УДК 681.142:155
УДК 681.142.1.01

ФОРМАЛИЗАЦИЯ ПРОЦЕССА ПРОЕКТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

И.В.Иловайский, Б.А.Сидристый

В работе формулируется подход к вопросам проектирования цифровых устройств на этапе логического синтеза (по терминологии [1]).

Существующая в настоящее время теория синтеза цифровых автоматов и комбинационных схем не используется для реального проектирования больших цифровых устройств. Оно, как правило, ведется интуитивными методами, поскольку разработанные в теории автоматов методы требуют детального и, следовательно, громоздкого задания информации для синтеза, причем в форме, далекой от обычных способов его формулирования. Отмеченные обстоятельства приводят к появлению ошибок уже на этапе записи задания, а в процессе машинного синтеза выдвигают нереальные требования к объему памяти и быстродействию ЦВМ, на которых ведется синтез. Возникает противоречие между принципиальными возможностями алгоритмического синтеза на базе теории автоматов и реальными требованиями к методам проектирования.

Чтобы преодолеть это противоречие и сформулировать методику алгоритмического проектирования цифровых устройств, мы попытаемся проанализировать, как выглядит общая картина процесса проектирования и какими приемами следует пользоваться при

разработке больших систем, и на основе этого предложим формальную модель процесса проектирования цифровых устройств.

Грубо говоря, процесс проектирования заключается в том, что преобразуются некоторые исходные данные (задание на проект) в результирующие (решение). При этом используются какие-то языковые и логические средства, орудия [2].

Задача проектирования в самом общем смысле включает в себя задание на проект и метод его реализации. Задание на проект состоит из описания условий и описания вида, который должен иметь результат проектирования. Формализация этой задачи состоит в том, что мы задаем класс допустимых исходных данных, класс допустимых решений и способ построения некоторого элемента класса результатов по произвольному элементу класса исходных данных. Алгоритмическое проектирование сводится к заданию конкретного элемента класса исходных данных ("задание на проект") и применению к этому "заданию" алгоритма перехода к классу решений.

Таким образом, мы приходим к необходимости задать, описать классы исходных и результирующих данных процесса проектирования устройства, системы. Поскольку мы имеем дело с большим устройством или системой, то вынуждены его задавать, описывать несколькими, не сводящимися друг к другу способами. Система таких способов описания устройства, объекта известна как конфигуратор [3]. Представление объекта конфигуратором делает явным следующие обстоятельства:

1. Исследователь или разработчик имеет дело не с объектом, а с его описаниями.

2. Описания неэквивалентны друг другу.

3. Представление о структуре объекта, системы (из каких "элементов" он состоит) зависит от того, каким описанием пользуются.

4. Объект, устройство задается системой своих описаний.

Описания могут быть классифицированы на группы по тому, какую сторону организации объекта (системы) они задают. Когда говорят об организации объекта (системы), то прежде всего имеют в виду его структуру и функцию [4]. Под структурой обычно понимают вещественную основу организации системы (элементы и связи между ними), под функцией - способ функционирования системы. Структура является статической характеристикой системы и, как правило, не отражает способа функционирования системы. Функция - динамическая характеристика системы и из функции, осуществляемой данной системой, обычно удается извлечь сведения о структуре системы.

Итак, конфигуратор системы, устройства есть набор описаний-функций и описаний-структур. Очевидно, можно выделить соответствующие друг другу описания - функции и описания - структуры. Такие пары будем именовать элементарными конфигураторами системы.

В терминах описаний - проекций конфигуратора процесс проектирования есть построение одних описаний по другим (в частности, одних элементарных конфигураторов по другим), подчиняющийся следующим требованиям:

1. В пределах каждого элементарного конфигуратора для функционального и структурного описаний заданы (определены) классы таких возможных описаний.

2. В пределах элементарного конфигуратора между классами функциональных и структурных описаний (если это возможно) установлена связь, позволяющая по произвольному описанию одного класса получить некоторое описание другого класса.

3. Для каждых двух элементарных конфигураторов между однотипными классами описаний (функциональными, либо структурными) установлена (если это возможно) связь, позволяющая по произвольному описанию одного класса получать некоторое описание другого класса.

