مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (38) العدد (4) 2016
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (38) No. (4) 2016

تقنية إضافة عناصر جديدة إلى قاعدة بيانات برنامج Multisim

فاتن فاروسي*

(تاريخ الإيداع 25 / 10 / 2015. قُبِل للنشر في 31/ 7 / 2016)

□ ملخّص □

يهدف البحث إلى دراسة كيفية إضافة عناصر جديدة إلى قاعدة بيانات برنامج Multisim أو كيفية نمذجة عنصر باستخدام اللغة البرمجية C++ من أجل استخدام هذا العنصر فيما بعد في تصميم وتركيب الدارات والأجهزة الالكترونية.

تحتوي قاعدة بيانات Multisim نماذج مدمجة لأغلب الأجهزة الالكترونية، وتهدف الدراسة إلى وضع أسس وطريقة لنمذجة العناصر الالكترونية غير الموجودة ضمن قاعدة بيانات برنامج Multisim (أو موجودة وبقيم مغايرة)، والتي نحتاجها أثناء استخدام هذا البرنامج في عملية النمذجة والمحاكاة لدارة ما.

وقد تم اقتراح أسلوب نمذجة الشيفرة Code modeling للوصول إلى هذا الهدف، ويعتمد هذا الأسلوب على سلوك الجهاز أو العنصر المنمذج.

وتبين الدراسة كيفية إنشاء نموذج شيفرة Code model لمكثف بقيم محددة ومغايرة للموجودة ضمن قاعدة البيانات وإضافته إليها.

الكلمات المفتاحية: نمذجة الشيفرة، الملف الطرفي، الملف التنفيذي.

* مشرفة على الأعمال – قسم الاتصالات–كلية الهندسة الميكانيكية و الكهربائية–جامعة تشرين– اللاذقية – سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (38) العدد (4) 2016
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (38) No. (4) 2016

Techniques to add new components to Multisim database

Faten Faroussi*

(Received 25 / 10 / 2015. Accepted 31 / 7 / 2016)

□ ABSTRACT □

The research aims to study how to add new components to Multisim database. Or how to model a component using the programing language C++ , to use this new component later in designing and making electronic circuits and devices.

Multisim has built-in models for most types of devices, , the study aims to lay the foundations and method for modeling of electronic items which is not located within the Multisim program database, (or present with different values) , and that we need while using this program in the modeling and simulation process for a given circuit.

Code modeling method has been proposed to reach this goal; this method relies on the behavior of the device or the modeled component.

The study shows how to create a Code model for a specific capacitor that has different values to those existing within the database and add to it.

Keywords: Code Modeling, Interface File, Implementation File.

*Work Supervisor, Department of Communication, Faculty of Mechanical & Electrical Engineering , Tishreen University, Lattakia, Syria.

مقدمة:

تطورت برامج النمذجة والمحاكاة في الآونة الأخيرة إلى درجة فائقة، وصممت الكثير من البرامج التي تقوم بعمليات النمذجة للعناصر والدارات الالكترونية. ويعد برنامج Electronic workbench من البرامج الأفضل عالميا في هذا المجال ومنه اخترنا برنامج Multisim الذي سيكون موضوع بحثنا [2].

يوضح هذا البحث كيفية نمذجة عنصر component باستخدام لغة برمجية عالية المستوى وقياسية صناعيا مثل لغة C [1]. وعند ذلك يمكن إضافة هذا العنصر الى قاعدة بيانات برنامج Multisim.

يوجد في برنامج Multisim نماذج مدمجة من أجل أغلب أنواع الأجهزة الالكترونية، قد يكون سلوك بعض الأجهزة صعب النمذجة للغاية كما في مجموعات من عناصر برنامج SPICE [6]، ولكن قد يكون من الأسهل وصفها بدلالة معادلات سلوكية عالية المستوى، ونتيجة لذلك يمكن نمذجة سلوك هذه الأجهزة باستخدام نمذجة الشيفرة.

أهمية البحث وأهدافه:

تكمن أهمية هذا البحث في وضع أسس وطريقة لنمذجة العناصر الالكترونية الغير موجودة في قاعدة بيانات برنامج Multisim، والتي نحتاجها أثناء العمل في البرنامج لمحاكاة دارة ما. وذلك عن طريق المقدرة على نمذجتها وإضافتها إلى قاعدة بيانات البرنامج، وحتى لو كان هذا العنصر موجوداً يمكن إضافة إمكانيات جديدة تسمح باستخدامه في أماكن أخرى.

طرائق البحث ومواده:

قمنا في هذا البحث باتباع الخطوات التالية:

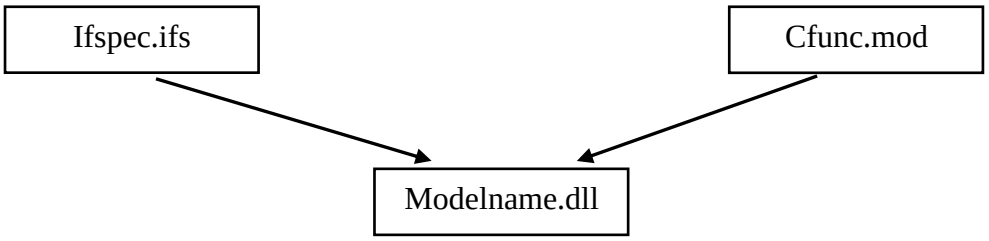
1. تعريف نمذجة الشيفرة ووضع الخطوات الأساسية لإنشاء نموذج شيفرة.
2. وضع الخطوات الأساسية لإنشاء نموذج شيفرة.
3. دراسة الملف الطرفي والتعريف بالجداول المستخدمة فيه مع وضع أمثلة تطبيقية عند إنشاء نموذج شيفرة.
4. دراسة الملف التنفيذي والتعريف بواحدات الماكرو لهذا الملف.
5. وضع النتائج والمناقشة ودراسة إنشاء نموذج شيفرة لمكثف وإضافته إلى قاعدة بيانات البرنامج.

حتى نستطيع القيام بنمذجة الشيفرة يجب أن يكون لدينا مترجم C مثل (Microsoft Visual C++) [4]، النسخة 4.1 أو نسخة أحدث منها وهي النسخة التي سنعمل عليها حتى نقوم بنمذجة عنصر وإضافته إلى قاعدة بيانات البرنامج Multisim.

1-1 تعريف نمذجة الشيفرة ووضع الخطوات الأساسية لإنشاء نموذج شيفرة:

نمذجة الشيفرة هي النمذجة السلوكية لأجهزة تتحكم بمعادلات معروفة، حيث يتألف نموذج الشيفرة من مجموعة من تعاريف الطرفيات ومن تطبيق التابع C الذي يصف سلوك الجهاز، إن تسمية وتموضع هذه الملفات مهم.

يتم إنشاء النموذج عن طريق دمج أو اتحاد ملفين هما (Ifspec.ifs) و (Cfunc.mod)، ويتوضع الملف الناتج عن عملية الدمج، والذي يعطى نفس الاسم كمجلد يحتوي ملفات المصدر في المجلد codemod1 كما يظهر في الشكل (1) [5]:



الشكل(1) يبيّن المخطط الصندوقي لإنشاء نموذج الشيفرة.

1-2 وضع الخطوات الأساسية لإنشاء نموذج شيفرة:

من أجل إنشاء نموذج شيفرة سنتبع الخطوات التالية وللحصول على أفضل النتائج سنستخدم في بحثنا هذا نافذة أوامر DOS، حيث سنقوم أولاً بتجهيز بيئة الحاسب من أجل (Microsoft Visual C++) عن طريق تشغيل VcVars32.bat، ومن ثم سننشئ مكتبة في نفس المكتبة التي يكون فيها برنامج Multisim قابلا للتنفيذ وتدعى codemod1 ويجب أن يوجد ضمن هذه المكتبة مكتبة فرعية وحيدة تدعى USER. كما يجب أن ننشئ مكتبات فرعية جديدة داخل المكتبة USER، واحدة من أجل كل نموذج شيفرة نريد إيجاده. وسيكون في النهاية اسم المكتبة الفرعية هو اسم نموذج الشيفرة القابل للتنفيذ، وبعدها نقوم بإنشاء ملف طرفي جديد داخل المكتبة الفرعية للنموذج المدروس يدعى Ifspec.ifs لإعداد التعريفات والطرفيات الأساسية (دخل/ خرج المنافذ والبارامترات)، ثم نقوم بإنشاء ملف تنفيذي جديد داخل المكتبة الفرعية للنموذج يدعى Cfunc.mod ويحتوي هذا الملف نموذج الشيفرة الفعلي، و يتضمن الملف Cfunc.mod بدقة قائمة لكل النماذج في الملف على الترتيب التالي:

```
SPICEdev * FAR DynDEVices [ ] = {
&<function_name>_ info
} ;
```

حيث أن: <function_name> هي اسم التابع C "C_Function_Name" المعرف في ملف Ifspec.ifs الخاص بك. وهذه المتطلبات خاصة ببرنامج Multisim، ولذلك سيكون هناك حاجة لإضافة هذا السطر إلى نماذج الشيفرة المستخدمة من مصادر أخرى.

ومن أجل ترجمة الملفات إلى شكل يمكن استخدامه بواسطة Multisim ، نذهب إلى المجلد codemod1\USER وننفذ الأمر التالي : MAKE_DLL < subdirectory > حيث أن < subdirectory >هي المكتبة التي وضعت فيها ملفات نموذج الشيفرة الخاصة بنا.

وإذا تمت عملية الترجمة بنجاح، سوف نحصل على ملف جديد امتداده dll. داخل المكتبة الفرعية وهذا الملف هو نموذج الشيفرة القابل للتنفيذ. يجب أن نقوم بعملية نسخ أو نقل لهذا الملف ذو الامتداد dll. إلى المكتبة الرئيسية codemod1 حتى يكون باستطاعة برنامج Multisim إضافته عند بداية التشغيل.

ومن أجل استخدام نموذج الشيفرة الخاص بنا في Multisim، سنختار نموذجاً بالطريقة المعتادة في المعالج التلقائي component wizard ثم نختار تعليمة "حمّل من الملف " . ثم نضع نوع الملف في مربع الحوار "Code Model DLL (*.dll)" ونستعرض مكتبة codemod1 إذا كان ذلك ضرورياً. ثم نختار الملف ذو الامتداد dll. الذي أنشأناه للتو.

اذا لم تعدل بنية المكتبة codemod1 ، فإنّ اسم النموذج و اسم نموذج spice سوف ينشر بشكل أوتوماتيكي من ملف ifspec.ifs . يجب أن تبدو الصيغة كما يلي:

.MODEL <DLL> <name> (<params>)

حيث أنّ <DLL> هي امتداد اسم dll. لنموذجنا (بدون الامتداد dll.)، <name> هي اسم نموذج "Spice_Model_Name" من ملفنا ifspec.ifs ، <params> هي قائمة إختيارية للشكل :

"<param_name> = <default value>" مفصولة عن بعضها بفراغات (وليس بفواصل) وتفيد لإبطال أو لإلغاء قيمة أي وحدة قياس أصلية معرفة في ifspec.ifs. نحذف الخط "_INSERT_PARAMS" ونطبع البارامترات، أو نترك علامتي الحصر <> فارغة إذا أردنا استخدام كل الإعدادات الأصلية.

1-3 دراسة الملف الطرفي (ifspec.ifs)، والتعريف بالجداول المستخدمة فيه مع وضع أمثلة تطبيقية عند إنشاء نموذج شيفرة:

إن الملف الطرفي يتوضع في جداول تتضمن الأسماء المستخدمة من قبل النموذج، الوصلات الكهربائية في الجهاز (المنافذ)، والمتغيرات (البارامترات) المعرفة للمستخدم التي تؤمن تحكم أفضل بسلوك النموذج. وهذه الجداول مشروحة في هذه الفقرة، مع أمثلة معطاة لكل منها. يحتاج الملف الطرفي مع ملف التنفيذ إلى أن يترجما إلى DLLمن أجل إتمام نموذج الشيفرة.

1-3-1 جدول الاسم Name Table : إنّ كلاً من اسم النموذج، نص المواصفات، واسم تابع التطبيق C، معرف في جدول الاسم. يجب أن يكون اسم النموذج هو نفسه كما في المكتبة الفرعية التي تحتوي ملفات نموذج الشيفرة. كما يوصى بأن يكون اسم النموذج مؤلفاً من ثمانية محارف، ويتكون جدول الاسم من البنية أو التركيب المبين في الجدول(1)[3] :

جدول(1): يبين بنية جدول الاسم.

```
NAME_TABLE :
C_Function_Name : function_name
Spice_Mode1Name : model_name
Description :     " text "
```

حيث أنّ :

- **اسم التابع function_name**: هو معرّف C المتاح وهو اسم نقطة الدخول الرئيسية (تابع) إلى نموذج الشيفرة. قد يكون مطابقاً لاسم النموذج SPICE أو أن لا يكون. ولتقليل من فرص التباسات الاسم، نوصي باستخدام prefix (اللاحقة العليا) " _UCM " نموذج شيفرة المستخدم، أو استخدام لاحقة عليا بأحرف استدلالية خاصة بنا. إن اللاحقات العليا الواردة في الجدول (2) تستخدم من قبل نواة المحاكي XSPICE ويجب عدم استخدامها في نماذج شيفرة المستخدم:

جدول(2): يبين اللاحقات العليا التي لا يمكن استخدامها في نماذج شيفرة المستخدم.

A2VERI	D_OPEN_C	8EW_RES	PLOY
A2VHDL	D_OPEN_E	EW_SCR	POT
A_555	D_OR	EW_SWTCH	PPT
ADC_BRDG	D_OSC	EW_VLT	PWL
ASRC	D_PULLDN	FTE	R_2_V
ASWITCH	D_PULLUP	GAIN	RDEDELAY
BJT	D_RAM	HLP	RES
BSIM	D_ROM	HYTC	RGAIN
CAP	D_SERIALPORT	ICM	S_XFER
CCCS	D_SOURCE	IDN	SCR
CCVS	D_SRFF	ILIMIT	SINE
CKT	D_SRLATC	IND	SLEW
CLIMIT	D_STATE	INDUCTOR	SNP
CM	D_TFF	INP	SQUARE
CMETER	D_TRISTA	INT	SRC_LVM
COR	D_VERI	IPC	SRC_LVM_CUR
CP	D_VERILOG	ISRC	SRC_TDM
CSW	D_VHDL	GFET	SRC_TDM_CUR
D_2_R	D_WGEN	LCOUPLE	SUMMER
D_AND	D_XNOR	LIMIT	SW
D_BUFFER	D_XOR	LMETER	TRA
D_SHIP	DAC_BRDG	MES	TRIANGLE
D_CPU	DAC_HIZ	MFB	URC
D_CPUPIC	DEV	MIF	VCCS
D_DFF	DIO	MOS1	VCVS
D_DLATCH	DIVIDE	MOS2	VERIZA
D_DT	ENH	MOS3	VHDL2A
D_FDIV	EVT	MULT	VHDL2HIZ
D_INV	EW_CAP	N1	VSRC
D_GKFF	EW_IND	NCO	XCAP
D_NAND	EW_NLA2D	NOISE	ZENER
D_NOR	EW_NOISE	ONESHOT	

- **اسم النموذج model_name:** هو معرف SPICE المتاح الذي سيستخدم على سطح SPICE. يسجل النموذج بحيث يشير إلى نموذج الشيفرة هذا. ويمكن أن يكون مطابقاً لاسم التابع C.
- **النص TEXT:** هو السلسلة التي تصف الهدف والوظيفة لنموذج الشيفرة.

مثال على جدول الاسم:

NAME_TABLE:

Spice_Model_Name : capacitor

C_Function_Name : cm_capacitor

Description : " Capacitor with voltage initial condition "

1-3-2 جدول المنفذ Port Table : تكون منافذ الجهاز معرفة في جداول المنفذ، ولجدول المنفذ البناء المبين في الجدول رقم (3) [3].

جدول(3): يبين بنية جدول المنفذ.

Port_Name : name Description : text Default_Type : default Allowed_Type : [type typetype] Vector : vector Vector_Bounds : size Direction : dataflow Null_Allowed : null

حيث أن:

- **الاسم name:** هو معرف SPICE متاح والذي يعطي اسم المنفذ.
- **النص text:** هو السلسلة التي تصف هدف ووظيفة المنفذ.
- **النمط الأساسي default:** يحدد الشكل المستخدم للمنفذ عند عدم وجود نمط محدد بوضوح. ويجب أن يكون واحداً من العناصر المجدولة في " type ".
- **النمط type:** يضع قائمة بالأنماط المسموح بها والتي يمكن للمنفذ أن يتصل معها. مع أسماء مفصولة عن بعضها بفواصل أو فراغات (مثال [d, g, h]).

جدول (4): يبين الأنماط التي يمكن للمنفذ أن يتصل معها.

اسم النمط	الاتجاهات المتاحة	الوصف
D	دخل ، خرج	رقمي
G	دخل ، خرج	ناقلية (دخل كمون، خرج تيار)
gd	دخل ، خرج	ناقلية تفاضلية (دخل كمون، خرج تيار)
h	دخل ، خرج	مقاومة (دخل تيار ، خرج كمون)
hd	دخل ، خرج	مقاومة تفاضلية (دخل تيار ، خرج كمون)
i	دخل ، خرج	تيار
id	دخل ، خرج	تيار تفاضلي
v	دخل ، خرج	كمون
vd	دخل ، خرج	كمون تفاضلي
vnam	دخل	تيار عبر مصدر كمون ذو تسمية

- **المتّجه (vector):** يحدد ما إذا كان المنفذ هو متجه أو لا، ويمكن اعتباره حزمة (bus).

▪ **الحجم:** من أجل المنفذ الذي يكون متجهاً فقط، يجب تحدد الروابط الأعلى والأخفض على حجم المتجه، فالرابط الأدنى يحدد العدد الأقل من العناصر، والرابط الأعلى يحدد العدد الأكبر من العناصر ومن أجل مجال مفتوح أو منافذ ليست متجهات، نستخدم الرمز ("-").

▪ **تدفق المعطيات dataflow:** يحدد اتجاه تدفق المعطيات عبر منفذ، ويمكن أن يكون إما خرج out، أو دخل in، أو دخل خرج inout.

▪ **الصفر null:** يحدد فيما إذا كان من الممكن ترك المنفذ متصلاً أو لا.

مثال على جدول المنفذ:

PORT_TABLE
Port_Name : cap
Description : " capacitor terminals "
Default_Type : hd
Allowed_Types : [hd]
Vector : no
Vector_Bounds : -
Null_Allowed : no

1-3-3 جدول البارامترات Parameter Table: تعرف بارامترات الجهاز في جداول البارامترات. ويكون لجدول البارامترات البناء المبين في الجدول رقم (5) [4] .

جدول(5): يبين بنية جدول البارامتر.

PARAMETER_TABLE :
Parameter_Name : name
Description : text
Data_Type : type
Vector : vector
Vector_Bounds : size
Default_Value : default
Limits : range
Null_allowed : null

حيث أن:

- **الاسم name:** هو معرف SPICE المتاح و الذي سيستخدم على سطح SPICE ليشير إلى هذا البارامتر.
- **النص text:** هو سلسلة تشرح هدف ووظيفة البارامتر.
- **النمط type:** هو شكل معطيات البارامتر. الذي يتطابق مع شكل معطيات C (مثال "double"). وليس الشكل المفاهيمي للبارامتر (مثال "voltage") . نختار إحدى الحالات التالية:

*Boolean: (إذا كان شكل معطيات C هي "Boolean_t" مع قيم متاحة MIF_TRUE وMIF_FALSE).

*complex: (إذا كان شكل معطيات C هي "Complex_t" مع أعضاء مضاعفة حقيقية وتخيلية).

*int: (إذا كان شكل معطيات C هي "int").

*real: (إذا كان شكل معطيات C هي "double").

*string: (إذا كان شكل معطيات C هي "*char").

*pointer: (إذا كان شكل معطيات C هي "*void").

- **المتجه vector:** يحدد ما إذا كان العنصر هو متجه أو كمية مقاسة (scalar). نختار إحدى الحالتين:
- **الحجم size:** يؤخذ للبارامترات المتجهة فقط، يحدد الروابط الأعلى والأدنى على حجم المتجه. الرابط الأدنى يحدد العدد الأقل من العناصر، الرابط الأعلى يحدد العدد الأكبر من العناصر. من أجل مجال مفتوح أو بارامترات ليست متجهة استخدم الرمز ("-"). وبشكل بديل، نحدد اسم المنفذ الذي حجم متجهه سيستخدم لهذا البارامتر.
- **النمط الأصلي default:** إذا كانت سماحية الإلغاء Null Allowed هي "نعم" فإنّ قيمة أصلية ستستخدم على سطح SPICE. إن خط النموذج لا يزود البارامتر بقيمة. وإنما يجب أن تستند القيمة إلى قيم المعطيات Data_type (عددية، بوليانية، مركبة، أو سلسلة حرفية).
- **المجال range:** هو مجال محدود من القيم (من أجل أنماط البارامترات"int" و"real" فقط).
- **الصفر null:** يحدد ما إذا كان مسموحاً للبارامتر أن يكون صفر أو لا يكون. نختار إحدى الحالتين:

*نعم – سطح SPICE المتطابق، بطاقة النموذج يمكن أن تحذف قيمة لهذا البارامتر، وعندها ستستخدم قيمة النمط الأصلي، أو إذا لم يكن هناك قيمة للنمط الأصلي فإنّ قيمة غير معرفة ستمرّر إلى نموذج الشيفرة.

*لا – يجب أن يكون لهذا البارامتر قيمة. سيعطي XSPICE تنبيه بوجود خطأ إذا كان سطح SPICE متطابق، بطاقة النموذج تحذف قيمة لهذا البارامتر.

مثال على جدول البارامتر:

PARAMETER_TABLE :		
Parameter_Name :	c	ic
Description :	"capacitance"	"voltage initial condition"
Data_Type :	real	real
Default_Value :	-	0.0
Limits :	-	-
Vector :	no	no
Vector_Bounds :	-	-
Null_Allowed :	no	no

4-1 دراسة الملف التنفيذي (Cfunc.mod)، والتعريف بواحدات الماكرو لهذا الملف:

في كل عملية محاكاة لدارة تستخدم نموذج الشيفرة، فإن محرك XSpice المحاكي لبرنامج Multisim يستدعي ملف التنفيذ، ونحتاج إلى دمج ملف التنفيذ مع الملف الطرفي ليتحول إلى DLL من أجل إتمام نموذج الشيفرة. وعندها فإن نموذج الشيفرة سيولد خرج جهاز نموذج الشيفرة المنمذج. وهذا الخرج يستند الى مايلي:

- الدخل الذي يقدمه XSpice إلى تابع نموذج الشيفرة [5].
- حالة النموذج، التي يخزنها ويعيدها XSpice .

يحتوي ملف التنفيذ واحد أو أكثر من وحدات الماكرو (الموضحة في الفقرة 1-4-1) والذي يزود ببرمجة التطبيقات الطرفية API بين XSpice ونموذج الشيفرة.

الفقرة التالية تعطي قائمة بوحدات الماكرو التي يمكننا اختيارها.

يجب ترجمة ملف التنفيذ مع الملف الطرفي إلى DLL لإتمام نموذج الشيفرة.

1-4-1 وحدات ماكرو ملف التنفيذ C:

سنستعرض في هذه الفقرة بعض وحدات ماكرو التي قد يحتوي ملف التنفيذ واحداً منها أو أكثر [5]:

AC_GAIN (outputname ,inputname)

النمط	Complex_t
المتحولات	y[i], x[i]
مطبق على	نماذج الشيفرة التشابهية فقط (المقادة بالحدث أو نموذجات الشيفرة الرقمية يجب ألا تقوم بأي شيء خلال تحليلات AC).
الوصف	يحدد قيمة لوحدة الماكرو هذه من أجل تحديد الربح من outputname إلى inputname عند تردد التيار. يدعى تابع نموذج الشيفرة مرة من اجل كل نقطة تردد تجري محاكاتها.

التحليلات ANALYSIS

النمط	Enum
المتحولات	لايوجد
مطبق على	كل نماذج الشيفرة ،مادام تصرفها يتغير بشكل نموذجي معتمدا على شكل التحليل المقدم، وهذا الماكرو يمكن استخدامه لتحديد وحدات خرج ماكرو مناسبة.
الوصف	يعيد صنف التحليلات التي قدمت: MIF_AC من أجل DC MIF_DC من أجل نقطة تشغيل DC MID_TRAN من أجل مؤقت transient

المتحولات ARGS

النمط	MIF_Private_t
المتحولات	لايوجد
مطبق على	كل نماذج الشيفرة
الوصف	يجب تقديم قائمة بارامتر نموذج الشيفرة، وعدم تعديلها.

الدخل INPUT (inputname)

النمط	Double or void*
المتحولات	name [i]

مطبق على	نماذج شيفرة تشابهية / نمط مختلط.
الوصف	فقط المداخل التشابهية هي المسموحة (من أجل المقادة بالحدث، استخدم INPUT_STATE و INPUT_STRENGTH). تعيد القيمة على العقدة أو المجموعة المتصلة الى inputname. يحدد النمط/ الوحدات لقيمة الدخل عندما يكون نمط الدخل محددا في الملف ifspec.ifs.

OUTPUT (outputname)

النمط	Double or void*
المتحولات	name [i]
مطبق على	نماذج شيفرة تشابهية / نمط مختلط.
الوصف	مسموح فقط للمخارج التشابهية (من أجل المقادة بالحدث استخدم OUTPUT_STATE و OUTPUT_STRENGTH و OUTPUT_DELAY). عيّن قيمة للعقدة أو حزمة متصلة إلى outputname . يحدّد النمط / الوحدات لقيم الخرج عندما يكون نمط الخرج محدداً في ملف ifspec.ifs.

PARTIAL

النمط	Double
المتحولات	y[i], x[i]
مطبق على	نماذج الشيفرة التشابهية / النمط المختلط
الوصف	لفظة مشتقة جزئية للخرج y مع اعتبار الدخل x.

PORT_NULL

النمط	Boolean_t
المتحولات	Name [i]
مطبق على	أي نموذج شيفرة
الوصف	هل تحدد هذا المنفذ كغير متصل؟

RAD_FREQ

النمط	Double
المتحولات	<none>
مطبق على	نماذج الشيفرة التشابهية / النمط المختلط
الوصف	التيار يحلل التردد ب راديان/ ثانية.

النتائج والمناقشة:

وكتطبيق على الدراسة المقدمة فقد تم إيجاد نموذج شيفرة لمكثف مع الحالات الإبتدائية للكمون، وتم تضمينه ضمن ملفين الأول هو الملف الطرفي (ifspec.ifs)، والثاني هو ملف التنفيذ (cfunc.mod).

الملف الطرفي (ifspec.ifs):

الغاية منه إعداد التعريفات والطرفيات الأساسية (دخل وخرج المنافذ، البارامترات)، ويحتوي هذا الملف التعريف لنموذج شيفرة مكثف مع حالات أحرف ابتدائية لنمط الكمون.

```
NAME_TABLE:
Spise_Model_Name :      capacitor
C_Function_Name :       cm_capacitor
Description :           "Capacitor with voltage initial condition"

PORT_TABLE :
Port_Name :             cap
Description :           "capacitor terminals"
Direction :             inout
Default_Type :          hd
Allowed_Types :         [hd]
Vector :                no
Vector_Bounds :         -
Null_Allowed :          no

PARAMETER_TABLE
Parameter_Name :    c               ic
Description :       "capacitance"   "voltage initial condition"
Data_Type :         real            real
Default_Value :     -               0.0
Limits :            -               -
Vector :            no              no
Vector_bounds :     -               -
Null_Allowed :      no              no
```

ملف التنفيذ (cfunc.mod):

يوضح ملف التنفيذ كيفية تنفيذ وحدة ماكرو، ويحتوي هذا الملف التعريف لنموذج شيفرة مكثف مع الحالات البدائية

لنمط الكمون.

```
#define VC   0
Void cm_capacitor (ARGS)
{
Complex_t       ac_gain ;
    double          partial  ;
    double          ramp_factor ;
    double          *vc ;
/*Get the ramp factor from the .option ramptime */
ramp_factor = cm_analog_ramp_factor (MIF_INSTANCE) ;
```

```
/*Intialize/acces instance specific storage for capacitor volt age */
If (INIT) {
      cm_analog_alloc (MIF_INSTANCE,VC,  sizeof (double) ) ;
      vc = cm_analog_get_ptr (MIF_INSTANCE,VC,  0) ;
     *vc = PARAM (ic)  * cm_analog_ramp_factor (MIF_INSTANCE) ;
}
Else  {
       vc = cm_analog_get_ptr (MIF_INSTANCE,VC,  0) ;
}
/* compute the output */
if (ANALYSIS == DC)  {
      OUTPUT (cap) = PARAM (ic)  * ramp_factor ;
      PARTIAL (cap, cap)  = 0 . 0 ;
}
else if (ANALYSIS ==AC)  {
ac_gain.real = 0 . 0 ;
     ac_gain.imag = -1 . 0 / RAD_FREQ / PARAM (c) ;
     AC_GAIN (cap, cap) = ac_gain ;
}
else if (ANALYSIS == TRANSIENT)  {
     if(ramp_factor< 1.0)  {
         *vc  = PARAM (ic) * ramp_factor ;
          OUTPUT (cap)  =  *vc ;
          PARTIAL (cap,  cap) =0.0 ;
       }
       else  {
             cm_analog_integrate (MIF_INSTANCE.INPUT(cap)  /  PARAM (c) ,
             vc, &partial) ;
            partial /= PARAM (c) ;
           OUTPUT (cap)  =  *vc ;
           PARTIAL (cap,  cap)  = partial;
       }
    }
}
```

ولإتمام نموذج الشيفرة يجب ترجمة (دمج) ملف التنفيذ مع الملف الطرفي إلى DLL. وعندها فإن نموذج الشيفرة سيولد خرج جهاز نموذج الشيفرة المنمذج، وهذا الخرج يستند الى ما يلي:

الدخل الذي يقدمه XSpice إلى تابع نموذج الشيفرة، وحالة النموذج التي يخزنها XSpice ويعيدها.

إن إتمام نموذج الشيفرة يكون بإتباع الخطوات التالية:

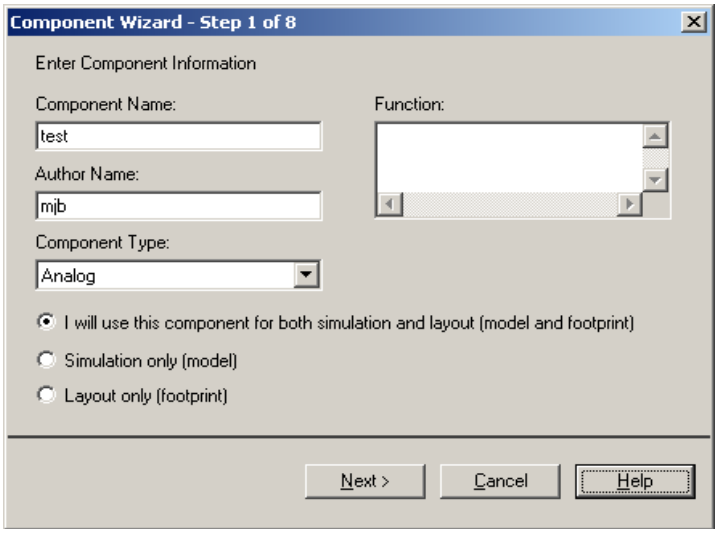
1. لترجمة الملفات إلى شكل يمكن استخدامه بواسطة برنامج Multisim، نذهب إلى مجلد Codemod1/USER وننفذ الأمر التالي: MAKE_DLL CAPACITOR. حيث أن CAPACITOR هو اسم المكتبة الفرعية التي أنشأناها.

2. بعد إتمام العملية بنجاح سنحصل على ملفين جديدين في مكتبتنا هما: Capacitor.c و Capacitor.dll . يحتوي الملف: Capacitor.c كل المعلومات من الملفين ifspec.ifs و Cfunc.mod متمددة إلى صيغة مفهومة من قبل المترجم C وتطبيق Multisim. وهذه الصيغة هي Xspice. أما الملف Capacitor.dll فيحتوي نموذج الشيفرة المترجم الجاهز للتنفيذ.

3. ننقل الملف Capacitor.dll إلى مكتبة Codmod1 الرئيسية حتى يتمكن Multisim من إضافته.

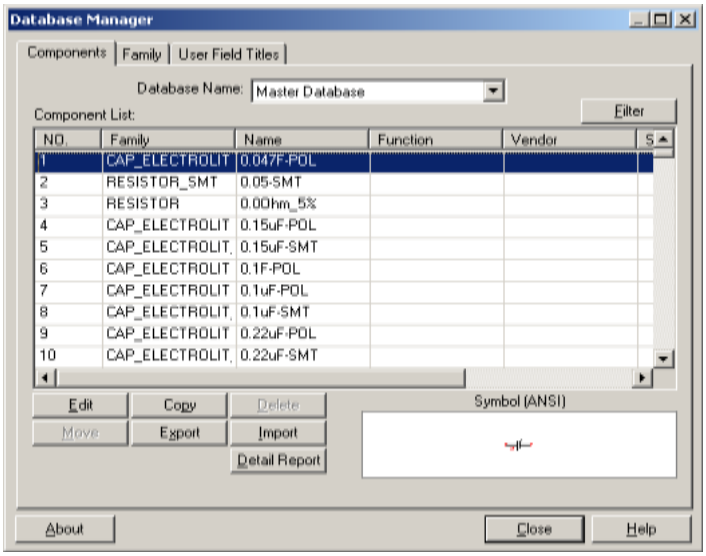
4. إذا كان برنامج Multisim قيد التشغيل، سنقوم بإطفائه ثم إعادة تشغيله كي يتمكن من تحميل النموذج الجديد. لأن Multisim يلاحظ نماذج الشيفرة الجديدة فقط عند الإقلاع.

5. لإنشاء العنصر التشابهي، نشغل المعالج التلقائي component wizard الذي سيقود خطوات إنشاء العنصر. ونختار نموذجاً بالطريقة المعتادة فيه (اسم العنصر، صنف العنصر، والغاية من استخدامه) كما في الشكل (2):



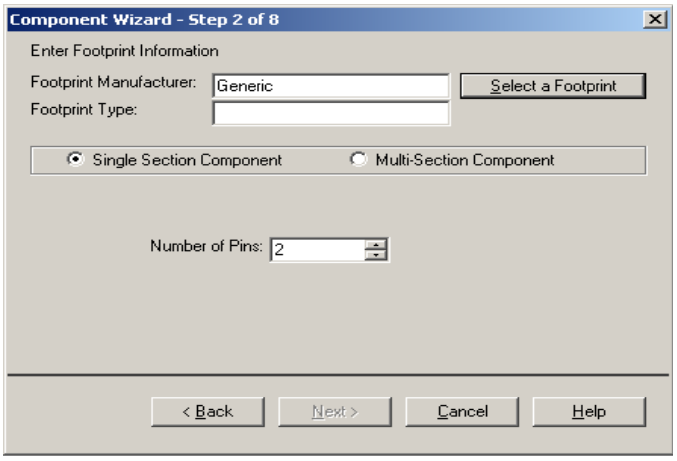
الشكل (2): يبين الخطوة الأولى لإنشاء العنصر.

– ولإنشاء عنصر تشابهي محاكي للمكثف (ننقر " Select Simulation Model "). ثم نختار رمز العنصر وقيمته كما في الشكل (3):



الشكل (3): يبين خطوة اختيار العنصر الذي نريد إنشاؤه (المكثف).

– ثم نعطي للعنصر طرفين: واحد للدخل والآخر للخرج. كما في الشكل (4):



الشكل (4): يبين خطوة تحديد عدد أطراف العنصر.

– ثم ننقر " Load from file ".

– في حالة اختيار ملف الحوار الذي يظهر، نغير المرشح إلى: "CodeMod1DLL(.*dll)" ونستعرض المكتبة CodMod1 إذا كان ضروريا.

– ثم نختار الملف الذي أنشأناه Capacitor.dll وننقر "Open".

–يجب تغيير اسم النموذج الى "CAPACITOR" ومعطيات النموذج إلى: "MODEL CAPACITOR capacitor (_INSERT_PARAMS)". يمكننا حذف _INSERT_PARAMS ووضع البارامترات المطلوبة أو ترك ما بين القوسين فارغا إذا أردنا استخدام الإعدادات الأصلية.

– في فقرة خارطة أطراف النموذج، نغير نمط نموذج "SPISE Model Type" إلى "XSPICE model(a)".

–أخيراً نضع النموذج في مكان منطقي في قاعدة بيانات المستخدم الخاصة بنا. ونختار العائلة التي ينتمي اليها. في حالة العنصر الذي أنشأناه العائلة هي Basic. كما في الشكل (5):

الشكل (5): يبين خطوة اختيار العائلة المناسبة للعنصر الجديد.

الاستنتاجات والتوصيات:

إن برنامج Multisim يمنحنا القدرة على إضافة عناصر الكترونية إلى تلك المتوفرة في قاعدة البيانات الرئيسية، وهذا يفيد عندما نحتاج إلى ملاءمة عناصر في احتياجات خاصة. قد يكون العنصر موجوداً ولكننا بحاجة إلى قيمة معينة غير متوفرة في قاعدة البيانات. وبذلك فإن هذه الطريقة التي اتبعناها توفر لنا احتياجنا لتصميم الدارة المنشودة.

إذاً فإن ما يميز طريقة النمذجة هذه هو أنه يتم إضافة إمكانية على العناصر الموجودة في قاعدة البيانات تسمح بجميع الاحتمالات الممكنة لهذه العناصر.

المراجع

1. د. عمران سليمان، علي. *مدخل الى الحاسب والخوارزميات*. منشورات جامعة تشرين، 2005.
2. د. صالح، السموءل؛ الكنا، ميساء؛ شحود، رندة؛ عبدو، آمنة؛ مقصود، سميرة. *أسس الهندسة الإلكترونية القسم العملي*. منشورات جامعة تشرين 2011.
3. *C++ الدليل الكامل*. ترجمة مركز التعريب والبرمجة. الدار العربية للعلوم، الطبعة الأولى، بيروت.
4. DEITEL, H.M; DEITEL, P.J. *C++ How to Program.* 5th Edition, Prentice Hall, New Jersey.
5. LOPEZ, D.B; CASTRO, F.G; VILLAGOMEZ, O.G. *Advanced Circuit Simulation Using Multisim Workbench.* Morgan &Claypool Publishers, 2012.
6. LUTHER, E; RODRIGUES, J. *Introduction to Multisim Schematic Capture and SPICE Simulation.* Rise University, Houston, Texas, 2006.
7. STROUSTRUP, B. *The C++ Programming Language*, 4th Edition, Addison-Wesley, 2013.
8. GILSTER, R. *PC Technician Black Book.* Coriolis Group, Arizona, USA, 2001.

9. GILSTER, R. *PC Interfacing Black Book.* Coriolis Group, Arizona, USA, 2002.

10. LEESTMA; SANFORD; NYHOFF, L. *Advanced Programming in Pascal with Data Structures.* Macmillan Publishing Company, New York, Collier Macmillan Publishers, London, Republic of Singapore, 1988.

11. DEITEL, H.M; DEITEL, P.J. *C++ How to Program.* Prentice Hall, New Jersey, Fifth Edition.

12. NIEMANN; THOMAS. *Sorting and Searching Algorithms.* Epaper Press.

13. LOPEZ, D.B; CASTRO, F.G. *Circuit Analysis with Multisim.* Morgan &Claypool Publishers, 1ST Edition.

14. NILSSON, J.W; RIEDEL, S. *Introduction to Multisim for Electric Circuits.* 10th Edition.

15. REEDER, J. *Troubleshooting DC/AC Circuits.* Tomson publishers. 4th Edition.