Программно-аппаратная реализация конструктивных геометрических моделей

Д.В. Волошинов[1], К.Н. Соломонов[2]

[1] СПбГУТ им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, Россия
denis.voloshinov@yandex.ru

[2]Филиал Ростовского государственного университета путей сообщения в г. Воронеж, Россия
konssol@list.ru

Аннотация. Статья посвящена рассмотрению ряда вопросов аппаратно-программной реализации конструктивных геометрических моделей. Богатый арсенал теоретических исследований в области конструктивной геометрии длительное время не находил должного применения в виду отсутствия инструментов воплощения таких моделей средствами вычислительной техники. Разработка и совершенствование системы геометрического моделирования Симплекс, в которой любая геометрическая конструкция рассматривается как преобразователь информации, представленной сигналами геометрической природы, открыла возможность применения достижений геометрической науки в вычислительных приложениях, а также разработки аппаратных средств, реализующих геометрические методы вычислений и предоставляющих новый графический интерфейс. Развиваемая авторами концепция направлена на создание специализированных ускорителей геометрических преобразований.

Ключевые слова: научная визуализация; конструктивное геометрическое моделирование; геометрический эксперимент, встраиваемые системы, графический интерфейс.

Abstracts. The article is devoted to consideration of a number of issues of hardware and software implementation of constructive geometric models. A rich arsenal of theoretical research in the field of constructive geometry has not been properly used for a long time due to the lack of tools for translating such models using computer technology. The development and improvement of the Simplex geometric modeling system, in which any geometric design is considered as a converter of information represented by signals of a geometric nature, has opened the possibility of applying the achievements of geometric science in computing applications, as well as the development of hardware that implements geometric calculation methods and provides a new graphical interface. The concept developed by the authors is aimed at creating specialized accelerators of geometric transformations.

Keywords: scientific visualization; constructive geometric modeling; geometric experiment, embedded systems, graphical interface.

1 Введение

Конструктивная геометрия является одним из разделов математики, который в недалеком прошлом находил широкое применение в практике инженерного проектирования, обработке и представления результатов физических экспериментов, наглядного отображения многопараметрических процессов. За свою историю синтетическая геометрия накопила богатейший арсенал средств моделирования, однако, в условиях современной информатизации и цифровизации он остается практически невостребованным из-за необходимости представления геометрической информации в аналитическом виде. Даже краткий перечень работ авторов прошлого и современности демонстрирует существо этого противоречия [1-3, 5, 14-21].

В развитие конструктивной геометрии внесли вклад многие ученые, среди которых особое место занимают работы профессора К.И. Валькова [4], предложившего рассматривать любую геометрическую конструкцию в контексте действия кибернетического устройства, обрабатывающую информацию особого рода – геометрическую. Такие устройства получили название геометрических машин, производящих определенную работу по переработке информации. К сожалению, во время возникновения этих идей уровень развития вычислительной техники не только не позволял соединить возможности геометрического метода с возникающими информационными технологиями, но имел обратный эффект, в результате которого геометрия стала противопоставляться достижениям в области кибернетики. Попытка ассоциировать геометрические конструкции с механическими системами, несмотря на принципиальную возможность такого подхода, не могла по естественным причинам конкурировать с новой развивающейся технологией, в результате чего геометрическая наука была отнесена к категории устаревающих и не имеющих перспектив в сравнении с аналитической ветвью математики. В результате сложившейся тенденции интерес исследователей к геометрическому знанию стал угасать, а невозможность практического приложения результатов исследований к новым информационным реалиям привела к сорокалетнему застою в геометрических изысканиях.

Переосмысление концепции геометрических машин в контексте новых информационных парадигм позволило по-новому подойти к решению застарелой проблемы, ликвидировать искусственное противостояние синтетической геометрии и информатики. Результатом проведенных исследований стало создание системы конструктивного геометрического моделирования Симплекс [8, 10], позволившей не только дать новый толчок к развитию геометрической науки, но и определить способы практической реализации ее результатов как новых информационных технологий: в виде программ, систем и их интерфейса, а также перспективных аппаратных реализаций. Результаты, которые удалось достичь авторам в этом направлении, представлены в работах [6, 7 9, 13]. В рамках данной статьи обсуждаются некоторые новые достижения в обозначенной области исследований, касающиеся программно-аппаратной реализации конструктивных геометрических моделей на электронных устройствах с микроконтроллерным управлением и в перспективе на устройствах ПЛИС.

