SPICE – *Simulation Program with Integrated Circuit Emphasis*

Benjamin.Barras@epfl.ch, Domaine-IT, Responsable Linux

We will discover in this article, a simulation of electronic circuits.

Nous allons découvrir dans cet article, un logiciel de simulation de circuits électroniques.

Fiche descriptive

SPICE		
Domaine d'utilisation ✦ Logiciel de simulation de circuits électroniques		
Licence ✦ BSD	**langue** ✦ anglais	**version** ✦ 3f5
Autre alternative libre ✦ Ngspice		
Alternatives non libres ✦ Pspice ✦ Micro-Cap ✦ MacSpice 3f5 ✦ ...		
Sites Web ✦ bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE ✦ ngspice.sourceforge.net		
Plates-formes supportées ✦		

Introduction

SPICE est un logiciel qui m'a beaucoup émerveillé lorsque j'étais étudiant, de par sa puissance et sa simplicité d'utilisation. Il est vrai que nous faisions beaucoup de travaux pratiques en électronique, et le simple fait de pouvoir mettre son circuit sous forme de liste, appelée *netlist*, et de le simuler était une découverte fascinante. SPICE a servi de modèle à nombre d'autres programmes de simulation, dans les universités et dans l'industrie, grâce à son modèle précoce *Open Source*. Aujourd'hui toute l'électronique se concentre sur une puce, et les logiciels de simulation utilisés sont plutôt du type VHDL [1], voire mixte continu/événementiel [2], mais dont la plupart sont inspirés de SPICE. Actuellement, on édite de manière graphique son schéma et on lance directement la simulation depuis son interface graphique. Ce qu'il faut savoir c'est que bon nombre de ces interfaces graphiques construisent une *netlist* et c'est un SPICE adapté qui fera la simulation par-derrière. La première version de SPICE date de 1972 et a été écrite en fortran, rapidement suivie par la deuxième version en 1975. Il faudra attendre 1989 pour voir la troisième version (définitive) de SPICE, écrite en C. Il est remarquable d'observer que la troisième version fût la dernière, avec des mises à jour mineures et qui se déclinent en lettre, la dernière étant 3f5. Aujourd'hui, et dans cet article, on utilisera Ngspice [3] qui est basé sur la dernière version de SPICE, et qui se trouve sous licence GNU GPL2. Il est fourni par le groupe de développement gEDA [4] qui a repris le flambeau pour le développement de tous les logiciels de simulation ou de conception de circuits électroniques.

Installation

L'installation de Ngspice se fait très facilement, sous Linux:

```
# Ubuntu
apt-get install ngspice
```

```
# Fedora
yum install ngspice
```

Pour les autres versions Linux/Unix, on pourra le compiler très facilement en récupérant les sources sur le site officiel de Ngspice. De plus, il existe également des binaires pour Windows et Mac OS X.

Description

SPICE utilise des composants modélisés par un ensemble d'équations, généralement non-linéaires. En reliant chaque élément les uns aux autres, cela permet à SPICE de créer un ensemble d'équations. SPICE calcule alors le point au temps T+1 en linéarisant le circuit autour de son point d'équilibre au temps T, équations qui seront résolues de manière itérative en appliquant les lois de Kirchhoff [5].

Structure du fichier

Le circuit à simuler est décrit par un ensemble de lignes, qui définissent la topologie du circuit ainsi que la valeur des éléments et un ensemble de lignes de contrôles qui définissent les paramètres des modèles ou les paramètres de la simulation. Deux lignes sont essentielles:

- la première ligne qui sera le titre par défaut;
- la dernière ligne qui se termine par **.END**.

Le reste des lignes n'ont pas d'ordre spécifique et les lignes vides seront ignorées. Une ligne qui commence par un point **.** définit une ligne de contrôle, par exemple une ligne qui commence par **.TITLE** nous permettra de changer le titre de notre simulation en cours de route.
Un commentaire est une ligne qui commence par un astérisque *****, et on mettra un point-virgule **;** si le commentaire est en fin de ligne. Une ligne commençant par un plus **+** signifie qu'elle décrit la suite de la ligne précédente.

Définition des éléments

Chaque élément est défini par une ligne qui contient:

- le nom de l'élément;
- les nœuds auxquels l'élément est connecté;
- la valeur et les paramètres qui décrivent l'élément.

