figure 3.1: PDK Development Tools [5]

3.2 Installation Guide for AWR Foundry SDK and the Model Wizards

In order to approach the thesis project, we start with PDK development training. To get familiar with how implement model wizard in AWR and using Microsoft visual studio to test model. The NI AWR version 12 and Microsoft visual studio 2013 is required to download. The software development Kit(SDK) is download from the AWRCorp. web site. The AWR_Dev Folder includes:

- AWR Folder includes the sample solution TEST_SDK, which is used for verifying SDK installation and working samples
- Documentation Folder-includes training presentations and reference documents
- Generic-includes a sample PDK used for training
- SDKv12-includes the Model and Cell Wizards, and also contain the Lib files required to build PDK Solutions
- Supporting Files-include example projects [5]

According to instruction, we registering Model and Cell Wizards, ensure wizard installed correctly. Open the file AWR_Dev\AWR\AWR.sln in Microsoft Visual Studio. We got the below message, which confirms the visual studio and AWR SDK setup correctly.

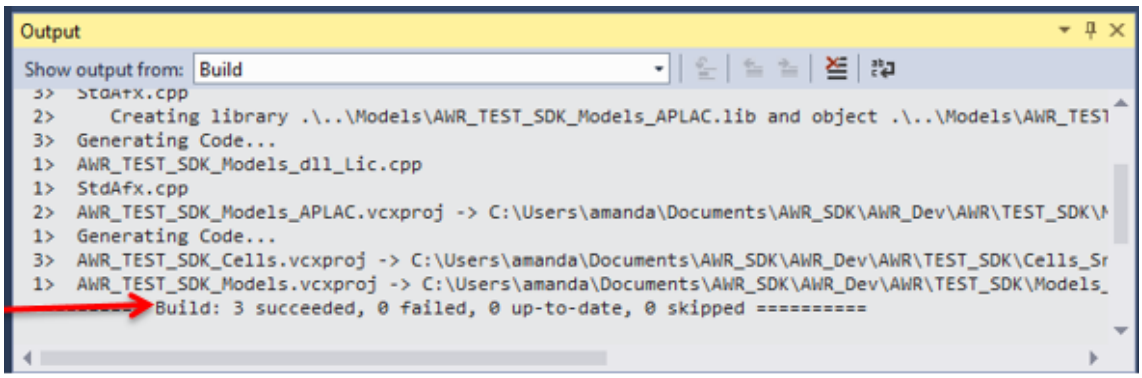


Figure 3.2: The output window of Visual studio [5]

The Script is used to generate a new PDK process with a different name. We use companyName, CoName, and ProName as general names, which is showed as below:

The new PDK directory structure has all Visual Studio project. Open Microsoft visual studio allows to build the projects and verify the project build.

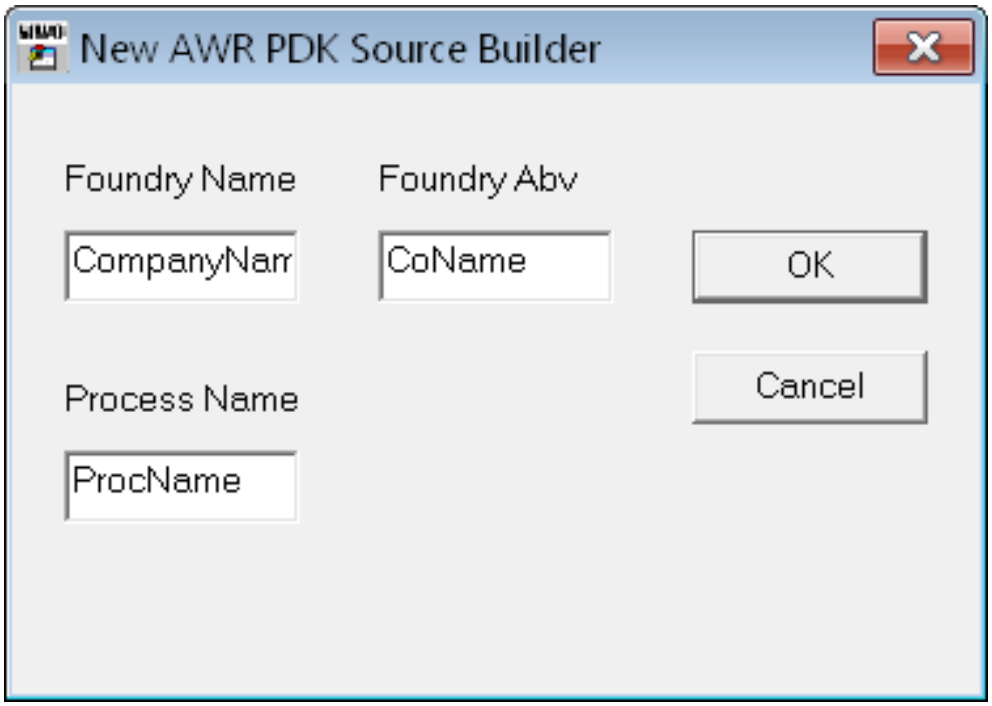


Figure 3.3: The "New AWR PDK Source Builder" dialog [5]

3.3 Getting started with the model Wizard

3.3.1 Overview

- The NI AWR design environment has various circuit simulators. Two simulator is named as AWR and Aplac. Both of them has linear and non-linear simulator. Both AWR and Aplac simulators can be used for any linear model. Non-linear models will require a dll for AWR and a different dll for Aplac simulator.
- Creating an aggregate model by importing it from the AWR schematic into the model wizard. Schematic can include both linear and nonlinear elements.
- The Model Wizard will produce C++ code, which is used for defines a new element.
- The following is an traning by using the wizard to get code for a MIM capacitor [5].

3.3.2 MIM Cap imported from schematic

Start NI AWRDE and open the 'AWR_CAP_HF.emp' project file. No only we are allowed to fill the properties by modify Model parameters, and also allows to enter parameter formula, which define the behavior of capacitance. We launch Visual studio after the

model wizard create the .cpp file. Open the solution file by using MS Visual Studio. MS visual studio allows source code of AWR_CAP_HF to be add into the project. Therefore, we can test the model.

Create the schematic of cap and add S21 Smith Chart Graph. Edit and continue option in Visual Studio enables the user to debug a model source code by inspecting at the code and the simulation results in NI AWRDE simultaneously. By modify the equation for C to read C=CA*W*L. S21 Smith chart Graph is changed shows as below:

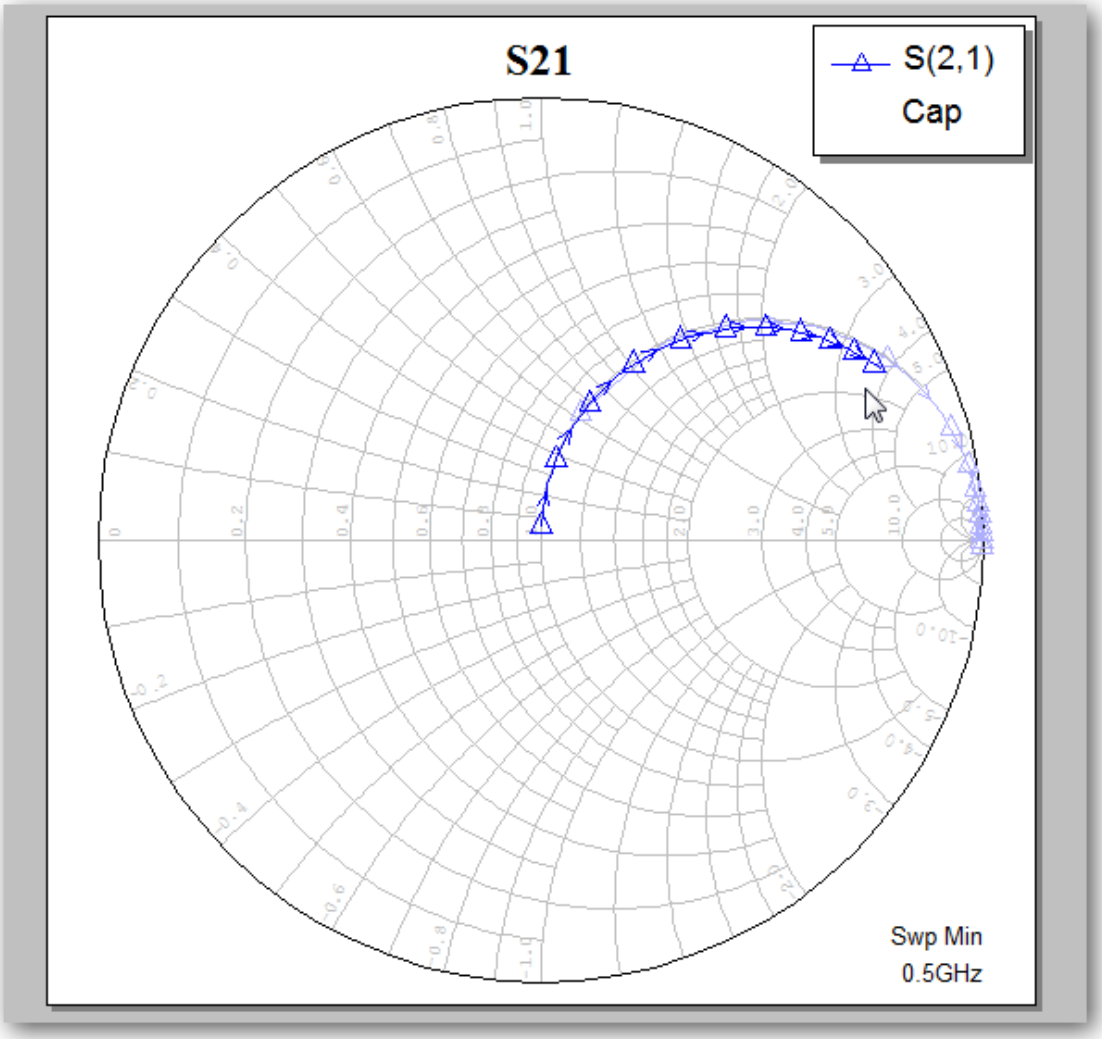


Figure 3.4: Result of Code change [5]

Edit and continue is effective for debugging models and making modify easily.

3.3.3 Curtice FET with Parasitics

Overview

- This time will again be an aggregate model from a schematic
- The model is a Curtice FET with some extra parasitic.
- We add an Enumerate parameter to set a gate-gate pitch value
- The parasitic as a read-only parameter [5]

The execute procedure is pretty much similar to MIN Cap, instead we use Curtice FET and add more extra parasitic. There are some issues when add the model, which is named as Curtice_2P. It shows as below:

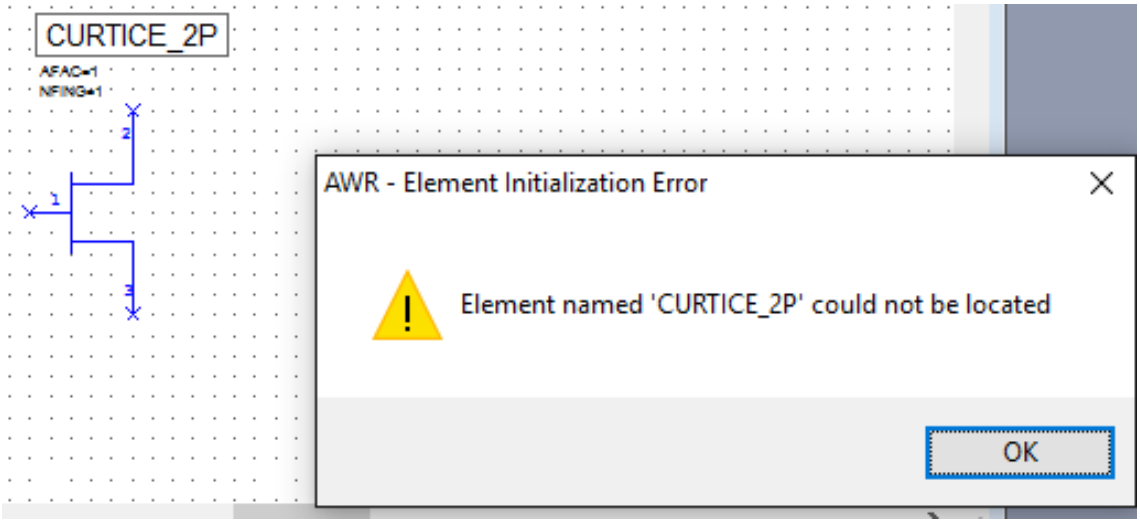


Figure 3.5: CURTICE Model [5]

Figure 3.6 , because we tick hide model name. Therefore, NI AWR could not find element, which is named as 'CURTICE_2P'.

Basic on Figure 3.7 , to determine the IV curve of CURTICE _2P.

3.3.4 Measure Internal Branch Current/Voltages

AWR nonlinear models allow to access the internal nodes of a nonlinear model by using branch labels. Adding branch labels directly to the Verilog-A allows the model wizard to deal with all of the branch labels automatically. Understanding Verilog-A is very important for achieve accurate models.

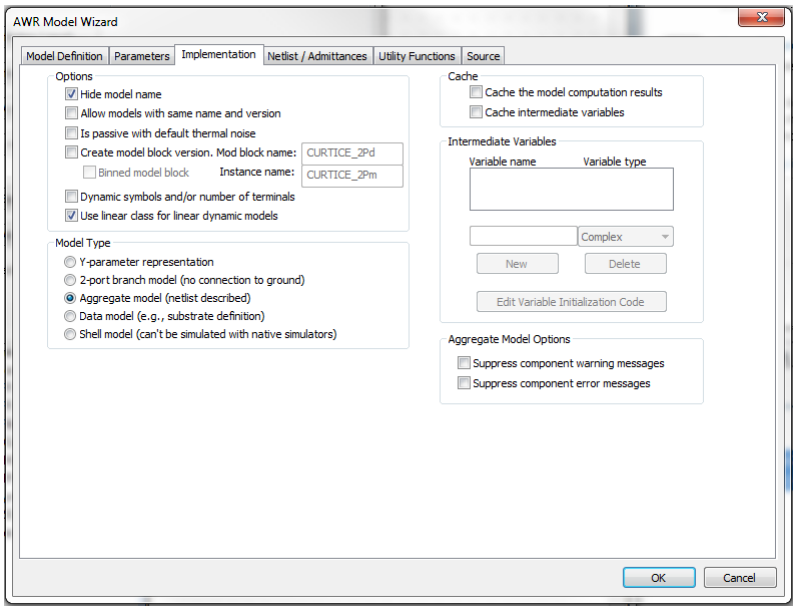


Figure 3.6: NI AWRDE element browser [5]

According to Figure 3.9 , the first line defines the name of the component, in this case is resistor. Two pins, which are nodes are t1 and t2. The second line declares the nodes as being electrical, meaning that the potential of each node is a voltage and the flow into the node is a current. Verilog-A defines the flow into the pin to be positive if the motion is into the component. The third line declares a parameter, r for the component. The values of parameter can be specified when we instantiate the resistor. The value given in the parameter declaration is set as the default value of the parameter. The value of r is keep constant within the module. The fourth line declares a branch is named as res and it present t1 and t2 pins are connected [1].

V(res) <+ r*I(res) using of the contribution operator (<+) achieve this a contribution statement. It represents an equation that the voltage on branch res must equal to the current through branch multiplied by r [1].