2 Механизмы расширения функциональных возможностей системы Симплекс

Исследования, проводимые в области конструктивного геометрического моделирования и осуществляемые с помощью системы Симплекс, приводят к необходимости разработки новых все более сложных функций, компонентами которых являются ранее разработанные объекты и функции. Для обеспечения расширения функционального

состава в системе в ней предусмотрены два механизма. Первый (внутренний) позволяет создавать процедуры-функции и иерархии производных объектов с использованием визуально-графических средств самой системы. Таким образом пользователь получает возможность неограниченно расширять функциональный состав и, что наиболее важно, исследовать и отлаживать геометрические алгоритмы, получая полный комплекс информационных средств для влияния на значения составляющих их объектов и наблюдения за получаемыми данными как в числовой, так и в графической форме. Для обеспечения действия этого механизма система осуществляет интерпретацию команд и данных, что непременно сказывается на скорости выполнения вычислений и на производительности системы в целом. По мере отладки модели необходимость интерпретационного исполнения алгоритмов отпадает, вследствие чего более рациональным становится подход, подразумевающий трансляцию алгоритма до уровня машинного кода. Второй предусмотренный в системе механизм реализует ретрансляцию внутрисистемного представления геометрических алгоритмов на язык Pascal системы Delphi, после чего полученный код транслируется, компонуется и становится одной из функций библиотеки системы, к которой обеспечивается типовой доступ через базовый системный интерфейс.

В работе [11] была подробно рассмотрена и обоснованы специфика организации системы конструктивного геометрического моделирования, приведены правила ее организации, соблюдение которых в наибольшей степени способствует удобному и естественному стилю деятельности разработчика конструкции геометрических машин. Особенно важными из них являются следующие требования: отказ от текстового ввода команд, соблюдения их порядка и «прозрачная» организация множественного процедурного вызова. Необходимость соблюдения этих правил определила структуру внутренних механизмов системы и способы организации функционального состава системы. Так, например, на нижнем уровне находится большинство функций, выполняющих операции над одним или несколькими семантически различными объектами геометрической модели, представленными единичными экземплярами. Пример такой функции, разработанной для решения задачи Аполлония и представленный текстом на языке Pascal, имеет префикс EExec (табл. 1). Ко второму уровню относятся функции, имеющие префикс XExec и отвечающие за организацию выходных списочных переменных, формируемых из единичных решений путем многократного обращения. В них также реализованы механизмы множественности [12], определяющие как стандартные, так и пользовательские методы согласования параметров для организации многократного процедурного вызова. Функции этого уровня имеют префикс Exec (табл. 1) и, несмотря на возникающий в этих механизмах прецедент параллелизма вычислений, выполняют это действие, как циклическую последовательность операций.

