LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ADS	*Advanced Design System*
AEL	*Application Extension Language*
ARV	Analyseur de réseaux vectoriel
BGA	*Ball Grid Array*
CAO	Conception assistée par ordinateur
c.c.	Court-circuit
c.o.	Circuit ouvert
CPLD	*Complex Programmable Logic Device*
DIP	*Dual Inline Package*
EMDS	*Electromagnetic Design System*
FDD	*Frequency Division Duplex*
FDM	Figure de mérite
FET	*Field Effect Transistor*
Im	Nombres imaginaires
MEMS	*Micro Electro-Mechanical System*
pHEMT	*Pseudomorphic High Electron Mobility Transistor*
Re	Nombres réels
RF	Radio fréquences
TDD	*Time Division Duplex*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*

LISTE DES SYMBOLES ET UNITÉS DE MESURE

SYMBOLES

I	Intensité de courant électrique
l_e	Longueur électrique
V	Tension électrique
Z_0	Impédance caractéristique
Z_L	Impédance de charge
Z_S	Impédance de source
ε	Permittivité électrique
ε_r	Permittivité relative
ε_0	Constante de permittivité du vide
λ_0	Longueur d'onde dans le vide
λ_g	Longueur d'onde guidée
μ	Perméabilité magnétique
μ_r	Perméabilité relative
μ_0	Constante de perméabilité du vide
Γ	Coefficient de réflexion
T	Coefficient de transmission

UNITÉS GÉOMÉTRIQUES

mil	Millième(s) de pouce
po	Pouce(s)
°	Degré(s)

UNITÉS ÉLECTRIQUES ET MAGNÉTIQUES

dBm	Rapport de puissance exprimé en décibels par rapport à 1 milliwatt.
mA	Milliampère(s)
V	Tension électrique en volt(s).
Ω	Ohm(s)

UNITÉS DE TEMPS

GHz	Gigahertz
MHz	Mégahertz
s	Secondes

UNITÉS CALORIFIQUES

°C	Degré(s) Celsius

UNITÉS D'INTENSITÉ

dB	Rapport de grandeurs en décibel(s)
%	Rapport de grandeurs en pourcentage

INTRODUCTION

La récente pénétration des radios logicielles dans la sphère des systèmes de communications tactiques a intensifié les besoins pour développer des composantes micro-ondes agiles en fréquences. Dans le cadre de la Chaire de recherche Ultra Electronics TCS, une préalable preuve de concept sur les possibilités de concevoir un syntoniseur électronique reconfigurable agile en bande III [1350 MHz; 1850 MHz] avait abouti sur des résultats prometteurs incitant à l'élaboration d'une méthode de conception et d'un deuxième prototype plus avancé. Cette recherche en est la concrétisation.

Au premier chapitre de ce travail, l'adaptation d'impédance et le fonctionnement du syntoniseur à tronçons sont abordés afin d'introduire les fondements du duplexeur. Au deuxième chapitre, les défis de l'élaboration d'une méthode de conception d'un syntoniseur reconfigurable sont exposés et une approche pratique et réalisable est proposée. Les possibles configurations de syntoniseur découlant de la méthode de conception proposée ainsi que leurs performances électriques simulées sont exposées au chapitre 3. Finalement, le chapitre 4 explique la fabrication du prototype en fonction de la configuration sélectionnée et le chapitre 5 en révèle les performances réelles mesurées en laboratoire et les comparent aux résultats de simulations présentés au chapitre 3.

CHAPITRE 1

PRÉSENTATION ET DÉFINITION D'UN DUPLEXEUR

1.1 Concepts de base

En premier lieu, avant d'aborder le sujet du fonctionnement d'un duplexeur, il est important de se rafraîchir la mémoire et de revenir sur les concepts de base sur lesquels s'appuient les principes d'adaptation d'impédances.

1.1.1 L'impédance

La définition d'impédance selon le Petit Robert est : « Rapport entre les valeurs efficaces de la tension aux bornes d'un circuit et de l'intensité du courant sinusoïdal qui le traverse » ou encore sa définition tirée de l'anglais « L'opposition au flux d'un courant alternatif dans un circuit ». L'impédance est donc une caractéristique d'intérêt lors de l'étude de systèmes soumis à un régime oscillatoire au même titre que la résistance pour les circuits en régime continu.

En effet, la définition de résistance étant très semblable, « L'opposition au flux d'un courant offert par un conducteur », cette dernière est souvent utilisée pour introduire le concept d'impédance. De plus, ces deux notions représentant essentiellement le même rapport (V / I), elles utilisent donc la même unité de mesure, soit l'ohm (Ω). Cependant, ce parallèle mène souvent à une mauvaise compréhension du phénomène. Par exemple, il est possible de croire à partir de ces similitudes qu'il est envisageable de mesurer l'impédance d'une ligne de transmission 50 Ω à l'aide d'un multimètre ou qu'une ligne de transmission de 75 Ω entraîne des pertes puisque son impédance est purement réelle. Ces interprétations sont pourtant fausses.

Le fait que la partie réelle d'une impédance n'est pas nécessairement résistive est à la source de telles interprétations du phénomène d'impédance. Si tel était le cas, cela impliquerait une

perte sous forme d'émission de chaleur, de lumière ou de radiation électromagnétique. Cependant, la partie réelle d'une impédance peut aussi être principalement, voire uniquement, générée par des composantes réactives (dans les cas idéaux).

Par exemple, la ligne de transmission coaxiale parfaite suivante, telle que présentée à la figure 1.1, peut présenter une impédance strictement réelle alors qu'elle est composée d'éléments infinitésimaux uniquement inductifs ou réactifs.

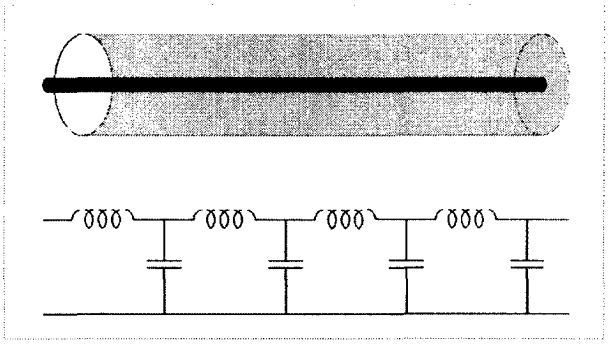


Figure 1.1 Ligne de transmission coaxiale et son modèle à éléments infinitésimaux.

En effet, si l'on approche la ligne de transmission parfaite selon la théorie des circuits (Pozar, 2005), chaque section infinitésimale peut être représentée par une inductance en série et un condensateur en parallèle. L'inductance représente les effets inductifs parasites des conducteurs alors que le condensateur symbolise les effets capacitifs entre ceux-ci. Le modèle réaliste quant à lui introduit les pertes énergétiques sous forme de deux résistances. Une première résistance en série avec la bobine représentant la résistance au flux d'un courant par les conducteurs et une deuxième en parallèle avec le condensateur représentant le courant de fuite dans le diélectrique se situant entre les conducteurs. Pour les besoins de la démonstration, seul le modèle parfait, présenté à la figure 1.2, sera étudié dans ce chapitre.

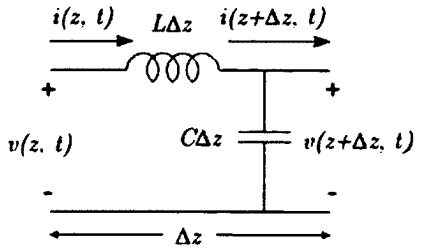


Figure 1.2 Section infinitésimale d'une ligne de transmission parfaite.

En dénotant par z la variable de position sur la ligne, selon la loi de Kirchhoff sur les tensions et les courants

$$v(z,t) - L\Delta z \frac{\partial i(z,t)}{\partial t} - v(z+\Delta z,t) = 0, \tag{1.1}$$

$$i(z,t) - C\Delta z \frac{\partial v(z+\Delta z,t)}{\partial t} - i(z+\Delta z,t) = 0. \tag{1.2}$$

où $v(z,t)$ et $i(z,t)$ dénotent respectivement la tension et le courant en un point z et en un instant t sur la ligne de transmission. À la limite $\Delta z \to 0$ et en régime sinusoïdal, on obtient les équations différentielles dites des télégraphistes (Pozar, 2005)

$$\frac{dV(z)}{dz} = -j\omega L \cdot I(z) \tag{1.3}$$

et

$$\frac{dI(z)}{dz} = -j\omega C \cdot V(z), \tag{1.4}$$

qui se résolvent à

$$V(z) = V_o^+ e^{-j\omega\sqrt{LC}z} + V_o^- e^{j\omega\sqrt{LC}z} \tag{1.5}$$

et

$$I(z) = I_o^+ e^{-j\omega\sqrt{LC}z} + I_o^- e^{j\omega\sqrt{LC}z} \tag{1.6}$$

sous leur forme s'appliquant à une onde progressive, où V_o^+ et I_o^+ dénotent les valeurs de tensions et de courants progressant dans le sens positif de z et V_o^- et I_o^- les valeurs de tensions et de courants progressant dans le sens négatif de z. Ainsi, à partir de (1.5) et de (1.6) on obtient

$$I(z) = \sqrt{\frac{C}{L}}\left(V_o^+ e^{-j\omega\sqrt{LC}z} - V_o^- e^{j\omega\sqrt{LC}z}\right) \tag{1.7}$$

qui amène à conclure que l'impédance d'une ligne de transmission sans pertes est

$$Z_o = \frac{V_o^+}{I_o^+} = -\frac{V_o^-}{I_o^-} = \sqrt{\frac{L}{C}}. \tag{1.8}$$

L'analyse des champs électromagnétiques pour une ligne de transmission coaxiale (Collin, 1966) révèle que

$$L = \frac{\mu}{2\pi}\ln\frac{b}{a} \tag{1.9}$$

et

$$C = \frac{2\pi\varepsilon'}{\ln b/a}. \tag{1.10}$$

où ε' est la composante réelle de la permittivité. Ce qui permet, pour des caractéristiques de diélectrique (ε et μ) connues, de finalement trouver la relation entre le rayon du conducteur central (a) et le rayon du conducteur extérieur (b), pour une impédance désirée (figure 1.3).

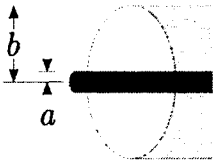


Figure 1.3 Grandeurs physiques d'un câble coaxial.

Si l'on revient à l'exemple énoncé précédemment, pour obtenir une impédance de 75 Ω avec une ligne coaxiale ayant comme diélectrique un mélange de polyéthylène et d'air ($\varepsilon_r = 1.43$ et $\mu_r = \mu_o$), on obtient le rapport $b/a = 4.463$, ce qui signifie que

$$\boxed{L = 299\,nH/m} \tag{1.11}$$

et

$$\boxed{C = 53.2pF/m.} \tag{1.12}$$

On constate donc que l'impédance d'une ligne de transmission sans pertes, c'est-à-dire en tenant uniquement compte des effets inductifs et capacitifs des conducteurs, peut effectivement présenter une impédance strictement réelle.

Donc, tout comme le concept de résistance, l'impédance sert à définir le lien entre la tension et le courant en un point du circuit. Cependant, comme l'impédance définit cette relation pour les circuits soumis à un courant alternatif elle doit aussi quantifier le déphasage entre ces derniers ce qui lui confère la forme d'un vecteur dans le plan complexe. Un vecteur pouvant être décrit soit dans sa forme cartésienne soit sous sa forme polaire il en va de même pour l'impédance. Tel qu'illustré à la figure 1.4, la transposition de l'impédance dans un plan cartésien complexe permet de constater ceci.

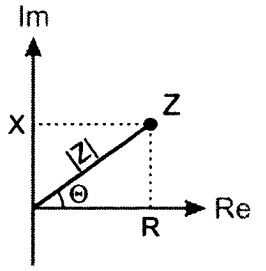


Figure 1.4 Plan complexe d'impédance.

Où on dénote l'impédance, selon le référentiel désiré, par

$$Z = |Z|e^{j\Theta} = R + jX\,. \tag{1.13}$$

1.1.2 L'adaptation d'impédance

Dans le domaine des radiofréquences et des micro-ondes, la principale motivation pour adapter l'impédance d'une charge à celle de sa source est de maximiser le transfert d'énergie entre ces derniers. En effet, lorsqu'une onde incidente se propage vers un milieu d'impédance différente, une portion du signal est reflétée vers la source, ce qui diminue la proportion du signal transmis. La portion réfléchie du signal incident au port est définie comme étant le coefficient de réflexion (Γ).

Le coefficient de réflexion (Γ) est fonction de l'impédance caractéristique de la source (Z_0) et de l'impédance de la charge (Z_L) et il est défini selon la relation

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0}\,. \tag{1.14}$$

En posant $\Gamma = 0$, il découle que pour n'importe quelle impédance de charge complexe $Z_L = R_L + jX_L$, il faut une impédance présentée par la source telle que $Z_S = R_L - jX_L$ afin

d'éviter toutes réflexions. Donc, pour toute impédance de charge, l'impédance présentée par la source afin d'éviter les réflexions se doit d'être son complexe conjugué.

1.2 Présentation d'un syntoniseur

Dans un système de télécommunication, adapter l'antenne au reste du système est très important et particulièrement critique à la réception. En effet, à l'émission, toute perte introduite avant l'antenne peut être compensée en amont par l'amplificateur. L'adaptation à l'émission a donc pour objectif principal d'augmenter l'efficacité énergétique de celle-ci. Cependant, à la réception, la puissance du signal reçu étant extrêmement faible et très proche du plancher de bruit, toute perte introduite avant l'amplification ne peut être compensée et est donc destructive. Il est alors critique de transférer le maximum de ce peu de puissance reçue à l'étage d'amplification afin de récupérer le maximum d'énergie possible et par conséquent le maximum d'information contenu dans ce signal. C'est ce qu'accomplit le syntoniseur.

Le syntoniseur se situe donc normalement entre l'antenne et le reste du système de télécommunication, schématisé à la figure 1.5.

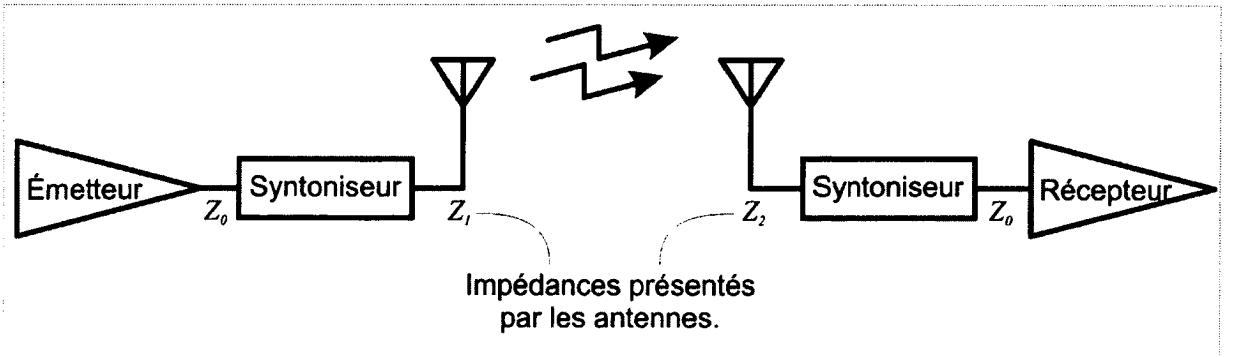


Figure 1.5 Syntoniseurs en contexte dans un système de communication sans-fil.

Il existe une multitude de configurations de syntoniseur en fonction des caractéristiques et contraintes du système conçu. La gamme de fréquences, la largeur de bande, la puissance, et la technologie de fabrication sont tous des éléments déterminants dans le choix du type de configuration désiré.

1.2.1 Syntonisation par éléments localisés

Dans le domaine des basses fréquences ou si les composantes utilisées sont suffisamment petites ($l_e << \lambda_g$), les circuits d'adaptation à éléments localisées sont normalement les plus simples à réaliser. Aussi, ils offrent la plus grande flexibilité puisque les composantes discrètes sont souvent offertes en version ajustable, ce qui permet l'adaptation de l'antenne sur une plus large bande.

Typiquement, un circuit d'adaptation à base d'éléments localisés utilise un agencement dit en L, basé sur deux éléments réactifs, soit un élément inductif et un élément capacitif. Étant donnée la possibilité d'inverser le circuit en L et de disposer la bobine et le condensateur sur deux positions pour chaque agencement, il en résulte quatre configurations possibles. Chacune des configurations pouvant adapter la moitié de la gamme d'impédances possibles présentée par un circuit, il en découle que pour tout point d'impédance donnée, deux configurations peuvent s'y adapter tel qu'illustré à la figure 1.6 (Agilent, 2000).

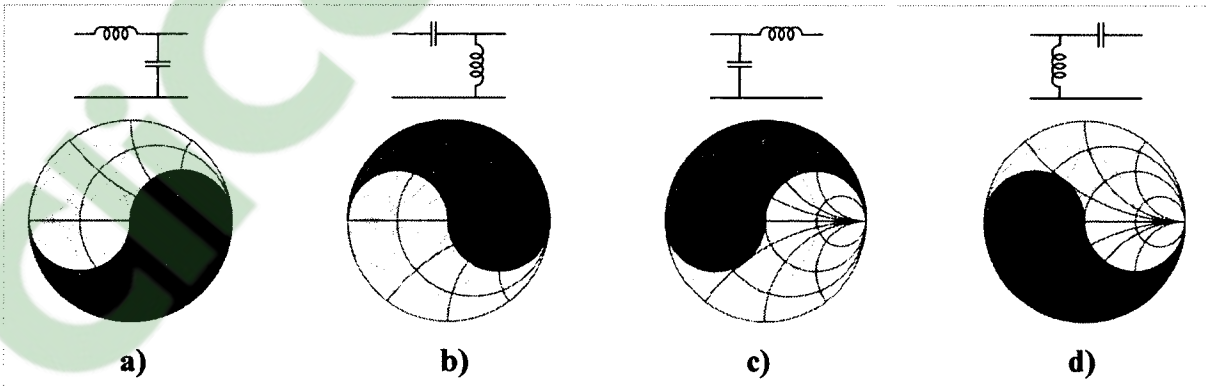


Figure 1.6 Zones de l'abaque de Smith couvertes par les 4 configurations du circuit d'adaptation dit « en L ».

La double solution pour chaque point d'adaptation est aussi évoquée par la solution analytique. Prenons pour exemple la solution analytique du circuit en L suivant illustré à la figure 1.7.

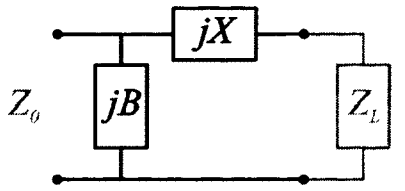


Figure 1.7 Circuit en L à éléments purement réactifs.

En posant $Z_L = R_L + jX_L$ on obtient

$$B = \pm \frac{\sqrt{(Z_0 - R_L)/R_L}}{Z_0} \text{ siemens} \tag{1.15}$$

et

$$X = \pm \sqrt{R_L(Z_0 - R_L)} - X_L \text{ ohms} \tag{1.16}$$

En sachant que ce type de configuration est capable d'adapter une impédance $z_L = Z_L/Z_0$ se situant à l'extérieur du cercle $1 + jX$ de l'abaque tel qu'illustré par la figure 1.6c et la figure 1.6d, on obtient que $R_L < Z_0$. Par conséquent, l'argument des racines des équations (1.15) et (1.16) est donc positif, ce qui confirme que deux solutions existent tel que mentionné précédemment.

1.2.2 Syntonisation par tronçons de ligne de transmission

L'adaptation d'impédance à l'aide de bobines et condensateurs est très répandue à des fréquences en deçà de 1 GHz et est utile pour introduire les concepts de base de l'adaptation. Cependant, les gammes de fréquences auxquelles est confronté l'ingénieur micro-ondes se prêtent généralement mal à l'adaptation par éléments discrets de par la dimension physique des composantes par rapport à la longueur d'onde. Dans les cas où la fréquence d'opération est trop élevée pour utiliser des éléments localisés, il faut alors se tourner vers une approche par éléments distribués, soit l'adaptation par tronçon de ligne de transmission par exemple.

En effet, une fois sorti du carcan du modèle à éléments localisés qui définit les composantes comme étant de dimensions électriques nulles, il est alors possible de concevoir qu'un conducteur puisse présenter une variation de tension et de courant le long de sa structure et donc une impédance spécifique. Si, au long de sa structure, ce conducteur est de géométrie uniforme, entouré par un milieu constant et qu'il est terminé par un court-circuit (c.c.) ou un circuit ouvert (c.o.) alors il peut être considéré comme un tronçon de ligne de transmission.

Un tronçon de ligne de transmission terminée par un c.c. ou un c.o. reflète la très large majorité de l'onde qui lui est injecté ce qui revient à dire que l'amplitude du coefficient de réflexion (Γ) de la terminaison est près de 1. En effet, si le tronçon est terminé en court-circuit, $Z_L = 0$, alors, selon l'équation (1.14),

$$\Gamma_{c.c.} = \left.\frac{Z_L - Z_0}{Z_L + Z_0}\right|_{Z_L \to 0} = -1 \tag{1.17}$$

et si le tronçon est terminé en circuit ouvert, $Z_L = \infty$, alors

$$\Gamma_{c.c.} = \left.\frac{Z_L - Z_0}{Z_L + Z_0}\right|_{Z_L \to \infty} = 1. \tag{1.18}$$

Ce qui représente respectivement le point le plus à gauche et le plus à droite de l'abaque de Smith.

D'après l'équation de l'impédance d'une ligne de transmission en fonction de la distance (l) de la charge, tiré de (Pozar, 2005),

$$Z(l) = Z_0 \frac{Z_L + jZ_0 \tan(\beta l)}{Z_0 + jZ_L \tan(\beta l)} \tag{1.19}$$

où

$$\beta = \varpi\sqrt{LC} \tag{1.20}$$

en c.c. se réduit à

$$Z(l) = jZ_0 \tan(\beta l) \tag{1.21}$$

qui peut donc prendre n'importe quelle valeur entre $j\infty$ et $-j\infty$ en fonction de l. C'est donc dire qu'il est possible, en ajustant la longueur du tronçon, d'obtenir n'importe quelle valeur de réactance, c'est-à-dire de simuler n'importe quelle inductance parfaite ou n'importe quel condensateur parfait à la fréquence d'opération.

Donc, s'il est possible de transformer l'impédance de la charge de manière à ce que sa composante réelle soit égale à Z_0, alors il sera possible d'adapter la composante réactive (jX_T) avec un tronçon présentant l'impédance réactive inverse ($-jX_T$). Ainsi ces composantes s'annulent mutuellement.

Si l'admittance est définie comme suit :

$$Y = \frac{1}{Z} = G + jB \tag{1.22}$$

où

$$G = \frac{R_L\left[1 + \tan^2(\beta d)\right]}{R_L^2 + \left[X_L + Z_0 \tan(\beta d)\right]^2} \tag{1.23}$$

et où

$$G = \frac{1}{Z_0} = Y_0 \tag{1.24}$$

il en découle que

$$\tan(\beta d) = \frac{X_L \pm \sqrt{R_L\left[(Z_0 - R_L)^2 + X_L^2\right]/Z_0}}{R_L - Z_0} \quad \text{pour } R_L \neq Z_0 \tag{1.25}$$

et

$$\tan(\beta d) = -\frac{X_L}{2Z_0} \text{ pour } R_L = Z_0. \tag{1.26}$$

Finalement, il s'agit de déterminer la composante réactive B pour la ou les solutions de (1.25) ou (1.26) et de l'adapter avec son complexe conjugué, grâce au tronçon de longueur l trouvée en (1.21).

Donc, en ajustant judicieusement la distance d entre le tronçon et la charge, les impédances présentées par les deux branches au point de jonction se complémentent pour présenter une impédance combinée Z_0.

Tel que présenté par (Pozar, 2005), une synthèse graphique à l'aide de l'abaque de Smith, un compas et une règle est généralement plus facile et rapide que d'effectuer les calculs analytiques formels et n'introduit normalement pas d'erreur significative. L'exemple suivant, représenté par la figure 1.8, en indique la démarche.

Il s'agit de trouver la longueur et la position par rapport à la charge d'un tronçon parallèle court-circuité afin d'adapter cette dernière. Premièrement, il faut placer le point d'impédance normalisé z_L de la charge dans l'abaque d'impédance. Puisque le tronçon à définir est disposé en parallèle, il faut passer à l'abaque d'admittance en transposant z_L à travers z_0 / y_0 afin d'obtenir y_L. Ensuite, il faut tracer le cercle passant par y_L et centré sur y_0 et marquer les deux points d'intersection, y_1 et y_2, avec le cercle d'admittance défini par $1+jx$. Les distances d_1 et d_2 mesurées dans le sens horaire sur l'échelle extérieure de l'abaque entre y_L et chacun des points y_1 et y_2 représentent les distances en longueurs d'onde entre la charge et le tronçon pour ces deux solutions possibles. À ces distances de la charge, la partie réelle de l'admittance normalisée est maintenant égale à 1, il ne reste donc à ces points que la partie imaginaire à annuler à l'aide de tronçons de longueur adaptée l_1 et l_2. Pour déterminer ces longueurs, il faut alors mesurer la longueur nécessaire entre une charge court-circuit (à l'extrémité droite de l'abaque) et le point de susceptance conjugué qui se trouve à

l'intersection du contour de l'abaque et de l'arc de susceptance passant par l'autre point d'admittance, soit l'arc passant par y_2 pour adapter la nouvelle charge y_1 et vice et versa.

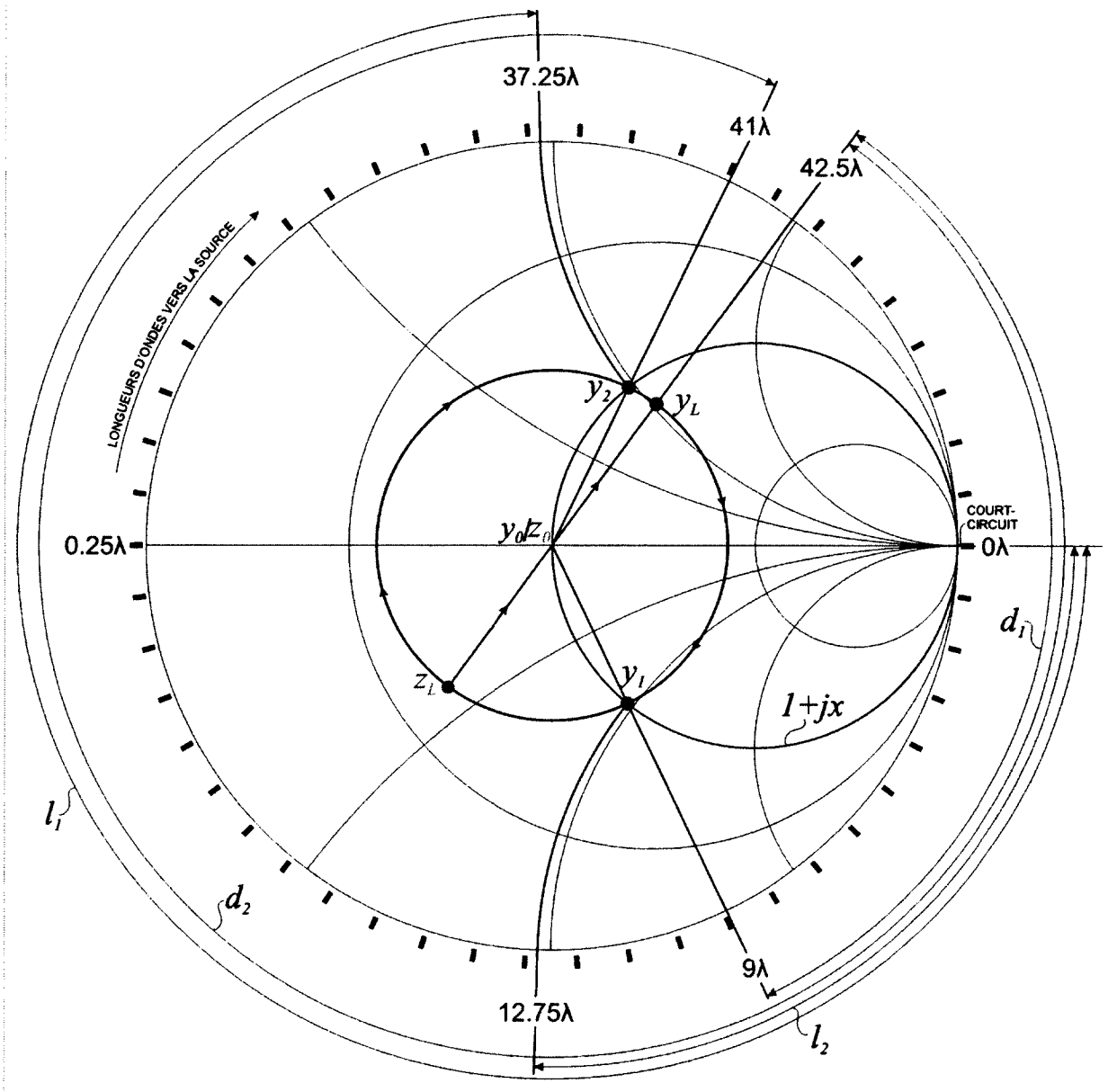


Figure 1.8 Synthèse graphique du syntoniseur à deux tronçons à l'aide de l'abaque de Smith.

Ainsi, deux solutions sont offertes, soit (l_1, d_1) et (l_2, d_2). La meilleure configuration est normalement celle qui a de plus petites dimensions puisque ceci lui confère de meilleures

propriétés de réponse fréquentielle. En effet, plus la longueur d'un tronçon est petite, moins la grandeur électrique variera en fonction de la fréquence.

Par contre, lorsqu'aucune des deux solutions n'est clairement avantagée par ses dimensions, c'est-à-dire que lorsqu'aucune des solutions n'a à la fois la plus petite valeur de l et de d simultanément, alors il est préférable de tracer la réponse fréquentielle afin de déterminer laquelle offre les caractéristiques les plus souhaitables dans le contexte.

1.2.3 Les limites théoriques de l'adaptation

Dans tous les cas d'adaptation d'une charge complexe à une impédance résistive, une limite théorique délimite l'amplitude minimum du coefficient de réflexion qui peut être réalisable sur une certaine largeur de bande. Le critère de Bode-Fano, (Bode, 1945) et (Fano, 1947), définit le lien entre la largeur de bande et une charge composée d'une combinaison d'un élément résistif et d'un élément réactif. Par exemple, pour une charge composée d'une résistance et d'un condensateur en parallèle, le critère est défini selon

$$\int_0^\infty \frac{1}{\omega^2} \ln \frac{1}{|\Gamma(\omega)|} d\omega < \pi RC \tag{1.27}$$

où $\Gamma(\omega)$ est le coefficient de réflexion présenté à l'entrée du syntoniseur branché sur la charge. Donc, si l'on pose que la charge doit être adaptée sur une largeur de bande $\Delta\omega$ et que l'amplitude du coefficient de réflexion maximum sur cette bande est de Γ_m et $\Gamma(\omega)$=1 ailleurs, alors le critère devient

$$\int_0^\infty \frac{1}{\omega^2} \ln \frac{1}{\Gamma_m} d\omega = \frac{\ln \Gamma_m}{\Delta\omega} < \pi RC\,. \tag{1.28}$$

Ce critère, en plus de définir la limite théorique de performance d'un syntoniseur, indique aussi que plus la largeur de bande souhaitée est élevée plus le coefficient de réflexion le sera

aussi. Ainsi, seule une fréquence ponctuelle peut être parfaitement adaptée, sinon, il faut accepter le compromis entre largeur de bande et adaptation.

1.3 Le duplexeur

Dans le schéma précédent d'un système de télécommunication, figure 1.5, seul un lien unidirectionnel est représenté. Ce qui n'est pas représentatif de la majorité des systèmes modernes. Actuellement, dans un système de communication bidirectionnelle simultanée l'antenne sert aussi bien de radiateur que de récepteur, il est donc nécessaire de rattacher un dispositif entre l'antenne et les branches d'émission et de réception afin d'isoler ces systèmes l'un de l'autre tout en leur donnant un accès simultané à l'antenne. Ce mode de communication se nomme multiplexage à duplex fréquentiel (FDD : *Frequency Division Duplex*). De plus, pour empêcher la fuite du signal émis dans la branche de réception, un circulateur peut être ajouté au circuit dans le but d'isoler le circuit de réception du circuit d'émission. Tel que présenté à la figure 1.9, ce dispositif, typiquement composé de filtres, de syntoniseurs et d'un circulateur se nomme duplexeur.