La première lettre définit le type, par exemple une résistance commencera par la lettre R et sera suivie d'un ou plusieurs caractères. Le nom est insensible à la casse. En conséquence R1, Rout, Rsave, R3ABX seront des noms corrects pour représenter une résistance. Voici une liste des principaux composants définis par la première lettre:

```
R pour une résistance
C pour une capacité
L pour une inductance
V pour une source de tension
I pour une source de courant
D pour une diode,
Q pour un transistor à jonction
J pour un transistor à effet de champs
X pour un sous-circuit
. ...
```

Pour avoir la liste complète, on se référera à la documentation officielle **[6]**. Ensuite, on placera notre élément entre deux ou plusieurs nœuds dont le nom est composé de caractères quelconques insensibles à la casse. Par défaut, le nœud `0` ou `gnd` représente la masse. Il suffit alors de décrire sur la même ligne les noms des nœuds auxquels notre élément est associé. Pour la valeur d'un élément, on utilisera le tableau suivant pour les facteurs d'échelle:

```
T   Tera    10E12
G   Giga    10E9
Meg Mega    10E6
K   Kilo    10E3
m   milli   10E-3
u   micro   10E-6
n   nano    10E-9
p   pico    10E-12
f   femto   10E-15
```

qui facilitera l'écriture, mais on peut également utiliser des nombres réels tels que 0.001, 10e3, 100E-6. Les lettres qui ne représentent pas des facteurs d'échelle et qui suivent directement le nombre ou les lettres qui suivent directement un facteur d'échelle, seront ignorées. Par exemple 10, 10V, 10Volts représentent le même nombre 10 ou 1K, 1Kohm représentent le même nombre 1000. Quelques exemples:

```
 +---------------- Élément
 |   +------------ Noeud +
 |   |   +-------- Noeud -
 |   |   |   +---- Valeur
 |   |   |   |
R10  1   2   18K
C20  3   4   10u
R20  3   0   10KOhm
C10  1   0   10nF
```

Exemple

Voici l'exemple d'un circuit électronique simple, dont le schéma est le suivant:

```
Simple circuit RC

           1k
Vin +---\/\/\/------------+ Vout
           R1             |
                         _|_
                      C1 ___ 10uF
                         _|_
                          |
```

et le fichier descriptif *(netlist)* correspondant qui contient tout ce que nous venons de décrire, soit:

```
Ceci est mon titre obligatoire

* Ligne de commentaire

R1   Vin   Vout   1K     ; Commentaire
C1   Vout  0      10uF

* Changement de titre
.TITLE Simple circuit RC

.END
```

Source de tension et de courant

La forme générale est:

```
 +----------------- Source de tension/courant
 |   +------------- Noeud +
 |   |   +--------- Noeud -
 |   |   |   +----- Type de source/analyse
 |   |   |   |
Vxx  N+  N-  <TYPE>
Ixx
```

On ne parlera ici, que de sources de tension ou de courant indépendantes. Le type de source est intimement lié au type d'analyse que l'on souhaite faire. Nous verrons en détail, les types d'analyses:

- DC: courant continu *(Direct Courant)*
  La source est de type courant continu, mais on peut faire varier la source en amplitude (balayage) afin d'étudier le circuit en fonction de cette variation.
- AC: courant alternatif *(Alternating Courant)*
  La source est de type sinusoïdal utilisée pour l'analyse fréquentielle de petits signaux. Ici aussi, on peut faire varier la source (balayage), mais en fréquence.
- TRAN: transitoire *(TRANsient)*
  L'analyse transitoire consiste à faire varier la source en fonction du temps.

Il n'est pas obligatoire de décrire le type de source sur cette ligne, on le fera par la suite lorsque l'on décrira les principaux types de source ainsi que les différents types d'analyses associés. On se référera à la documentation pour une description complète des autres types d'analyses.