Таблица 1. Фрагмент программы на языке Pascal, сгенерированный системой Симплекс для моделирования геометрической конструкции задачи Аполлония

```
function EExecAppolonius(in_prm1,in_prm2,in_prm3: TObj; var
out_prm1,out_prm2: TObj; Att_1,Att_2: TAtt; Sg1,Sg2,Sg3: integer;
OW1,OW2: pointer): boolean;
var d1, d2, d3, d4, d5, d6, d7, d8, d9, o1, o2, o3, o4, o5, o6, o7, p1, p10,
p11, p12, p13, p14, p15, p16, p17, p18, p19, p2, p20, p21, p3, p4, p5, p6, p7,
p8, p9: TObj;
```

```
      Success: boolean;
    begin

MassNIL([@d1,@d2,@d3,@d4,@d5,@d6,@d7,@d8,@d9,@o1,@o2,@o3,
@o4,@o5,@o6,@o7,@p1,@p10,@p11,@p12,@p13,@p14,@p15,@p16,@p17
,@p18,@p19,@p2,@p20,@p21,@p3,@p4,@p5,@p6,@p7,@p8,@p9]);

       Result:=TRUE;
       d1:=in_prm1.CreateCopy(NIL);
       d1.OAtt:=in_prm1.OAtt;
       d2:=in_prm2.CreateCopy(NIL);
       d2.OAtt:=in_prm2.OAtt;
       d3:=in_prm3.CreateCopy(NIL);
       d3.OAtt:=in_prm3.OAtt;

       repeat
          Success:=FALSE;
          if not Assigned(o1) and not Assigned(p1) and not Assigned(p2)
then

Success:=EExecO3(d1,d2,o1,p1,p2,Att5,Att0,Att0,Sg1,Sg2,NIL,NIL,NIL) or
Success;
     if not Assigned(o2) and not Assigned(p3) and not Assigned(p4) then

Success:=EExecO3(d2,d1,o2,p3,p4,Att5,Att0,Att0,Sg2,Sg1,NIL,NIL,NIL)
or Success;
    if not Assigned(o3) and not Assigned(p6) and not Assigned(p7) then
      Success:=EExecO3(d1,d3,o3,p6,p7,Att5,Att0,Att0                Sg1
Sg3,NIL,NIL,NIL) or Success;
    if not Assigned(o4) and not Assigned(p8) and not Assigned(p9) then
      Success:=EExecO3(d3,d1,o4,p8,p9,Att5,Att0,Att0,Sg3
Sg1,NIL,NIL,NIL) or Success;
    if not Assigned(o5) and not Assigned(p11) and not Assigned(p12) then
      Success:=EExecO3(d2,d3,o5,p11,p12,Att5,Att0,Att0              Sg2
Sg3,NIL,NIL,NIL) or Success;
    if not Assigned(o6) and not Assigned(p13) and not Assigned(p14) then
      Success:=EExecO3(d3,d2,o6,p13,p14,Att5,Att0,Att0              Sg3
Sg2,NIL,NIL,NIL) or Success;
    if not Assigned(d4) then
      Success:=EExecOK001(d1,d2,d3,d4,Att5,Sg1,Sg2,Sg3,NIL)          or
Success;
    if not Assigned(p5) then
      Success:=EExecP2(o1,o2,p5,Att5,1,1,NIL) or Success;
    if not Assigned(p10) then
      Success:=EExecP2(o3,o4,p10,Att5,1,1,NIL) or Success;
    if not Assigned(p15) then
      Success:=EExecP2(o5,o6,p15,Att5,1,1,NIL) or Success;
    if not Assigned(o7) then
      Success:=EExecO000(p5,p10,p15,o7,Att5,1,1,1,NIL) or Success;
    if not Assigned(d5) then
      Success:=EExecOK001(d1,d4,o7,d5,Att5,Sg1,1,1,NIL) or Success;
    if not Assigned(d6) then
      Success:=EExecOK001(d2,d4,o7,d6,Att5,Sg2,1,1,NIL) or Success;
    if not Assigned(d7) then
      Success:=EExecOK001(d3,d4,o7,d7,Att5,Sg3,1,1,NIL) or Success;
    if not Assigned(p16) and not Assigned(p17) then
      Success:=EExecP3(d1,d5,p16,p17,Att0,Att0,Sg1,1,NIL,NIL)          or
```

```
Success;
    if not Assigned(p18) and not Assigned(p19) then
      Success:=EExecP3(d2,d6,p18,p19,Att0,Att0,Sg2,1,NIL,NIL)          or
Success;
    if not Assigned(p20) and not Assigned(p21) then
      Success:=EExecP3(d3,d7,p20,p21,Att0,Att0,Sg3,1,NIL,NIL)          or
Success;
    if not Assigned(d8) then
      Success:=EExecD4(p17,p18,p20,d8,Att5,1,1,1,NIL) or Success;
    if not Assigned(d9) then
      Success:=EExecD4(p16,p19,p13,d9,Att5,1,1,1,NIL) or Success;
       until not Success;
       out_prm1:=d9.CreateCopy(OW1);
       out_prm1.OAtt:=Att_1;
       out_prm2:=d8.CreateCopy(OW2);
       out_prm2.OAtt:=Att_2;

MassFree([d1,d2,d3,d4,d5,d6,d7,d8,d9,01,02,03,04,05,06,07,p1,p10,p11,p1
2,p13,p14,p15,p16,p17,p18,p19,p2,p20,p21,p3,p4,p5,p6,p7,p8,p9]);
    end; // EExecAppolonius

    function XExecAppolonius (FM: string; _W1,_W2,LC1,LC2,LC3:
TNewList; Att1,Att2: TAtt): boolean;
    var
      I,J,s_N, I1,I2,I3, Sg1,Sg2,Sg3: integer;
      CI,CO,Seq: TNewList;
      s_Prm1,s_Prm2,s_Prm3 : TObj;
    label lb;

    procedure CALC;
    var
       s_Out1,  s_Out2: TObj;