3.3.5 Analysis of issues

There are some issues occur during PDK development training. These may involve with AWR Version issue in simulation. The one we use is AWR V12, however the PDK development training is implement in AWR V11. Somehow, the simulation is not running in AWR V12, which may because some properties of AWR V12 is different with AWR V11. Configurations for Win32 platform and x64 platform used for test the model in Visual Studio, which only operate successful in x64 platform, but not Win32 platform. It may because our computer is running at 64-bits, that may not compatibility with Win32 platform.

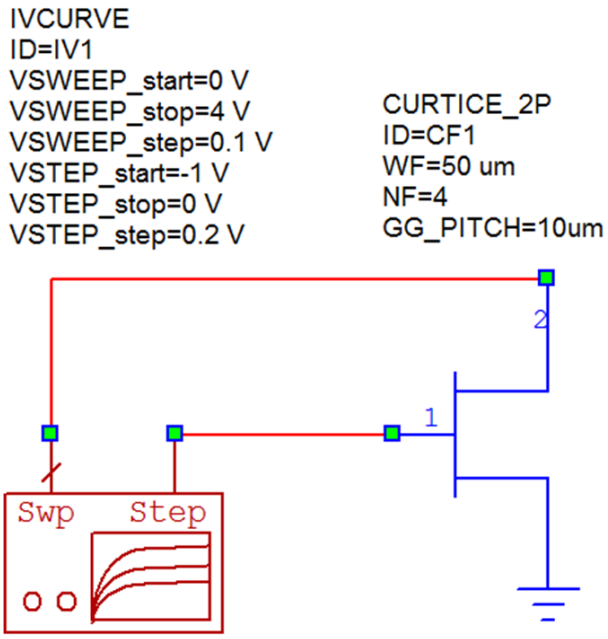


Figure 3.7: Connect the VSWEEP_step param to 1.0 to the Curtice_2P [5]

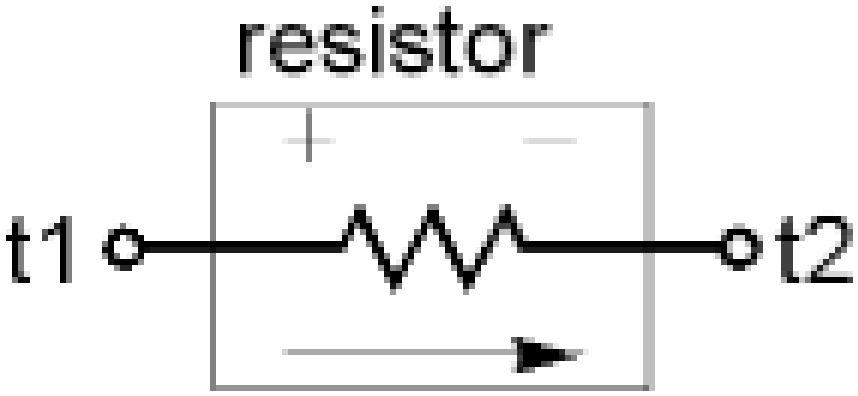


Figure 3.8: Resistor Model [1]

```
module resistor (t1, t2);
electrical t1, t2;
parameter real r=1;
branch (t1, t2) res;

analog V(res) <+ r*I(res);
endmodule
```

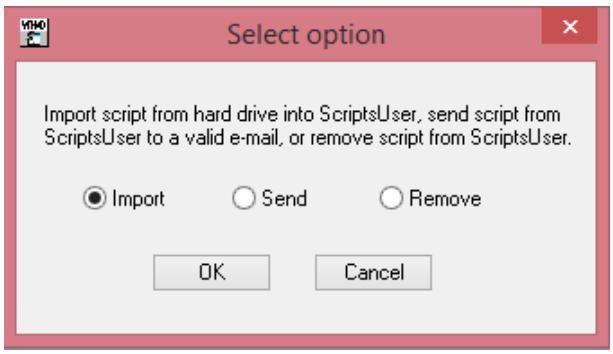
Figure 3.9: Resistor in Verilog-A [1]

Chapter 4

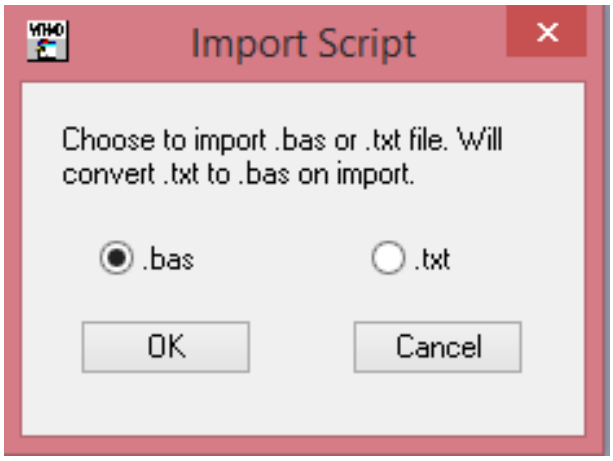
Create models in NI AWRDE PDK

4.1 The Procedure of creatiing models

- Start NI AWRDE and choose Scripts → Configuration → Import_Send_Remove_Script



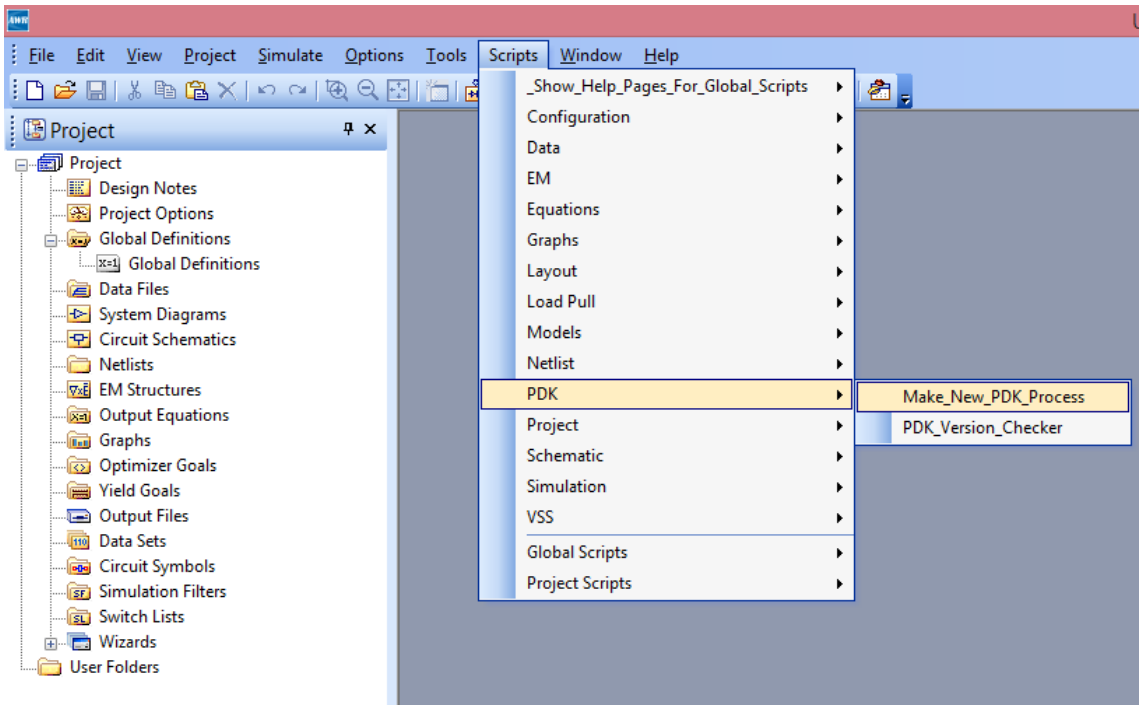
Select Import and press OK.



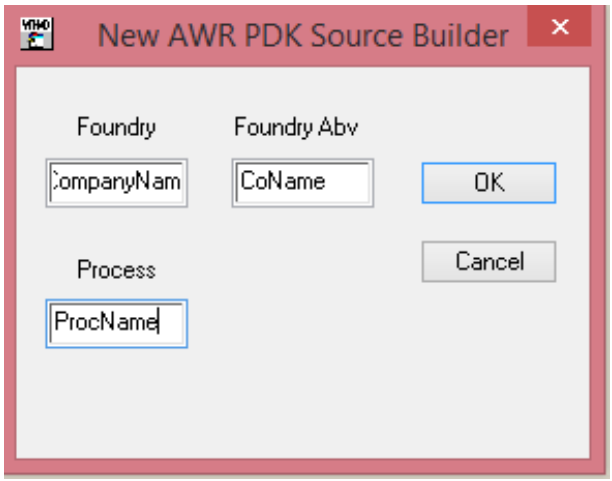
Select .bas and press OK.

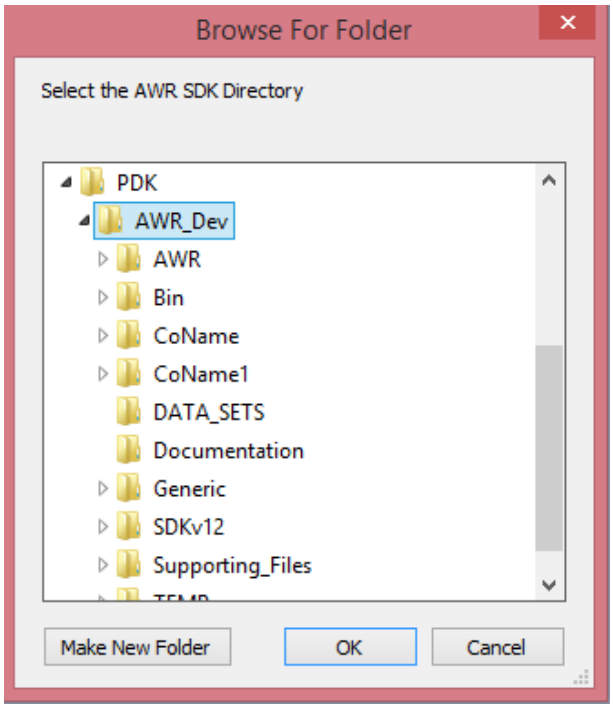
- Browse to AWR_Dev\SDKv12\ scripts\ MakeNewPDKProcess.bas to import it as a global script. The script will be appear in the Scripts menu when we restart Microwave office.

- Choose Scripts→PDK→Make_New_PDK_Process



- The "New AWR PDK Source Builder" dialog is appeared. Fill it out.

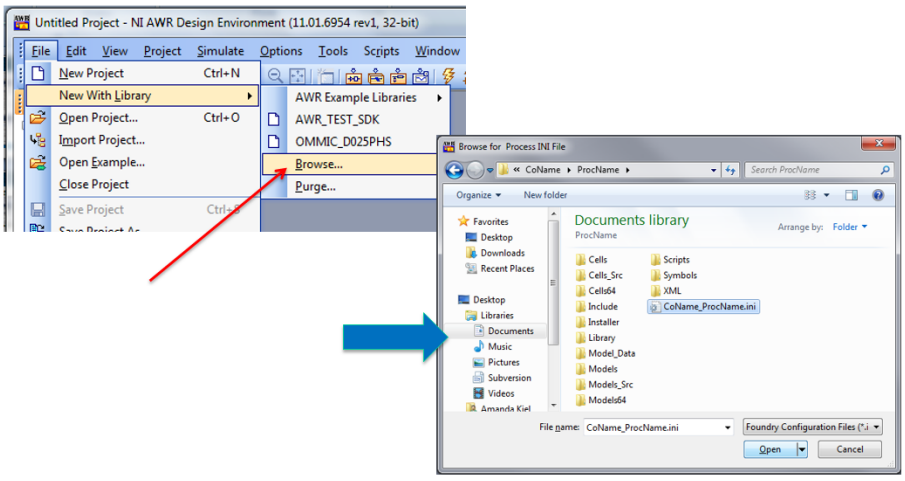




- Press OK. Select the AWR SDK Directory. In our case, we select AWR_Dev\ SDKv12 folder.

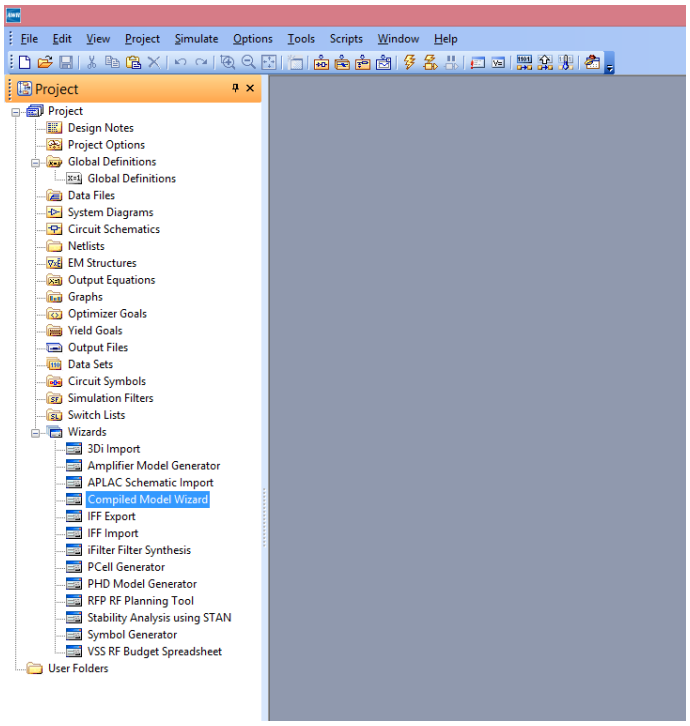
Therefore a new PDK folder will be created in the AWR_Dev folder.

- Add the CoName_ProcName PDK to the project file by choosing File→ New With Library Select "Browse" and navigate to the file CoName_ProcName.ini then click open.