Lignes de contrôles

Modèles

Il est important de comprendre que toute la simulation est basée sur des modèles dont la description et les paramètres sont donnés dans la documentation. La commande **.MODEL** permet de définir un modèle pour un composant. Par exemple pour la diode 1n4007 qui est très utilisée dans les alimentations, on aura ceci:

```
D1 0 1 1n4007

.MODEL 1n4007 d (is=76.9p rs=42.0m bv=1.00k
+ ibv=5.00u jo=26.5p m=0.333 n=1.45 tt=4.32u)
```

Sous-circuit

Il peut être utile de décrire des parties de circuits qui se répètent souvent par un bloc de commande, prenons le circuit:

```
N1     10K      Ni      10k     N2
+----\/\/\/-----+-----\/\/\/----+
                |
               _|_
               ___ 1uF
                |
               _|_ N0
```

on mettra ce bloc sous la forme suivante:

```
.SUBCKT simpleFiltre N1 N0 N2
R1    N1    Ni    10K
C1    Ni    N0    1u
R2    Ni    N2    10k
.ENDS
```

L'appel au sous-circuit se fera ainsi:

```
* Deux sous-circuits en cascade
XF1 7 0 8 simpleFiltre
XF2 8 0 9 simpleFiltre
```

où la lettre `X` est réservée aux éléments *sous-circuit*. Ici les éléments F1 et F2 se trouvent entre les nœuds 7 0 et 9 et font appel au sous-circuit `simpleFiltre`. Cela correspond au circuit:

```
7    10K         10k    8    10K         10k     9
+--\/\/\/--+--\/\/\/--+--\/\/\/--+--\/\/\/--+
           |                     |
          _|_                   _|_
          ___ 1uF               ___ 1uF
           |                     |
          _|_                   _|_
           0                     0
```

On peut également passer des paramètres aux sous-circuits, un peu de la même manière que l'appel à des fonctions ou sous-programmes dans les langages de programmation. On se référera à la documentation pour en savoir plus.

Inclure des fichiers

Une commande importante qui nous permet de réutiliser facilement des modèles ou des sous-circuits, est la commande

```
.INCLUDE nomDuFichier
```

qui permet, comme son nom l'indique, d'inclure une portion de fichier dans notre fichier principal.

Analyse

Il s'agit ici de définir le type d'analyse que l'on veut faire, afin que *Ngspice* sache ce qu'il doit calculer. C'est une partie un peu technique, difficile à comprendre en première lecture et quasiment jamais expliquée dans les divers tutoriels. On va commencer gentiment, et il faudra bien regarder les exemples afin de bien comprendre le sujet. On ne va parler que des trois principaux types d'analyse qui sont:

DC

Utilisée pour étudier la fonction de transfert d'un circuit ou d'un composant par balayage d'une source continue de tension (Vxx) ou de courant (Ixx). On peut également faire varier une résistance (Rxx) ou la température (TEMP):

```
* Syntaxe
.DC nomDeLaSource vStart vStop vIncrément
+ [nomDeLaSource2 vStart2 vStop2 vIncrément2]
```

Exemples:

```
* Tension continue fixe de 5 Volts
Vcc 10 0 DC 5

* Tension d'entrée
Vin 10 0

* .DC nomDeLaSource vStart vStop vIncrément
.DC Vin 0 5 0.1
```

On peut faire varier une seconde source (facultatif). Dans ce cas, la première source sera balayée pour chacune des valeurs de la seconde source. Cela est très souvent utilisé pour mesurer les caractéristiques de semi-conducteur.

AC

Utilisé pour étudier l'analyse fréquentielle à l'aide de petits signaux analogiques, on varie la fréquence à l'entrée et on mesure la réponse du circuit ou d'un composant:

```
*Vxx N1 N- AC ACMAG ACPHASE
Vin 10 0 AC 3 0

*.AC DEC nd fStart fStop
*.AC OCT no fStart fStop
*.AC LIN n1 fStart fStop

.AC LIN 100 1M 10M
```

Avec `ACMAG` qui correspond à l'amplitude du signal (1 par défaut) et `ACPHASE` la phase du signal (0 par défaut). DEC signifie une variation de fréquence par décade, et `nd` est le nombre de points par décade. OCT signifie une variation de fréquence par octave, et `no` est le nombre de points par octave. LIN signifie une variation de fréquence linéaire, et `np` est le nombre de points. `fStart` est la fréquence de départ et `fStop` la fréquence finale.

