АВТОМАТНЫЕ МЕТОДЫ СИНТЕЗА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

В.С. Любченко

В программировании сложилась парадоксальная ситуация: одни довольны возможностями многопоточности, многоядерности, а другие к этому же предъявляют претензии. Но и те и другие, по сути, в растерянности. Первые не знают, какую научную базу подвести под то, что есть, а вторые – под то, что будет.

Но есть теория, на которую программистам необходимо обратить внимание. Далее на примере разработки алгоритма управления роботом мы покажем, как структурная теория автоматов, которая добросовестно служит электронщикам, может стать основой для, казалось бы, совсем иного рода деятельности – разработки [параллельных] программ.

1. Постановка задачи

Пусть имеется задание для системы управления роботизированным шлифовальнымавтоматом [1]:

1. По прибытии заготовка зажимается захватом;
2. После того, как сработал захват, шпиндель переводится из парковочного положения в рабочее (влево);
3. После того, как шпиндель перешел в рабочее положение, включается фреза;
4. После включения фрезы осуществляется рабочая (плавная) подача влево до крайнего положения (конец операции);
5. По завершении операции шпиндель подается вправо обратно до рабочего положения;
6. После того, как шпиндель занял рабочее положение, позиционер поворачивает стол на четверть оборота;
7. После фиксации стола осуществляется очередная операция до тех пор, пока стол не завершит один оборот;
8. После завершения одного оборота шпиндель паркуется, захват разжимается, посылается сигнал готовности детали;
9. Перед парковкой выключить фрезу, дождаться останова.

И программист, и электронщик могут начать с того, что нарисуют блок-схему. Только для ее реализации у второго, в отличие от первого, в распоряжении структурная теория автоматов. Наша задача продемонстрировать, как применить ее же, но для проектирования параллельных программ. Включая и их автоматическое распараллеливание.

2. Варианты решения

На рис.1 приведен граф, определяющий поведение робота (его можнополучить формально из упомянутой блок-схемы). Он рассматривается, как первый подход к реализации систем, когда алгоритм представлен в форме единственного графа/автомата, пошагово определяющего функционирование системы. Т.е. речь идет об обычной последовательной программе.

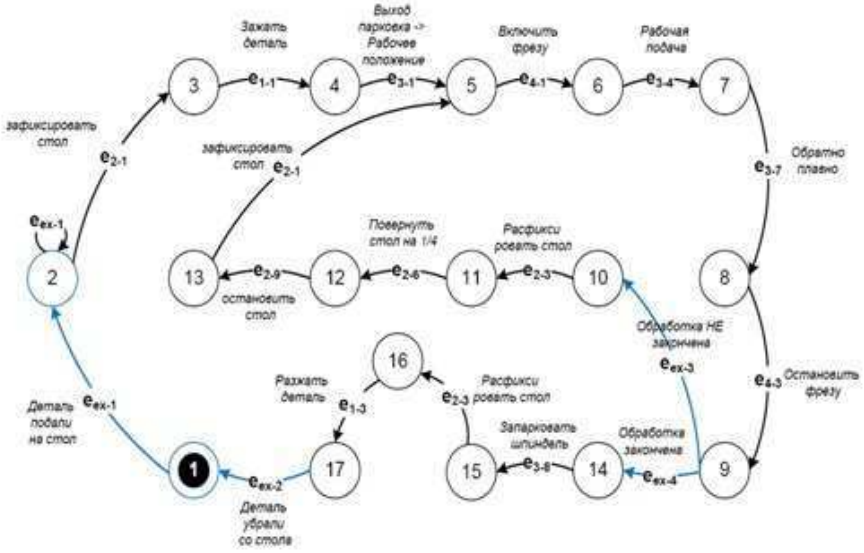

Рис.1. Поведение станка на языке директивных спецификаций.

Другой подход, назовем его структурно-автоматным, включает следующие этапы:

1. определение/выделение множества объектов (блоков, компонентов) системы;
2. формирование для каждого из них интерфейса и протокола взаимодействия с любыми другими объектами системы;
3. создание схемы связей между объектами;
4. задание алгоритмов работы объектов;
5. реализация отдельных блоков и системы в целом

Далее, реализуя поведение шлифовального автомата, мы представим его в деталях.

3. Канонический синтез параллельных программ

Но сначала рассмотрим метод, фактически связующий упомянутые выше два подхода, кстати, фактически решающий задачу автоматического распараллеливания последовательных программ в общем случае. Это канонический метод синтеза цифровых автоматов, когда последовательный автомат реализуется схемой из заданных автоматов, называемых элементарными. Данная схема состоит из комбинационной (КЧ) и запоминающей (ЗЧ) частей, где КЧ представляет логическую схему, реализующей функции возбуждения элементов памяти, входящих в ЗЧ. В качестве элементов памяти, кодирующих состояния автомата, выступают автоматы Мура с полной системой выходов [2]. Говорят, что система элементарных автоматов структурно полна, если позволяет реализовать произвольный автомат.

Если электронщик реализует схему, используя набор микросхем, то и программист может заранее создать структурно полную систему программных элементов/блоков. Пусть это будут логические элементы И, ИЛИ, НЕ и элемент памяти – триггер с раздельными входами. На рис.2 показан фрагмент структурной схемы из подобных блоков, которая реализует автомат на рис.1. Состояния автомата закодированы двоичным кодом: 1 – 00000, 2 – 00010, 3 – 00011, 4 – 00100, 5 – 00101, 6 – 00110, 7 – 00111, 8 – 01000, 9 – 01001, 10 – 01010, 11 – 01011, 12 – 01100, 13 – 01101, 14 – 01110, 15 – 01111, 16 – 10000, 17 – 10001.

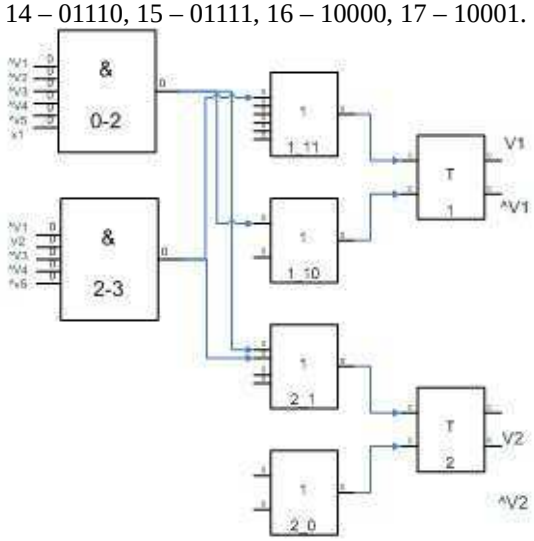

Рис.2. Фрагмент логической схемы для автомата на рис.1.

Функции возбуждения схемы будут следующими (конъюнкции, представляющие состояние элементов памяти, заменены кодами состояний):

$$t1_0 = 01011 \vee 01111 \vee 10001 \vee 00011 \vee 00101 \vee 00111 \vee 01001\,\overline{x2} \vee 01001\,x2;$$
$$t1_1 = 01010 \vee 01100 \vee 01110 \vee 10000 \vee 00010 \vee 00100 \vee 00110 \vee 01000;$$
$$t2_0 = 01011 \vee 01111 \vee 00011 \vee 00111;$$
$$t2_1 = 00000\,x1 \vee 00010 \vee 00101 \vee 01001\,\overline{x2} \vee 001001\,x2;$$
$$t3_0 = 01111 \vee 00111;$$
$$t3_1 = 01011 \vee 00011 \vee 01001\,x2;$$
$$t4_0 = 01101 \vee 01111;$$
$$t4_1 = 00111;$$
$$t5_0 = 10001;$$
$$t5_1 = 01111;$$

Рис.3. Функции возбуждения элементов памяти

Здесь tN_0, tN_1 – функции установки N-го элемента памяти в нулевое или единичное состояние.

При программной реализации схемы число параллельных процессов зависит от числа логических элементов. В схеме число элементов И определяется числом переходов автомата – их восемнадцать, число элементов ИЛИ равно числу входов элементов памяти – десяти (см. также рис.3), число элементов памяти - пяти. В результате последовательный автомат можно представить суммой тридцати трех параллельных процессов.

Таким образом, метод канонического синтеза может рассматриваться не только, как метод реализации автоматов, но и как универсальный метод распараллеливания любых программ. Выбирая элементарные автоматы, метод кодирования, можно получить необходимые качества распараллеливания.

4. Структурно-автоматный подход

Представленный канонический метод особенно привлекателен для сред, реализующих принципы визуального программирования, где программы создаются из готовых блоков, как обычная схема из микросхем. Но упомянутые тридцать три процесса, реализующие всего один автомат, даже для многоядерных систем серьезная «ноша». Но можно поступить по-другому…

Легко видеть, что множество объектов модели робота может определяться числом компонентов, из которых он состоит. Назовем их: Захват, Шпиндель, Фреза и Столик. Интерфейс блоков – множество входных/выходных каналов/сигналов объекта, где по входным каналам поступает информация, по выходным – выдаются сигналы во внешнюю среду. К выходным сигналам будем относить и информацию о внутреннем состоянии объекта.

4.1 Этап выделения структурных блоков

Рассмотрим блок Захват. Ему необходим канал, по которому поступает информация о наличии детали, и канал для определения момента, когда необходимо ее освободить. Нужен также выходной канал, который индицирует состояние зажима детали, что соответствует сигналу готовности детали.

Блоку Шпиндель необходим входной сигнал о начале работы. По нему он перейдет в рабочее состояние. Необходимы сигналы о работе Фрезы и состоянии Столика. Во внешнюю среду он должен выдавать сигналы о своем состоянии: парковка, рабочее положение, завершение операции, переход к состоянию парковки.

Блок Фреза выходными состояниями должен отражать состояние фрезы, а два входных канала будут определять моменты включения/отключения фрезы.

Для Столика потребуются два входных сигнала - начало работы и сигнал к повороту стола. Внешней же среде важна информация о его состоянии. Нужен и выходной сигнал о завершении полного цикла поворотов.

4.2 Создание схемы системы

Приведенной выше информации достаточно для создания схемы связей блоков. Она приведена на рис.4. Из нее следует, что информация о захвате детали сигнализирует Шпинделю о переходе из парковочного состояния в рабочее.

Выход блока Шпиндель, индицирующий переход в рабочее состояние, служит сигналом включения фрезы блоком Фреза. Состояние включения фрезы приводит к выполнению шпинделем рабочей операции. Стол, получая сигналы от блока Шпиндель, выполнит свои функции и т. д. и т. п.

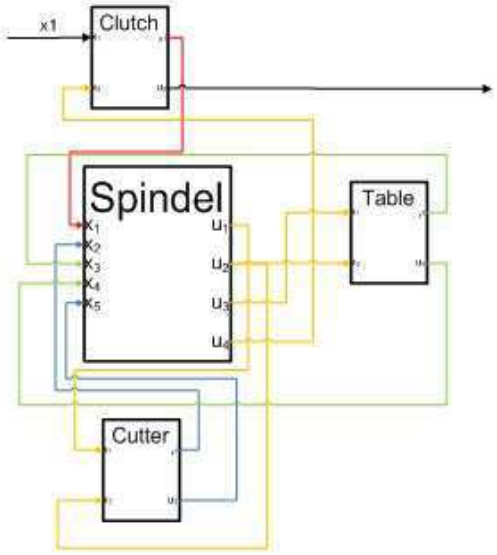

Рис.4 Схема соединения блоков робота

Соединением готовых блоков заканчивается создание системы управления. Например, для случая обычной электронной схемы это может быть макетирование. Программисту блоки предстоит реализовать в виде программ. А для этого необходимо создать их алгоритмические модели и только потом - программный код.

4.3 Алгоритмические модели блоков

Алгоритмические модели блоков Захват, Шпиндель, Фреза и Столик представлены на рис.5-8.

Рис.5 Автоматная модель блока Захват.

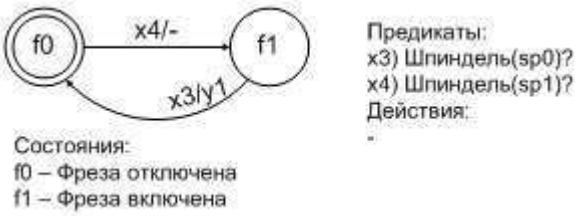

Рис.6 Автоматная модель блока Фреза.

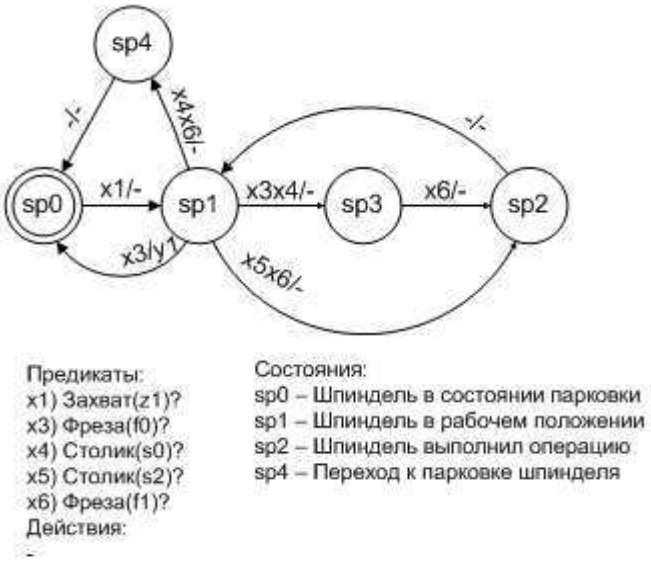

Рис.7 Автоматная модель блока Шпиндель.

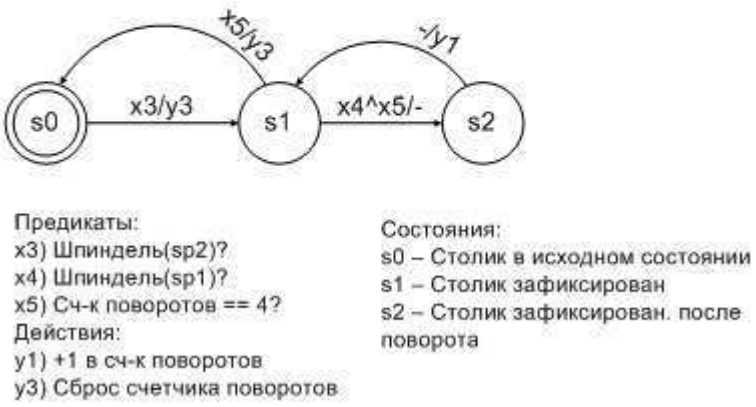

Рис.8 Автоматная модель блока Столик.

Модели достаточно просты и каких-то особых пояснений не требуют.

4.4 Сетевая автоматная модель робота

Перейдем к понятию сети - множеству взаимосвязанных автоматов, работающих в едином дискретном времени. Вид сетевой модели робота показан на рис.9. На ней стрелками и «полочками» отражены связи между автоматами. Стрелка от состояния к полочке означает истинность соответствующего входного сигнала.

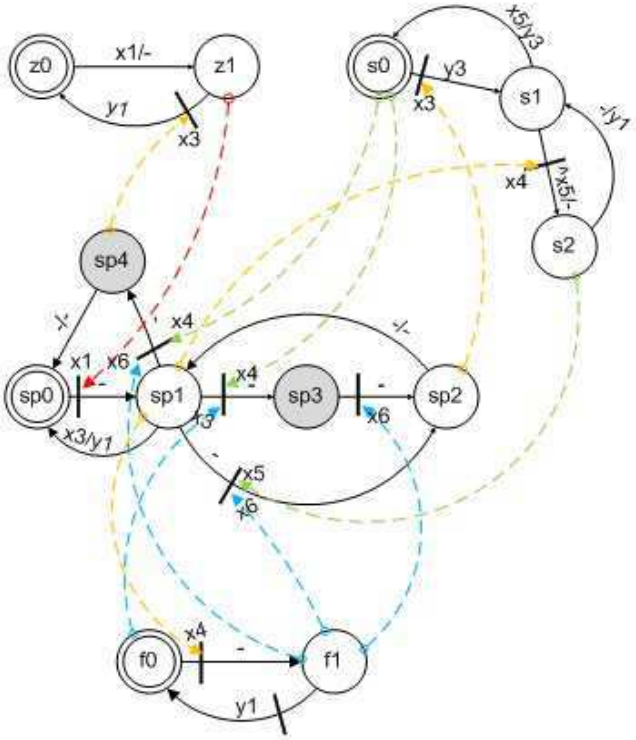

Рис.9. Сетевая автоматная модель робота.

Созданным автоматным моделям соответствует четыре параллельных процесса. Они создадут явно меньше накладных расходов, чем упомянутые тридцать три. Но будет ли автоматная сеть соответствовать заданию?

5. Тестирование автоматных сетей

Автомат на рис.1 и схема, созданная методом канонического синтеза, в явно форме определяют алгоритм работы. В случае структурно-автоматного подхода эту последовательность «домысливает» сеть, т. к. разработчик только создает алгоритмы блоков и определяет связи между ними. Выполняет ли сеть, поставленную задачу, можно узнать: 1) протестировав ее работу или 2) построив эквивалентный автомат.

Протокол тестирования работы сети показан на рис.10. Из него следует, что сеть, включив фрезу, выключает ее при завершении работы, а автомат на рис.1 делает это при каждом повороте столика.

ном.	Время	Захват	Шпиндель	Фреза	Столик
0	0.4998	z1	sp1	f0	s0
1	0.9998	z1	sp3	f1	s0
2	1.4998	z1	sp2	f1	s0
3	1.9998	z1	sp1	f1	s1
4	2.4998	z1	sp1	f1	s2
5	2.9998	z1	sp2	f1	s1
6	3.4998	z1	sp1	f1	s1
7	3.9998	z1	sp1	f1	s2
8	4.4998	z1	sp2	f1	s1
9	4.9998	z1	sp1	f1	s1
10	5.4998	z1	sp1	f1	s2
11	5.9998	z1	sp2	f1	s1
12	6.4998	z1	sp1	f1	s1
13	6.9998	z1	sp1	f1	s2
14	7.4998	z1	sp2	f1	s1
15	7.9998	z1	sp1	f1	s0
16	8.4998	z1	sp4	f1	s0
17	8.9998	z0	sp0	f1	s0

Рис.10.Протокол работы сети на рис.9.

Надо ли выключать фрезу на каждом шаге обработки? Если надо, то для этого, возможно, придется изменить не только алгоритмы блоков, но и связи между ними. Но мы сделаем по-другому…

6. Декомпозиция последовательных процессов

При структурно-автоматном подходе мы проектируем сеть, поведение которой, строго говоря, может отличаться от необходимого. Его мы можем рассчитать формально или протестировав работу сети. Но и то и другое все же дополнительная и, порой, непростая работа. С целью демонстрации, что сеть может функционировать в точности соответствии с заранее заданным алгоритмом, рассмотрим «родственный» каноническому методу формальный метод проектирования систем путем разложения последовательной системы на множество параллельных компонент.

Как и при каноническом синтезе, множество состояний исходного автомата кодируется состояниями компонентных автоматов (только на этом этапе они фактически не известны). В нашем случае это состояния автоматов, имеющих те же имена – Захват, Шпиндель, Фреза и Столик. Исходя из предыдущего опыта (см. рис.5-8), логики работы компонент робота и т. п. , определим множества состояний компонентных автоматов. Заметим, что в процессе декомпозиции они могут быть изменены.

На рис.11 приведен размеченный граф переходов. Первая цифра в коде состояния означает текущее состояние автомата Захват, вторая – автомата Шпиндель, третья – Фрезы, четвертая – Столика.

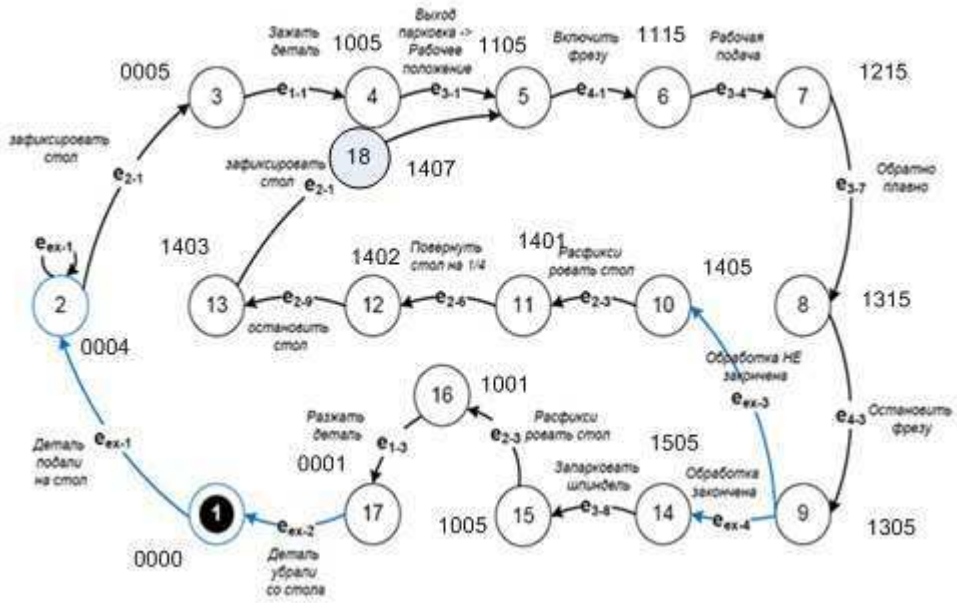

Рис.11. Размеченный граф переходов.

Декомпозиция приводит к автоматам, показанным на рис.12. На рис.13 приведена схема их соединения. Протокол тестирования приведен на рис.14. На протоколе цифры справа соответствуют кодам состояний. Видно, что декомпозиция проведена правильно, и сеть на рис.12 функционирует в точности, как и автомат на рис.11 (см. также рис.1.).

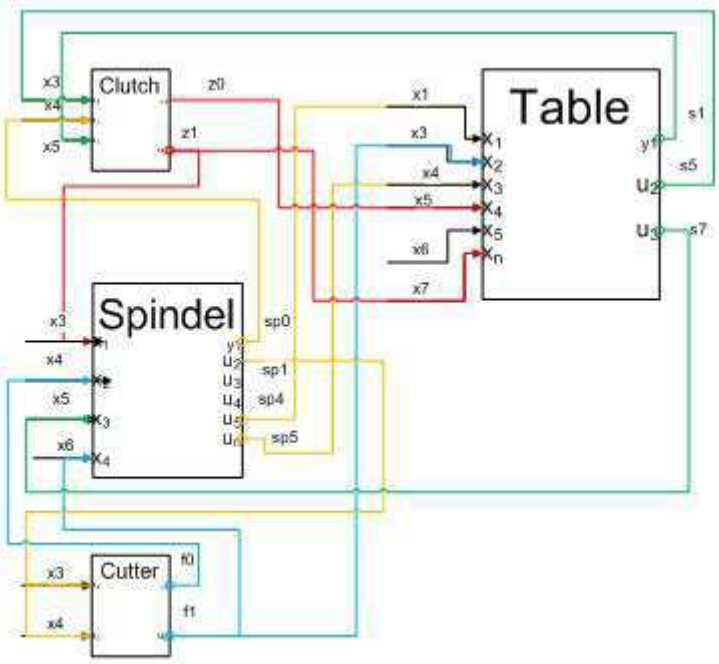

Рис.12. Автоматная сеть, эквивалентная автомату на рис.11

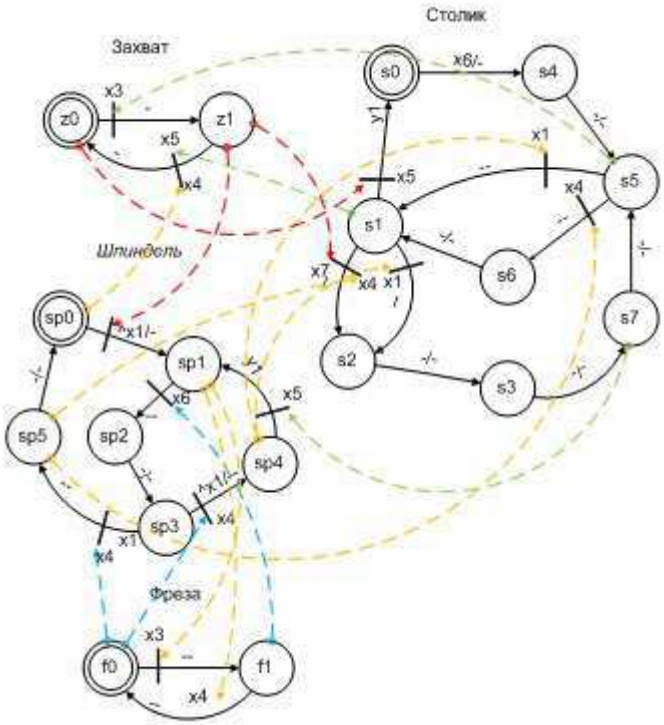

Рис.13. Схема соединения автоматов автоматной сети на рис.12

ном.	Время	Захват	Шпиндель	Фреза	Столик
0	0.0937	z0	sp0	f0	s4 0004
1	0.1933	z0	sp0	f0	s5 0005
2	0.2936	z1	sp0	f0	s5 1005
3	0.3936	z1	sp1	f0	s5 1105
4	0.4936	z1	sp1	f1	s5 1115
5	0.5935	z1	sp2	f1	s5 1215
6	0.6934	z1	sp3	f1	s5 1315
7	0.7936	z1	sp3	f0	s5 1305
8	0.8938	z1	sp4	f0	s5 1405
9	0.9935	z1	sp4	f0	s1 1401
10	1.0934	z1	sp4	f0	s2 1402
11	1.1934	z1	sp4	f0	s3 1403
12	1.2933	z1	sp4	f0	s7 1407
13	1.3934	z1	sp1	f0	s5 1105
14	1.4934	z1	sp1	f1	s5 1115
15	1.5934	z1	sp2	f1	s5 1215
16	1.6936	z1	sp3	f1	s5 1315
17	1.7934	z1	sp3	f0	s5 1305
18	1.8934	z1	sp4	f0	s5 1405
19	1.9934	z1	sp4	f0	s1 1401
20	2.0934	z1	sp4	f0	s2 1402
21	2.1935	z1	sp4	f0	s3 1403
22	2.2934	z1	sp4	f0	s7 1407
...					
37	3.7935	z1	sp3	f0	s5 1305
38	3.8934	z1	sp4	f0	s5 1405
39	3.9934	z1	sp4	f0	s1 1401
40	4.0935	z1	sp4	f0	s2 1402
41	4.1935	z1	sp4	f0	s3 1403
42	4.2934	z1	sp4	f0	s7 1407
43	4.3935	z1	sp1	f0	s5 1105
44	4.4934	z1	sp1	f1	s5 1115
45	4.5934	z1	sp2	f1	s5 1215
46	4.6934	z1	sp3	f1	s5 1315
47	4.7934	z1	sp3	f0	s5 1305
48	4.8934	z1	sp5	f0	s5 1505
49	4.9934	z1	sp0	f0	s6 1006
50	5.0938	z1	sp0	f0	s1 1001
51	5.1935	z0	sp0	f0	s1 0001
52	5.2934	z0	sp0	f0	s0 0000

Рис.14. Протокол работы автоматной сети рис.12

7. Достигаемые цели

Рассматривая автоматно-структурную реализацию, мы считали, что общего алгоритма нет (есть задание). В этом случае автомат на рис.1 — пример одного из вариантов эквивалентного автомата работы сети. Если алгоритм работы сети нужно изменить, то необходимо изменить компонентные автоматы и связи между ними, добиваясь соответствия работы сети поставленному заданию.

Итак, актуальная проблема автоматического распараллеливания программ фактически решена в рамках структурной теории автоматов. Декомпозиция востребована на практике и по многим другим причинам. Но если автомат на рис.1 нас устраивает, то, безусловно, проще его реализовать напрямую и на том сдать работу заказчику. В рамках декомпозиции отдельной строкой можно рассматривать канонический метод структурного синтеза автоматов: с его помощью можно создавать любые алгоритмы из готовых блоков, фактически не прибегая к встроенному языку программирования в визуальном программировании.

Заказчик хочет большего, а потому в целом выгоднее разбить систему на блоки и изменять тот, к которому претензии, или который нужно просто заменить. Кроме того, решение проблемы сложности, повышение надежности, гибкость, модульность и множество других качеств, факторов и причин, в силу которых переход на модульное/структурное проектирование также оправдан.

Уход от «моноблочного» последовательного программирования предрешен самим развитием технологий, как программирования, так и аппаратных средств.

8. Заключение

Параллельное программирование ассоциируется с проблемами. Может, потому, что мы не до конца понимаем «почему» и уж совсем далеки от знания того «как» параллельно программировать? Все вышесказанное, по сути, ответ на вопрос «почему» надо и демонстрация того «как» проектировать параллельные программы. В том числе и как, используя теорию автоматов, автоматически распараллеливать в общем случае.

Можно привести множество аргументов в пользу параллелизма, но главное, что с ним программирование выходит на иной, можно даже сказать, интеллектуальный уровень. Но «интеллект машин» должен быть контролируемым. Без четкой параллельной вычислительной модели и теории это невозможно.

Выше мы показали, какой может быть теоретически верная и практически эффективная универсальная модель параллелизма. Рассмотрели, как ее можно использовать, и как, если необходимо, контролировать. Все это более чем реально, т. к. опробовано в области проектирования цифровых схем и может быть, как показано выше, адаптировано для проектирования параллельных программных систем.

В рамках представленной автоматной технологии проектирования «машинная мысль» остается свободной, параллельной и, главное, без признаков … «многопоточной анархии» [3].

ЛИТЕРАТУРА:

1. Амбарцумян А. А. Дискретно-событийное моделирование в технических системах. Пленарный доклад на УКИ-12. 2012г.
2. Глушков В. М. Синтез цифровых автоматов. М. : Физматгиз, 1962.

Любченко В. С. , Ломакин Р. Л. , Перфилов С. А. Система управления прессом «АСУ ТП - Пресс» на базе автоматной модели параллельных процессов // Свидетельство о государственной регистрации программы для ЭВМ №2012615341. Зарегистрировано 14.06.2012.