    begin

Result:=EExecABCD(s_Prm1,s_Prm2,s_Prm3,s_Out1,s_Out2,Att1,Att2,S
g1,Sg2,Sg3,_W1,_W2);
        if Assigned(s_Out1) then
        begin
          s_Out1.Parents.Add(s_Prm1);         s_Out1.Parents.Add(s_Prm2);
s_Out1.Parents.Add(s_Prm3);
          _W1.Add(s_Out1);
          case Att1.View of
          view_AND:   Tobj(_W1[_W1.Count-1]).View:=s_Prm1.View   and
s_Prm2.View and s_Prm3.View;
          view_OR:      Tobj(_W1[_W1.Count-1]).View:=s_Prm1.View      or
s_Prm2.View or s_Prm3.View;
          end; // case
        end;
        if Assigned(s_Out2) then
        begin
          s_Out2.Parents.Add(s_Prm1);         s_Out2.Parents.Add(s_Prm2);
s_Out2.Parents.Add(s_Prm3);
          _W2.Add(s_Out2);
          case Att2.View of
          view_AND:   Tobj(_W2[_W2.Count-1]).View:=s_Prm1.View   and
s_Prm2.View and s_Prm3.View;
          view_OR:      Tobj(_W2[_W2.Count-1]).View:=s_Prm1.View      or
```

```
s_Prm2.View or s_Prm3.View;
          end; // case
        end;
    end; // CALC

    var
    Lim1,Lim2,Lim3,Lim4: integer;
    begin // XExecAppolonius
       if FM = '0' then
       begin
       s_N:=Amin1(Amin1(LC1.Count-1,LC2.Count-1),LC3.Count-1);
       for I:=0 to s_N do
       begin
          s_Prm1:=GetLC(LC1,I,SG1,Lim1);
          s_Prm2:=GetLC(LC2,I,SG2,Lim2);
          s_Prm3:=GetLC(LC3,I,SG3,Lim3);
         CALC;
       end;
       goto lb;
       end; // FM=0

       if FM = '1' then
       begin
         for I1:=0 to LC1.Count-1 do
         for I2:=0 to LC2.Count-1 do
         for I3:=0 to LC3.Count-1 do
         begin
     s_Prm1:=LC1[I1]; s_Prm2:=LC2[I2]; s_Prm3:=LC3[I3];
     s_Prm1:=GetLC(LC1,I1,SG1,Lim1);
     s_Prm2:=GetLC(LC2,I2,SG2,Lim2);
     s_Prm3:=GetLC(LC3,I3,SG3,Lim3);
     CALC;
         end;
         goto lb;
       end; // FM=1

       CI:=TNewList.Create(NIL);
       CI.Add(LC1);  CI.Add(LC2);  CI.Add(LC3);
       SEQ:=TNewList.Create(NIL);
       Sequensor(FM,CI,CO,SEQ);
       for I:=0 to CO.Count-1 do
       begin
          s_Prm1:=TNewList(CO[I])[integer(SEQ[0]^)];
          s_Prm2:=TNewList(CO[I])[integer(SEQ[1]^)];
          s_Prm3:=TNewList(CO[I])[integer(SEQ[2]^)];
          if zn_Minus in TNewList(CO[I]).Chars[integer(SEQ[0]^)] then
Sg1:=-1 else Sg1:=1;
          if zn_Minus in TNewList(CO[I]).Chars[integer(SEQ[1]^)] then
Sg2:=-1 else Sg2:=1;
          if zn_Minus in TNewList(CO[I]).Chars[integer(SEQ[2]^)] then
Sg3:=-1 else Sg3:=1;
          CALC;
          TNewList(CO[I]).DelDestroy;
       end;
       CO.DelDestroy;
       CI.DelDestroy;
       for I:=0 to Seq.Count-1 do FreeMem(SEQ[I],SizeOf(integer));
```

```
        SEQ.DelDestroy;

      lb:
    end; // XAppolonius

    function ExecAppolonius (var PTS: TStroka): boolean;
    var
      All,R: boolean;      I: integer;    WL1,WL2: TNewList;_W1,_W2:
TNewList;  Att1,Att2: TAtt;
      LC1,LC2,LC3: TNewList; LCR1,LCR2,LCR3: TNewList;    PN1,PN2:
TNames;

    begin // ExecAppolonius
      All:=TRUE;
      Result:=FALSE;
      WL1:=PTS.ULeft[0];  WL2:=PTS.ULeft[1];
      _W1:=TPointer(WL1[0]).PN.OList;
_W2:=TPointer(WL2[0]).PN.OList;
      Att1:=TPointer(WL1[0]).Att; Att2:=TPointer(WL2[0]).Att;
      Pn1:=TPointer(WL1[0]).Pn;   All:=MakeAtt(PN1.AList,Att1,PN1.FAtt)
and All;
      Pn2:=TPointer(WL2[0]).Pn;
All:=MakeAtt(PN2.AList,Att2,PN2.FAtt) and All;

      All:=TAlg(PTS.Alg).Present(PTS,0,['?'],LC1,LCR1) and All;
      All:=TAlg(PTS.Alg).Present(PTS,1,['?'],LC2,LCR2) and All;
      All:=TAlg(PTS.Alg).Present(PTS,2,['?'],LC3,LCR3) and All;

      if (LC1.Count=0) or (LC2.Count=0) or (LC3.Count=0)
      or not All then
      begin
        LC1.DelDestroy;  LC2.DelDestroy; LC3.DelDestroy;  Exit;
      end;

      R:=XExecABCD(PTS.FM,_W1,_W2,LC1,LC2,LC3,Att1,Att2);
      PTS.Executed:=R;
      Result:=R;
      LC1.DelDestroy;  LC2.DelDestroy;  LC3.DelDestroy;

      Markus(_W1,Att1.View);  Markus(_W2,Att2.View);
    end; // ExecAppolonius
```