TRAN

Il s'agit ici, d'une analyse purement temporelle. On utilise une source de forme variable, qui sera utilisée pour calculer le comportement du circuit en fonction du temps. On doit tout d'abord définir le temps de la manière suivante:

```
* .TRAN tStep tStop <tStart <tMax>> <UIC>
.TRAN 10m 1s 0s
```

Avec bien sûr `tStart` le temps de départ (facultatif, `0s` par défaut), `tStop` le temps final et `tStep` le pas temporel. `tMax` est le pas de calcul, par défaut *ngSpice* calcule `(tStop-tStart)/50.0` et utilise ce dernier s'il est plus petit que `tmax`. `UIC` *(Use Initial Condition)* indique à Ngspice d'utiliser les conditions initiales définies par **.IC** dans une ligne de contrôle. Il nous reste encore à définir la forme du signal, les trois plus importants types de signaux variables dans le temps sont:

SINE

Signal de type sinusoïdal , qui s'utilise comme suit:

```
* Vxx N+ N- SIN(V0 VA FREQ TD THETA)

* Vxx = V0 si t < TD
* Vxx = V0 + VA*exp[(t-TD)*THETA]*
* sin(2*pi*FREQ*(t-TD)) si t > TD ou t = TD
* Par défaut, TD et THETA sont à 0 :
* Vxx = V0 + VA*sin(2*pi*FREQ*t)

* Sinusoïde de fréquence 1MHz, amplitude 3V et
* offset de 0V (pas de composante continue)
Vin 10 0 SIN(0 3 1MEG)
```

PULSE

Signal de type impulsion, périodique ou non périodique, dont la description est la suivante:

```
* PULSE(V1 V2 TD TR TF PW PER)

*         +-----+ V2
*        /       \
*       /         \
*------+           +----- V1
*      ^  ^      ^  ^
*      |  |      |  |
*    >|  |< TR
*      |  |     >|  |< TF
*      |  |      |  |
*      | >|      |< PW
*      |
*     TD

* Pulse de 1V de durée de 5ms, avec temps de
* montée et descente de 2ns,
* qui commence à 100ns et de période de 20ms
VIN 3 0 PULSE(0 1 100NS 2NS 2NS 5MS 20MS)
```

Pour une unique impulsion, il faudra utiliser `PER=tStop` ou `tStop` qui sera défini dans l'analyse transitoire.

PWL

Linéaire par morceaux *(Piece-Wise Linear)*

```
*                    T5
*                    + V5
*                   / \
*             T3    /   \
*          V3 +----+ V4   \
*            /     T4      \     T7
*           /                V6 +---+ V7
*          /                    T6   \
* V1 +---+ V2                      V8 +---+ V9
*    T1  T2                          T8  T9

* PWL(T1 V1 <T2 V2 T3 V3 T4 V4 . . .Tn Vn >)
* + <r=value> <td=value>
```

où la tension (ou le courant) prend la valeur `V1` au temps `T1`, la valeur `V2` au temps `T2`, et passe linéairement de `V1` à `V2`.
`r` est la valeur du temps pour lequel le signal sera périodique, par exemple si `r=T3` le signal sera répétitif de `T3` à `Tn`, une fois `Tn` atteint. Si `r` n'est pas présent, le signal ne sera pas périodique.
`td` est le délai avant d'envoyer le signal.

Simulation

Il faut donner à Ngspice des instructions ou des consignes afin qu'il sache ce qu'il doit afficher. Dans tous les cas, Ngspice calcule le point d'équilibre du système (OP: *Operating Point*) avant de commencer sa simulation. Il existe deux sortes d'informations disponibles pour l'utilisateur, la première sous forme graphique (`.PLOT`) et la deuxième sous forme de texte (`.PRINT`) où les données peuvent être utilisées par une autre application. La syntaxe est:

```
.PRINT TYPE OV1 OV2 OV3 ...
.PLOT TYPE OV1 OV2 OV3 ...
```

où `TYPE` définit le type d'analyse que l'on souhaite, à savoir

- DC
- AC
- TRAN
- NOISE
- DISTO.

Les variables de sorties sont `OV1`, `OV2`, etc. qui peuvent être des sources de tensions ou de courant. Les valeurs de tension (ou de courant) peuvent être spécifiées de la manière suivante, avec le préfixe `V` pour tension ou `I` pour courant :

- M: amplitude *(Magnitude)*,
- DB: amplitude en décibel (dB),
- P: phase,
- R: partie réelle *(Real part)*,
- I: partie imaginaire *(Imaginary part)*.

Exemples:

```
.PLOT DC V(1,2) V(3) I(Vmeas)
.PRINT TRAN V(3,1) I(Vmeas)
.PLOT AC VM(3,0) VDB(4,2) VM(2,1) VP(3,1) IR(V2)
```

Une tension est toujours définie entre deux nœuds, la masse est prise (nœud 0) par défaut pour le deuxième nœud si une seule valeur est donnée.

