Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

High-fidelity behavioural model for electronic support measure microwave receiver front-end

Chen Wu

Defence R&D Canada – Ottawa

Canada

Technical Memorandum
DRDC Ottawa TM 2010-071
May 2010

High-fidelity behavioural model for electronic support measure microwave receiver front-end

Chen Wu
DRDC Ottawa

Defence R&D Canada – Ottawa

Technical Memorandum
DRDC Ottawa TM 2010-071
May 2010

Principal Author

Original signed by Dr. Chen Wu

Dr. Chen Wu

Defence Scientist / Radar Electronic Warfare Section

Approved by

Original signed by Dr. Jean-François Rivest

Dr. Jean-François Rivest

Head/Radar Electronic Warfare Section

Approved for release by

Original signed by Dr. Brian Eatock

Dr. Brian Eatock

Chair/Document Review Panel

Abstract

For many years, hardware development has been an essential step in electronic support measures (ESM) R&D programs carried out in DRDC Ottawa. The objectives of hardware development are:

- To investigate new electronic support measures (ESM) system architectures,
- To test new radar signal detection algorithms using field-collected data,
- To conduct the field trials and demonstrate different operational concepts, and
- To polish researchers' knowledge through using state-of-the-art hardware technologies and applying new theories, so that researchers can better support their client from ESM perspective.

Experience has shown that designing and building ESM hardware or systems is costly in an R&D environment because:

- There are a lot of 'cut-and-tries' in hardware development where sometimes a new design can be completely different from the previous iteration.
- It is very time-consuming and labour-intensive to put ESM system hardware together.
- Nowadays ultra wideband microwave and digital components used in electronic warfare systems are still very expensive.

In order to mitigate the problem, reduce the development time, and partially or sometimes fully achieve the objectives mentioned above, model-based system design and simulation using high-fidelity behavioural models (HFBM) is a good approach and should be a part of the ESM system development cycle.

This report presents how to develop a HFBM of an ESM receiver microwave front-end using measured data and manufacturers' specifications, and the Matlab/Simulink(R) scripting language. The front-end behavioural model and a digitizer behavioural model, which will be presented in another report, will be used to form different ESM digital receiver systems depending on the system architectures. These systems can be applied in different radar electronic warfare scenarios that are created by Satellite Tool Kit(R) software for high-fidelity model-based 'system'-in-the-loop simulation and data generation.

Résumé

Depuis de nombreuses années, la mise au point de matériel constitue une étape essentielle des programmes de recherche et de développement (R et D) sur les mesures de soutien électronique (MSE) menés à RDDC Ottawa. En voici les objectifs :

- étudier les nouvelles architectures des systèmes intégrant des MSE,
- mettre à l'essai les nouveaux algorithmes de détection des signaux radars qui utilisent des données recueillies sur le terrain,
- mener des essais sur le terrain et démontrer différents concepts de fonctionnement,
- perfectionner les connaissances des chercheurs en faisant appel à des techniques de matériel de pointe et en appliquant de nouvelles théories, de sorte que les chercheurs puissent offrir un meilleur appui à leur client du point de vue des MSE.

L'expérience montre qu'il est coûteux de concevoir et de construire du matériel ou des systèmes qui intègrent des MSE dans le cadre de la R et D, pour les raisons suivantes :

- il y a beaucoup d'essais et d'erreurs dans le cadre de la R et D lorsque la nouvelle itération est tout à fait différente de l'itération précédente;
- le rassemblement de matériel pour des systèmes intégrant des MSE prend beaucoup de temps et fait appel à beaucoup de main-d'œuvre;
- de nos jours, les composants numériques et hertziens à bande ultralarge dont on se sert dans les systèmes de guerre électronique (GE) sont encore très coûteux.

Si l'on souhaite atténuer le problème, réduire le délai de mise au point et atteindre, en partie ou parfois en totalité, les objectifs susmentionnés, une bonne approche consiste à concevoir des systèmes à partir de modèles et de mener des simulations en utilisant des modèles de comportement à haute fidélité (HFBM), et cette approche devrait faire partie du cycle de mise au point des systèmes intégrant des MSE.

Le présent rapport indique de quelle façon mettre au point un modèle HFBM de l'étage d'entrée hertzienne d'un récepteur intégrant des MSE utilisant des données mesurées et les spécifications du fabricant, ainsi que le langage de script MATLAB/Simulink(R). Le modèle de comportement de l'étage d'entrée et un modèle de comportement d'un numériseur, qui seront présentés dans un autre rapport, serviront à la mise au point de différents systèmes de réception numérique GE, selon les architectures de système. Ces systèmes peuvent se rattacher dans différents scénarios GE radars qui sont créés par le logiciel Satellite Tool Kit(R) (STK) en vue de la simulation d'un système fondé sur un modèle à haute fidélité dans la boucle et de la génération de données.

Executive summary

High-fidelity behavioural model for electronic support measure microwave receiver front-end

Chen Wu; DRDC Ottawa TM 2010-071; Defence R&D Canada – Ottawa; May 2010.

Introduction: The use of HFBM in model-based system design and simulation plays an important role in modern electronic system research and development [1] [2]. Many examples of commercial electronic system development show that it is a very flexible and versatile approach and has following merits:

- It can greatly reduce system design time and cost.
- It can be used to demonstrate system design concepts without building any hardware.
- It can be easily used in computer simulated scenarios to perform 'system'-in-the-loop simulations.
- It can easily and quickly encapsulate the system design architectures and information at various development stages.
- It can explore new design concepts using hardware technologies that may not be available today.
- It can be used to generate data for signal processing algorithm development and for final system validation.
- It can provide researchers a means to study the system performance versus the characteristics of key components to be used in the system.

In radar electronic warfare (REW) development, the hardware development is an important step towards a successful ESM R&D program [3] to [7]. The main reasons to build system hardware in the ESM R&D period are:

- to create new ESM system architectures using state-of-the-art technologies,
- to verify new signal detection algorithms using field-collected data,
- to demonstrate new ESM system operational concepts through field trials,
- to sharpen researchers' knowledge, and
- to improve current commercial-off-the-shelf ESM systems used by our client.

There is clear evidence to show that ESM system hardware development is a time consuming and expensive process. In order to mitigate the problem and reduce the development time and cost, the high-fidelity model-based system design and simulation approach should be part of the ESM R&D process. From the ESM system perspective, the model should not only be capable of processing the correct amplitude information, but also the signal phase when the signal passes through the system model, taking into account both system noise and ambient noise. Although this approach cannot replace the system hardware development and field trials, it can achieve certain objectives, such as to demonstrate ESM system operational concepts, encapsulate ESM system design, and generate more realistic data for algorithm development, that are set by R&D program. Furthermore, this approach affords researchers a cost effective way to explore many

new design options that cannot to be built in hardware or tested in field trials using currently available technologies.

There is no doubt that behavioural models will be more and more accurate, since the simulation tools, such as Matlab/Simulink [8] and STK [9], are being continuously improved. From a hardware development perspective, this approach allows researchers to investigate the performance of many different hardware systems and select the one with optimal performance, without having to actually build each combination. This report presents the details on how to develop high-fidelity ESM receiver microwave front-end behavioural models using Matlab/Simulink software and their RF Toolbox/RF Blockset.

Results: An MIC/MMIC version of a RF/microwave front-end for a miniature ESM receiver has been designed at DRDC Ottawa. Based on the design, an HFBM of the front-end has been developed using the manufacturers' measured data, and computer simulation tools, namely Matlab/Simulink and their RF Toolbox/RF Blockset. Results show that the model can replicate the RF/Microwave front-end behaviour from the system perspective, and is ready to be connected to an analog-to-digital converter (ADC) behavioural model that has also been built in the Matlab/Simulink environment. These components can be used to build and simulate the receiver channels of an ESM system.

Significance: In order to reduce development cost and time, quickly try different ESM system architectures for different operational concepts, and provide a means to investigate the receiver system performance related to characteristics of the key components in the system, model-based ESM system design should be an integral part of the ESM system R&D cycle. This report is the first to demonstrate the use of Matlab/Simulink and their RF Toolbox/RF Blockset to develop an ES receiver microwave front-end HFBM, which is based on the most recent version of a miniature ESM payload receiver front-end design developed at DRDC Ottawa. The approach can be applied to other research areas, such as: radar systems, communication and navigation electronic warfare system development, and 'system'-in-the-loop tests.

Future plans: This single channel ESM receiver front-end model will be connected to an ADC behavioural model to form an ESM digital receiver channel. Four such digital receiver channels will be paired with four spiral antennas to build a miniature ESM payload model. The payload can be installed on ships, UAVs or small aircrafts, which are modeled in STK software [10]. These platforms that carry the ESM payload behavioural model will be used in different STK scenarios [11] to [12] to demonstrate different ESM operational concepts and produce data that can be used in ESM algorithm developments and for ESM system validation tests.

Sommaire

High-fidelity behavioural model for electronic support measure microwave receiver front-end

Chen Wu; DRDC Ottawa TM 2010-071; R & D pour la défense Canada – Ottawa; Mai 2010.

Introduction : L'utilisation de modèles de comportement à haute fidélité (HFBM) dans la conception et la simulation de systèmes à partir de modèles joue un rôle important dans la recherche et le développement (R et D) sur les systèmes électroniques modernes [1][2]. De nombreux exemples de mise au point de systèmes commerciaux montrent que c'est une approche très souple et polyvalente qui comporte les points positifs suivants :

- elle permet de réduire de beaucoup les délais et les coûts de conception des systèmes;
- elle peut servir à démontrer les principes de conception des systèmes sans qu'il soit nécessaire d'assembler du matériel;
- elle peut facilement être intégrée à des scénarios simulés par ordinateur pour l'exécution de simulations de « systèmes » dans la boucle;
- elle permet d'encapsuler facilement et rapidement les architectures et les données ayant servi à la conception des systèmes à différentes étapes de mise au point;
- elle permet d'explorer de nouveaux principes de conception en faisant appel à des techniques de matériel dont on ne dispose pas nécessairement de nos jours;
- elle peut servir à générer des données en vue de la mise au point d'algorithmes de traitement des signaux et de la validation finale des systèmes;
- elle peut représenter pour les chercheurs un moyen d'étudier le rendement des systèmes par rapport aux caractéristiques des éléments clés dont on se servira dans les systèmes.

Pour ce qui est du perfectionnement de la guerre électronique par radar (GER), la mise au point de matériel constitue une étape importante vers la réussite du programme de R et D sur les mesures de soutien électronique (MSE) [3] à [7]. Voici les principales raisons de construire le matériel des systèmes dans le cadre de la R et D sur les MSE :

- créer de nouvelles architectures de système intégrant des MSE à l'aide de techniques de pointe,
- vérifier de nouveaux algorithmes de détection des signaux à l'aide de données recueillies sur le terrain;
- démontrer de nouveaux principes de fonctionnement des systèmes intégrant des MSE dans le cadre d'essais sur le terrain,
- permettre aux chercheurs de perfectionner leurs connaissances,
- améliorer les systèmes commerciaux courants intégrant des MSE dont notre client se sert à l'heure actuelle.

Il y a des preuves claires qui montrent que la mise au point du matériel des systèmes intégrant des MSE prend du temps et est coûteuse. Pour atténuer le problème et réduire le délai et les coûts de mise au point, il faudrait que l'approche de conception et de simulation des systèmes fondés sur des modèles à haute fidélité fasse partie de la R et D sur les MSE. Du point de vue des systèmes

intégrant des MSE, le modèle devrait pouvoir traiter non seulement les données correctes sur l'amplitude, mais aussi les données sur la phase du signal lorsque le signal passe dans le modèle, compte tenu du bruit du système et du bruit ambiant. Bien que cette approche ne puisse pas remplacer la mise au point du matériel de système et les essais sur le terrain, elle peut permettre d'atteindre certains objectifs, comme la démonstration des principes de fonctionnement des systèmes intégrant des MSE, l'encapsulation de la conception des systèmes intégrant des MSE et la génération de données plus réalistes en vue de la mise au point d'algorithmes, objectifs qui sont fixés dans le cadre du programme de R et D. En outre, cette approche représente pour les chercheurs un moyen rentable d'explorer de nombreuses options de conception qui ne peuvent pas être intégrées au matériel ou mises à l'essai sur le terrain au moyen des techniques en place.

Il ne fait pas de doute que les modèles de comportement seront de plus en plus précis, du fait que les outils de simulation, comme le langage MATLAB/Simulink [8] et le logiciel Satellite Tool Kit (STK) [9], sont sans cesse améliorés. Du point de vue de la mise au point de matériel, cette approche permet aux chercheurs d'étudier le rendement de nombreux systèmes différents de matériel et de choisir celui qui offre le rendement optimal sans avoir à acheter et à assembler chaque combinaison. Le présent rapport donne des détails sur la façon de mettre au point des modèles de comportement de l'étage d'entrée hertzienne d'un récepteur intégrant des MSE à haute fidélité au moyen du langage MATLAB/Simulink et de ses fonctions RF Toolbox et RF Blockset.

Résultats : Une version MIC/MMIC d'un étage d'entrée RF/hertzienne d'un récepteur miniature intégrant des MSE a été conçue à RDDC Ottawa. D'après la conception, un modèle HFBM de l'étage d'entrée a été mis au point à partir de données mesurées du fabricant et d'outils de simulation informatique, à savoir le langage MATLAB/Simulink et ses fonctions RF Toolbox et RF Blockset. Les résultats montrent que le modèle peut reproduire le comportement de l'étage d'entrée RF/hertzienne du point de vue d'un système, et qu'il est prêt à être rattaché au modèle de comportement d'un convertisseur analogique/numérique aussi mis au point à l'aide du langage MATLAB/Simulink. Ces composants peuvent servir à la mise au point et à la simulation des canaux de réception d'un système intégrant des MSE.

Portée : Pour réduire les coûts et les délais de mise au point, mettre rapidement à l'essai différentes architectures de système intégrant des MSE pour différents principes de fonctionnement et prévoir un moyen d'étudier le rendement du système de réception par rapport aux caractéristiques des principaux composants du système, il faudrait que la conception des systèmes intégrant des MSE fondés sur des modèles fasse partie intégrante du cycle de R et D des systèmes intégrant des MSE. Le présent rapport est le premier à démontrer l'utilisation du langage MATLAB/Simulink et de ses fonctions RF Toolbox et RF Blockset pour la mise au point d'un modèle HFBM de l'étage d'entrée hertzienne d'un récepteur de soutien électronique (SE), qui est fondé sur la dernière version de la conception, à RDDC Ottawa, d'un étage d'entrée d'un récepteur miniature de charge utile intégrant des MSE. L'approche peut être appliquée à d'autres domaines de recherche, comme la mise au point de systèmes radars et de systèmes GE de navigation et de communication, et des essais de systèmes dans la boucle.

Recherches futures : Le modèle d'étage d'entrée d'un récepteur intégrant des MSE à un seul canal sera rattaché à un modèle de comportement d'un convertisseur analogique/numérique pour former un canal de réception numérique intégrant des MSE. Quatre canaux de réception numérique seront appariés à quatre antennes en spirale dans le cadre du montage du modèle miniature de charge utile intégrant des MSE. La charge utile peut être installée à bord de navires,

de VAT ou de petits aéronefs, modélisés à l'aide du logiciel STK [10]. Ces plates-formes qui transportent le modèle de comportement de charge utile intégrant des MSE serviront, dans différents scénarios du logiciel STK [11] à [12], à démontrer différents principes de fonctionnement des MSE et à produire des données pouvant servir à l'élaboration d'algorithmes de définition des MSE et à des essais de validation de systèmes intégrant des MSE.

This page intentionally left blank.

Table of contents

List of figures

List of tables

Acknowledgements

The author would like to thank Mr. Tim Reeves, Mr. Mike O'Brien, Mr. John Irza and the Mathworks support team for their great support. Especially thanks to Mr. Reeves for detailed discussions on a number of technical issues on how to use Matlab/Simulink and methods of implementing behavioural models in the Simulink RF Blockset. The author would also like to specially thank Mr. Michael Low for helping to review and edit the report and for providing a number of valuable comments and suggestions.

1 Introduction

Model-based system design should be an integral part of modern electronic system design and development, since it can be used throughout the design, development, testing, and validation process. Another important factor is that modern software design tools give electronic system designers powerful means of building high-fidelity system behavioural models and performing realistic simulations. The use of the HFBMs in model-based system design and simulation is one of the key elements that make model-based system design widely accepted by scientists and system engineers. The HFBM approach is motivated by the aim of establishing a framework for system analysis that respects the underlying physics, sets up the appropriate mathematical concepts, or uses measured data to represent a system and its components.

A HFBM reproduces the behaviour of the original system (or device) such that there is a one-to-one correspondence between the behaviour of the original system and its behavioural model. The level of fidelity of a behavioural model depends on:

1. how well the designer understands the system and the components to be modeled;
2. the capabilities of the tools used to build the model; and
3. the quantity and quality of available data on the components comprising the model.

It is reasonable to assume that designers are subject matter experts and know how to use state-of-the-art software tools, such as Matlab/Simulink for dynamic digital and analog system modeling and design, and STK for real time 3D scenario development. Therefore, it is primarily the quality of the data used to create the model that will affect its fidelity. For example, using a manufacturer's measured data to build a component model will result in a lower fidelity model than if the model is based on measured data taken from the actual components to be used. Moreover, from the ESM system and its related signal processing perspective, the model should not only correctly process signal amplitude, but also phase information. This is especially crucial at microwave frequency bands. However, high-fidelity system models usually require longer simulation times. For example, SADM (Ship Air Defence Model) [13] is a very popular tool for ship defence simulations. However, its electronic system models process only amplitude information, making it difficult to accurately model signal waveform distortions introduced by microwave or optical system models.

This report demonstrates how to develop behavioural models of ESM receiver microwave front-ends using Matlab/Simulink software, based on lab-measured and manufacturers' component data. The rest of the report is organized as follows. The next section briefly describes the design of the ESM receiver microwave front-end, and components used in the system design. Section 3 elaborates on the subsystem behavioural models developed using Matlab and its RF Toolbox. Section 4 describes how to use the subsystem behavioural models prepared in Section 3 to build the microwave receiver front-end in Simulink. The simulated results, demonstrating that the behavioural model can accurately represent the ES microwave receiver front-end, are given in Section 5. The conclusions and future work are described in the last section.

2 ES receiver microwave front-end design

A MIC/MMIC-based ES receiver front-end has been designed at DRDC Ottawa. Figure 1 shows all the devices in the front-end from receiving antennas to intermediate frequency (IF) output. The detailed front-end design and a list of all the components are presented in this section.

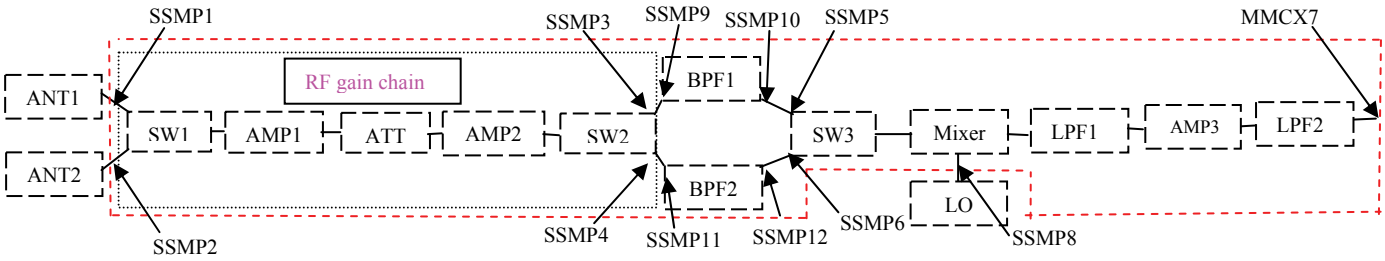

Figure 1 Block diagram of the dual-band ES receiver microwave front-end

In the design, the first switch (SW1) selects the signal between antenna 1 (ANT1) and 2 (ANT2), and the amplitude of the signal is adjusted by the amplifier 1 (AMP1), 2 (AMP2) and the attenuator (ATT). The switch 2 (SW2) and 3 (SW3) select the band pass filter, i.e. BPF1 and BPF2, which determines the front-end operational frequency band. The mixer converts the filtered RF signal to an IF signal. The amplifier 3 (AMP3) is an IF amplifier. Low pass filters (LPF1 and LPF2) eliminate unwanted spurs caused by the nonlinearity of the mixer. The microstrip lines, a MMCX (micro-miniature coaxial) connector and SSMP (Small Solder Mount Push-On) connectors are not shown in the figure.

Table 1 lists components and their main characteristics. Detailed information can be obtained from [14] and [15]. Note that a BPF determines the front-end operation band, which can be anywhere from 2 to 13 GHz. This is because the attenuator operates up to 13 GHz. The attenuator will be replaced by a 2 to 18 GHz MMIC attenuator once it is available on the market.

In Figure 1, the RF devices inside the red dashed lines are to be built on the two printed circuit boards (PCBs) shown in Figure 2. The spacing between two boards is 360 mil. Two BPFs are etched on the bottom board, and the other components are installed on the top board. There are a number of SSMP connectors around the edges of each board to transfer signals back and forth between the top and bottom boards. For example, if ANT1 is selected, the signal from ANT1, which is connected to SSMP1 (in Figure 3) through a piece of coaxial cable, is processed by the RF gain chain consisting of AMP1, ATT and AMP2, and then passes through SSMP3 on the top board to SSMP9 on the bottom board. A cable will be used between these two connecters. After the signal is filtered by BPF1 at the bottom board, it is sent back to the top board from SSMP10 to SSMP5 to be down-converted to IF by the mixer. The output IF signal is available at the MMCX7 connector, after being filtered and amplified at the IF frequency band.