Подсистема трансляции на язык Pascal позволяет автоматически получать сгенерированные пользователем функции обслуживающих функций всех трех уровней, обеспечивая тем самым возможность расширения самой системы Симплекс. Однако разработанная концепция позволяет расширить ее применение и в других целях. В частности, генерация только лишь функций с форматом префикса «EExec» позволяет создавать отторгаемые от системы Симплекс приложения для решения конкретных пользовательских задач без применения самой системы Симплекс. Этот же механизм в совокупности с разработкой соответственных библиотек на языках JavaScript и MaxScript позволили получить скрипты, обеспечивающие интерактивную работоспособность геометрических схем в среде интернет-браузеров, а также разработать методику автоматизированного синтеза программ, функционирующих в среде системы 3DSMax фирмы Autodesk и обеспечивающих высококачественную визуализацию геометрических объектов в моделях

не только трехмерного пространства, но и на трехмерных проекционных картинах пространств более высоких размерностей [7]. Ввиду того что языки JavaScript и MaxScript являются интерпретируемыми, время получения результатов в интернет-браузерах и в системе 3DSMax значительно больше, чем собственно в системе Симплекс, которая пополняет свой состав за счет трансляции кода. Однако, несмотря на неизбежные временные издержки, предложенный подход позволил качественно изменить ситуацию с системным распространением геометрических знаний и обеспечить их представление в доступной и понятной визуальной форме. Предлагаемая информационная технология позволяет рассматривать синтетическую геометрию как специализированный раздел информатики.

3 Аппаратная реализация конструктивных геометрических моделей

Развитие концепции геометрических машин как информационных преобразователей универсального назначения естественным образом согласуется с современными тенденциями развития автономных электронных устройств. Так, например, фирма Mikroelektronika (Сербия) специализируется на разработке и внедрении систем быстрого прототипирования современных электронных устройств, основными областями применения которых являются получение и обработка данных, автоматизация, локальная и мобильная связь, средства организации человеко-машинного интерфейса в распределенных и встраиваемых системах. Парадигма, которой придерживается фирма MikroElektronika, заключается в организации аппаратно-модульного принципа проектирования, в котором центральное место отводится специальным интегрированным отладочным платам (таких как EasyMx PRO v7 for STM32, Fusion for ARM v8 и многие др.), функциональность которых может неограниченно расширяться дополнительными модулями и устройствами, получившими название Click, подключающихся к фирменной шине MikroBUS™. Единая концепция, в рамках которой удалось охватить широкую номенклатуру микроконтроллеров с различной архитектурой и аппаратной реализацией, позволила создать удобную развивающуюся аппаратно-программную платформу для проведения экспериментальных исследований как в традиционных для разработчиков электронной техники областях, так и в различных смежных направлениях

Программирование устройств системы осуществляется в среде быстрой подготовки программ, внешне напоминающей среду программирования Delphi, с опциональным использованием трех алгоритмических языков: MikroPascal, MikroC и MikroBasic, являющихся расширенными диалектами широко известных языков программирования общетехнического назначения Pascal, C и Basic. Общая функциональность этих языков расширяется за счет использования специализированных библиотек, функции и процедуры которых в большинстве своем направлены на организацию работы аппаратной части системы прототипирования и поддержки протоколов интерфейсов связи.

Следует отметить, что представленные фирмой MikroElektronika языки программирования не являются объектно-ориентированными, что, в известной степени, обусловлено отсутствием необходимости поддержки такой парадигмы программирования при разработке аппаратной части радиоэлектронных устройств. В свою очередь, создание графического интерфейса и иных прикладных систем без поддержки

объектно-ориентированного способа проектирования сопряжено с большими трудностями.

Для развития концепции представления и обработки информации средствами геометрических машин, система прототипирования, разработанная фирмой MikroElektronika, представляет большой интерес, поскольку входные параметры геометрических моделей могут быть легко ассоциированы с сигналами различных физических датчиков и устройств. В свою очередь, это открывает перед геометрическим моделированием новые возможности приложения своих методов и превращает его в средство программирования, расширения функциональных возможностей электронных устройств, приборов и гаджетов, создания специализированного графического интерфейса таких устройств посредством плоских LCD экранов и манипуляторов.

Для проведения экспериментальных исследований в обозначенной области был разработан комплекс функций, запрограммированных на языке MikroPascal. В виду отсутствия в этом языке средств программирования классов и типизированных указателей, в библиотеке реализован метод интерпретации объектно-ориентированного принципа проектирования методами декларативного программирования на основе расширения структур полями, имитирующими записи о классах. Библиотека состоит из функций общего назначения, выполняющих такую интерпретацию и ряд вычислительных действий общего назначения, а также комплекса функций геометрического содержания, выполняющих операции с объектами геометрической природы на проективной плоскости с учетом возможности работы с комплекснознаными данными объектов. Данный подход позволил в полной мере реализовать методы проекционного моделирования пространств произвольных размерностей и разработать их графическую интерпретацию под управлением устройств быстрого прототипирования микроконтроллерных систем (рис. 1).