Lancer Ngspice

Il y a deux manières de lancer Ngspice, en mode interactif ou en mode script (automatique).

Mode interactif

Le premier lancement se fait comme suit:

```
* Lancement de Ngspice
ngspice

ngspice 1 ->source monFichier.cir
ngspice 2 ->run
ngspice 3 ->plot v(1) v(2) ...

* Autre manière, équivalente
ngspice monFichier.cir

ngspice 1 ->run
ngspice 2 ->plot v(1) v(2) ...
```

> Si pour le fichier de description (netlist) la casse n'a pas d'importance, il en est tout autrement en mode interactif. Bien que la documentation soit muette sur le sujet, les minuscules sont recommandées. Par exemple, la tension VDB(2) sera inconnue alors que vdb(2) sera connue.

Mode script

On peut lancer une simulation de manière automatique ainsi:

```
ngspice monFichier.cir
```

Il suffit de placer toutes les commandes que l'on aurait exécutées en mode interactif entre les deux lignes de contrôle `.CONTROL` et `.ENDC`, et elles seront exécutées automatiquement. Par exemple:

```
.CONTROL
run
plot V(1) vdb(2)
.ENDC
```

Sauver des données

On peut sauver et récupérer des données à l'aide des deux commandes suivantes:

```
* Sauvegarde
ngspice 1 ->run data.raw

* Récupération
ngspice 1 ->load data.raw
ngspice 2 ->plot v(1) v(2)
```

> La commande `display` est une commande utile pour le mode interactif qui vous permet de voir tous les vecteurs qui ont été calculés.

Pratique

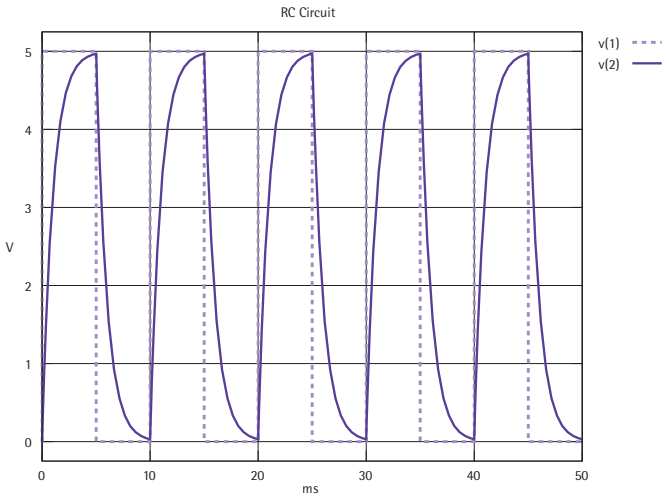
La meilleure façon d'apprendre à utiliser Ngspice, est de l'utiliser. Pour cela, on va passer en revue quelques exemples bien concrets que l'on trouve dans la pratique.

Filtre RC

Reprenons notre filtre RC du début qui correspond à notre premier exemple. On va faire deux analyses, une en fréquence et une temporelle avec une fonction carrée. Commençons par l'analyse temporelle:

```
* RC Circuit
R1   1  2   1K
C1   2  0   1uF
* Tau = R1*C1 = 1ms
* On compte 5*Tau = 5ms pour charger C1
* Pulsation de 5V, période 10ms et symétrique 5ms
Vin 1 0 DC 0 PULSE(0 5 0ms 1us 1us 5ms 10ms)
* On regarde 5 impulsions
.TRAN 0.5ms 50ms
.END
```

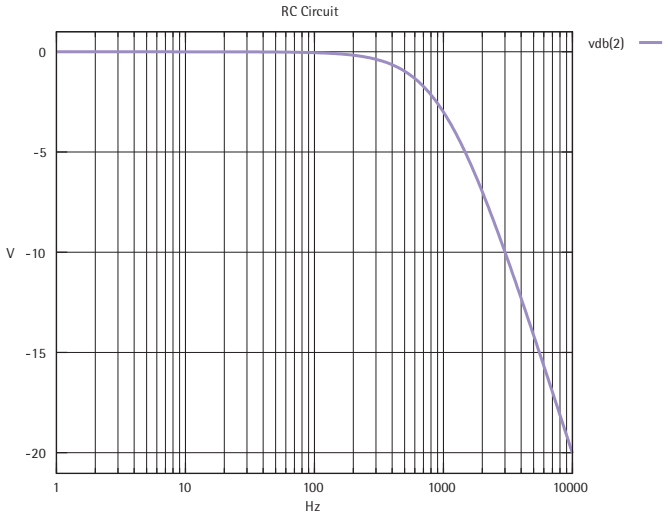
```
ngspice rc1.cir

ngspice 1 ->run
ngspice 2 ->plot v(1) v(2)
```