The gain of each amplifier is fixed and the filters cannot be changed once they are installed. The attenuator has 7 different attenuation levels, i.e. attenuation is equal to 0, 1, 2, 4, 8, 16, or 32 dB,

and SW2 and SW3 can have 4 different combinations, thus the front-end behavioural model will need to account for a total of 28 modes of operation.

Table 1 Main components used in the front-end and some of their characteristics

Name in Figure 1	Name	Manufacturer model number	Frequency	Manufacturer
RF/microwave Components				
SW1	switch	HMC347	DC-20 GHz	Hittite microwave corp.
AMP1	amplifier	HMC-ALH482	2-22 GHz	Hittite microwave corp.
ATT	attenuator	HMC424	DC-13 GHz	Hittite microwave corp.
AMP2	amplifier	HMC-ALH482	2-22 GHz	Hittite microwave corp.
SW2	switch	HMC347	DC-20 GHz	Hittite microwave corp.
SW3	switch	HMC547LP3	DC-20 GHz	Hittite microwave corp.
Mixer	mixer	M2-0218	2-18 GHz	Marki microwave
IF and LO Components				
AMP3	IF amplifier	HMC589ST89	DC-4 GHz	Hittite microwave corp.
LO	local oscillator	SMS2040	2-4 GHz	Spinnaker Microwave
		SMS0812	8-12 GHz	Spinnaker Microwave
Band and Low Pass Filters				
BPF1	band pass filter	S-band filter	F_c = 3.11 GHz bandwidth = 500 MHz	In house
BPF2	band pass filter	X-band filter	F_c = 9.04 GHz bandwidth = 500 MHz	In house
LPF1	low pass filter	LFCN-575	DC to 700 MHz	Mini-Circuits
LPF2	low pass filter	LFCN-2000	DC to 2200 MHz	Mini-Circuits

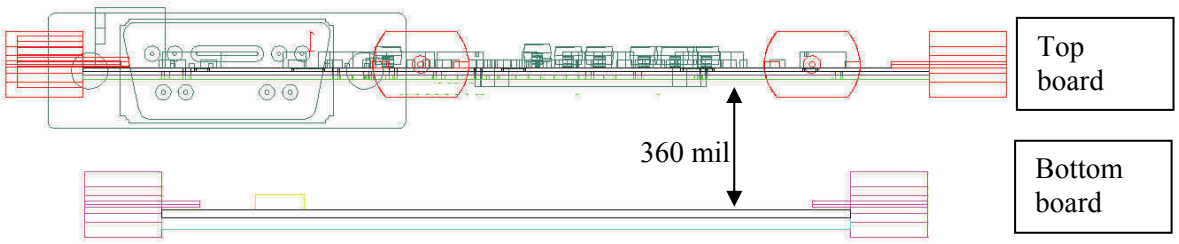

Figure 2 Side view of the two PCBs

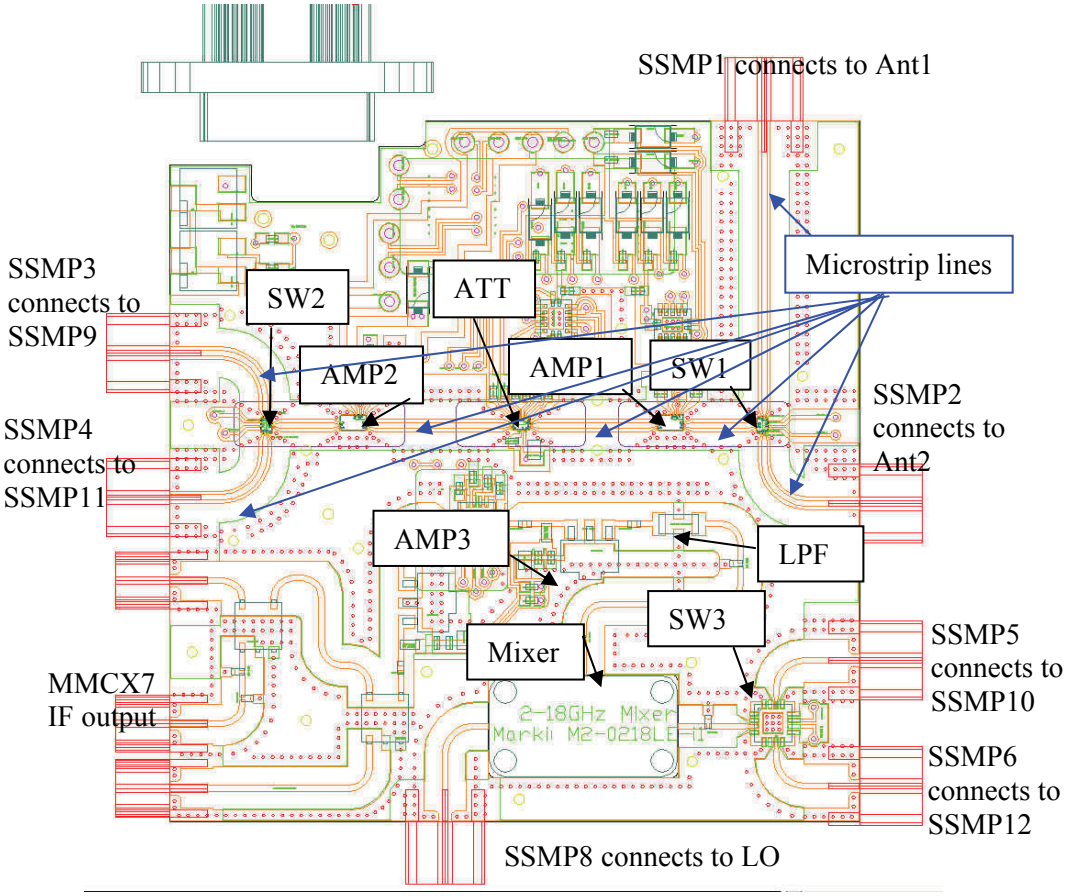

Figure 3 Top view of the top board

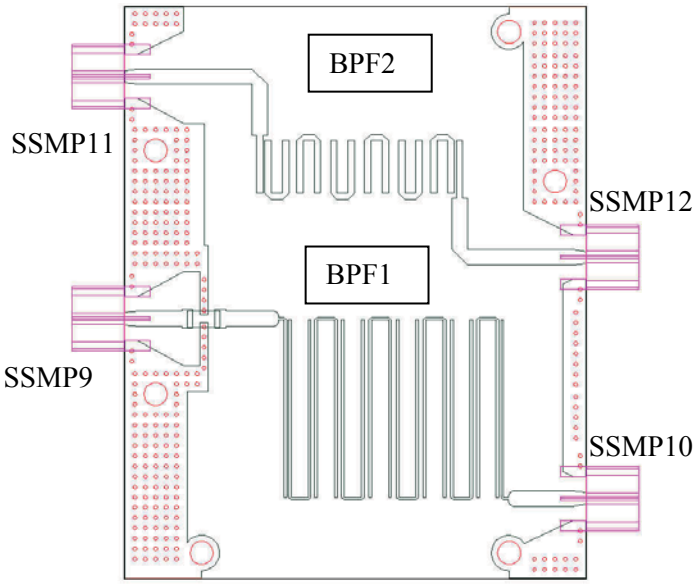

Figure 4 Top view of the bottom board

Since the procedure for building each of the 28 sub-models is the same, without loss generality, this paper only focuses on the development of the sub-model that models the circuit shown in Figure 5, in which SW1 connects ANT1, frequency Band1 is selected, and the attenuator has 0-dB attenuation.

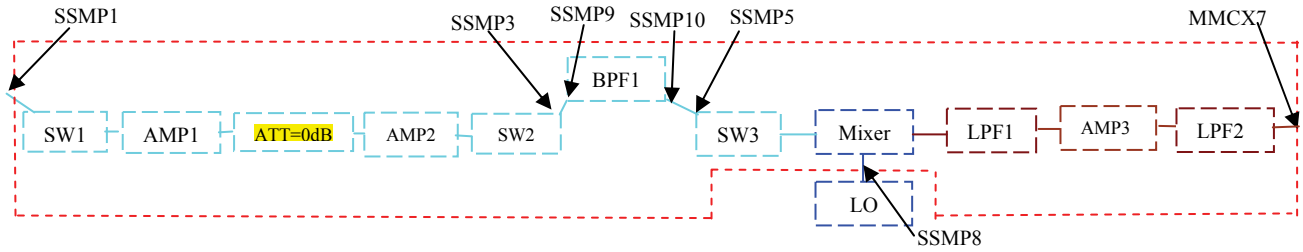

Figure 5 Configuration with SW1 selecting ANT1, SW2 and SW3 selecting BPF1 and 0 dB attenuation from the ATT

2.1 .amp files and component data

The .amp file format is the format developed by Mathworks to store RF/microwave devices' or systems' linear and nonlinear characteristics that can be used in Matlab or Simulink models [7]. Its function is similar to that of the Touchstore® format [16] or Agilent device format [8]. The data contained in an .amp file are: s-parameters, noise figure, and third-order intercept point (IP3) presented in the frequency domain. Using the data posted on the manufacturers' websites and/or provided in device datasheets, each device in Figure 5 is characterized by an .amp file generated by the Matlab RF Toolbox. By cascading a number of devices into a sub-system, the sub-system .amp file can be calculated using the cascade function in the RF Toolbox.

In next section, we will discuss how to build an .amp file for each component in Figure 5 and how to group a number of devices into a sub-system and calculate the sub-system's .amp file in Matlab. These .amp files will be used in the final behavioural model built in Simulink, which simulates system in time domain.

3 Using Matlab RF Toolbox to generate sub-system .amp data files

The goal of building the receiver front-end behavioural model is to simulate the dynamic receiver system and process the time domain signal, which is the radar signal intercepted by the receiver antenna, amplified by the RF gain chain, filtered by the BPF and down-converted by the mixer. There are a number of ways to build the model. In this section, we discuss our approach on how to build the model using Matlab and Simulink, and give details on how to use measured and calculated data to create .amp files that represent the components shown in Figure 5. Then the components in the figure will be grouped into a number of modules, which can be loaded in Simulink. The way for grouping the modules to be used to represent the components in Figure 5 is the model designer's choice. In our model, three modules are used to represent the circuit in Figure 5.

3.1 Method of constructing the time domain Simulink model

Simulink is an environment for multi-domain simulations, i.e. analog and digital domains, and model-based design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let user design, simulate, implement, and test a variety of time-varying systems. For this reason, we chose our final model to be presented in the Simulink environment rather than in the Matlab environment.

Although Simulink can directly use measured data to model component behaviours by using .amp files, this could end up a lot of blocks in the final Simulink model. Even though the model can give the same simulation results, it will result in difficult to manage the model, and reducing the simulation speed. In order to overcome this problem, we group components into three modules or networks, and calculate each network's .amp file in the Matlab environment in the frequency domain. Figure 6 shows the main blocks inside the final Simulink front-end model, which represents the circuit in Figure 5. In the model, 'General RF Amplifier' and 'General IF Amplifier' contain all the components before and after the mixer, respectively. For 'General RF Amplifier', all the components operate at the frequency band that is determined by the BPF1, and 'General IF Amplifier' operates at the IF band, which is centered at 310 MHz with 500 MHz bandwidth. The 'General Mixer' model includes the mixer and its local oscillator.

Figure 6 Main blocks in the Simulink model with the RF, mixer and IF portions of the model indicated in red

In order to include the effects of the transmission lines connecting each of the MIC/MMIC devices, they have been identified as noted in Figure 7 which shows the PCB layout for a part of 'General RF Amplifier' including SSMP1, Microstrip941mil, SW1, Microstrip230mil, AMP1, Microstrip403mil, ATT=0dB (as given in Figure 5), Microstrip460mil, AMP2, Microstrip213mil, SW2, Microstrip444mil and SSMP3. 'General RF Amplifier' also consists of components not

shown, including: Coax-cable1500mil that routes the signal from the top board to the bottom board, SSMP9, BPF1, SSMP10 (in Figure 4), Coax-cable1500mil that routes the filtered signal back to the top board, SSMP5, Microstrip360mil, SW3, and Microstrip254mil (in Figure 8). However, 'General Mixer' only consists of M2-0218 mixer [15], and its LO, while 'General IF Amplifier' consists of Microstrip1402mil, LPFs, Microstrip510mil, AMP3, Microstrip989mil. The Matlab command 'rfckt.cascade' is used to connect the components in each group.

The transmission line names used above indicate not only the type of the transmission line, and also the length of the line. For example: Microstrip1000mil and coax-cable1500mil name a piece of microstrip and a piece of RG86 hand-formable coaxial cable, and their lengths are 1000 mil and 1500 mil, respectively.

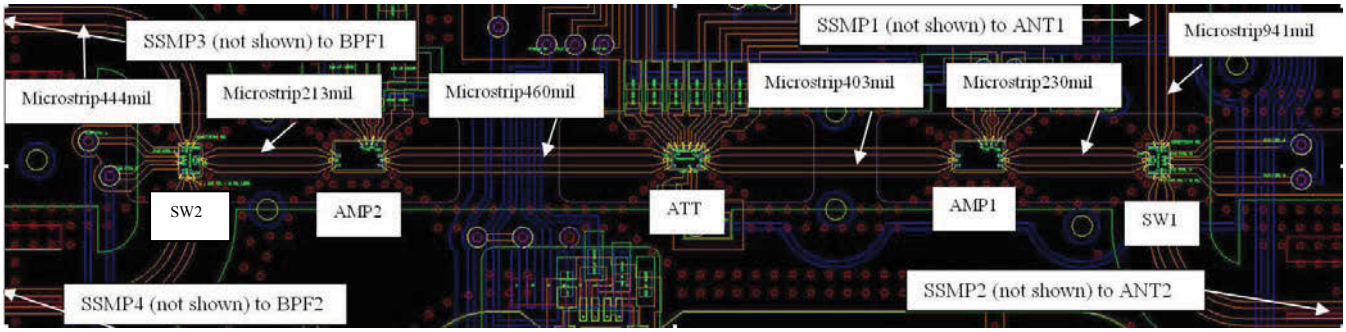

Figure 7 Detailed PCB layout of RF gain chain, which is a part of the General RF Amplifier

Note that:

1. The 'General Mixer' block in Figure 6 is a two-port block. It combines the M2-0218 mixer and its local oscillator. This is because the Simulink RF Blockset models a RF mixer by including the LO model inside the mixer model to take into account the phase noise of the LO synthesizer.

2. The transmission line and connectors between the M2-0218 mixer and its LO (see Figure 5) cannot be modeled in 'General Mixer', fortunately the insertion loss and phase delay caused by the transmission and its connections can be neglected.

3. The microstrip lines connected at the input and output of the M2-0218 mixer are the last component in 'General RF Amplifier', i.e. Microstrip254mil, and the first component in 'General IF Amplifier', i.e. Microstrip1402mil, respectively.

4. The switches are actually three port devices, as shown in Figure 8 and Figure 9, but for modeling purposes, only two ports are relevant. When the front-end is set in the state shown in Figure 5, SW1 connects RFC to RF2 (Figure 9 right), SW2 connects RFC to RF1 (Figure 9 left), and SW3 connects RFC to RF2 (Figure 8), so that each switch shown in Figure 5 effectively becomes a 2-port device. The s-parameters for the 2-port equivalent device will be extracted from the measured 3-port s-parameters measured by vender. Details are given in section 3.2.1.4.

5. In general, the user can arbitrarily group the components in a circuit. Each group will be represented by an .amp file generated by Matlab. The .amp files will be loaded to the Simulink model. More details on how to generate the .amp are given in the following sections.

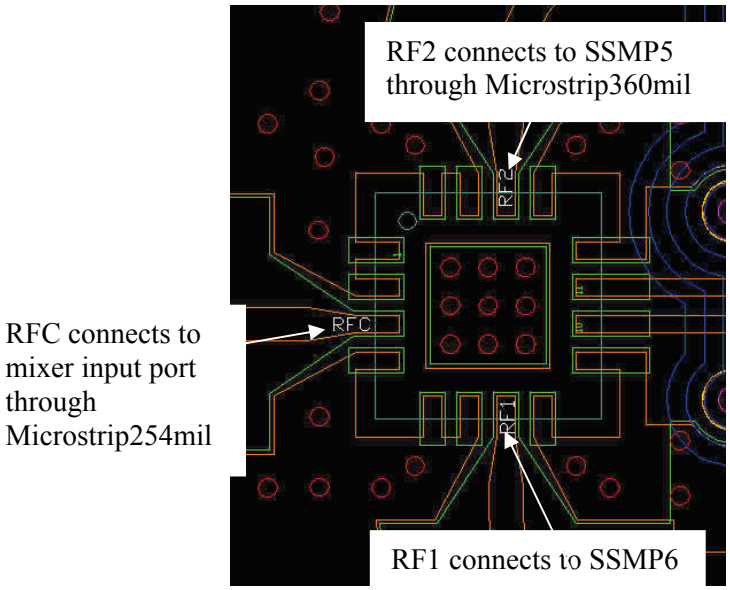

Figure 8 Ports of SW3 using HMC547LP3

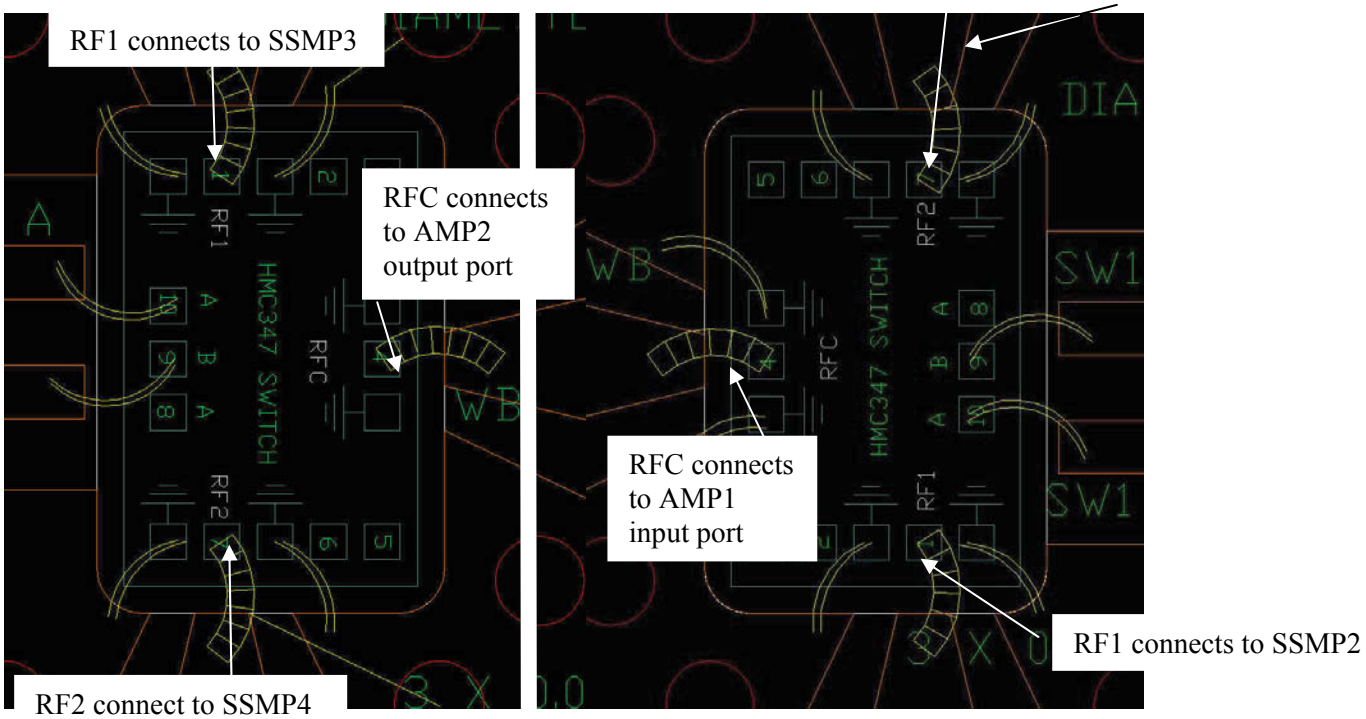

Figure 9 Ports of SW2 (left) and SW1 (Right) using HMC347

3.2 Using Matlab to generate General_RF_Amplifer.amp file

The 'General RF Amplifier' block in Figure 6 represents 22 components from SSMP1 to the microstrip line connected to the output port of SW3. Those components are shown in cyan in Figure 5. These components can be grouped into seven categories. They are listed in the following table.

Table 2 Components in 'General RF Amplifier' sub-system and their categories

Categories		Component Name in Figure 5
1	Switch	SW1, SW2, SW3
2	Amplifier	AMP1, AMP2
3	Attenuator	ATT = 0 dB
4	Band pass filter	BPF1
5	Microstrip	Microstrip941mil, Micrestrip230mil, Microstrip403mil, Microstrip460mil, Microstrip213mil, Microstrip444mil, Microstrip360mil, Microstrip254mil
6	Cable	Coax-cable1500mil × 2
7	Connector	SSMP1, SSMP3, SSMP9, SSMP10, SSMP5

The Matlab RF Toolbox provides two ways to model these components.

- The first way is to use measured s-parameter data, noise, and device nonlinearity data to model the components listed in category 1 to 4 by using 'rfckt.amplifer' class.
- The second way is to use RF Toolbox built-in physical models to model microstrip and coax transmission lines. In this way, the user simply inputs the physical dimensions and dielectric constants of the transmission lines to these models, and the Toolbox calculates the s-parameters for each transmission line.

Since there are no connector models built in the RF Toolbox, in order to take into account the effect of each SSMP connector and the connection between the connector and transmission line, we add 0.3 dB extra insertion-loss to a transmission line that is connected to the SSMP.