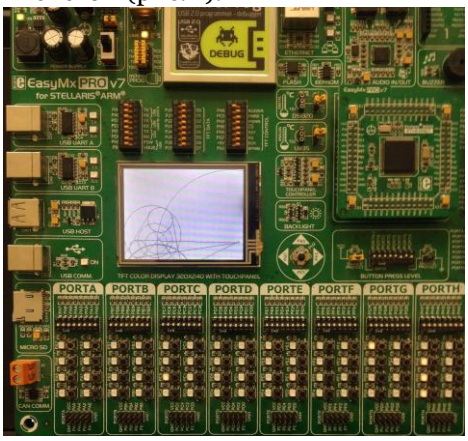

Рис. 1. Результат функционирования геометрической модели на плате разработки EasyMx Pro v7 for Stellaris® Arm® фирмы MikroElektronika

Специально разработанная для этих целей подсистема трансляции проекта системы Симплекс на язык MikroPascal синтезирует программу, выполняющую согласованное обращение к функциям геометрической библиотеки в точном соответствии с алгоритмом функционирования геометрической модели. Полученный текст программы представляет собой текст основного модуля программы, размещаемый в трансляторе системы быстрого прототипирования, который транслируется, компонуется и загружается в аппаратную часть платы разработки микроконтроллерной системы (табл. 2). Дальнейшее функционирование полученной программы могут выполняться автономно и управляться

интерфейсными средствами этой платы.

Таблица 2. Фрагмент программы на языке MikroPascal фирмы MikroElektronika, сгенерированный системой Симплекс для моделирования геометрической конструкции задачи Аполлония

```
var
  Chisl1,Chisl2,Chisl3,Chisl4,Chisl5,Chisl6,Chisl7,Chisl8,Chisl9: DWord;
var

d1,d2,d3,o1,p1,p2,o2,p3,p4,p5,o3,p6,p7,o4,p8,p9,p10,o5,p11,p12,o6,p13,p14,p15,o7,d4
,d5,p16,p17,d6,p18,p19,d7,p20,p21,d8,d9: DWord;
const
   Att_d1: TAtt = (0,'basic',1,1,0,0,0);
   Att_d2: TAtt = (0,'basic',1,1,0,0,0);
   . . . . .
   Att_d9: TAtt = (5,'basic',1,1,0,0,0);
begin
   InitSimplex;
   IY:=0;
   MM_Init();
   TFT_Init_ILI9340_8bit(320, 240);
   TFT_Fill_Screen(cl_white);

   Chisl1:=TOChisl_Create(-53.19,0);
   Chisl2:=TOChisl_Create(-154.6,0);
   Chisl3:=TOChisl_Create(100,0);
   Chisl4:=TOChisl_Create(99,0);
   Chisl5:=TOChisl_Create(152,0);
   Chisl6:=TOChisl_Create(148,0);
   Chisl7:=TOChisl_Create(307.7,0);
   Chisl8:=TOChisl_Create(-82.44,0);
   Chisl9:=TOChisl_Create(47,0);

   B:=EExecD00(Chisl1,Chisl2,Chisl3,d1,1,1,1,Att_d1);
   B:=EExecD00(Chisl4,Chisl5,Chisl6,d2,1,1,1,Att_d2);
   B:=EExecD00(Chisl7,Chisl8,Chisl9,d3,1,1,1,Att_d3);
   B:=EExecO3(d1,d2,o1,p1,p2,1,1,Att_o1,Att_p1,Att_p2);
   B:=EExecO3(d2,d1,o2,p3,p4,1,1,Att_o2,Att_p3,Att_p4);
   B:=EExecP2(o1,o2,p5,1,1,Att_p5);
   B:=EExecO3(d1,d3,o3,p6,p7,1,1,Att_o3,Att_p6,Att_p7);
   B:=EExecO3(d3,d1,o4,p8,p9,1,1,Att_o4,Att_p8,Att_p9);
   B:=EExecP2(o3,o4,p10,1,1,Att_p10);
   B:=EExecO3(d2,d3,o5,p11,p12,1,1,Att_o5,Att_p11,Att_p12);
   B:=EExecO3(d3,d2,o6,p13,p14,1,1,Att_o6,Att_p13,Att_p14);
   B:=EExecP2(o5,o6,p15,1,1,Att_p15);
   B:=EExecO000(p5,p10,p15,o7,1,1,1,Att_o7);
   B:=EExecOK001(d1,d2,d3,d4,1,1,1,Att_d4);
   B:=EExecOK001(d1,d4,o7,d5,1,1,1,Att_d5);
   B:=EExecP3(d1,d5,p16,p17,1,1,Att_p16,Att_p17);
   B:=EExecOK001(d2,d4,o7,d6,1,1,1,Att_d6);
   B:=EExecP3(d2,d6,p18,p19,1,1,Att_p18,Att_p19);
   B:=EExecOK001(d3,d4,o7,d7,1,1,1,Att_d7);
   B:=EExecP3(d3,d7,p20,p21,1,1,Att_p20,Att_p21);
   B:=EExecD4(p17,p18,p20,d8,1,1,1,Att_d8);
   B:=EExecD4(p16,p19,p13,d9,1,1,1,Att_d9);

   TList_Create(List);
```

```
    TList_Add(List,d1);
    TList_Add(List,d2);
    . . . . .
    TList_Add(List,d9);

    DrawList(List);

    TList_Destroy(List);

    Destroy(d1); … Destroy(d9);
end.
```