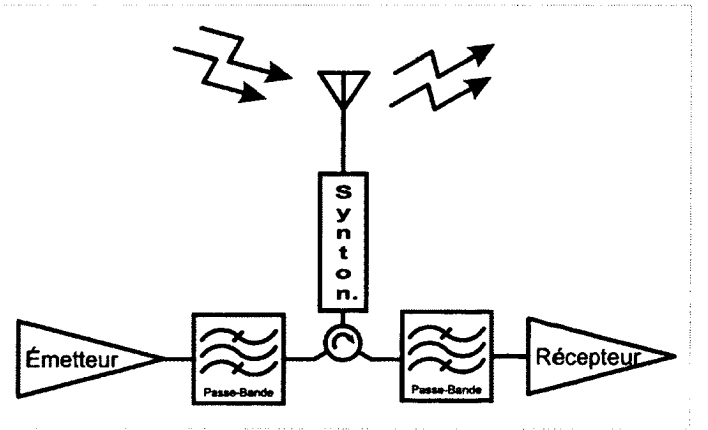


Figure 1.9 Éléments d'un duplexeur type.

Certaines topographies de circuits n'ont pas de syntoniseur puisque l'antenne est déjà bien adaptée sur la petite bande utilisée ou parce que les pertes engendrées par la désadaptation sont acceptables pour l'application. Ceci dit, dans les systèmes de pointe à large bande où la

performance en débit ou en distance est primordiale, toute perte à cet étage est néfaste et donc indésirable.

À noter que certains systèmes de communication bidirectionnelle fonctionnent plutôt en mode de multiplexage temporel (TDD : *Time Division Duplex*), c'est-à-dire que l'émission et la réception ne s'effectuent pas simultanément, mais séquentiellement. Ces systèmes ne requièrent normalement pas de duplexeur, mais plutôt des interrupteurs RF ou un multiplexeur RF.

CHAPITRE 2

PROBLÉMATIQUE ET APPROCHE

Étant donné le nombre continuellement croissant d'applications requérant une grande agilité en fréquence et donc en impédance de la tête RF, l'objectif principal de ce projet de recherche est de généraliser les méthodes de conception d'un syntoniseur à tronçons commutés de manière à pouvoir l'appliquer à d'autres contextes.

Dans ce cas-ci, le type de configuration ainsi que les grandeurs physiques du syntoniseur sont les principales inconnues recherchées.

2.1 Approche analytique

L'approche analytique est normalement la méthode privilégiée pour résoudre les problèmes d'ingénierie puisqu'elle permet justement de généraliser les paramètres d'intérêts en fonction des variables applicables. Dans le cas présent, les variables dépendantes sont donc les grandeurs physiques du circuit en fonction des variables indépendantes de *gamme de fréquences d'opération* et de *largeur de bande*.

Cependant, deux principales difficultés s'imposent rapidement en tentant cette approche : le développement de la synthèse mathématique et la quantité de calculs impliquée.

Normalement, la méthode scientifique implique dans l'ordre : l'expérimentation, l'observation et la modélisation. Ceci présuppose que les équations découlant de la modélisation sont des outils mathématiques présentés sous une forme permettant de faire l'analyse du phénomène. C'est-à-dire que les équations sont normalement présentées de manière à prévoir le comportement du phénomène observé en fonction du contexte de l'expérimentation.

À l'inverse, dans le cas où l'on souhaite connaitre le contexte de l'expérience en fonction du comportement observé, alors des manipulations mathématiques peuvent, dans certains cas, permettre de transformer les équations originales afin que les variables indépendantes deviennent les variables dépendantes. Il s'agit alors d'une synthèse mathématique.

Les modèles d'analyse des circuits micro-ondes sont souvent présentés sous forme matricielle en terme de paramètres-S. L'insertion de c éléments commutatifs dans la chaine de composantes ajoute c variables indépendantes à l'équation analytique, qui est équivalent à être confronté à 2^C équations analytiques différentes. La synthèse simultanée de 2^C états du circuit est la principale difficulté à surmonter pour réussir par cette approche.

Le temps requis pour de tels travaux nous aiguille vers une approche ayant une plus haute probabilité de réussite : l'approche d'optimisation par simulations.

2.2 Approche d'optimisation par simulations

Puisque l'approche analytique n'est pas privilégiée pour définir les grandeurs physiques optimales, alors une alternative est d'aborder le problème par l'approche d'essais/erreurs. Déjà, certains outils offerts à même le logiciel de simulation d'Agilent, *Advanced Design System* (ADS), utilisent cette approche dont la fonction *optimize*, c'est-à-dire, conduire et aiguiller des simulations afin d'atteindre un objectif particulier. C'est l'approche mise de l'avant pour conduire cette recherche et illustrée à la figure 2.1.

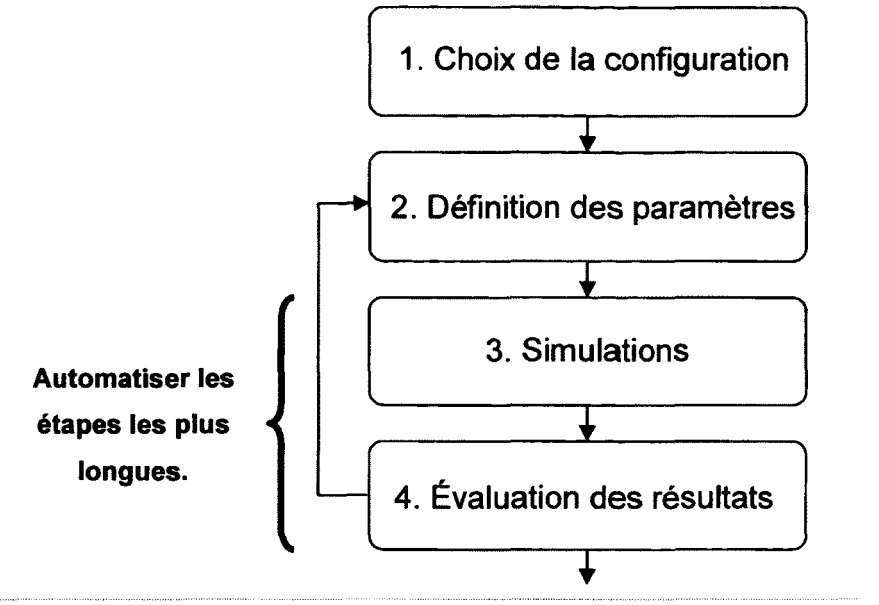


Figure 2.1 Ordinogramme du processus d'optimisation des dimensions physiques du circuit.

La première étape, c'est-à-dire le choix de configurations à optimiser, est conduite manuellement sur un nombre de circuits de différentes configurations choisies judicieusement au préalable. Les configurations les plus prometteuses sont ensuite conservées pour être optimisées. Le chapitre suivant présente les configurations candidates retenues dans le cadre de cette recherche.

Les étapes de définition des paramètres et de simulation sont conduites dans ADS et automatisées grâce au langage de programmation ADS *Application Extension Language* (AEL) qui, à la manière de macros, permet d'appeler la plupart des commandes disponibles par l'interface graphique du logiciel. De plus, AEL offre aussi un environnement de programmation standard basé sur le langage de programmation C permettant de créer des variables, de programmer des boucles et prendre des décisions basées sur une comparaison.

Cependant, l'absence complète d'interface de programmation, d'outils de déverminage et la documentation minimaliste rend le développement logiciel rébarbatif et semé d'embuches. La communauté d'utilisateurs d'AEL échangeant sur le forum prévu à cet effet sur le site web d'Agilent est très reduite et les réponses constructives viennent presque exclusivement

d'employés d'Agilent aussi inscrits à ce forum. À titre d'exemple, l'existence de la fonction *api_set_timer()*, indispensable pour ce projet, n'est nullement documentée et fut découverte à la suite d'une discussion avec un des modérateurs du forum. Pour ces raisons et tel que présenté dans le tableau 2.1, seules les étapes nécessitant ADS, soit les étapes 2 et 3, ont été programmées avec AEL. La quatrième étape fut développée dans Matlab. Ainsi, les données de simulations générées doivent être exportées sous forme de paramètres-S dans des fichiers texte de format Touchstone afin de pouvoir les importer et les traiter dans Matlab.

Tableau 2.1

Environnement logiciel de développement.

Étape	**Outil de programmation utilisé**
1. Choix d'une configuration	Agilent ADS 2006A
2. Définition des paramètres	Agilent ADS 2006A – AEL
3. Simulation	Agilent ADS 2006A – AEL
3.1 Exportation des données	Agilent ADS 2006A – AEL
3.2 Importation des données	Matlab 7.1 (R14 SP3)
4. Évaluation des résultats	Matlab 7.1 (R14 SP3)

À la quatrième étape « évaluation des résultats », les deux principales qualités à évaluer sont le positionnement du nuage d'impédances généré par rapport à la zone à adapter ainsi que l'uniformité de ce nuage. Les figures suivantes donnent un exemple d'un nuage mal positionné (figure 2.2), d'un nuage bien positionné mais non uniforme (figure 2.3) et finalement d'un nuage bien positionné et uniformément réparti (figure 2.4).

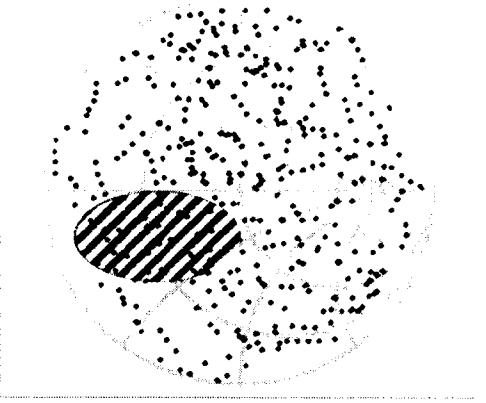


Figure 2.2 Nuage de points mal positionné sur la zone ciblée.

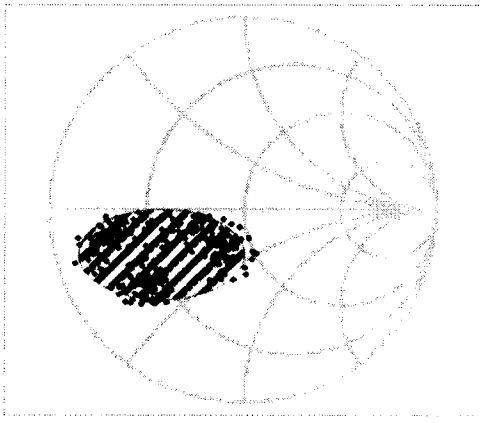


Figure 2.3 Nuage de points bien positionné sur la zone cible, mais sans uniformité.

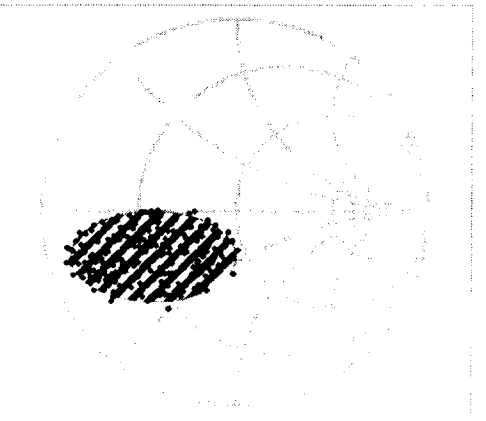


Figure 2.4 Nuage de points uniformément positionné sur la zone cible.

Étant donnée la nature à priori subjective d'une telle qualité, il est nécessaire de concevoir des outils capables de quantifier ces deux qualités afin d'éliminer la variabilité d'un jugement porté par une personne humaine et surtout d'avoir la possibilité d'automatiser ce processus de décision durant l'optimisation.

2.2.1 Méthode de quantification

Dans le but de quantifier ces deux qualités, la zone à adapter est segmentée en sous-régions à partir desquelles sont extraites les statistiques du nombre de points moyen par sous-région. Dans le cas du syntoniseur d'Ultra Électronique pour la bande 3, soit de 1350 MHz à 1850 MHz, la zone d'impédances à adapter présentée par l'antenne est contenue à l'intérieur du cercle $\Gamma = 0{,}5$ tel qu'illustré à la figure 2.5.

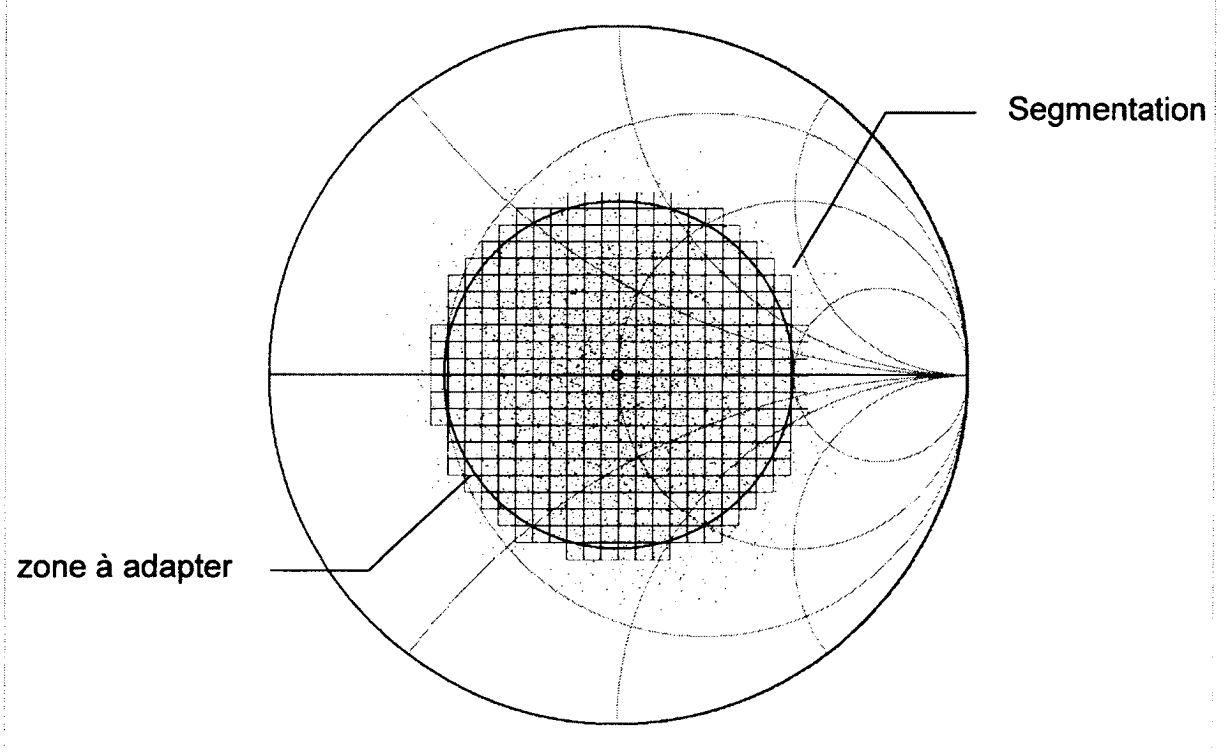


Figure 2.5 Zone cible à adapter, segmentation de la zone cible et nuage de 4096 points d'impédances générés par la simulation d'un syntoniseur.

De cette manière, la moyenne du nombre de points par sous région ($\overline{nb\_pts}$) est l'indicateur du ciblage de la zone par le nuage de points. En effet, plus la moyenne s'approche de sa valeur plafond $\overline{nb\_pts}_{MAX}$, plus les points sont contenus à l'intérieur de la zone à adapter.

$$\overline{nb\_pts} = \frac{1}{nb\_regions} \sum_{i=1}^{nb\_regions} nb\_pts_i \tag{2.1}$$

$$\overline{nb\_pts}_{MAX} = \frac{2^{Nb\_stubs}}{Nb\_regions} \tag{2.2}$$

La moyenne ne peut servir d'unique critère d'évaluation d'un nuage d'impédances puisqu'elle indique seulement la proportion de points se trouvant à l'intérieur de la zone. Elle ne donne aucune information sur la répartition des points à l'intérieur de celle-ci.

L'écart-type (s_{nb_pts}), donnant la dispersion du nombre de points par sous région (nb_pts_i) autour de la moyenne ($\overline{nb\_pts}$), fournit pour sa part une information pertinente sur la répartition du nuage. Plus il est petit, plus la répartition des points dans la zone est uniforme alors que s'il est de valeur élevée, les points sont alors dispersés en grappes plus denses, tel que schématisé à la figure 2.3.

$$s_{nb\_pts} = \sqrt{\frac{1}{nb\_regions} \sum_{i=1}^{nb\_regions} \left(nb\_pts_i - \overline{nb\_pts}\right)^2} \tag{2.3}$$

2.2.1.1 Densité de segmentation

Un des axes de liberté de cette méthode de quantification est la densité de la segmentation, où le nombre de sous-régions qui subdivisent la zone cible. Cette variabilité n'introduit pas de distorsions dans la mesure du ciblage ($\overline{nb\_pts}$) mais a un effet sur la mesure

d'uniformité de la distribution des points dans la zone (s_{nb_pts}). Plus, cette segmentation est fine, plus elle assure une mesure précise de l'uniformité de la distribution, mais aussi, plus elle nécessite de calculs.

Lors d'évaluations comparatives basées sur les mêmes nuages de points, plusieurs mesures furent prisent avec différentes densités de grilles. La figure 2.6 démontre l'effet quasi linéaire de l'augmentation de la densité sur la charge de calcul.

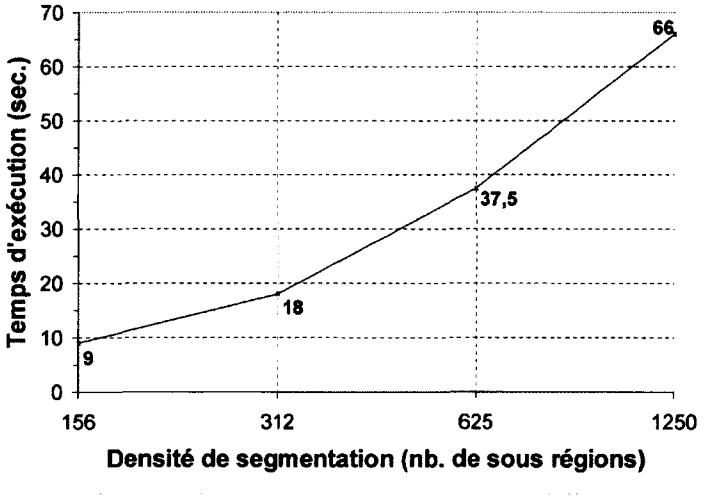


Figure 2.6 Relation du temps d'évaluation en fonction de la densité de la grille d'évaluation.[1]

Il a aussi été constaté qu'une augmentation très importante de l'efficacité de calcul est introduite pour ce type de calculs avec la version 7.4 (R2007a) de Matlab. Puisque la majorité des calculs sont effectués de manière procédurale et que Matlab est à la base optimisé pour les calculs matriciels, il est possible que ce gain de performance soit entraîné

[1] Temps d'exécution mesuré sur un ordinateur Windows XP Professionnel à processeur AMD Athlon64 3200+ cadencé à 2.0GHz avec 1.25GB de mémoire RAM de type PC2700 (DDR333).

par une optimisation de l'exécution de code procédural dans la version 7.4. Comme le démontre la figure 2.7, un gain de facteur 6 peut être obtenu en utilisant cette plus récente version.

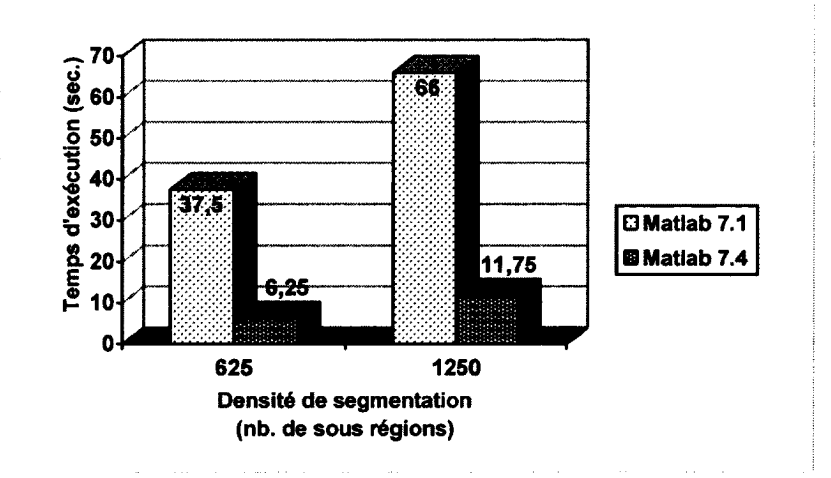


Figure 2.7 Comparatif des temps d'exécution de la routine d'évaluation des versions 7.1 et 7.4 de Matlab.[2]

2.2.1.2 Types de grilles de segmentation

Un autre paramètre variable lors de l'évaluation de la répartition est la géométrie des sous-régions. Différents types de grilles de segmentation génèreront différentes mesures de dispersion. Dans le cadre de ce projet, plusieurs types de grilles ont été développés dans l'objectif de trouver la grille qui générerait les résultats les plus fidèles à la réalité tout en nécessitant une faible charge de calcul. Les trois types de grilles développés sont la grille rectangulaire (figure 2.8a), la grille radiale (figure 2.8b) et la trame de cercles (figure 2.8c). La trame de cercles fut développée puisqu'elle semblait initialement imposer une moindre charge de calcul. En effet, puisqu'elle n'implique qu'une seule comparaison scalaire pour

[2] Temps d'exécution mesuré sur un ordinateur Windows XP Professionnel à processeur AMD Athlon64 3200+ cadencé à 2.0GHz avec 1.25GB de mémoire RAM de type PC2700 (DDR333).

tester si un point d'impédance est membre de sa région ou non, soit la distance du point en comparaison au rayon de la sous-région. La grille simple quant à elle fut développée puisque sa géométrie laissait aussi croire que sa charge de calcul serait aussi relativement faible. Cependant, puisque la grille ne couvre par très bien la zone aux extrémités, la grille radiale semblait être la grille qui s'adaptait le mieux à une zone de couverture de géométrie ronde, au risque d'augmenter la charge de calcul.

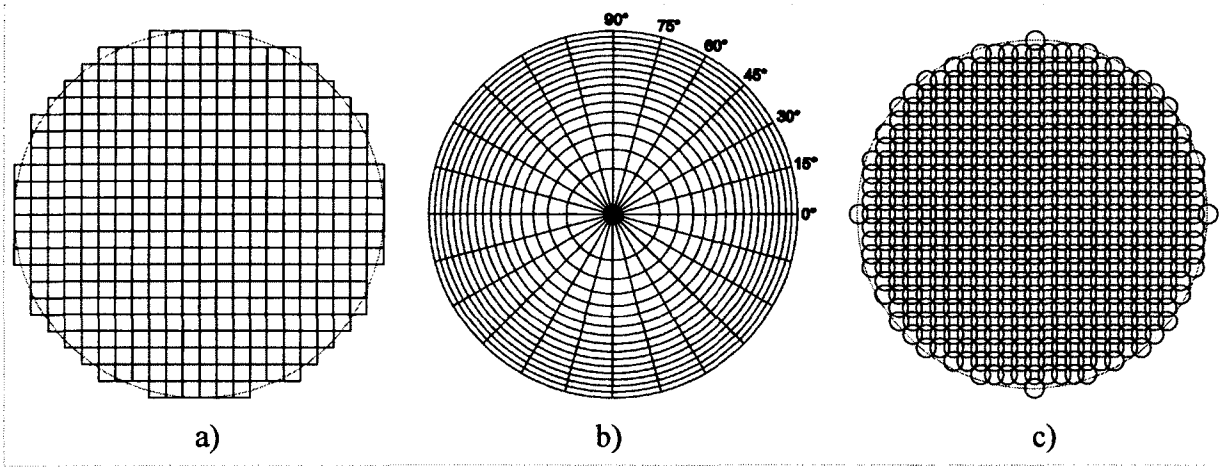


Figure 2.8 Types de grilles de segmentation : a) Grille rectangulaire b) Grille radiale c) Trame de cercles.

Toutes ces grilles ont étés programmées de manière à ce que les sous-régions soient toutes d'égale superficie peu importe le type ou la densité choisie afin de ne pas prioriser certaines répartitions. En effet, si une grille avait démontré une segmentation à densité plus élevée dans une certaine région, alors les configurations de circuit générant des nuages de points ayant une plus grande concentration de points dans cette même région auraient étés favorisés lors de l'évaluation. C'est le cas de la grille radiale qui, si elle est trop simplement implémentée, risque de favoriser les nuages ayant une plus grande densité de points au centre de la zone.

Après des tests initiaux, la segmentation par trame de cercles présentée à la figure 2.8c fut écartée puisqu'elle ne constitue pas une grille valide de par le chevauchement de ses sous-régions. En effet, il a été observé que cette grille privilégiait les configurations de

circuits produisant des points d'impédances à l'intérieur de ces zones de chevauchement puisque ces points, comptés deux fois, augmentent la moyenne du nombre de points par segment.

En termes de rapidité d'évaluation, les temps requis pour les trois grilles sont plus comparables qu'initialement anticipé, avec un certain désavantage pour la trame de cercle, ce qui va (aussi) à l'encontre des prédictions initiales (voir figure 2.9 ci-dessous).

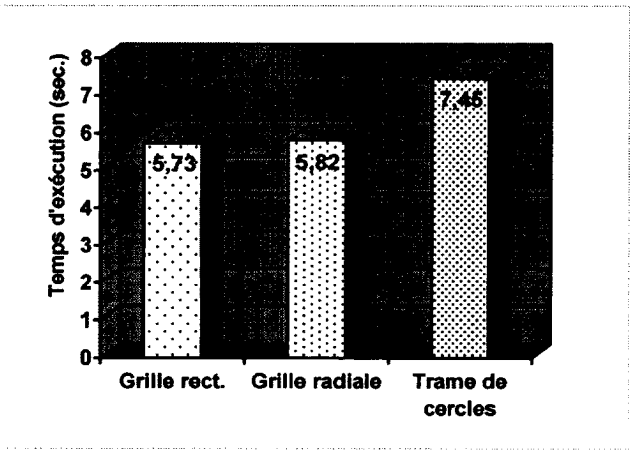


Figure 2.9 Comparatif des temps d'exécution des différents algorithmes de segmentation.

2.2.2 Pondération

Une fois les critères de sélection définis et quantifiés, il s'agit de pondérer chacun d'eux afin de procéder à la décision finale. Effectivement, comme les deux critères ne sont pas liés entre eux, il est très improbable que la configuration qui présente la meilleure uniformité offre aussi le meilleur ciblage de la zone à adapter.

La méthode de pondération élaborée présélectionne un premier groupe de configurations candidates basé sur la qualité de leur ciblage de la zone, à partir duquel est sélectionné la configuration optimale, offrant la meilleure uniformité de points dans la zone cible. Le seuil de qualification des candidats potentiels peut être fixé à un seuil absolu ou basé sur la performance relative du ciblage par rapport au meilleur du groupe. L'avantage du seuil

absolu est qu'il permet de garantir que les configurations candidates auront un nombre au moins égal à X points à l'intérieur de la zone ciblée quitte à n'en choisir aucun, alors que le seuil relatif trouvera toujours le meilleur candidat d'un lot même si celui-ci est composé de candidats de moindre qualité. Dans le cadre de cette recherche, le seuil choisi est de type relatif et à été fixé à 7,5 %, c'est-à-dire que les candidats atteignant au moins 92,5 % des performances de ciblage du meilleur candidat sont présélectionnés pour l'évaluation d'uniformité.

CHAPITRE 3

SOLUTIONS PROPOSÉES ET SIMULATIONS

Afin d'appliquer les méthodes développées et présentées au chapitre précédent, plusieurs configurations de circuits et composantes ont subit une évaluation initiale afin d'apprécier leur potentiel en tant que solutions possibles. Ce chapitre présente ces configurations potentielles, leurs résultats préliminaires de simulation ainsi que la configuration « gagnante » et ses résultats de simulations plus avancées.

3.1 Conception et évaluation des configurations retenues

L'inspiration du style de configuration idéal est issue de différentes sources : des configurations de syntoniseurs existants (ajustables et non-ajustables), d'articles scientifiques ou purement de l'imagination basée sur les connaissances et les expériences passées. Identifier, parmi toutes ces configurations possibles, les styles de configurations potentiellement candidates a d'abord nécessité une période d'expérimentation avec ADS afin d'observer leur potentiel.

3.1.1 Effets des dimensions

Au départ, nous confirmons avec ADS la théorie présentée à la section 1.2.2 du chapitre 1, qui stipule qu'un tronçon en parallèle sur une ligne de transmission terminée par un circuit ouvert se comporte comme un condensateur lorsque sa longueur est moins de 90°, et comme une inductance lorsque supérieure à 90° mais inférieure à 180°, tel qu'illustré à la figure 3.1.

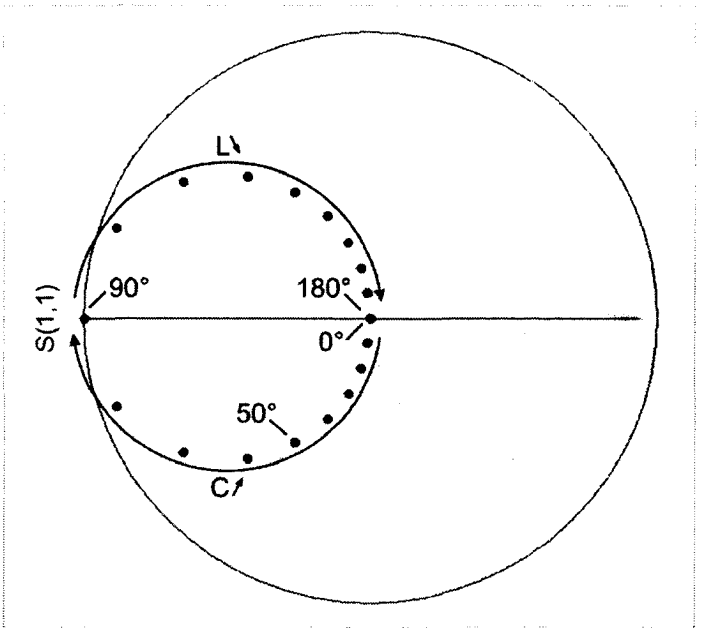


Figure 3.1 Impédances d'une ligne de transmission intégrant un tronçon parallèle en circuit ouvert de différentes longueurs électriques et leur équivalence en éléments finis.

Nous confirmons aussi que déplacer un tronçon le long d'une ligne de transmission de la source vers la charge déplace effectivement son point d'impédance dans le sens horaire autour du point d'impédance caractéristique, soit le centre de l'abaque. En effet, augmenter la distance entre la source et le tronçon augmente l'angle de déphasage du signal retourné et correspond à une rotation du point du coefficient de réflexion dans le plan complexe sur lequel se superpose l'abaque de Smith. Sur la figure 3.2, l'impédance présentée par la ligne de transmission un tronçon de 50° de longueur électrique, aussi visible à la figure 3.1, subit une rotation autour de Z_0 au fur et à mesure que le tronçon s'éloigne de la source. Il est d'ailleurs possible de constater que le déphasage introduit au signal de retour (S11) est de 360° lorsque le tronçon se situe à une demi-longueur d'onde de la source puisque le signal doit parcourir la distance dans les deux sens.

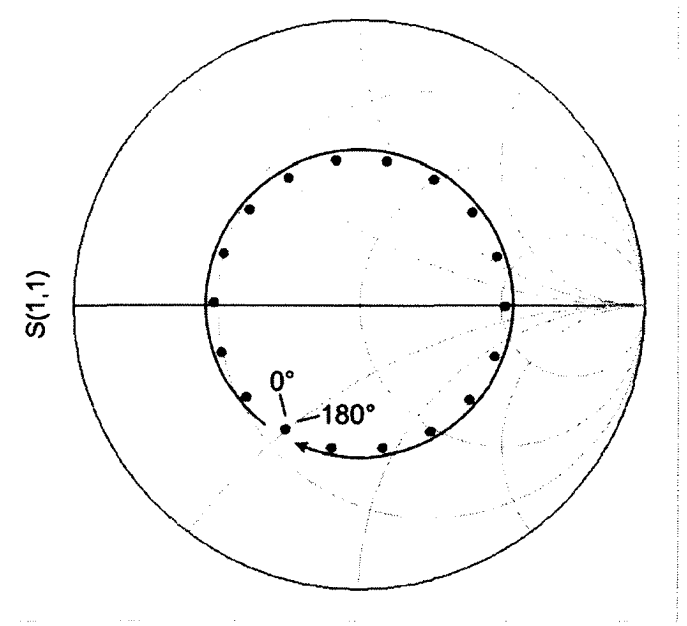


Figure 3.2 Impédances d'un tronçon parallèle en circuit ouvert de 50° de longueur électrique à différentes distances de la source.

Une fois ces simples constats vérifiés, il est dès lors possible d'établir rapidement les dimensions approximatives des éléments d'un circuit. Par exemple, si la surface à adapter est centrée sur Z_0 alors il serait nécessaire que la longueur totale du syntoniseur soit supérieure à 180° afin de couvrir toute la circonférence de la zone. Aussi, plus la surface à adapter est grande, plus les tronçons devraient s'approcher de la longueur de 90° pour que le nuage de points s'étende jusqu'au bord de la zone à adapter.