Note that although the Toolbox s-parameter cascade function ('rfckt.cascade') can take into account the mismatch between cascaded components, we assume that each component used in the front-end has 50-ohm input and output ports, so the mismatching between components can be neglected.

The Matlab code that calculates the General_RF_Amplifer.amp file is attached in Annex A - 'ESM_FrentEnd_20091030_Figure5_single_ch_ATT0dB.m'. Here we give a detailed discussion

of the implementation, whose block diagram is shown in Figure 10. In the figure, we present the main structure of the Matlab code in Annex A. After that, we depict the details of each line of the code, so the reader can have a clear idea on how to model different RF components using Matlab RF Toolbox.

The Matlab code has four main parts. They are to:

- initialize the program and create components,
- cascade components,
- display results, and
- create General_RF_Amplifer.amp.

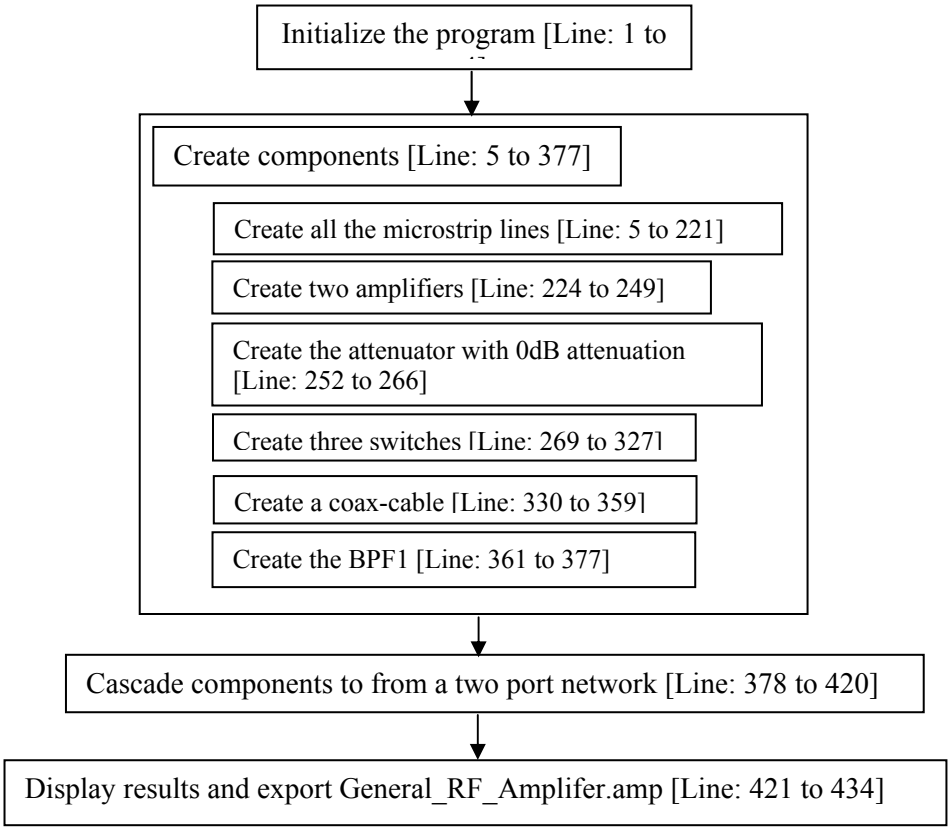

Figure 10 Block diagram of the Matlab code to generate the General_RF_Amplifer.amp file

3.2.1 Create component models using 'rfckt.amplifier' class

The 'rfckt.amplifier' class is a built-in Matlab class. It is used to represent RF/microwave amplifiers. Although it is designed to construct RF/microwave amplifiers, it can also be used to construct passive RF/microwave devices. This is because the objects created by the class contain the most complete set of data for RF/microwave simulations. These data include network

parameters (such as s-parameters), noise data, and nonlinearity data, which can be used in a cascaded RF/microwave network to calculate system characteristics.

In this sub-section, we first use Matlab RF Toolbox built-in transmission line classes, such as 'rfckt.microstrip' and 'reckt.coaxial', to create transmission line models, and then use the data calculated in these models to create transmission line objects by using 'rfckt.amplifier' class. For the amplifiers, attenuators, and switches we use the 'rfckt.amplifier' class directly to create the corresponding objects. A list of the data contained in the objects created by the 'rfckt.amplifier' class is shown in Figure 13.

3.2.1.1 Initialization constants

Since the $1/1000^{th}$ of an inch (mil) is used as the linear unit of measurement when we design the layout of MIC and MMIC circuits, Line 2 defines a constant to convert mils to meters. Except for ATT and BPF1, most of the components have data from 2 to 20 GHz, so the frequency used to create these components is defined by 'freq' in Hz. Z0 (= 50 ohm) is the characteristic impedance of transmission lines and is also the microwave receiver front-end system impedance.

3.2.1.2 Microstrip line objects in Matlab

Lines 11 to 20 define the microstrip line template object using the 'rfckt.microstrip' class, which includes all the physical dimensions of the microstrip cross-section and material parameters. Figure 11 lists the properties in the microstrip line template object. When the template is used, only 'LineLength' has to be changed for different microstrip lines.

Using the template, the first microstrip is created from Lines 22 to 46. The 'LineLength' of 941 mil is input at Line 27. At Line 28, the RF Toolbox command - 'analyze' calculates all the parameters associated with the microstrip line at each frequency point defined by 'freq'. The calculated results are saved in the variable - 'AnalyzedResult' in Figure 11, and the details in the 'AnalyzedResult' are given in Figure 12. The parameters listed in Figure 12 are applicable to general microwave passive or active devices. From Figure 12, one can find that:

1) There are total 1801 data points which is equal to the number of points in 'freq',
2) NF is the noise figure of the device,
3) The device has OIP3 data at each frequency,
4) The impedances of the source (ZS), load (ZL) and the characteristic impedance (Z0) of the line are all 50 ohm.

Since the microstrip line is very short, the line loss contributed by the material loss and radiation loss of the line are neglected. So the noise figure contributed by the line loss is zero. However, if the line loss is significant the model can take into account the line loss as well. Since the microstrip line is a passive device, the OIP3 calculated by the 'analyze' is infinite.

```
                Name: 'Microstrip Transmission Line'
               nPort: 2
    AnalyzedResult: [1x1 rfdata.data]
          LineLength: 0.0239
           StubMode: 'NotAStub'
         Termination: 'NotApplicable'
               Width: 4.5720e-004
              Height: 2.5400e-004
           Thickness: 1.7780e-005
            EpsilonR: 4.5000
         LossTangent: 1.0000e-003
           SigmaCond: 58130000
```

Figure 11 Properties of a microstrip object in Matlab

```
                Name: 'Data object'
                Freq: [1801x1 double]
    S_Parameters: [2x2x1801 double]
           GroupDelay: [1801x1 double]
                  NF: [1801x1 double]
                OIP3: [1801x1 double]
                  Z0: 50
                  ZS: 50
                  ZL: 50
             IntpType: 'Linear'
```

Figure 12 Details in 'AnalyzedResult' shown in Figure 11 after 'analyze'

As we mentioned earlier, in order to take into account the insertion losses caused by 1) the SSMP1 connector, 2) the connection between the SSMP1 connector and the microstrip line and 3) the connection between the microstrip line and the port RF2 on SW1 (Figure 9 right), we lump these losses into iL941 at Line 23, in which 0.2 dB is contributed by the connector and another 0.1 dB is due to the connections. The user can adjust these insertion loss values accordingly based on his/her experiences. Lines 30 to 34 add the lumped insertion loss to the s12 and s21 of the microstrip without changing the phase. For a passive device, its noise figure usually is equal to its insertion loss [17], and its OIP3 can be assumed to be very large, for example infinity. Using the modified s-parameter (Line 33 and 34), NF, and OIP3 (Line 37 and 39), we create an object, called Microstrip941mil, using the 'rfckt.amplifier' class to represent the microstrip line (Line 45 and 46). Figure 13 lists the data contained in the Microstrip941mil object. The reasons for using Microstrip941mil object to replace the microstrip object given in Figure 11 are:

1. The model in Figure 11 does not take into account the extra insertion losses;
2. There is not NF and OIP3 properties listed in the Figure 11; and

3. As we discussed, the ‘rfckt.amplifier’ is a general model that can be used to model any passive or active RF/microwave component and can be easily cascaded as one will see in later sections.

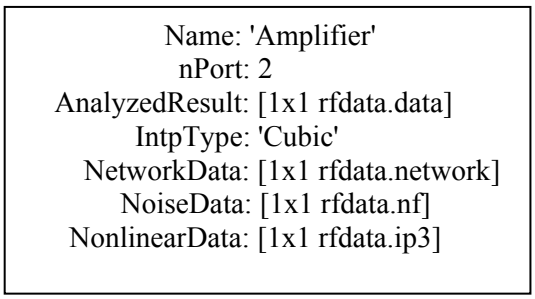

Figure 13 Parameters in an rfckt.amplifier object

In summary, we use the RF Toolbox built-in microstrip class to create and calculate RF parameters of a microstrip line, such as s-parameters and Z0, from its physical dimensions and material information, and then use these calculated RF parameters, including insertion losses caused by discontinuities to generate an ‘rfckt.amplifier’ object to fully characterize the microstrip line. The amount of insertion loss caused by discontinuities is estimated based on some simulations in [18]. From Line 48 to 221, the Matlab code creates another seven ‘rfckt.amplifier’ objects of microstrip lines with different line lengths.

3.2.1.3 Amplifier and attenuator objects in Matlab

Two amplifier objects, LNA_HMC_ALH482_1 and LNA_HMC_ALH482_2, are created on Lines 228 and 241 using the ‘rfckt.amplifier’ class. Since the two amplifiers are of the same model made by Hittite [14], the measured amplifier s-parameters, downloaded from the Hittite website, is loaded into the model. The NF and OIP3 data are taken from the device datasheet, and are set in Lines 225 to 227 and Lines 238 to 240 for each amplifier, respectively.

The ‘rfckt.amplifier’ class is also used to create the attenuator model, which is called ‘ATT_HMC42400p0dB’ (Lines 256 to 265). The measured s-parameters, when the attenuator has 0 dB attenuation, are loaded into the model. Again the NF and OIP3 data are taken from the data sheet and input at Lines 260 to 263.

3.2.1.4 Switch objects in Matlab

Because the switches used in the front-end are SPDT (single-pole double-throw) switches, each switch has two possible configurations, i.e. RFC connects to RF1 or RF2 (Figure 8 and Figure 9

Figure 9). For each connection, the manufacturer provides the 3-port s-parameters, which has file extension ‘.s3p’. In a 3-port s-parameter, for example: ‘SW_HMC347_rf1_selected_deembedded.s3p’, RFC is port #1, RF1 is port #2, and RF2 is port #3.

Lines 274, 276 and 278 create three 'rfckt.passive' objects, which help us to load the .s3p files into the Matlab environment. To model the 2-port switches in Figure 5, we have to extract the 2-port s-parameters (.s2p) from the 3-port s-parameters (.s3p). To do so, the 'analyze' command is used to interpolate measured data at the frequency points given in 'freq' at Line 282, 283 and 284. The interpolated data are saved in the 'AnalyzedResults' of three switch objects. The Matlab RF Toolbox command - 'snp2smp' is used on Line 288, 303 and 317 to extract the 2-port data. Here 'snp2smp' means 's3p-to-s2p' conversion. Using the example of SW1 shown in Figure 9 (right), since the signal input is applied to RF2 (port #3) and the output is connected to RFC (port #1), the '[3 1]' in Lines 288 and 289 mean the input port of the equivalent 2-port device for SW1 is RF2 and the output port is RFC.

Once the 'NetworkData' of the switch is obtained, the 'NoiseData' and 'NonlinearData' are given on Line 292 to 297 for SW1. The NF of SW1 is equal to the insertion loss at each frequency obtained from |s21|, and OIP3 data is obtained from the device datasheet. Finally on Line 298, 312 and 326, three 'rfckt.amplifier' objects are created to represent three 2-port switches.

3.2.1.5 RG188 A/U coaxial object in Matlab

Using the 'rfckt.coaxial' class, a cable object is created on Line 332. Since the characteristics impedance of the cable is 49.81 ohm, which is shown in the following equation, and it is very close to 50 ohm, so we use 50 ohm for the impedance of the cable.

$$Z0 = \frac{1}{2\pi}\sqrt{\frac{Mu_0\ MuR}{Epsilon_0\ EpsilonR}}\ \ln\left(\frac{\text{OuterRadius}}{\text{InnerRadius}}\right)$$
$$= \frac{1}{2\pi}\sqrt{\frac{4\pi\times10^{-7}\times1}{8.854\times10^{-12}\times2.1}}\ln\left(\frac{0.001524}{4.5719998\text{e-}004}\right)$$
$$= 49.81\,ohm$$

$Mu_0 = 4\pi\times10^{-7}\ H/m$ and $Epsilon_0 = 8.854\times10^{-12}\ F/m$.

```
                Name: 'Coaxial Transmission Line'
               nPort: 2
      AnalyzedResult: [1x1 rfdata.data]
          LineLength: 0.0381
            StubMode: 'NotAStub'
         Termination: 'NotApplicable'
         OuterRadius: 0.001524
         InnerRadius: 4.57199998e-004
                MuR: 1
            EpsilonR: 2.1
         LossTangent: 5.00e-004
           SigmaCond: 58130000
```

Figure 14 Parameters defining RG188 A/U coaxial object in Matlab

As was done with the microstrip lines, an 'rfckt.amplifier' object is created to represent the cable (Line 358), where the noise figure is equal to the insertion loss of the cable and the extra loss from connecters and connections between the cable and connecters (Line 343 to 350). The OIP3 is equal to infinity (Line352).

3.2.1.6 S-band filter model in Matlab

The final frequency points for the 'General_RF_Amplifier' block will be determined by the frequency points used for the BPF1 object, i.e. freqs' on Line 361. The measured s-parameters of BPF1 (S_band_BFP_1.s2p) are loaded into an amplifier object – 'A' on Line 363. After interpolating the measured data to the frequency points given in 'freqs', the interpolated data is saved in the 'AnalyzedResult' of object 'A' at Line 364, the noise figure of the device is obtained using the insertion loss of the filter (Line 365), and OIP3 is assumed infinity, since a BPF is a passive device. The S-band BPF1 is represented by an 'rfckt.amplifier' object, whose name is 'S_BPF1' (Line 373).

One may notice that here we use an 'rfckt.amplifier' object – 'A' to load the measured s-parameter of the filter, while in the switch sub-section the 'rfckt.passive' object is used to load the switches' data in Line 274, 276 and 278. The reason is that an 'rfckt.passive' object can have a number of ports, while an 'rfckt.amplifier' object can only be a 2-port device. However, for a 2-port passive device, such as a filter, we also can use an 'rfckt.passive' object to load its s-parameters.

3.2.2 Cascading two-port devices and generating General_RF_Amplifier.amp

From Line 380 to 418, there are three steps required to create the object that represents the all the components in cyan shown in Figure 5. This is because the frequency bandwidths of the circuits before and after BPF1 are different. An 'rfckt.amplifier' object – 'NetworkBeforeBPF1' is created on Line 401 using the data generated by the 'rfckt.cascade' object (Line 380), which contains twelve 2-port devices given from Line 381 to 392. An 'rfckt.cascade' object – 'NetworkAfterBPF1', which has four 2-port devices (Line 405 and 406), is created on Line 415. Finally the 'General_RF_Amplifier' is created on Line 419 by cascading three objects – 'NetworkBeforeBPF1', 'S_BPF1' and 'NetworkAfterBPF1'. After the 'analyze' function is run on the 'General_RF_Amplifier' object, the object's 'AnalyzeResults' data is saved to the hard drive and is ready to be used in the Simulink model (Figure 6).

The network, noise figure and OIP3 data of 'General_RF_Amplifier' are plotted in Figure 15. One can find that:

- The 'amplifier' has good return loss from 2.8 to 3.5 GHz (less than -10 dB), which is the receiver frequency band in S-band.
- Its gain is flat in the band and is about 20 dB.
- It has about 5 ns differential group delay between the edges and the center of the band.

- Its noise figure is about 4.8 dB across the band.
- Its OIP3 is about 35 dBm.
- Its 3 dB bandwidth of the RF pass band is about 500 MHz, which is from 2.86 to 3.36 GHz, and the center frequency is at 3.11 GHz.

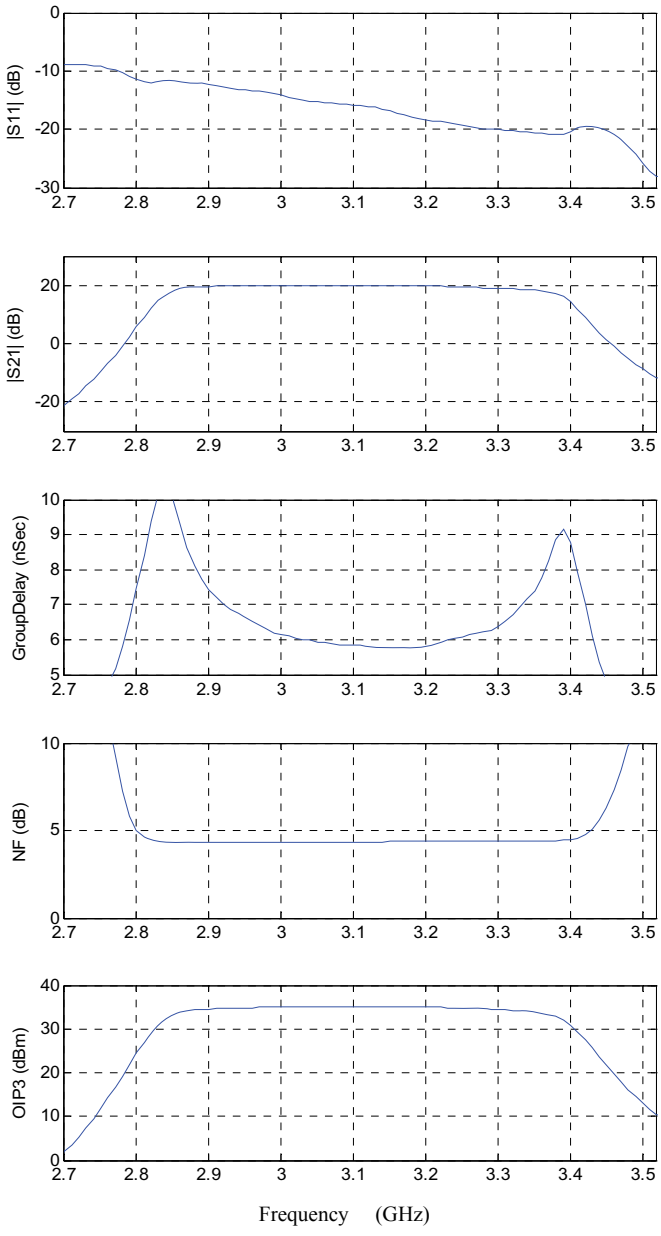

Figure 15 Characteristics of General_RF_Amplifier.amp

3.3 Using Matlab to generate General_Mixer.amp file

Annex B lists the Matlab code to generate the 'General_Mixer.amp' file to be used in the Simulink model. An 'rfckt.mixer' object is created on Line 2 and its parameters are shown in Figure 16. From the figure one can observe that:

1. Matlab RF Toolbox models a RF mixer as a 2-port device, i.e. 'nPort' = 2;
2. The 'NetworkData', 'NoiseData', and 'NonlinearData' are same as those in an 'rfckt.amplifier' object. However, the frequencies at the input and output ports of this 2-port device are different. For example in the model that we are building, the center frequency of the signal at the input port is 3.11 GHz, and the output signal center frequency is 310 MHz, since the LO frequency is 2.8 GHz
3. The MixerSpurData is unique to the 'rfckt.mixer' Matlab class. Although this parameter is implemented in the Matlab RF Toolbox mixer model, it is not used by the Simulink RF Blockset mixer model, since the mixer spur model has not been implemented yet.
4. The last three parameters in the list are the parameters for the LO device. In this case, the LO frequency is 2.8 GHz, and the phase noise performance of the LO oscillator is defined by 'FreqOffset' and 'PhaseNoiseLevel'. Minimum two frequency points are required for LO phase noise model.

```
                Name: 'Mixer'
                nPort: 2
      AnalyzedResult: [1x1 rfdata.data]
             IntpType: 'Cubic'
         NetworkData: [1x1 rfdata.network]
           NoiseData: [1x1 rfdata.nf]
       NonlinearData: [1x1 rfdata.ip3]
      MixerSpurData: [1x1 rfdata.mixerspur]
           MixerType: 'Downconverter'
                FLO: 2.8000e+009
           FreqOffset: [3x1 double]
    PhaseNoiseLevel: [3x1 double]
```

Figure 16 Parameters in the 'rfckt.mixer' object

When the input signal is 3.11 GHz, and power level is -10dBm, the output signal and spurs generated by the mixer are shown in Figure 17. From the figure, one can find that:

1. The output signal power level is -17.29 dBm, which is 7.29 dB lower than the input signal power (-10 dBm).

2. The power of the first spur at 620 MHz is about 67 dB lower than that of the desired signal.
3. In the design shown in Figure 5, LPF1 mainly removes spurs between 2 to 4 GHz, before the signal enters the IF amplifier, since AMP3 operates from DC to 4 GHz (Table 1). The second LPF will further suppress the spurs.