Maintenant, l'analyse fréquentielle:

```
* RC Circuit
R1   1  2   1.59K
C1   2  0   0.1uF
* Source sinusoïdale d'amplitude 1V
Vin 1 0 DC 0 AC 1
* Analyse fréquentielle fc = 1/2*pi*R1*C1 = 1kHz
.AC DEC 100 1 10K
.END
```

```
ngspice rc2.cir

ngspice 1 ->run
ngspice 2 ->plot vdb(2)
```



Doubleur de tension

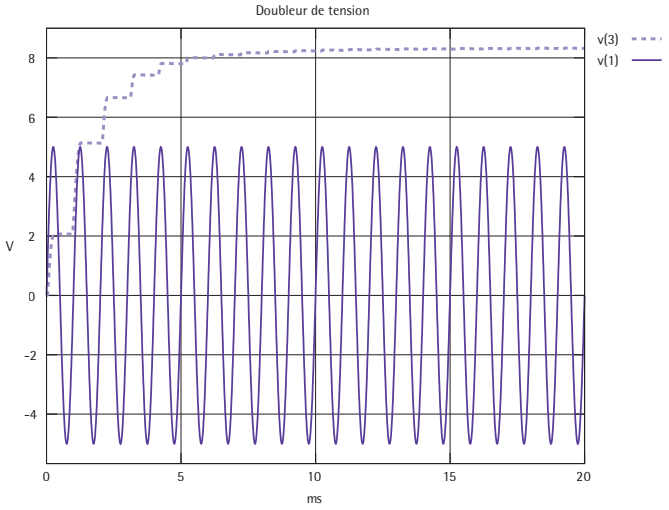
Un joli circuit qui montre bien toute la puissance de la simulation:

```
    1     ||    2    |\|   3
Vin +-----||---+---|-|---+-----+
          ||   |    |/|   |     |
          C1   |    D2   |    ---
               ---       -----  | |
               /|\       -----  | |
          D1   ---      C2  |    --- Ru
    0          |          |     |
    +----------+----------+-----+
```

On le lancera automatiquement de cette manière:

```
ngspice doubleur.cir
```

```
* Circuit : doubleur.cir
Ru 3 0 1k
C1 1 2 470u
C2 3 0 470u
D1 0 2 1n4007
D2 2 3 1n4007
Vin 1 0 DC 0 SIN(0 5 1k)
.MODEL 1n4007 d(is=76.9p rs=42.0m bv=1.00k
+ ibv=5.00u cjo=26.5p m=0.333 n=1.45 tt=4.32u)
.TRAN 0.01m 20m 0
.CONTROL
RUN
PLOT v(3) v(1) ydelta 2
.ENDC
.END
```



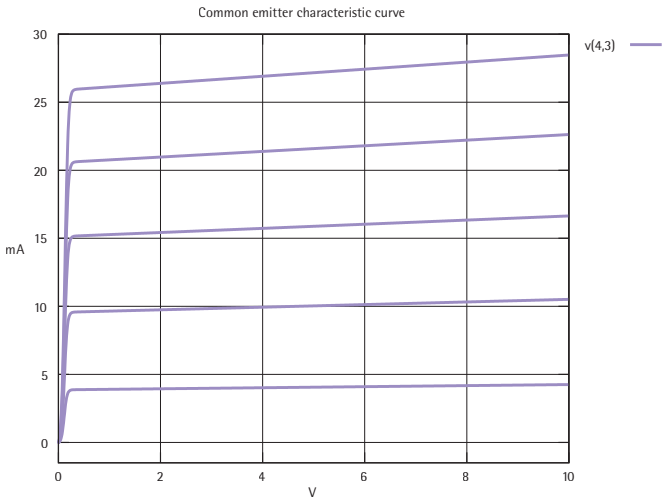
Transistor

Voici un exemple où l'on a deux sources variables:

```
                          3             4
                          +---/\/\/\---+ VCC
                         /       R2
   1      R1      2     |/
V1 +---/\/\/\---+---+   | 2N3904
                    |   |\
                    |    \
                    |     + E
                    |     |
                    |    _|_
```

```
# BJT Caractéristique de sortie
Q1 3 2 0 Q2N3904
VCC 4 0 DC 10
R2 4 3 1
V1 1 0 DC 10
R1 1 2 100k
.DC VCC 0 10 0.01 V1 2 10 2
.MODEL Q2N3904 NPN (IS=1E-14 VAF=100 Bf=300
+ IKF=0.4 XTB=1.5 BR=4 CJC=4E-12 CJE=8E-12
+ RB=20 RC=0.1 RE=0.1 TR=250E-9 TF=350E-12 ITF=1
+ VTF=2 XTF=3)
.PLOT DC V(4,3)
.END
```

```
ngspice transistor.cir

ngspice 1 -> run
ngspice 2 -> plot v(4,3)
```



Circuit logique

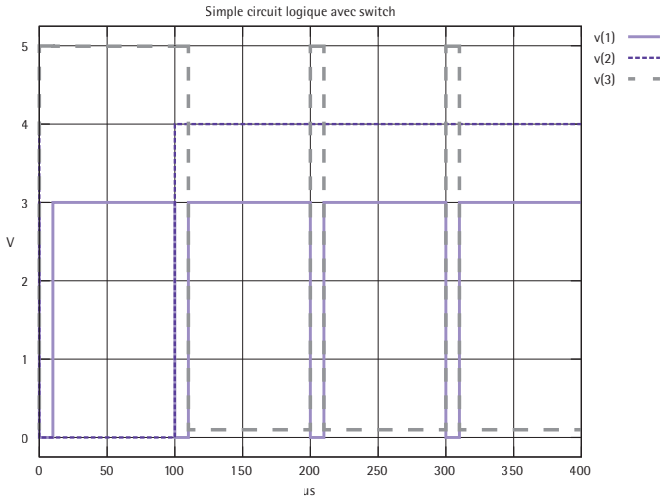
Un dernier circuit logique pour montrer que tout a été bien pensé dans ce simulateur:

```
Simple circuit logique avec switch
*
VCC 7 0 5V

* A AND B
VA 1 0 PULSE(3V 0V 0n 10ns 10ns 10us 100us)
VB 2 0 PULSE(4V 0V 0n 10ns 10ns 100us 400us)

XNAND1 1 2 3 7 NAND

.SUBCKT NAND 1 2 3 4
* A B OUT VCC
RL 3 4 1k
S1 3 5 1 0 switch
S2 5 0 2 0 switch
.ENDS

*      4      RL      3
* VCC +---/\/\/\---+--- OUT
*                  |
*      1           +
*    A +---------> \
*                   + 5
*    B +---------> \
*      2           |
*                 _|_

.MODEL switch sw vt=2.4 vh=0.2 ron=10 roff=1Meg
.TRAN 5us 400us
.PLOT TRAN V(1) V(2) V(3)
.END
```

On peut également faire des portes AND, NAND, OR, ET en s'inspirant du listing et du schéma ci-dessus.

```
ngspice logic.cir

ngspice 1 -> run
ngspice 2 -> plot v(1) v(2) v(3)
```

Les tensions de 3V, 4V, 5V sont volontairement différentes afin de bien les distinguer sur l'image suivante:



Il existe d'autres modèles pour la simulation digitale. De plus on peut également combiner l'analyse analogique/digitale, mais il faut déjà une bonne expérience de la simulation avant d'entrer dans ce domaine. Le lecteur trouvera probablement son bonheur sur la toile en recherchant des exemples précis selon ses besoins.

Conclusion

Comme vous le voyez, la puissance et les possibilités offertes par ce magnifique logiciel ne sont plus à démontrer. Pour aller plus loin, vous trouverez de nombreux exemples sur la toile, la base ayant été décrite dans le présent article.

Références

[1] VDHL *(VHSIC Hardware Description Language)*, **fr.wikipedia.org/wiki/VHDL**;

[2] Simulation informatique, **fr.wikipedia.org/wiki/Simulation_informatique**;

[3] Ngspice, **ngspice.sourceforge.net/download.html**;

[4] GeDA *(GPL'd suite and toolkit of Electronic Design Automation tools)*, **www.geda-project.org**;

[5] Lois de Kirchhoff, **fr.wikipedia.org/wiki/Lois_de_Kirchhoff**;

[6] Documentation Ngspice, **ngspice.sourceforge.net/docs.html**.