منهج لتوليد تعليمات برمجية بصفة آلية للأنظمة المطمورة بالاعتماد على محلل لغوي

آمنة القلال [1]، ياسين العودني[1]، محمد عبيد[1]
[1] العنوان البريدي: جامعة صفاقس، المدرسة الوطنية للمهندسين بصفا قس، ص.ب. 1173 ، الرمز البريدي 3038 ، صفاقس، تونس

البريد الالكتروني:

kallelemna@yahoo.fr yassine.aoudni@gmail.com mohamed.abid@enis.rnu.tn

الخلاصة. في هذا البحث، نبين كيف أن منهج معتمد على محلل لغوي قادر على تسهيل و تسريع مهمة مصممين الأنظمة المطمورة . حيث أن المحلل اللغوي للغة سي الذي قمنا بتصميمه يستنتج خاصيات رئيسية اثنين: أولا، تعقيد نص البرنامج من حيث العمليات الجبرية و ثانيا، موقع المعطيات المتصلة بكل عملية في نص البرنامج المصدر. ثم بعد ذلك سيتم استخدام التلقائية لتوليد التعليمات البرمجية المادية للعمليات الجبرية. لقد قمنا ببرمجة برنامج توليد التعليمات المخصصة سي اي جي تي (CAGT) الذي يقوم بإدماج كل من المحلل اللغوي للغة سي و المحلل اللغوي للغة في هاش دي ال (VHDL) التي قمنا بتصميمها لغرض التوليد الآلي للتعليمات البرمجية للأجهزة اللينة / الصلبة . الدراسة التجريبية المنجزة قد ساعدتنا على إظهار فعالية و سرعة البرنامج سي اي جي تي التي يمكن أن تصل إلى 90% مقارنة بتصميم يدوي.

الكلمات الجوهرية. محلل لغوي، الأنظمة المطمورة، سي اي جي تي ، التوليد الآلي للتعليمات البرمجية.

1 مقدمة

تستخدم الأنظمة المطمورة في عدة مجالات: الاتصالات, التحكم, التتبع, المراقبة عن بعد...الخ. الارتفاع المتواصل في تعقيد الخوارزميات يجعل من هذه التطبيقات مكلفة من حيث الوقت والتصميم. لقد ولد مفهوم التصميم المزدوج للبرمجيات و العتاد الصلب مع الضرورة المعاصرة للحد من الفجوة بين إنتاج الدوائر المتكاملة و إنتاجية التطبيقات. ومع ذلك فان الأنظمة المطمورة لا تزال تشكل صعوبات في الاستخدام لشخص غير خبير في مجال الإلكترونيات.

من هنا تأتي الحاجة إلى استحداث منهجيات برمجية جديدة تسهل تصميم وإدارة هذه النظم استناداً إلى "زيادة مستوى التجريد" لغاية التخفيف من تعقيد البرمجيات. يمثل التوليد الآلي للتعليمات البرمجية أحد الحلول التي ظهرت مؤخرا للاستجابة لهذه الأنواع من الاحتياجات في مجال تطوير البرمجيات.

لقد تواجدت العديد من برمجيات التوليد الآلي لبرامج الأنظمة المطمورة[2,1]. ومع ذلك، فهي تتطلب من مستعمليها مجهودا كبيرا في صياغة المواصفات التوظيفية للنظام. في هذا البحث، سوف نبين كيف أن منهج معتمد على محلل لغوي قادر على تسهيل و تسريع مهمة مصممين الأنظمة المطمورة.

تتضمن هذه الوثيقة ثلاثة محاور رئيسية هم:

المحور الأول :سنقدم فيه دراسة حول بعض الأعمال السابقة و الحالية في مجال التوليد الآلي للتعليمات البرمجية.

المحور الثاني: سنقدم فيه بعض المقترحات العملية لإعداد المشروع سي اي جي تي [3] حيث سنقوم بتعريف المحلل اللغوي للغة سي الذي قمنا بتصميمه لغرض التوليد الآلي للتعليمات البرمجية للأجهزة اللينة / الصلبة.

المحور الثالث: : سنعرض فيه مرحلة انجاز المشروع سي اي جي تي . بوجه خاص، سوف نقدم وحدة توليد "تعليمات مخصصة". ثم نختم بالدراسات التجريبية التي من خلالها قمنا باختبار فعالية وسرعة البرنامج سي اي جي تي على مجموعة من التطبيقات للتحقق من صحة المنهج الذي نتبعه.

تنتهي المقالة بخاتمة سنركز فيها على بعض الجوانب التي يمكن أن تلقى دراسة معمقة أكثر في مشروع سي اي جي تي و التي قد تتحول إلى مشاريع أخرى.

2 عرض بعض الأعمال في مجال التوليد الآلي للتعليمات البرمجية

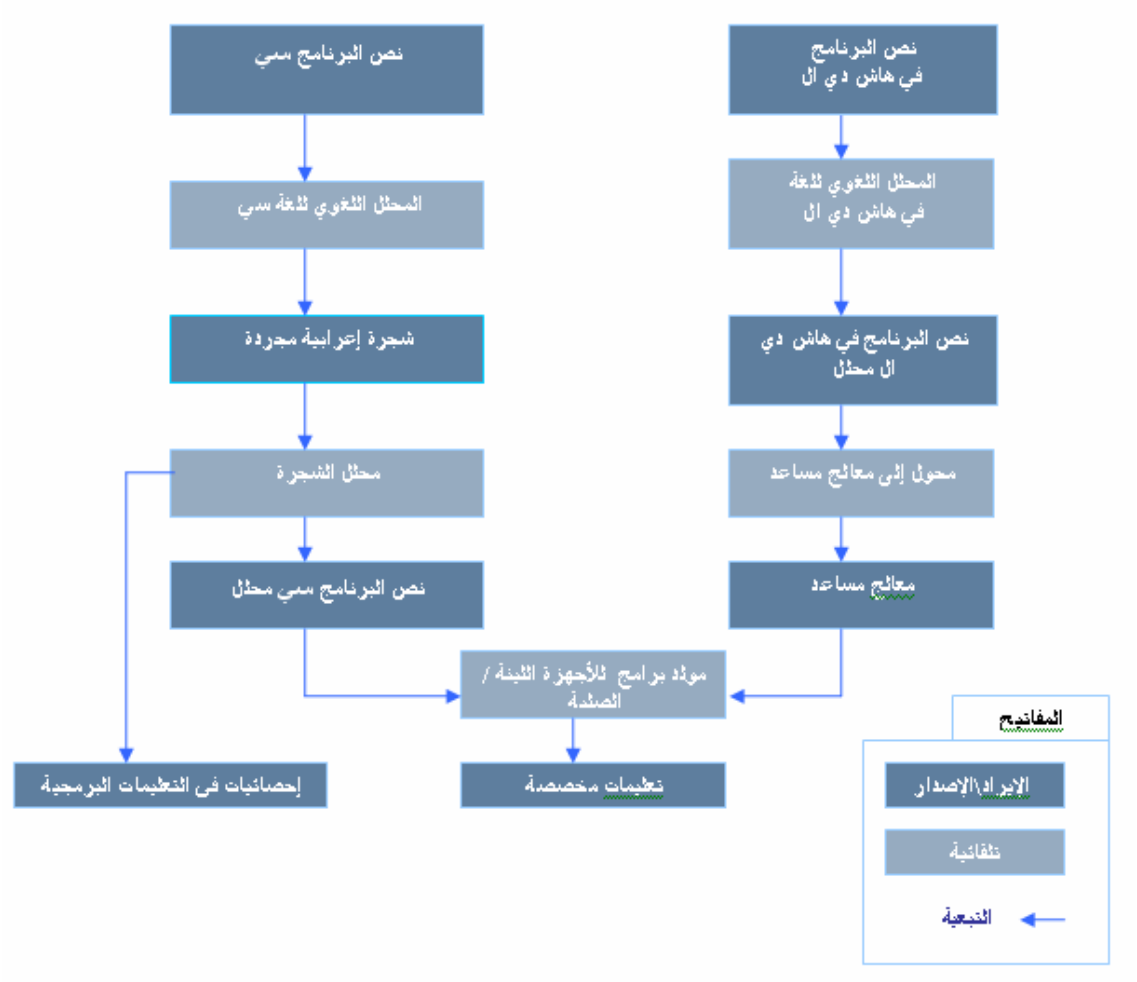
لقد ظهر مؤخرا العديد من المناهج التي تقدم حلولا للتوليد الآلي الأمثل للتعليمات البرمجية. حيث أن المشروع [4] يهدف إلى تصميم سريع لمحلل لغوي باستخدام مُصرّفات افلوس "FLOSS" (البرمجيات الحرة و مفتوحة المصدر) و قواعد لغوية. المنهج المقترح يعتمد على تصميم محلل معجمي قادر على كتابة سلسلة من الوحدات اللغوية في شكل أكس أم أل " XML ". وتخصيص مولد المحلل اللغوي لكي يكون قادرا على إنتاج قواعد لغوية و تعليمات برمجية تستطيع قراءة الوحدات اللغوية المكتوبة في شكل أكس أم أل. إن تحديث التعليمات البرمجية للمُصرّفات افلوس متاحة بجودة عالية لذلك فان المحللات اللغوية التي تستعمل افلوس تحتوي على جودة عالية. بيد أن هذا البحث يعتمد على تقنية واحدة لتصميم المحلل اللغوي بالإضافة إلى أن الخوارزميات الخاصة لبرمجة المحلل المعجمي واللغوي محدودة للغاية. في حين أن البحث [8] قد قدم أساليب للتوليد الآلي للتطبيقات الخاصة لأنظمة التشغيل. لكن هذه النهج تركز أساسا على قضايا جدولة التعليمات، و لا يقوم بتوليد تعليمات برمجية فعالة من خلال مواصفات النظام. المشروع سوكوس "SoCOS" [7]يقدم برمجية للتوليد الآلي للتعليمات البرمجية من خلال نماذج عالية المستوى لنظام التشغيل. ومع ذلك، سوكوس يحتاج إلى تعديل يدوي للحصول على التعليمات البرمجية. المشروع [5] يهدف إلى توليد التعليمات البرمجية من خلال لغة التصميم سيستام سي "SystemC". لكن لا يوجد هناك معلومات تتعلق بكيفية جدولة العمليات في نموذج الإدخال سيستام سي في مستوى الانجاز النهائي.
نهدف في هذا البحث إلى توليد تعليمات برمجية للأجهزة اللينة / الصلبة من خلال مواصفات عالية المستوى للنظام لغاية تسريع و تسهيل عملية تصميم الأنظمة المطمورة. لهذا الغرض قمنا خصيصا

بتصميم كل من المحلل اللغوي للغة سي و المحلل اللغوي للغة في هاش دي ال التي تقوم بدور رئيسي في عملية التوليد الآلي للتعليمات البرمجية. إن هذا البحث شبيه بما قدمه جاجسكي [6] من حيث الأهداف، حيث اقترح منهجا للإنتاج الآلي لبرمجيات للأجهزة اللينة / الصلبة من خلال مواصفات وظيفية.

3 منهج التوليد الآلي للتعليمات البرمجية

3.1 المخطط العام لمنهج التوليد الآلي للمشروع سي اي جي تي

يهدف مشروع سي اي جي تي إلى وضع نظام متكامل مشكل من عدة آليات مترابطة بجميع مستويات النظام كما يوضحه الشكل1



الشّكل1. المخطط العام لمشروع سي اي جي تي

يمكن أن نشرح المخطط العام لسير المشروع كالآتي:

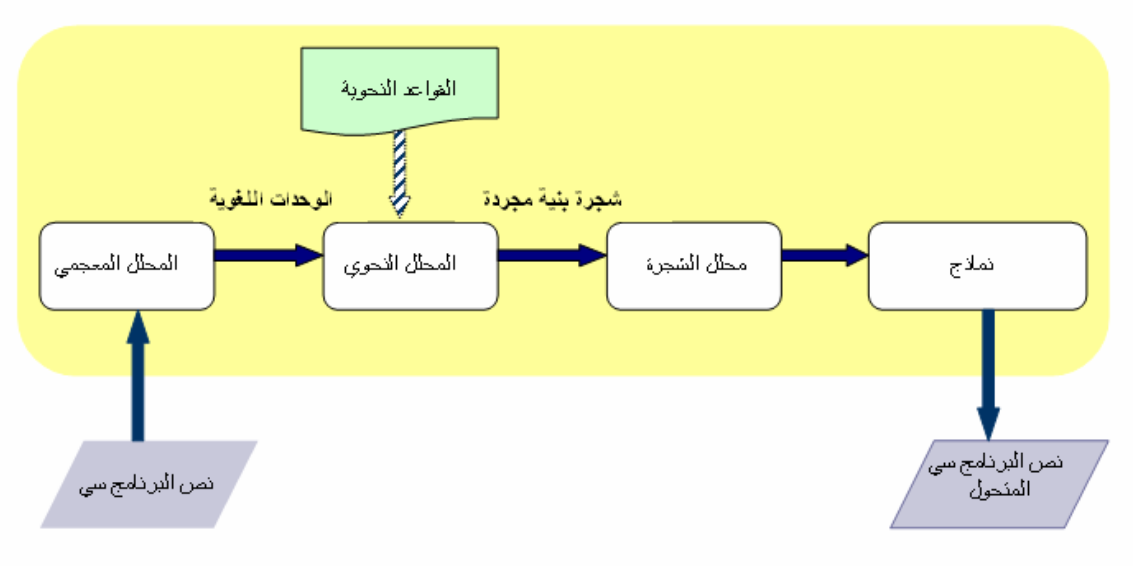
1- المحللات اللغوية سي و في هاش دي ال تحلل على التوالي نصوص البرنامج سي و في هاش دي ال
2- يتم تحويل نص البرنامج في هاش دي ال إلى معالج مساعد
3- يقوم المحلل سي بإصدار شجرة إعرابية مجردة ليتم من خلالها عملية إحصاء على العقد و توليد نص برنامج سي محلل
4- أخيرا، يقوم مولد برامج للأجهزة اللينة / الصلبة بتوليد تعليمات مخصصة بالاستناد إلى نص البرنامج سي المحلل و المعالج المساعد

في هذه المقالة، سوف نركز على الجانب اللغوي للمحلل سي حيث سنقوم بعرض تفاصيل تصميم و تطوير المحلل اللغوي سي.

3.2 المحلل اللغوي سي

لقد قمنا بتصميم محلل لغوي سي مخصص حسب حاجياتنا للتوليد الآلي للتعليمات البرمجية للأجهزة اللينة / الصلبة.حيث انه قادر على:

- الكشف عن تعقد الحلقات "for".
- الكشف عن تعقيد نص البرنامج من حيث العمليات الجبرية.
- الكشف عن إعلان المتغيرات.
- الكشف عن أنواع مختلفة من العمليات الجبرية في نص البرنامج سي.
- تحليل العمليات الجبرية في نص البرنامج سي.
- الكشف عن اسم المتغيرة التي تمثل نتيجة لعملية جبرية معينة.
- الكشف عن موقع المعطيات المتصلة بكل عملية في نص البرنامج المصدر



الشّكل2. تدفق بيانات المحلل اللغوي سي

يمكن أن نشرح المخطط العام لتدفق بيانات المحلل اللغوي سي(الشكل2) كالآتي:

1- مرحلة التحليل المعجمي: تعرض كل كلمة موجودة في النص إلى محلل معجمي حيث يتم تحويل أحرف نص البرنامج سي إلى عدد كبير من الرموز التي تكون متطابقة مع القواعد النحوية وهي ما نعبر عنها بالتعابير النمطية.

2- مرحلة التحليل النحوي: يتم تحليل تدفق الرموز رمزا برمز وإنشاء شجرة إعرابية مجردة.

3-تحليل شجرة إعرابية مجردة: يقوم محلل الشجرة بعبور الشجرة و تنفيذ العمليات الحسابية الفعلية على العقد؛

4- أيضا ، يستطيع محلل شجرة إنشاء نماذج لتوليد نص البرنامج سي بعد تغييرها.

إن التدقيق النحوي للغة سي يوفر حرية كبيرة للكتابة بالنسبة للمبرمج بخلاف لغة في هاش دي ال، ولهذا السبب، فان عملية تطوير المحلل اللغوي سي تعتبر معقدة للغاية.

لتسهيل هذه المهمة، قمنا باستخدام واجهة برمجة التطبيقات "regex" جافا لتصريف التعابير النمطية المستخدمة للتعرف على النص. في الفقرة الموالية، سنركز على مرحلة التحليل المعجمي أين قمنا بإنتاج التعابير النمطية.

3.3 إنتاج التعابير النمطية

التعابير النمطية هي أسلوب لوصف النصوص والتعرف عليها بطريق وصف مكوناتها من رموز، ووصف علاقات تلك الرموز من توال وتكرار، وذلك بكيفية نظامية يمكن لخوارزمية أن تفسرها وتطبقها على نص مُعطى لاستخراج الجزء المنطبق عليه التعبير النمطي[9].

في جافا، الفئتان باترن "Patern" و ماتشر"Matcher" يشاركان في توظيف التعابير النمطية. حيث أن طريقة كومبايل "compile" الموجودة في الفئة باترن تقوم بدور المحلل اللغوي بإنتاجها لشجرة البنية المجردة من خلال الوحدات اللغوية. اما الفئة ماتشرفهي التي تقوم بالبحث عن تطابق العبارة الواردة والتعبير النمطي من خلال الطريقة فايند "find".

لبناء التعابير النمطية نقوم باستخدام الحروف الخاصة المعروضة في جدول1

الرموز	الدلالة
.	يعوض كل الحروف
*	يعوض 0،1،او أكثر من حرف
?	يعوض حرف واحد
()	مجموعة
[]	مجال الحروف
{}	معدد
/	يجرد خوصصة الحرف الذي يواليه
^	النفي أو أول السطر
+	بسط
$	آخر السطر
|	علاقة منطقية

جدول1: الحروف الخاصة

أيضا يوجد فئات معرفة مسبقا للحروف يمكن استعمالها في كتابة التعابير النمطية(جدول2). كما أن المبرمج باستطاعته برمجة فئات أخرى حسب احتياجاته.

الفئة	الدلالة
\d	رقم مساوي ل :[0-9]
\D	ليس رقم:[^0-9]
\s	حرف ابيض: [\ t\n\x0B\ f\ r]
\S	ليس حرف ابيض: [^\ s]
\w	حرف من كلمة: [a-zA-Z_0-9]
\W	ليس حرف من كلمة : [^\w]
.	أي حرف

جدول2: فئات الحروف

لقد قمنا ببرمجة المحلل المعجمي بلغة جافا الذي يقوم بتحويل حروف نص البرنامج المصدر إلى التعابير النمطية المصنوعة لعدة أهداف من بينها :

- وجوب الرجوع إلى السطر في عدة حالات
- حذف التعليقات
- تحديد و إظهار الأقواس

في الخوارزمية الموالية ، قمنا بعرض جزء من برنامج التحليل المعجمي

```
private void cagt_lexer() {
    // @author Emna Kallel
    String ligne;
    try { BufferedReader in = new BufferedReader(new FileReader(C_file));
        try {
            while ((ligne = in.readLine()) != null){
                if (!isForStatement(ligne))
                    ligne = ligne.replaceAll(";",";\n");
                    analysableStringFile += ligne +"\n"; ";"العودة إلى السطر إذا كان هناك//
            }
            in.close();
            originalStringFile = analysableStringFile;
           } catch (IOException ex) {ex.printStackTrace(); }
        } catch (FileNotFoundException ex) {ex.printStackTrace();}

    analysableStringFile = analysableStringFile.replaceAll("(?s)(/\\*)(.*?)(\\*/)",""); حذف التعليقات//
    analysableStringFile = analysableStringFile.replaceAll("(//.+)","");   حذف التعليقات//
    analysableStringFile = analysableStringFile.replaceAll("="," = ");
    analysableStringFile = analysableStringFile.replaceAll("\\)"," ) "); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("\\("," ( "); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("(\\)  \\))","))"); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("(\\)  \\))","))"); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("(\\(  \\()","(("); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("(\\(  \\()","(("); تحديد الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("\\{","\n{\n"); إظهار الأقواس//
    analysableStringFile = analysableStringFile.replaceAll("\\}","\n}\n"); إظهار الأقواس//
}
```

خوارزمية التحليل المعجمي

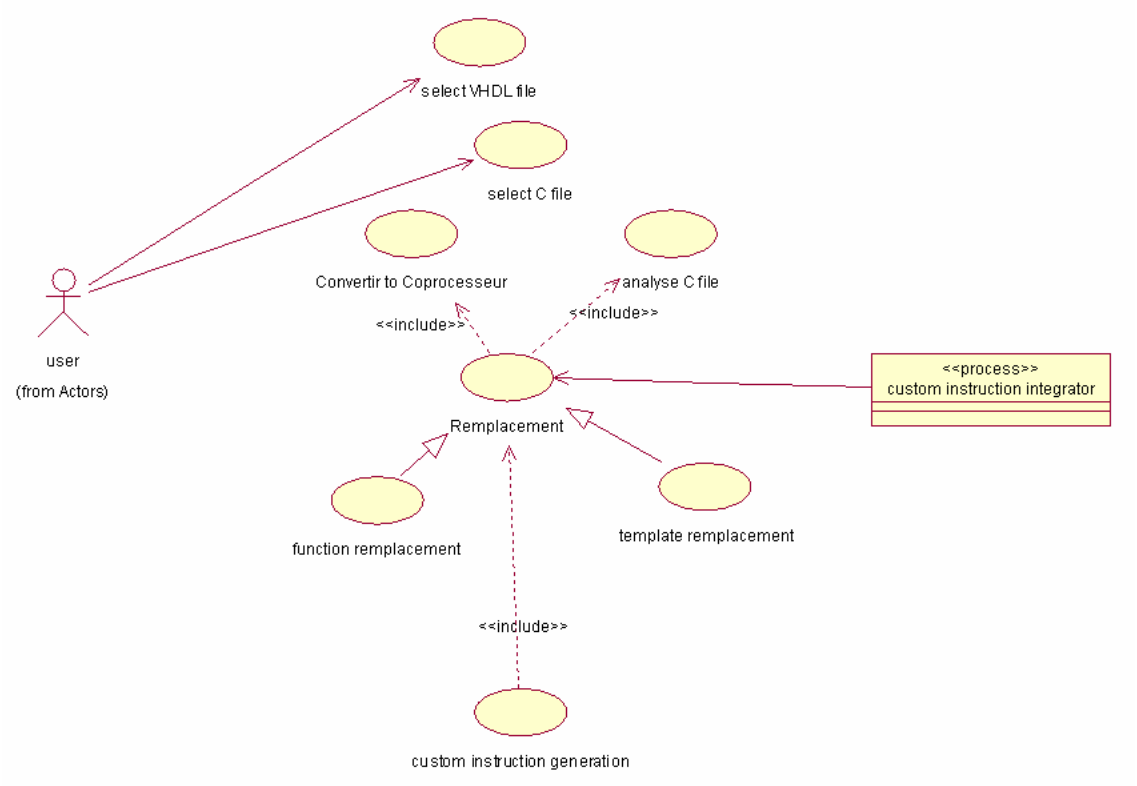
4 انجاز المشروع سي اي جي تي (CAGT)

4.1 تقديم المشروع سي اي جي تي (CAGT)

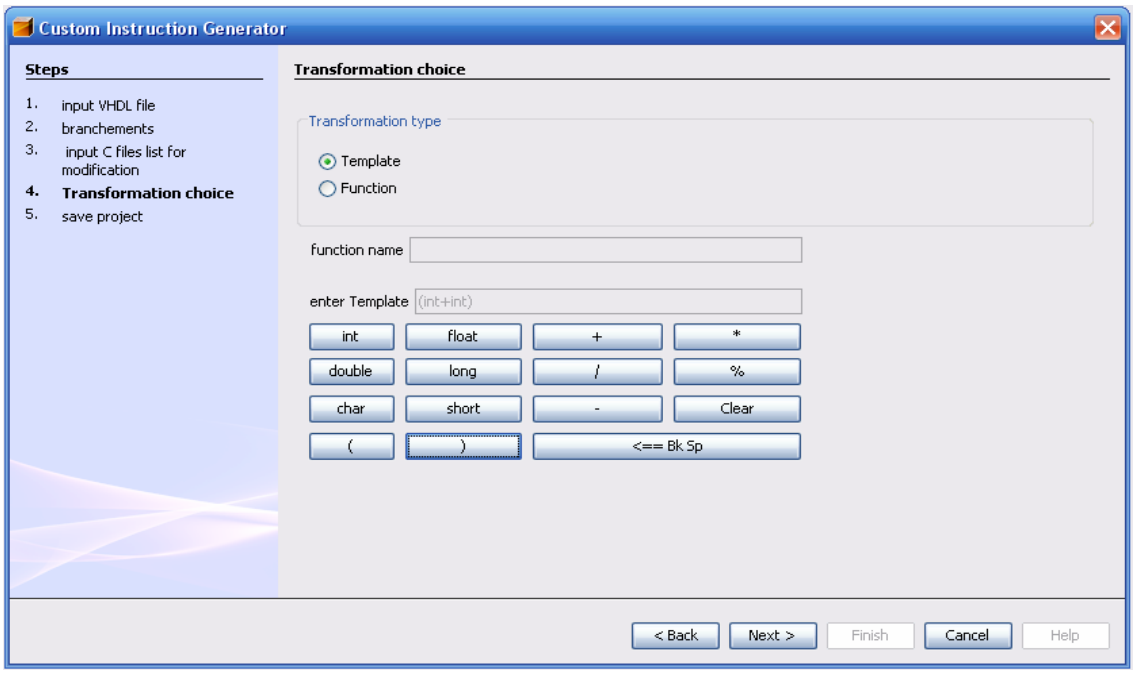
لقد قمنا بتطبيق منهج التوليد الآلي للتعليمات البرمجية بانجاز المشروع سي اي جي تي الذي يعتمد على التحليل اللغوي للتوليد الآلي لبرمجيات للأجهزة اللينة / الصلبة من خلال مواصفات وظيفية عالية المستوي. حيث أن سي اي جي تي يوفر واجهة مستخدم رسومية شفافة التي بدورها تستطيع أن تسهُل وتسرُع في عملية تصميم تطبيقات للأجهزة اللينة / الصلبة (شكل 4). سي اي جي تي يتكون من عدة وحدات معتمدة علي التحليل اللغوي وهي:

- وحدة إدماج التعليمات المخصصة
- وحدة تحويل عملية جبرية
- وحدة توليد إحصائيات

يبين الشكل 3 مخطط لغة العرض الموحدة لوحدة توليد التعليمات المخصصة. إذ أن المستخدم يقوم بإدخال نصوص البرامج سي و في هاش دي ال . ثم يقوم مولد التعليمات المخصصة بالتحويلات المناسبة للحصول على تعليمات مخصصة.



الشّكل3. مخطط لغة العرض الموحدة لوحدة توليد التعليمات المخصصة



الشّكل4. واجهة المستخدم الرسومية لتوليد التعليمات المخصصة

4.2 مثال لتوليد آلي للتعليمات المخصصة

نبين ، في هذا المثال، كيف تتم عملية التوليد الآلي للتعليمات المخصصة بالنسبة لنص البرنامج سي. حيث أن المستعمل يقوم بإدخال نص البرنامج سي ثم يقوم باختيار نوع التغيير(الشّكل4).

في هذا المثال، اخترنا أن نولد تعليمات مخصصة للعملية الجبرية (int+int). نلاحظ في نص البرنامج المصدري سي في الشكل5 أن العملية الجبرية الوحيدة من نوع (int+int) هي ser=(e+f) . يبين نص البرنامج سي المعدل في الشكل5 نجاح المحلل اللغوي سي الذي قمنا بتصميمه في الكشف عن العملية الجبرية من نوع (int+int) و تحويلها إلى تعليمات مخصصة.

و بالتالي فهو قادر على كشف أنواع مختلفة من العمليات الجبرية في نص البرنامج سي.

نص البرنامج سي المصدر	نص البرنامج سي المعدل

`#include <stdio.h>`
`char a,b,h,U,n;`
`int e,f, ser,s,g;`
`void main(){`
 	
`int i = 0;`
`s = ajouter(a[i],b);`
`s = (a+b);`
 	
`ser = (e+f);`
`for(i=0;i<10;i++){`
`h = (n+h/U);`
`}`
 	
`}`
	`#include "system.h"`
	`#include <stdio.h>`
	`char a,b,h,U,n;`
	`int e,f, ser,s,g;`
	`void main ( )`
	`{`
	`int i = 0;`
	`s = ajouter ( a[i],b ) ;`
	`s = ( a+b ) ;`
	`//'e' placed in (DATAA( 15 DOWNTO 0 ))`
	`//'f' placed in (DATAA( 31 DOWNTO 16 ))`
	`long DATAA = e << f;`
	`long DATAB = 0;`
	`ser = ALT_CI_tt_COP(DATAA,DATAB);`
	`for ( i = 0;i<10;i++ )`
	`{`
	`h = ( n+h/U ) ;`
	`}`
	`}`

الشّكل5. مثال لتوليد آلي للتعليمات المخصصة

4.3 الدراسات التجريبية

للتحقق من فعالية سي اي جي تي ، قمنا باختباره على مجموعة من التطبيقات. حيث قمنا بالمقارنة بين نتائج النمذجة التمثيلية لهذه التطبيقات على مصفوفة البوابات القابلة للبرمجة حقلياً . فالدراسات التجريبية تكمن في مقارنة مدة تصميم التطبيقات مع او دون سي اي جي تي . الجدول3 يعطينا فكرة عن المكاسب التي حققها المشروع سي اي جي تي . حيث انه تبين انه باستعمال سي اي جي تي نحقق مدة تصميم أسرع ب10 مرات لتطبيقات بسيطة ووقت أسرع ب5 مرات لتطبيقات أكثر تعقيداً.

التطبيقات	حجم نص البرنامج سي المصدر	عدد التعليمات المخصصة	مدة التصميم التقليدية	مدة التصميم باستعمال سي اي جي تي	مكسب من حيث سرعة سي اي جي تي
تقدير الحركة	306 أسطر	5	15 يوم	3 أيام	×5
التحديد الكمي	180 سطر	3	10 أيام	2 أيام	×5
ترقيم هوفمان	210 أسطر	3	15 يوم	3 أيام	×5
إسقاط	153 سطر	2	7 أيام	1 يوم	×7
التطبيع	159 سطر	2	7 أيام	1 يوم	×7
ضرب المصفوفات	80 سطر	1	10 أيام	1 يوم	×10

الجدول3: المكاسب التي حققها المشروع سي اي جي تي

5 الخاتمة

لقد بينا في هذا المقال كيف أن منهج يستند إلى محلل لغوي قادر علي تسريع وتسهيل تصميم الأنظمة المطمورة. حيث أن المحلل اللغوي للغة سي الذي قمنا بتصميمه يقوم بتحليل نص البرنامج سي ليستنتج عدة خاصيات رئيسية مثل الكشف عن أنواع مختلفة من العمليات الجبرية. ثم بعد ذلك سيتم استخدام آليات لتوليد التعليمات المخصصة للعمليات الجبرية. لقد قمنا ببرمجة البرنامج سي اي جي تي الذي يقوم بإدماج كل من المحلل اللغوي للغة سي و المحلل اللغوي للغة في هاش دي ال التي قمنا بتصميمها لغرض التوليد الآلي للتعليمات البرمجية للأجهزة اللينة / الصلبة. دراسة تجريبية، قد ساعدتنا على إظهار فعالية و سرعة البرنامج سي اي جي تي التي يمكن أن تصل إلى 90 في المائة مقارنة بتصميم يدوي.
كأعمال مستقبلية، سنحرص على تحسين المحلل اللغوي سي ليكون قادر على معالجة قضايا التحويل الآلي إلى تعليمات برمجية متوازية.

المراجع

[1] Christian Haubelt, Thomas Schlichter, Joachim Keinert and Mike Meredith. “SystemCoDesigner: Automatic Design Space Exploration and Rapid Prototyping from Behavioral Models”, DAC 2008, June 8– 13, 2008, Anaheim, California, USA.

[2] R. Leupers. “Code Optimization Techniques for EmbeddedProcessors – Methods, Algorithms, and Tools”. Kluwer Academic Publishers, Nov. 2000.

[3] I. Emna Kallel ,Yassine Aoudni, Mohamed Abid (2009), Rapid Design of Specific MPSoC prototype within FPGA, International Conference On Signals, Circuits & Systems (IEEE SCS 2009).

[4] Kazuaki Maeda (2009), Quick Parser Development Using Modified Compilers and Generated Syntax Rules , Proceedings of the international conference on Computational and information science, Houston, USA ,410-415.

[5] F. Herrera, H. Posadas, P. Snchez, and E. Villar. “Systematic embedded software generation from systemc. Proceedings of Design Automation and Test in Europe”, DATE, 2003.

[6] Daniel Gajski and Samar Abdi. “Automatic Generation of Equivalent Architecture Model from Functional Specification”. DAC 2004, June 7–11, 2004, San Diego,

California, USA.
[7] D. Desmet. “Operating system based software generation for systemon- hip”. In Proceedings of the Design Automation Conference, June 2000.
[8] L. Gauthier. “Automatic generation and targeting of application specific operating systems and embedded systems software”, IEEE Trans. on CAD, November 2001.
[9] http://ar.wikipedia.org/wiki

الملخص باللغة الانجليزية

Title: Parser-based automatic code generation approach for embedded systems

Abstract: In this paper, we show how an approach parsers-based can facilitate and accelerate the embedded systems designer’s tasks. In fact, the C parser accepts in input a standard C code. After a syntactic analysis, it allows to deduct two characteristic keys: the code complexity in term of arithmetical operator and the instructions place associated to every operator in the code. Then, an automaton of code generation will be used to generate the operator’s hardware code and to update the C code with the new code. We developed the CAGT environment which integrates our C and VHDL parsers and allows the automatic generations of a software/hardware code to the target architecture through a high level specification. An experimental study allows us proving the CAGT efficiency and speed that can attain to 90% compared with a manual conception.

Keywords: embedded systems, parser, automatic generation, CAGT.

المصطلحات

تعليمات برمجية	code
الأنظمة المطمورة	Embedded Systems
العمليات الجبرية	Arithmetic operation
محلل لغوي	parser
التعليمات المخصصة	Custom Instructions
برنامج توليد التعليمات المخصصة	Custom Instructions Generator Tool (CAGT)
لغة سي	Language C
لغة في هاش دي ال	Language VHDL
التوليد الآلي للتعليمات البرمجية	Automatic code generation
تعليمات البرمجية للأجهزة اللينة / الصلبة	Code software/hardware
بتصميم يدوي	Manuel conception
الخوارزميات	algorithms
التطبيقات	application
الدوائر المتكاملة	integrated circuit

زيادة مستوى التجريد	Increase abstraction level
المواصفات التوظيفية	Functional specification
مُصرّفات	compilers
محلل نحوي	Syntactical analyzer
محلل معجمي	Lexical analyzer
مخصص	specific
جدولة التعليمات	Scheduling
نظام التشغيل	Operating system
مواصفات عالية المستوى	High level specification
شجرة إعرابية مجردة	Abstract syntactical tree
المتغيرات	variables
الحروف الخاصة	Special characters
واجهة برمجة التطبيقات	API
تعبير نمطي	Regular expression
فئات	classes
حرف	characters
مخطط لغة العرض الموحدة	UML diagram
النمذجة التمثيلية	prototyping
مصفوفة البوابات القابلة للبرمجة حقلياً	FPGA
تقدير الحركة	Motion estimation
التحديد الكمي	quantification
ترقيم هوفمان	Huffman coding
إسقاط	projection
التطبيع	normalization
ضرب المصفوفات	Matrix multiplication
تعليمات برمجية متوازية	Parallel code
آليات،تلقائية	Automaton