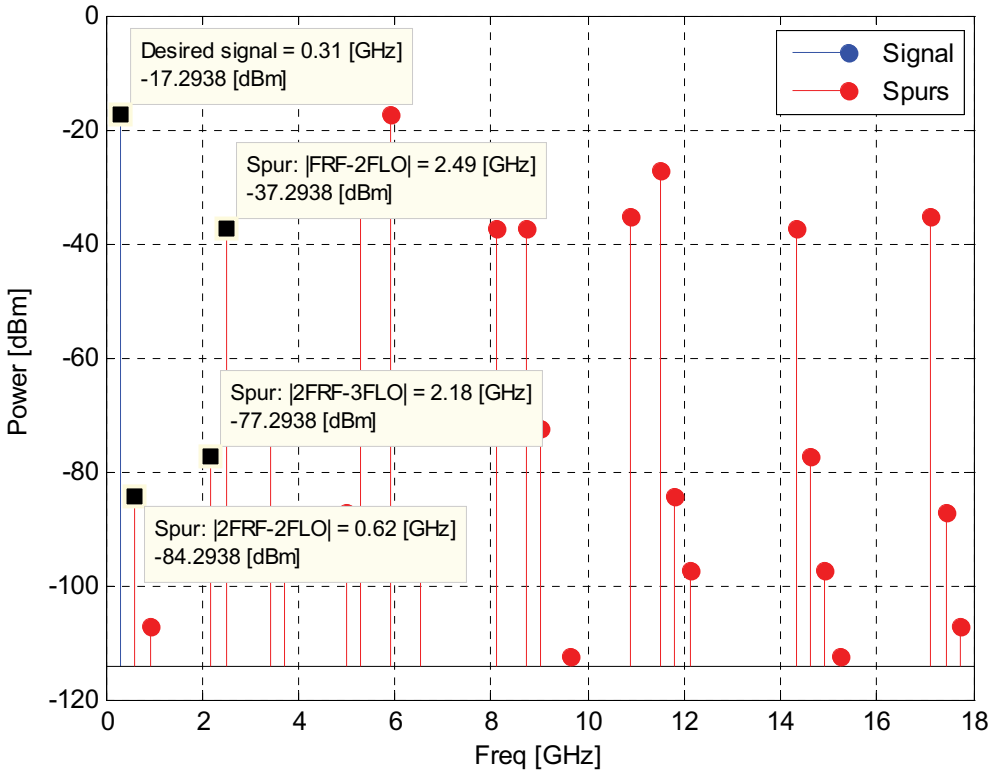

Figure 17 Desired signal and spurs at the output port of the mixer, when input signal is 3.11 GHz at -10 dBm power level

3.4 Using Matlab to generate General_IF_Amplifer.amp file

Following the same approach as described in section 3.2, the General_IF_Amplifier.amp file is used to represent the circuit that is after the mixer in Figure 5. The Matlab code is listed in Annex C, and the data contained in the file are plotted in Figure 18. From the plots, we find that:

1. The circuit has very good return loss from DC to 650 MHz.
2. There is more than 60 dB gain difference between the IF pass-band signal and signal between 2 to 4 GHz.
3. There is more than 50 dB gain difference between the IF pass-band signal and signal between 5 to 6 GHz.

4. The differential group delay in the IF band that we are interested in (60 to 560 MHz) is about 0.5 ns.
5. The noise figure is between 4 to 5 dB.
6. The OIP3 is at a level of 45 dBm.

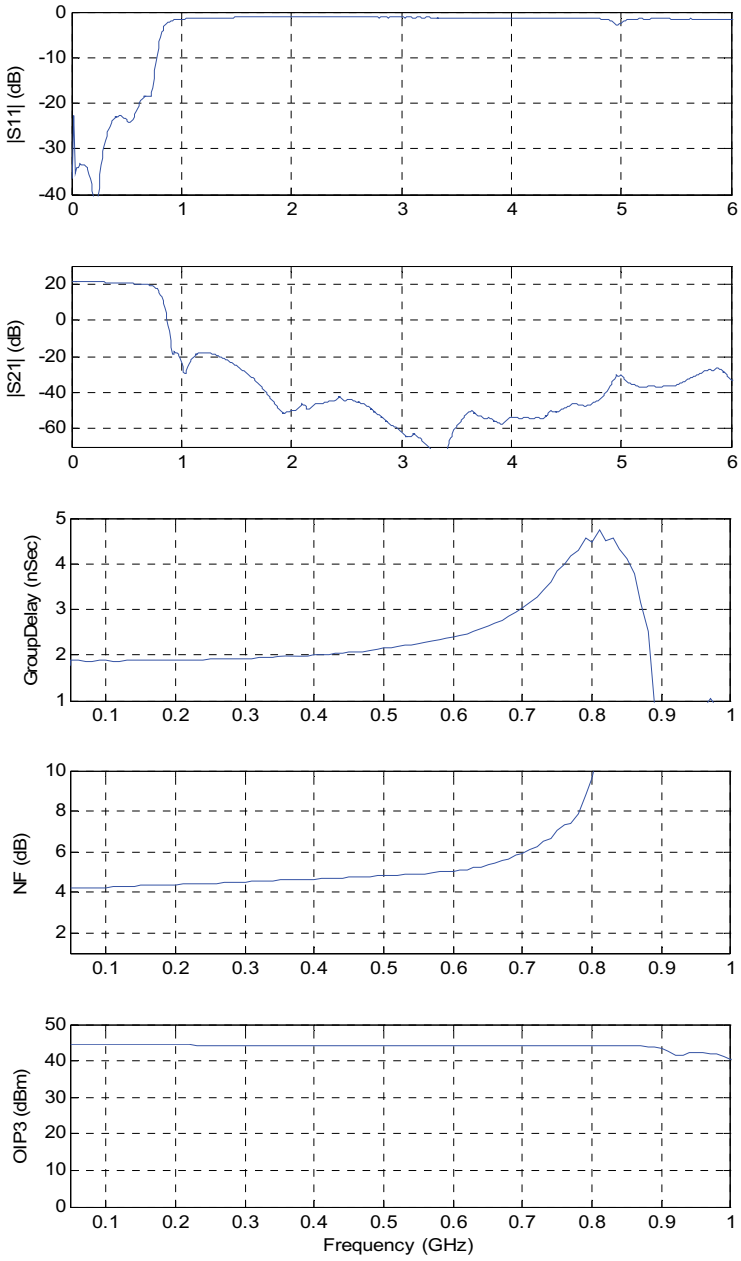

Figure 18 Characteristics of General_IF_Amplifier.amp

3.5 Section summary

In this section, first we briefly introduced the design of the ES payload receiver microwave front-end, and then discussed how to build microwave passive and active objects in Figure 5 using the Matlab RF Toolbox. We then grouped individual objects in three composite objects (Figure 6), each with its own .amp file that will be used in the Simulink model.

4 Simulink model of the receiver front-end

In the previous section, the frequency domain data, which are provided by component manufacturers or calculated from RF Toolbox device models, are used to generate three .amp files. In this section, we discuss how to use these three files in the Simulink RF Blockset to create the front-end behavioural model, which can be used to process time domain signals.

4.1 How the Simulink RF Blockset simulates RF circuits

When Simulink simulates a model that contains physical RF blocks which contain frequency data, the RF Blockset software automatically determines the modeling frequencies of the physical RF system using the 'Center frequency' inside the 'Input Port' block (Figure 20 and Figure 21), and builds a baseband equivalent model with the pass-band data input to each block. The main reason the Simulink RF Blockset uses this approach is that Simulink performs simulations in time domain, while the physical RF blocks are specified using frequency domain data, as can be seen in the three .amp files generated in the last section. The documentations and help for the Simulink RF Blockset in [7] provide details on how Simulink models a system that has physical RF blocks.

4.2 Construction of the RF front-end Simulink model

Figure 19 shows the Simulink model of the receiver front-end. In the figure, the blocks are grouped by different colors, and Table 3 gives the functionalities of the different groups.

There are three Simulink subsystems. They are 'Front-End', 'Complex Multiply' and 'dBW to Linear' blocks shown in Figure 19. From the Simulink perspective, a subsystem is a block comprised of a group of blocks. For example, the 'Front-End' subsystem represents the five blocks shown in Figure 20.

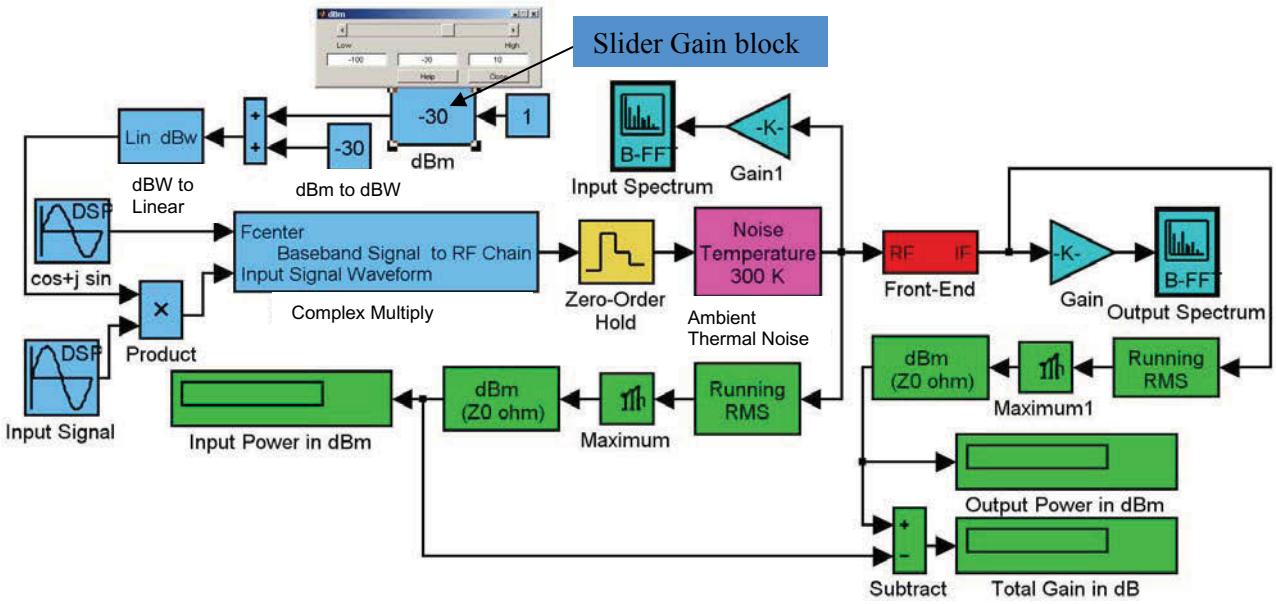

Figure 19 Simulink model of RF front-end

Table 3 Blocks in different colors perform different functions in Figure 19

Color	Functions
Red	RF Front-End subsystem
Light blue	To generate the complex baseband equivalent RF signal
Yellow	To sample the continuous-time baseband signal at a specified sampling rate
Pink	To model thermal noise at the ambient temperature, which is assumed to be 300^{o}K
Green	To measure signal power levels at the input and output ports of the RF front-end
Cyan	To measure signal frequency and amplitude

4.2.1 RF Front-End subsystem

As shown in Figure 6, the RF front-end subsystem contains five Simulink RF blocks (Figure 20). The 'RF' and 'IF' before the 'Input Port' and after the 'Output Port', respectively, are merely the labels that are displayed on the 'Front-End' block in Figure 19.

Figure 20 Simulink RF blocks in the Front-End subsystem

One can see that the link lines between the red blocks are different from those shown in Figure 19. This is because the data used in red blocks are frequency-domain data. In order to use these physical models, Simulink has to build the subsystem baseband impulse response function by using the frequency-domain data.

4.2.1.1 'Input Port' and 'Output Port' blocks

The detailed functionalities of the 'Input Port' block can be found in RF Blockset help, and the parameters that are set in the block interface are shown in Figure 21. The variables used in the 'Input Port' are defined in the main model 'InitFcn' given in Figure 22. In addition to the ambient temperature noise, the noise of the front-end receiver is added in the calculation, since the 'Add noise' box is checked in the 'Input Port' block. The only parameter defined in the 'Output Port' block is the load impedance that is equal to Z0 (Figure 23).

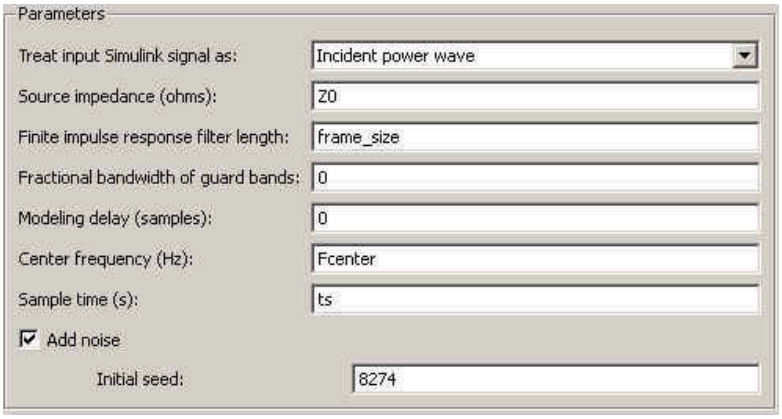

Figure 21 Parameters of Input Port

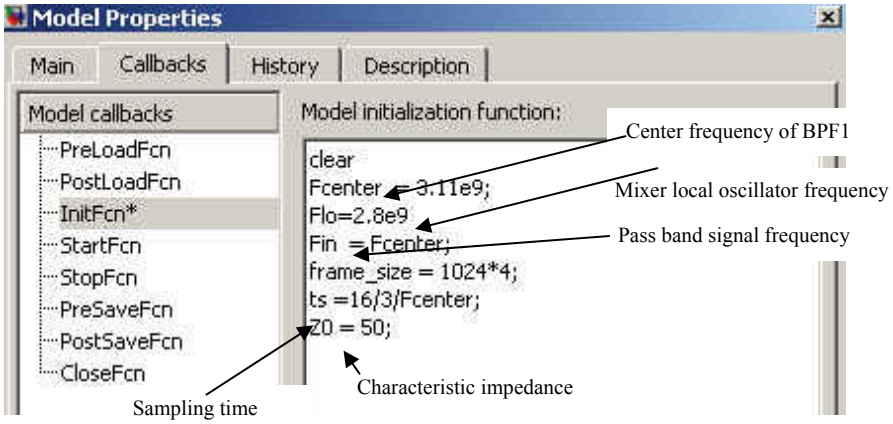

Figure 22 Variables defined in the model 'InitFcn'

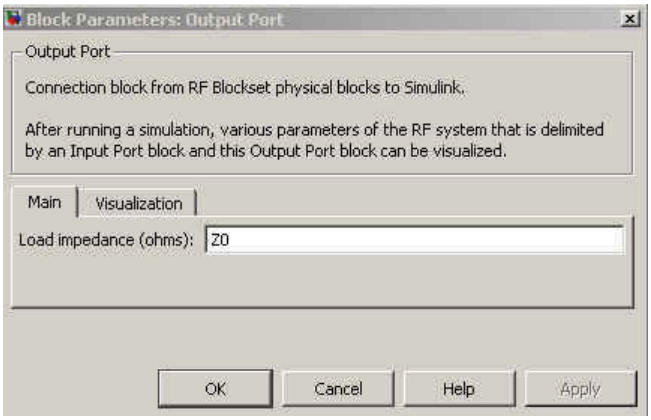

Figure 23 Parameter of Output Port

4.2.1.2 Load and display of data for 'General_RF_Amplifier', 'General_Mixer' and 'General_IF_Amplifier' blocks

The General_RF_Amplifier.amp is loaded in the 'General_RF_Amplifier' block through the Block Parameters Window (Figure 24), which includes network, noise and nonlinearity data. From the Visualization of the Block Parameters Window (Figure 25), one can verify that the loaded data. For example the s-parameters of the 'General_RF_Amplifier' are displayed in the Visualization page (Figure 26). They are the same as the first two plots in Figure 15.

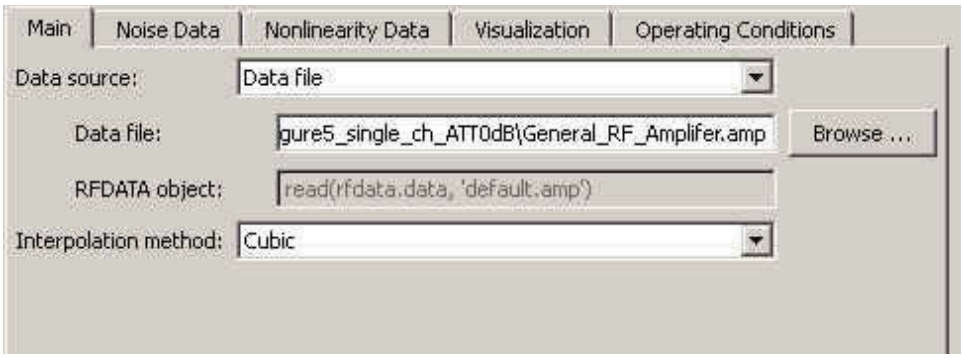

Figure 24 'General_RF_Amplifer' Block Parameters Window

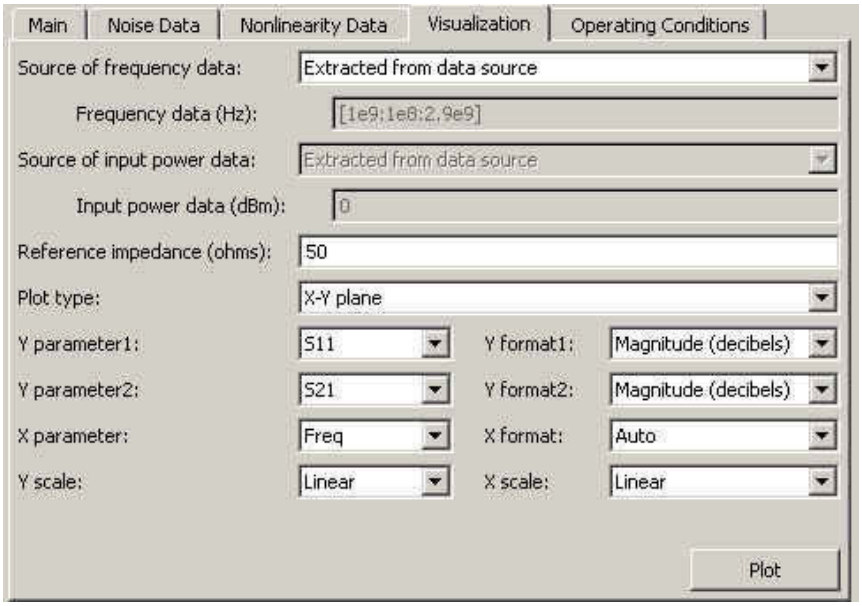

Figure 25 Visualization page in 'General_RF_Amplifer' Block Parameters

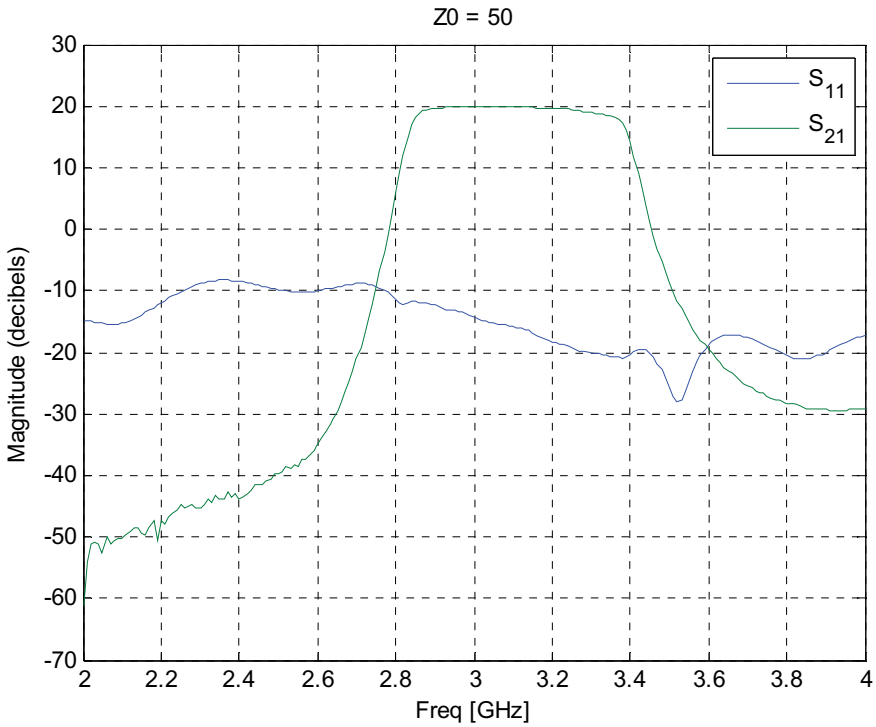

Figure 26 S-parameters of the 'General_RF_Amplifier' displayed by its Visualization page

The Block Parameters Windows of the 'General_Mixer' and 'General_IF_Amplifier' blocks are shown in Figure 27 and Figure 28, respectively. For 'General_Mixer', the LO frequency is Flo and the mixer is defined as a Downconverter.