Следует отметить, что, в отличие от микропроцессорных средств общетехнического назначения, работающих под управлением операционных систем, микроконтроллерные устройства обладают существенно меньшей оперативной памятью и недостаточно развитыми инструментами ее обслуживания. В частности, менеджер динамической памяти, представленный фирмой Mikroelektronika в ее инструментах прототипирования, не позволяет с достаточной эффективностью производить компактификацию памяти, требующей динамического распределения и освобождаемой после выполнения промежуточных вычислений. Это обстоятельство послужило причиной ряда ограничений, накладываемых на сложность реализованных к настоящему времени геометрических моделей. Ограниченными возможностями обладают также встроенные в платы дисплейные модули, скорость вывода изображений на которые не соответствует задачам конструктивного геометрического моделирования. Однако эти ограничения не являются принципиальными и могут быть преодолены в результате реорганизации вычислительных функций и посредством внешнего подключения к средствам прототипирования внешних высокоскоростных средств синтеза и отображения графической информации.

4 Перспективы реализации конструктивных геометрических алгоритмов в HDL-системах

Анализ структуры конструктивных геометрических моделей показывает их высокую предрасположенность к распараллеливанию реализуемых с их помощью вычислений, а также возможность конвейерной обработки данных. В настоящее время система Симплекс выполняет однотипные, но параллельные по своей вычислительной природе операции, в едином цикле, последовательно в соответствии с представленным в [11, 12] алгоритмом выполнения вычислительной работы геометрической машины. На рис. 2. представлена структура упрощенного алгоритма решения задачи Аполлония о сопряжении трех окружностей [7]. Несмотря на свою простоту, она демонстрирует особенности геометрических алгоритмов, которые могут получить высокоэффективную реализацию в аппаратуре, построенной на базе программируемых логических интегральных схем. Каждый представленный на схеме блок представляет собой законченную геометрическую операцию, порождающую набор выходных геометрических объектов из совокупности входных объектов. Алгоритм подразумевает обращение к двадцати двум блокам-модулям семи типов (построение окружности по координатам центра и радиусу; построение касательных к двум окружностям; нахождение точки пересечения прямых, построение прямой по трем колинейным точкам; построение ортогональной окружности; нахождение точек пересечения окружностей; построение окружности по трем точкам). Автоматический анализ структуры алгоритмов позволяет выделить в нем те группы блоков,

которые могут совершать вычислительную работу согласованно и единовременно по мере готовности данных. Такие группы на схеме (рис. 2.) организовались в вертикальные ряды, число которых равно семи. Условно считая время выполнения операций разного типа одинаковым, получаем трехкратное сокращение общего времени вычислений. Разрабатывая архитектуру вычислительного устройства, в котором данные могут синхронно передаваться между представленными слоями, можно обеспечить конвейерную обработку потока данных, при которой скорость вычислений возрастает еще в семь раз. Предложенный подход оказывается особенно эффективным при множественном согласовании параметров геометрических функций [11], когда модули, принадлежащие одному ряду, многократно реплицируются, не нарушая общую структуру алгоритма, что позволяет кратно увеличивать производительность вычислений.

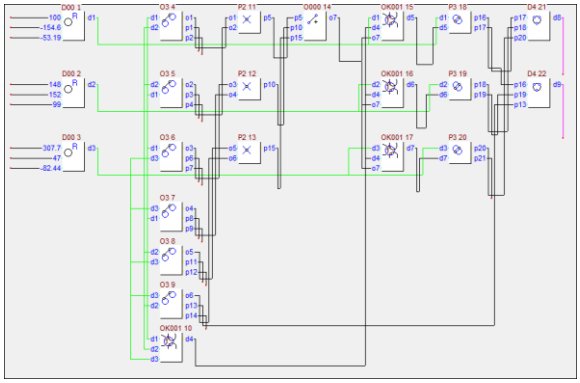

Рис. 2. Структура алгоритма решения задачи Аполлония, допускающая распараллеливание вычислений и их конвейерную организацию

Как было показано ранее, система Симплекс позволяет создавать новые геометрические операции, представляемые в виде отдельных иерархически организованных алгоритмов и использовать их в качестве процедур, вызываемых из других алгоритмов проекта. Возможность пополнения функционального состава системы, безусловно, расширяет области ее применения. Однако используемый для этого процедурный подход непременно приводит к снижению производительности вычислений из-за неизбежного появления дублированных операций, устранение которых при процедурном стиле программирования затруднительно, а в ряде случаев и невозможно. При аппаратной реализации алгоритмов избыточность вычислений недопустима. В связи с этим представляется актуальной задача выполнения логического анализа развернутых структур алгоритмов с целью устранения дублирований и приведения «геометрических подобных» по аналогии с алгебраическими преобразованиями с целью поиска оптимальных аппаратных конструкций по критериям минимизации времени выполнения операций и общего числа вычислительных элементов.