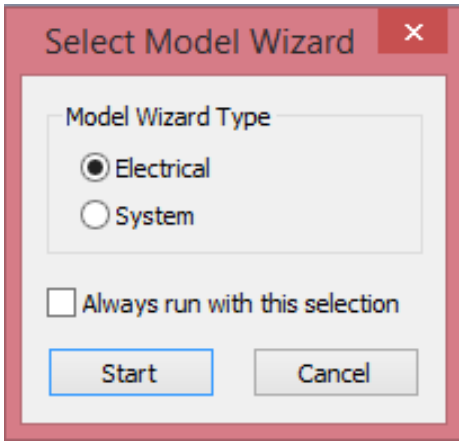


4.1.1 Define the model

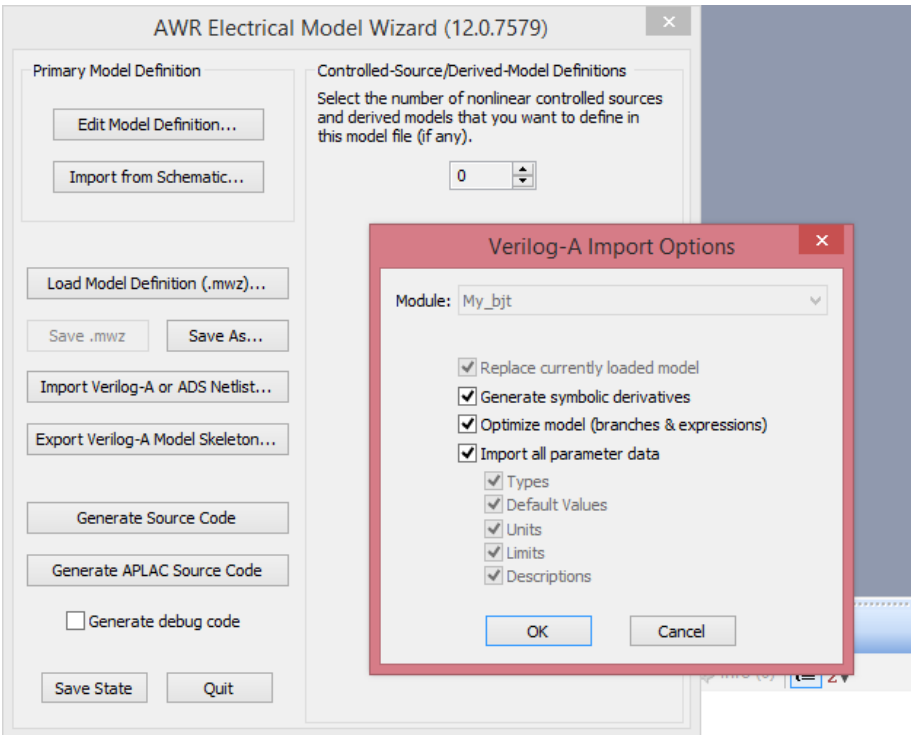
- Launch the Model Wizard from the Wizards branch of Project Browser.



- Choose 'electrical' as the model wizard type and import Verilog-A of My_bjt.



- Press 'Edit Model Definition'. In the Model Definition tab, enter the model description, which is called as My bjt. Model category is defined as user\nonlinear. Choose BJT@system.syf as our Schematic Symbol. Delete the contents of the Help file and Help context ID.



- In the parameters tab shows there are 62 model parameters in our bjt model. No Modifications Needed.

- In the Implementation tab, untick 'Hide model name' to allow the model from appearing in the NI AWRDE element browser. Aggregate model is chosen automatically.

- No Modification Needed in Netlist/Admittances tab.

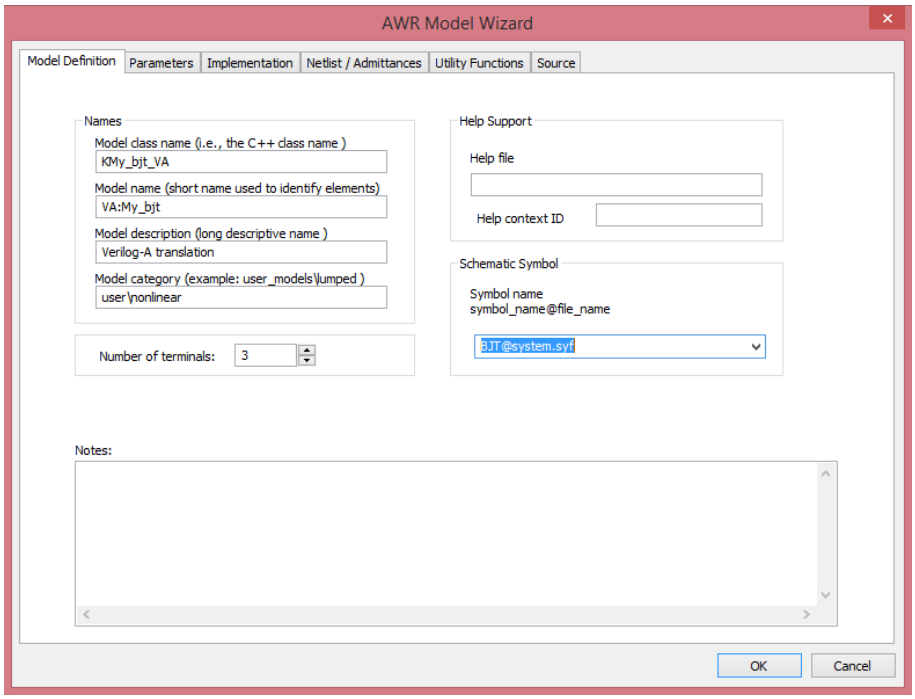
- Finally go to the Source tab. Set the output filename to "My_bjt.cpp." Click OK to finished editing the model.

- Save the Model Definition in the folder: AWR_Dev\CoName\ProcName\Models_Src with the name My_bjt.mwz.

- Click on 'Generate Source Code.' The model wizard will generate the .cpp file and launch Visual studio.

There is 1 Controlled-Sources so it is necessary to generate APLAC source code. In order to operate APLAC DC Simulator in later section.

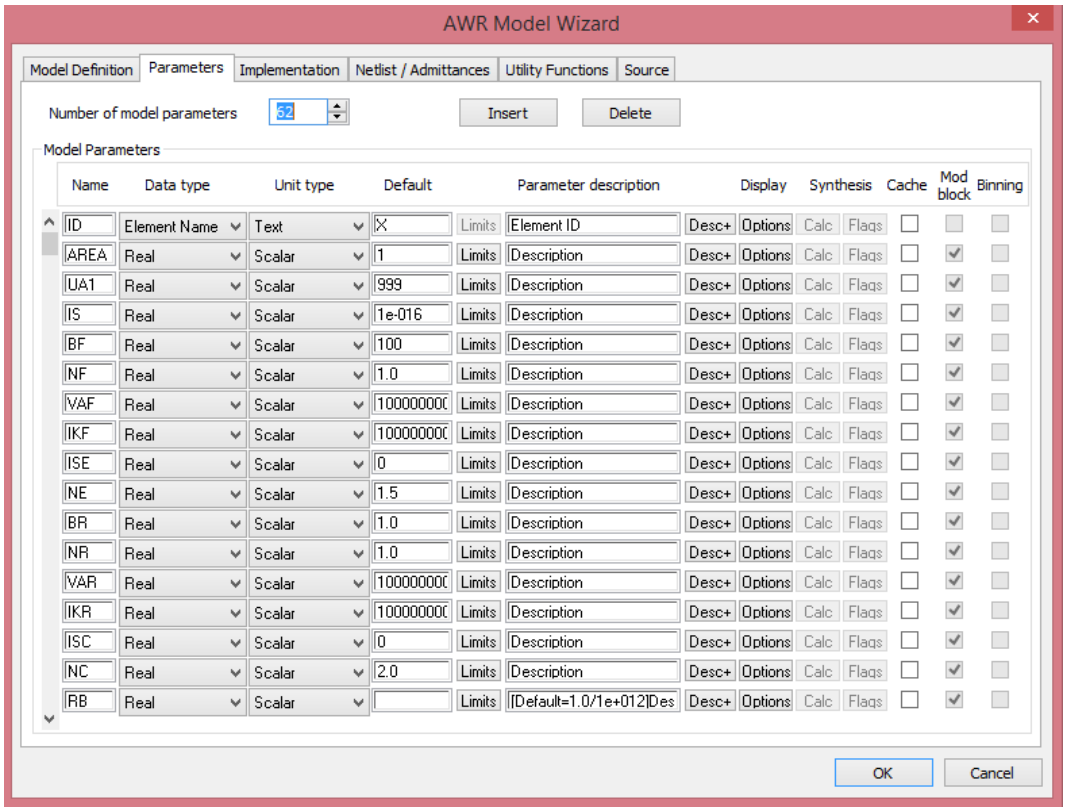
- From the file menu, open the solution file AWR_Dev\CoName\CoName.sln. Right clicks on CoName_ProcName_models and select Set as Startup project. Click on the dropdown for the CoName_ProcName_Models project.



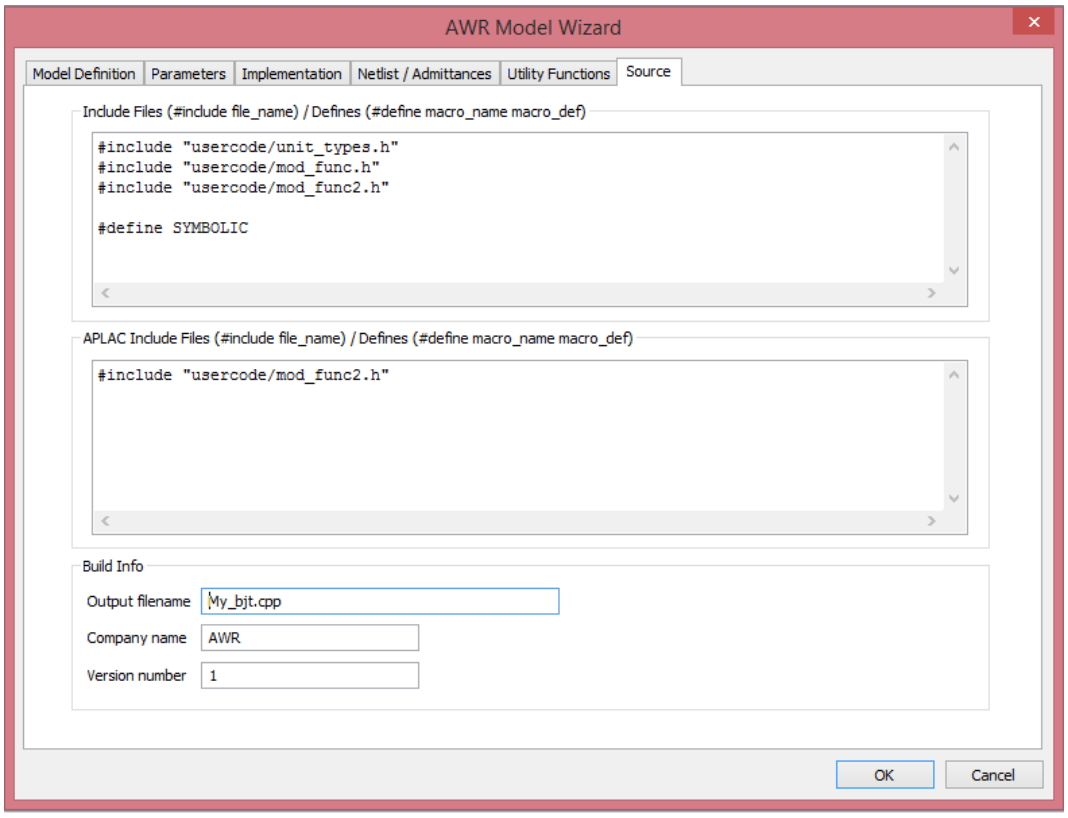
- Right mouse clicks on 'Source File', select Add→ Existing Item . . . and add My_bjt.cpp. Right mouse clicks on CoName_ProcName_Models and select Rebuild to compile the My_bjt.cpp file and create a model dll.

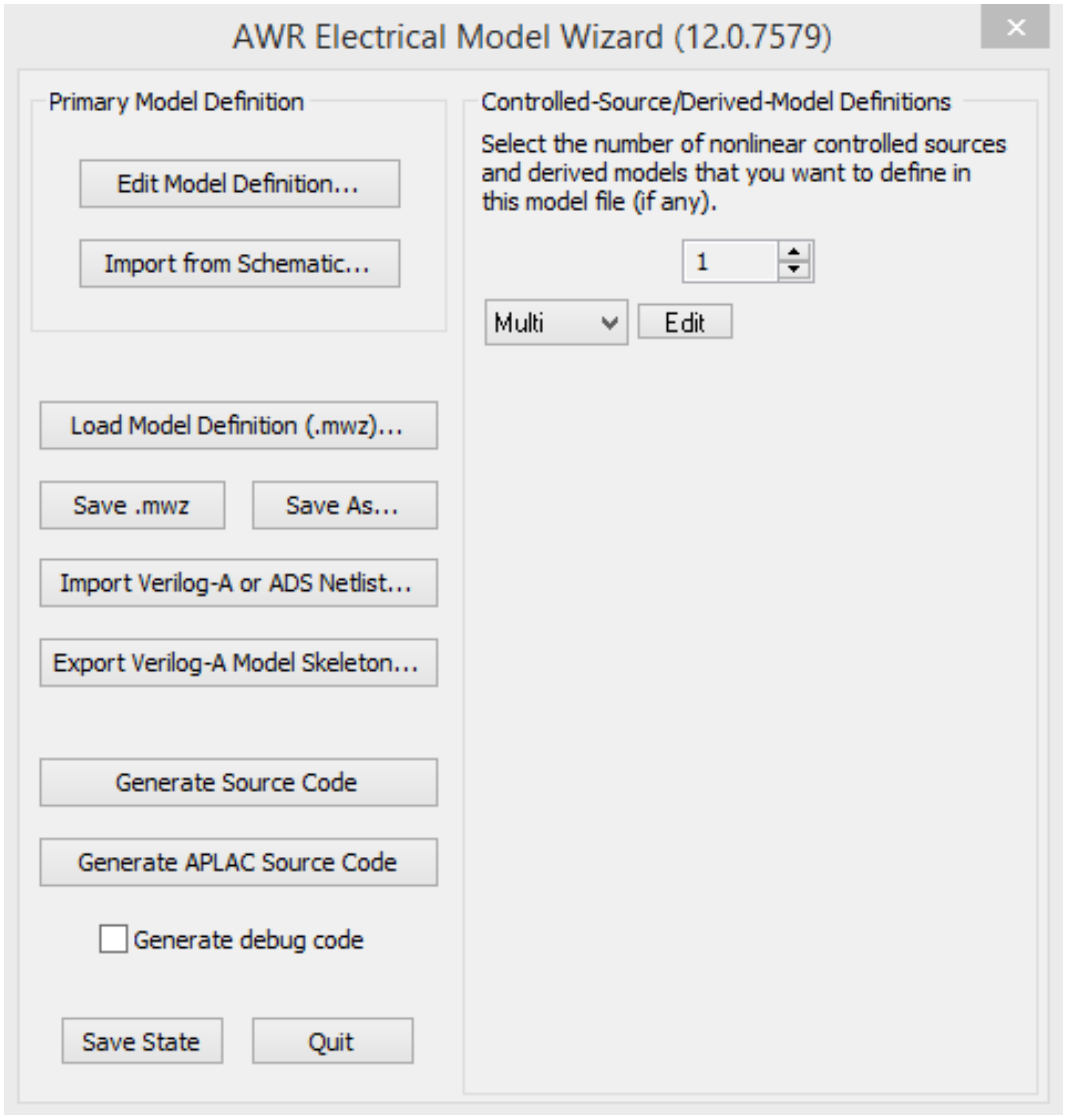
- The output window should show a message that the rebuild is successful.