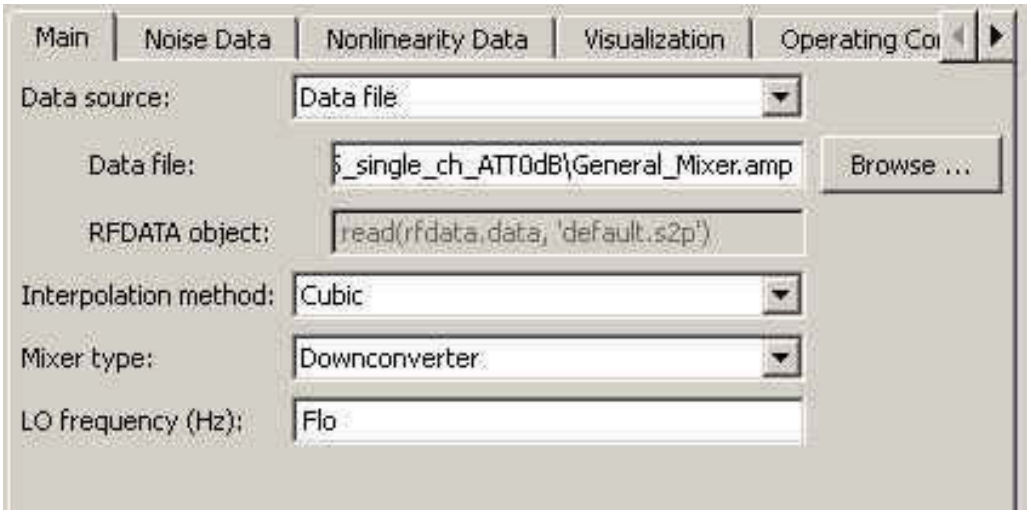

Figure 27 'General_Mixer' Block Parameters window

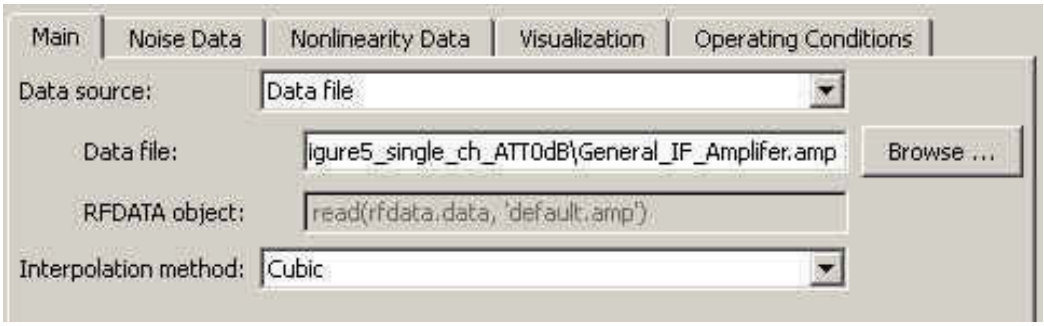

Figure 28 'General_IF_Amplifer' Block Parameters window

4.2.2 Generation of the complex baseband equivalent signal

The light blue blocks in Figure 19 are responsible for providing the 'Front-End' block with a complex baseband equivalent signal. The 'Input Signal' block generates a complex continuous-time signal, whose amplitude is equal to one, and whose frequency is Fin, as defined in Figure 22. In order to control the amplitude of the signal, the slider gain block ('dBm') allows the user to change the signal power in dBm. After the power unit is changed to Watts using the 'dBW to Linear' subsystem (Figure 29), it is applied to the amplitude of the complex sine signal by the Product block.

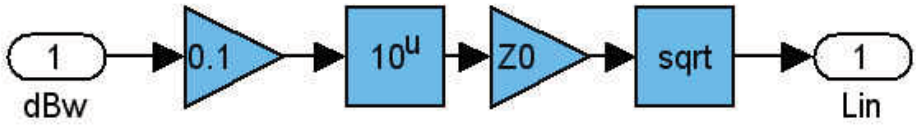

Figure 29 Blocks inside the 'dBW to Linear' subsystem

The formulation represented by the 'dBW to Linear' block is $\sqrt{Z0 \cdot 10^{\frac{Power(dBw)}{10}}}$ $(Watts)$, and Z0 is the characteristic impedance of the system that is defined in Figure 22.

Since the Front-End block processes the data in baseband, the input signal has to be converted to a complex baseband signal by 'Complex Multiply' subsystem in Figure 30. The mathematical function of this subsystem is to perform complex multiplication of two input signals.

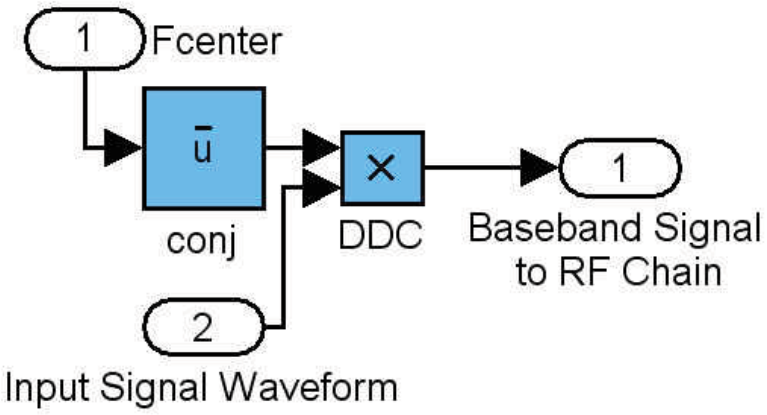

Figure 30 Blocks inside the Complex Multiply subsystem

4.2.3 Other blocks in Simulink model of RF front-end

The Zero-Order Hold block converts the complex-baseband continuous-time signal to a sampled signal, since Simulink RF blocks only process sampled signals. The sample time used in this block must be equal to the sample time of the RF Input Port shown in Figure 21. The sampling frequency can be selected much greater than the Nyquist frequency of the signal of interest in order for the RF blocks to operate on the signal with high precision. For example, in our simulations the sampling frequency is about 50 times higher than the signal frequency. Before the baseband signal sent to the RF Front-End, ambient thermal noise is added. Here the temperature is set at 300^{o}K.

The signal power levels at the input and output ports of the RF chain are monitored by calculating the RMS of the voltage levels and by using FFT Spectrum blocks. The Gain block in front of each FFT Spectrum block calibrates the power to the correct level. The detail is in Figure 31.

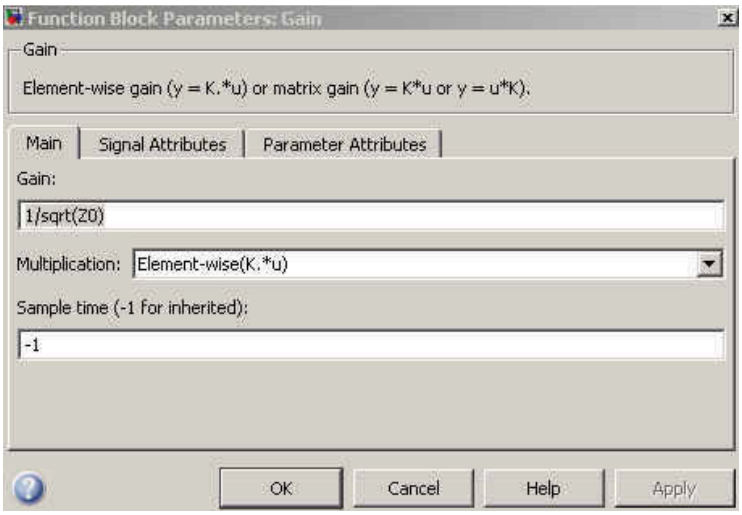

Figure 31 Gain block parameter window

4.3 Section summary

This section discusses how to use the frequency data created in the last section to build the front-end Simulink model for time domain simulations. The important points are:

- To correctly load frequency data corresponding to Simulink RF physical blocks;
- To make sure the Input Port in the RF chain receives an equivalent complex baseband signal which represents the real pass-band signal received by the antenna;
- The sampling time of the complex baseband signal must be equal to the sampling time set in the Input Port;
- The output signal from the Output Port of the RF Front-End chain is a complex baseband signal. It has to be converted back to the equivalent IF pass-band signal before it is digitized by the ADC. This conversion follows the same principle discussed in the sub-section 4.2.2.

5 Simulation results of RF front-end Simulink model

This section presents the simulation results of the RF front-end Simulink model developed in the last section. The simulations include one scenario in which the front-end model is excited by a single-tone, and another in which the input signal is a double-tone. The Simulink model shown in Figure 19 is the single-tone case, and Simulink model for the double-tone case is given in Figure 39. When we present the simulation results, we will also focus on discussing the front-end nonlinearity, such as the 1-dB compression point (P1dB), and 3rd inter-modulation distortion (3rd IMD).

5.1 Characterizing RF receiver performance

5.1.1 RF receiver nonlinear effects

This section gives a quick review of RF receiver system nonlinearities and some definitions [19] that we will use to present the simulation results in the following sections.

The nonlinear effect of the devices in a RF receiver system limits a maximum output power which can be delivered to the load. Above a certain input power, the receiver gain is no longer a constant and power transfer is no longer linear versus the input power. The following figure shows that typical two-port active RF devices have a linear operation range. In the linear operation range the ratio of the output power to the input power is a constant, which is called gain, $G = P_{Out} / P_{In}$. As the input power increases, the gain will decrease when the receiver starts to operate in the nonlinear range. The 1dB gain compression point is defined as the amount of power that causes the gain to drop by 1dB due to the receiver nonlinearity. It can be referenced either to the input or output power, i.e. IP1dB or OP1dB.

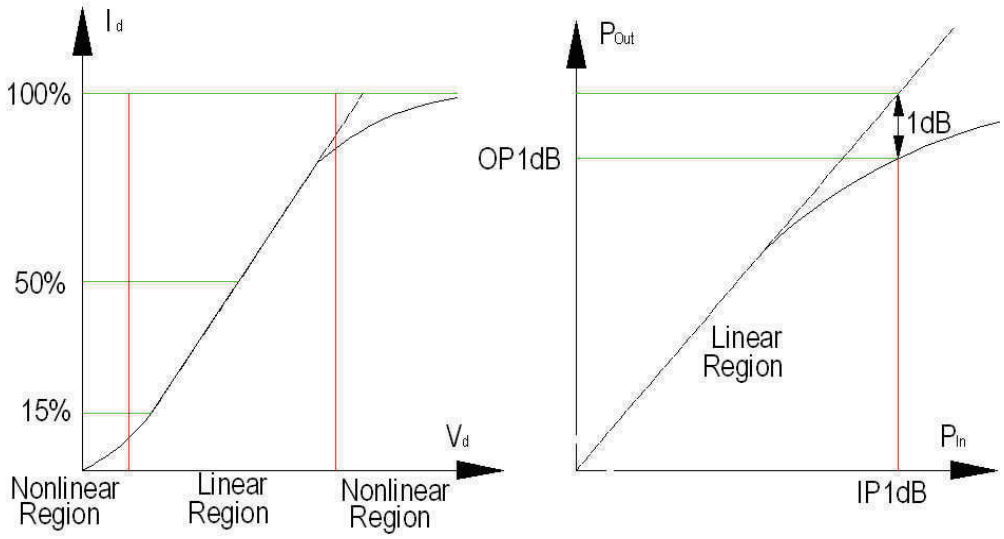

Figure 32 Typical two-port device operating curve and 1dB gain compression point

The relationship between the IP1B and OP1dB can be written as:

$$IP1dB = OP1dB - G + 1dB \tag{1}$$

or

$$OP1dB = IP1dB + G - 1dB \tag{2}$$

When the signal input power is higher than IP1dB, the output signal will contain more frequency components, as shown in Figure 33 a. Intermodulation distortion (IMD) is the product of nonlinearities in the receiver. The 3rd intermodulation products are mixing products of the second harmonic of f_1 with the fundamental f_2 and the second harmonic of f_2 mixed with the fundamental of f_1. It is because the frequencies of the 3rd intermodulation products are very close to the fundamental frequencies, as shown in Figure 33 b, that sometimes it is very difficult to filter them. Changing $f_1$and f_2 input power by 1 dB changes both 3rd intermodulation products by 3dB, which means the receiver output power of the 3rd intermodulation products grows (drops) 3 times faster than the receiver fundamental signals' output power. The third-order intercept point (IP3) is the parameter that defines a receiver's 3rd IMD property. The relationship of the receiver gain curve and IMD properties is illustrated in Figure 34. In the figure, the black line is the receiver gain curve, and the blue line is the 3rd IMD line, whose slope is 3:1. The line for the 2nd IMD product is shown in the figure as well. The intercept point of the black line and blue line is called the receiver third-order intercept point, which is represented by IIP3 or OIP3. Using these curves, one can very easily find the output powers for different products. For example, when the input signal power is P_I, one can find the output power is P_O, and the output power for the 3rd IMD products is P_O(IP3).

The following formulas can be used to calculate P_O(IP3), when one knows the input power,

$$OIP3 = IIP3 + G\,, \tag{3}$$

and

$$P_O(IP3) = 3P_O - 2OIP3\,. \tag{4}$$

The power difference between the fundamental signals and the 3rd IMD products is

$$IP3R = 2(OIP3 - P_O)\,, \tag{5}$$

which is also called the 3rd order IMD rejection ratio.

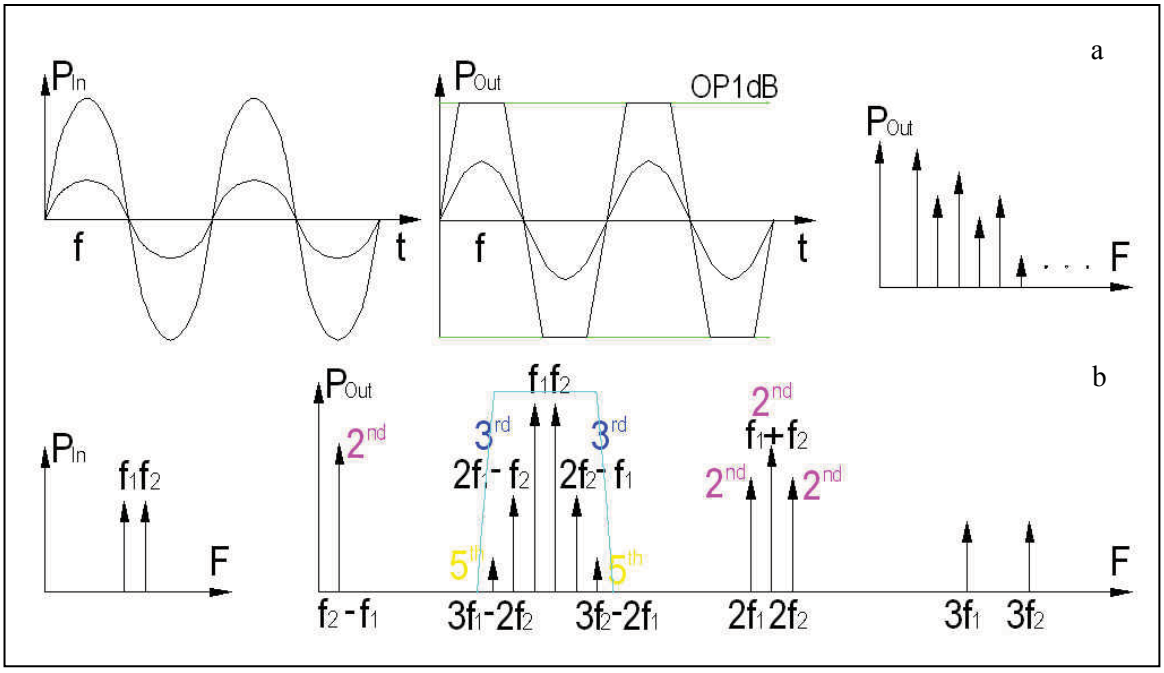

Figure 33 Sources of distortions

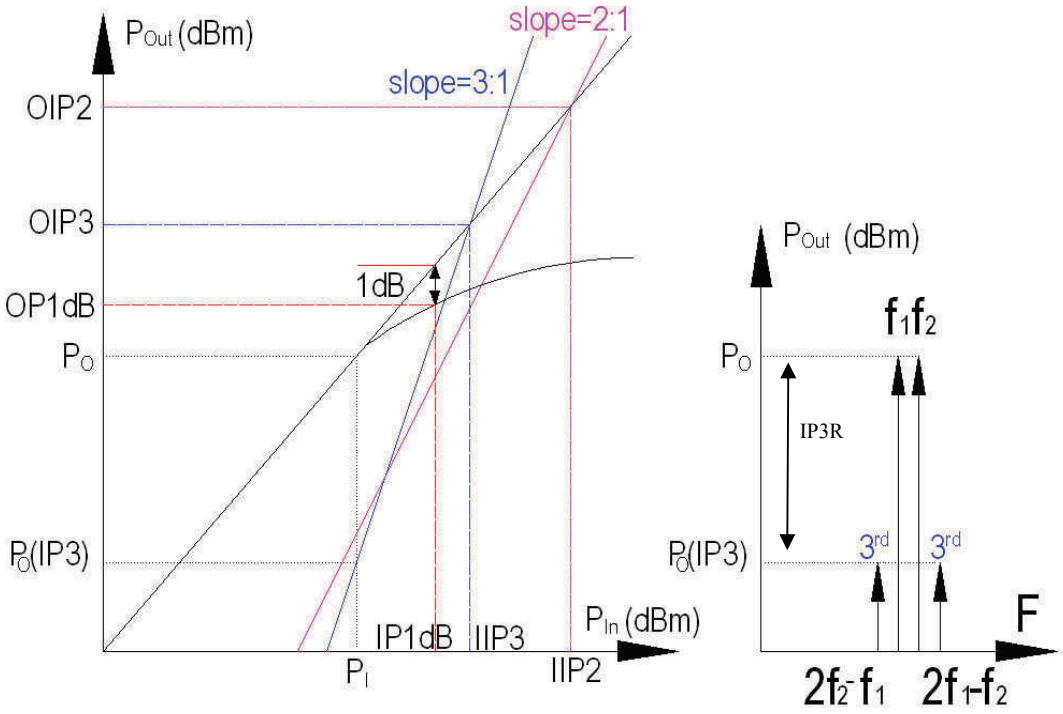

Figure 34 Receiver gain curve and IMD properties

5.1.2 RF receiver noise floor and dynamic range

The receiver noise floor at room temperature 300^0K can be calculated by

$$NoiseFloor(dBm) = -174 + NF + 10\log_{10}(B_R) \tag{6}$$

where NF is the receiver noise figure and B_R is the receiver RF bandwidth in Hz. Based on the receiver noise floor, one can define receiver three dynamic ranges, which are illustrated in Figure 35.

The receiver full scale dynamic range (FSDR) is the difference in power level between receiver's OP1dB point and the receiver output noise floor, and dynamic range (DR) is the difference in power level between receiver's OP1dB point and the noise floor plus $(SNR)_o$ that is required for quality reception. Here $(SNR)_o$ is the receiver output signal to noise ratio. The receiver DR can be estimated using

$$DR = OP1dB + 174 - NF - 10\log_{10}(B_R) - (SNR)_o \tag{7}$$

The receiver spur-free dynamic range (SFDR), which is also shown in Figure 35, is another parameter that is used to quantify the receiver nonlinearity. It is defined as the difference between the fundamental signal power level and the noise power level when the distortion products are equal to the noise power. SFDR is similar to the 3^{Rd} order IMD rejection ratio, when the distortion product power level is the same as the noise floor. The receiver SFDR can be calculated by

$$SFDR(dB) = 2/3(OIP3 + 174 - NF - 10\log_{10}(B_R)). \tag{8}$$

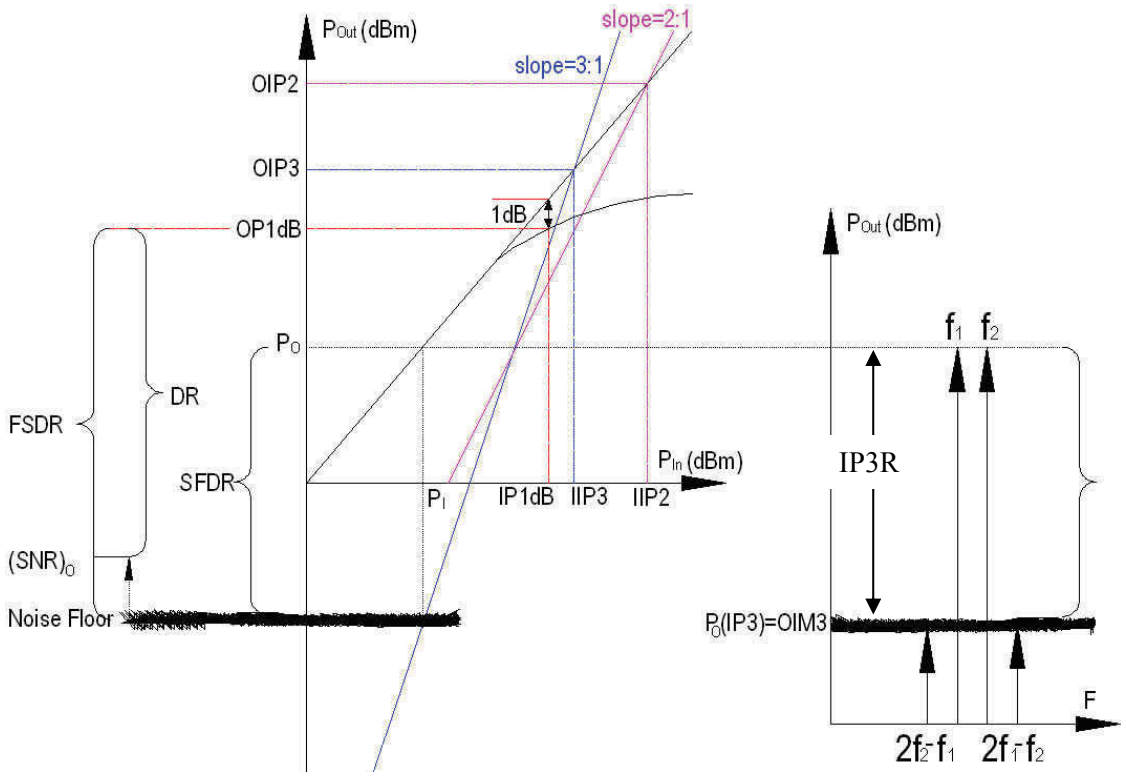

Figure 35 Receiver full-scale dynamic range, dynamic range and spur-free dynamic range

5.2 Case of single-tone sine wave input signal

When the receiver front-end model is excited by a sine-wave signal, its gain and 1-dB compression point (P1dB) can be studied. In the following simulation, the input signal frequency is at 3.11 GHz, which is the center frequency of S-band (BPF1). Figure 36 shows a typical simulation, in which input signal power level is at -60dBm. During the simulation, the 'Input Power in dBm' and the 'Out Power in dBm' blocks in the model measure the input and output power levels, and the difference between these power levels is the front-end gain shown in the 'Total Gain in dB' block. In this case, the gain is 32.90 dB. The data of the output power level versus the input power level are given in Table 4. From the table, we note that the gain of the receiver front-end model is 32.9 dB at the low input power levels. The gain decreases as the input power level increases, and when the input power level reaches about -17 to -16 dBm, the gain has dropped by 1dB. So the front-end's IP1dB (Input P1dB) is about -16.5 dBm.