Sachant que l'impédance des structures micro-ondes est cyclique en fonction de leur longueur, il est aussi possible d'utiliser des structures étant quelques demi-longueurs d'onde plus grandes. Cependant, en plus d'augmenter les pertes introduites, ces structures réduisent la largeur de bande de l'adaptation puisque plus sélectives (Pozar, 2005).

3.1.2 Configurations évaluées

Un aspect important à considérer lors du choix des configurations qui seront évaluées est de sélectionner des configurations de complexité identique, c'est-à-dire qu'il faut que toutes les

configurations puissent prendre le même nombre d'états. Le syntoniseur développé par M. Christian Talbot (Talbot, 2006) dans le cadre d'un précédent projet de recherche comptait 4096 (2^{12}) états et offrait une couverture adéquate sans pour autant imposer une trop grande charge de calcul lors des simulations. C'est donc ce même degré de complexité qui a été visé lors de la conception des configurations présentées dans cette section à des fins de comparaison et d'évaluation de potentiel. De plus, toutes les simulations initiales furent conduites à l'aide de sections de lignes de transmission parfaites disponibles dans ADS où le paramètre de longueur est saisi en distance électrique (degrés) et est donc sans égards à la longueur réelle pour cette étape.

3.1.2.1 Trois et quatre branches à tronçons commutables distribués

Deux configurations à géométries complexes, soit 12 tronçons commutables distribués sur trois (figure 3.3) ou quatre (figure 3.5) branches en parallèles avec la ligne de transmission principale, tentant de générer des nuages d'impédances très étendus.

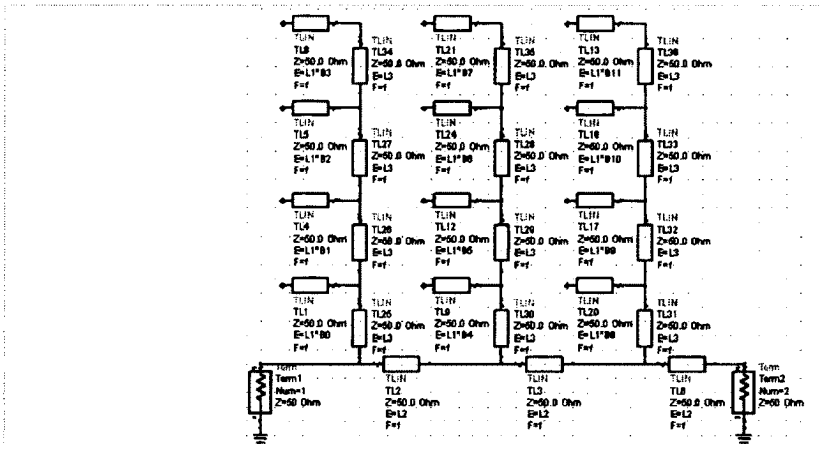


Figure 3.3 Configuration trois branches à tronçons commutables distribués.

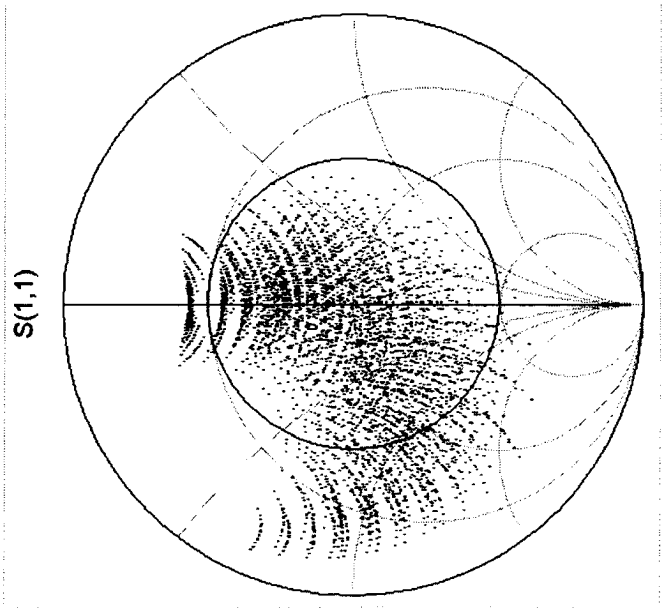


Figure 3.4 Meilleur résultat des simulations initiales sur la configuration trois branches à tronçons commutables distribués à 1.55 GHz.

Mesure de ciblage (moyenne) : 4,79

Mesure d'uniformité (écart-type) : 4,11

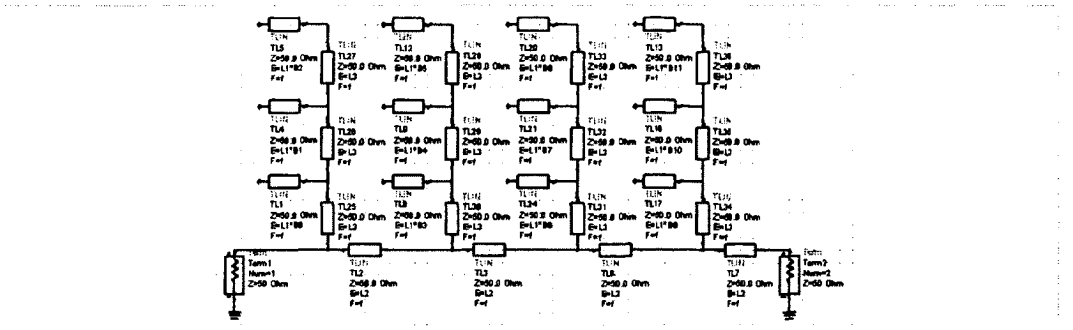


Figure 3.5 Configuration quatre branches à tronçons commutables distribués.

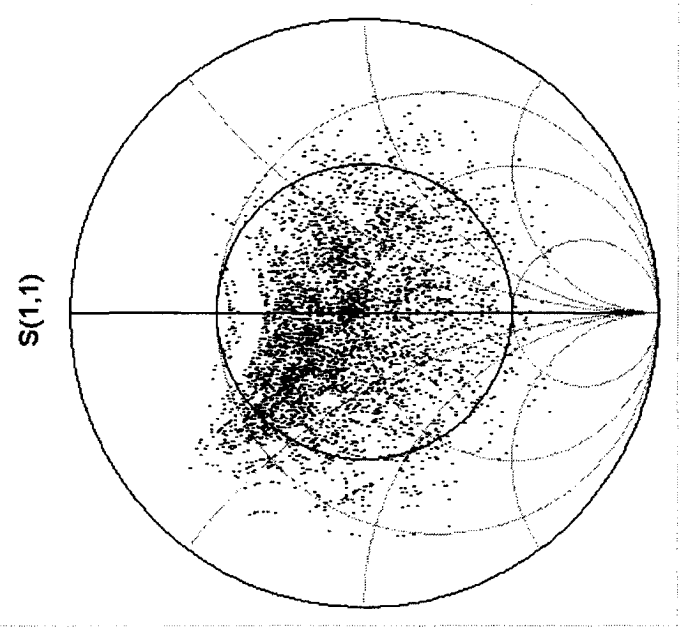


Figure 3.6 Meilleur résultat des simulations initiales sur la configuration quatre branches à tronçons commutables distribués à 1.55 GHz.

Mesure de ciblage (moyenne) : 5,59

Mesure d'uniformité (écart-type) : 4,79

Les résultats de simulations initiales montrent que l'objectif de générer un nuage d'impédances très étendu est difficilement atteint avec un ciblage acceptable mais non-uniforme tel que l'illustre les figure 3.4 et figure 3.6.

3.1.2.2 Lignes parallèles à interconnexions distribuées

Cette configuration, illustrée par la figure 3.7, est basée sur un article de journal (Minnis, 1987) où l'auteur décrit un syntoniseur ajustable manuellement où deux lignes de transmissions en parallèles sont interconnectées par deux ponts placés manuellement en deux points judicieusement choisis. La figure 3.8 révèle une bonne uniformité, cependant les performances de ciblage dans le présent contexte sont décevantes.

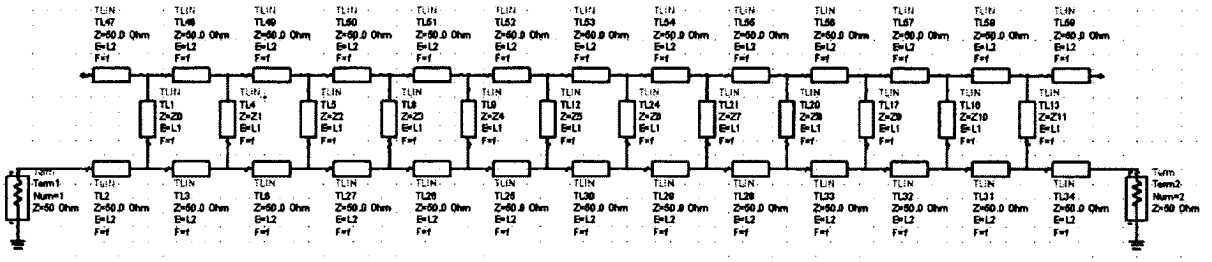


Figure 3.7 Configuration lignes parallèles à interconnections distribuées.

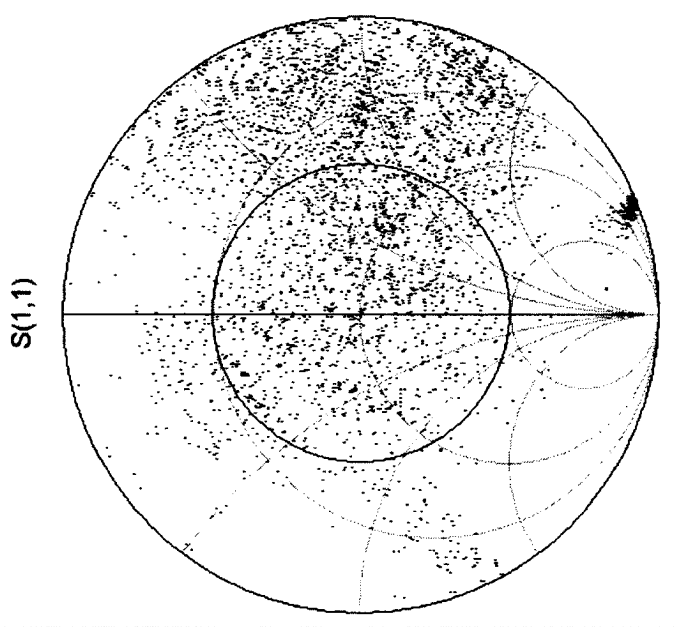


Figure 3.8 Meilleur résultat des simulations initiales sur la configuration lignes parallèles à interconnections distribuées à 1.55 GHz.

Mesure de ciblage (moyenne) : 2,34

Mesure d'uniformité (écart-type) : 2,03

3.1.2.3 Six tronçons distribués de longueur commutable

Cette configuration, illustrée à la figure 3.9, est inspirée du syntoniseur à double tronçons présenté à la section 1.2.2. Six tronçons distribuées au long de la ligne de transmission pouvant prendre 4 longueurs différentes (0, L1, 2×L1, 3×L1) chacun. La figure 3.10 et les mesures révèlent un excellent ciblage et une bonne uniformité.

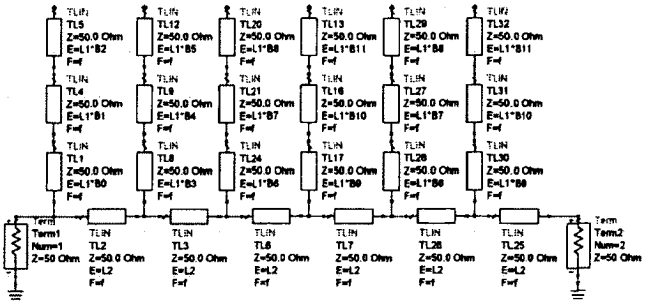


Figure 3.9 Configuration six tronçons distribués de longueur commutable.

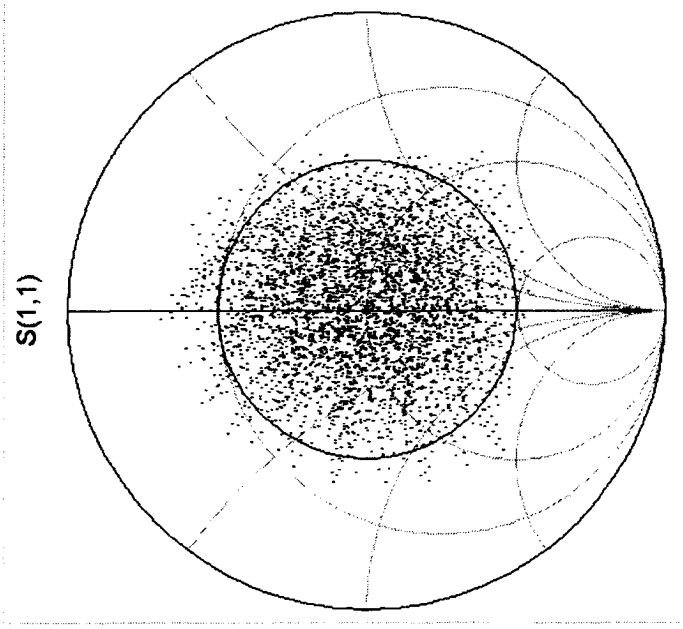


Figure 3.10 Meilleur résultat des simulations initiales sur la configuration six tronçons distribués de longueur commutable à 1.55 GHz.

Mesure de ciblage (moyenne) : 6,11

Mesure d'uniformité (écart-type) : 4,00

3.1.2.4 Douze tronçons commutables distribués

Cette configuration présentée à la figure 3.11 est basée sur le syntoniseur de (Talbot, 2006) et présente d'excellentes performances, autant en ce qui à trait au ciblage qu'à l'uniformité de la distribution d'impédances révélés par les mesures et la figure 3.12.

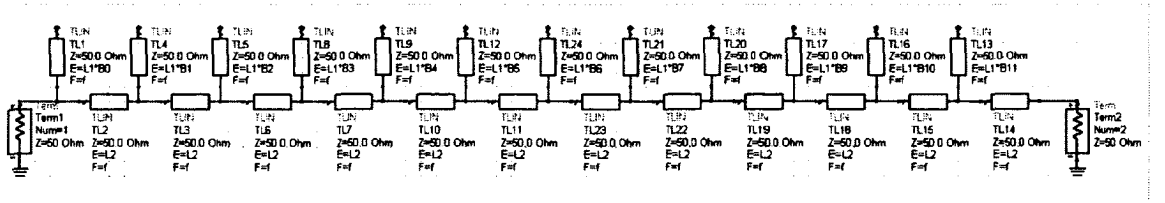


Figure 3.11 Configuration 12 tronçons commutables distribués.

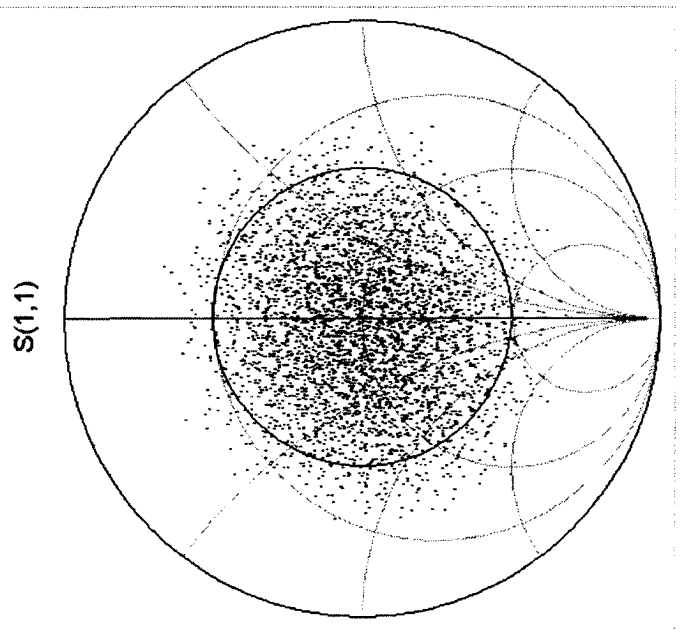


Figure 3.12 Meilleur résultat des simulations initiales sur la configuration 12 tronçons commutables distribués à 1.55 GHz.

Mesure de ciblage (moyenne) : 5,97

Mesure d'uniformité (écart-type) : 3,40

3.1.3 Sélection de la meilleure configuration

Considérant les résultats obtenus lors des simulations initiales, deux configurations se démarquent du lot :

- configuration à six tronçons distribués de longueur commutable,
- configuration à 12 tronçons commutables distribués.

La qualité de leur couverture étant très similaire il est alors nécessaire de comparer leurs performances aux limites de la bande de fréquence visée, soit à 1,35 GHz et 1,85 GHz afin de déterminer laquelle offre la meilleure couverture sur toute la bande. Les figure 3.13 et figure 3.14 illustrent les résultats de simulation des deux configurations à 1,35 GHz et la figure 3.15 et figure 3.16 à 1,85 GHz.

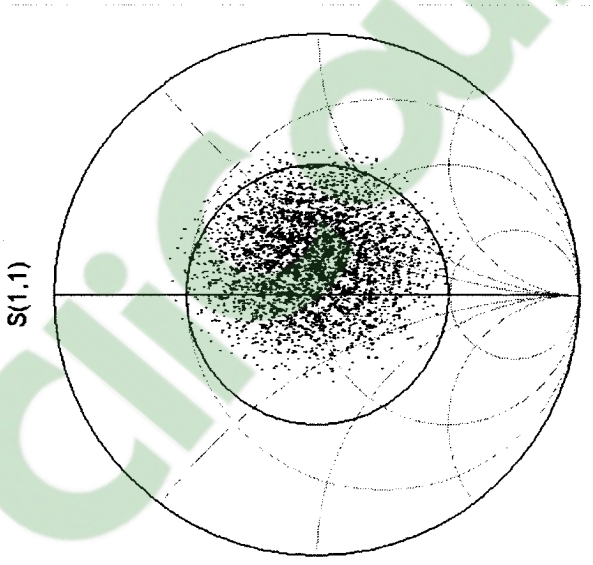


Figure 3.13 Répartition des impédances de la configuration six tronçons distribués de longueur commutable à 1,35 GHz.

Mesure de ciblage (moyenne) : 6,24

Mesure d'uniformité (écart-type) : 6,85

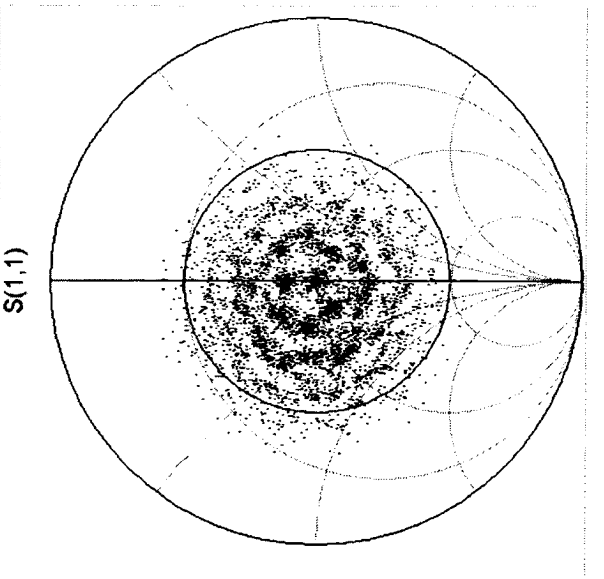


Figure 3.14 Répartition des impédances de la configuration 12 tronçons commutables distribués à 1,35 GHz.

Mesure de ciblage (moyenne) : 6,39

Mesure d'uniformité (écart-type) : 6,65

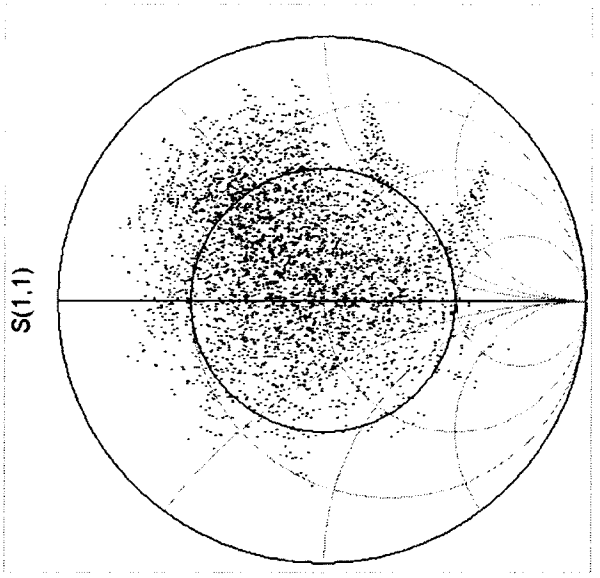


Figure 3.15 Répartition des impédances de la configuration six tronçons distribués de longueur commutable à 1,85 GHz.

Mesure de ciblage (moyenne) : 4,49

Mesure d'uniformité (écart-type) : 3,10

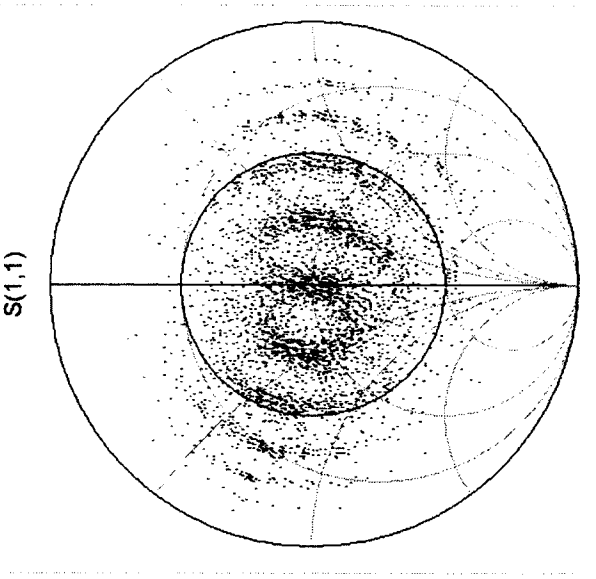


Figure 3.16 Répartition des impédances de la configuration 12 tronçons commutables distribués à 1,85 GHz.

Mesure de ciblage (moyenne) : 5,24

Mesure d'uniformité (écart-type) : 5,02

Cette comparaison révèle que les deux configurations offrent, encore une fois, sensiblement les mêmes performances avec un léger avantage pour la configuration à 12 tronçons commutables distribués en ce qui a trait au ciblage de la zone. Ces résultats ne permettent pas de décider clairement d'une architecture gagnante. Puisque les deux semblent bonnes, d'autres critères de sélection sont considérés.

La conception de la configuration à six tronçons à longueur commutable nécessite 18 interrupteurs RF contre 12 pour sa concurrente. Le nombre accrus d'interrupteurs RF et, plus généralement, de composantes dans un circuit micro-ondes comporte plusieurs désavantages : augmentation des pertes d'insertion (S21), accroissement des risques de défaillance et augmentation des coûts de fabrication.

Pour toutes ces raisons, la configuration à 12 tronçons distribués fut sélectionnée pour les étapes subséquentes de développement.

3.2 Choix de la composante de commutation micro-ondes

Pour commuter électroniquement un signal micro-ondes, quelques alternatives sont offertes : relais micro-ondes, interrupteur MEMS RF, diode p-i-n et transistor FET. Cependant, lorsque nous imposons une restriction physique sur la taille relative du commutateur par rapport à la longueur d'onde, les relais micro-ondes ne peuvent être considérés puisqu'ils sont plusieurs fois plus grands que la fraction de longueur d'onde que doit mesurer le tronçon. Le tableau 3.1 expose les différentes caractéristiques des alternatives valides. L'interrupteur pHEMT est le AWS5532R de Anadigics, la diode p-i-n est la diode HPDN-4038 de Avago Technologies et finalement l'interrupteur MEMS est le TT2214 de la compagnie Teravicta.

Tableau 3.1

Comparatif des composantes de commutation micro-ondes.

	Interrupteur pHEMT	Diode p-i-n	Interrupteur MEMS RF
Résistance à l'état actif (R_S)	1,2 Ω	1,5 Ω	0,3 Ω
Capacitance parasite (C_P)	0,32 pF	0,055 pF	0,33 pF (inactif)[3] 0 pF (actif)[3]
Fréquence maximale	2.5 GHz	20 GHz	7 GHz
Figure de mérite (*FDM*)	414 GHz	1929 GHz	1608 GHz
Puissance acceptée à l'état actif	10 W	7.5 W	15 W (continu) 30 W (instantané)
Isolation @ 1.5 GHz	-29 dB	-25 dB	-22 dB
Complexité du circuit de polarisation	Élevée	Faible	Moyenne

[3] Selon Hand, T., et S. Cummer. 2007. « Characterization of Tunable Metamaterial Elements Using MEMS Switches ». Antennas and Wireless Propagation Letters, IEEE, vol. 6, p. 401-404.

La figure de mérite, tiré de (Gutmann et Fryklund, 1987), généralement utilisée est présentée ci-dessous et sert à comparer les composantes de commutation micro-ondes. Elle prend en considération les deux paramètres déterminants dans la performance d'une composante de commutation RF, soit la résistance à l'état « fermé », R_S, et la capacité parasite à l'état « ouvert », C_P :

$$FDM = \frac{1}{2\pi R_S C_P}. \tag{3.1}$$

Étant donné que les trois composantes comparées répondent toutes aux exigences en fréquence et en puissance de l'application et que l'isolation est comparable, les paramètres critiques sur lesquels la décision est basée sont donc les pertes par insertion. En effet, dans une application de syntoniseur où le signal reçu peut être extrêmement faible, toute perte de puissance doit être évitée. Dans cette optique, l'interrupteur MEMS qui a la plus faible résistance R_S (correspond à une perte d'insertion de 0,07 dB à 1,55 GHz) est la composante sélectionnée pour les simulations subséquentes et la fabrication.

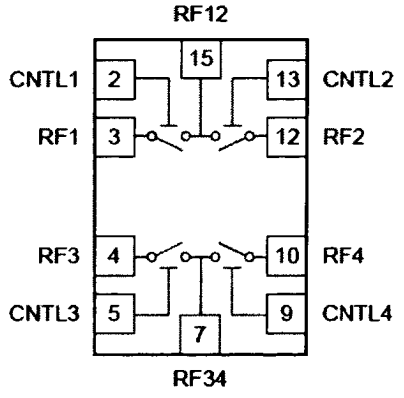


Figure 3.17 Diagramme fonctionnel de l'interrupteur micro-ondes TT2214 de Teravicta.

Source : Teravicta, 2007.

Chaque interrupteur TT2214 comporte deux commutateurs bidirectionnels de type MEMS à contacts ohmiques. Afin d'éviter une topologie de circuit complexe et les interférences

coplanaires, seul un pole et un commutateur par composante sont utilisés pour ce prototype. Les broches de contrôle identifiées par « CNTL » requièrent une tension de 68 V afin de commuter un interrupteur à l'état actif. Étant donnée la nature électrostatique du mécanisme MEMS, aucun courant n'est consommé par la broche de contrôle, ce qui signifie que la source de tension doit simplement fournir assez de courant pour polariser le circuit de contrôle et charger les capacités parasites induites par les traces entre la source et les MEMS. Avec un circuit de contrôle par interrupteur MEMS, la source de tension, provenant elle aussi de la compagnie Teravicta, peut s'accommoder de 30 interrupteurs grâce à un courant maximal de sortie de 3 mA.

3.3 Choix du substrat

Une autre variable à fixer avant de conduire les simulations finales est le type de substrat utilisé afin de minimiser les pertes par insertion dans les lignes de transmission micro-rubans. Après évaluation des familles de substrats disponibles, celle offrant les caractéristiques les mieux adaptées aux applications micro-ondes est la famille de substrat téflon-céramique RO3000 de Rogers Corporation. Cette famille offre une gamme de substrats avec différentes constantes diélectriques (ε_r) adaptés aux applications micro-ondes offrant une excellente stabilité des propriétés diélectriques en fréquence et en température. De plus, ces caractéristiques diélectriques sont uniformes le long de la surface du substrat.

Trois substrats aux constantes diélectriques différentes sont offerts dans cette famille : RO3003 ($\varepsilon_r = 3{,}00$), RO3006 ($\varepsilon_r = 6{,}15$) et RO3010 ($\varepsilon_r = 10{,}20$). Le tableau 3.2 compare les dimensions et les pertes d'une ligne micro-ruban 50 Ω sur ce type de substrat pour différentes valeurs de constantes diélectriques et d'épaisseur dans l'objectif de trouver la meilleure solution de fabrication. La largeur d'une ligne de transmission présentant une impédance de 50 Ω est donnée par la première valeur de chaque case et est en millièmes de pouces (mil). Les pertes introduites sont données sur la deuxième ligne de chaque case et est donné en décibel par longueur d'onde (dB / λ_g). Ces valeurs proviennent de simulations à l'aide de l'outil LineCalc disponible dans ADS.

Tableau 3.2

Comparaison des largeurs de traces micro-rubans
et des pertes sur différents substrats.

Type / **Épaisseur**	RO3003 ($\varepsilon_r = 3{,}00$)	RO3006 ($\varepsilon_r = 6{,}15$)	RO3010 ($\varepsilon_r = 10{,}20$)
5 mil	10,61 mil 0,398 dB / λ_g	5,74 mil 0,420 dB / λ_g	3,25 mil 0,450 dB / λ_g
10 mil	22,68 mil 0,239 dB / λ_g	12,71 mil 0,257 dB / λ_g	7,58 mil 0,278 dB / λ_g
20 mil	46,78 mil 0,153 dB / λ_g	N.D.	N.D.
25 mil	N.D.	33,78 mil 0,140 dB / λ_g	20,93 mil 0,160 dB / λ_g
30 mil	70,07 mil 0,123 dB / λ_g	N.D.	N.D.
50 mil	N.D.	66,79 mil 0,107 dB / λ_g	42,13 mil 0,116 dB / λ_g
60 mil	133,15 mil 0,092 dB / λ_g	N.D.	N.D.

Étant donnée les dimensions de l'interrupteur MEMS, la largeur de trace maximale pour accéder à l'interrupteur est de 50 millièmes de pouces. Le substrat permettant de rencontrer cette contrainte tout en minimisant les pertes par insertion est le RO3010 de 0,050" d'épaisseur.

3.4 Simulations

Suite à la sélection de la configuration à 12 tronçons sur substrat RO3010 0,050" et de l'interrupteur MEMS TT2214 de Teravicta comme étant la combinaison offrant les meilleures perspectives en termes de performances, il est nécessaire d'exécuter des simulations plus poussées avant de passer à la fabrication. Ces simulations doivent intégrer ces éléments afin d'être le plus fidèle possible à la réalité et ainsi minimiser l'erreur entre les résultats simulés et mesurés.

Étant donnée l'approche empirique de ces travaux, un grand nombre de simulations doit être exécuté pour générer un résultat valide. Afin d'éviter que les lots de simulation ne prennent trop de temps à exécuter, il est préférable d'exécuter des lots de simulation progressivement de plus en plus complexes en utilisant les valeurs trouvées au lot précédent comme point de départ pour le suivant. Ainsi, les premières simulations qui doivent être exécutées en plus grand nombre afin de couvrir une plus grande plage de valeurs sont individuellement moins longues à exécuter. Les simulations finales qui doivent être le plus réaliste possible sont plus longues à exécuter mais convergent avec moins d'itérations puisque leurs paramètres physiques initiaux sont plus proches de la solution.

3.4.1 Simulations à complexité moyenne

Pour ces simulations, les lignes de transmissions de type micro-ruban ainsi que leur substrat ont été intégrés au circuit simulé grâce au modèles disponibles dans ADS à l'intérieur de la boîte d'outils « TLines-Microstrip ». Les interrupteurs MEMS sont modélisés et intégrés à la simulation par leurs paramètres-S mesurés et fournis par Teravicta.

Donc, à partir des résultats initiaux de comparaison de configurations exposés à la section 3.1.2.4, il est possible d'établir les grandeurs physiques approximatives de départ de ces simulations à complexité moyenne. Soit 188 mil (23°) pour la longueur des tronçons et 572.5 mil (70°) entre chacun d'eux. La figure 3.18 présente le circuit tel que simulé sous ADS.

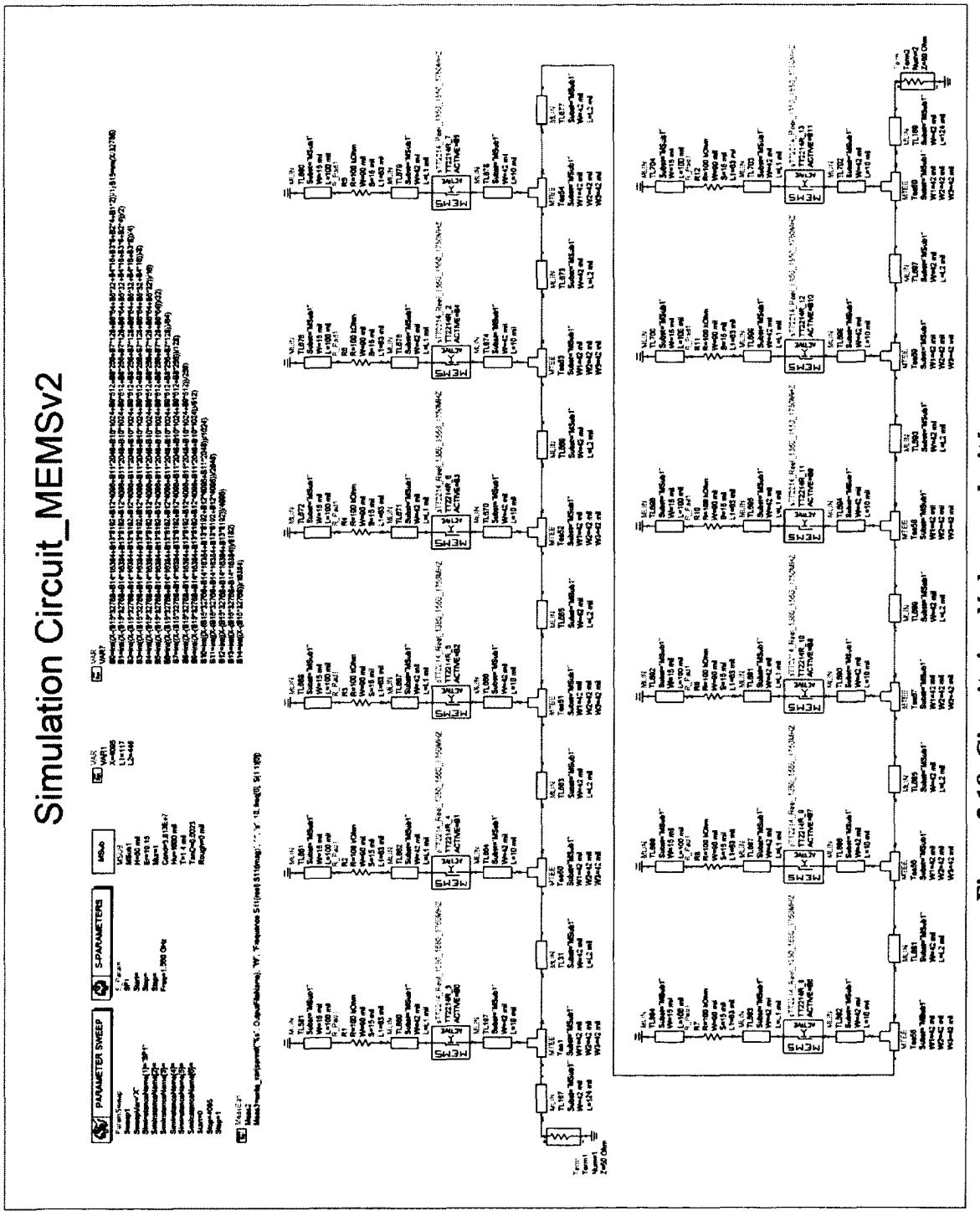


Figure 3.18 Circuit simulé de complexité moyenne.

Suite à l'exécution de nouveaux lots de simulation menant à l'optimisation des grandeurs physiques du circuit, soit des tronçons d'une longueur de 117 mil séparés par 445 mil, les résultats présentés ci-dessous sont obtenus.

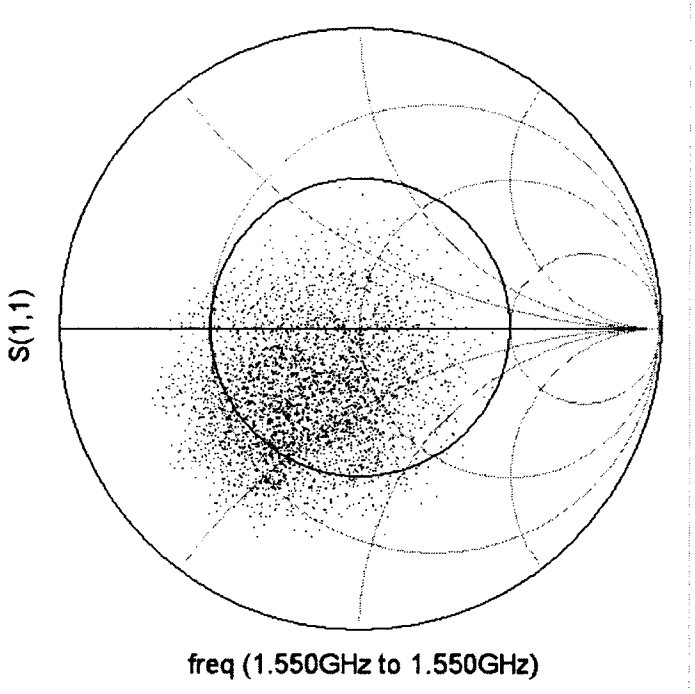


Figure 3.19 Meilleure répartition des impédances des simulations à complexité moyenne.

Mesure de ciblage (moyenne) : 5,18

Mesure d'uniformité (écart-type) : 5,75

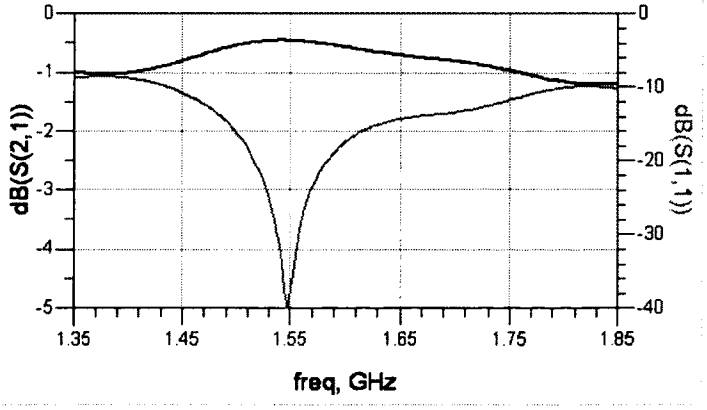


Figure 3.20 Paramètres S21 (pertes d'insertion) et paramètres S11 simulées de l'état neutre (50 Ω à 1,55 GHz).

La première observation sur la répartition des impédances à la figure 3.19 est le décentrage du nuage de points d'impédances vers la gauche et vers le bas. Cette translation s'explique par les bouts de tronçons reliant les interrupteurs MEMS à ligne de transmission principale agissant comme condensateurs parasites lorsque leurs tronçons respectifs sont déconnectés par les interrupteurs. En effet, tel qu'illustré par la figure 3.1, l'ajout de courts tronçons en parallèle a pour effet de simuler des condensateurs et déplacer l'impédance de la ligne de transmission du centre vers l'extrémité gauche de l'abaque de Smith en décrivant un arc de cercle concave dans le sens horaire.

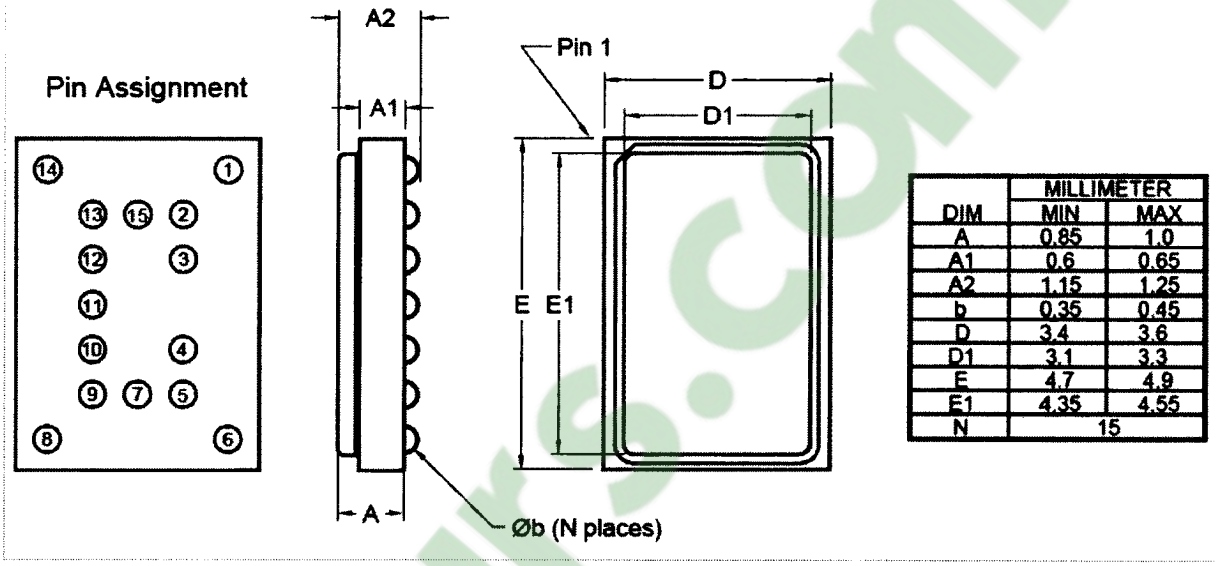
Cependant, en faisant abstraction de ce décalage, les impédances générés forment un nuage uniforme et les pertes par insertion, lorsque le syntoniseur est dans un état adapté à l'analyseur de réseaux vectoriel (ARV), soit 50 Ω, sont excellentes avec 0,462 dB à la fréquence de 1,55 GHz.

3.4.2 Simulations finales à complexité élevée.

Pour les lots de simulations finales, en plus de l'intégration des éléments du circuit réel décrits à la section 3.4.1, d'autres raffinements peuvent être apportés, tant à la configuration du circuit qu'au réalisme de la simulation.

Par exemple, en ce qui a trait à la configuration du circuit, tel que mentionné à la section précédente, le signal de la ligne principale aux interrupteurs MEMS est acheminé via un petit tronçon de ligne de transmission. En effet, ce petit tronçon agi comme condensateur parasite lorsque l'interrupteur est désactivé. Il est néfaste pour la localisation optimale du nuage d'impédances. Il est plutôt préférable de brancher directement la broche de l'interrupteur à la ligne de transmission principale et d'ainsi éliminer ces effets indésirables. Ceci signifie qu'il faut acheminer la ligne de transmission sous chacun des 12 interrupteurs puisque les composantes MEMS sont à base de céramique et que leurs broches d'interconnexions sont de type « Ball Grid Array » (BGA) (figure 3.21). Il est alors nécessaire d'adapter la largeur de la trace sous le MEMS de manière à ce que l'impédance caractéristique de la ligne de

transmission reste constante à 50 Ω malgré son nouvel environnement diélectrique illustré à la figure 3.22



DIM	MILLIMETER MIN	MAX
A	0.85	1.0
A1	0.6	0.65
A2	1.15	1.25
b	0.35	0.45
D	3.4	3.6
D1	3.1	3.3
E	4.7	4.9
E1	4.35	4.55
N	15	

Figure 3.21 Dimensions d'un interrupteur MEMS TT2214 de Teravicta.

Source : Teravicta, 2007.

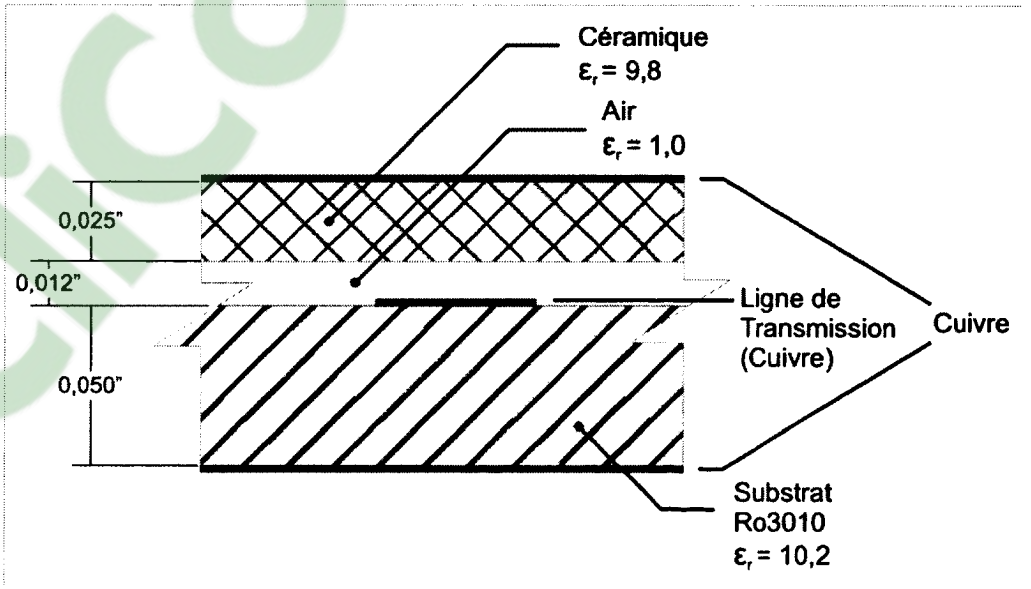


Figure 3.22 Environnement diélectrique de la ligne de transmission sous l'interrupteur MEMS.

Étant donnée la nature non homogène autour de la ligne de transmission, il est possible de déterminer, à l'aide d'outils de simulation à 2,5 ou 3 dimensions, la largeur de trace qui présenterait une impédance de 50 Ω dans ces conditions. Dans le cadre de cette recherche, le logiciel Agilent EMDS 2006C (*Electromagnetic Design System*) fut utilisé pour conduire ces simulations et optimiser la largeur de la trace.

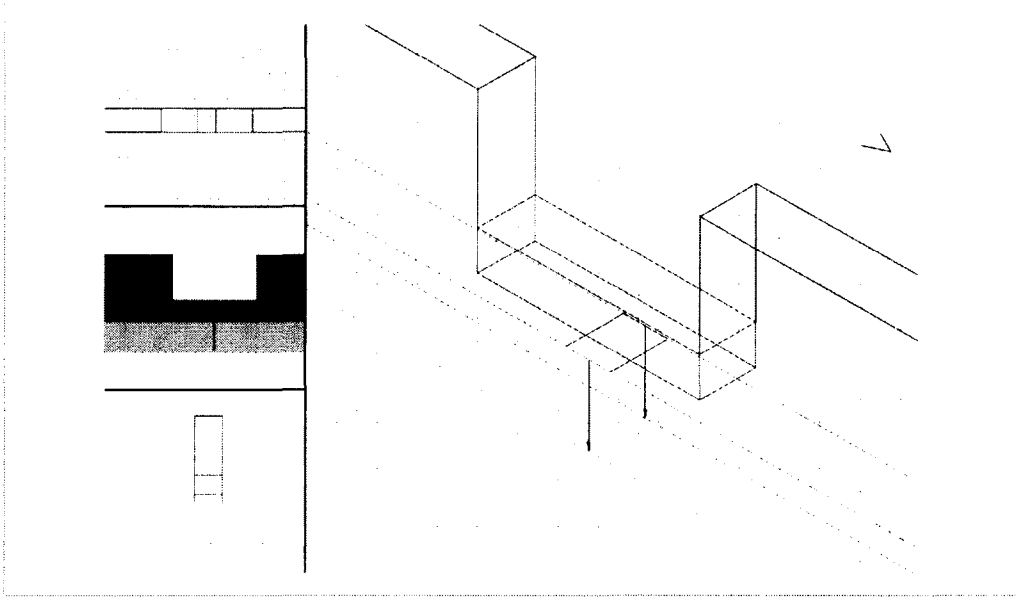


Figure 3.23 Capture d'écran de la structure simulé en vues isométriques à partir du logiciel EMDS 2006C.

Ainsi, une largeur de trace de 28 mil est requise pour obtenir une impédance de 50 Ω sous le MEMS. De plus, le logiciel EMDS a permis de déterminer qu'il est préférable de procéder à la transition de manière graduelle en biseautant la ligne de transmission entre les deux largeurs de traces afin de compenser pour les franges de champs électromagnétiques parasites. Ainsi, les lignes de transmissions se trouvant sous l'interrupteur ont étés intégrées au *footprint* de la composante ainsi qu'à son modèle dans ADS, la transformant ainsi à une composante à trois ports tel qu'observé sur la figure 3.24.

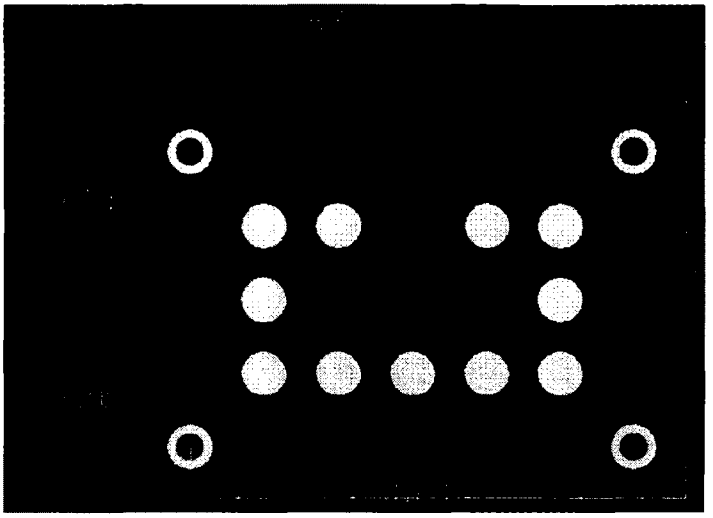


Figure 3.24 Empreinte de l'interrupteur MEMS intégrant les lignes de transmissions au dessous et autour de sa structure.

Donc, pour les derniers lots de simulation, les grandeurs physiques de départs seront celles obtenues lors du lot de simulation de moyenne complexité, soit des tronçons d'une longueur de 117 mil séparés par une distance de 445 mil. La figure 3.25 présente le circuit tel que simulé sous ADS.

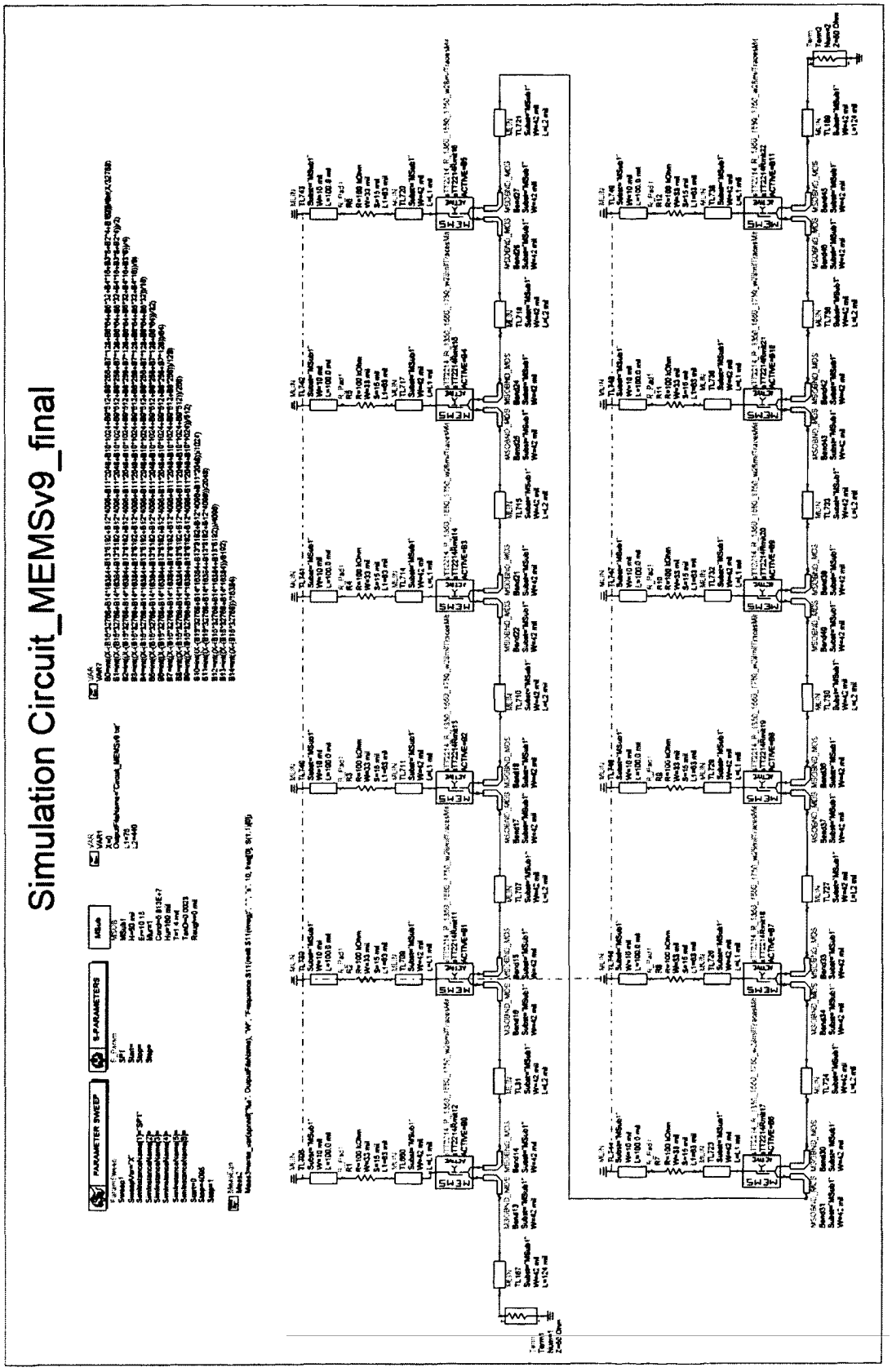


Figure 3.25 Circuit simulé de complexité élevée.

L'optimisation finale des paramètres physiques du circuit convergent vers une longueur de tronçon de 75 mil et une distance de 440 mil entre ceux-ci.

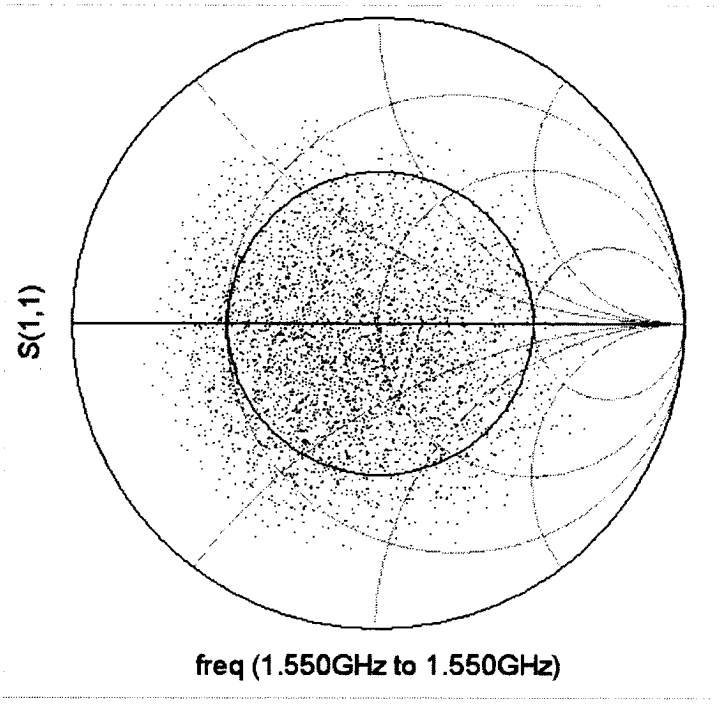


Figure 3.26 Meilleure répartition des impédances des simulations à complexité élevée.

Mesure de ciblage (moyenne) : 5,04

Mesure d'uniformité (écart-type) : 2,76

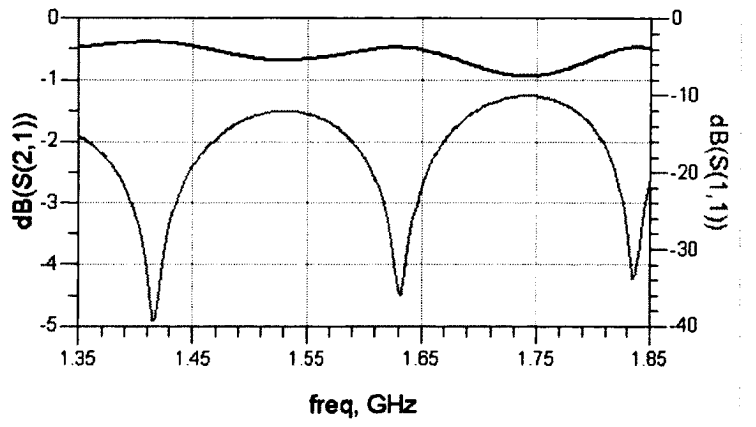


Figure 3.27 Paramètres S21 (pertes d'insertion) et paramètres S11 simulées de l'état neutre (50 Ω à 1.55 GHz).

Tel qu'anticipé, la figure 3.26 révèle que les effets parasites observés lors des simulations à complexité moyenne ont été éliminés par la modification du tracé de la ligne de transmission principale. Le nuage de répartition d'impédances résultant est bien ciblé et relativement uniforme tel que le révèlent les métriques qui sont comparables à celles mesurées lors de la première simulation idéalisée de cette configuration présenté à la section 3.1.2.4. Les pertes par insertion, c'est-à-dire S21 lorsque le syntoniseur est bien adapté (1,418 GHz, 1,633 GHz et 1,815 GHz), sont toujours très bonnes, soit moins de 0,481 dB.

C'est à partir de cette configuration et de ces résultats que le prototype est conçu (chapitre 4) et ses mesures réelles comparées (chapitre 5).

CHAPITRE 4

FABRICATION DU PROTOTYPE

4.1 Diagramme schématique

Afin de pouvoir fabriquer le circuit imprimé du prototype, le circuit de contrôle des interrupteurs MEMS reste à être défini. Ce circuit doit permettre de commander chaque interrupteur individuellement aussi bien électriquement que manuellement. Pour une question de facilité et de rapidité d'implantation, le port parallèle (standard IEEE 1284), encore disponible sur plusieurs ordinateurs personnels, a été choisi comme port d'accès pour le contrôle par commande électrique. Aussi, une banque de commutateurs DIP à été intégrée au schéma du circuit de contrôle afin de permettre une utilisation manuelle et indépendante du syntoniseur en l'absence d'un ordinateur ou autre unité de contrôle.

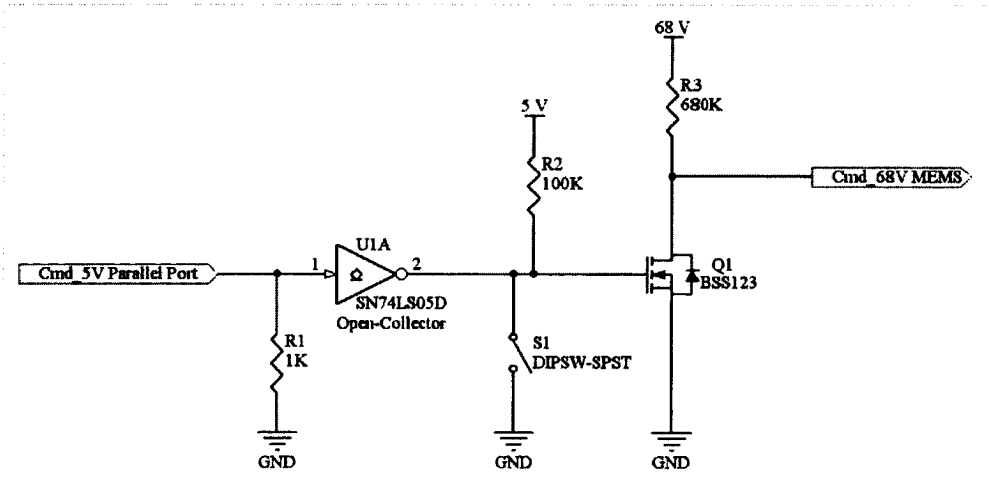


Figure 4.1 Diagramme schématique du circuit de contrôle pour un interrupteur MEMS.

Pour chacun des 12 interrupteurs MEMS, un circuit de contrôle tel que celui présenté à la figure 4.1 est intégré au circuit prototype. L'alimentation 68 V nécessaire au fonctionnement des MEMS est générée par une source de tension survolteur fabriquée par Teravicta s'alimentant sur la source de tension externe à 5 V.

4.1.1 Circuit d'interface 8 bits - 12 bits

Puisque la largeur du bus de données du port parallèle est de huit bits et que le syntoniseur, ayant 12 composantes de commutation, requiert tout autant de bits de contrôle, un circuit d'adaptation est donc requis entre ceux-ci. Un circuit à CPLD a été préalablement conçu par M. Christian Talbot (Talbot, 2006) lors de ses travaux précédents. Ce circuit servait alors de démultiplexeur entre le port parallèle et deux bus de données 12 bits, il est donc réutilisé comme interface dans le cadre de ce projet moyennant une simple reprogrammation du CPLD en langage VHDL. Le diagramme schématique, la topologie, ainsi que le programme VHDL chargé dans ce circuit d'interface peuvent être consultés à l'ANNEXE III.

4.2 Topologie

La topologie du circuit de contrôle est générée automatiquement grâce à la fonction *Auto-Route* d'Altium Designer qui s'exécute en respectant les règles de design fournies par l'utilisateur sur les capacités de fabrications. Une fois le routage terminé, quelques modifications manuelles furent nécessaires afin d'éviter certaines aberrations comme le détour trop important d'une trace ou le manque d'uniformité lors du routage de sections identiques. De plus, la topologie de circuit générée par ADS pour le circuit micro-ondes fut intégrée manuellement au reste du circuit via Altium. En effet, l'outil CAMtastic permet d'importer la topologie de circuit générée par ADS en format *Gerber* pour l'intégrer au reste du circuit et ensuite exporter le tout en fichiers *Gerbers* homogènes[4] pour fabrication. Les images des topologies des deux surfaces du circuit imprimé sont disponibles à l'ANNEXE IV.

---

[4] Format : Appertures RS-274-X, format décimal 2.5, suppression des zéros à gauche, référentiel absolu et unités en pouces.

4.3 Fabrication

Certains éléments de la topologie étant très petits et très rapprochés, la fabrication du prototype nécessitait un fabricant étant capable de rencontrer des exigences de fabrications dites « avancées ». Les plus importantes spécifications minimales à rencontrer pour la fabrication du prototype sont énumérées dans le tableau 4.1.

Tableau 4.1

Spécifications du circuit imprimé prototype.

Largeur de trace	10 mil.
Distance conducteur-conducteur	5,9 mil.
Distance annulaire interne entre via et pad	3 mil.
Distance annulaire entre pad et masque de soudure	2,5 mil.
Distance entre deux trous dans le masque de soudure	5,5 mil
Diamètre intérieur d'un via (avant plaquage)	10 mil.
Ratio épaisseur du substrat / diamètre d'un trou	5.00
Substrat	RO3010 Composé téflon-céramique
⇒ Traitement pour plaquage des via	Traitement au plasma

Suite à plusieurs soumissions par différents fabricants et sur la recommandation de M. Dubouil, ingénieur chez Ultra Electronics, la fabrication du circuit imprimé fut confiée à Metaplast Circuits Limited de la région de Toronto qui fut capable de rencontrer les exigences de fabrication pour ce prototype et possède une bonne expérience dans la fabrication de circuits micro-ondes.

4.4 Assemblage

L'assemblage du prototype a demandé une attention toute particulière au montage des interrupteurs MEMS sur le circuit imprimé. Les composantes MEMS ayant des broches de

type BGA, il n'est pas possible de les souder à l'aide d'un fer à souder conventionnel. La méthode suggérée pour l'assemblage manuel de puces BGA est par soudure à l'air chaud. Le procédé, illustré de la figure 4.2 à la figure 4.5, consiste à déposer de la pâte à souder sur les pads BGA, chauffer progressivement la composante et le substrat, déposer la composante le plus précisément possible à son emplacement et chauffer la composante jusqu'à ce qu'elle descende de quelques millièmes de pouces suite à la fusion de la soudure sous ses broches. Cependant, étant donné que les interrupteurs MEMS ne peuvent être chauffés à plus de 250°C (au risque d'être endommagés) et que le point de mouillage optimal de la pâte à souder se situe entre 210~225°C, un contrôle précis de la température du fer à air chaud est indispensable. La Figure 4.6 Circuit du syntoniseur assemblé.figure 4.6 illustre le syntoniseur prototype assemblé.

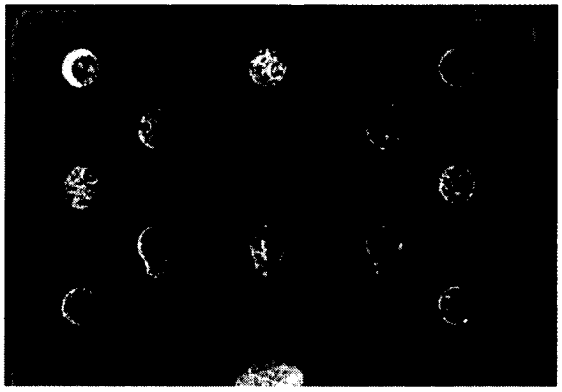


Figure 4.2 Pads d'un interrupteur MEMS.
Source : Teravicta, 2007.

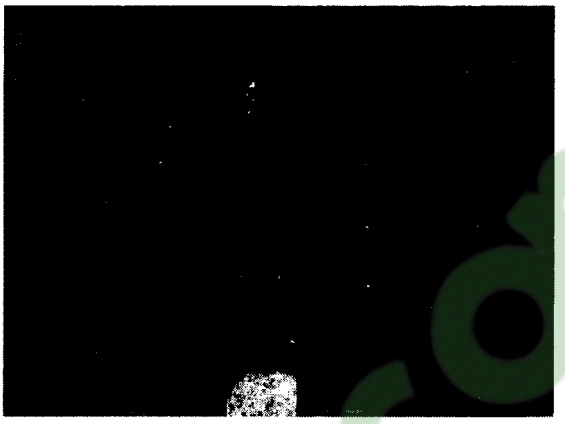


Figure 4.3 Application de pâte à souder sur les pads.
Source : Teravicta, 2007.

Figure 4.4 Mise en place du MEMS sur les pads recouverts de pâte à souder.

Source : Teravicta, 2007.

Figure 4.5 Soudage du MEMS par air chauffé à ~240°C.

Source : Teravicta, 2007.

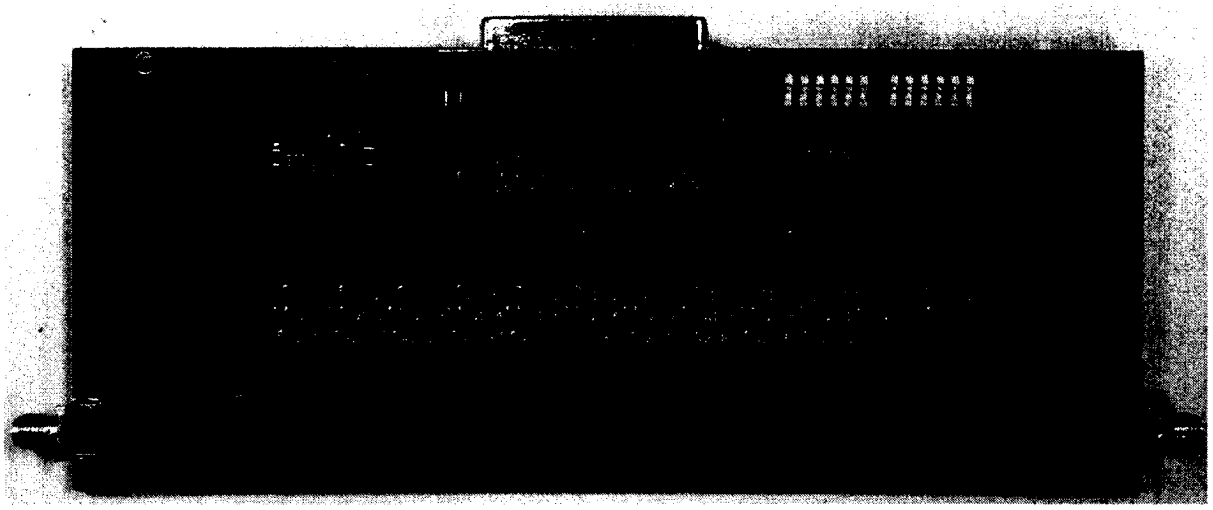


Figure 4.6 Circuit du syntoniseur assemblé.

CHAPITRE 5

RÉSULTATS

5.1 Mise en place du poste de mesures

Dans l'objectif d'évaluer les caractéristiques électriques et les performances du syntoniseur développé dans le cadre de cette recherche, ce dernier fut soumis à une caractérisation complète sur toute sa bande de fréquence et ce, pour chacun de ses états à l'aide d'un poste de mesures automatisé comprenant un ordinateur personnel, un analyseur de réseaux vectoriel (ARV), une source de tension et le circuit d'interface 8 bits – 12 bits présenté à la section 4.1.1.

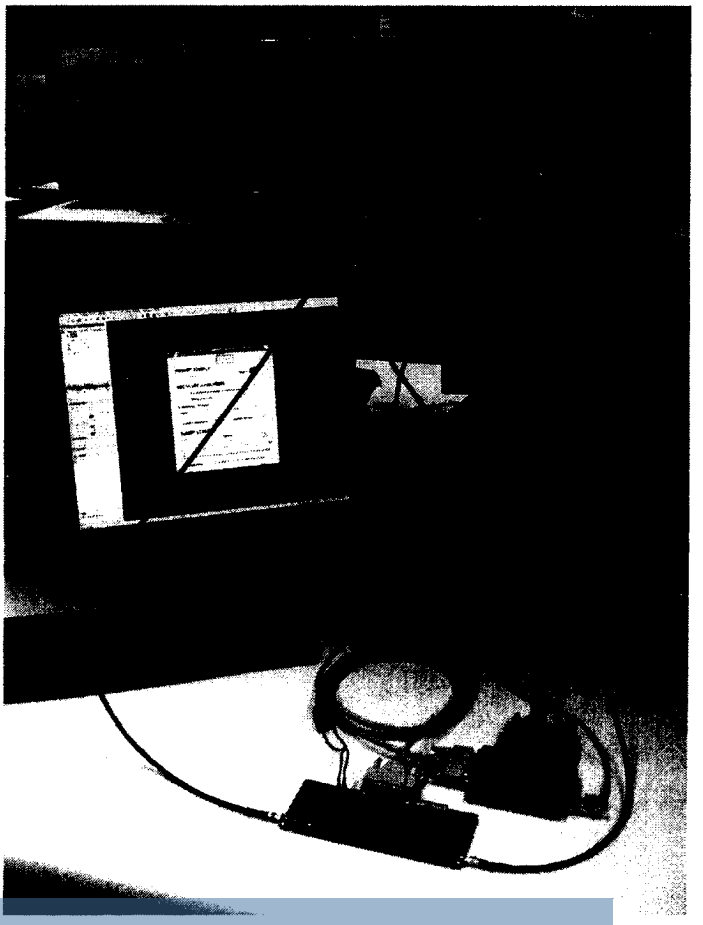


Figure 5.1 Poste de mesures.

Afin de pouvoir caractériser les 4096 états du syntoniseur sur la bande d'intérêt, il fut nécessaire d'automatiser la procédure de prise de mesures à l'aide de l'ordinateur personnel. En effet, l'ordinateur, étant équipé d'une carte de communication GP-IB permettant de contrôler les appareils de mesures et d'un port parallèle pouvant asservir le syntoniseur, peut servir de maître grâce au logiciel spécialisé Agilent VEE. Par le développement d'un programme de test (voir ANNEXE V) dans cet environnement dédié, il fut donc possible de recueillir toutes les données pertinentes sous formes de paramètres-S sauvegardées dans des fichiers textes. Une fois la prise de mesures complétée, ces fichiers peuvent être importés, analysés et interprétés sous Agilent ADS grâce à l'outil *Data Access Component* tel qu'illustré à la figure 5.2.

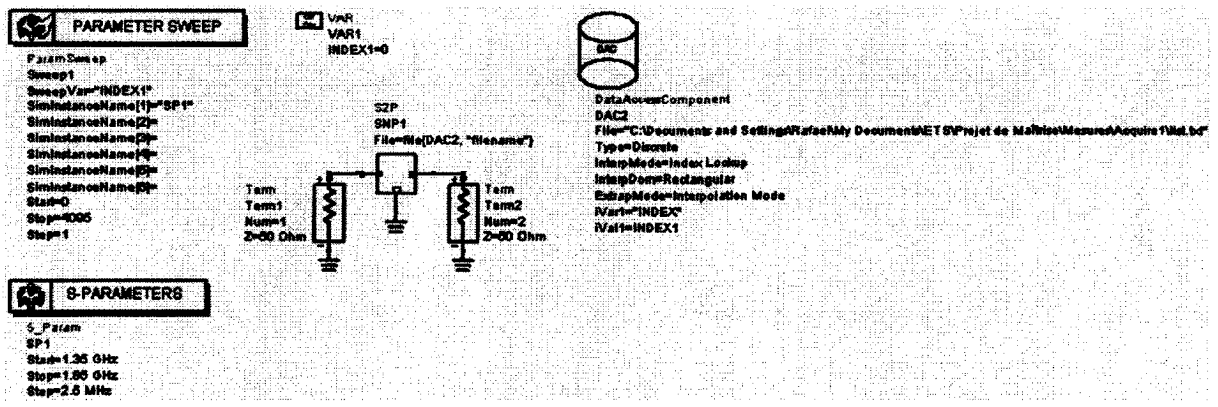


Figure 5.2 Importation des mesures dans Agilent ADS.

5.2 Mesures et interprétation

5.2.1 Impédances générées

La principale caractéristique d'intérêt d'un syntoniseur est sa capacité de générer une grande diversité d'impédances à l'intérieur d'une zone ciblée de l'abaque de Smith afin de maximiser le transfert d'énergie entre l'antenne et les circuits d'émission ou de réception. Cette mesure s'effectue en relevant les paramètres S11 du circuit à chacun des états du syntoniseur. Mis à part le nombre élevé de mesures à relever, facilité malgré tout par

l'automatisation de la tâche, ces mesures s'effectuent directement avec l'ARV et sont facilement présentables à l'intérieur d'un seul abaque.

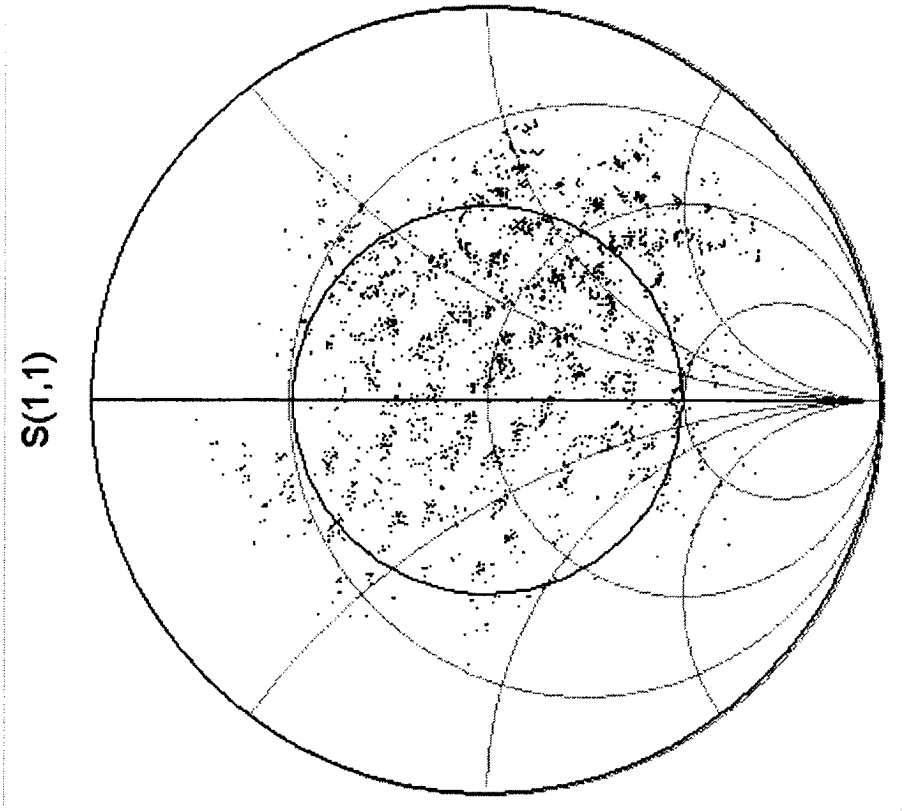


Figure 5.3 Répartition des impédances mesurée à 1,55 GHz.

Mesure de ciblage (moyenne) : 4,78
Mesure d'uniformité (écart-type) : 7,15

En comparaison avec les résultats de simulation à complexité élevée présentés à la figure 3.26 de la page 53, on constate rapidement une différence évidente dans l'uniformité de la distribution des points d'impédances avec une répartition en grappes de points. En effet, si le ciblage est presque aussi bon, tel qu'en fait foi la moyenne de 4,78 comparativement à 5,04 pour les simulations, l'uniformité de répartition est beaucoup moins bonne avec un écart-type de 7,15 comparativement à 2,76 pour la simulation.

Cette différence peut s'expliquer par différents facteurs. L'expérience acquise lors des dimensionnements des grandeurs physiques du circuit révèle que l'uniformité de la répartition des points se contrôle principalement par la variation de la distance entre les tronçons (*d*), alors que le ciblage de la zone est corrélé principalement par la longueur des tronçons (*l*). Si la répartition du nuage d'impédances n'est pas uniforme c'est donc que la longueur électrique du prototype n'est pas équivalente à celle obtenue par le modèle de simulation.

Une première hypothèse pourrait être que la constante diélectrique (ε_r) réelle du substrat utilisé ne correspond par à la valeur utilisée pour les simulations. Les spécifications du fabricant étant ε_r= 10,2 ± 0,3, des simulations complémentaires furent conduites sur le modèle le plus réaliste présenté à la section 3.4.2 afin d'évaluer l'effet de ces variations possibles sur la répartition du nuage de points.

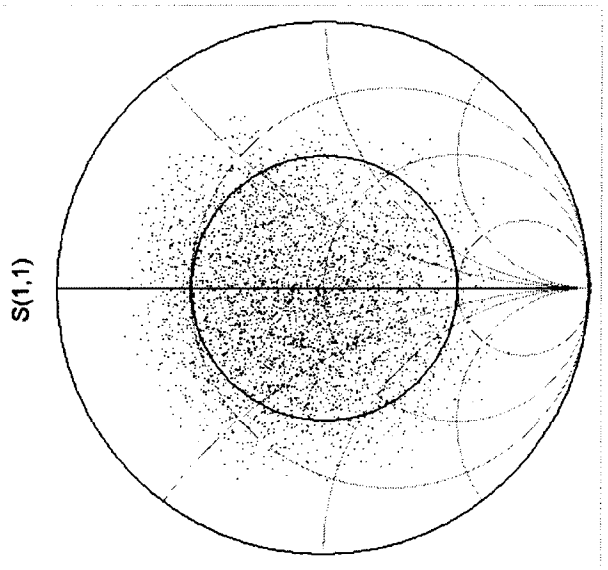


Figure 5.4 Répartition des impédances des simulations à complexité élevée avec ε_r=9,9.

Mesure de ciblage (moyenne) : 5,14

Mesure d'uniformité (écart-type) : 2,95

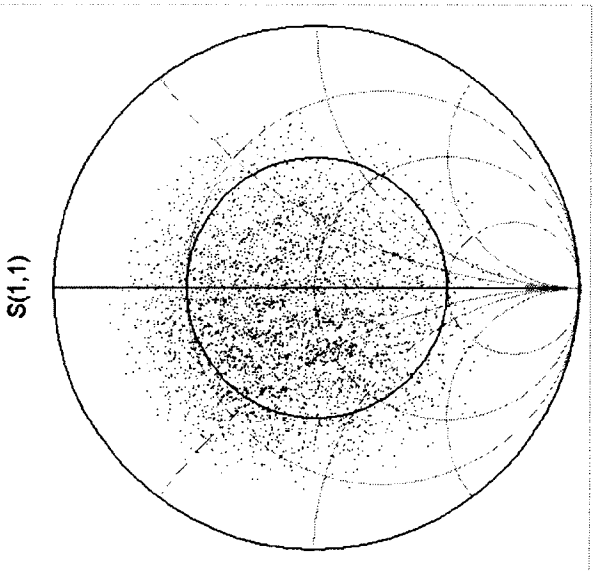


Figure 5.5 Répartition des impédances des simulations à complexité élevée avec ε_r=10,5.

Mesure de ciblage (moyenne) : 4,77

Mesure d'uniformité (écart-type) : 3,09

Comme le révèle la figure 5.4 et la figure 5.5, cette hypothèse ne semble pas corroborer de tels effets sur la distribution des impédances puisque les mesures d'uniformités varient tout au plus de 12 %, contrairement à la différence de 260 % constatée entre les résultats mesurés et simulés.

Une seconde hypothèse est que la longueur électrique de la branche principale soit différente de celle modélisée dans ADS par une densité de courant à l'intérieur de la ligne de transmission principale différente de la réalité. En effet, si les densités de courant à l'intérieur de la ligne de transmission micro-ruban sont différentes des simulations, ceci peut avoir comme effet d'allonger la distance parcourue par l'onde électromagnétique dans le cas d'un chemin en « S » ou en zigzag par exemple. Ces différences pourraient être particulièrement importantes à proximité des interrupteurs MEMS puisqu'ADS ne peut prendre compte des effets de cette structure à proximité d'une ligne de transmission puisque ce n'est par un simulateur électromagnétique à trois dimensions.

Finalement, une dernière hypothèse est que les paramètres-S modélisant l'interrupteur MEMS fourni par Teravicta comportent des erreurs de phase importantes qui se répercutent sur les simulations effectués avec ces fichiers. D'autant plus que ces erreurs s'accumulent pour chaque interrupteur présent sur la ligne de transmission.

5.2.2 Pertes d'insertion

La deuxième caractéristique d'intérêt est la perte d'insertion introduite par le syntoniseur puisqu'elle réduit la puissance utilisable du signal, déjà particulièrement faible à la réception. Cependant, l'ARV ne peut faire la distinction entre les pertes introduites par le dispositif testé et celles introduites par la désadaptation de l'impédance d'entrée. Donc, les pertes d'insertion ne peuvent être approximés par le paramètre S21 que lorsque la puissance réfléchie (S11) est relativement faible, ce qui indique que l'impédance du syntoniseur est suffisamment près de celle présentée par la source, soit 50 Ω. De plus, les mesures de puissance transmise (S21) et réfléchie (S11) s'interprètent habituellement sous forme de

courbe dans un diagramme de Bode qui s'étend sur la plage de fréquence d'intérêt. Contrairement aux mesures d'impédances, représenter les 4096 courbes sur un seul graphique s'avère difficile à interpréter, alors que les présenter toutes individuellement serait fastidieux et de faible intérêt. Les résultats présentés dans ce chapitre ont donc étés choisis judicieusement en fonction d'états représentatifs du syntoniseur.

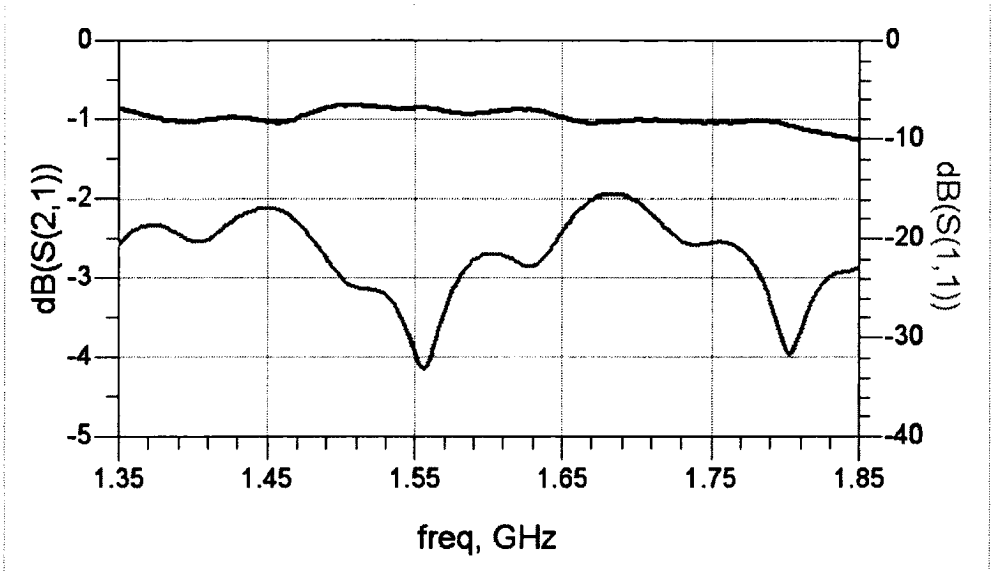


Figure 5.6 Mesures des paramètres S21 et S11 lorsque les 12 MEMS sont inactifs (état adapté à 50 Ω).

À l'état adapté à l'ARV (figure 5.6), soit avec tous les interrupteurs MEMS désactivés, les pertes d'insertion sont du même ordre de grandeur de celles obtenues lors des simulations présentées au chapitre 3.4.2. En effet, pour les fréquences les mieux adaptées, soit 1,555 GHz et 1,803 GHz, les pertes par insertion sont respectivement 0,85 dB et 1,07 dB, ce qui n'est que 0,59 dB ou 15 % de plus que celles relevés sur la figure 3.27 de la page 53.

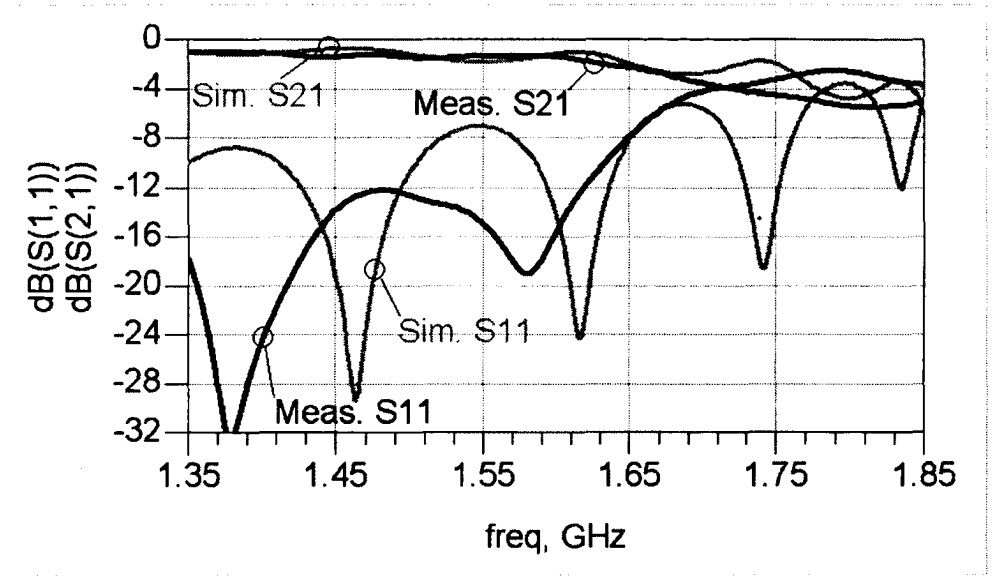


Figure 5.7 Mesures et simulations des paramètres S21 et S11 lorsque les 12 MEMS sont activés (état non adapté).

Afin d'évaluer les pertes introduites par les interrupteurs MEMS, l'état où ils sont tous activés est considéré comme étant un autre état judicieux à présenter et interpréter. Les mesures relevées à la figure 5.7 à cet état indiquent qu'aux fréquences les mieux adaptées, soit 1,380 GHz et 1,580 GHz, les pertes d'insertion sont respectivement de 1,06 dB et 1,36 dB contre 0,727 dB et 1,063 dB en simulation. Encore une fois, la différence entre les deux est minime et révèle que les modèles fournis par Teravicta, s'ils sont peut-être erronés sur la phase sont à tout le moins plus précis sur l'amplitude.

De plus, en comparaison avec les mesures prisent avec les interrupteurs inactifs, les pertes d'insertion attribuables aux MEMS sont de l'ordre de 0,504 dB, soit 12,4 % plus élevées que sans interrupteurs actifs. Par interpolation, chaque interrupteur MEMS TT2214 de Teravicta disposé selon cette configuration introduit donc approximativement 1% (0,05 dB) plus de pertes autour de 1.55 GHz que lorsqu'il est désactivé. Cette valeur correspond au 0,07 dB de pertes annoncés par le fabricant mentionné à la section 3.2.

CONCLUSION

La motivation initiale de ce travail était de concevoir un syntoniseur électronique reconfigurable à faibles pertes pour la bande III [1350 MHz ; 1850 MHz]. Cependant, les techniques de conception d'un syntoniseur électronique étant quasi-nullement documentée, il fut donc inévitable d'intégrer l'élaboration d'une méthodologie à cette recherche. Puisque ce type de circuit est difficilement synthétisable mathématiquement et que le cadre d'un projet de maîtrise n'offre typiquement pas assez de temps pour s'aventurer dans le développement d'une telle théorie, une approche plus pratique fut privilégiée.

Cette approche, en trois étapes, est basée sur une approche de force brute exploitant un environnement de CAO micro-ondes, tel Agilent ADS. La personnalisation d'ADS à l'aide de son langage de programmation (AEL) et le développement de métriques de performances, assiste le logiciel de simulation à trouver une configuration adéquate et en optimiser les paramètres physiques afin d'adapter une région spécifique de l'abaque de Smith.

Il est d'abord indispensable de connaître la région cible de l'abaque en mesurant le dispositif à adapter à tous ses états et fréquences d'intérêt. Dans notre cas, cette région représente les impédances présentés par une antenne en bande III et est délimitée par un cercle de rayon $|\Gamma| = 0{,}5$ centrée sur le point d'impédance caractéristique (z_0) de l'abaque.

La première étape, soit le choix de la configuration, consiste à ébaucher, avec des lignes de transmissions idéalisées, différentes configurations ayant le potentiel de générer une grande quantité d'impédances distinctes. La conception de ces configurations candidates est inspirée de la connaissance et de l'expérimentation du comportement de différentes structures micro-ondes comme, par exemple, le tronçon de ligne de transmission ou la cavité résonante. Pour évaluer la configuration offrant les meilleures perspectives d'adaptation de la région ciblée, une macro d'automatisation est développée afin de simuler tous les candidats pour différentes dimensions de structures. Grâce à l'observation des résultats et aux outils d'évaluation de performances développés, la configuration offrant les meilleures perspectives

d'adaptation de la région ciblée est sélectionnée en vue de l'étape suivante. Dans le cadre de cette recherche, une configuration basée sur une distribution uniforme le long d'une ligne de transmission de tronçons commutables d'égales dimensions présente les meilleures possibilités.

Une fois cette configuration choisie, la seconde étape consiste à établir la faisabilité d'un tel circuit en ébauchant un modèle de simulation plus réaliste en y intégrant les paramètres physiques de fabrication. Ces paramètres incluent les propriétés physiques et diélectriques du substrat sélectionné, le modèle réaliste de ligne de transmission et de la composante de commutation choisie, le cas échéant. Une fois ces paramètres intégrés à ADS, des lots de simulations automatisées, où peuvent varier plusieurs paramètres (3 maximum), sont exécutés. Ainsi, toutes les combinaisons des grandeurs physiques de la configuration peuvent être couvertes grossièrement afin de déterminer quelle combinaison présente les meilleures distributions d'impédances. Dans le cas du syntoniseur développé ici, avec un substrat à haute constante diélectrique et une composante de commutation MEMS la combinaison de dimensions la plus prometteuse est des tronçons d'une longueur de 117 mil séparés par 445 mil. En tenant compte de la dimension relative de ces longueurs par rapport aux composantes MEMS et à la ligne de transmission, la fabrication d'un tel circuit est réaliste.

Finalement, la dernière étape consiste à modéliser, de la manière la plus réaliste possible, le syntoniseur en y intégrant tous les paramètres physiques du circuit pour ensuite exécuter un dernier lot de simulations couvrant plus finement les combinaisons possibles de grandeurs physiques autour de la combinaison vers laquelle a convergé l'étape précédente. Ainsi, les paramètres vers lesquels converge le dernier lot de simulations serviront à la fabrication du prototype. Parmi les améliorations apportées au modèle du syntoniseur, les transitions de milieu de la ligne de transmission ainsi que son environnement diélectrique sous le MEMS ont été simulés et intégrés au modèle de l'interrupteur. Les grandeurs finales trouvées pour la fabrication sont donc de 440 mil entre chaque MEMS et 75 mil de longueur de tronçon.

La topologie du circuit exigeant des capacités de fabrication avancées de par les dimensions des conducteurs et vias sous l'interrupteur MEMS, la liste des fabricants de circuits imprimés ayant les capacités requises se limita qu'à quelques noms. La fabrication du prototype fut donc confiée à Metaplast Circuits Limited pour leur expérience dans la fabrication de circuits micro-ondes sur substrat plus exotique tel le RO3010 de type téflon-céramique utilisé pour ce circuit.

Les 4096 états du prototype furent mesurés à l'aide d'une station de tests automatisées et les résultats furent traités par le logiciel Agilent ADS pour comparaison avec les résultats de simulation et interprétation. Les outils de mesure révèlent que les résultats obtenus en ce qui à trait au premier critère de performance, la répartition des impédances générées, sont mitigés. Si l'étendue de la zone adaptable est comparable aux résultats de simulation, l'uniformité de la couverture est beaucoup moins bonne avec une mesure d'uniformité deux fois et demie moins bonne qu'en simulation traduisant ainsi ce qui est observé sur l'abaque, soit des impédances regroupées sous formes de grappes. Cependant, l'observation des courbes S21 et S11 corroborent les résultats de simulations et les spécifications du fabricant puisque les pertes d'insertion attribuables à chaque MEMS du syntoniseur sont de l'ordre de 0,05 dB.

ANNEXE I

MACRO ADS AEL D'AUTOMATISATION DES SIMULATIONS – CODE SOURCE

```
File: C:\Documents and Settings\Rafael\My Documents\ETS\Projet de Maîtrise\Docur
ments projet Duplexeur\ADS macros\automate_simul_export.ael  2008-01-21, 17:09:3
```

```
// Pour appeler, taper : load("C:\\Documents and Settings\\Rafael\\My
Documents\\ETS\\Projet de Maîtrise\\Documents projet Duplexeur\\ADS
macros\\automate_simul_export.ael");
// ou load("C:\\rlabedan\\ADS macros\\automate_simul_export.ael");

/*****************************************************************************

 Fichier:        automate_simul_export.ael

 Description:    Automatise des simulations multiples et exporte les
                 paramètres de réflexions S11 dans un fichier texte
                 pour traitement subséquent dans MATLAB. Variation possibles
                 sur 3 dimensions/variables, VAR1, VAR2 et VAR3.

 Auteur:         Rafael Labedan

 Date:           Avril  2007

 Language:       Application Extention Language (AEL) pour
                 Advanced Design System (ADS) (ver.: 2004~2006)

*****************************************************************************

 Notes:

  1-  !ATTENTION! Ce programe à été conçu pour les versions de ADS
      suivantes:
                  ADS2004A
                  ADS2006A

      La fonctionalité sur les versions autres n'a pas été testé et devrait
      donc l'être avant une utilisation sur celles-çi.

  2-  Certaines fonctions de ADS ne sont pas documentés dans la litérature
      officielle. La documentation non officielle se trouve sur le forum AEL
      de ADS sur le site de support "Agilent EESof Knowledge Center":
      https://edasupportweb.soco.agilent.com

  3-  VAR1 est la variable intérieure et VAR3 est la variable extérieure.
      C'est-à-dire que VAR1 prends toutes ses valeurs possibles, ensuite
      VAR2 passe à sa valeur suivante et ainsi de suite jusqu'à ce que
      toute la plage de valeurs de VAR2 ai été couverte. Ensuite, le processus
      se répète pour la valeur suivante de VAR3 jusqu'à ce que toute sa plage
      de
      valeurs ai aussi été couverte.

  4-  Sauvegarde les données (paramètres S) en "Datasets" ou fichiers textes
      sous les noms suivants:

        "Nom du circuit"__"Nom de VAR3"_"Valeur de VAR3"__"Nom de
        VAR2"_"Valeur de VAR2"__"Nom de VAR1"_"Valeur de VAR1".txt (ou .ds
        pour les datasets)

      et dans le répertoire suivant:
```

```
        $HOME/PROJECT/data/ (ex: C:\users\default\Diode_prj\data)

*****************************************************************************/

// *** Variables à initialiser en fonction du design et des paramètres de la
simulation ***
//_____________________________________________________________________________
_____________

decl btchsim_ProjectName = "aTunerMEMS_prj";
decl btchsim_DesignName = "Circuit_MEMSv4";

decl btchsim_NameVAR1 = "L1";      //Nom de la variable intérieure (VAR1).
decl btchsim_VAR1_Start = 50;    //    | Start, Step et Stop pour VAR1.
decl btchsim_VAR1_Step = 42;     // <=| Définit la plage de valeur à assigner à
VAR1.
decl btchsim_VAR1_Stop = 302;    //    | Ignoré si une liste de valeurs est
assigné à btchsim_VAR1_Values.
decl btchsim_VAR1_Values = list();  //Optionnel. Liste des valeurs à assigner.
Si vide utilise Start, Step et Stop.

decl btchsim_NameVAR2 = "L2";    //Nom de la variable extérieure (VAR2).
decl btchsim_VAR2_Start = 300;     //    | Start, Step et Stop pour VAR2.
decl btchsim_VAR2_Step = 71;     // <=| Définit la plage de valeur à assigner à
VAR2.
decl btchsim_VAR2_Stop = 797;    //    | Ignoré si une liste de valeurs est
assigné à btchsim_VAR2_Values.
decl btchsim_VAR2_Values = list();  //Optionnel. Liste des valeurs à assigner.
Si vide utilise Start, Step et Stop.

decl btchsim_NameVAR3 = "L3";    //Nom de la variable extérieure (VAR3).
decl btchsim_VAR3_Start = 1;     //    | Start, Step et Stop pour VAR3.
decl btchsim_VAR3_Step = 1;    // <=| Définit la plage de valeur à assigner à
VAR3.
decl btchsim_VAR3_Stop = 1;    //    | Ignoré si une liste de valeurs est
assigné à btchsim_VAR3_Values.
decl btchsim_VAR3_Values = list();  //Optionnel. Liste des valeurs à assigner.
Si vide utilise Start, Step et Stop.

// Optionnel. Liste des triplés de valeurs à tester.
// À préséance sur les 2 autres modes d'assignation de valeurs.
// Entrez sous la forme:  = list([Valeur#1_VAR3, Valeur#1_VAR2,
Valeur#1_VAR1], [Valeur#2_VAR3, Valeur#2_VAR2, Valeur#2_VAR1], etc...)
decl btchsim_TripletsOfValues = list(); //[8, 32, 24], [18, 19, 20], [26, 22,
8], [27, 9, 1], [23, 23, 23], [64, 14, 18], [67, 21, 54]);

decl btchsim_SaveResultsTo = "TXT_FILES";   // Type de fichiers de sauvegarde:
"DATASETS" ou "TXT_FILES" (fichiers textes).

// Répertoire d'exportation des fichiers textes. Ignoré pour les DATASETS.
decl btchsim_SaveDir = "C:\\DataOut\\";
//_____________________________________________________________________________
_____________

// *** INITIALISATION - Initialise les autres variables et peuple les
```

```
tableaux. ***

// Si des triplés de valeurs ne sont pas fournies dans la liste
btchsim_TripletsOfValues
// alors on utilise les valeurs fournies pour chaque variable pour générer des
triplés.
if (listlen(btchsim_TripletsOfValues) == 0)
{
  // Si des valeurs ne sont pas fournies dans les listes btchsim_VAR?_Values
  // alors on utilise les variables Start, Step et Stop pour les peupler.
  decl btchsim_ThisValue;

  if (listlen(btchsim_VAR1_Values) == 0)
  {
    btchsim_ThisValue = btchsim_VAR1_Start;
    while (btchsim_ThisValue <= btchsim_VAR1_Stop)
    {
      btchsim_VAR1_Values = append(btchsim_VAR1_Values,
      list(btchsim_ThisValue));
      btchsim_ThisValue = btchsim_ThisValue + btchsim_VAR1_Step;
    }
  }
  decl btchsim_NbElementsVAR1 = listlen(btchsim_VAR1_Values);

  if (listlen(btchsim_VAR2_Values) == 0)
  {
    btchsim_ThisValue = btchsim_VAR2_Start;
    while (btchsim_ThisValue <= btchsim_VAR2_Stop)
    {
      btchsim_VAR2_Values = append(btchsim_VAR2_Values,
      list(btchsim_ThisValue));
      btchsim_ThisValue = btchsim_ThisValue + btchsim_VAR2_Step;
    }
  }
  decl btchsim_NbElementsVAR2 = listlen(btchsim_VAR2_Values);

  if (listlen(btchsim_VAR3_Values) == 0)
  {
    btchsim_ThisValue = btchsim_VAR3_Start;
    while (btchsim_ThisValue <= btchsim_VAR3_Stop)
    {
      btchsim_VAR3_Values = append(btchsim_VAR3_Values,
      list(btchsim_ThisValue));
      btchsim_ThisValue = btchsim_ThisValue + btchsim_VAR3_Step;
    }
  }
  decl btchsim_NbElementsVAR3 = listlen(btchsim_VAR3_Values);

  // Remplir la liste de toutes combinaisons (triplés) de valeurs possibles
  pour
  // les 3 variables correspondant à toutes les simulations à effectuer.
  // Chaque itération de simulation correspond à élément dans la liste.
  // Chaque élément est un tableau de 3 valeurs. [Valeur de VAR3, Valeur de
  VAR2,Valeur de VAR1]
  decl btchsim_h;
```

```
  decl btchsim_i;
  decl btchsim_j;


  for (btchsim_h = 0; btchsim_h < btchsim_NbElementsVAR3; btchsim_h++)
  {
    for (btchsim_i = 0; btchsim_i < btchsim_NbElementsVAR2; btchsim_i++)
    {
      for(btchsim_j = 0; btchsim_j < btchsim_NbElementsVAR1; btchsim_j++)
      {
        btchsim_TripletsOfValues =
        append(btchsim_TripletsOfValues,list([nth(btchsim_h,
        btchsim_VAR3_Values), nth(btchsim_i, btchsim_VAR2_Values),
        nth(btchsim_j, btchsim_VAR1_Values)]));
      }
    }
  }
}
decl btchsim_NbLoopTotal = listlen(btchsim_TripletsOfValues);

// Message à l'utilisateur.
if (btchsim_NbLoopTotal < 2)
  de_info("Avertissement: Ce script n'est pas nécéssaire, moins de 2
  simulations à effectuer.");

// Variables de gestion de programme
decl btchsim_QueueState;
decl btchsim_Num = 0;

// Variable pour le message de bilan.
decl btchsim_MessageBilan = "";

// Variable pour les messages d'erreurs à ajouter au bilan.
decl btchsim_ErrorMessageBilan = "";

// Liste pour triplés de valeurs qui ne semblent pas avoir simulés
correctement. BUG ADS
decl btchsim_SuspectedTriplet = list();

// Triplé de valeurs en traitement. Pourrais être une variable locale à la
fonction
// btchsim_SimulateDesign si ce n'étais pas du BUG ADS
decl btchsim_CurrentTripletOfValues;

// Temps minimum en sec. qu'UNE simulation doit durer pour ne pas être
suspecte. BUG ADS
decl btchsim_MinOK_SimTime = 5;

// Nombre d'essais maximum pour reprendre une simulation qui ne s'est pas
exécuté. BUG ADS
decl btchsim_MaxRetries = 10;

// Variable de gestion de programme. Pour savoir combien de fois on a repris
une simulation. BUG ADS
decl btchsim_NumOfRedo = 0;
```

```
// Heure de départ des simulations.
decl btchsim_StartDateTime = date_time();

/*****************************************************************************
**
  Fonction principale. Appelée à la fin de ce fichier.
*****************************************************************************
*/
defun btchsim_Main()
{
  // On ferme tout ce qui pourrais être ouvert.
  de_close_all();
  de_close_project();

  // On ouvre le projet d'intérêt et le design à simuler.
  de_open_project(btchsim_ProjectName);
  decl MyWindowHandle = de_open_window(SCHEM_WIN, btchsim_DesignName);
  if (MyWindowHandle == NULL)
  {
    print_function_error("btchsim_Main()", strcat(btchsim_DesignName, " not a
    valid schematic design."));
    return;
  }

  // Les datasets sont toujours sauvés dans ce répertoire.
  if (btchsim_SaveResultsTo == "DATASETS")
  btchsim_SaveDir = strcat(getcwd(),"\\data\\");

  // On ne veut pas que la "data display window" s'ouvre après chaque
  simulations.
  de_set_simulation_datadisptoggle("OFF");

  // On s'assure que le simulateur est libre... sinon on entreprend rien.
  if(desim_foreground_simulation_running())
  {
     de_info("Une simulation est en cours.\nS.V.P. attendez qu'elle soit
     terminée.");
     return;
  }

  // Début du message de bilan.
  btchsim_MessageBilan = "Bilan de la simulation de groupe\n";
  btchsim_MessageBilan =
  strcat(btchsim_MessageBilan,"==========================================\
  n");
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, "Design: ",
  btchsim_ProjectName, "/", btchsim_DesignName,"\n\n");

  /*  Si on sauvegarde les données des différentes simulations dans des
  fichiers textes,
      alors on attribut au dataset son nom par défaut, soit le nom du design.
      Sinon c'est qu'on sauvegarde dans des datasets alors il faut assigner
      "None" à la
      variable OutputFileName du design afin d'éviter d'écrire les données
```

```
      dans un
      fichier texte pour rien. */
  if (btchsim_SaveResultsTo == "TXT_FILES")
    de_set_simulation_dataset(de_short_design_name(de_current_design_name()));
  else
    btchsim_SetVarVal("OutputFileName", "None");

  // desim_use_queued_simulators() existe seulement depuis ADS 2004A alors
  // si la fonction n'existe pas, une fonction ayant un comportement similaire
  est crée.
  if ( ! is_word( find_word( "desim_use_queued_simulators" ) ) )
       execute( "defun desim_use_queued_simulators( )  { return FALSE; }" );

  // Sauvegarde de l'état du simulateur
  btchsim_QueueState = desim_use_queued_simulators(FALSE);

  // Démarage de la première simulation
  btchsim_SimulateDesign();
}

/*****************************************************************************
**
  Fonction de lancement d'une nouvelle itération de simulation.
*****************************************************************************
*/
defun btchsim_SimulateDesign()
{
  // Assigne les prochaines valeurs à tester trouvés dans la liste aux valeurs
  courantes.
  btchsim_CurrentTripletOfValues = car(btchsim_TripletsOfValues);

  // Met à jour les variables du design avec ces valeurs.
  btchsim_SetVarVal(btchsim_NameVAR1, btchsim_CurrentTripletOfValues[2]);
  btchsim_SetVarVal(btchsim_NameVAR2, btchsim_CurrentTripletOfValues[1]);
  btchsim_SetVarVal(btchsim_NameVAR3, btchsim_CurrentTripletOfValues[0]);

  // Enlève ce triplé de la liste des valeurs à tester.
  btchsim_TripletsOfValues = cdr(btchsim_TripletsOfValues);

  // Converti les valeur en strings pour message utilisateur et nom de
  fichier.
  decl StrValVAR1 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[2]);
  decl StrValVAR2 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[1]);
  decl StrValVAR3 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[0]);

  //Mise-à-jour du message de bilan des simulations.
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, " ***Début de la
  simulation #", sprintf("%i",btchsim_Num+1), "***\n", date_time());
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, "Variable extérieure:
  ");
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, btchsim_NameVAR3, " = ",
  StrValVAR3, "\n");
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, "Variable centrale: ");
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, btchsim_NameVAR2, " = ",
  StrValVAR2, "\n");
```

```
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, "Variable intérieure:
  ");
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, btchsim_NameVAR1, " = ",
  StrValVAR1, "\n");

  // Crée un nom de fichier unique pour sauvegarder les données de simulation:
  StrValVAR1 = strcat(leftstr(StrValVAR1,
  index(StrValVAR1,".")),"_",rightstr(StrValVAR1,
  strlen(StrValVAR1)-index(StrValVAR1,".")-1));
  StrValVAR2 = strcat(leftstr(StrValVAR2,
  index(StrValVAR2,".")),"_",rightstr(StrValVAR2,
  strlen(StrValVAR2)-index(StrValVAR2,".")-1));
  StrValVAR3 = strcat(leftstr(StrValVAR3,
  index(StrValVAR3,".")),"_",rightstr(StrValVAR3,
  strlen(StrValVAR3)-index(StrValVAR3,".")-1));
  decl DataFileName = strcat(btchsim_DesignName, "__", btchsim_NameVAR3, "_",
  StrValVAR3, "__", btchsim_NameVAR2, "_", StrValVAR2, "__", btchsim_NameVAR1,
  "_", StrValVAR1);

  if (btchsim_SaveResultsTo == "TXT_FILES")
    {
      DataFileName = strcat("\"", btchsim_SaveDir, DataFileName, ".txt\""); //
      Dans une sting \" => "
      btchsim_SetVarVal("OutputFileName", DataFileName);
    }
  else
    de_set_simulation_dataset(DataFileName);

  //decl wInst=api_get_current_window();   // ??? Pas rap.

  // Sauvegarde l'horloge pour déterminer le temps de simulation plus tard.
  start_timer();

  // Démarre le simulateur.
  de_analyze();

  //sleep(1);

  // Attendre que la simulation en cours soit terminée à l'aide d'un timer qui
  // vérifie à toutes les 1 sec.
  // Function uniquement documentée sur le forum AEL d'ADS. Voir en-tête pour
  adresse.
  api_set_timer("btchsim_Timer",  // Nom du Timer
                1000,             // Temps enrte les appel de la fonction par
                le timer (en ms).
                1,              // Auto repeat.
                "btchsim_TimerCallbackFunc", // Nom de la fonction du timer
                (callback).
                NULL);                  // cbData of callback
}

/*****************************************************************************
**
  Fonction du timer - Vérifie si la simulation en cours est terminée.
*****************************************************************************
```

```
*/
defun btchsim_TimerCallbackFunc(cbData, callData)   // Arguments nécéssaires,
tel que montré sur Agilent EESof Knowledge Center.
{
  if (!desim_foreground_simulation_running())
  {
    // Désactiver le timer.
    api_remove_timer("btchsim_Timer");

    // Détermine le temps de la simulation.
    decl Time_Elapsed = total_elapsed_time();
    decl Seconds_Elapsed = nth(0, Time_Elapsed);
    // Mise-à-jour du message de bilan des simulations.
    btchsim_MessageBilan = sprintf("%s%s%i%s", btchsim_MessageBilan, "Durée :
    ", Seconds_Elapsed, " sec.\n");

    // Simulation suspectée de ne pas s'être executée. BUG ADS
    if (Seconds_Elapsed < btchsim_MinOK_SimTime)
    {
      // Après un certain nombre d'essais on abandonne et on averti
      l'utilisateur dans le message de bilan.
      if (btchsim_NumOfRedo < btchsim_MaxRetries)
      {
        // On incrémente le compteur de reprises.
        btchsim_NumOfRedo++;
        // Mise-à-jour du message de bilan des simulations.
        btchsim_MessageBilan = sprintf("%s%s%i%s%i%s", btchsim_MessageBilan,
        "Durée moins de ", btchsim_MinOK_SimTime, " sec.! Simulation
        possiblement incomplète.\nReprise #", btchsim_NumOfRedo,"... \n");
        // On remet le triplé de valeurs courante comme prochaines triplé à
        simuler.
        if (listlen(btchsim_TripletsOfValues) == 0)
          btchsim_TripletsOfValues = list(btchsim_CurrentTripletOfValues);
        else
          btchsim_TripletsOfValues = insert(btchsim_TripletsOfValues,
          btchsim_CurrentTripletOfValues);
      }
      else
      {
        // Incrémenter le compte des simulations effectuées.
        btchsim_Num++;
        // La prochaine simulation n'est pas une reprise.
        btchsim_NumOfRedo = 0;
        // À ce point on vérifie si un fichier correspondant à la simulation a
        été exporté
        if (btchsim_ConfirmCurrentFileExported())
        {
          // Mise-à-jour du message de bilan des simulations.
          btchsim_MessageBilan = sprintf("%s%s%i%s%i%s", btchsim_MessageBilan,
          "Durée moins de ", btchsim_MinOK_SimTime, " sec. mais fichier
          trouvé!\nVérifier si simulation complète.\nLimite de reprises
          atteinte (", btchsim_MaxRetries,") \n ***État de simulation
          incertain***\n\n");
          // Mise-à-jour du message de bilan des erreur avec le triplé de
          valeurs courante comme étant non-simulés.
```

```
      btchsim_ErrorMessageBilan =
      sprintf("%s%s%s%s%6.2f%s%s%s%6.2f%s%s%s%6.2f%s",
      btchsim_ErrorMessageBilan, "Simulation à vérifier: ",
      btchsim_NameVAR3, "=", btchsim_CurrentTripletOfValues[0], " ",
      btchsim_NameVAR2, "=", btchsim_CurrentTripletOfValues[1], " ",
      btchsim_NameVAR1, "=", btchsim_CurrentTripletOfValues[2], "\n");
    }
    else
    {
      // Mise-à-jour du message de bilan des simulations.
      btchsim_MessageBilan = sprintf("%s%s%i%s%i%s", btchsim_MessageBilan,
      "Durée moins de ", btchsim_MinOK_SimTime, " sec.! Simulation
      incomplète.\nLimite de reprises atteinte (", btchsim_MaxRetries,")
      \n ***Simulation non complétée***\n\n");
      // Mise-à-jour du message de bilan des erreur avec le triplé de
      valeurs courante comme étant non-simulés.
      btchsim_ErrorMessageBilan =
      sprintf("%s%s%s%s%6.2f%s%s%s%6.2f%s%s%s%6.2f%s",
      btchsim_ErrorMessageBilan, "Simulation non complétée: ",
      btchsim_NameVAR3, "=", btchsim_CurrentTripletOfValues[0], " ",
      btchsim_NameVAR2, "=", btchsim_CurrentTripletOfValues[1], " ",
      btchsim_NameVAR1, "=", btchsim_CurrentTripletOfValues[2], "\n");
    }
  }
}
else
{
  // Incrémenter le compte des simulations effectuées.
  btchsim_Num++;
  // La prochaine simulation n'est pas une reprise. BUG ADS
  btchsim_NumOfRedo = 0;
  // Mise-à-jour du message de bilan des simulations.
  btchsim_MessageBilan = sprintf("%s%s", btchsim_MessageBilan, " ***Fin de
  la simulation***\n\n");
}

// Est-ce qu'il reste des simulations à faire?
if (btchsim_Num < btchsim_NbLoopTotal)
  btchsim_SimulateDesign();           // Si oui, on en redémarre une
  autre.
else
{
  // Réinitialisation du nom de fichier d'exportation
  btchsim_SetVarVal("OutputFileName", strcat("\"", btchsim_DesignName,
  ".txt\"")); // Dans une sting \" => "
  // Réinitialisation de l'état de la file d'attente.
  desim_use_queued_simulators(btchsim_QueueState);
  // Mise-à-jour et affichage du message du bilan des simulations.
  btchsim_MessageBilan = strcat(btchsim_MessageBilan, "FIN DE LA
  SIMULATION DE GROUPE\nHeure de départ: ", btchsim_StartDateTime, "Heure
  de fin   : ", date_time());

  // Nom du fichier texte dans lequel le Bilan est sauvegardé.
  decl MyFileName = fix_path(strcat(getcwd(),"\data\btchsim.log"));
```

```
      // S'il y a des erreurs dans btchsim_ErrorMessageBilan alors on les mets
      à la fin du bilan. BUG ADS.
      if (strlen(btchsim_ErrorMessageBilan) != 0)
      {
        btchsim_MessageBilan = strcat(btchsim_MessageBilan, "\nERREURS DE
        SIMULATIONS :\n", btchsim_ErrorMessageBilan);
        // et on averti l'utilisateur.
        de_info(strcat("Erreurs possibles durant les simulations.\nVoir bilan
        des simulations pour les détails:\n", MyFileName));
      }

      // On ajoute le nom de fichier et l'emplacement du bilan à la fin du
      bilan même.
      btchsim_MessageBilan = strcat(btchsim_MessageBilan, "\nCe bilan est
      sauvegardé sous ", MyFileName);

      // Sauvegarde du bilan dans le fichier texte.
      decl MyFileID = fopen(MyFileName, "W");
      if (MyFileID == NULL)
        print_function_error("btchsim_TimerCallbackFunc()", strcat(MyFileName,
        " not a valid file."));
      fputs(MyFileID, btchsim_MessageBilan);
      fclose(MyFileID);

      // Affiche le bilan.
      de_data_dialog("Résultat de Simulation de Groupe", btchsim_MessageBilan,
      0, NULL);
    }
  }
}

/*****************************************************************************
**
  Fonction d'assignation d'une variable du design.
*****************************************************************************
*/
defun btchsim_SetVarVal(VarName, Value)
{
  // On commence avec un tableau blanc. Pas de sélection active.
  de_deselect_all();

  // Liste de toute les instances "VAR" du design. (Voir Table 14.1 du
  document "AEL" 2006)
  decl btchsim_lst_inst = de_get_design_instances(current_design_name(), NULL,
  ITEM_VARIABLE);

  // Selectionne une par une toutes les instances "VAR" de la liste.
  while (btchsim_lst_inst)
  {
    de_select_by_name(car(btchsim_lst_inst),1);  //2e argument : "1" => Par
    nom d'instance
    btchsim_lst_inst=cdr(btchsim_lst_inst);
  }

  // Mise-à-jour de la valeur. Toutes les instances "VAR" sont sélectionnées
```

```
  // mais seulement celle qui contient "VarName" sera modifiée.
  de_group_edit_parameter_value(VarName, Value);

  // On quitte les lieux propres.
  de_deselect_all();
}

/*****************************************************************************
**
  NOTE:***Fonction pour patcher un BUG d'ADS: À l'occasion certaines
  simulations
  commandés ne sont pas lancées et donc pas exportés.***
  Fonction vérifiant si la simulation a sauvegardé un fichier qui prouverait
  que
  la simulation à eu lieu. NON TESTÉ!!!
*****************************************************************************
*/
defun btchsim_ConfirmCurrentFileExported()
{
  // Converti les valeur en strings pour message utilisateur et nom de
  fichier.
  decl StrValVAR1 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[2]);
  decl StrValVAR2 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[1]);
  decl StrValVAR3 = sprintf("%08.2f", btchsim_CurrentTripletOfValues[0]);

  // Crée un nom de fichier unique pour sauvegarder les données de simulation:
  StrValVAR1 = strcat(leftstr(StrValVAR1,
  index(StrValVAR1,".")),"_",rightstr(StrValVAR1,
  strlen(StrValVAR1)-index(StrValVAR1,".")-1));
  StrValVAR2 = strcat(leftstr(StrValVAR2,
  index(StrValVAR2,".")),"_",rightstr(StrValVAR2,
  strlen(StrValVAR2)-index(StrValVAR2,".")-1));
  StrValVAR3 = strcat(leftstr(StrValVAR3,
  index(StrValVAR3,".")),"_",rightstr(StrValVAR3,
  strlen(StrValVAR3)-index(StrValVAR3,".")-1));
  decl DataFileName = strcat(btchsim_DesignName, "__", btchsim_NameVAR3, "_",
  StrValVAR3, "__", btchsim_NameVAR2, "_", StrValVAR2, "__", btchsim_NameVAR1,
  "_", StrValVAR1);

  if (btchsim_SaveResultsTo == "TXT_FILES")
    DataFileName = strcat(DataFileName, ".txt");
  else
    DataFileName = strcat(DataFileName, ".ds");

  decl MyFilesList = get_dir_files(btchsim_SaveDir);

  if (member(DataFileName, MyFilesList) == NULL)
    return FALSE;
  else
    return TRUE;
}

btchsim_Main();
```

ANNEXE II

PROGRAMME MATLAB D'IMPORTATION ET ÉVALUATION AUTOMATISÉES DES DONNÉES DE SIMULATIONS ADS – CODE SOURCE


```
function prog_run()
% prog_run

% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Février 2007 - ...

%Libère le "workspace"
clear();

%% Initialisation des variables

% OPTION DU MINIMA
% Si une valeur entière > 0 est spécifiée alors
% seuls les circuits ayant au minimum ce nombre
% de points par zone seront considérés comme
% candidats potentiels.
minima_requis = 1;

% Proportion de la population des nuages de
% de points ayant la meilleure moyenne à garder
% comme candidats pour évaluer le meilleur
% écart-type. Plus cette valeur est élevée plus
% l'accent est mis sur la précicion du nuage.
% À l'opposé plus cette valeur est faible plus
% l'accent est mis sur l'uniformité du nuage.
% MÉTHODE DE LA PROPORTION DE CANDIDATS
% Proportion des candidats à garder (Valeur entre 0.00 et 1.00)
prop_de_candidats = 0.0;  % Pour la trame de cercle un bon compromis est: 0.50;
% MÉTHODE DU NOMBRE DE CANDIDATS
% ou specifier le nombre des meilleurs candidats
% à garder avec la meilleure moyenne.
% (0 pour les autres méthodes)
x_ieme = 0;
% MÉTHODE DE LA PERFORMANCE RELATIVE
% ou spécifier l'intervalle de recherche
% jusqu'au candidat étant à moins de X% de
% la performance du meilleur candidat.
% (Valeur entre 0.00 et 1.00 - 0 pour les autres méthodes)
perf_relative = 0.30;

nb_pointes_grille_rad = 36;
working_directory = 'C:\DataOut\Test\';
type_fichier = 'CUSTOM';     % CUSTOM, S2PMDIF ou ADS_DATA_DISP_EXPORT
type_trame_d_analyse = 'TRAME_CERCLES';     %GRILLE, GRILLE_RADIALE ou TRAME_CERCLES
rayon_couv = 0.5;
rayon_zone = 0.02;

aire_zone = pi*rayon_zone^2;

fprintf(1, ['\n\nDébut de l''analyse des fichiers S11 générés avec ADS\n'   ...
        'Répertoire de travail : %s\n' ], working_directory);

start_clock_str = datestr(clock());
start_cputime = cputime();

%% Début du corps du programme
```

```
% Liste les fichiers dans le répertoire à traiter (répertoire d'exportation)
data_dir_content = dir(strcat(working_directory,  '*.txt'));
nb_de_fichiers = size(data_dir_content, 1);

footer_str = '';

% Boucle principale, importe, traite et calcule les statistiques de
% dispersion pour chaque fichier de données trouvé dans le répertoire.
for n_circuit=1 : nb_de_fichiers

    my_files(n_circuit, :) = strcat(working_directory, data_dir_content(n_circuit).name);

    fprintf(1, '\n(%s) Fichier %i/%i : %s' , datestr(clock()), n_circuit, nb_de_fichiers, ↙
data_dir_content(n_circuit).name);

%% Importation des données
    % Importer le nuage de points à partir du fichier spécifié et les retourne
    % dans une structure.
    fprintf(1, '\n(%s) Lecture du fichier et importation des paramètres S...' , datestr(clock()));
    [data, nb_freq] = importSparam(my_files(n_circuit, :), type_fichier);

    % Si un fichier n'a pas le même nombre de fréquences que les autres
    % alors soulever un avertissement puisque les résultats seront
    % biaisés.
    if exist('old_nb_freq','var') && exist('nb_freq','var') && (nb_freq ~= old_nb_freq)
        warning('prog_run:not_Equal_Nb_of_Freq' , ...
            ['\n\nATTENTION: Nombre différents de fréqunces entre les fichiers, résultats biaisés. ↙
\n' ...
             'Fichier précédent : %s (%i fréquences)\n'  ...
             'Fichier courant : %s (%i fréquences)\n\n' ], ...
            data_dir_content(n_circuit-1).name, old_nb_freq, data_dir_content(n_circuit).name, ↙
nb_freq)
    end;
    old_nb_freq = nb_freq;

%% Analyse de la dispersion
    % Extrait des statistiques du nuage de points (données) afin d'en mesurer
    % la dispersion et les retournes dans une structure.
    fprintf(1, '\n(%s) Analyse...' , datestr(clock()));

    switch upper(type_trame_d_analyse)
    case 'GRILLE'
      stats(n_circuit) = extraire_stats_nuage_grille(data, nb_freq, rayon_couv, aire_zone);
    case 'GRILLE_RADIALE'
      stats(n_circuit) = extraire_stats_nuage_grille_radiale(data, nb_freq, rayon_couv, aire_zone, ↙
nb_pointes_grille_rad);
    case 'TRAME_CERCLES'
      stats(n_circuit) = extraire_stats_nuage_trame_cercles(data, nb_freq, rayon_couv, aire_zone);
    otherwise
      warning('prog_run:Unknown_type_trame' , ...
          ['\n\nATTENTION: Le type d''analyse suivant n''est pas reconnu: %s\n'  ...
           'Utilisation de la trame par défaut: ''GRILLE_RADIALE''' ], ...
          type_trame_d_analyse)
      type_trame_d_analyse = 'GRILLE_RADIALE';
      stats(n_circuit) = extraire_stats_nuage_grille_radiale(data, nb_freq, rayon_couv, aire_zone, ↙
nb_pointes_grille_rad);
    end;
end;
```

```
%Sauvegarde des données statistiques extraites des nuages des différents
%circuits.
save(strcat(working_directory, 'stats_struct', datestr(clock(), 30), '.mat'), 'stats',↙
'nb_de_fichiers', 'nb_freq', 'my_files', 'type_trame_d_analyse' );

%Pour utiliser le Xieme candidat ou le pourcentage
%de performance relative comme limite de recherche.
if exist('x_ieme', 'var') && (x_ieme ~= 0);
    prop_de_candidats = (x_ieme - 0.1)/nb_de_fichiers;
end;

%Analyse les données statistiques afin de trouver le meilleur candidat.
save_str = analyse_donnees_stats(working_directory, prop_de_candidats, perf_relative, minima_requis, ↙
nb_de_fichiers, nb_freq, stats, type_trame_d_analyse, my_files);

%Fin de l'accumulation du temps processeur utilisé.
stop_clock_str = datestr(clock());
stop_cputime = cputime();
save_str = [save_str sprintf( 'Début : %s  Fin : %s  (Temps CPU : %i secondes) \n' , start_clock_str, ↙
stop_clock_str, stop_cputime - start_cputime)];

%% Sauvegarde et affichage des résultats
%Enregistre les résultats dans le fichier "Resultats de l'analyse.log"
footer_str = sprintf( 'Fichiers analysés:\n' );
for n_circuit=1 : nb_de_fichiers
    footer_str = [footer_str sprintf( '%s\n', my_files(n_circuit, :))];
end;
save_str = [save_str footer_str];

fid = fopen(sprintf( '%sResultats de l''analyse %s.log' , working_directory, datestr(clock(), 30)), ↙
'wt');
fprintf(fid, '%s', save_str);
fclose(fid);

end
```

```
function [param, nb_freq] = importSparam(path, type_fichier, nb_freq)
% Fonction qui importe et extrait les paramètres S du fichier ADS.
% IMPORTSPARAM(PATH, TYPE_FICHIER)
% Fonction:     Importe et extrait les paramètres S du fichier exporte par ADS.
%               Léger traitement sur les indices pour avoir des nombres entiers.
% Paramètres    1- Emplacement du fichier à importer sous forme de chaîne de
%               caratères
%               2- Type de fichier importé:
%               'CUSTOM', 'ADS_DATA_DISP_EXPORT' ou 'S2PMDIF'
% Retourne:     1- Structure contenant les données.
%               2- Nb. de fréquences differentes contenues dans le fichier.
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Février 2007

%% Initialisation des variables et ouverture de fichiers
% Création de la structure pour acceuillir les paramètres S extraits.
data = struct ('freq', {}, 'gamma', {}, 'theta', {}, 're', {}, 'im', {}, 'cplx', {});

% Ouverture du fichier ADS en lecture seule.
[fid, error_message] = fopen(path, 'r');
if fid == -1 errordlg(sprintf( '%s\n%s%s',path, '=> ', error_message), 'File Error', 'on'); end;

% Importe le contenu du fichier dans un tableau de "cell" (stings). Chaque
% string est délimitée par un espace, un TAB, un CR ou un LF.
contenu_fich = textscan(fid, '%s');

%% Noyeau du programme
% Remplissage du tableau de données en fonction du type de fichier importé.
if strcmp(type_fichier, 'CUSTOM')
    % Nb. de strings par paramètre S. Propre au fichier d'exportation des
    % paramètres S effectué avec le code suivant dans un objet MeasEqn d'ADS :
    % Meas3=write_var(sprintf("%s", OutputFileName), "W", % "Frequence S11(reel) S11(imag)", " ",
"s", 10, freq[0], S(1,1)[0])
    nb_strings_par_elements = 3;

    % Offset propre au header du fichier.
    offset = 3;

    % Nb de strings trouvés dans le fichier.
    nb_strings = size(contenu_fich{1,1}, 1);

    % Calcul du nombre de fréquences simulés.
    i_dern_occur = 1;
    nb_freq_simules = 1;
    contenu_fich_pts =[];
    for iter=2:nb_strings
        if  strcmp(contenu_fich{1,1}(iter), 'Frequence')
            nb_freq_simules = nb_freq_simules + 1;
            contenu_fich_pts = vertcat(contenu_fich_pts, contenu_fich{1,1}(i_dern_occur+offset:iter-
1));
            i_dern_occur = iter;
        end;
    end;
    contenu_fich_pts = vertcat(contenu_fich_pts, contenu_fich{1,1}(i_dern_occur+offset:iter));

    % Calcul du nombre de paramètres S (points) dans le fichier.
```

```
    nb_points = round((nb_strings - nb_freq_simules * offset) / nb_strings_par_elements);

    % Remplissage du tableau des coordonnées des paramètres S de réflections
    % (S11) en coordonnées cartésiennes et polaires.
    n = 0;
    while n < nb_points
        data(n+1).freq = str2num(char(contenu_fich_pts(n * nb_strings_par_elements + 1)));
        data(n+1).re = str2double(char(contenu_fich_pts(n * nb_strings_par_elements + 2)));
        data(n+1).im = str2double(char(contenu_fich_pts(n * nb_strings_par_elements + 3)));
        data(n+1).cplx = data(n+1).re + data(n+1).im*i;
        data(n+1).gamma = abs(data(n+1).cplx);
        data(n+1).theta = angle(data(n+1).cplx);
        n = n + 1;
    end;
elseif strcmp(type_fichier, 'S2PMDIF')
    % Nb. de strings par paramètre S. Propre au fichier d'exportation des
    % paramètres MDIF S2P
    nb_strings_par_elements = 41;

    % Offset propre au header du fichier.
    offset = 7;

    % Calcul du nombre de paramètres S (points) dans le fichier.
    nb_points = ceil((size(contenu_fich{1,1},1) - offset) / nb_strings_par_elements);

    % Remplissage du tableau des coordonnées des paramètres S de réflections
    % (S11) en coordonnées cartésiennes et polaires.
    nb_freq_simules = 0;
    n = 0;
    while n < nb_points
        data(n+1).freq = str2num(char(contenu_fich{1,1}(n * nb_strings_par_elements + 23 +  ↙
offset)));
        data(n+1).gamma = str2double(char(contenu_fich{1,1}(n * nb_strings_par_elements + 24 +  ↙
offset)));
        data(n+1).theta = str2double(char(contenu_fich{1,1}(n * nb_strings_par_elements + 25 +  ↙
offset)))/180 * pi;
        data(n+1).re = data(n+1).gamma * cos(data(n+1).theta);
        data(n+1).im = data(n+1).gamma * sin(data(n+1).theta);
        data(n+1).cplx = data(n+1).re + data(n+1).im*i;
        n = n + 1;
        if data(n+1).freq ~= freq_preced
            nb_freq_simules = nb_freq_simules + 1;
            freq_preced = data(n+1).freq;
        end;
    end;
 else
    % Nb. de strings par paramètre S. Propre au fichier d'exportation des
    % paramètres S d'ADS
    nb_strings_par_elements = 14;

    % Calcul du nombre de paramètres S (points) dans le fichier.
    nb_points = ceil(size(contenu_fich{1,1},1) / nb_strings_par_elements);

    % Remplissage du tableau des coordonnées des paramètres S de réflections
    % (S11) en coordonnées cartésiennes et polaires.
    nb_freq_simules = 0;
    n = 0;
    while n < nb_points
```

```
        data(n+1).freq = str2num(char(contenu_fich{1,1}(n * nb_strings_par_elements + 6)));
        data(n+1).gamma = str2double(char(contenu_fich{1,1}(n * nb_strings_par_elements + 7)));
        data(n+1).theta = str2double(char(contenu_fich{1,1}(n * nb_strings_par_elements + 9)))/180 *
pi;
        data(n+1).re = data(n+1).gamma * cos(data(n+1).theta);
        data(n+1).im = data(n+1).gamma * sin(data(n+1).theta);
        data(n+1).cplx = data(n+1).re + data(n+1).im*i;
        n = n + 1;
        if data(n+1).freq ~= freq_preced
            nb_freq_simules = nb_freq_simules + 1;
            freq_preced = data(n+1).freq;
        end;
    end;
end;

%% Fermeture de fichiers
fclose(fid);

%% Retour de fonction
param = data;
nb_freq = nb_freq_simules;

end
```

```
function stats = extraire_stats_nuage_grille(param_S, nb_freq, rayon_couv, aire_zone)
% Retourne les statistiques de dispersion du nuage de points basées sur une grille.
% EXTRAIRE_STATS_NUAGE_GRILLE(PARAM_S, NB_FREQ, RAYON_COUV, AIRE_ZONE)

% Fonction:     Extrait des données statistiques (moy, médianne,
%               écart-type) sur la dispersion du nuage de paramètres S pour
%               chaque fréquence.
%               Mesures basées sur une grille carré.
% Paramètres    1- Structure de données telles que générée par la fonction
%               importSparam.m
%               2- Nombre de fréquences differentes ou pareilles mais
%               non-contigues dans le fichier des parametres S tel que
%               trouves par la fonction importSparam.m
%               3- Rayon de la région à couvrir par le nuage.
%               4- Aire de la zone de couverture (carreau de la grille).
% Retourne:     1- Structure contenant les statistiques extraites.
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Mars 2007 - ...

t1 = cputime();      %Profiling
%% Initialisation des variables et ouverture de fichiers
% Création de la structure pour acceuillir les statistiques extraites.
stats_extraites = struct ( 'freq', {}, 'moyenne', {}, 'medianne', {}, 'ecart_type', {}, 'minima',↙
{});

%% Noyeau du programme
% Évalue la dispersion du nuage de points en extrayant des statistiques sur
% le nombre de points par carreau de la grille.

% Détermine le côté d'un carreau. Le multiple des coordonnées des axes
% formant la grille.
cote = sqrt(aire_zone);

% Calcul le nombre d'axes(abscisse ou ordonnée) faisant intersection avec
% le cercle de couverture dans chaque dimension  + 1 à gauche ou en
% bas dépendant de quel type d'axe il est question (raison: position de la
% coordonnée d'un carreau).
offset = ceil(rayon_couv / cote);
nb_axes = 2 * offset;

% Calcul la plage d'axes(abscisse ou ordonnée) en question.
n = 0;
plage_abs_ord = zeros(nb_axes,1);
for n = 1 : nb_axes
    plage_abs_ord(n) = (n-1-offset) * cote;
end;

% Création le la liste de carreaux à garder pour le mesures.
% Pour qu'un carreau de la grille soit considéré il faut qu'au moins ~50%
% de sa superficie se trouve dans le cercle de couverture. Une bonne
% approximation est de vérifier si le point central du carreau y est inclus.
% Les coordonnées d'un carreau sont données par son point inférieur gauche.
demi_cote = cote / 2;
coord_grille = [];
for m = 1 : nb_axes
    for n = 1  : nb_axes
```

```
        if sqrt((plage_abs_ord(m) + demi_cote)^2 + (plage_abs_ord(n) + demi_cote)^2) < rayon_couv
            coord_grille(size(coord_grille, 1)+1,:) = [plage_abs_ord(m), plage_abs_ord(n)];
        end;
    end;
end;

nb_pts_data = size(param_S, 2);
nb_carr_couv = size(coord_grille,1);
grille_occur = zeros(nb_freq, nb_carr_couv);

% Évalue le nombre de points dans chaques zones
t2 = cputime();      %Profiling
for n_car = 1 : nb_carr_couv
    n_freq = 0;
    freq_preced = 0;
    for n_d = 1 : nb_pts_data
        if param_S(n_d).freq ~= freq_preced
            n_freq = n_freq + 1;
            freq_preced = param_S(n_d).freq;
        end;
        if coord_grille(n_car, 1) < param_S(n_d).re && param_S(n_d).re < coord_grille(n_car, 1) +
cote && ...
        coord_grille(n_car, 2) < param_S(n_d).im && param_S(n_d).im < coord_grille(n_car, 2) + cote
            grille_occur(n_freq, n_car) = grille_occur(n_freq, n_car) + 1;
        end;
    end;
end;
t_main_loop = (cputime() - t2)/60;      %Profiling

% Entrer les fréquences trouvés dans le tableau de retour
n_freq = 0;
freq_preced = 0;
for n_d = 1 : nb_pts_data
    if param_S(n_d).freq ~= freq_preced
        n_freq = n_freq + 1;
        stats_extraites(1).freq(n_freq) = param_S(n_d).freq;
        freq_preced = param_S(n_d).freq;
    end;
end;

% Extraire les statistiques par fréquence
for n_freq = 1 : nb_freq
    stats_extraites(1).moyenne(n_freq) = mean(grille_occur(n_freq, :));
    stats_extraites(1).medianne(n_freq) = median(grille_occur(n_freq, :));
    stats_extraites(1).ecart_type(n_freq) = std(grille_occur(n_freq, :));
    stats_extraites(1).minima(n_freq) = min(grille_occur(n_freq, :));
end;

%% Retour de fonction
stats = stats_extraites;

t_overhead = (cputime() - t1)/60 - t_main_loop;      %Profiling
end
```

```
function stats = extraire_stats_nuage_grille_radiale(param_S, nb_freq, rayon_couv, aire_zone,
nb_pointes)
% Retourne les statistiques de dispersion du nuage de points basées sur une grille radiale.
% EXTRAIRE_STATS_NUAGE_GRILLE_RADIALE(PARAM_S, NB_FREQ, RAYON_COUV, AIRE_ZONE, NB_POINTES)

% Fonction:     Extrait des données statistiques (moy, médianne,
%               écart-type) sur la dispersion du nuage de paramètres S pour
%               chaque fréquence.
%               Mesures basées sur une grille radiale.
% Paramètres    1- Structure de données telles que générée par la fonction
%               importSparam.m
%               2- Nombre de fréquences differentes ou pareilles mais
%               non-contigues dans le fichier des parametres S tel que
%               trouvés par la fonction importSparam.m
%               3- Rayon de la région à couvrir par le nuage.
%               4- Aire de la zone de couverture (carreau de la grille).
%               5- Nombre de pointes désirées pour établir la grille
%               radiale. Minimum : 1 pointe = tout le cercle.
% Retourne:     1- Structure contenant les statistiques extraites.
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Mars 2007 - ...

t1 = cputime();      %Profiling
%% Initialisation des variables et ouverture de fichiers
% Création de la structure pour acceuillir les statistiques extraites.
stats_extraites = struct ( 'freq', {}, 'moyenne', {}, 'medianne', {}, 'ecart_type', {}, 'minima',
{});

%% Noyeau du programme
% Évalue la dispersion du nuage de points en extrayant des statistiques sur
% le nombre de points par carreau de la grille radiale.

%Calcul des pentes des différentes arêtes constituant les pointes.
nb_pointes = round(nb_pointes);
if ~nb_pointes
    nb_pointes = 1;
end;

angle_base = 2*pi / nb_pointes;

for n = 1 : nb_pointes + 1
    angle_seg(n) = -pi + (n-1) * angle_base;
end;

% Détermine les rayons des cercles concentriques d'aires multiples de
% l'aire de base, aire_zone.
rayon(1) = 0;
n = 1;
while rayon(n) < rayon_couv
    n = n + 1;
    rayon(n) = sqrt( (nb_pointes * aire_zone * (n-1)) / pi );
end;

% Dans la liste des rayons trouvés, garder seulement le dernier rayon
% s'il est plus près de la région à couvrir que le précédent.
if abs(rayon(n) - rayon_couv) > abs(rayon_couv - rayon(n-1))
```

```
    rayon(n) = [];
end;

nb_cercles = size(rayon, 2) - 1;

nb_pts_data = size(param_S, 2);

grille_rad_occur = zeros(nb_freq, nb_cercles, nb_pointes);

% Évalue le nombre de points dans chaques zones
t2 = cputime();      %Profiling
for n_c = 1 : nb_cercles
    for n_p = 1 : nb_pointes
        n_freq = 0;
        freq_preced = 0;
        for n_d = 1 : nb_pts_data
            if param_S(n_d).freq ~= freq_preced
                n_freq = n_freq + 1;
                freq_preced = param_S(n_d).freq;
            end;
            if angle_seg(n_p) < param_S(n_d).theta && param_S(n_d).theta < angle_seg(n_p + 1) &&   ...
            rayon(n_c) < param_S(n_d).gamma && param_S(n_d).gamma < rayon(n_c + 1)
                grille_rad_occur(n_freq, n_c, n_p) = grille_rad_occur(n_freq, n_c, n_p) + 1;
            end;
        end;
    end;
end;
t_main_loop = (cputime() - t2)/60;      %Profiling

% Entrer les fréquences trouvés dans le tableau de retour
n_freq = 0;
freq_preced = 0;
for n_d = 1 : nb_pts_data
    if param_S(n_d).freq ~= freq_preced
        n_freq = n_freq + 1;
        stats_extraites(1).freq(n_freq) = param_S(n_d).freq;
        freq_preced = param_S(n_d).freq;
    end;
end;

% Extraire les statistiques par fréquence
for n_freq = 1 : nb_freq
    stats_extraites(1).moyenne(n_freq) = mean(grille_rad_occur(n_freq, :));
    stats_extraites(1).medianne(n_freq) = median(grille_rad_occur(n_freq, :));
    stats_extraites(1).ecart_type(n_freq) = std(grille_rad_occur(n_freq, :));
    stats_extraites(1).minima(n_freq) = min(grille_rad_occur(n_freq, :));
end;

%% Retour de fonction
stats = stats_extraites;

t_overhead = (cputime() - t1)/60 - t_main_loop;      %Profiling
end
```

```
function stats = extraire_stats_nuage_trame_cercles(param_S, nb_freq, rayon_couv, aire_zone)
% Retourne les statistiques de dispersion du nuage de points basées sur une trame de cercles.
% EXTRAIRE_STATS_NUAGE_TRAME_CERCLES(PARAM_S, NB_FREQ, RAYON_COUV, AIRE_ZONE)

% Fonction:     Extrait des données statistiques (moy, médianne,
%               écart-type) sur la dispersion du nuage de paramètres S pour
%               chaque fréquence.
%               Mesures basées sur une trame de cercles.
% Paramètres    1- Structure de données telles que générée par la fonction
%               importSparam.m
%               2- Nombre de fréquences differentes ou pareilles mais
%               non-contigues dans le fichier des parametres S tel que
%               trouvés par la fonction importSparam.m
%               3- Rayon de la région à couvrir par le nuage.
%               4- Aire des zones de couverture.
% Retourne:     1- Structure des statistiques extraites.
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Mars 2007 - ...

t1 = cputime();     %Profiling
%% Initialisation des variables et ouverture de fichiers
% Création de la structure pour acceuillir les statistiques extraites.
stats_extraites = struct ( 'freq', {}, 'moyenne', {}, 'medianne', {}, 'ecart_type', {}, 'minima',↙
{});

%% Noyeau du programme
% Évalue la dispersion du nuage de points en extrayant des statistiques sur
% le nombre de points par cercle de la trame de cercles.

% Détermine le rayon de la zone en fonction de l'aire de la zone.
rayon_zone = sqrt(aire_zone / pi());

% Génère une trame de cercles adjacents de dimension donnée par dim_zone
% afin de couvrir la région du nuage de points donné par rayon_couv.
% Compte le nombre de points par zones.
trame = gen_trame(rayon_couv, rayon_zone);

trame_occur = zeros(nb_freq, size(trame,2));

nb_pts_data = size(param_S,2);
nb_pts_trame = size(trame,2);

% Évalue le nombre de points dans chaques zones
t2 = cputime();     %Profiling
for n_t = 1 : nb_pts_trame
    n_freq = 0;
    freq_preced = 0;
    for n_d = 1 : nb_pts_data
        if param_S(n_d).freq ~= freq_preced
            n_freq = n_freq + 1;
            freq_preced = param_S(n_d).freq;
        end;
        if abs(param_S(n_d).cplx - trame(n_t)) < rayon_zone
            trame_occur(n_freq, n_t) = trame_occur(n_freq, n_t) + 1;
        end;
    end;
```

```
end;
t_main_loop = (cputime() - t2)/60;      %Profiling

% Entrer les fréquences trouvés dans le tableau de retour
n_freq = 0;
freq_preced = 0;
for n_d = 1 : nb_pts_data
    if param_S(n_d).freq ~= freq_preced
        n_freq = n_freq + 1;
        stats_extraites(1).freq(n_freq) = param_S(n_d).freq;
        freq_preced = param_S(n_d).freq;
    end;
end;

% Extraire les statistiques par fréquence
for n_freq = 1 : nb_freq
    stats_extraites(1).moyenne(n_freq) = mean(trame_occur(n_freq, :));
    stats_extraites(1).medianne(n_freq) = median(trame_occur(n_freq, :));
    stats_extraites(1).ecart_type(n_freq) = std(trame_occur(n_freq, :));
    stats_extraites(1).minima(n_freq) = min(trame_occur(n_freq, :));
end;

%% Retour de fonction
stats = stats_extraites;

t_overhead = (cputime() - t1)/60 - t_main_loop;      %Profiling
end
```

```
function points = gen_trame(rayon_couv, rayon_zone)
% Génère les coordonnees complexes de cercles couvrant une région (circulaire) désirée.
% GEN_TRAME(RAYON_COUV, RAYON_ZONE)
% Fonction:     Génère les coordonnées complexes des centres d'un nuage de cercles
%               couvrant une région complexe de géométrie circulaire.
% Paramètres    1- Rayon de la région circulaire à couvrir.
%               2- Rayon de la zone (cercle) voulue pour couvrir la
%               région circulaire.
% Retourne:     1- Tableau des coordonnées (centre) des zones (cercles).
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Février 2007

%% Initialisation des variables

% Zone au centre (toujours présente)
trame(1) = 0 + 0*i;

% Calcul de la distance entre deux zones circulaires contigues en fct. du rayon.
dist_inter_zone = rayon_zone * sqrt(2);

%% Noyeau du programme
% Génère les zones dans les quadrants de la région à couvrir
x = 0;
y = 0;
fin = false;
fin_ligne = false;
while ~fin
    y = y + dist_inter_zone;
    x = y + dist_inter_zone;
    if sqrt(x^2 + y^2) > rayon_couv
        fin = true;
    else
        fin_ligne = false;
    end;
    while ~fin_ligne
        if sqrt(x^2 + y^2) > rayon_couv
            fin_ligne = true;
        else
            trame(size(trame,2)+1) = x + y*i;
            trame(size(trame,2)+1) = y + x*i;
            trame(size(trame,2)+1) = x - y*i;
            trame(size(trame,2)+1) = y - x*i;
            trame(size(trame,2)+1) = -x + y*i;
            trame(size(trame,2)+1) = -y + x*i;
            trame(size(trame,2)+1) = -x - y*i;
            trame(size(trame,2)+1) = -y - x*i;
            x = x + dist_inter_zone;
        end;
    end;
end;

% Genere les zones sur les diagonales de la région à couvrir
x = dist_inter_zone;
while sqrt(2)*x <= rayon_couv
    trame(size(trame,2)+1) = x + x*i;
    trame(size(trame,2)+1) = x - x*i;
```

```
    trame(size(trame,2)+1) = -x + x*i;
    trame(size(trame,2)+1) = -x - x*i;
    x = x + dist_inter_zone;
end;

% Génère les zones sur les axes de la région à couvrir
x = dist_inter_zone;
while x <= rayon_couv
    trame(size(trame,2)+1) = x;
    trame(size(trame,2)+1) = -x;
    trame(size(trame,2)+1) = x*i;
    trame(size(trame,2)+1) = -x*i;
    x = x + dist_inter_zone;
end;

%% Retour de fonction
points = trame;
end
```

```
function save_str = analyse_donnees_stats(working_directory, prop_de_candidats, perf_relative,
minima_requis, nb_de_fichiers, nb_freq, stats, type_trame_d_analyse, my_files)
% Sélectionne les meilleurs candidats en fct des critères.
% Retourne l'information sur les dans une string.
% ANALYSE_DONNEES_STATS(WORKING_DIRECTORY, PROP_DE_CANDIDATS,
% PERF_RELATIVE, MINIMA_REQUIS, NB_DE_FICHIERS, ND_FREQ, STATS, TYPE_TRAME_D_ANALYSE, MY_FILES)
% Fonction:     Analyse les données statistiques et sélectionne les meilleurs
%               candidats en fct des critères. Retourne l'information sur les
%               meilleurs candidats dans une string affichable et/ou enregistrable
%               dans un fichier texte.
% Paramétres    1- Répertoire de travail
%               2- Critère de rétention basé sur la
%               proportion de candidats à évaluer.
%               3- Critère de rétention basé sur la
%               performance relative des candidats à évaluer.
%               4- Critère de rétention supplémentaire basé sur
%               un minimum de points dans chacune des sous-régions.
%               5- Nb de fichiers comparées.
%               6- Nb de fréquences comparées par fichier.
%               7- Statistiques extraites préalablememt.
%               8- Type de trame d'analyse préalablement utilisée
%               pour extraire les statistiques.
%               9- Liste des fichiers préalablement analysés.
% Retourne:     1- String contenant l'information sur les candidats gagnants.
% Auteur:       Rafael Labedan
% Orgaisation:  École de technologie supérieure
%               Chaire de recherche Ultra-Electronics
% Date: Avril 2007 - ...

header_str = '';
minima_requis = round(minima_requis);
if minima_requis > 0
    temp_stats = struct ( 'freq', {}, 'moyenne', {}, 'medianne', {}, 'ecart_type', {}, 'minima', {});
end;
i_originel = zeros(nb_freq, nb_de_fichiers);
candidats_ecart_type = infinities2d(nb_freq, nb_de_fichiers);
minima_atteint = ones(1, nb_freq);

%% Noyeau du programme
%Recherche du meilleur candidat à partir des données statistiques.
fprintf(1, '\n(%s) Recherche des meilleurs candidats pour chaque fréquence...' , datestr(clock()));

if  perf_relative <= 0 || 1 < perf_relative
    nb_candidats = ceil(prop_de_candidats * nb_de_fichiers);
end;

for n_freq = 1:nb_freq
    if minima_requis > 0
        minimas_f(n_freq,:) = [arrayfun(@(s) s.minima(n_freq), stats)];
        [dummy, i_candidats_moy(n_freq, :)] = sort(minimas_f(n_freq, :),   'descend');
        iter = 1;
        while stats(i_candidats_moy(n_freq, iter)).minima(n_freq) >= minima_requis && iter <=
nb_de_fichiers
            i_originel(n_freq, iter) = i_candidats_moy(n_freq, iter);
            temp_stats(iter).freq(n_freq) = stats(i_candidats_moy(n_freq, iter)).freq(n_freq);
            temp_stats(iter).moyenne(n_freq) = stats(i_candidats_moy(n_freq, iter)).moyenne(n_freq);
            temp_stats(iter).medianne(n_freq) = stats(i_candidats_moy(n_freq, iter)).medianne
(n_freq);
```

```
            temp_stats(iter).ecart_type(n_freq) = stats(i_candidats_moy(n_freq, iter)).ecart_type
(n_freq);
            temp_stats(iter).minima(n_freq) = stats(i_candidats_moy(n_freq, iter)).minima(n_freq);
            iter = iter + 1;
        end;
    else
        temp_stats = stats;
        i_originel(n_freq, :) = 1:nb_de_fichiers;
    end;

    if not(minima_requis) || minimas_f(n_freq, i_candidats_moy(n_freq, 1)) ~= 0
        moyennes_f = [arrayfun(@(s) s.moyenne(n_freq), temp_stats)];
        [dummy, i_candidats_ecart_type(n_freq, :)] = sort(moyennes_f(n_freq, :),   'descend');
        iter1 = 1;
    %% if temp_stats(i_candidats_ecart_type(n_freq, 1)).moyenne(n_freq) ~= 0
        if exist('nb_candidats', 'var')
            while i_originel(n_freq, i_candidats_ecart_type(n_freq, iter1)) ~= 0 && iter1 <=
nb_candidats
                candidats_ecart_type(n_freq, iter1) = temp_stats(i_candidats_ecart_type(n_freq,
iter1)).ecart_type(n_freq);
                iter1 = iter1 + 1;
            end;
        else
            meilleure_moy = temp_stats(i_candidats_ecart_type(n_freq, 1)).moyenne(n_freq);
            moyenne_limite = meilleure_moy * (1 - perf_relative);
            while i_originel(n_freq, i_candidats_ecart_type(n_freq, iter1)) ~= 0 &&   ...
                  temp_stats(i_candidats_ecart_type(n_freq, iter1)).moyenne(n_freq) >=
moyenne_limite && iter1 <= nb_de_fichiers
                candidats_ecart_type(n_freq, iter1) = temp_stats(i_candidats_ecart_type(n_freq,
iter1)).ecart_type(n_freq);
                iter1 = iter1 + 1;
            end;
            nb_candidats = iter1 - 1;
        end;
        [dummy, i_meilleurs_candidats(n_freq,:)] = sort(candidats_ecart_type(n_freq,:),   'ascend');
    else
        minima_atteint(n_freq) = 0;
        nb_candidats = 0;
    end;
end;

%Affiche et enregistre les résultats dans la variable save_str
fprintf(1, '\n(%s) Fin de l''analyse.\n\n' , datestr(clock()));

save_str = sprintf([ 'RÉSULTATS\n' ...
    '---------\n' ...
    'Meilleur candidat pour chaque fréquence:\n' ]);
for n_freq=1 : nb_freq
    if minima_atteint(n_freq) || minima_requis <= 0
        disp_str = sprintf([ 'Fréquence = %-15.2f \n'  ...
        'Moyenne = %-15.2f\n'  ...
        'Écart-type = %-15.2f\n'  ...
        'Minima = %-15.0f\n'  ...
        'Fichier : %s\n\n' ], ...
        temp_stats(i_candidats_ecart_type(n_freq, i_meilleurs_candidats(n_freq, 1))).freq(n_freq),
...
        temp_stats(i_candidats_ecart_type(n_freq, i_meilleurs_candidats(n_freq, 1))).moyenne
(n_freq), ...
```

```
        temp_stats(i_candidats_ecart_type(n_freq, i_meilleurs_candidats(n_freq, 1))).ecart_type
(n_freq), ...
        temp_stats(i_candidats_ecart_type(n_freq, i_meilleurs_candidats(n_freq, 1))).minima(n_freq),
...
        my_files(i_originel(n_freq, i_candidats_ecart_type(n_freq, i_meilleurs_candidats(n_freq,
1))),:));
        fprintf(1, '%s', disp_str);
    else
        disp_str = sprintf([ 'Fréquence = %-15.2f \n' ...
        'Aucun candidat trouvé (Minima non rencontré).\n\n' ], ...
        stats(1).freq(n_freq));
    end;
    save_str = [save_str disp_str];  % Concatenation de string
end;

if exist('moyenne_limite', 'var')
    methode_selection = 'Performance relative';
    valeur_methode_selection = perf_relative * 100;
else
    methode_selection = 'Proportion de candidats';
    valeur_methode_selection = prop_de_candidats * 100;
end;

header_str = sprintf([ 'Résultats de l''analyse des %i fichiers du répertoire suivant:\n'  ...
    '%s\n' ...
    'Voir la liste complète au bas de ce fichier.\n\n'  ...
    'PARAMÈTRES DE L''ANALYSE:\n'  ...
    'Type de grille utilisée pour l''analyse du nuage d''impédances: %s\n\n'  ...
    'Minima par zone (par carrau de grille): %-12.0d\n\n'  ...
    'Méthode de sélection des meilleurs candidats:\n'  ...
    '%s = %-3.2f%% (%i candidats)\n\n' ], ...
    nb_de_fichiers, working_directory, type_trame_d_analyse, minima_requis, methode_selection,
valeur_methode_selection, nb_candidats);

%% Retour de fonction
save_str = [header_str save_str];

end
```

ANNEXE III

CIRCUIT D'INTERFACE 8 BITS – 12 BITS

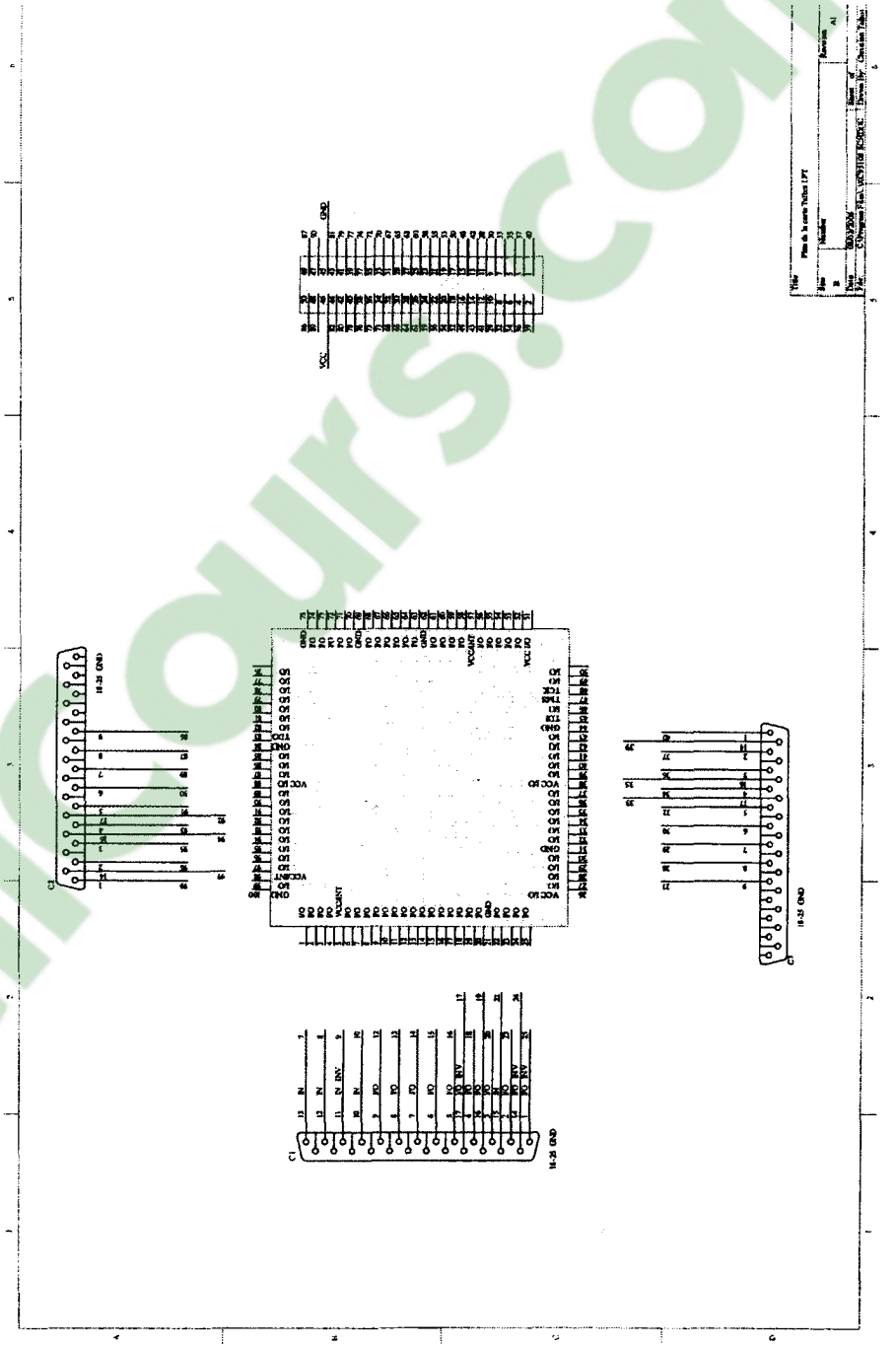


Figure III.1 : Diagramme schématique du circuit d'interface CPLD 8 bits – 12 bits

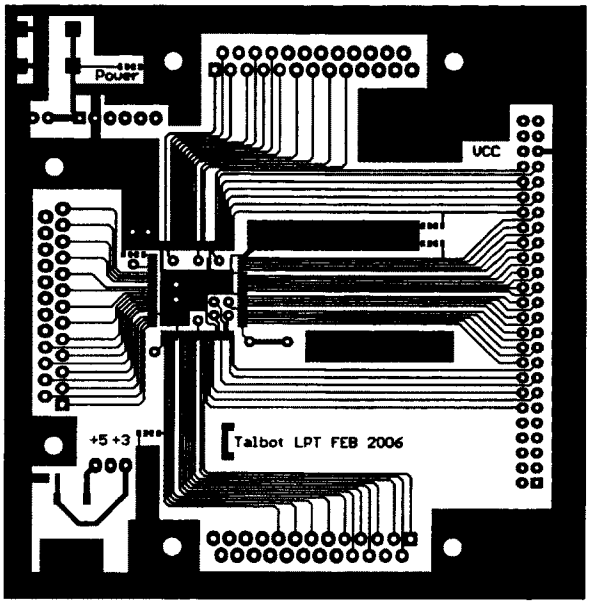


Figure III.2 : Topologie du circuit d'interface CPLD 8 bits – 12 bits

ANNEXE IV

TOPOLOGIE DE CIRCUIT IMPRIMÉ DU SYNTONISEUR

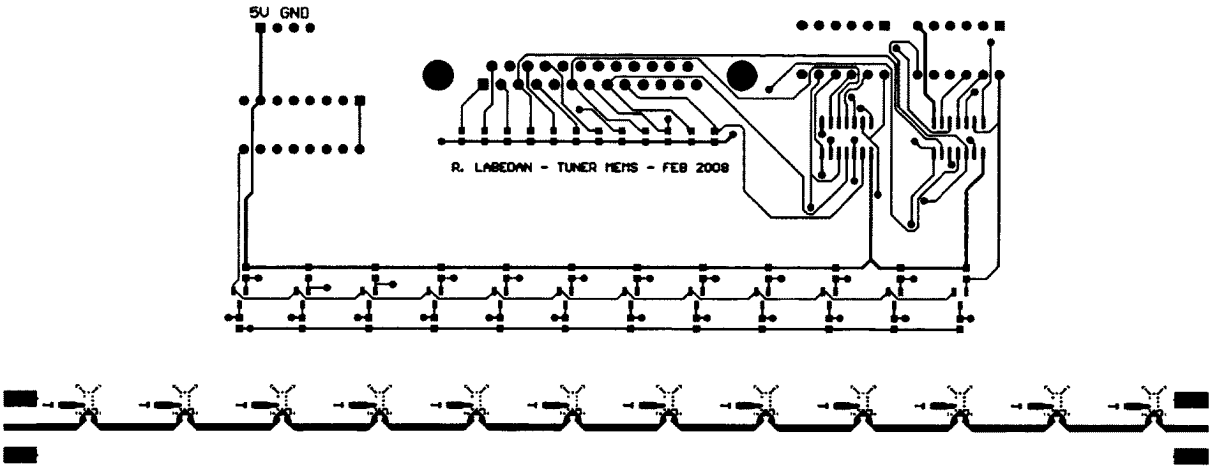


Figure IV.1 : Topologie du dessus du circuit imprimé.

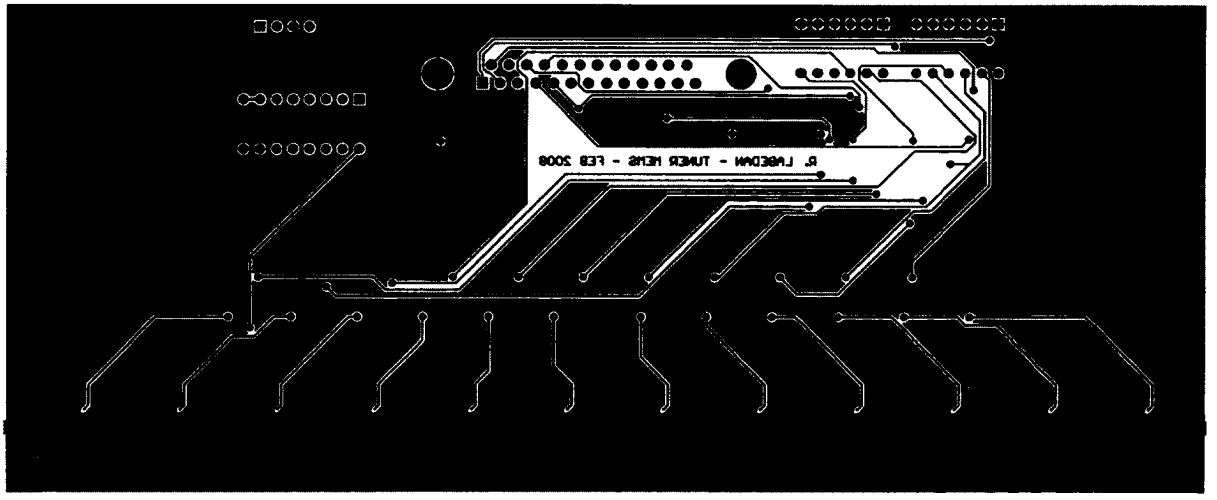


Figure IV.2 : Topologie du dessous du circuit imprimé.

ANNEXE V

PROGRAMME DE PRISE DE MESURES AGILENT VEE

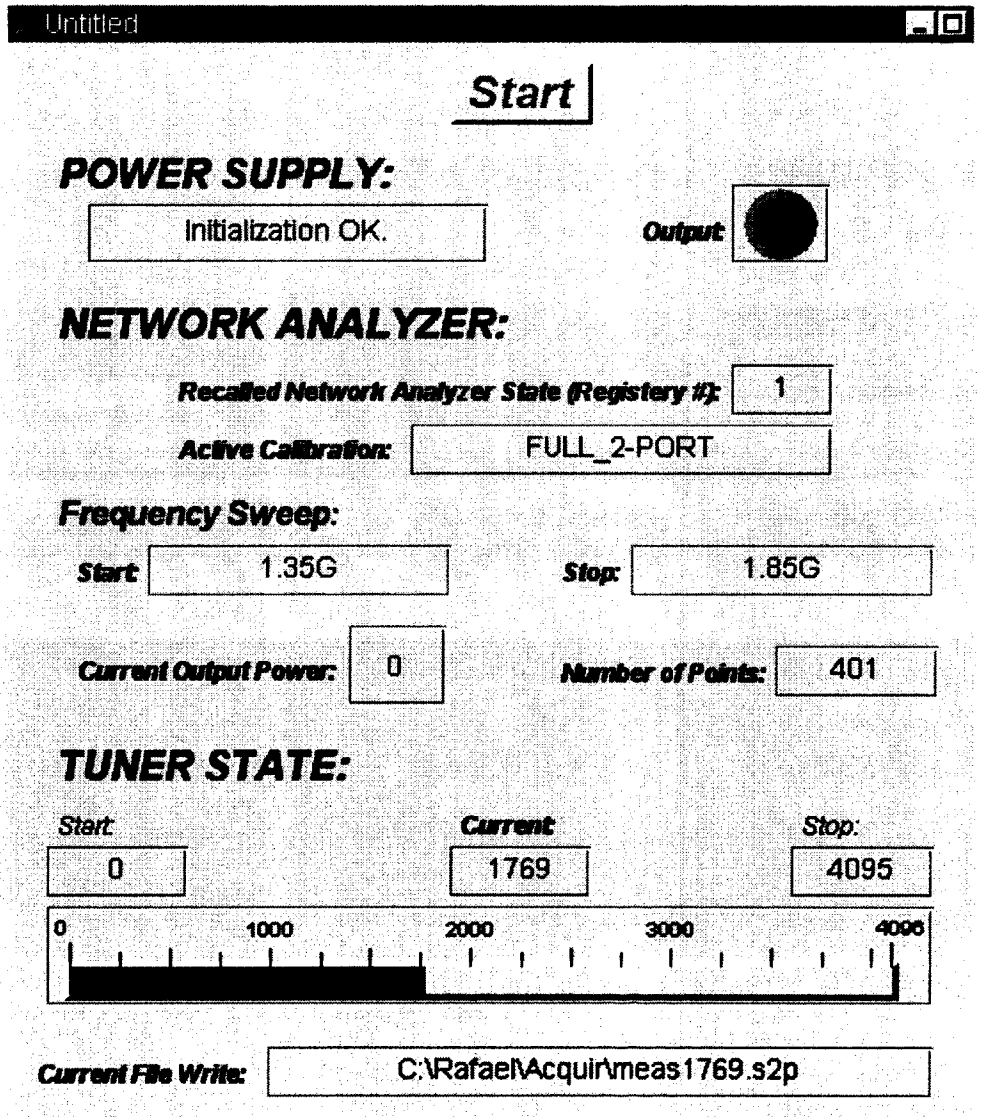


Figure V.1 : Interface du programme de prise de mesures.

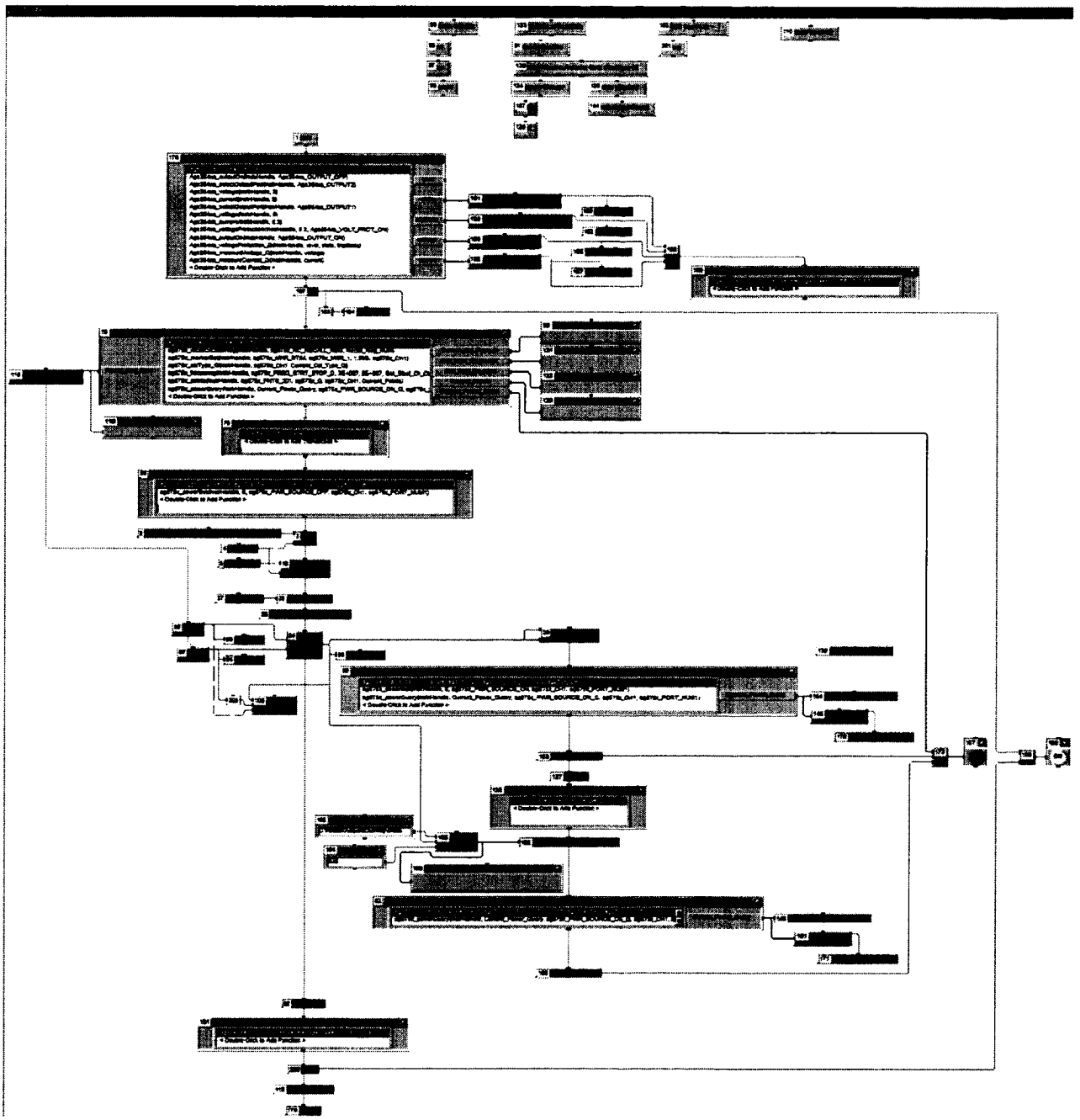


Figure V.2 : Boulcle principale du programme de prise de mesures.

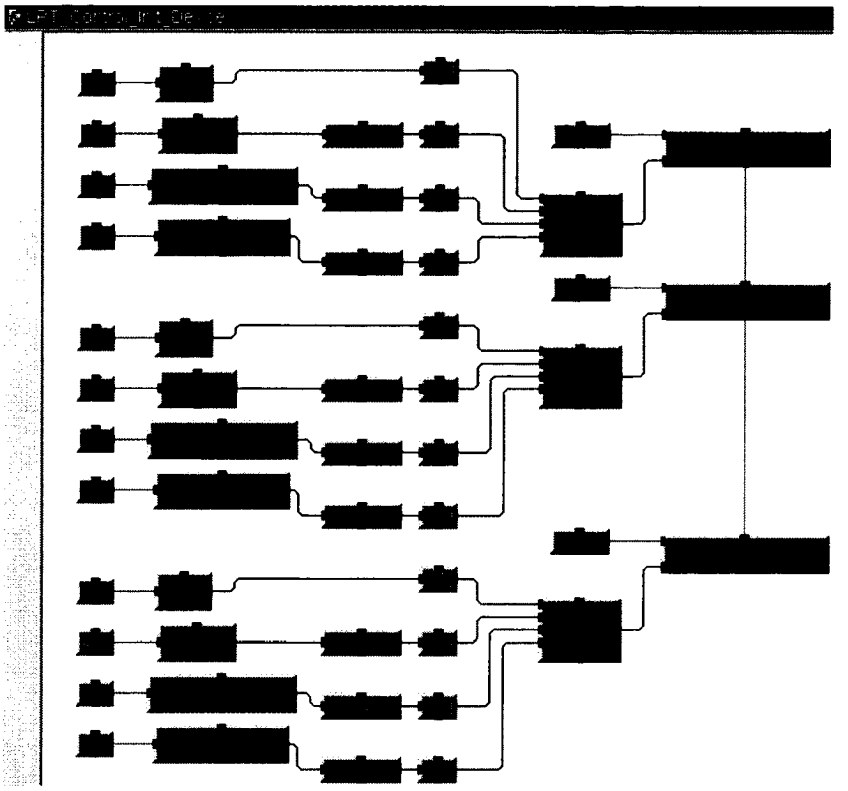


Figure V.3 : Détails de la sous fonction d'initialisation de l'interface 8 bits -12 bits.

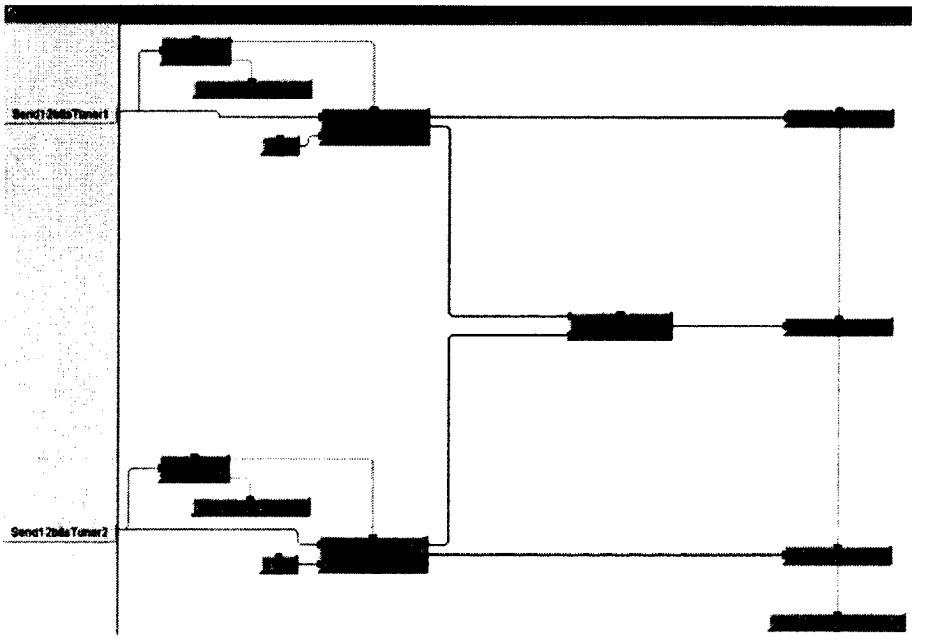


Figure V.4 : Détails de la sous fonction d'envois des 12 bits de contrôle au syntoniseur.

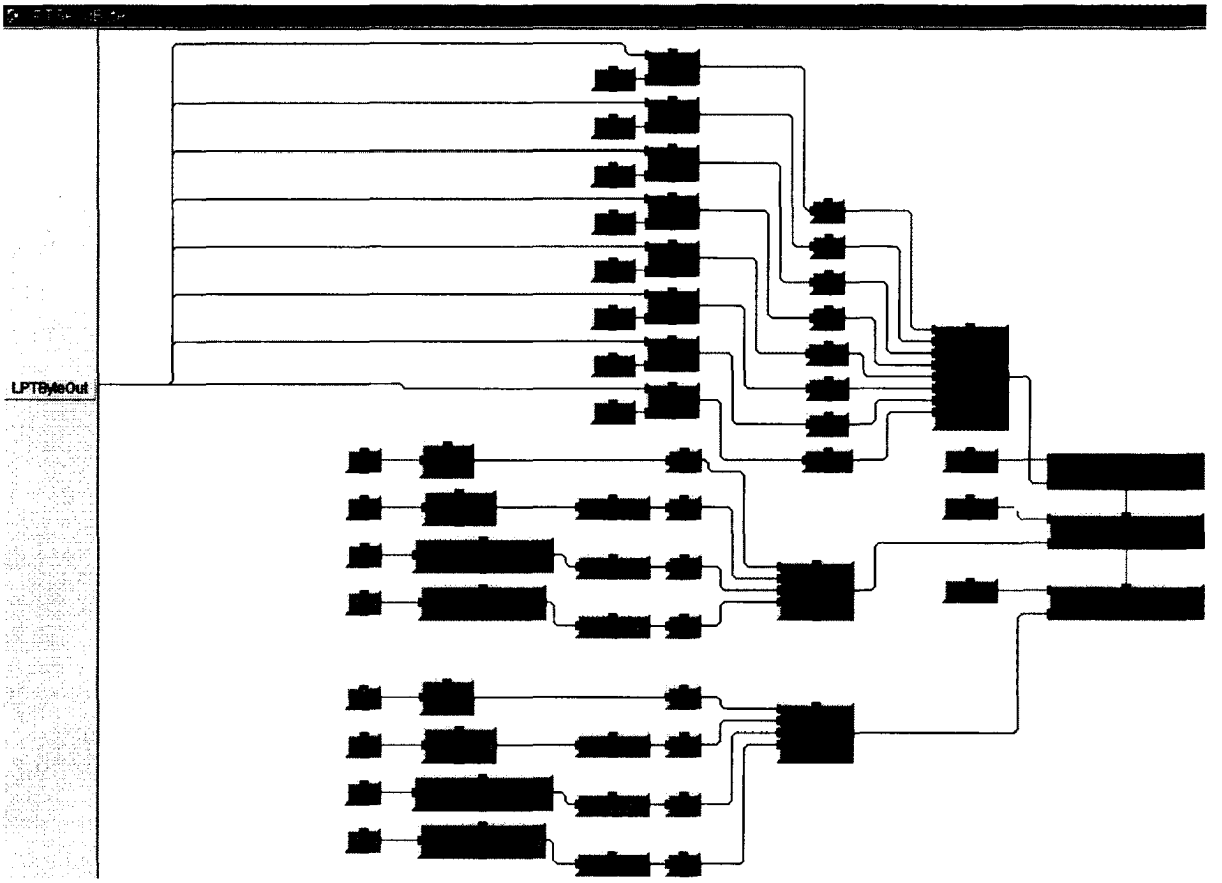


Figure V.5 : Détails de la sous fonction d'envois d'un octet sur le port parallèle.

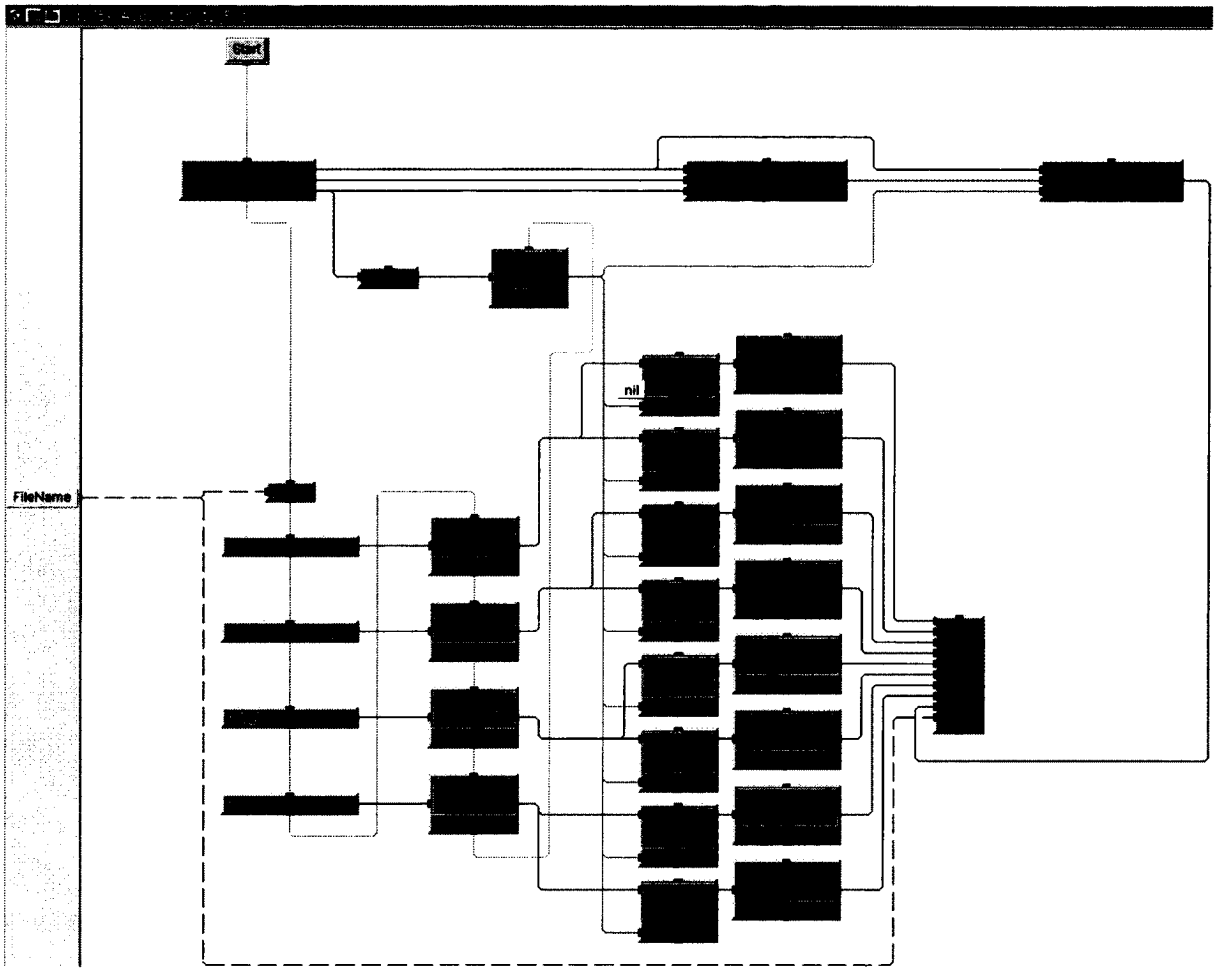


Figure V.6 : Détails de la sous fonction d'acquisition des paramètres-S et de leur sauvegarde dans des fichiers texte.

LISTE DE RÉFÉRENCES

Agilent. 2000. *Interactive Impedance Matching Model* [Modèle intéractif d'adaptation d'impédances.]. En ligne. <http://contact.tm.agilent.com/Agilent/tmo/an-95-1/classes/imatch.html>. Consulté le 11 décembre 2007.

Bode, H.W. 1945. *Network analysis and feedback amplifier design*. Ouvrage/Imprime. Coll. « Bell Telephone Laboratories Series ». N.Y.: Van Nostrand, 551 p.

Collin, Robert E. 1966. *Foundations for microwave engineering*. Coll. « Mcgraw-Hill physical and quantum electronics series ». New York, N.Y.: McGraw-Hill, xiv, 589 p.

Fano, Robert M. 1947. « Theoretical limitations on the broadband matching of arbitrary impedances ». Thesis, Cambridge, Massachusetts Institute of Technology, 108 p. <http://hdl.handle.net/1721.1/12909>.

Gutmann, R. J., et D. J. Fryklund. 1987. « Characterization of Linear and Nonlinear Properties of GaAs MESFET's for Broad-Band Control Applications ». Microwave Theory and Techniques, IEEE Transactions on, vol. 35, n° 5, p. 516-521.

Hand, T., et S. Cummer. 2007. « Characterization of Tunable Metamaterial Elements Using MEMS Switches ». Antennas and Wireless Propagation Letters, IEEE, vol. 6, p. 401-404.

Minnis, B. J. 1987. « A Printed Circuit Stub Tuner for Microwave Integrated Circuits ». Microwave Theory and Techniques, IEEE Transactions on, vol. 35, n° 3, p. 346-349.

Pozar, David M. 2005. *Microwave engineering*, 3rd. Hoboken: J. Wiley, xvii, 700 p. <http://www.loc.gov/catdir/toc/wiley041/2003065001.html>.

Talbot, Christian. 2006. *Rapport de recherche sur le duplexeur pour la compagnie Ultra Electronics TCS*. Papier, Document Word. Montréal: LACIME - École de technologie supérieure, 27 p. Consulté le 1 juin 2006.

BIBLIOGRAPHIE

Abramowicz, A. 1995. « New model of coupled transmission lines ». Microwave Theory and Techniques, IEEE Transactions on, vol. 43, n° 6, p. 1389-1392.

Badoual, Robert, Christian Martin et Sylvie Jacquet. 1993. *Les micro-ondes*, 2e éd. revue et corr. Coll. « Technologies de l'université à l'industrie. Série électronique ». Paris: Masson, 2 v. p.

Bode, H.W. 1945. *Network analysis and feedback amplifier design.* Ouvrage/Imprime. Coll. « Bell Telephone Laboratories Series ». N.Y.: Van Nostrand, 551 p.

Bushyager, N., K. Lange, M. Tentzeris et J. A. Papapolymerou J. Papapolymerou. 2002. « Modeling and optimization of RF reconfigurable tuners with computationally efficient time-domain techniques ». In *Microwave Symposium Digest, 2002 IEEE MTT-S International*, sous la dir. de Lange, K. Vol. 2, p. 883-886.

Cameron, R. J. 2003. « Advanced coupling matrix synthesis techniques for microwave filters ». Microwave Theory and Techniques, IEEE Transactions on, vol. 51, n° 1, p. 1-10.

Cameron, R. J., J. C. Faugere et F. Seyfert. 2005. « Coupling matrix synthesis for a new class of microwave filter configuration ». In *Microwave Symposium Digest, 2005 IEEE MTT-S International.* p. 4 pp.

Collin, Robert E. 1966. *Foundations for microwave engineering.* Coll. « Mcgraw-Hill physical and quantum electronics series ». New York, N.Y.: McGraw-Hill, xiv, 589 p.

Deve, N., A. B. Kouki et V. Nerguizian. 2004. « A compact size reconfigurable 1-3 GHz impedance tuner suitable for RF MEMS applications ». In *Microelectronics, 2004. ICM 2004 Proceedings. The 16th International Conference on.* p. 101-104.

Fano, Robert M. 1947. « Theoretical limitations on the broadband matching of arbitrary impedances ». Thesis, Cambridge, Massachusetts Institute of Technology, 108 p. <http://hdl.handle.net/1721.1/12909>.

Gilmore, Rowan, et Les Besser. 2003. *Practical RF circuit design for modern wireless systems.* Coll. « Artech House microwave library ». Boston, Mass.: Artech House, 2 v. p.

Girbau, D., A. Lazaro et L. Pradell. 2004. « Characterization of dynamics and power handling of RF MEMS using vector measurement techniques ». Microwave Theory and Techniques, IEEE Transactions on, vol. 52, n° 11, p. 2627-2633.

Hand, T., et S. Cummer. 2007. « Characterization of Tunable Metamaterial Elements Using MEMS Switches ». Antennas and Wireless Propagation Letters, IEEE, vol. 6, p. 401-404.

Hoarau, C., P. E. Bailly, J. D. Arnould, P. Ferrari et P. Xavier. 2007. « Accurate measurement method for characterisation of RF impedance tuners ». Electronics Letters, vol. 43, n° 25, p. 1434-1436.

Ishii, T. Koryu. 1966. *Microwave engineering*. New York, N.Y.: Ronald Press, x, 339 p.

Kabir, H., Y. Wang, M. Yu et Q. J. Zhang. 2007. « Applications of Artificial Neural Network Techniques in Microwave Filter Modeling, Optimization and Design ». PIERS Online, vol. 3, n° 7, p. 1131-1135.

Lange, K. L., J. Papapolymerou, C. L. Goldsmith, A. Malczewski et J. Kleber. 2001. « A reconfigurable double-stub tuner using MEMS devices ». In *Microwave Symposium Digest, 2001 IEEE MTT-S International*. Vol. 1, p. 337-340 vol.1.

Lee, C., C. Lugo, Jr., Zheng Guizhen et J. Papapolymerou. 2005. « Low cost reconfigurable impedance tuner in organic substrate for system-on-package (SOP) applications ». In *Electronic Components and Technology Conference, 2005. Proceedings. 55th.* p. 1888-1890 Vol. 2.

Liao, Samuel Y. 1980. *Microwave devices and circuits*. Englewood Cliffs,Nj: Prentice-Hall.

McLaughlin, J. C., et K. L. Kaiser. 2007. « “Deglorifying” the Maximum Power Transfer Theorem and Factors in Impedance Selection ». Education, IEEE Transactions on, vol. 50, n° 3, p. 251-255.

Minnis, B. J. 1987. « A Printed Circuit Stub Tuner for Microwave Integrated Circuits ». Microwave Theory and Techniques, IEEE Transactions on, vol. 35, n° 3, p. 346-349.

Misra, Devendra. 2004. *Radio-frequency and microwave communication circuits : analysis and design*, 2nd. Hoboken, N.J.: Wiley-Interscience, xii, 614 p. <http://www.loc.gov/catdir/description/wiley042/2003026691.html http://www.loc.gov/catdir/toc/ecip0412/2003026691.html>.

Papapolymerou, J., K. L. Lange, C. L. Goldsmith, A. Malczewski et J. Kleber. 2003. « Reconfigurable double-stub tuners using MEMS switches for intelligent RF front-ends ». Microwave Theory and Techniques, IEEE Transactions on, vol. 51, n° 1, p. 271-278.

Pozar, David M. 2005. *Microwave engineering*, 3rd. Hoboken: J. Wiley, xvii, 700 p. <http://www.loc.gov/catdir/toc/wiley041/2003065001.html>.

Qin, Shen, et N. S. Barker. 2005a. « Reconfigurable matching with a 10-30 GHz distributed RF-MEMS tuner ». In *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings*. Vol. 4, p. 4 pp.

Qin, Shen, et N. S. Barker. 2005b. « A reconfigurable RF MEMS based double slug impedance tuner ». In *Microwave Conference, 2005 European*. Vol. 1, p. 4 pp.

Qin, Shen, et N. S. Barker. 2006. « Distributed MEMS tunable matching network using minimal-contact RF-MEMS varactors ». Microwave Theory and Techniques, IEEE Transactions on, vol. 54, n° 6, p. 2646-2658.

Russer, P. 2006. *Electromagnetics, microwave circuit and antenna design for communications engineering*, 2nd. Coll. « Artech House antennas and propagation library ». Boston, Mass.: Artech House, xx, 729 p.

Simon, W., T. Mack, B. Schauwecker, K. M. Strohm et J. F. Luy. 2005. « Toggle switch: investigations of an RF MEMS switch for power applications ». Microwaves, Antennas and Propagation, IEE Proceedings -, vol. 152, n° 5, p. 378-384.

Symeon, Nikolaou, R. Bairavasubramanian, C. Lugo, Jr., I. Carrasquillo, D. C. Thompson, G. E. Ponchak, J. Papapolymerou et M. M. Tentzeris. 2006. « Pattern and frequency reconfigurable annular slot antenna using PIN diodes ». Antennas and Propagation, IEEE Transactions on, vol. 54, n° 2, p. 439-448.

Tomiyasu, K. 1955. « Intrinsic Insertion Loss of a Mismatched Microwave Network ». Microwave Theory and Techniques, IEEE Transactions on, vol. 3, n° 1, p. 40-45.

Unlu, M., K. Topalli, H. I. Atasoy, E. U. Temocin, I. Istanbulluoglu, O. Bayraktar, S. Demir, O. A. Civi, S. Koc et T. Akin. 2006. « A Reconfigurable RF MEMS Triple Stub Impedance Matching Network ». In *Microwave Conference, 2006. 36th European*. p. 1370-1373.

Vaha-Heikkila, T., et G. M. Rebeiz. 2004. « A 20-50 GHz reconfigurable matching network for power amplifier applications ». In *Microwave Symposium Digest, 2004 IEEE MTT-S International*. Vol. 2, p. 717-720 Vol.2.

Vaha-Heikkila, T., J. Varis, J. Tuovinen et G. M. Rebeiz. 2004. « A reconfigurable 6-20 GHz RF MEMS impedance tuner ». In *Microwave Symposium Digest, 2004 IEEE MTT-S International*. Vol. 2, p. 729-732 Vol.2.

Vaha-Heikkila, T., J. Varis, J. Tuovinen et G. M. Rebeiz. 2005. « W-band RF MEMS double and triple-stub impedance tuners ». In *Microwave Symposium Digest, 2005 IEEE MTT-S International*. p. 4 pp.

Whatley, R. B., Zhou Zhen et K. L. Melde. 2006. « Reconfigurable RF impedance tuner for match control in broadband wireless devices ». Antennas and Propagation, IEEE Transactions on, vol. 54, n° 2, p. 470-478.

Zhen, Zhou, et L. Melde Kathleen. 2006. « Packaging Considerations for a Compact Reconfigurable RF Tuner with a Broadband Tuning Range ». In *Electrical Performance of Electronic Packaging*. p. 237-240.

Zhen, Zhou, et K. L. Melde. 2007. « Frequency Agility of Broadband Antennas Integrated With a Reconfigurable RF Impedance Tuner ». Antennas and Wireless Propagation Letters, vol. 6, p. 56-59.

Zheng, G., P. L. Kirby, S. Pajic, A. A. Pothier A. Pothier, P. A. Blondy P. Blondy, J. A. Papapolymerou J. Papapolymerou et Z. A. Popovic Z. Popovic. 2004. « A monolithic reconfigurable tuner with ohmic contact MEMS switches for efficiency optimization of X-band power amplifiers ». In *Silicon Monolithic Integrated Circuits in RF Systems, 2004. Digest of Papers. 2004 Topical Meeting on*, sous la dir. de Kirby, P. L. p. 159-162.