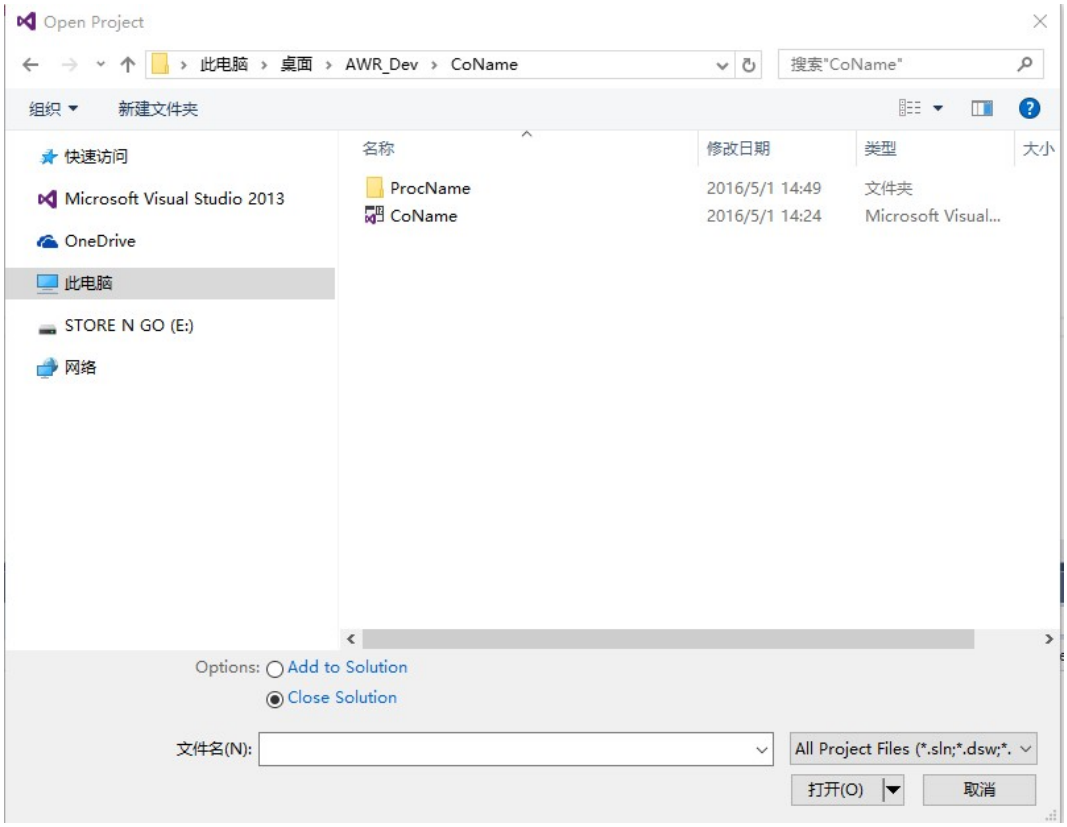
- Press 'Generate APLAC Source Code' and Select 'InterfaceTable.c' to create a .c file. Right click on CoName_ProcName_models and select Set as Startup project. Click on the dropdown for the CoName_ProcName_Models_APLAC project. Right mouse clicks on 'Source File', select Add→Existing Item . . . Add My_bjt.c and InterfaceTable.c that can be used to create an APLAC model dll. Double click on the file InterfaceTable.c in the Source Files list to bring the file into the Visual Studio Editor. Copy the model name from InterfaceTable.c and paste the name into the file AWR_netlister_APLAC.cpp in location of AWR_ADD_OBJ_TO_TRANSFORMER (basic_APLAC,"First_Model_Name_Here", MWO_TO_APLAC) for first model name. In order to add more models by remove the // at the beginning of the line, therefore the line is no longer a comment. The CoName_ProcName_netlister_APLAC file is used to give the proper model name in NI AWRDE when creating the APLAC netlist. To add additional model names to the APLAC netlister, Removing the "//" from the beginning of the line. Paste the model name into the 'AWR_ADD_EXISTING_OBJ' macro when require to create additional model name. Right mouse clicks on CoName_ProcName_Model and choose 'Rebuild' to update the models dll. Rebuilding to allow automatically save the modification in the file CoName_ProcName_netlister_aplac.cpp.

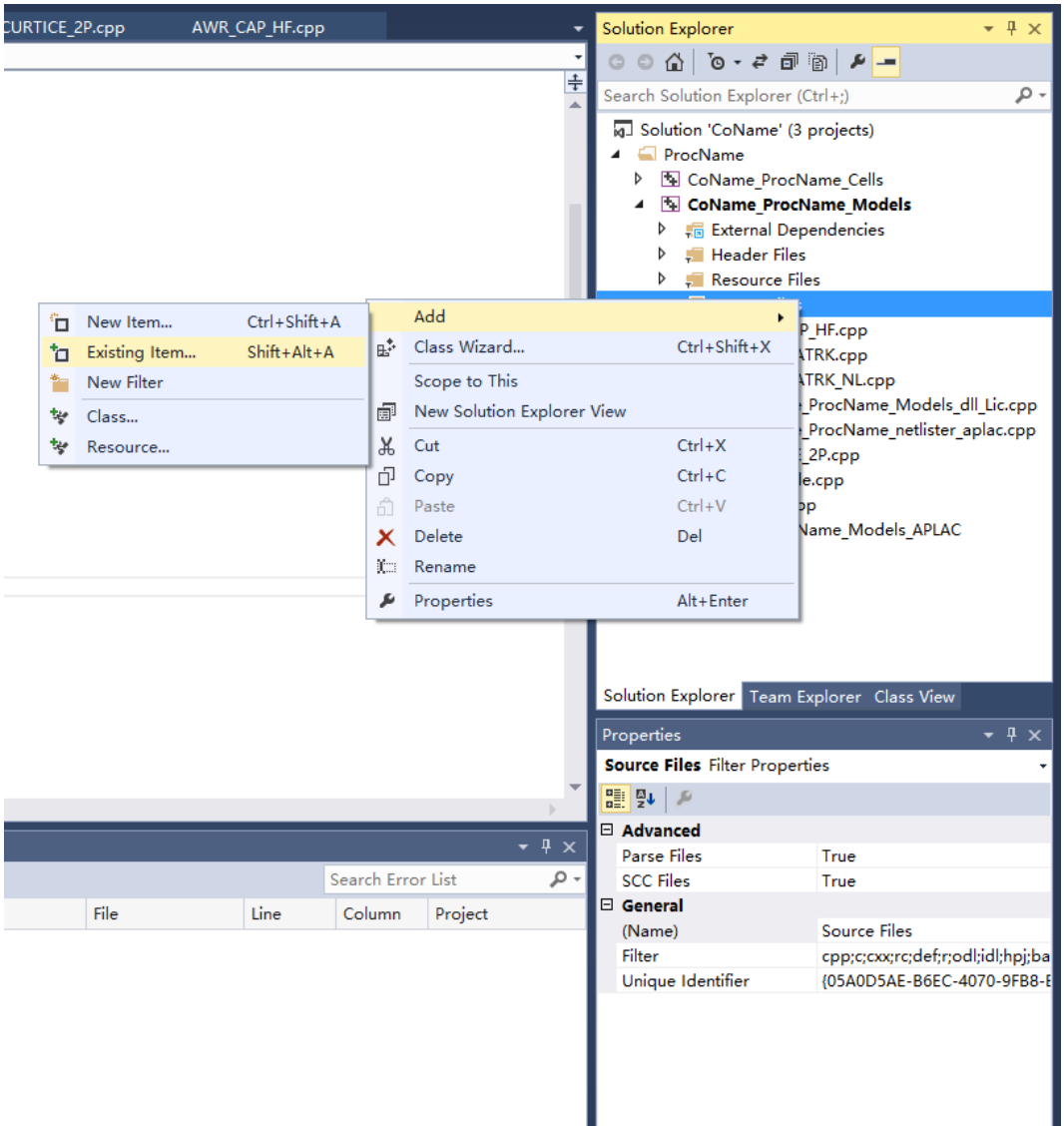


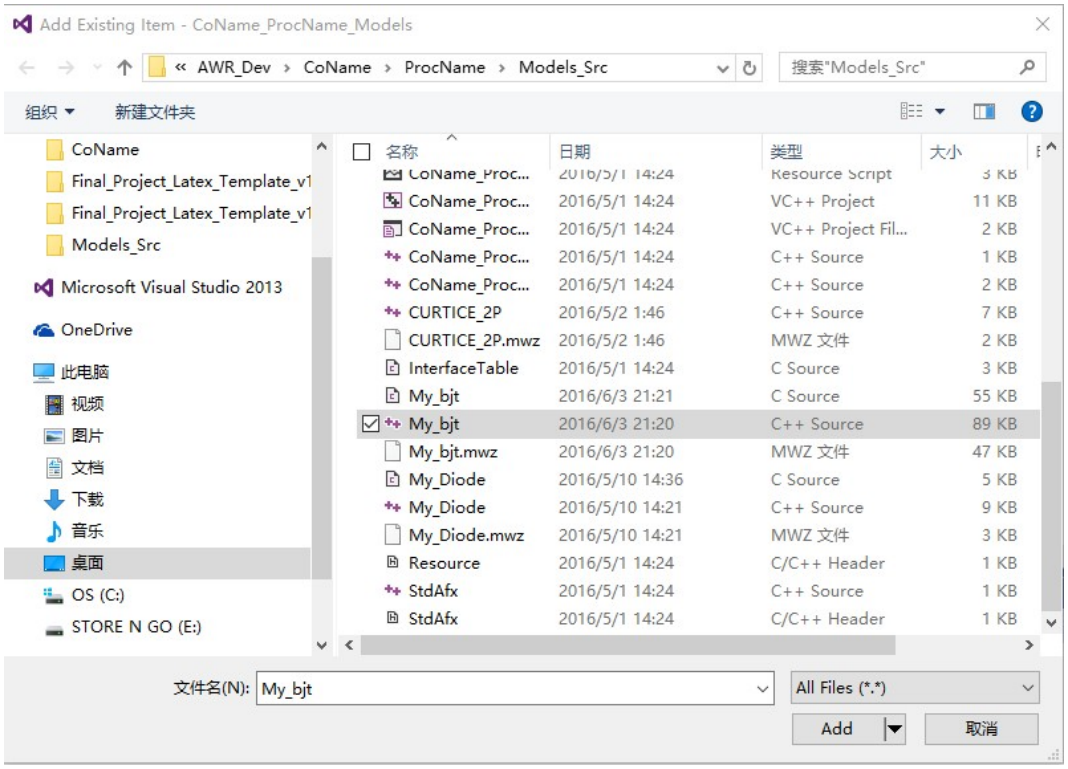
- Restart NI AWRDE and open new library with → CoName_ProName. In elements tab, we will find our model in user\nonlinear.

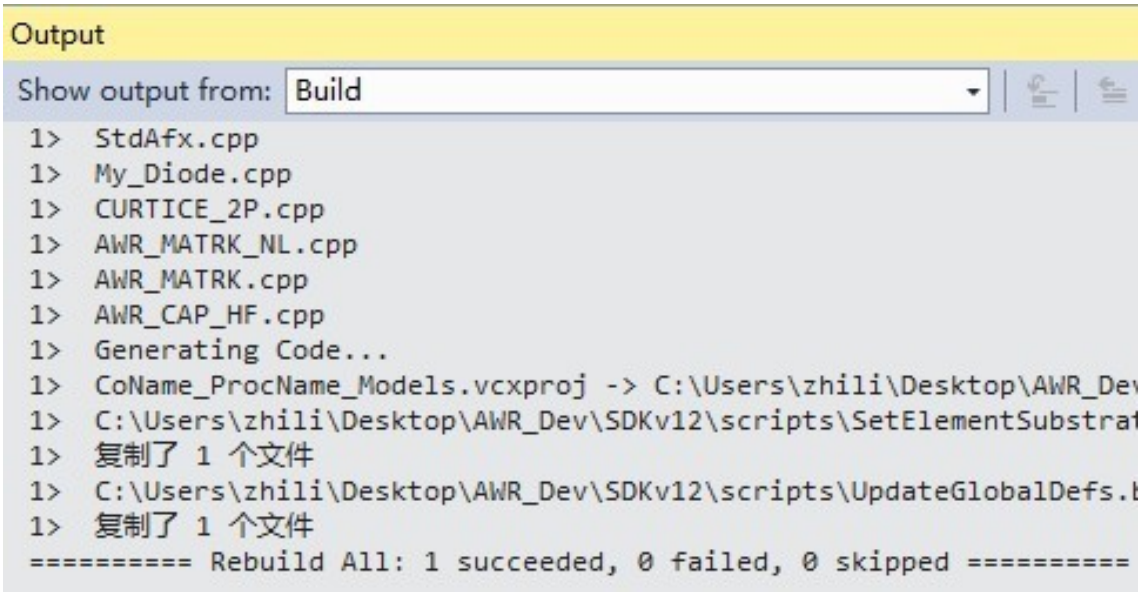

AWR Model Wizard
Model Definition
Parameters
Implementation
Netlist / Admittances
Utility Functions
Source
Include Files (#include file_name) / Defines (#define macro_name macro_def)
#include "usercode/unit_types.h"
#include "usercode/mod_func.h"
#include "usercode/mod_func2.h"
#define SYMBOLIC
APLAC Include Files (#include file_name) / Defines (#define macro_name macro_def)
#include "usercode/mod_func2.h"
Build Info
Output filename
My_bjt.cpp
Company name
AWR
Version number
1
OK
Cancel

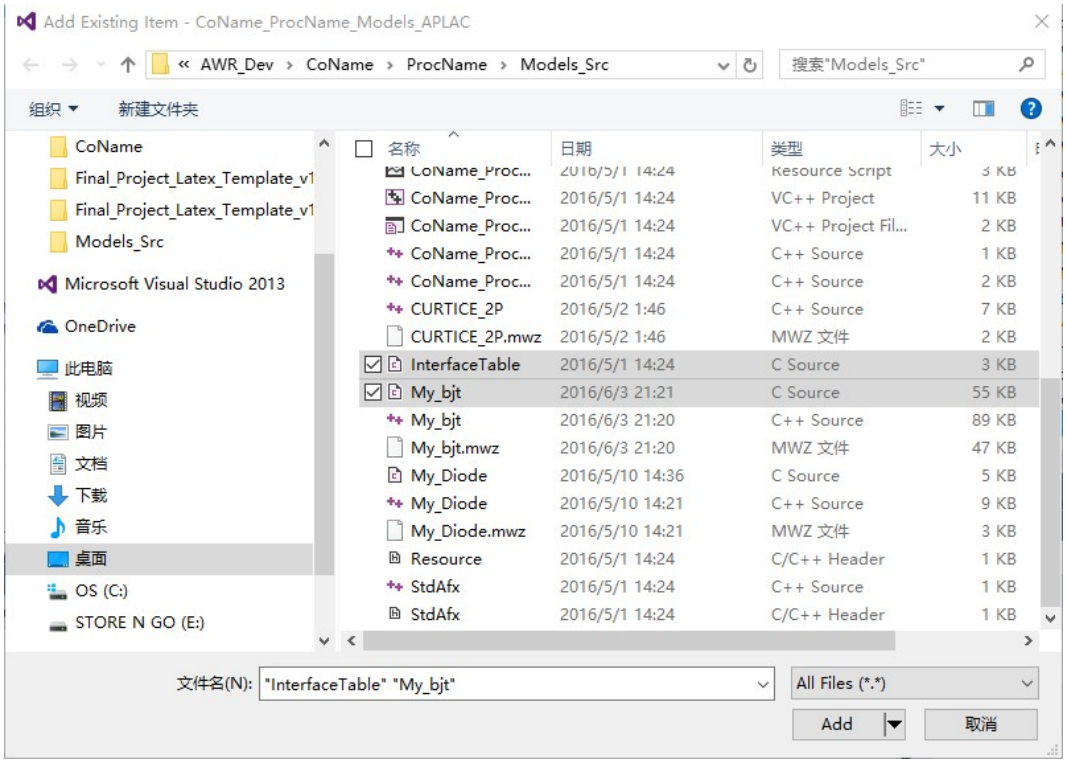

AWR Electrical Model Wizard (12.0.7579)
Primary Model Definition
Edit Model Definition...
Import from Schematic...
Controlled-Source/Derived-Model Definitions
Select the number of nonlinear controlled sources and derived models that you want to define in this model file (if any).
1
Multi
Edit
Load Model Definition (.mwz)...
Save .mwz
Save As...
Import Verilog-A or ADS Netlist...
Export Verilog-A Model Skeleton...
Generate Source Code
Generate APLAC Source Code
Generate debug code
Save State
Quit