The input and output signal spectra are displayed in Figure 37 and Figure 38. It is noted that when the input power level approaches the IP1dB compression point, the system output spectrum noise floor increases dramatically. It has to be mentioned that the noise floor calculated from the Simulink model is lower than the expected level. To remedy the problem, the additional noise can be easily added in the model.

.

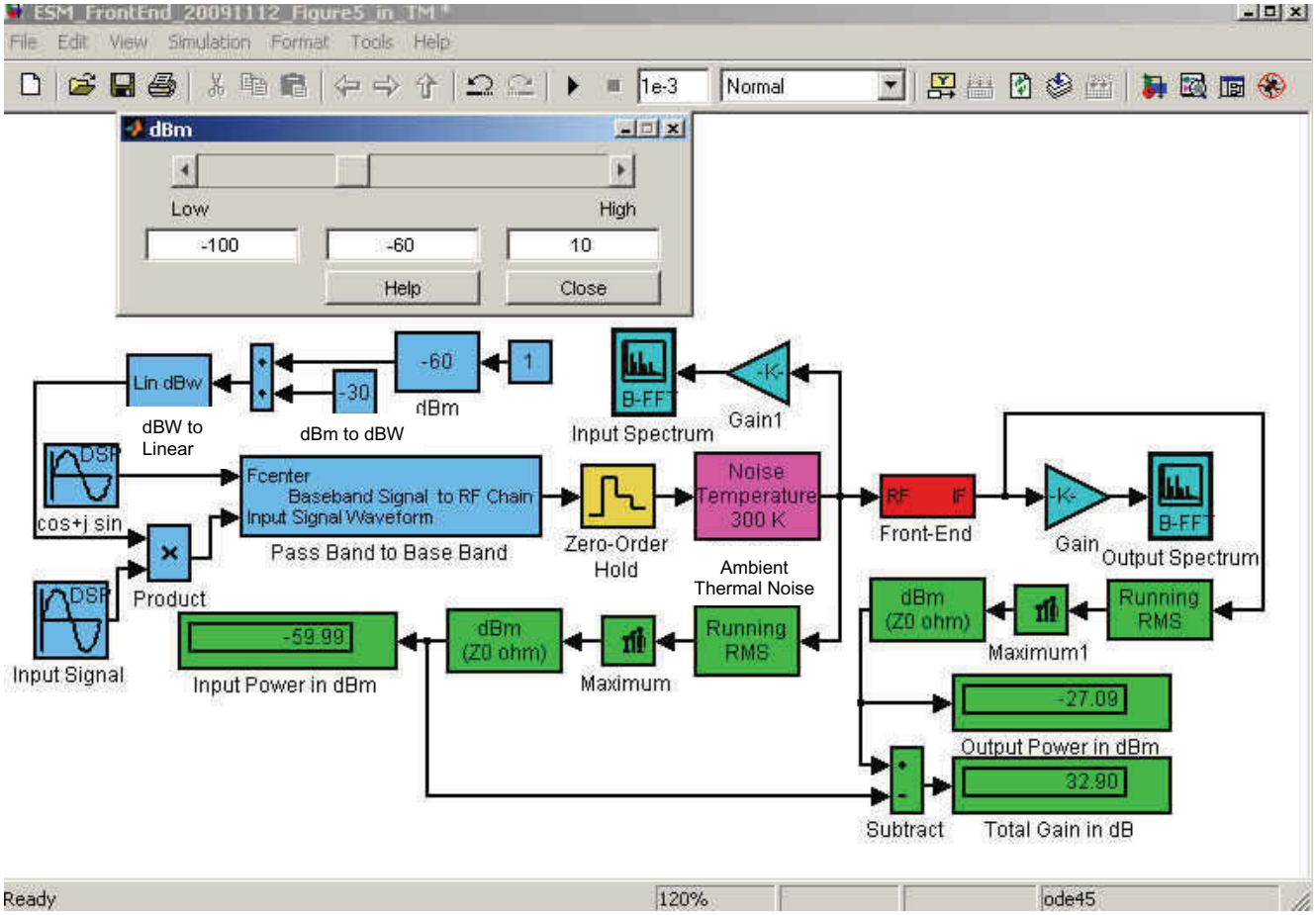

Figure 36 Input signal is at -60dBm power level

Table 4 Receiver front-end model input and output powers

Input Power (dBm)	Output Power (dBm)	Gain (dB)
-70	-37.09	32.09
-60	-27.09	32.90
-50	-17.09	32.94
-40	-7.12	32.89
-30	2.82	32.88
-20	12.42	32.44
-19	13.28	32.30
-18	14.18	32.20
-17	14.94	31.99
-16	15.75	31.75
-15	16.44	31.45
-14	17.15	31.15
-13	17.70	30.70
-12	18.17	30.17
-11	18.53	29.53
-10	18.79	28.79
-9	18.93	27.93
-8	18.98	26.98
-7	18.98	25.98
-6	18.99	24.99

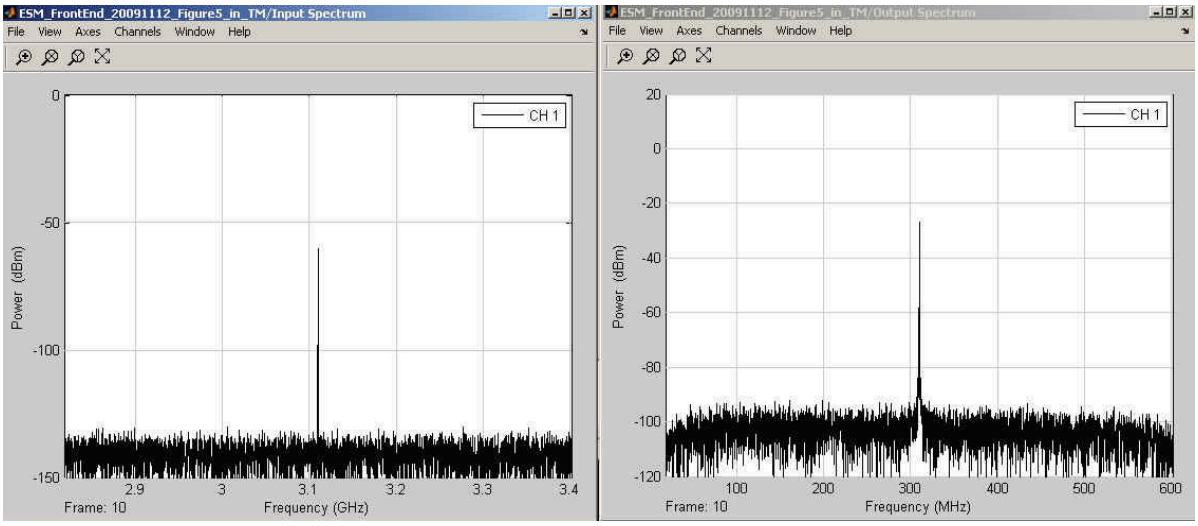

Figure 37 Input and output signal spectrum when input power level is -60 dBm

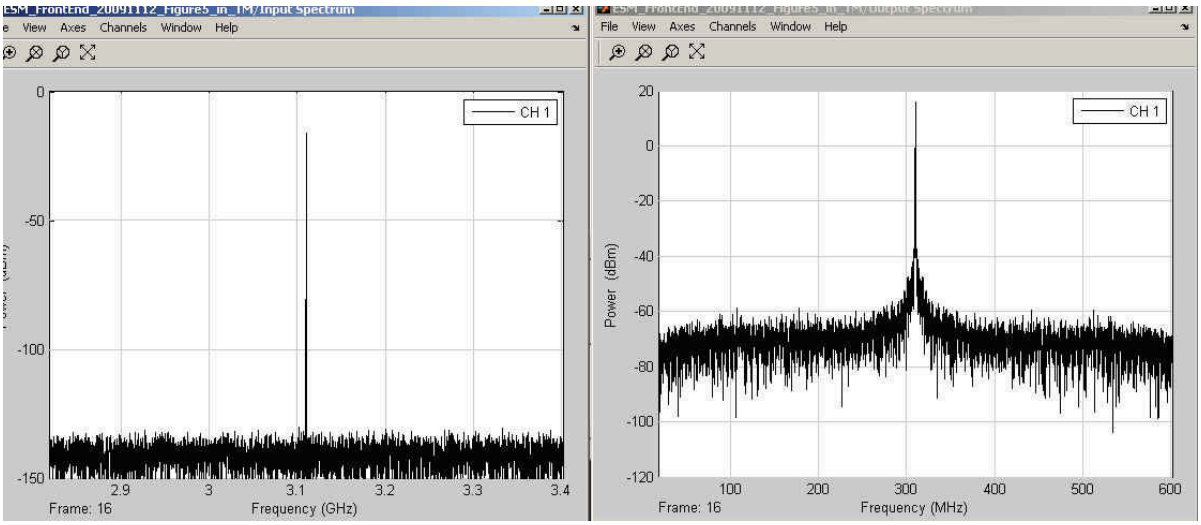

Figure 38 Input and output signal spectrum when input power level is -16 dBm

5.3 Case of two-tone input signal

In this section, two sine-wave signals are sent to the front-end model. One can arbitrarily select the input frequencies in the two-tone signal. The frequencies of the signals in this case are 3105.15625 and 3114.84375 MHz. They are 9.6875 MHz away from each other, and the initial phase difference between these sine waves is zero. To send a two-tone signal to the front-end model, another sine-wave signal is added in the Simulink model in Figure 19 and Figure 36, and the model is displayed in Figure 39. Three blocks are new in the figure compared with the Simulink model in Figure 36. They are 'Input Signal1', 'Product1' and 'Add1'. One can find that the input power level control blocks control the two sine-wave power levels through 'Product' and 'Product1' blocks. When these two tones are added together by the 'Add1' block, the signal is changed to complex baseband equivalent signal. The green blocks still measure the signal power levels at the input and output ports of the front-end model. Figure 40 shows the output power spectrum of the front-end model when it is excited by the two-tone signal with input power level of -40.93 dBm. Two 3rd IMD components on each side of the fundamental tones are clearly shown in the figure, and the power level of 3rd IMD components is about -75 dBm.

Table 5 lists the calculated 3rd IMD power level when the two-tone signal at different input power levels. It can be found that the receiver 3rd IMD output power increases about 3 dB, when the fundamental tones' input power increases 1 dB.

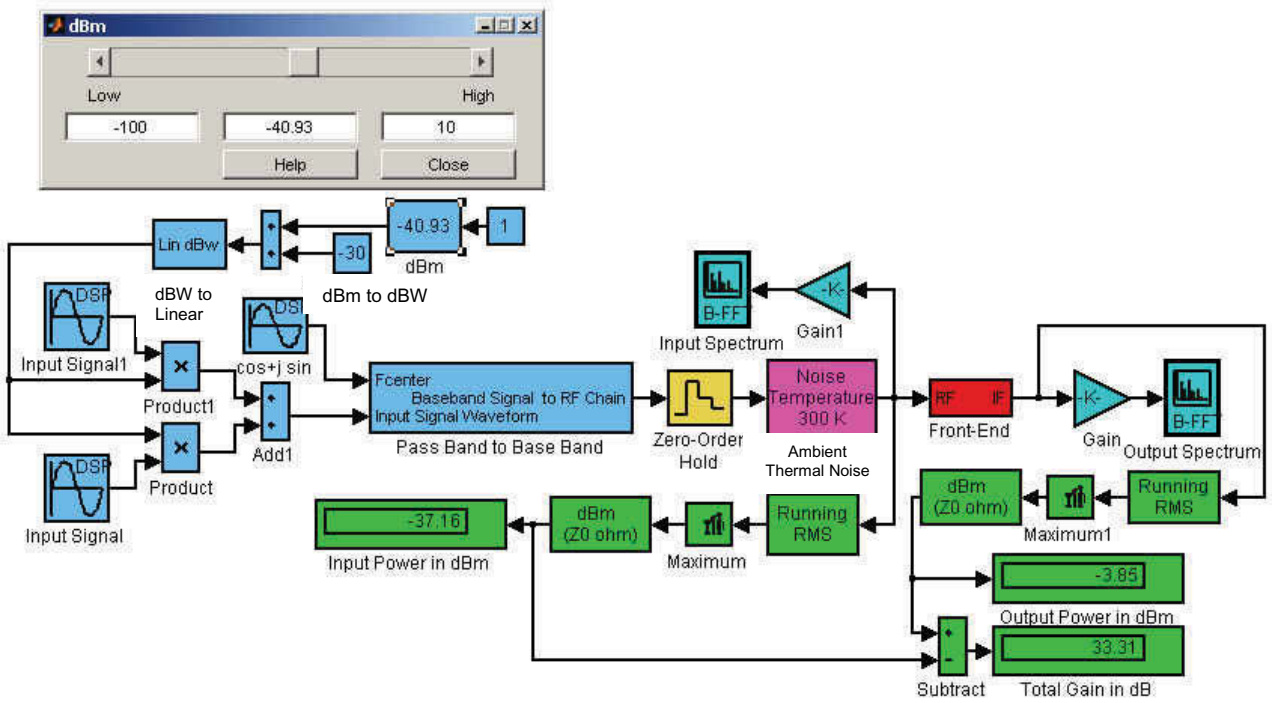

Figure 39 Two-tone signal exciting the front-end model and input power is -40.93 dBm for each tone

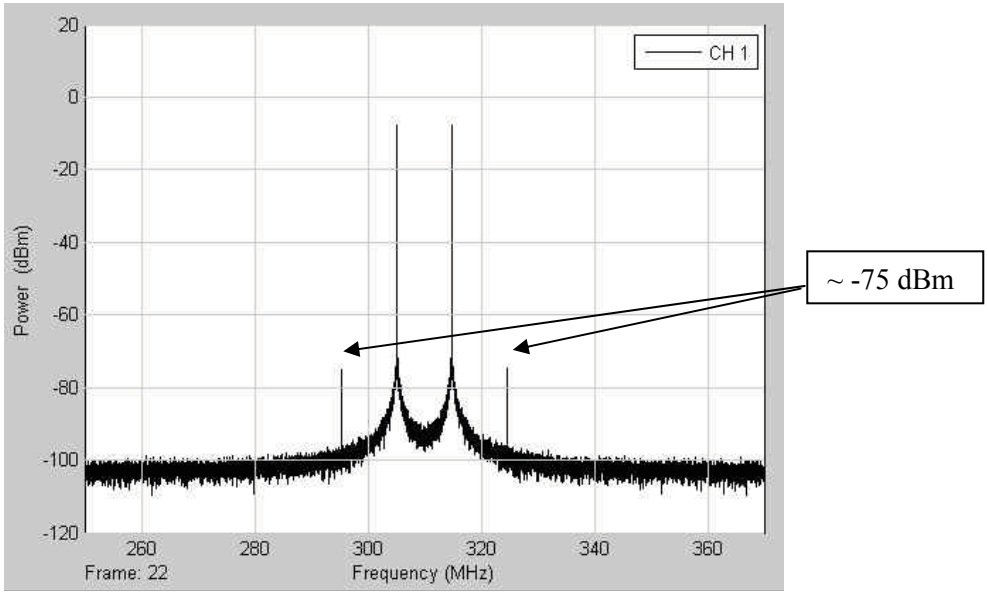

Figure 40 Output power spectrum of the front-end model when it is excited by a two-tone signal with input power level at -40.93 dBm

Table 5 3rd intermodulation distortions at different two-tone power levels

Input Power (dBm)	Input Power Increase (dBm)	3rd IMD Power Level (dBm)	3rd IMD Power Level Increase (dBm)	Increase in 3th IMD when input power increases by 1dB (dBm)
-45	N/A	-87.33	N/A	N/A
-40	5	-72.1	15.23	3.05
-35	5	-57.15	14.95	2.99
-30	5	-43.07	14.08	2.82
-25	5	-28.06	15.01	3.00
-20	5	-13.80	14.26	2.85

5.4 Performance of the front-end Simulink model

In the previous two sections, we presented some simulation results using the front-end Simulink model when it is excited by a single-tone signal and a double-tone signal, respectively. In this section we will discuss the front-end model performance based on the receiver characteristics discussed in section 5.1.

The receiver noise figure calculated from the cascaded RF circuit in Figure 20 is about 6 dB. By taking into account the noise contributed by the LO phase noise, which Simulink cannot model now, the total estimated noise figure of the front-end is about 12 dB [17]. The RF bandwidth is about 500 MHz, which is defined by the BPF1 shown in Figure 15. Using Eq. (6) the noise floor of the receiver is estimated at about -75 dBm.

Figure 41 plots the simulation data given in Table 4 and Table 5. The two solid lines are the linear functions obtained by using the Matlab curve-fitting function. The intercept point of these curves is at (-6.62, 25.74). So the IIP3 and OIP3 of the receiver front-end model are -6.62 dBm and 25.74 dBm, respectively. From Figure 41, we can find that:

1. The IP1dB and OP1dB of the receiver front-end model are -16.5 dBm and 15.96 dBm. They are about 10 dB lower than the values of IIP3 (= -6.62 dBm) and OIP3 (= 25.74 dBm), respectively.
2. Using OP1dB and noise floor, the FSDR is 90.96 dB.
3. The IMD power level will be the same as the noise floor, when a double-tone signal input power in -40.93 dBm, so that SFDR is 66.78 dB (= -8.14 dBm – (-75 dBm)). Using Eq. (8), the calculated SFDR is equal to 67.17 dB.
4. If we assume that the application required minimum $(SNR)_o$ is 10 dB, then the DR of the front-end model is 80.96 dB (= 15.96 dBm – (-75 dBm + 10 dB)).

All the receiver front-end model performance results are summarized in the following table.

Table 6 Performance data of the receiver front-end model read from Figure 41

Name of the characteristic in Figure 35	Read from Figure 41
IP1dB	-16.5 dBm
OP1dB	15.96 dBm
IIP3	6.62 dBm
OIP3	25.74 dBm
P_I	-40.93 dBm
P_O	-8.14 dBm
FSDR	90.96 dB
DR	80.96 dB
SFDR	66.86 dB

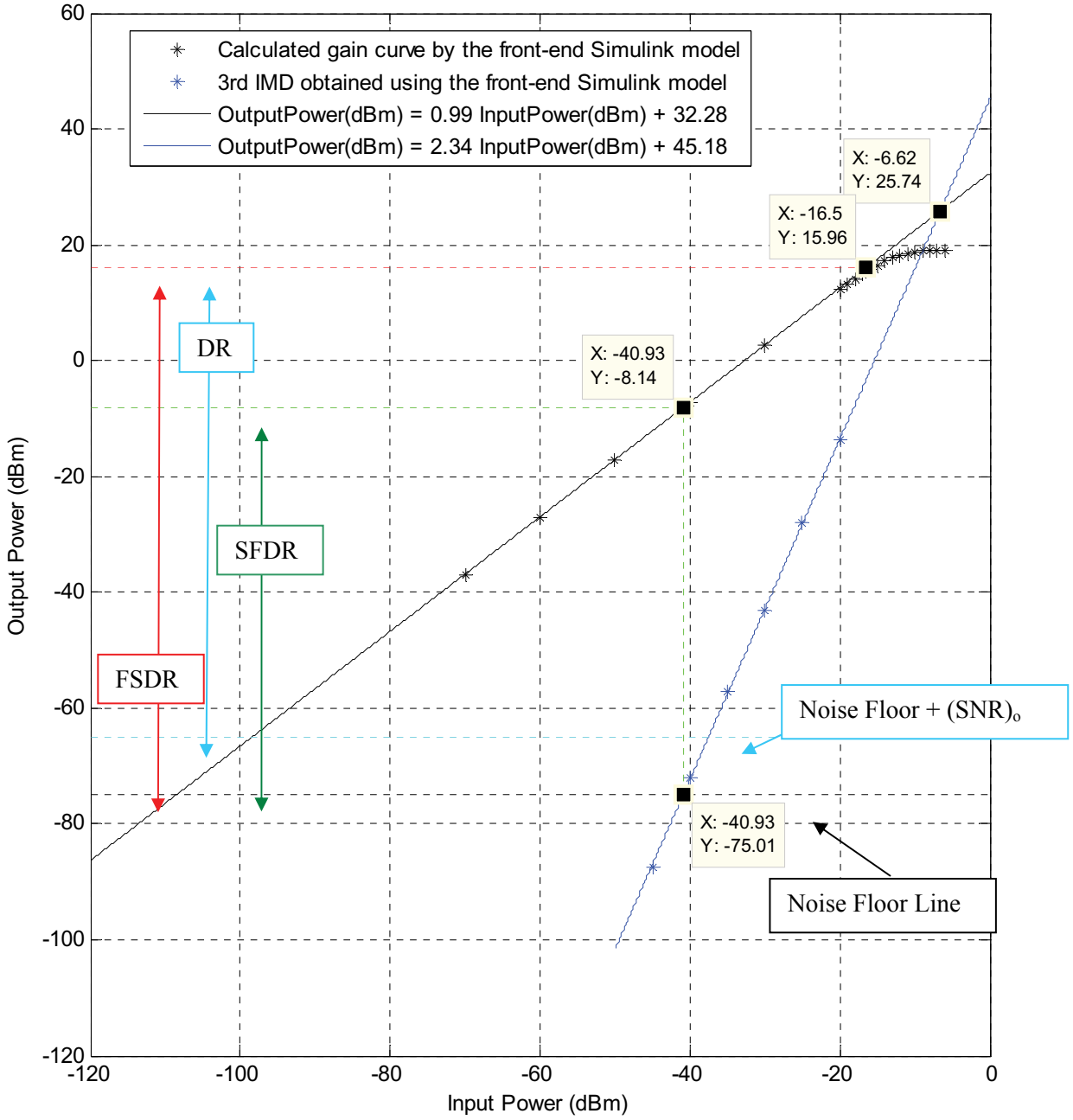

Figure 41 Front-end Simulink model gain and the 3rd IMD curves

6 Conclusion and future work

A high-fidelity model based on an S-band ESM receiver front-end designed using MIC and MMIC technologies has been developed. The details on how to implement the model using Matlab/Simulink, and its simulated results are presented in this report. Since Simulink is used for the final implementation of the model, the model can be used for dynamic system simulations. For example, it can be used as a module to use in to ESM scenarios developed in the STK environment. Using Matlab and Simulink, a high-fidelity analog-to-digital converter model is being developed. The ESM receiver front-end model will connect to the ADC model to form a digital ESM system. The complete ESM receiver model will be reported in a separate document.