Чтобы конкретизировать эту картину, мы должны задать некоторый порядок, систему соотношений между элементарными конфигураторами устройства. Эта система соотношений может быть установлена в рамках блочно-иерархического подхода, который характеризуется тем, что описания (или элементарные конфигураторы) устройства (системы) представляются в виде композиции описаний отдельных подустройств (подсистем). При этом описание всего устройства мы относим к нулевому уровню иерархии, а элементы, из которых оно составляется, - к первому. Каждое из описаний, относящихся к первому уровню иерархии, может быть представлено в виде композиции более мелких элементов второго уровня и т.д., до тех пор, пока не дойдем до описания, составленного

из элементов, принятых простейшими, из которых в конечном счете строится разрабатываемая система.

Задача проектирования описания сложной системы распадается на более простые и обозримые задачи проектирования описаний i-го уровня из описаний $i+1$-го и других более глубоких уровней иерархии. При этом используются только внешние характеристики элементов $i+1$-го и более глубоких уровней. В частности, например, при проектировании цифровых устройств инженер пользуется лишь инструкцией по применению схемных элементов, не интересуясь ни их природой, ни принципом работы.

Каждая такая задача проектирования на i-м уровне иерархии может включать в себя, помимо прочего, и оптимизацию по различным критериям, характерным для данного уровня иерархии. Естественно, что при использовании блочно-иерархического подхода исключается возможность глобальной оптимизации проекта.

Блочно-иерархическим подходом сознательно или бессознательно пользуются, в частности, в процессе блочного программирования больших задач и при разработке цифровых устройств. Техника применения этого подхода при программировании отдельных задач на ЦВМ хорошо описана в работе [5]. Хотя автор и не формулирует явно этот подход в виде общего принципа, тем не менее в работе можно найти описание и рекомендации к использованию почти всех его основных приемов.

Известные формализации процесса проектирования цифровых устройств, направленные на практическую реализацию, также используют элементы блочно-иерархического подхода. Так, например, в основу сквозной системы автоматического проектирования ЦВМ "Проект" [6] положена идея представления описаний проектируемых устройств в языке АЛОС в виде иерархии блоков, минимальные из которых рассматриваются как конечные автоматы, а процесс проектирования в целом состоит из нескольких этапов на разных уровнях.

Система *ALERT* [7] является программной записью инженерного опыта и в своей структуре использует многие элементы этого подхода.

Практика ручного проектирования цифровых устройств неформализованными способами опирается на выработанную в результате многолетнего опыта систему иерархий, материализованную в форме

технической документации – описаний проекта. Виды этих описаний в известной степени регламентированы ГОСТами. Эта документация, во-первых, содержит две группы описаний: функциональные (временные диаграммы) и структурные (схемы); во-вторых, внутри группы схемных описаний существует определенная иерархия (блок-схема, функциональная схема, логическая схема и т.п.). Каждой схеме соответствует своя временная диаграмма. Таким образом, и при ручном проектировании широко используются элементы блочно-иерархического подхода.

Сформулируем основные черты этого подхода.

1. Иерархичность, т.е. в ходе разработки уровень за уровнем степень подробности возрастает и на каждом новом уровне в работу вовлекаются только необходимые данные.

2. Блочность – на каждом, i-м уровне конструируются описания блоков i-го уровня из элементов $i+1$-го и более глубоких уровней.

3. Работа с не до конца определенными объектами (элементами, подблоками); на каждом i-м уровне иерархии описания "элемента" $i+1$-го более глубоких уровней рассматриваются просто как "то, что делает то самое, что от него требуется".

В основе предлагаемого нами алгоритмического подхода к проектированию цифровых устройств лежит иерархически упорядоченная система элементарных конфигураторов, каждый из которых состоит из алгоритмического и схемного описаний. Нами выбрано три элементарных конфигуратора таких, что составляющие их описания близки к используемым в ручном проектировании.

Анализ процесса проектирования позволил вначале вычленить два элементарных конфигуратора [8]: <алгоритм функционирования устройства (АФ), блок-схема устройства (СхБ)> и <микропрограммное описание работы устройства (МПО), функциональная схема устройства (СхФ)>. Входом в конструкторский синтез является принципиальная схема, и поэтому подлежит вычленению конфигуратор <система логических уравнений устройства (СЛУ), принципиальная схема (СхП)>. Однако этот конфигуратор с точки зрения процесса проектирования на логическом этапе несет примерно ту же информацию, что и конфигуратор < СЛУ, логическая схема (СхЛ) >. Этот последний мы и включаем в конфигуратор устройства. Отметим, что СЛУ должна включать в себя временные

переменные.

Таким образом, проектируемое устройство (система) оказывается заданным конфигуратором

$$\mathcal{K} = \begin{cases} K_0 = \langle \{\text{АФ}\}, \{\text{СхБ}\} \rangle \\ K_1 = \langle \{\text{МПО}\}, \{\text{СхФ}\} \rangle \\ K_2 = \langle \{\text{СЛУ}\}, \{\text{СхЛ}\} \rangle \end{cases}$$

В дальнейшем символы $\{\ \}$ и упоминание о классе описаний мы будем для простоты изложения опускать, имея их все время в виду.

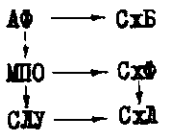
Переходим к построению блочно-иерархической структуры процесса проектирования цифрового устройства, заданного конфигуратором $\mathcal{K}$. Очевидно, что в процессе проектирования мы будем переходить от K_0 к K_1 и далее к K_2. Процесс построения K_{i+1} по K_i, как правило, требует внесения в систему описаний дополнительной информации извне.

Так, при построении МПО по АФ требуются характеристики операторов АФ (в простейшем случае длительности выполнения); при построении СхФ по СхБ нужна библиотека функциональных схем элементов Схф; при построении СЛУ по МПО требуется библиотека СЛУ операторов МПО.

Эта добавочная информация может рассматриваться как конфигуратор подустройств данного устройства. Естественно, в свою очередь, синтез самих подустройств требует аналогичной добавочной информации и т.д. до тех пор, пока мы не приходим к столь элементарным подустройствам, что их синтез может вестись методами теории автоматов и методами синтеза комбинационных схем без обращения к блочно-иерархическому подходу.

Преобразования в пределах каждого элементарного конфигуратора от алгоритмических (А)-к схемным (Сх)-описаниям всегда возможны, поскольку в данном случае А-описания несут информацию о структуре устройства. Проведение обратных преобразований затруднительно, а в ряде случаев (например, от СхБ к АФ) просто невозможно.

Учитывая сделанные замечания, а также то обстоятельство, что результатом проектирования на логическом этапе синтеза должна быть логическая схема устройства и соответствующая ей временная диаграмма, мы приходим к следующей системе разумных преобразований в $\mathcal{K}$:



Из этого представления конфигуратора немедленно следует, что исходное задание на проект должно быть алгоритмом функционирования устройства.

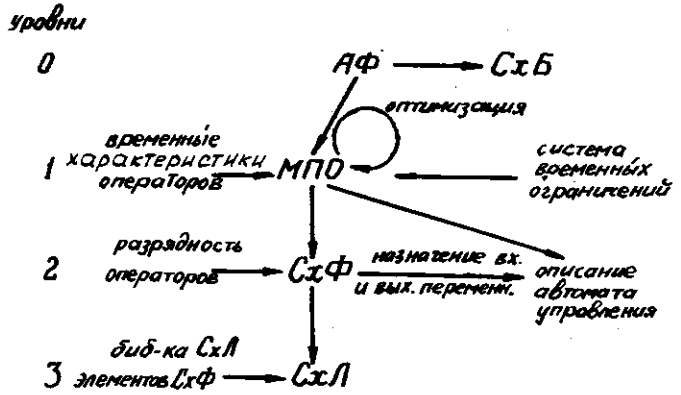
Построение СхЛ через СЛУ является невыгодным, поскольку при этом, во-первых, до самого конца процесса проектирования фактически сохраняется и участвует во всех преобразованиях, вся информация, во-вторых, использование СЛУ затрудняет построение логической схемы в произвольном базисе. Напротив, построение СхЛ по СхФ делается просто, поскольку несложно построить небольшие СхЛ элементов в произвольном базисе, а их подстановка в СхФ проводится тривиальным образом. Дополнительным аргументом в пользу варианта построения СхЛ через СхФ служит то, что методы построения СхФ по МПО и СхЛ по СхФ не зависят от базисов СхЛ и от каких-либо характеристик проектируемого устройства. Но отказ от использования СЛУ приводит к выводу, что из всего СЛУ требуется лишь описание автомата управления (причем в этом случае необязательно в форме СЛУ).

С учетом указанных замечаний мы строим граф иерархии процесса проектирования (см. рисунок, стр. 80).

Поскольку установлена единственность порождения описаний в процессе проектирования, формализмы и алгоритмы могут носить специализированный характер и учитывать особенности конкретных преобразований.

То обстоятельство, что А-описания несут вначале всю информацию об устройстве, приводит к необходимости использования в задачах построения А-описаний более богатых формализмов нежели в задачах построения Сх-описаний. Поэтому ниже излагаются две системы формализмов: для синтеза А-описаний и для синтеза Сх-описаний устройства.

Формальная модель задачи проектирования А-описаний устройства. Модель включает в себя формализмы АФ, МПО и АУ описаний,



соответствующие алгоритмы композиции (перехода от описания к описанию) и формулировку задачи оптимизации на I-м уровне. Для задания исходного в системе проектирования (см. рисунок) А-описания (АФ) можно использовать любой формальный язык. Однако если при разработке конкретных систем проектирования выбор языка диктуется возможностями входных устройств и требованием обеспечить естественность и удобство описания алгоритма для человека, при общетеоретической постановке задачи и в процессе описания самой задачи синтеза на первый план выдвигается требование выявить существенные черты А-описания объекта. Это может быть выполнено лишь при использовании достаточно абстрактных языковых средств, независимых от технических факторов реализации процесса проектирования. Таким требованиям удовлетворяет язык, основанный на понятиях отношения и функции.

Алгоритм функционирования устройства представляется совокупностью отношений между его элементами (операторами, переменными и функциями) [9]. Основными в этой совокупности являются следующие отношения:

непосредственного следования операторов в записи алгоритма, включающее в себя и следование по меткам операторов условных переходов (это отношение задает граф-схему алгоритма);

принадлежности входных переменных операторам;

принадлежности выходных переменных операторам;

принадлежности функций операторам (здесь имеется в виду преобразование, реализуемое оператором);

вхождения меток в операторы условного и безусловного перехода;

эквивалентности типа операторов (условные, безусловные, и т.д.).

Микропрограммное описание есть алгоритм функционирования, дополненный информацией о распределении операторов по микропрограммам, их длительностях, временах и моментах начала выполнения, схемной интерпретации элементов операторов (функций и переменных), т.е. о том, какие свойства должны быть у схем, реализующих эти функции и переменные. Отметим, что здесь речь идет не о том, как выглядят эти схемы, а об их обобщенных характеристиках: времени срабатывания, существовании памяти, принадлежности к потенциальной, импульсно-потенциальной и т.п. системам элементов. Вся информация представляется отношениями и числовыми функциями, определенными на множестве объектов алгоритма [9].

Процесс получения микропрограммного описания слагается из решения задач на двух уровнях иерархии.

На нулевом уровне вначале получается блок-схема исходного алгоритма (блоками являются такие участки алгоритма, которые затем сопоставляются микропрограммам, т.е. участки без контуров). Блок-схема задается отношениями принадлежности операторов блокам и следования блоков. Критерии разбиения алгоритма на блоки зависят от принятых принципов управления процессами обработки информации в проектируемом устройстве (системе). В частности, при проектировании устройств универсальных цифровых вычислительных машин можно использовать следующие два критерия: обязательное отсутствие циклов (контуров) внутри блоков и минимум информационных связей между блоками в множестве решений, удовлетворяющих первому критерию.

На первом уровне решаются задачи схемной интерпретации переменных и функций операторов для каждого блока в отдельности и распределения выполнения операторов внутри блоков. Иными словами, на этом уровне проектируются отдельные элементы структу-

ры более высокого уровня (микропрограммы).

Проектирование микропрограммы сопровождается решением задачи оптимизации. Если задана схемная интерпретация объектов алгоритма, задача сводится к оптимизации по времени, т.е. к такому распределению операторов внутри микропрограмм,которое минимизирует их длительность. Известны эвристические приемы ее решения при некоторых ограничениях [10].

В случае, когда каждому объекту алгоритма соответствует множество возможных схемных интерпретаций, возникает более общая задача оптимизации по времени и оборудованию, которая отнюдь не является тривиальной, поскольку, во-первых, при заданных множествах возможных схемных интепретаций в зависимости от того, как сформулирован критерий оптимальности, будут получаться весьма различные решения, во-вторых, даже при простых критериях оптимальности неизвестно общее ее решение.

После проведения для всех микропрограмм схемной интерпретации и получения их временных диаграмм процесс проектирования возвращается на нулевой уровень, и решается задача совместимости схемных интерпретаций объектов в различных микропрограммах и построения временной диаграммы всего устройства совмещением во времени микропрограмм. В зависимости от критерия оптимальности возможны различные решения этой задачи оптимизации по времени и оборудованию на нулевом уровне иерархии процесса проектирования. В частности, при предположении о полной несовместимости микропрограмм во времени решается более обозримая задача совместимости оборудования (объектов) различных микропрограмм. Если же вместе с алгоритмом была задана схемная интерпретация, микропрограммы выполняются строго последовательно, необходимость в повторном переходе на нулевой уровень иерархии отсутствует.

Формальная модель задачи проектирования Сх-описаний устройства. Модель включает в себя формализмы АФ, МПО, СхФ и СхЛ описаний и соответствующие алгоритмы композиции (перехода)[11].

В этой модели для задания алгоритмов функционирования применены такие алгоритмы, определяемые граф-схемами, у которых операторы и распознаватели суть системы логических уравнений (СЛУ). Временные характеристики операторов учитываются через

множество T управляющих переменных СЛУ. Задание на этом множестве отношения порядка фиксирует микропрограммное описание (см. формальную модель получения А-описаний). Конкретный алгоритм (МПО) может быть изображен ориентированным графом, вершины которого введены операторами (микрокомандами), а дуги-условиями перехода.

Для уточнения понятия "схема" вводится ориентированная сеть (орсеть). Орсеть есть пара $\langle W, \mathcal{E} \rangle$ из множества вершин W и множества $\mathcal{E}$ упорядоченных наборов (E_{i1} , E_{i2}),составленных из вершин. Если сопоставить элементы схемы наборам,входы и выходы элементов - компонентам наборов, а пучки связей - вершинам, получим описание схемы в виде орсети.

Для орсетей определяется операция суперпозиции (подстановки более подробной сети в элемент).

Каждому набору орсети (элементу) сопоставляется СЛУ так, что переменные СЛУ сопоставляются соответствующим полюсам элемента. При этом вся орсеть сопоставляется некоторой совокупной СЛУ.

Итак, СЛУ оказывается посредником, связывающим синтаксическую структуру алгоритмов с орсетями. На этой основе устанавливается непосредственная система соответствий между алгоритмами и схемами.

Алгоритму любого устройства соответствует орсеть - обобщенная функциональная схема. Отображению,реализуемому алгоритмом, в этой схеме соответствует обобщенное операционное устройство, обобщенному предикату алгоритма - обобщенное устройство управления, отношению порядка на множестве T- автомат управления.

Заметим, что поскольку отношение порядка на T влияет лишь на структуру автомата управления, одна и та же схема будет соответствовать целому набору алгоритмов при условии, что эти алгоритмы не отличаются наборами операторов, переменных и вхождениями переменных в операторы. Эта особенность формальной модели фиксирует отмеченный нами факт большей информативности А-описаний по сравнению со Сх-описаниями.

Синтез функциональной схемы идет следующим образом. По множеству операторов алгоритма строится множество элементов схемы.По входным и выходным переменным операторов вводятся по-

люса элементов. В силу того, что переменная может входить в несколько операторов, у разных элементов будут одноименные полюса. Они объединяются в пучок связей, узел (вершину орсети).

Подстановка логических сетей в элементы функциональной схемы дает логическую сеть, являющуюся логической схемой устройства.

Итак, мы в общих чертах рассмотрели процесс проектирования цифровых устройств и выделили формальные объекты, фиксирующие на различных уровнях иерархии классы возможных исходных и результирующих данных. Все эти объекты: отношения следования и отношения зависимости между элементами алгоритма, характеристики элементов алгоритма и микропрограммного описания, алгоритмы, определяемые граф-схемами, орсети над каталогами СФАЛ -могут быть представлены целочисленными функциями и n- арными отношениями. Процессы преобразования одних формальных описаний в другие при этом сведутся к построению одних отношений (функций) по другим.

Для описания алгоритмов проектирования, которые могли бы служить инвариантной записью процесса проектирования цифровых устройств для разных его программных (технических и машинных) реализаций, необходим язык, построенный на отношениях и функциях, позволяющий строить одни из них по другим.

В распространенных языках программирования нет средств , учитывающих специфику работы с отношениями. В недавно появившихся новых языках [I2] введены такие средства. Однако недостатком этих языков является то, что отношения в них рассматриваются как неделимые элементарные объекты. Человек же, осмысливая и формализуя задачу, следит не столько за отношениями как таковыми, сколько интересуется, главным образом, объектами задачи, которые вступают друг с другом в те или иные отношения. Возникает противоречие между естественной логикой формулирования задач и логикой построения алгоритмических языков. Известно, что подобное противоречие частично преодолевается в математике средствами языка узкого исчисления предикатов, позволяющего записывать одни логические соотношения через другие.

Несмотря на то, что цели использования языка исчисления предикатов в математической логике далеки от целей программирования, оказалось возможным, введя дополнительные средства,определить язык, удобный для работы с отношениями .

Язык не является алгоритмическим, он имеет некоторые особенности исчислений, описание на этом языке представляет собой систему определений одних отношений (функций), определенных на множестве имен объектов задачи, через другие.

Язык использует средства узкого исчисления предикатов со следующими тремя дополнительными способами введения новых объектов:

I. Определение через равенство.

2. Определение по условиям.

3. Определение итерацией.

Для удобства введен ряд операторов.

Поскольку в настоящее время транслятор с этого языка не вступил в действие, этот язык имеет смысл использовать как язык публикаций систем алгоритмов преобразования информации, что важно для фиксации способов решения тех или иных сложных задач в строгой и обозримой форме.

По нашему мнению, изложенный подход может быть полезен и при разработках систем автоматического программирования, и при проектировании неинформационных сложных устройств

Литература

I. ГЛУШКОВ В.М. Проблемы синтеза цифровых автоматов.В кн. "Теория конечных и вероятностных автоматов" Тр. международ. симп.) М., Наука, I965.

2. РЕЙТМАН У.Р. Познание и мышление.Моделирование на уровне информационных процессов ."Мир", I968.

3. ЛЕФЕВР В.А. Конфликтующие структуры ."Высная школа" , I967.

4. АНТОМОНОВ Ю.Г. Системы. Сложность. Динамика . "Наукова думка", Киев, I969.

5. БРАНДОН Р. Организация работы в вычислительном центре ."Статистика", I970.

6. ГЛУШКОВ В.М., КАПИТОНОВА Ю.В., ЛЕТИЧЕВСКИЙ А.А. Математическое обеспечение автоматизированной системы проектирования вычислительных машин и систем (Проект).- "Кибернетика", I970, №4.

7. FRIEDMAN T.D.,YANG S.C. Methods used in an automated logic generater (ALERT). IEEE Trans.Comput.C-18 July 1969.

8. ИЛОВАЙСКИЙ И.В. О некоторых однозначных преобразованиях описаний процессора ЦВМ.-"Всесоюзная межвуз.конферен.по алгоритмическим методам проектирования цифровых систем", Тезисы докладов, Ленинград, 1969.

9. СИДРИСТЫЙ Б.А. Эквивалентный переход от описания ЦВМ на Ф-языке к микропрограммному описанию.-"Вычислительные системы", Новосибирск,"Наука" СО, 1969, вып. 34.

10. СИДРИСТЫЙ Б.А. Формальный метод построения временных диаграмм работы вычислительных устройств. - "Материалы к II-й респ.конф. молодых исследователей по кибернетике", Киев,1968.

11. ИЛОВАЙСКИЙ И.В. О применении теории сетей к синтезу функциональных и логических схем однопроцессорных УЦВМ. -"Проектирование блоков и узлов вычислительных машин" (Материалы I Всесоюзной конференции по автоматизированным системам проектирования вычислительных машин), Киев, 1970.

12. NICOLAS V.,FINDLER, WILEY R.,Mc KINZIL. On a new tool in artificial intelligence research. Proc. of the International Joint Conf. on Artificial Intelligence, May 1969.

Поступила в редакцию
5.5.1971.