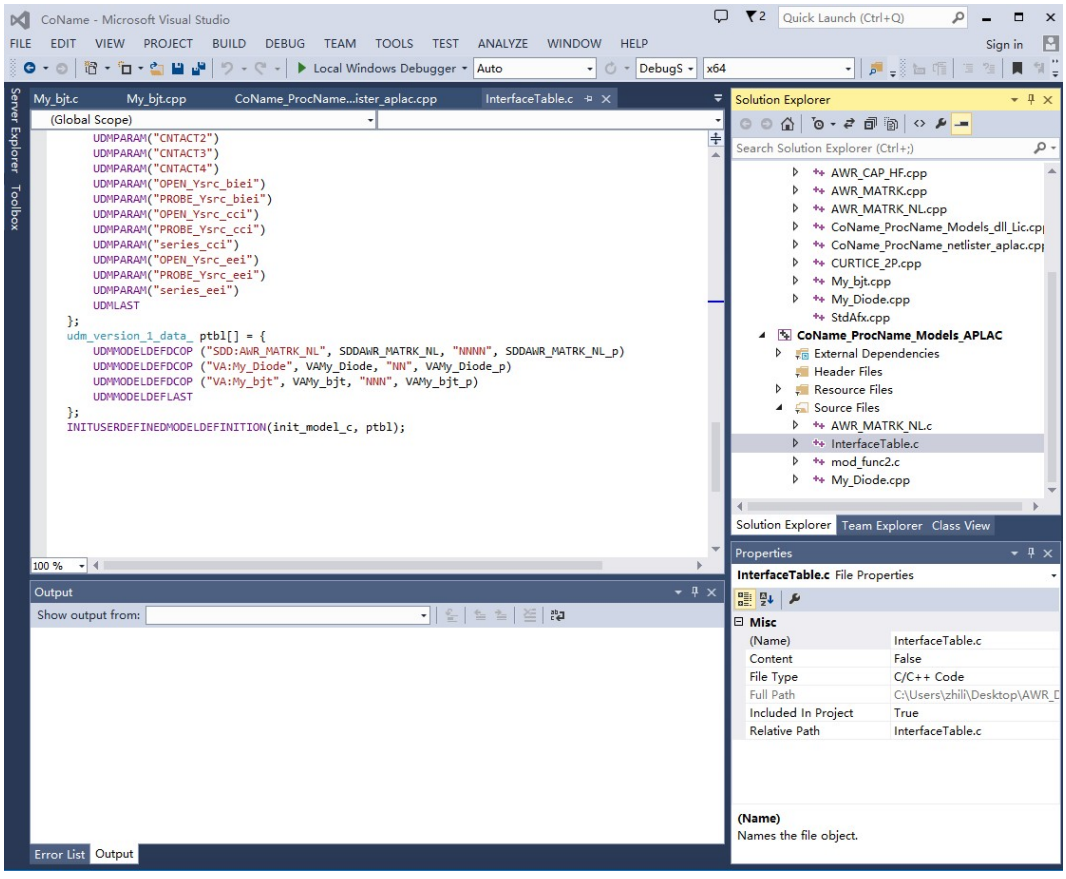

Open Project
此电脑 › 桌面 › AWR_Dev › CoName
搜索"CoName"
组织
新建文件夹
快速访问
Microsoft Visual Studio 2013
OneDrive
此电脑
STORE N GO (E:)
网络
名称
修改日期
类型
大小
ProcName
2016/5/1 14:49
文件夹
CoName
2016/5/1 14:24
Microsoft Visual...
Options:
Add to Solution
Close Solution
文件名(N):
All Project Files (*.sln;*.dsw;*.
打开(O)
取消


CURTICE_2P.cpp
AWR_CAP_HF.cpp
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CoName' (3 projects)
ProcName
CoName_ProcName_Cells
CoName_ProcName_Models
External Dependencies
Header Files
Resource Files
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Filter
Class...
Resource...
Add
Class Wizard... Ctrl+Shift+X
Scope to This
New Solution Explorer View
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename
Properties Alt+Enter
P_HF.cpp
ATRK.cpp
ATRK_NL.cpp
_ProcName_Models_dll_Lic.cpp
_ProcName_netlister_aplac.cpp
_2P.cpp
le.cpp
pp
Name_Models_APLAC
Solution Explorer
Team Explorer
Class View
Properties
Source Files Filter Properties
Advanced
Parse Files True
SCC Files True
General
(Name) Source Files
Filter cpp;c;cxx;rc;def;r;odl;idl;hpj;ba
Unique Identifier {05A0D5AE-B6EC-4070-9FB8-E
Search Error List
File
Line
Column
Project


Add Existing Item - CoName_ProcName_Models
« AWR_Dev › CoName › ProcName › Models_Src
搜索"Models_Src"
组织
新建文件夹
CoName
Final_Project_Latex_Template_v1
Final_Project_Latex_Template_v1
Models_Src
Microsoft Visual Studio 2013
OneDrive
此电脑
视频
图片
文档
下载
音乐
桌面
OS (C:)
STORE N GO (E:)
名称
日期
类型
大小
CoName_Proc... 2016/5/1 14:24 VC++ Project 11 KB
CoName_Proc... 2016/5/1 14:24 VC++ Project Fil... 2 KB
CoName_Proc... 2016/5/1 14:24 C++ Source 1 KB
CoName_Proc... 2016/5/1 14:24 C++ Source 2 KB
CURTICE_2P 2016/5/2 1:46 C++ Source 7 KB
CURTICE_2P.mwz 2016/5/2 1:46 MWZ 文件 2 KB
InterfaceTable 2016/5/1 14:24 C Source 3 KB
My_bjt 2016/6/3 21:21 C Source 55 KB
My_bjt 2016/6/3 21:20 C++ Source 89 KB
My_bjt.mwz 2016/6/3 21:20 MWZ 文件 47 KB
My_Diode 2016/5/10 14:36 C Source 5 KB
My_Diode 2016/5/10 14:21 C++ Source 9 KB
My_Diode.mwz 2016/5/10 14:21 MWZ 文件 3 KB
Resource 2016/5/1 14:24 C/C++ Header 1 KB
StdAfx 2016/5/1 14:24 C++ Source 1 KB
StdAfx 2016/5/1 14:24 C/C++ Header 1 KB
文件名(N): My_bjt
All Files (*.*)
Add
取消


Output
Show output from: Build
1> StdAfx.cpp
1> My_Diode.cpp
1> CURTICE_2P.cpp
1> AWR_MATRK_NL.cpp
1> AWR_MATRK.cpp
1> AWR_CAP_HF.cpp
1> Generating Code...
1> CoName_ProcName_Models.vcxproj -> C:\Users\zhili\Desktop\AWR_Dev
1> C:\Users\zhili\Desktop\AWR_Dev\SDKv12\scripts\SetElementSubstrat
1> 复制了 1 个文件
1> C:\Users\zhili\Desktop\AWR_Dev\SDKv12\scripts\UpdateGlobalDefs.b
1> 复制了 1 个文件
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========


Add Existing Item - CoName_ProcName_Models_APLAC
« AWR_Dev › CoName › ProcName › Models_Src
搜索"Models_Src"
组织
新建文件夹
CoName
Final_Project_Latex_Template_v1
Final_Project_Latex_Template_v1
Models_Src
Microsoft Visual Studio 2013
OneDrive
此电脑
视频
图片
文档
下载
音乐
桌面
OS (C:)
STORE N GO (E:)
名称
日期
类型
大小
CoName_Proc... 2016/5/1 14:24 Resource Script 3 KB
CoName_Proc... 2016/5/1 14:24 VC++ Project 11 KB
CoName_Proc... 2016/5/1 14:24 VC++ Project Fil... 2 KB
CoName_Proc... 2016/5/1 14:24 C++ Source 1 KB
CoName_Proc... 2016/5/1 14:24 C++ Source 2 KB
CURTICE_2P 2016/5/2 1:46 C++ Source 7 KB
CURTICE_2P.mwz 2016/5/2 1:46 MWZ 文件 2 KB
InterfaceTable 2016/5/1 14:24 C Source 3 KB
My_bjt 2016/6/3 21:21 C Source 55 KB
My_bjt 2016/6/3 21:20 C++ Source 89 KB
My_bjt.mwz 2016/6/3 21:20 MWZ 文件 47 KB
My_Diode 2016/5/10 14:36 C Source 5 KB
My_Diode 2016/5/10 14:21 C++ Source 9 KB
My_Diode.mwz 2016/5/10 14:21 MWZ 文件 3 KB
Resource 2016/5/1 14:24 C/C++ Header 1 KB
StdAfx 2016/5/1 14:24 C++ Source 1 KB
StdAfx 2016/5/1 14:24 C/C++ Header 1 KB
文件名(N): "InterfaceTable" "My_bjt"
All Files (*.*)
Add
取消


CoName - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
DebugS
x64
My_bjt.c
My_bjt.cpp
CoName_ProcName...ister_aplac.cpp
InterfaceTable.c
(Global Scope)
UDMPARAM("CNTACT2")
UDMPARAM("CNTACT3")
UDMPARAM("CNTACT4")
UDMPARAM("OPEN_Ysrc_biei")
UDMPARAM("PROBE_Ysrc_biei")
UDMPARAM("OPEN_Ysrc_cci")
UDMPARAM("PROBE_Ysrc_cci")
UDMPARAM("series_cci")
UDMPARAM("OPEN_Ysrc_eei")
UDMPARAM("PROBE_Ysrc_eei")
UDMPARAM("series_eei")
UDMLAST
};
udm_version_1_data_ ptbl[] = {
UDMMODELDEFDCOP ("SDD:AWR_MATRK_NL", SDDAWR_MATRK_NL, "NNNN", SDDAWR_MATRK_NL_p)
UDMMODELDEFDCOP ("VA:My_Diode", VAMy_Diode, "NN", VAMy_Diode_p)
UDMMODELDEFDCOP ("VA:My_bjt", VAMy_bjt, "NNN", VAMy_bjt_p)
UDMMODELDEFLAST
};
INITUSERDEFINEDMODELDEFINITION(init_model_c, ptbl);
100 %
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
AWR_CAP_HF.cpp
AWR_MATRK.cpp
AWR_MATRK_NL.cpp
CoName_ProcName_Models_dll_Lic.cp
CoName_ProcName_netlister_aplac.cp
CURTICE_2P.cpp
My_bjt.cpp
My_Diode.cpp
StdAfx.cpp
CoName_ProcName_Models_APLAC
External Dependencies
Header Files
Resource Files
Source Files
AWR_MATRK_NL.c
InterfaceTable.c
mod_func2.c
My_Diode.cpp
Solution Explorer
Team Explorer
Class View
Properties
InterfaceTable.c File Properties
Misc
(Name) InterfaceTable.c
Content False
File Type C/C++ Code
Full Path C:\Users\zhili\Desktop\AWR_D
Included In Project True
Relative Path InterfaceTable.c
(Name)
Names the file object.

```
AWR_GET_TRANSFORMER(MWO_TO_APLAC)
AWR_ADD_OBJ_TO_TRANSFORMER(basic_APLAC,"VA:My_Diode",MWO_TO_APLAC)
//Add additional models between the _OBJ_BEGIN(1)/_OBJ_END(1)
AWR_ADD_TRANSFORM_OBJ_BEGIN(1)
AWR_ADD_EXISTING_OBJ(basic_APLAC, "VA:My_bjt", MWO_TO_APLAC)
//AWR_ADD_EXISTING_OBJ(basic_APLAC,"LIB_My_Resistance",MWO_TO_APLAC)
AWR_ADD_TRANSFORM_OBJ_END(1)
```

Chapter 5

Results and Discussion

In pervious chapters, we have discussed the theroy about FBH HBT models and how to create models through NI AWRDE and visual studio. In this chapter, we will provide our simulation result from our model, which generated by using NI AWRDE and visual studio. We are going to discuss each simulation result of our models and Compare the simulation result to theorical solution . The error of simulation result will be investiaged. In the following section, we are going to explore the characteristic of our model in NI AWRDE and explain the Verilog-A code given in Appendix in details.

5.1 Resistor Model

5.1.1 A Linear Resistor in Verilog-A

```
module My_Resistance(p,n);
inout p,n;
electrical p,n;
parameter real resistance=50;
analog
V(p,n) <+ resistance*I(p,n);
endmodule
```

Between line 1 and line 7 declares the module block, within is how the model behaviour is being defined. The first line defines the name of our model, which is named as My resistance. There are two ports, which called “p” and “n”. “p” is input port and “n” is output port. The nature of those ports declared in the electrical discipline. In our case refer to the flow enter the pin is a current and the potential of each pin is voltage. The fourth line declares a parameter which is called as resistance and set is as a default value of 50.0. In our case, the parameter is defined as real type. The keyword analog in the fifth line define the analog block. The line 7 declare the relationship between voltage

and current. V(p,n) represents the potential difference measured between ports p and n. I(p,n) represents the current flowing from port p to n. The value of current is multiplied by the value of the parameter resistance. The "<+ " is named as the contribution operator, which indicates an equation that must be satisfied by the simulator. It indicates that the potential difference on branch between port p and n must equal to the current through this branch multiplied by resistance.

5.1.2 The simulation result of reisstor model in NI AWRDE

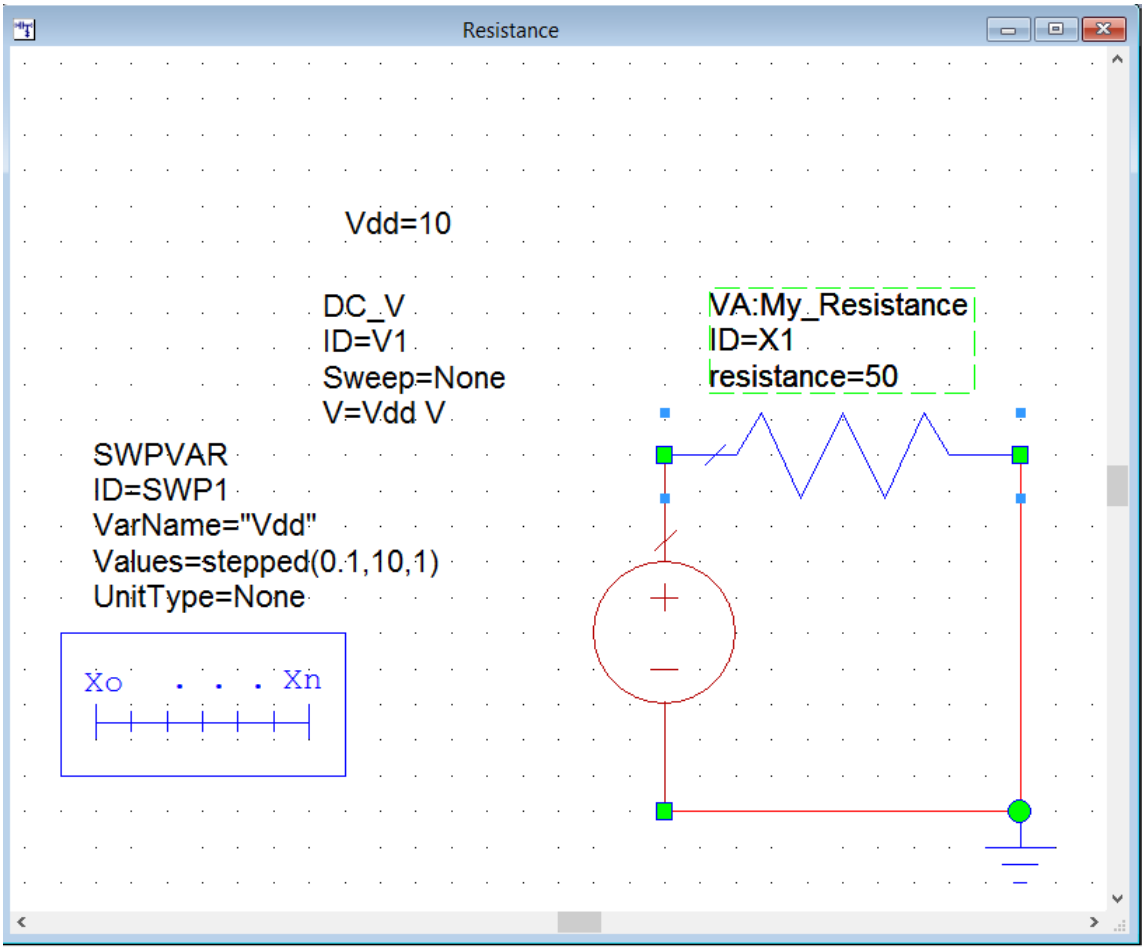


Figure 5.1: Circuit of Our Resistance

In Figure 5.1 shows the circuit of our resistance. Stepped (start, stop, step) uses to generate a vector of real values from start to stop

In our case, the AC voltage start at 0.1V and keep increase by 1V until reach 10V.

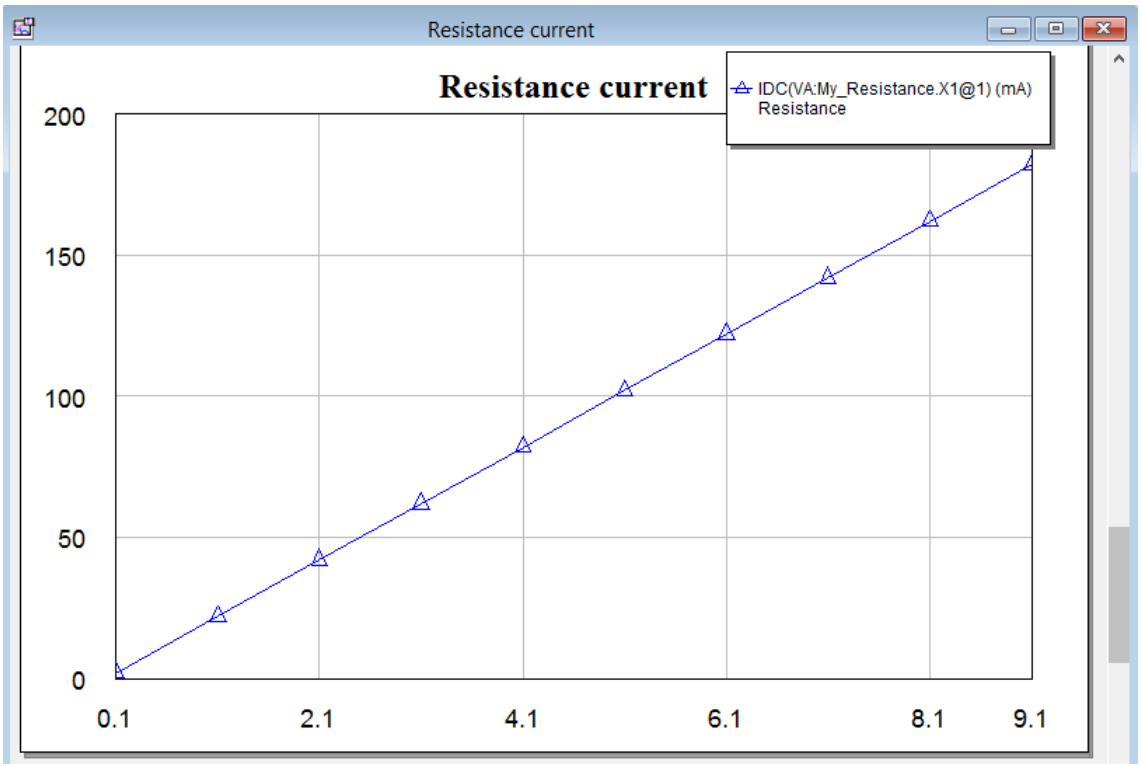


Figure 5.2: I-V characteristic Curves of our resistor in both Harmonic Balance simulator and APLAC DC simulator

Figure 5.2 shows the values of current through the resistance by AC voltage across it. The simulation result shows a good agreement between theory and simulation. Linear current obeys Ohms Law as the voltage across the resistor is linearly proportional to the current through it.

5.2 Diode Model

5.2.1 A Nonlinear Diode in Verilog-A

Verilog-A is well-suited for describing nonlinear behaviour. The Verilog-A code of PN junction diode is defined as below:

‘include “disciplines.h”
‘include “constants.h”
module My_Diode(anode,cathode);
inout anode,cathode;
electrical anode,cathode;
parameter real is=1e-14 from (0:inf);
parameter real n=1 from (0:inf);

parameter real tt=1 from (0:inf);
real Vd, Id, Qd;
analog
begin
V_d = V(anode,cathode);
$I_d = i_s e^{qV_d/nkT} - 1$
$Q_d = tt * I_d$;
I(anode,cathode) <+ I_d + ddt(Q_d);
end
endmodule

The first line defines the name of our model, which is named as My Diode. There are two ports, which called "anode" and "cathode". "anode" is input port and "cathode" is output port. The nature of those ports declared in the electrical discipline. It means pass through it is current and the potential between these two pins is voltage. Between line 4 and line 6 declares three parameters, which are i_s, n, and tt. i_s stands for Saturation current, and assigns it a default value of 1e-14. N refers to ideality factor, which is between 1 and 2. In our case, we set it as 1. Tt stands for transit time, assign it with a value of 1. Those parameters are declared in real type. V_d=V(anode,cathode) is declares V_d is equal to the voltage across the diode connected between nodes anode and cathode of the module. I_d is net current flowing through the diode. We know $I_d = i_s e^{qV_d/nkT} - 1$ for actual diode. Where k is Bolzmanns constant , q is absolute value of electron charge, T is absolute temperature, which is also called as room temperature and equal to 300K. $V_T = kTq = 0.026V$ is thermal voltage. Q_d stand for charge and equal to transit time multiplied by current through the diode.

I(anode,cathode) <+ I_d + ddt(Q_d);

It is same behaviour to following two statements.

I(anode,cathode) <+ I_d
I(anode,cathode) <+ ddt(Q_d);

The "<+" is called the contribution operator and in our case contributes the value of I_d or ddt(Q_d) as the voltage from anode to cathode.

In Figure 5.3 shows the circuit of our diode. Stepped (start, stop, step) uses to generate a vector of real values from start to stop

In our case, the AC voltage start at 0.5V and keep increase by 0.01V until reach 1.2V. The symbol I_s is Saturation current. The symbol n is emission efficient, between 1 and 2. The symbol tt is transit time, which is equal to 1s for our situation.

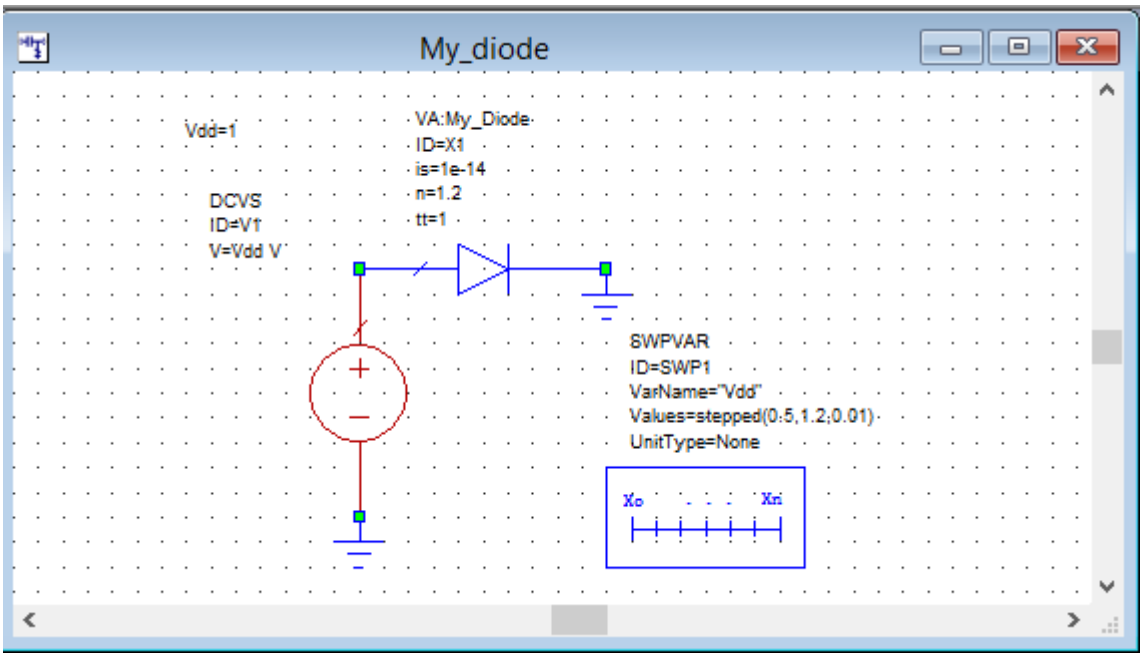


Figure 5.3: Circuit of Our Diode

In Figure 5.4, the diode is forward biased, anode positive with respect to the cathode, a positive or forward current passes through the diode. Starting at zero, the forward current and voltage are very small. Until the forward voltage exceeds the diodes P-N junctions internal barrier voltage, which is about 0.7 volts, the forward current start to increases rapidly for an extremely small increase in voltage create a non-linear curve. The simulation results shows a good agreement between theoretical characteristic of diode and simulation.

5.3 Transistor Model

5.3.1 A Nonlinear Transistor in Verilog-A

The electrical behaviour of the devices transistors required to be described accurately in Verilog-A.

In Figure 5.5 shows the circuit of our transistor., we use IVCURVEI to apply the voltage in collector and the current in base to construct our transistor circuit.

All previously components (resistor, diode) only have two terminals, therefore be characterized by the simple relationship between the current flow through and the voltage across the two terminals. Differently, a transistor has three-terminals, which are emitter, base and collector.

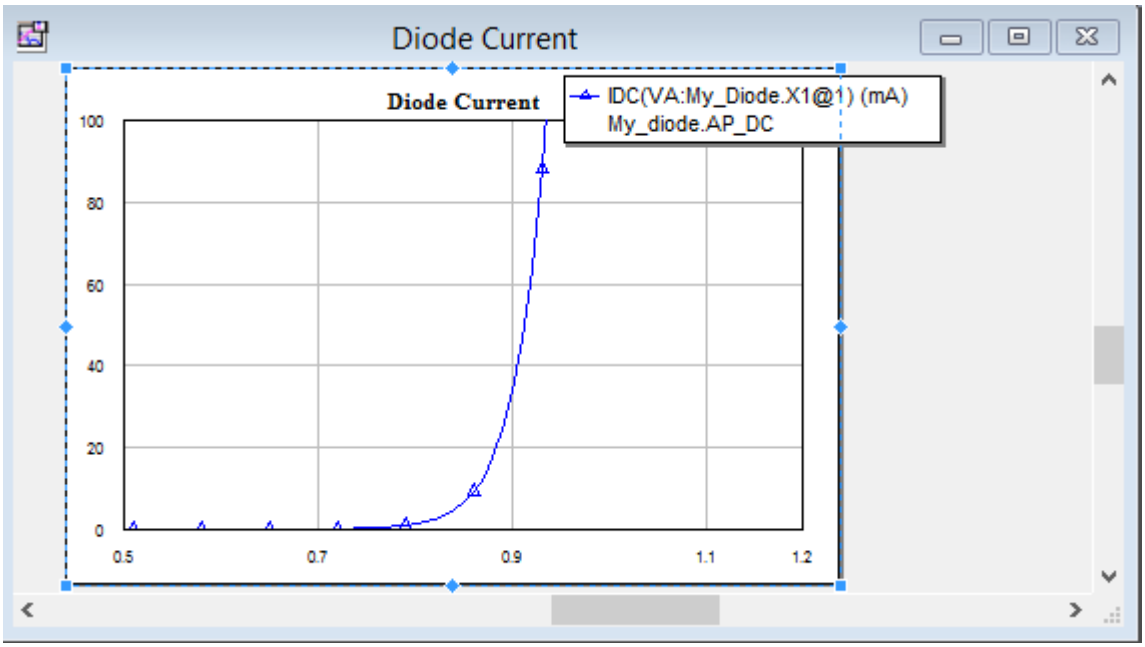


Figure 5.4: I-V Characteristic Curve of Our diode in both Harmonic Balance simulator and APLAC DC simulator

```
module My_bjt(c,b,e);
inout c,b,e;
electrical ci,b,ei, bi, c,e;
```

In Verilog-A code, three terminals of transistor present in My_bjt(c,b,e). NI AWRDE the three terminals of transistor appear in number. 1 corresponding to base, 2 corresponding to collector, 3 corresponding to emitter. The error of simulation may due to three thermals of transistor not corresponding to correct number in NI AWRDE, which how NI AWRDE presented terminals of transistor.

Compare to resistor model and diode model. Transistor model requires to use much more parameters. All parameters must determine in accuracy way. Obviously the code will be much complicated to previous. And also required to add the internal noises in transistor model. ci refers to internal collector current, ei stands for internal emitter current. bi stands for internal base current. It allows to access the internal current of transistor.