References

[1] John, I. and O'Brien, M., "Using Matlab and Simulink in electronic warfare applications", Workshop at Defence R&D Canada Ottawa, private communications, April 28, 2009.

[2] Moore, M. S., Brooks, M, Willden, G. and Neema, S., "Key to success in model-based design", Southwest Research Institute and Institute Software Integrated, October, 12, 2004, http://www.omg.org/news/meetings/workshops/MIC_2004_Manual/01-1_Moore_etal.pdf, Dec. 2009.

[3] Wu, C., Du, H. J., Lee, J. P. Y., Campbell, S. and Renaud, E., "Electronic support payload development for small unmanned aerial vehicles: Part 1: hardware development, integration and installation on Cessna 182", Defence R&D Canada Ottawa, DRDC Ottawa TM 2006-268, December 2006.

[4] Poliakov, E., Wu, C. and Lee, J. P. Y., "Use of Matlab for UAV electronic support payload software implementation and system integration – Part I," Defence R&D Canada Ottawa, DRDC Ottawa TN 2008-101, May 2008, Internal Publication.

[5] Poliakov, E., Wu, C. and Lee, J. P. Y., "Use of Matlab for UAV electronic support payload software implementation and system integration – Part II," Defence R&D Canada Ottawa, DRDC Ottawa TN 2008-109, May 2008, Internal Publication.

[6] Poliakov, E., Wu, C. and Lee, J. P. Y., "Use of Matlab for UAV electronic support payload software implementation and system integration – Part III," Defence R&D Canada Ottawa, DRDC Ottawa TN 2008-138, June 2008, Internal Publication.

[7] Poliakov, E., Wu, C. and Lee, J. P. Y., "Use of Matlab for UAV electronic support payload software implementation and system integration – Part IV," Defence R&D Canada Ottawa, DRDC Ottawa TN 2008-173, September 2008, Internal Publication.

[8] The Mathworks, "Matlab/Simulink," http://www.mathworks.com/, February 2010.

[9] Analog Graphics Inc. "STK version 9," http://www.stk.com/, February 2010.

[10] Wu, C., "Electronic Support Payload for UAV," DRDC Ottawa Seminar, May 31, 2007, private communication.

[11] Wu, C., Du, H. J. and Lee, J. P. Y., "Canadian Report - Lakehurst Trial on ES Payload and Operational Concepts for UAVs, 11-29 September 2006," The Technical Cooperation Program (TTCP) Electronic Warfare Systems (EWS), Technical Panel 4 (TP4), Task 9, September 2007.

[12] Wu, C. and Lee, J.P.Y., "CSI ESM Naval Littoral Scenario Version 2", submitted to PMO HCM/FELEX, DMCS # 5578, file # 2900-50, July 22, 2009, 2 pages with data files.

[13] BAE Australia, "SADM – ship air defence model", Manual, 2009, 677 Victoria Street, Abbotsford, Victoria 3067, Australia.

[14] Hittite Microwave Corporation, http://www.hittite.com/, February 2010.

[15] Marki Microwave, 'M2-0218 mixer,' http://www.markimicrowave.com/, February 2010.

[16] GEIA/IBIS Open Forum, "Touchstone® 2.0 file format specification", Rev. 1.0.

[17] Erst, S. J., “Receiving System Design,” Artech House, 1984.

[18] Wu, C., and Lee, J., “Bandpass Filter Design Used in Electronic Support Payload for Small Unmanned Aerial Vehicles”, Defence R&D Canada Ottawa, DRDC Ottawa TM 2005-116, August 2005.

[19] Majewski, J. J., “Digital receiver system design”, Workshop Notes, 2009 International Microwave Symposium, June 8, 2009.

Annex A Matlab code for General_RF_Amplifer.amp

09/11/09 4:41 PM F:\2...\ESM FrentEnd 20091030 Figure5 single ch ATT0dB in TM.m 1 of 9

```
clear, close all
mil_to_m=0.001*25.39999999999999e-3;
freq=(2:0.01:20)*1.e9;
Z0=50;
%%%%%%%%%%%%%%%%%%%%
% Create RF objets %
%%%%%%%%%%%%%%%%%%%%
%*************************************************************************
% MIcrostrip lines
    %Create microstrip line template
    Microstrip = rfckt.microstrip(...
            'Width'      , 4.571999998e-4, ...
            'Height'     , 2.539999999e-4, ...
            'Thickness'  , 1.777999999e-5, ...
            'EpsilonR'   , 4.500000000e+0, ...
            'LossTangent', 0.001000000e+0, ...
            'SigmaCond'  , 5.813000000e+7, ...
            'LineLength' , 0.01, ...
            'StubMode'   , 'None'    , ...
            'Termination', 'None');    %define substrate

     %The 1st microstrip line length=941mil ------------------------------
    L941 = 941; %microstrip line length
    iL941 = 0.3; %extra insertion loss for microstrop941mil

         mp = copy(Microstrip);                %create a new line
     set(mp,'LineLength',L941*mil_to_m);        %change length
         a = analyze(mp,freq);                 %calcuate s-par od M-line
         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
         inserLoss = -iL941; %dB Extra inserLoss for connections between
                             %M-line and other devices, especially there is a
                             %SMA is connected at one end of the M-line
         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
                                      %inersion loss an NF in dB
         ip3(:)= inf;        %IP3 for M-Line assumed inf
         spdata  = rfdata.network('Freq',freq,'Data',mp_sp);
         nfdata  = rfdata.nf('Freq',freq,'Data',nf');
         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
     %using Amp-model to represent M-line in order to use
     %the inerstion loss of M-line as NF, when system NF is calculted
     Microstrip941mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);

    %The 2ed microstrip line length=230mil-----------------------------------
    L230 = 230;
    iL230 = 0.1;
         mp = copy(Microstrip);                %create a new line
```

```
52      set(mp,'LineLength',L230*mil_to_m);         %change length
53          a = analyze(mp,freq);                  %calcuate s-par od M-line
54          mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
55          inserLoss = -iL230; %dB Extra inserLoss for connections between
56                              %M-line and other device, especially there is a
57                              %SMA is connected at one end of the M-line
58          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
59          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
60          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
61         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
62         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
63                                        %inersion loss an NF in dB
64          ip3(:)=inf;
65          spdata=rfdata.network('Freq',freq,'Data',mp_sp);
66          nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
67          ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
68      %using Amp-model to represent M-line in order to take into account
69      %the inerstion loss of M-line as NF,when system NF is calculted
70      Microstrip230mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
71          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
72
73     %The 3rd microstrip line length=403mil---------------------------------
74     L403 = 403;
75     iL403 = 0.1;
76          mp = copy(Microstrip);                 %create a new line
77      set(mp,'LineLength',L403*mil_to_m);         %change length
78          a = analyze(mp,freq);                  %calcuate s-par od M-line
79          mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
80          inserLoss = -iL403; %dB Extra inserLoss for connections between
81                              %M-line and other device, especially there is a
82                              %SMA is connected at one end of the M-line
83          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
84          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
85          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
86         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
87         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
88                                        %inersion loss an NF in dB
89          ip3(:)=inf;
90          spdata=rfdata.network('Freq',freq,'Data',mp_sp);
91          nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
92          ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
93      %using Amp-model to represent M-line in order to take into account
94      %the inerstion loss of M-line as NF,when system NF is calculted
95      Microstrip403mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
96          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
97
98     %The 4th microstrip line length=460mil ---------------------------------
99     L460 = 460;
100    iL460 = 0.1;
101         mp = copy(Microstrip);                 %create a new line
102     set(mp,'LineLength',L460*mil_to_m);         %change length
```

```
103         a = analyze(mp,freq);                  %calcuate s-par od M-line
104         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
105         inserLoss = -iL460; %dB Extra inserLoss for connections between
106                            %M-line and other device, especially there is a
107                            %SMA is connected at one end of the M-line
108         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
109         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
110         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
111        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
112        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
113                                    %inersion loss an NF in dB
114         ip3(:)=inf;
115        spdata=rfdata.network('Freq',freq,'Data',mp_sp);
116        nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
117        ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
118     %using Amp-model to represent M-line in order to take into account
119     %the inerstion loss of M-line as NF,when system NF is calculted
120     Microstrip460mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
121         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
122
123    %The 5th microstrip line length=213mil --------------------------------
124    L213 = 213;
125    iL213 = 0.1;
126         mp = copy(Microstrip);                 %create a new line
127     set(mp,'LineLength',L213*mil_to_m);         %change length
128         a = analyze(mp,freq);                  %calcuate s-par od M-line
129         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
130         inserLoss = -iL213; %dB Extra inserLoss for connections between
131                            %M-line and other device, especially there is a
132                            %SMA is connected at one end of the M-line
133         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
134         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
135         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
136        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
137        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
138                                    %inersion loss an NF in dB
139         ip3(:)=inf;
140        spdata=rfdata.network('Freq',freq,'Data',mp_sp);
141        nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
142        ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
143     %using Amp-model to represent M-line in order to take into account
144     %the inerstion loss of M-line as NF,when system NF is calculted
145     Microstrip213mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
146         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
147
148    %The 6th microstrip line length=444mil --------------------------------
149    L444 = 444;
150    iL444 = 0.3;
151         mp = copy(Microstrip);                 %create a new line
152     set(mp,'LineLength',L444*mil_to_m);         %change length
153         a = analyze(mp,freq);                  %calcuate s-par od M-line
```

```
154         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
155         inserLoss = -iL444; %dB Extra inserLoss for connections between
156                             %M-line and other device, especially there is a
157                             %SMA is connected at one end of the M-line
158         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
159         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
160         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
161        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
162        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
163                                        %inersion loss an NF in dB
164        ip3(:)=inf;
165        spdata=rfdata.network('Freq',freq,'Data',mp_sp);
166        nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
167        ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
168     %using Amp-model to represent M-line in order to take into account
169     %the inerstion loss of M-line as NF,when system NF is calculted
170     Microstrip444mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
171         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
172
173         %The 7th microstrip line length=360mil --------------------------------
174      L360 = 360;
175     iL360 = 0.3;
176         mp = copy(Microstrip);                  %create a new line
177      set(mp,'LineLength',L360*mil_to_m);         %change length
178         a = analyze(mp,freq);                   %calcuate s-par od M-line
179         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
180         inserLoss = -iL360; %dB Extra inserLoss for connections between
181                             %M-line and other device, especially there is a
182                             %SMA is connected at one end of the M-line
183         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
184         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
185         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
186        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
187        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
188                                        %inersion loss an NF in dB
189        ip3(:)=inf;
190        spdata=rfdata.network('Freq',freq,'Data',mp_sp);
191        nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
192        ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
193     %using Amp-model to represent M-line in order to take into account
194     %the inerstion loss of M-line as NF,when system NF is calculted
195     Microstrip360mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
196         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
197
198         %The 8th microstrip line length=254mil --------------------------------
199      L254 = 254;
200     iL254 = 0.1;
201         mp = copy(Microstrip);                  %create a new line
202      set(mp,'LineLength',L254*mil_to_m);         %change length
203         a = analyze(mp,freq);                   %calcuate s-par od M-line
204         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
```

```
205          inserLoss = -iL254; %dB Extra inserLoss for connections between
206                              %M-line and other device, especially there is a
207                              %SMA is connected at one end of the M-line
208          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
209          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
210          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
211         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
212         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
213                                        %inersion loss an NF in dB
214         ip3(:)=inf;
215         spdata=rfdata.network('Freq',freq,'Data',mp_sp);
216         nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
217         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
218      %using Amp-model to represent M-line in order to take into account
219      %the inerstion loss of M-line as NF,when system NF is calculted
220      Microstrip254mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
221          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
222
223 %*************************************************************************
224     %Create Amplifiers ****************************************************
225     famp1 = [2.00 12.00 13.00 22];
226     NFamp1 = [1.60  2.20  2.3  2.85 ];
227     ip3amp1 = [35   34    32   31];
228     LNA_HMC_ALH482_1 = read(rfckt.amplifier,'LNA_HMC-ALH482_housing.s2p');
229         f = famp1*1.e9;
230         nf=NFamp1; %NF from device data sheet
231         nfdata = rfdata.nf('Freq',f,'Data',nf);
232         ip3=ip3amp1; %OIP3 data from device data sheet
233         ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',ip3);
234         set (LNA_HMC_ALH482_1,'IntpType','Cubic','NoiseData',nfdata,...
235             'NonlinearData',ip3data);
236         clear f nf nfdata ip3 ip3data
237
238     famp2 = [2.00 11.00 14.00 22];
239     NFamp2 = [1.70 2.10  2.3   2.9 ];
240     ip3amp2 =[35.5 34    32   31.5];
241     LNA_HMC_ALH482_2 = read(rfckt.amplifier,'LNA_HMC-ALH482_housing.s2p');
242         f = famp2*1.e9;
243         nf=NFamp2; %NF from device data sheet, with a little change
244         nfdata = rfdata.nf('Freq',f,'Data',nf);
245         ip3=ip3amp2; %OIP3 data from device data sheet, with a little change
246         ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',ip3);
247         set (LNA_HMC_ALH482_2,'IntpType','Cubic','NoiseData',nfdata,...
248             'NonlinearData',ip3data);
249         clear f nf nfdata ip3 ip3data i
250
251 %*************************************************************************
252     %Create Digital Control Attenuator using rfckt.amplifier class
253     %since the class can take into accout noise and nonliearity************
254
255     % Digital Att at 0.0dB ----------------------------------------------
```

```
256      Att = read(rfckt.amplifier,'ATT_HMC424_00p0dB_state_deembedded.s2p');
257         f=Att.NetworkData.Freq;
258         nf=-20*log10(abs(squeeze(Att.NetworkData.Data(2,1,:)))); %using
259                                                    %insertion loss as NF
260         nfdata = rfdata.nf('Freq',f,'Data',nf);
261          f1 = [4 7]*1.e9;
262         ip3 = [49.35 46.1];
263         ip3data = rfdata.ip3('Type','IIP3','Freq',f1,'Data',ip3);
264      set(Att,'IntpType','Cubic','NoiseData',nfdata,'NonlinearData',ip3data)
265         ATT_HMC42400p0dB=copy(Att);
266         clear f nf nfdata ip3 ip3data Att i
267
268 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
269 %Create a passive component using HMC242 SPDT-switch measured s-parmeter
270 %rfc is the port on one side of the switch, and
271 %rf1, rf2 are two ports on other side of the switch.
272 %figure
273 %S-parameter of the SW1 or SW2 is turned on rf1 port
274 SW_on_rf1=read(rfckt.passive,'SW_HMC347_rf1_selected_deembedded.s3p');
275 %S-Parameter of the SW1 or SW2 is turned on rf2 port
276 SW_on_rf2=read(rfckt.passive,'SW_HMC347_rf2_selected_deembedded.s3p');
277 %S-Parameter of the SW3 is turned on rf2 port
278 SW3_on_rf2=read(rfckt.passive,'SW_hmc5471p3_sparam_rf2_selected.s3p');
279
280 %In order to cscade the switch to RFGainChain, we have to change the port
281 %order (see help of 'cascadesparams' and 'snp2smp')
282 analyze(SW_on_rf1,freq);
283 analyze(SW_on_rf2,freq);
284 analyze(SW3_on_rf2,freq);
285 %-------------------------------------------------------------------------
286 %convert s3p to s2p for SW1
287     %A 2-port S-par when the SW1 is on rf2
288     s2p_rf2_to_rfc = snp2smp(SW_on_rf2.AnalyzedResult.S_Parameters,...
289                           Z0, [3 1], Z0); %input from rf2 output to rfc
290
291 % built an 'amplifier' using the 2-port S-par when the SW1 is on rf2
292         nf=-20*log10(abs(squeeze(s2p_rf2_to_rfc(2,1,:))));
293         nfdata = rfdata.nf('Freq',freq,'Data',nf);%using switch
294                                                  %insertion loss as NF in dB
295         ip3=ones(1,length(freq))*45; %IIP3 from device data sheet
296         ip3data = rfdata.ip3('Type','IIP3','Freq',freq,'Data',ip3);
297         netdata= rfdata.network('Freq',freq,'Data',s2p_rf2_to_rfc);
298      SW1_rf2_to_rfc = rfckt.amplifier('IntpType','Cubic','NetworkData'...
299           ,netdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
300 %-------------------------------------------------------------------------
301 %convert s3p to s2p for SW2
302     %A 2-port S-par when the SW2 is on rf1
303     s2p_rfc_to_rf1 = snp2smp(SW_on_rf1.AnalyzedResult.S_Parameters,...
304                            Z0, [1 2], Z0); %input from rfc output to rf1
305 %built an 'amplifier' using the 2-port S-par when the SW2 is on rf1
306         nf=-20*log10(abs(squeeze(s2p_rfc_to_rf1(2,1,:))));
```

```
307         nfdata = rfdata.nf('Freq',freq,'Data',nf); %using switch
308                                                    %insertion loss as NF in dB
309         ip3=ones(1,length(freq))*44; %from data sheet with a little change
310         ip3data = rfdata.ip3('Type','IIP3','Freq',freq,'Data',ip3);
311         netdata= rfdata.network('Freq',freq,'Data',s2p_rfc_to_rf1);
312      SW2_rfc_to_rf1 = rfckt.amplifier('IntpType','Cubic','NetworkData'...
313          ,netdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
314 %-------------------------------------------------------------------------
315 %convert s3p to s2p for SW3
316     %A 2-port S-par when the switch is on rf1
317     s2p_rf2_to_rfc = snp2smp(SW3_on_rf2.AnalyzedResult.S_Parameters,...
318                              Z0, [3 1], Z0); %input from rf1 output to rfc
319 %built an 'amplifier' using the 2-port S-par when the SW3 is on rf1
320         nf=-20*log10(abs(squeeze(s2p_rf2_to_rfc(2,1,:))));
321         nfdata = rfdata.nf('Freq',freq,'Data',nf); %using switch
322                                                    %insertion loss as NF in dB
323         ip3=ones(1,length(freq))*46; %from data sheet with a little change
324         ip3data = rfdata.ip3('Type','IIP3','Freq',freq,'Data',ip3);
325         netdata= rfdata.network('Freq',freq,'Data',s2p_rf2_to_rfc);
326      SW3_rf2_to_rfc = rfckt.amplifier('IntpType','Cubic','NetworkData'...
327          ,netdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
328
329 %*************************************************************************
330     %Create coax line 1500 mil long
331     iL1500 = 0.50; %dB which include both side connector loss
332     coax = rfckt.coaxial( ...
333            'LineLength', 1500 * mil_to_m, ...
334            'OuterRadius',  60 * mil_to_m, ...
335            'InnerRadius',  18 * mil_to_m, ...
336            'MuR', 1                     , ...
337            'EpsilonR', 2.1000           , ...
338            'LossTangent', 0.0005        , ...
339            'SigmaCond', 5.813e7);
340
341          a = analyze(coax,freq);              %calcuate s-par od M-line
342          coax_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
343          inserLoss = -iL1500; %dB Extra inserLoss for connections between
344                               %coax and other devices, especially there is a
345                               %SMMP is connected at one end of the M-line
346          coax_sp(1,2,:)= coax_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
347          coax_sp(2,1,:)= coax_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
348          nf = zeros(length(coax_sp(2,1,:)),1); %create a noise vector
349         ip3 = zeros(length(coax_sp(2,1,:)),1); %create a IP3 vector
350         nf = abs(20*log10(abs(squeeze(coax_sp(2,1,:))))); %using coax-line
351                                          %inersion loss an NF in dB
352         ip3(:)= inf;        %IP3 for M-Line assumed inf
353         spdata  = rfdata.network('Freq',freq,'Data',coax_sp);
354         nfdata  = rfdata.nf('Freq',freq,'Data',nf');
355         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
356      %using Amp-model to represent coax-line in order to use
357      %the inerstion loss of coax-line as NF, when system NF is calculted
```

```
358      Cable1500mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
359          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
360 %*************************************************************************
361 freqs=(2.:0.01:4.)*1.e9;
362 %S_Band BPF1
363     A = read(rfckt.amplifier,'S_band_BPF_1.s2p');
364         analyze(A,freqs);
365         nf=-20*log10(abs(squeeze(A.AnalyzedResult.S_parameter(2,1,:))));
366         ip3 = ones(1,length(freqs)); %create a IP3 vector
367         ip3(:)=inf;
368          nfdata = rfdata.nf (              'Freq',freqs,'Data',nf);
369         ip3data = rfdata.ip3('Type','OIP3','Freq',freqs,'Data',ip3');
370         netdata = rfdata.network(         'Freq',freqs,...
371             'Data',A.AnalyzedResult.S_parameter);
372
373        S_BPF1= rfckt.amplifier('IntpType','Cubic',...
374               'NetworkData',  netdata,...
375               'NoiseData',     nfdata,...
376               'NonlinearData',ip3data);
377         clear nf nfdata ip3 ip3data netdata A i ans s11 s21 AX H1 H2
378 %*************************************************************************
379 % Cascading a network before BPF1++++++++++++++++++++++++++++++++++++++++++
380     A= rfckt.cascade('Ckts',...
381              {Microstrip941mil, ...
382                 SW1_rf2_to_rfc,  ...
383                Microstrip230mil, ...
384                 LNA_HMC_ALH482_1, ...
385                Microstrip403mil, ...
386                 ATT_HMC42400p0dB, ...
387                Microstrip460mil, ...
388                 LNA_HMC_ALH482_2, ...
389                Microstrip213mil, ...
390                 SW2_rfc_to_rf1,  ...
391                Microstrip444mil, ...
392                 Cable1500mil});
393     analyze(A,freq);
394     f= A.AnalyzedResult.Freq;
395     s= A.AnalyzedResult.S_parameter;
396     n= A.AnalyzedResult.NF;
397     p= A.AnalyzedResult.OIP3;
398     netdata = rfdata.network(          'Freq',f,'Data',s);
399     nfdata  = rfdata.nf(               'Freq',f,'Data',n);
400     ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',p);
401     NetworkBeforeBPF1=rfckt.amplifier('IntpType','cubic',...
402                          'NetworkData', netdata,...
403                         'NoiseData',nfdata,'NonlinearData',ip3data);
404 % Cascading a network after BPF1++++++++++++++++++++++++++++++++++++++++++
405     A= rfckt.cascade('Ckts',{Cable1500mil,Microstrip360mil, ...
406                    SW3_rf2_to_rfc,Microstrip254mil});
407     analyze(A,freq);
408     f= A.AnalyzedResult.Freq;
```

```
409     s= A.AnalyzedResult.S_parameter;
410     n= A.AnalyzedResult.NF;
411     p= A.AnalyzedResult.OIP3;
412     netdata = rfdata.network(          'Freq',f,'Data',s);
413     nfdata  = rfdata.nf(               'Freq',f,'Data',n);
414     ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',p);
415     NetworkAfterBPF1=rfckt.amplifier('IntpType','cubic',...
416                          'NetworkData', netdata,...
417                        'NoiseData',nfdata,'NonlinearData',ip3data);
418 % Cascading {A1,BPF1,A2}++++++++++++++++++++++++++++++++++++++++++++++++++
419     General_RF_Amplifer= rfckt.cascade('Ckts',{NetworkBeforeBPF1,...
420                                              S_BPF1,NetworkAfterBPF1});
421     analyze(General_RF_Amplifer,freqs);
422     clear a
423     a= General_RF_Amplifer.AnalyzedResult;
424 subplot(5,1,1),plot(a.Freq/1e9,20*log10(abs(squeeze(a.S_Parameters(1,1,:)))))
425 axis([2.7 3.52 -30 0]),grid,ylabel('|S11| (dB)')
426 subplot(5,1,2),plot(a.Freq/1e9,20*log10(abs(squeeze(a.S_Parameters(2,1,:)))))
427 axis([2.7 3.52 -30 30]),grid,ylabel('|S21| (dB)')
428 subplot(5,1,3),plot(a.Freq/1e9,a.GroupDelay*1e9)
429 axis([2.7 3.52 5 10]),grid,ylabel('GroupDelay (nSec)')
430 subplot(5,1,4),plot(a.Freq/1e9,a.NF)
431 axis([2.7 3.52 0 10]),grid,ylabel('NF (dB)')
432 subplot(5,1,5),plot(a.Freq/1e9,a.OIP3)
433 axis([2.7 3.52 0 40]),grid,ylabel('OIP3 (dBm)')
434 xlabel('Frequency (GHz)')
435
436     write(a,'General_RF_Amplifer.amp')
437
438
439
```