В настоящее время авторами ведется разработка библиотеки общих и геометрических функций, представленных на языке VHDL, для их апробации и корректировки на отладочных платах с ПЛИС фирм Xilinx и Altera .

5 Заключение

Результатом проведенных исследований стала разработка концепции представления конструктивных геометрических моделей в виде

программ на устройствах специализированного назначения.
Разработана методика трансляции и подготовки программ конструктивного геометрического синтеза на отладочных платах фирмы MikroElektronika.
Созданы и отлажены библиотеки геометрических процедур на языках Pascal, MikroPascal, JavaScript, MaxScript.
Осуществляется разработка библиотеки геометрических функций на языке VHDL с целью создания экспериментального образца геометрического процессора

Список используемых источников

1. *Адамар Ж.* Элементарная геометрия. Часть I. Планиметрия. М.: Учпедгиз,1948. —608 с.
2. *Аргунов Б.И., Балк М.Б.* Геометрические построения на плоскости. М.: Учпедгиз,1957. — 268 с.
3. *Бакельман И.Я.* Инверсия. М.: Наука, 1966. — 79 с.
4. *Вальков К.И.* Введение в теорию моделирования [Текст] / К.И. Вальков. — Л.: ЛИСИ. — 1974. — 152 с.
5. *Волков В.Я., Юрков В.Ю., Панчук К.Л., Кайгородцева Н.В.* Курс начертательной геометрии на основе геометрического моделирования: учебник / В.Я.Волков, В.Ю.Юрков, К.Л.Панчук, Н.В.Кайгородцева. Омск: Изд.-во СибАДИ, 2010. — 253 с.
6. *Волошинов Д.В.* Геометрическая лаборатория. Закладываем основы [Текст] // Качество графической подготовки: проблемы, традиции и инновации: Материалы VII международной Интернет-конференции. Февраль - март 2017 г. Пермь, 2017.
7. *Волошинов Д.В.* Конструктивная геометрическая модель четырехмерного пространства как основа для решения задач зонирования и позиционирования при проектировании сетей мобильной связи [Текст] // Труды учебных заведений связи. 2018. Т . 4. No 2. С. 44–60.
8. *Волошинов Д.В.* Конструктивное геометрическое моделирование. Теория, практика, автоматизация: монография [Текст] / Д.В. Волошинов. — Saarbrücken: Lambert Academic Publishing, 2010. — 355 с.
9. *Волошинов Д.В.* О перспективах развития геометрии и ее инструментария [Текст] // Проблемы качества графической подготовки: материалы IV Междунар. интернет-конф.; февраль-март 2014 г. — Пермь, 2014.
10. *Волошинов Д.В.* Симплекс. Свидетельство о регистрации программы для ЭВМ RU 2019619710, 23.07.2019.
11. *Волошинов Д.В.* Технологии применения геометрического инструмента. Избавление от рутины. [Текст] // Проблемы качества графической подготовки: материалы VIII Междунар. интернет-конф.; февраль-март 2019 г. – Пермь, 2019.
12. *Волошинов Д.В.* Технологии применения геометрического инструмента. Логический синтез. [Текст] // Проблемы качества графической подготовки: материалы VIII Междунар. интернет-конф.; февраль-март 2019 г. – Пермь, 2019.
13. *Волошинов Д.В., Соломонов К.Н.* Конструктивное геометрическое моделирование как перспектива преподавания графических дисциплин [Текст] / Д.В. Волошинов // Геометрия и графика. — 2013. — Т. 1. — № 2. — С. 10–13.
14. *Вольберг А.О.* Основные идеи проективной геометрии. М.-Л.:

Учпедгиз, 1949. — 188 с.13.
15. *Гирш А.Г.* Наглядная мнимая геометрия / А.Г. Гирш. М.: Маска, 2008. — 216 с.14.
16. *Глаголев Н.А.* Проективная геометрия. М.: Высшая школа, 1963. — 342 с.15.
17. *Жижилкин И.Д.* Инверсия. М.: Изд-во МЦНМО, 2009. —72 с.
18. *Короткий В.А.* Центральное проецирование двух компланарных коник в две окружности// Проблемы качества графической подготовки Материалы IV международной Интернет-конференции. Февраль–март 2014 г. Пермь, 2014.
19. *Пеклич В.А.* Мнимая начертательная геометрия: учеб. пособие / В.А. Пеклич. — М.: АСВ, 2007. —104 с.
20. *Сальков Н.А.* Свойства циклид Дюпена и их применение. Ч.1 / Н.А.Сальков // Геометрия и графика, 2015. — Т.3.—No1.—С. 16–25. DOI: 10.12737/10454.
21. *Филиппов П.В.* Начертательная геометрия многомерного пространства и ее приложения. Изд. 2-е. М.: ЛЕНАНД, 2016. —282 с.