5.4 Chapter Summary

In this chapter, we import verilog-A code into NI AWRDE. Edit model definition. Generate source code and Aplac source code in order to create model. The simulation result for resistor and diode, which obtained from NI AWRDE is quite good. It has good agreement with theoretical solutions. There are some issues with transister model. It may because

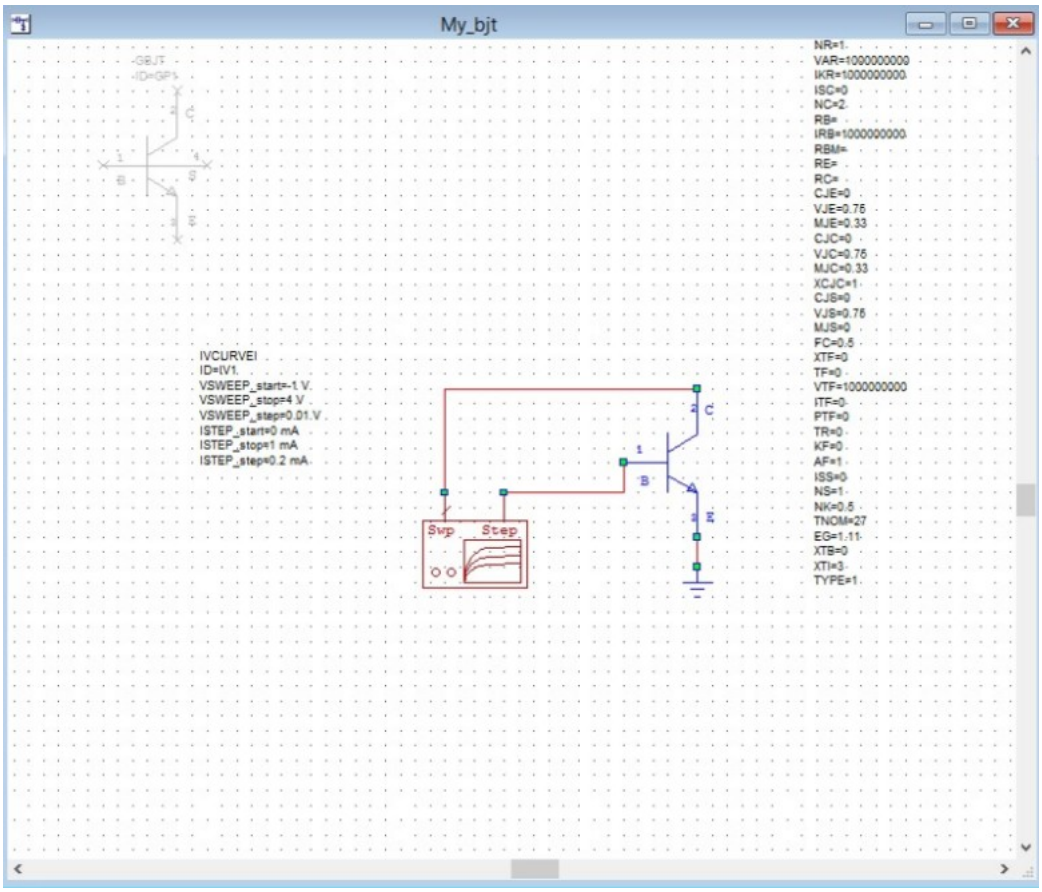


Figure 5.5: Circuit of Our Transistor

three thermal declared in Verilog-A is different to three thermal declared in NI AWRDE. The way to solve it, we can try voltage controlled current source and set schematic symbol of voltage controlled current source to default. It will help use to determine how NI AWRDE present thermials of volatage controlled current source. Unfortunately we do not have enough to try it. Once we figure out that, we can try our transistor model again. It may work.

Finally, we have learned how to use NI AWRDE and visual studio to create model in PDK.

Chapter 6

Conclusions and Future Work

6.1 Future Work

The previous chapters presents activities perfromed involve with this project. This chapter will briefly discuss some possible enhancements that can be made base on the current model. The thesis focused on how implementing model wizard in NI AWRDE and using Microsoft visual studio to create model. The follwing section will presented some suggestions can be explored in the future projects.

6.1.1 Add Noise to the Verilog-A Resistor

Verilog-A provides a couple of ways to add noise, contain frequency-dependent, frequency-independent and piecewise linear frequency -dependent noise. The thermal noise is defined as $4 * K * T/R$. T is termperature of the circuit, which will be changed due to outside of the model. White_noise() operator is used for contribute noise.

analog V(p,n) <+ R * I(p,n);
change to
V(p,n) <+ R * I(p,n) + white_noise(4 * 'P_K * *temperature*/*R*,"thermal");

By change this part of our previous code. It will help use to achieve resistor with noise. We can investiage I-V characteristic of resistor. To explore how noise effect in I-V characteristic of resistor.

6.1.2 Add an Internal Node to the diode

The prominent diode require to contain the drop in the junction voltage because of the series resistance. The way to implementing the series resistance will require to ad an internal nide. An internal node is also called as a net, which can be add by declaring the node as electrical.

electrical anode, cathode, internal;

It allow designer to explore the internal port of diode.

6.1.3 Add limiting to the diode for get better convergence

In our diode code use the exponential function, which result in large swings in currents for small modifications in voltage. exp() can be replaced by a special operator(limexp()), which enable the simulator algorithms to restrict the exponential in simulator-specific ways.

6.1.4 The Verilog-A of transistor

// voltage controlled current source module vcvs (pout, nout, pin, nin);

electrical pout, nout, pin, nin;
parameter real gain=1;

analog V(pout,nout) <+ gain*I(pin,nin);
endmodule

The terminal port of transistor present in verilog-A by name, however the terminal port of transistor present in NI AWRDE in number. First, we can try Voltage controlled current source and set the schematic symbol of Voltage controlled current source to be default, to explore how port name present in verilog-A corresponding to port number in NI AWRDE. Once we know that, it will help us to figure out how port name present in verilog-A of transistor corrsponding to the port number of transistor in NI AWRDE. It may result our simulation work fine. To investigate how the internal noise of transistor effect the characterstic of transistor, which include self-heating effect.

Chapter 7

Abbreviations

Analog HDLs	Analog Hardware Description languages
RFICs	Radio Frequency Integrated Circuits
HBT	Heterojunction Bipolar Transistor
GP model	Gummel-Poon model
FETs	The field-effect transistors
VBIC model	Vertical Bipolar Inter-Company
PDK	Process Design Kit
SDK	Software development Kit
FBH	Ferdinand-Braun-Institut fr Hchstfrequenztechnik
GaAs	Gallium arsenide
AlGaAs	Aluminium gallium arsenide
GaInP	Gallium indium phosphide

Appendix A

Verilog-A Code

A.1 Overview

All the Verilog-A codes written during the project will be provided here for the reference of readers.

A.2 Resistor in Verilog-A

```
module My_Resistance(p,n);
 inout p,n;
 electrical p,n;
 parameter real resistance=50;

 analog
    V(p,n) <+ resistance*I(p,n);
endmodule
```

A.3 Diode in Verilog-A

A.4 Transistor in Verilog-A

```
`include "disciplines.h"
`include "constants.h"

module My_Diode(anode,cathode);
 inout anode,cathode;
 electrical anode,cathode;

 parameter real is=1e-14 from (0:inf);
 parameter real n=1 from (0:inf);
 parameter real tt=1 from (0:inf);

 real Vd, Id, Qd;
 analog
   begin
     Vd = V(anode,cathode);
                  Id = is*(exp(Vd/(n*0.026)) - 1.0);
                  Qd=tt*Id;
                  (* name="Id" *)  I(anode,cathode) <+ Id + ddt(Qd);
   end

endmodule
module My_Resistance(p,n);
 inout p,n;
 electrical p,n;
 parameter real resistance=50;

 analog
    V(p,n) <+ resistance*I(p,n);
endmodule
```

```
`include "disciplines.h"
`include "constants.h"
`define PI_SQ_24    2.43170840742
`define PI_SQ_144 14.5902504445
`define NOT_GIVEN 999
`define DEFAULT_TNOM 27
`define SPICE_GMIN 1e-12
`define SPICE_GMAX 1e+12

module My_bjt(c,b,e);
 inout c,b,e;
 electrical ci,b,ei, bi, c,e;

        parameter real AREA = 1 from (0.0:inf]; // Area scaling factor
        parameter real UA1 = `NOT_GIVEN;       // Device temperature override [C]
        parameter real IS = 1e-16 from (0.0:inf]; // Transport saturation current [A]
        parameter real BF = 100 from (0.0:inf]; // Ideal max fwd beta
        parameter real NF = 1.0 from (0.0:inf]; // Forward current emission coeff
        parameter real VAF = 1e9 from [0.0:inf]; // Forward early voltage [v]
        parameter real IKF = 1e9 from (0.0:inf]; // Forward beta high-current roll-off [A]
        parameter real ISE = 0 from [0.0:inf]; // B-E leakage current [A]
        parameter real NE = 1.5 from (0.0:inf]; // B-E p-n leakage emission coeff
        parameter real BR = 1.0 from (0.0:inf]; // Ideal max rev beta
        parameter real NR = 1.0 from (0.0:inf]; // Reverse current emission coeff
        parameter real VAR = 1e9 from [0.0:inf]; // Forward Early voltage [v]
        parameter real IKR = 1e9 from (0.0:inf]; // Reverse beta high-current roll-off [A]
        parameter real ISC = 0 from [0.0:inf]; // B-C leakage current [A]
        parameter real NC = 2.0 from (0.0:inf]; // B-C p-n leakage emission coeff
        parameter real RB = 1.0/`SPICE_GMAX from [0.0:inf]; // Zero-bias maximum base resistance [Ohm]
        parameter real IRB = 1e9 from [0.0:inf]; // Current where Rb falls halfway [A]
        parameter real RBM = RB from [0.0:inf]; // Zero-bias minimum base res (def=Rb) [Ohm]
        parameter real RE = 1.0/`SPICE_GMAX from [0.0:1.0/`SPICE_GMIN]; // Emitter res[Ohm]
        parameter real RC = 1.0/`SPICE_GMAX from [0.0:1.0/`SPICE_GMIN]; // Collector res[Ohm]
        parameter real CJE = 0.0 from [0.0:inf]; // B-E zero-bias p-n cap [F]
        parameter real VJE = 0.75 from [0.0:inf]; // B-E built-in potential [v]
        parameter real MJE = 0.33 from (0.0:inf]; // B-E p-n grading factor
        parameter real CJC = 0.0 from [0.0:inf]; // B-C zero-bias p-n cap [F]
        parameter real VJC = 0.75 from [0.0:inf]; // B-C built-in potential [v]
        parameter real MJC = 0.33 from (0.0:inf]; // B-C p-n grading factor
        parameter real XCJC = 1.0 from [0.0:inf]; // Fraction of Cjc connected to Rb
        parameter real CJS = 0.0 from [0.0:inf]; // C-S zero-bias p-n cap [F]
        parameter real VJS = 0.75 from [0.0:1e9]; // Substrate built-in potential [v]
        parameter real MJS = 0.0 from [0.0:inf]; // Substrate p-n grading factor
        parameter real FC = 0.5 from (0.0:inf]; // Forward bias depletion cap coeff.
        parameter real XTF = 0.0 from [0.0:inf]; // Transit time bias coeff
        parameter real TF = 0.0 from [0.0:inf]; // Ideal forward transit time [s]
        parameter real VTF = 1e9 from [0.0:inf]; // Transit time dependency on Vbe [v]
        parameter real ITF = 0 from [0.0:inf]; // Transit time dependence on Ic [A]
        parameter real PTF = 0.0 from [0.0:inf]; // Excess phase at 1/(2*pi*TF), degrees
        parameter real TR = 0.0 from [0.0:inf]; // Ideal reverse transit time [s]
```

```
parameter real KF = 0 from [0.0:inf]; // Flicker noise coeff
parameter real AF = 1 from (0.0:inf]; // Flicker noise exponent
parameter real ISS = 0 from [0.0:inf]; // Unused: Substrate P-N staturation current [A]
parameter real NS = 1.0 from (0.0:inf]; // Unused: Substrate p-n emission coeff
parameter real NK = 0.5 from (0.0:inf]; // High current roll-off coeff
parameter real TNOM = `DEFAULT_TNOM from (-`P_CELSIUS0:inf); // Param meas temp [C]
parameter real EG = 1.11 from [0.0:inf]; // Bandgap voltage [eV]
parameter real XTB = 0.0 from [0.0:inf]; // Fwd and rev beta temp coeff
parameter real XTI = 3.0 from [0.0:inf]; // IS temperature effect exponent

parameter integer TYPE = 1 from [-1:1] exclude 0;

analog function real limexpLocal;
        input V;
        real V;
        begin
                if (V < 40.0)
                        limexpLocal = exp(V);
                else
                        limexpLocal = exp(40.0)*(V-40.0)+exp(40.0);
        end
endfunction // limexpLocal

(*desc="transConductance"*) real gm;
(*desc="CjeOP"*) real CjeOP;
(*desc="collectorCurrent"*) real collectorCurrent;
(*desc="baseCurrent"*)real i_rb;

real Vbe, Vbi, Vt, Vbcx, Vbci,Vbc, Vcs;
//real Ib, Ibe1, Ibe2, Ibc1, Ibc2, i_rb, Ic, rb;
real Ib, Ibe1, Ibe2, Ibc1, Ibc2, Ic, rb;
real vaf, var, vtf, Vtn;
real Kq1, Kq2, Kqb;
real Qje, Qbe, Qbc, Qjcx, Qjs;
real x, tanx;
real tff, t0, t1, t2, t3;
real T, EG_T, IS_T, ISE_T, ISC_T, BF_T, BR_T;
real T_NOM;
real Cjc_1, Cjc_2, Cjc0, Qjc0, Qjc, fcpe, fcpc;
real Cje_1, Cje_2, Cje0, Qje0, vdiff;
real Cjs_1, Cjs0, Qjs0;
real T0, T1, Jrb, Ib_Jrb;
real ratioT;

analog function real IS_of_T;
input IS, ratioT, EG, Vth, XTI;
real  IS, ratioT, EG, Vth, XTI;
begin
IS_of_T = IS * exp((ratioT - 1) * EG / Vth)
           * pow(ratioT, XTI);
end
```

```
        endfunction // IS_T

        analog function real I_of_T;
    input IS, ratioT, EG, N, Vth, XTI, XTB;
    real  IS, ratioT, EG, N, Vth, XTI, XTB;
    begin
        I_of_T = IS / pow(ratioT, XTB) * exp( ( ratioT - 1)
                  * EG / (N * Vth)) * pow(ratioT, XTI/N);
    end
        endfunction // I_of_T

        analog function real B_of_T;
    input B, ratioT, XTB;
    real B, ratioT, XTB;
    begin
        B_of_T = B * pow(ratioT, XTB);
    end
        endfunction // B_of_T

        analog function real EG_of_T;
    input T;
    real T;
    begin
        EG_of_T = 1.16- 0.000702 * T * T / (T + 1108);
    end
        endfunction // EG_of_T


analog begin
       @(initial_step) begin
                if (UA1 == `NOT_GIVEN)
                        T = $temperature;
                else
                        T = UA1 + `P_CELSIUS0;

                Vt = $vt(T);
                T_NOM = TNOM + `P_CELSIUS0;

                ratioT = T / T_NOM;

                EG_T = EG_of_T(T);
                IS_T = IS_of_T(IS, ratioT, EG_T, Vt, XTI);
                ISE_T = I_of_T(ISE, ratioT, EG_T, NE, Vt, XTI, XTB);
                ISC_T = I_of_T(ISC, ratioT, EG_T, NC, Vt, XTI, XTB);
                BF_T = B_of_T(BF, ratioT, XTB);
                BR_T = B_of_T(BR, ratioT, XTB);
       end

       Vbe = TYPE * V(bi, ei);
    Vbci = TYPE * V(bi, ci);
       Vbcx = TYPE * V(b, ci);
```

```
    Vbi = TYPE * V(b, bi);



    Vtn = Vt * NF;
vtf = VTF;
var = VAR;
vaf = VAF;


Ibe1 = IS_T  * (limexpLocal(Vbe / (NF * Vt)) - 1);
Ibe2 = ISE_T * (limexpLocal(Vbe / (NE * Vt)) - 1);

Ibc1 = IS_T  * (limexpLocal(Vbci / (NR * Vt)) - 1);
Ibc2 = ISC_T * (limexpLocal(Vbci / (NC * Vt)) - 1);

Kq1 = 1 / (1 - Vbci / vaf - Vbe / var);

Kq2 = Ibe1 / IKF + Ibc1 / IKR;
Kqb = Kq1 * (1 + pow(1 + 4 * Kq2, NK)) / 2;

if (IRB == 1e9 || IRB == 0) begin
    rb = (RBM + (RB - RBM) / Kqb) / AREA;
    end
    else begin
    Ib = AREA * (Ibe1/BF_T + Ibe2 + Ibc1/BR_T + Ibc2);
    if (Ib >= 0) begin
        Jrb = AREA * IRB;
        Ib_Jrb = Ib / Jrb;
        Ib_Jrb = max(Ib_Jrb, 1e-9);
        T0 = sqrt(1.0 + `PI_SQ_144 * Ib_Jrb);
        T1 =  `PI_SQ_24 * sqrt(Ib_Jrb);
        x = (-1.0 + T0) / T1;
        tanx = tan(x);
        rb = (RBM + 3.0 * (RB - RBM) * (tanx - x) /
            (x * tanx * tanx)) / AREA;
        end
        else begin
        rb = RB / AREA;
    end
end
    if (rb!=0.0)
    i_rb = Vbi / rb;

// Base-emitter diffusion
if (Ibe1 + ITF != 0.0) begin
    t0 = Ibe1 / (Ibe1 + ITF);
    tff = TF * (1 + XTF * t0 * t0 * limexpLocal(Vbci / (1.44 * VTF)));
end
else begin
    tff = 0.0;
```

```
    end
    t3 = 1 - FC;

// Calc base-emitter junction capacitance
Cje0 = CJE;
if (MJE == 1.0) begin
    Cje_1 = Cje0 / t3;
    Cje_2 = 1.0 / (VJE * t3);
    Qje0 = Cje0 * VJE* (0.5 + ln(VJE * t3));
end
else begin
    Cje_1 = Cje0/pow(t3, MJE);
    Cje_2 = MJE / (VJE * t3);
    Qje0  = Cje_1 / (2.0 * Cje_2) * (1.0 + MJE) / (1.0 - MJE);
end

fcpe = FC * VJE;
if (Vbe <= fcpe)  begin
    vdiff = VJE - Vbe;
    if (MJE == 1.0)
        Qje = -Cje0 * VJE * ln(vdiff);
    else if (MJE == 0.5)
        Qje = -2.0 * vdiff * Cje0 / sqrt(vdiff / VJE);
    else
        Qje = -Cje0 * pow(vdiff / VJE, -MJE) * vdiff / (1.0 - MJE);

    Qje = Qje + Qje0;
end
else begin
    t1 = 1.0 + Cje_2 * (Vbe - fcpe);
    Qje = Cje_1 * t1 * t1 / (2.0 * Cje_2);
end

CjeOP = ddx(Qje, V(bi,ei));

Qbe = tff * Ibe1 / Kqb + Qje;


 // Intrinsic base-collector junction capacitance
Cjc0 = CJC;
if (MJC == 1.0) begin
    Cjc_1 = Cjc0 / t3;
    Cjc_2 = 1.0 / (VJC * t3);
    Qjc0  = Cjc0 * VJC * (0.5 + ln(VJC * t3));
end else begin
    Cjc_1 = Cjc0 / pow(t3, MJC);
    Cjc_2 = MJC / (VJC * t3);
    Qjc0  = Cjc_1 / (2.0 * Cjc_2) * (1.0 + MJC) / (1.0 - MJC);
end

fcpc = FC * VJC;
```

```
if (Vbci <= fcpc) begin
    vdiff = VJC - Vbci;
    if (MJC == 1.0)
        Qjc = -ln(vdiff) * Cjc0 * VJC;
    else if (MJC == 0.5)
        Qjc = -2.0 * vdiff * Cjc0 / sqrt(vdiff / VJC);
    else
        Qjc = -Cjc0 * pow(vdiff / VJC, -MJC) * vdiff / (1.0 - MJC);
    Qjc = Qjc + Qjc0;
end else begin
    t1 = 1.0 + Cjc_2 * (Vbci - fcpc);
    Qjc = Cjc_1 * t1 * t1 / (2.0 * Cjc_2);
end

Qjc = Qjc * XCJC;


    // Calc external base-collector junction capacitance
if (Vbcx <= fcpc) begin
    vdiff = VJC - Vbcx;
    if (MJC == 1.0)
        Qjcx = -Cjc0 * VJC *ln(vdiff);
    else if (MJC == 0.5)
        Qjcx = -2.0 * vdiff * Cjc0 / sqrt(vdiff / VJC);
    else
        Qjcx = -Cjc0 * pow(vdiff / VJC, -MJC) * vdiff / (1.0 - MJC);
    Qjcx = Qjcx + Qjc0;
end
else begin
    t1 = 1.0 + Cjc_2 * (Vbcx - fcpc);
    Qjcx = Cjc_1 * t1 * t1 / (2.0 * Cjc_2);
end

Qjcx = Qjcx * (1.0 - XCJC) * AREA;

Qbc = TR * Ibc1 + Qjc;

// Branch contributions
I(bi, ei ) <+ TYPE * AREA * (Ibe1/BF_T + ddt(Qbe) +  Ibe2);
I(bi, ci ) <+ TYPE * AREA * (Ibc1/BR_T + ddt(Qbc) + Ibc2);

I(b, ci) <+ TYPE * ddt(Qjcx);

I(ci,  ei )  <+  TYPE * AREA * ((Ibe1 - Ibc1) / Kqb);
    I(b,  bi) <+ V(b, bi)/rb;


    if (RC>0)
            I(c, ci) <+ V(c, ci)/(RC/AREA)+ white_noise(4 * `P_K * T / (RC / AREA), "Rc");
    else
            V(c, ci) <+ 0.0;
```

```
        if (RE>0)
                I(e, ei) <+ V(e, ei)/(RE/AREA)+ white_noise(4 * `P_K * T / (RE / AREA), "Re");
        else
                V(e, ei) <+ 0.0;

        I(b, bi) <+ white_noise(4 * `P_K * T / rb, "Rb");

        Ic = Ibe1 / Kqb - Ibc1 / Kqb - Ibc1 / BR_T - Ibc2;

    gm = ddx(Ic, V(bi,ei));
        collectorCurrent=Ic;

    I(bi,ei) <+ white_noise(2 * `P_Q * i_rb, "shotRb");
    if (AF > 0 && KF > 0)
                I(bi, ei) <+ flicker_noise(KF * pow(AREA * abs(i_rb), AF), 1.0, "flicker");


 end

endmodule
```

Appendix B

Project Plan and Attendance Form

B.1 Overview

This section includes the project plan (Gnatt Chart) represent project specification. And also include the consultation meetings attendance form as need by the department.

B.2 Consultation Meetings Attendance Form

B.3 Project Plan

While allocating a thesis project to a student, the student is expected to have analytical problem solving skills, willing to learn CAD software and Verilog-A computer program and knowledge obtained during learning course of last 3 years in electronic engineering degree. Some of the generic skills would contain using Gantt chart excel, Microsoft words and AWR. In order to get simulation results and write in formal report with correct format and referencing. Projects are allocated to the students on the fundamental of their corresponding engineering knowledge in electronic. The student undertaking this project are required to have those techniques as below:

- Good mathematical analysis knowledge
- Good understanding about the characteristic of transistor models
- Use of simulation software such as AWR and visual studio.
- Good understanding in Verilog-A computer program.

The whole thesis project involves the concept of search, design and implementation and testing. The thesis projects begin the first week of semester 1 2016. The first week will be start off with the fundamental literature review about the concept of Verilog-A modelling of transistor. And also install the software to use for implement Verilog-A modelling of transistor which are AWR and visual studio 2013. The next couple weeks is to get familiar with how implement model wizard in AWR and using Microsoft visual studio to test model. The feedback received from the progress report will be used to improve the final report. It will include all the simulation results from AWR. The followed by seminar presentation where students will need to demonstrate their own project to audience of academic and final year students. It will require to present what we did during our thesis and what achievement from that project.

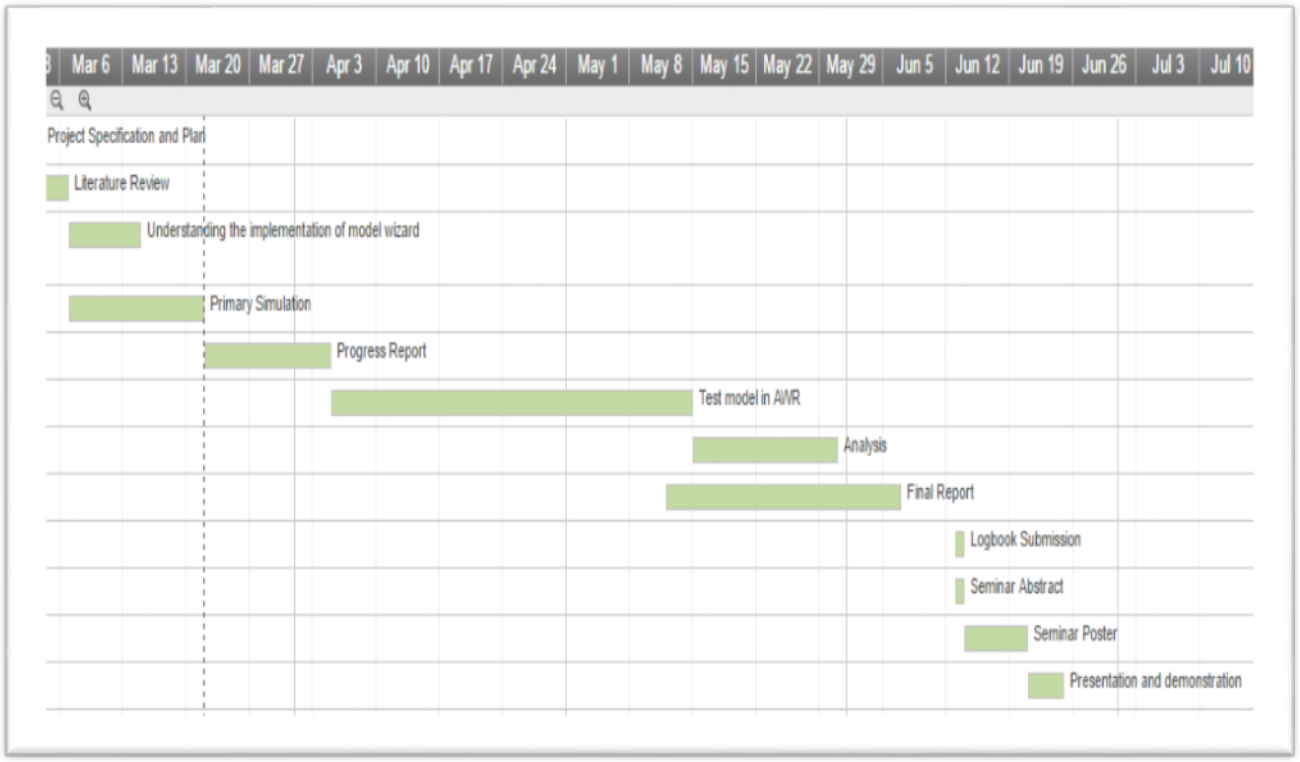
B.4 Project Consultation meetings

The students are required to meet with their academic supervisors every week and to provide a brief outline about the progress reports every two weeks.

B.5 LOGBOOK

The students are required to have a bound logbook recording the work undertaken in the project.

Task Name	Start Date	End Date	Duration
Project Specification and Plan	03/01/16	03/03/16	3d
Literature Review	03/01/16	03/06/16	6d
Understanding the implementation of model wizard	03/07/16	03/14/16	7d
Primary Simulation	03/07/16	03/21/16	15d
Progress Report	03/22/16	04/04/16	15d
Test model in AWR	04/05/16	05/14/16	10d
Analysis	05/15/16	05/30/16	15d
Final Report	05/12/16	06/06/16	20d
Logbook Submission	06/13/16	06/13/16	1d
Seminar Abstract	06/13/16	06/13/16	1d
Seminar Poster	06/14/16	06/20/16	7d
Presentation and demonstration	06/21/16	06/24/16	4d



Bibliography

[1] (2016, Apr.) Introduction to verilog-a. Accessed: 4th April 2016. [Online]. Available: http://verilogams.com/tutorials/vloga-intro.html

[2] X. Cao, J. Mcmacken, K. Stiles, P. Layman, J. J. Liou, A. Oritz-Conde, and S. Moinian, "Comparison of the new vbic and conventional gummel-poon bipolar transistor models," *IEEE Transactions on Electron Devices*, vol. 47, pp. 427–433, Feb 2000.

[3] S. V. Cherepko and J. C. M. Hwang, "VBIC model applicability and extraction procedure for InGap/GaAs HBT," in *Microwave Conference, 2001. APMC 2001. 2001 Asia-Pacific*, vol. 2, Dec 2001, pp. 716–721 vol.2.

[4] G. J. Coram, "How to (and how not to) write a compact model in verilog-a," in *Behavioral Modeling and Simulation Conference, 2004. BMAS 2004. Proceedings of the 2004 IEEE International*, Oct 2004, pp. 97–106.

[5] A. Corporation. (2014) Pdk development training. Accessed: 4th April 2016.

[6] X. J. C. Lan. (2010, Apr.) Development of ic 6.1s pdk based on pas. Accessed: 4th April 2016. [Online]. Available: https://www.cadence.com/cn/cadence/events/Documents/CDNLive!%20Beijing%202012/Track%202-Custom%20Analog/%E8%AE%BA%E6%96%87/CDNLive2012_Development%20of%20IC6.1's%20PDK%20based%20on%20PAS_EDA%20CAS.pdf

[7] O. Mysore, "Compact modeling of circuits and devices in verilog-a," Master's thesis, Massachusetts Institute of Technology, 2011. [Online]. Available: http://dspace.mit.edu/bitstream/handle/1721.1/77441/826502878-MIT.pdf?sequence=2

[8] M. Rudolph, "Documentation of the fbh hbt model," Tech. Rep., 2005.

[9] A. Samelis and D. Pavlidis, "Dc to high-frequency HBT-model parameter evaluation using impedance block conditioned optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, pp. 886–897, Jun 1997.

[10] B. Senapati and C. K. Maiti, "Advanced SPICE modelling of SiGe HBTs using VBIC model," *IEE Proceedings - Circuits, Devices and Systems*, vol. 149, pp. 129–135, Apr 2002.

[11] C.-J. Wei, J. M. Gering, and Y. A. Tkachenko, "Enhanced high-current vbic model," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 4, pp. 1235–1243, April 2005.

[12] Q. M. Zhang, H. Hu, J. Sitch, R. K. Surridge, and J. M. Xu, "A new large signal HBT model," *IEEE Transactions on Microwave Theory and Techniques*, vol. 44, no. 11, pp. 2001–2009, Nov 1996.