Annex B Matlab code for General_Mixer.amp

09/11/09 12:48 PM F:\20090...\ESM FrontEnd 20091007 Marki M2 0218 mixer in TM.m 1 of 1

```
clear
m20218 = rfckt.mixer('IntpType','cubic');

freq=[2:1.6:18]*1e9;
VSWR=[2 2.8 2.9 2.7 2.3 2.0 2.10 2.6 2.8 2.2 1.8]; %read from data sheet
s11m=20*log10((VSWR+1)./(VSWR-1));
s11p=[0 0 0 0 0 0 0 0 0 0 0]; % phase no data
s12m=[-30 -30 -30 -30 -30 -30 -30 -30 -30 -30 -30];%assuming
s12p=[0 0 0 0 0 0 0 0 0 0 0];  %phase no data
s21m =-[8 7.2 7.3 7.2 7.3 7.3 7.35 7.4 7.7 7.8 8.0];
s21p=[0 0 0 0 0 0 0 0 0 0 0]; %phase no data
s22m=[-10 -10 -10 -10 -10 -10 -10 -10 -10 -10 -10];
s22p=[0 0 0 0 0 0 0 0 0 0 0]; %phase no data

[freq' s11m' s11p' s12m' s12p' s21m' s21p' s22m' s22p'];

m20218.NetworkData.Freq=freq';
for i=1:length(freq)
    data(:,:,i)=[10^(s11m(i)/20) 10^(s12m(i)/20)
                 10^(s21m(i)/20) 10^(s22m(i)/20)];
 end
m20218.NetworkData.Data=data;

m20218.NoiseData=...
    rfdata.nf('Freq',freq,'Data',abs(s21m));

m20218.NonlinearData=...
    rfdata.ip3('Type','IIP3','Freq',freq,'Data',ones(11,1)*15);

data=[99 99 99 99 99 99 %from data sheet
      99 0  20 10 20 18
      99 55 67 60 70 75
      99 80 80 90 90 80
      99 95 99 99 90 99
      99 99 99 99 99 99];
m20218.MixerSpurData= ...
    rfdata.mixerspur('Data',data,'PinRef',-10,'PLORef',10);

m20218.FLO=2.8e9;
m20218.FreqOffset      = [0.1 1 1000]*1e3;
m20218.PhaseNoiseLevel = [-60 -90 -120]; %from data sheet of LO

analyze(m20218,(2:0.1:20)*1.e9)
write(m20218,'General_Mixer.amp');

CktIndex = 1;       % Plot the output only
Pin = -10;          % Input power is -10dBm
Fin = 3.11e9;        % Input center frequency
plot(m20218, 'MIXERSPUR', CktIndex, Pin, Fin);
```

Annex C Matlab code for General_IF_Amplifer.amp

12/11/09 3:29 PM F...\ESM FrontEnd 20091111 Figure5 General IF amplider in TM.m 1 of 5

```
clear, close all
mil_to_m=0.001*25.39999999999999e-3;
freq=(0:0.01:20)*1.e9;
Z0=50;
%%%%%%%%%%%%%%%%%%%%
% Create RF objets %
%%%%%%%%%%%%%%%%%%%%
%*************************************************************************
% MIcrostrip lines
    %Create microstrip line template
    Microstrip = rfckt.microstrip(...
            'Width'      , 4.571999998e-4, ...
            'Height'     , 2.539999999e-4, ...
            'Thickness'  , 1.777999999e-5, ...
            'EpsilonR'   , 4.500000000e+0, ...
            'LossTangent', 0.001000000e+0, ...
            'SigmaCond'  , 5.813000000e+7, ...
            'LineLength' , 0.01, ...
            'StubMode'   , 'None'    , ...
            'Termination', 'None');    %define substrate

     %The 1st microstrip line length=1402mil ------------------------------
    L1402 = 1402; %microstrip line length
    iL1402 = 0.3; %extra insertion loss for microstrop941mil

         mp = copy(Microstrip);                %create a new line
     set(mp,'LineLength',L1402*mil_to_m);        %change length
         a = analyze(mp,freq);                 %calcuate s-par od M-line
         mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
         inserLoss = -iL1402; %dB Extra inserLoss for connections between
                              %M-line and other devices, especially there is a
                              %SMA is connected at one end of the M-line
         mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
         mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
         nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
        ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
        nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
                                     %inersion loss an NF in dB
        ip3(:)= 999;         %IP3 for M-Line assumed inf
        spdata  = rfdata.network('Freq',freq,'Data',mp_sp);
        nfdata  = rfdata.nf('Freq',freq,'Data',nf');
        ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
     %using Amp-model to represent M-line in order to use
     %the inerstion loss of M-line as NF, when system NF is calculted
     Microstrip1402mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
         ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);

    %The 2ed microstrip line length=510mil---------------------------------
     L510 = 510;
    iL510 = 0.1;
         mp = copy(Microstrip);                %create a new line
```

```
52      set(mp,'LineLength',L510*mil_to_m);         %change length
53          a = analyze(mp,freq);                  %calcuate s-par od M-line
54          mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
55          inserLoss = -iL510; %dB Extra inserLoss for connections between
56                              %M-line and other device, especially there is a
57                              %SMA is connected at one end of the M-line
58          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
59          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
60          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
61         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
62         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
63                                        %inersion loss an NF in dB
64         ip3(:)=999;
65         spdata=rfdata.network('Freq',freq,'Data',mp_sp);
66         nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
67         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
68      %using Amp-model to represent M-line in order to take into account
69      %the inerstion loss of M-line as NF,when system NF is calculted
70      Microstrip510mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
71          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
72
73     %The 3ed microstrip line length=459mil---------------------------------
74      L459 = 459;
75     iL459 = 0.1;
76          mp = copy(Microstrip);                 %create a new line
77      set(mp,'LineLength',L510*mil_to_m);         %change length
78          a = analyze(mp,freq);                  %calcuate s-par od M-line
79          mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
80          inserLoss = -iL510; %dB Extra inserLoss for connections between
81                              %M-line and other device, especially there is a
82                              %SMA is connected at one end of the M-line
83          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
84          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
85          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
86         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
87         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
88                                        %inersion loss an NF in dB
89         ip3(:)=999;
90         spdata=rfdata.network('Freq',freq,'Data',mp_sp);
91         nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
92         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
93      %using Amp-model to represent M-line in order to take into account
94      %the inerstion loss of M-line as NF,when system NF is calculted
95      Microstrip459mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
96          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
97
98     %The 4rd microstrip line length=989mil---------------------------------
99     L989 = 989;
100    iL989 = 0.3;
101         mp = copy(Microstrip);                 %create a new line
102     set(mp,'LineLength',L989*mil_to_m);         %change length
```

```
103          a = analyze(mp,freq);                  %calcuate s-par od M-line
104          mp_sp = a.AnalyzedResult.S_Parameters; %take the line s-parameter
105          inserLoss = -iL989; %dB Extra inserLoss for connections between
106                           %M-line and other device, especially there is a
107                           %SMA is connected at one end of the M-line
108          mp_sp(1,2,:)= mp_sp(1,2,:)*10^(inserLoss/20); %add extra insertion
109          mp_sp(2,1,:)= mp_sp(2,1,:)*10^(inserLoss/20); %loss M-lin s21,s12
110          nf = zeros(length(mp_sp(2,1,:)),1); %create a noise vector
111         ip3 = zeros(length(mp_sp(2,1,:)),1); %create a IP3 vector
112         nf = abs(20*log10(abs(squeeze(mp_sp(2,1,:))))); %using M-line
113                                        %inersion loss an NF in dB
114         ip3(:)=999;
115         spdata=rfdata.network('Freq',freq,'Data',mp_sp);
116         nfdata  = rfdata.nf ('Freq',freq,'Data',nf');
117         ip3data = rfdata.ip3('Type','OIP3','Freq',freq','Data',ip3');
118      %using Amp-model to represent M-line in order to take into account
119      %the inerstion loss of M-line as NF,when system NF is calculted
120      Microstrip989mil = rfckt.amplifier('IntpType','Cubic','NetworkData'...
121          ,spdata,'NoiseData',nfdata ,'NonlinearData',ip3data);
122
123 %*************************************************************************
124     %Create IF Amplifiers ***************************************************
125     famp1 =  [0   0.5  1.0  1.5  2.0  2.5  3.0  3.5 4.0  4.5  5.0 ];
126     NFamp1 = [3.6 3.64 3.71 3.95 4.15 4.2  4.4  4.6 5.0  5.2  5.35];
127     ip3amp1 =[35  33   33   32   31.5 30.8 30.6 29  28.8 28.5 27 ];
128     HMC589ST89 = read(rfckt.amplifier,'HMC589ST89_de-embedded.s2p');
129         f = famp1*1.e9;
130         nf=NFamp1; %NF from device data sheet
131         nfdata = rfdata.nf('Freq',f,'Data',nf);
132         ip3=ip3amp1; %OIP3 data from device data sheet
133         ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',ip3);
134         set (HMC589ST89,'IntpType','Cubic','NoiseData',nfdata,...
135             'NonlinearData',ip3data);
136         clear f nf nfdata ip3 ip3data
137         analyze(HMC589ST89,freq);
138
139         nfdata  = rfdata.nf (              'Freq',freq,'Data',...
140                                             HMC589ST89.AnalyzedResult.NF);
141         ip3data = rfdata.ip3('Type','OIP3','Freq',freq,'Data',...
142                                          HMC589ST89.AnalyzedResult.OIP3');
143         netdata = rfdata.network('Freq',freq,'Data',...
144                                  HMC589ST89.AnalyzedResult.S_Parameters);
145         set (HMC589ST89,'IntpType','Cubic','NetworkData',netdata,...
146                                             'NoiseData',nfdata,...
147                                         'NonlinearData',ip3data);
148 %*************************************************************************
149 %LPF1
150     A = read(rfckt.amplifier,'LFCN_575___Plus25degC.S2P');
151         analyze(A,freq);
152         nf=-20*log10(abs(squeeze(A.AnalyzedResult.S_parameter(2,1,:))));
153         ip3 = ones(1,length(freq)); %create a IP3 vector
```

```
154         ip3(:)=999;
155          nfdata = rfdata.nf (              'Freq',freq,'Data',nf);
156         ip3data = rfdata.ip3('Type','OIP3','Freq',freq,'Data',ip3');
157         netdata = rfdata.network(         'Freq',freq,...
158             'Data',A.AnalyzedResult.S_parameter);
159
160        LPF1= rfckt.amplifier('IntpType','Cubic',...
161               'NetworkData',  netdata,...
162               'NoiseData',     nfdata,...
163               'NonlinearData',ip3data);
164         clear nf nfdata ip3 ip3data netdata A i ans s11 s21 AX H1 H2
165 %LPF2
166     A = read(rfckt.amplifier,'LFCN_2000___Plus25degC.S2P');
167         analyze(A,freq);
168         nf=-20*log10(abs(squeeze(A.AnalyzedResult.S_parameter(2,1,:))));
169         ip3 = ones(1,length(freq)); %create a IP3 vector
170         ip3(:)=999;
171          nfdata = rfdata.nf (              'Freq',freq,'Data',nf);
172         ip3data = rfdata.ip3('Type','OIP3','Freq',freq,'Data',ip3');
173         netdata = rfdata.network(         'Freq',freq,...
174             'Data',A.AnalyzedResult.S_parameter);
175
176        LPF2= rfckt.amplifier('IntpType','Cubic',...
177               'NetworkData',  netdata,...
178               'NoiseData',     nfdata,...
179               'NonlinearData',ip3data);
180         clear nf nfdata ip3 ip3data netdata A i ans s11 s21 AX H1 H2
181 %*************************************************************************
182 % Cascading a network
183     A = rfckt.cascade('Ckts',...
184               {Microstrip1402mil, ...
185                  LPF1,...
186                Microstrip510mil, ...
187                  HMC589ST89, ...
188                Microstrip459mil, ...
189                  LPF2, ...
190                Microstrip989mil});
191     analyze(A,freq);
192     f= A.AnalyzedResult.Freq;
193     s= A.AnalyzedResult.S_parameter;
194     n= A.AnalyzedResult.NF;
195     p= A.AnalyzedResult.OIP3;
196     netdata = rfdata.network(          'Freq',f,'Data',s);
197     nfdata  = rfdata.nf(               'Freq',f,'Data',n);
198     ip3data = rfdata.ip3('Type','OIP3','Freq',f,'Data',p);
199     General_IF_Amplifer =rfckt.amplifier('IntpType','cubic',...
200                         'NetworkData', netdata,...
201                        'NoiseData',nfdata,'NonlinearData',ip3data);
202     analyze(General_IF_Amplifer,freq);
203     a= General_IF_Amplifer.AnalyzedResult;
204 subplot(5,1,1),plot(a.Freq/1e9,20*log10(abs(squeeze(a.S_Parameters(1,1,:)))))
```

```
205 axis([0 6 -40 0]),grid,ylabel('|S11| (dB)')
206 subplot(5,1,2),plot(a.Freq/1e9,20*log10(abs(squeeze(a.S_Parameters(2,1,:)))))
207 axis([0 6 -70 30]),grid,ylabel('|S21| (dB)')
208 subplot(5,1,3),plot(a.Freq/1e9,a.GroupDelay*1e9)
209 axis([.05 1 1 5]),grid,ylabel('GroupDelay (nSec)')
210 subplot(5,1,4),plot(a.Freq/1e9,a.NF)
211 axis([.05 1 1 10]),grid,ylabel('NF (dB)')
212 subplot(5,1,5),plot(a.Freq/1e9,a.OIP3)
213 axis([.05 1. 0 50]),grid,ylabel('OIP3 (dBm)')
214 xlabel('Frequency (GHz)')
215
216     write(a,'General_IF_Amplifer.amp')
217
218
219
```

List of symbols/abbreviations/acronyms/initialisms

.amp	The AMP data file describes a single linear or nonlinear device. The file extension is .amp.
ADC	Analog-to-Digital Converter
AMP	Amplifier
ANT	Antenna
ATT	Attenuator
BPF	Band Pass Filter
DC	Direct Current
DND	Department of National Defence
DR	Dynamic Range
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
ES	Electronic Support
ESM	Electronic Support Measure
EW	Electronic Warfare
FSDR	Full Scale Dynamic Range
HFBM	High-Fidelity Behavioural Model
IIP3	Input Third-Order Intercept Point
IM	Intermodulation
IMD	Intermodulation Distortion
IF	Intermediate Frequency
IP1dB	Input P1dB
LPF	Low Pass Filter
LO	Local Oscillator
MIC	Microwave Integrated Circuit
MMIC	Monolithic Microwave Integrated Circuit
MMCX	Micro-Miniature Coaxial
NF	Noise Figure

OP1dB	Output P1dB
OIP3	Output Third-Order Intercept Point
P1dB	1-dB Compression Point
PCB	Printed Circuit Board
P_I	Input Power
P_O	Output Power
REW	Radar Electronic Warfare
RF	Radio Frequency
R&D	Research & Development
.s2p	the file extension of s-parameter files for two-port RF/microwave device
.s3p	the file extension of s-parameter files for three-port RF/microwave device
SFDR	Spur-Free Dynamic Range
$(SNR)_o$	receiver output signal-to-noise ratio
SPDT	Single Pole and Double Throw
SSMP	Small Solder Mount Push-On
STK	Satellite Tool Kit
SW	Switch
UAV	Unmanned Aerial Vehicle
Z0	Characteristic impedance of a transmission line
ZS	Source impedance
ZL	Load impedance

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario K1A 0Z4

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

High-fidelity behavioural model for electronic support measure microwave receiver front-end

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Wu, C.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
May 2010	80	19

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Ottawa
3701 Carling Avenue
Ottawa, Ontario K1A 0Z4

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
15dz07-02 , 15df02	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Ottawa TM 2010-071	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

For many years, hardware development has been an essential step in electronic support measures (ESM) R&D programs carried out in DRDC Ottawa. The objectives of hardware development are:

- To investigate new electronic support measures (ESM) system architectures,
- To test new radar signal detection algorithms using field-collected data,
- To conduct the field trials and demonstrate different operational concepts, and
- To polish researchers' knowledge through using state-of-the-art hardware technologies and applying new theories, so that researchers can better support their client from ESM perspective.

Experience has shown that designing and building ESM hardware or systems is costly in an R&D environment because:

- There are a lot of 'cut-and-tries' in hardware development where sometimes a new design can be completely different from the previous iteration.
- It is very time-consuming and labour-intensive to put ESM system hardware together.
- Nowadays ultra wideband microwave and digital components used in electronic warfare systems are still very expensive.

In order to mitigate the problem, reduce the development time, and partially or sometimes fully achieve the objectives mentioned above, model-based system design and simulation using high-fidelity behavioural models (HFBM) is a good approach and should be a part of the ESM system development cycle.

This report presents how to develop a HFBM of an ESM receiver microwave front-end using measured data and manufacturers' specifications, and the Matlab/Simulink$^{(R)}$ scripting language. The front-end behavioural model and a digitizer behavioural model, which will be presented in another report, will be used to form different ESM digital receiver systems depending on the system architectures. These systems can be applied in different radar electronic warfare scenarios that are created by Satellite Tool Kit$^{(R)}$ software for high-fidelity model-based 'system'-in-the-loop simulation and data generation.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

electronic support measure receiver, RF/microwave receiver, receiver front-end, high fidelity modeling, model-based design

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca