Автоматное программирование

Определение. Модель. Реализация.

В.С.Любченко
п.Балакирево
sllubch@mail.ru

2018

1. Текущее состояние

- SWITCH-технология
  - Определение
  - Модель
  - Реализация
- UML
  - Определение
  - Модель
  - Реализация
- Языки визуального программирования
  - Stateflow (MATLAB)
  - Шаблоны State machine (LabVIEW)
  - Microsoft Robotics Studio
- Библиотеки программ и т.п.
  - Qt.
- Автоматные языки
  - SDL

SWITCH технология

UML

Визуальные ситемеы

Qt

SDL

2. Проблемы (визуального) проектирования

- Низкая скорость реального времени
  - тесты: сортировки, алгоритм сдваивания, …
- Проблемы реализация вложенной иерархии, рекурсии
  - Тесты: числа Фибоначчи, Ханойские башни, …
- Структурные проблемы – реализация обратных (циклических) связей
  - Тесты: RS-триггер
- Ошибки параллелизма операторов
  - Тесты: параллельные арифметические операторы, модель: объект- система управления.
- Неадекватность физическому параллелизму
  - Тесты: полоса пропускания, транспортные и инерционные задержки (цифровые схемы)

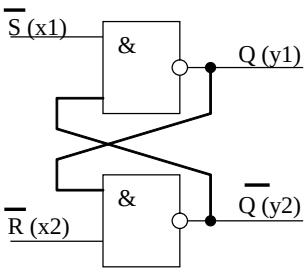
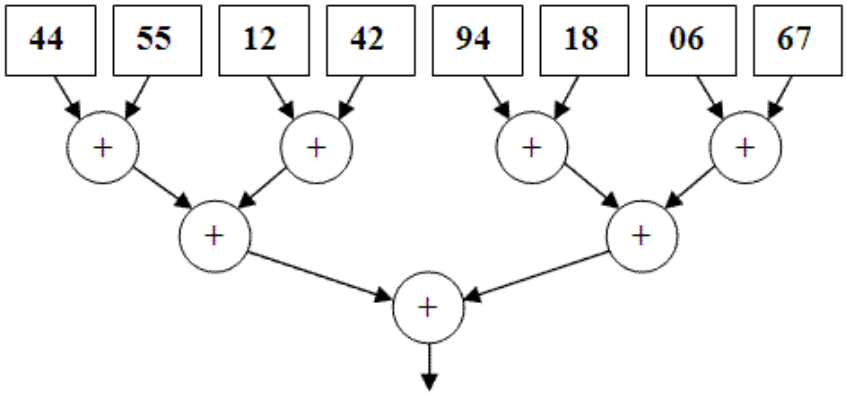


Схема RS-триггера



3. Определение автоматного программирования

- Модель схем программ: S = (M, A, G)
  - M – множество ячеек памяти, A – множество операторов, G – управление в форме конечного автомата.

Определение 1. Автоматной программой (АП) будем называть программу, имеющую модель управления в форме модели конечного автомата.

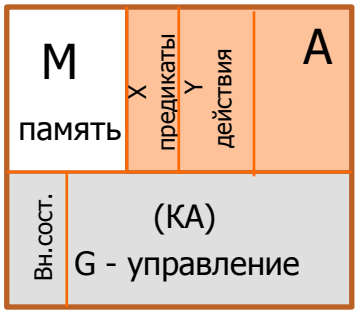
Определение 2. Инерционным законом функционирования автоматов будем называть закон, представленный функциями переходов:

- автоматы Мили

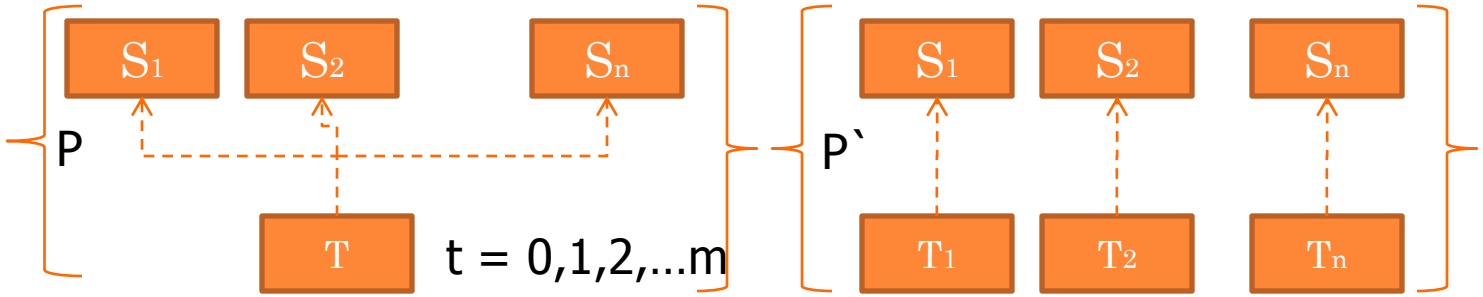
$a(t+1) = \delta(a(t), x(t)), y(t+1)=\lambda(a(t), x(t)), (t = 0, 1, 2, \ldots)$

- автоматы Мура

$a(t+1) = \delta(a(t), x(t)), y(t+1)=\lambda(a(t+1)), (t = 0, 1, 2, \ldots)$



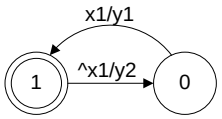
- Модель управления параллельных процессов
  - Параллельное соединение автоматов
  - Сети из автоматов
    - P – сеть автоматов в едином дискретном времени
    - P`– сеть не связных по времени автоматов



4. Формальная модель управления автоматных программ

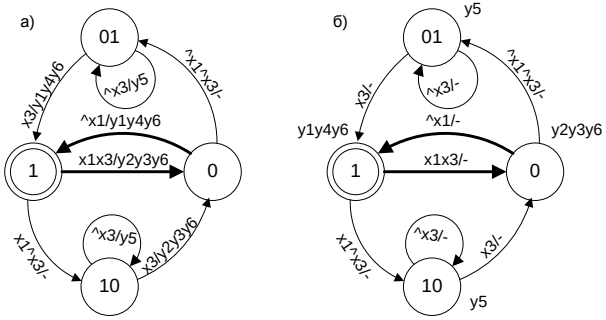
- Модель отдельного компонента
  - **Дизъюнктивные конечные автоматы**

Определение 3. Назовем *дизъюнктивной нормальной формой конечных автоматов (ДНКА)* автоматы, переходы которых помечены элементарными конъюнкциями логических переменных.



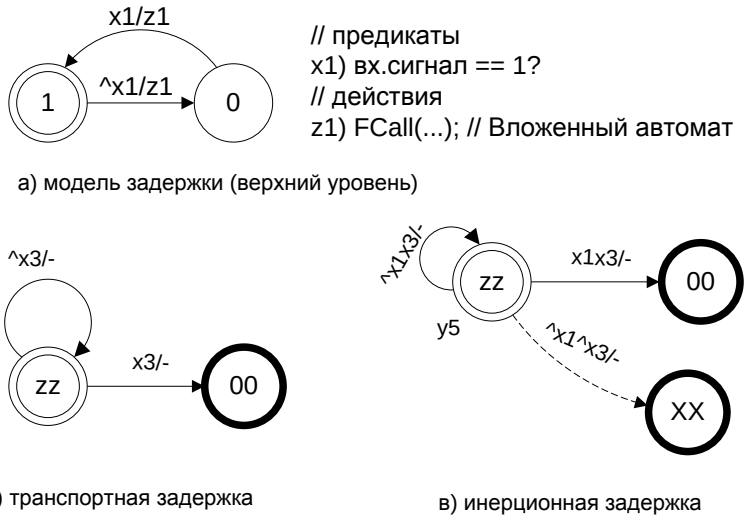

```
// предикаты
x1) вх.сигнал == 1?
// действия
y1) вых. сигнал := 1;
y2) вых. сигнал := 0;
```

Определение 4. Назовем *дизъюнктивной формой конечного автомата* (*ДКА*) автомат в форме ДНКА, содержащий только *результативные* переходы.



4.1.Вложенные автоматы

Определение 5. ***Вложенными автоматами*** будем называть вызываемые автоматы, имеющие заключительное состояние, переход в которое запускает процедуру возврата на предыдущий уровень вложенности.



Определение 6. Автоматы, включающие вызов вложенных автоматов, у которых есть инерционное заключительное состояние (в нашем случае ***состояние с именем «XX»***), будем называть ***инерционными автоматами***.

5. Вопросы теории автоматных схем программ

- Автоматные схемы программ
  - Анализ и синтез компонентов в рамках теории автоматов
    - анализ переходов на полноту и ортогональность
  - Преобразование моделей
    - ГСА->КА (разметка ГСА по С.И.Баранову)
    - SWITCH-технология
    - Сети Петри -> КА
  - Вложение автоматов
    - схема вложения, рекурсия автоматов
- Параллельные автоматные схемы программ
  - Параллельное соединение автоматов
  - Алгебра автоматов
    - Операция композиции (умножения) автоматов,
    - Операция декомпозиции (деления) автоматов.
  - Параллелизм действий

Отметка БС

RS-триггер

6. Язык параллельного автоматного программирования

- Текстовые и графические языки описания автоматов
  - Таблицы переходов
  - Автоматные графы
- Операторы и память автоматной модели программ
  - Глобальная и локальная память процессов
  - Операторы: предикаты, действия
- Объектная автоматная модель
  - Управление активного [автоматного] объекта
  - Методы: предикаты, действия
  - Свойства

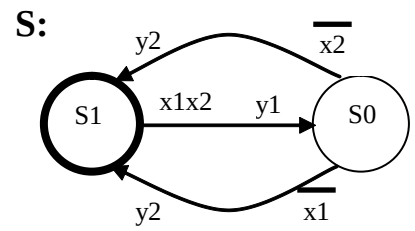
C++

Класс (ООП)

свойства

методы

тек. сост.	след. сост.	Входн. сигн.	Вых. сигн.
s1	s0	x1x2	y1
s0	s1	^x1	y2
s0	s1	^x2	y2



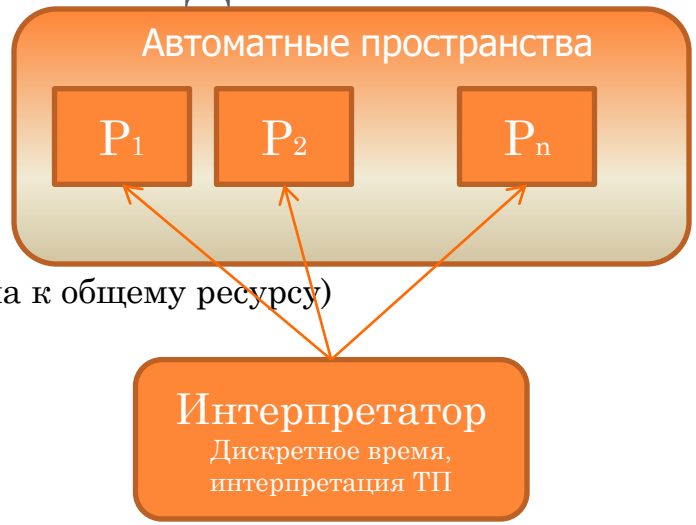
модель элемента И-НЕ

Активный класс (ВКПа)

свойства	M
методы	A: предикаты; действия
управление	G – таблица переходов КА

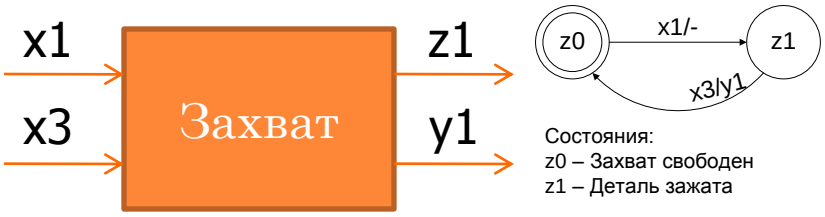
7. Реализация автоматной модели вычислений в рамках среды ВКПа

- Принцип интерпретации
  - Реализация дискретного времени
  - Интерпретация ТП автоматов
- Модель памяти
  - Глобальная и локальная память процессов
  - Теневая память
  - Последовательный тип памяти (как модель доступа к общему ресурсу)
  - Механизм блокировка элементов памяти
- Структурные свойства среды проектирования
  - блочная организация компонентов и библиотек
  - динамический вызов автоматных компонентов



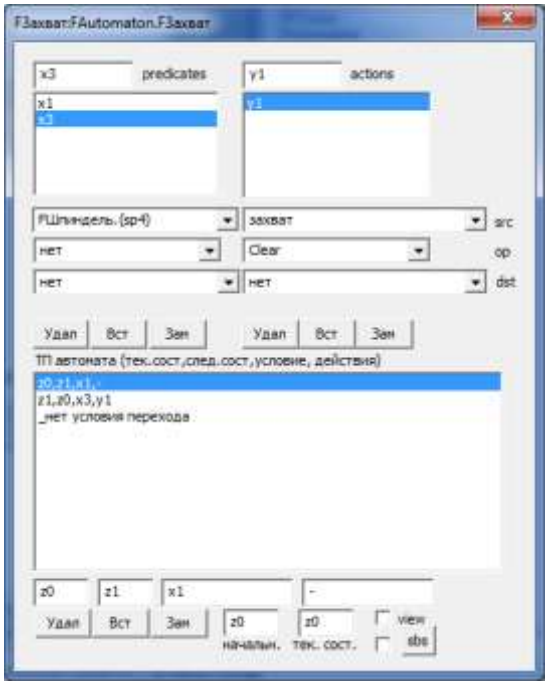
8. Визуальная автоматная машина

- Блок FAutomaton библиотеки FsaNet
- Создание структурной и формальной модели процесса
  - Определение входов/выходов процесса
- Проектирование процесса в терминах автоматной модели программ
  - Создание операторов, управления, локальной памяти процесса
- Визуализация текущего состояния процесса
  - Визуализация внутреннего состояния и переходов КА
- Протоколирование состояний
  - Доступ к текущему состоянию процесса
- Отладка в реальном времени
  - Режим step-by-step
  - Установка текущего состояния извне



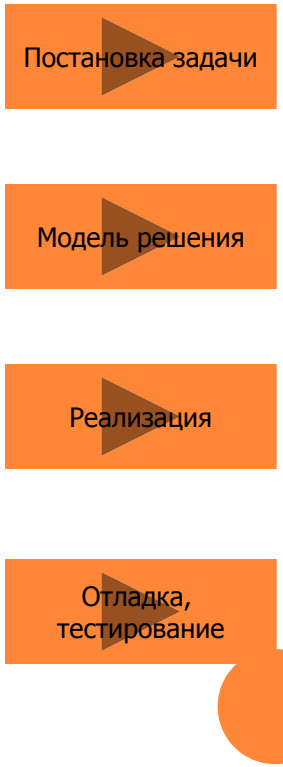
Состояния:
z0 – Захват свободен
z1 – Деталь зажата

Предикаты:
x1) Есть заготовка?
x3) Шпиндель(sp4)?
Действия:
y1) Сигнал готовности детали



9. Технология визуального автоматного программирования

- Модели жизненного цикла информационных систем
  - Каскадная модель
  - Спиральная модель жизненного цикла ПО
- Этапы/стадии жизненного цикла
  - 1. Определение требований/спецификаций
    - Постановка задачи
  - 2. Проектирование системы
    - Структурная модель
    - Алгоритмическая модель
  - 3. Программирование
    - Реализация алгоритмов на языке программирования среды
  - 4. Отладка
    - Прогон в режиме тестирования
    - Создание протокола состояний процессов
  - 5. Сопровождение
    - Документирование процессов
    - Корректировка проекта



10. Тестовые примеры

- Дискретное время
  - Цифровые фильтры, ПИД-регулятор
- Синхронизация через состояния
  - Управление клапаном
  - Роботизированный станок
- Теневая память
  - Параллельные арифметические операторы
- Последовательная память
  - Доступ к общему ресурсу
- АПЛ-500

Адаптивный ПИД-регулятор

Роботизированный станок

АПЛ-500

11. Уникальные свойства технологии ВКПа

- **1) Концепция дискретных [автоматных] пространств**
  - оперирование множеством пространств,
  - асинхронный/синхронный режимы работы пространств,
  - индивидуальное дискретное время.
  - скорость реализации
    - дискретный такт<1мсек (в режиме интерпретации автоматов)
- **2) Параллельная модель процессов**
  - отдельный процесс – дизъюнктивный автомат (ДКА),
  - вложение автоматов,
  - множество процессов - сеть ДКА в едином времени,
- **3) Принципы взаимодействия памяти, операторов и управления, теневая память**
  - адекватность физическому параллелизму (использование теневой памяти),
  - локальная и глобальная память процессов,
  - интерпретация табличной формы описания автоматов.
- **4) активный C++**
  - включение в состав класса управления в форме таблицы переходов,
  - выделение методов – предикатов, действий,
  - активные объекты,
  - рекурсия объектов/процессов.
- **5) Визуальная «автоматная машина»**
  - проектирование «на лету»,
  - визуализация процесса исполнения,
  - отладка в реальном времени.

ВЫВОДЫ

- Актуальна замена параллелизма на базе потоков/нитей
  - реализация концепции кибер-физических систем
- Развитие систем проектирования:
  - переход от последовательного программирования к параллельному
  - разработка универсальных систем визуального программирования.
- Новый виток использования автоматных моделей.
- ВКП(а) – универсальная технология проектирования программных систем управления

Литература

- Любченко В. С. К решению проблемы обедающих философов Дейкстры //Высокопроизводительные параллельные вычисления на кластерных системах: Материалы четвертого Международного научно-практического семинара. / Самара. 2004. с. 186-193.
- Любченко В.С. К проблеме создания модели параллельных вычислений // Труды III Международной конференции «Параллельные вычисления и задачи управления» PACO `2006. Москва, 2-4 октября 2006 г. Институт управления им. В.А. Трапезникова РАН. М: Институт проблем управления им. В.А. Трапезникова РАН, 2004. (ISBN 5-201-14990-1) – С. 1359-1374 http://paco.sicpro.org/paco2006/code/proc_rus.htm, http://www.softcraft.ru/auto/ka/modproblem/modproblem.pdf
- Любченко В.С. Параллельные сортировки: быстрее, проще... умнее. “Открытые системы”, #5/2004, http://www.osp.ru/os/2004/05/049.htm
- Любченко В.С., Тяжлов Ю.А. Осторожно: многоядерный процессор. “Открытые системы”, #6/2007, http://www.osp.ru/os/2007/06/4337893/
- Любченко В.С. Параллелизм истинный и мнимый. “Открытые системы”, #1/2010, http://www.osp.ru/os/2010/01/13000689/
- *Любченко В.С. Автоматные методы синтеза параллельных программ. //Н*аучный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). - М.: Изд-во МГУ, 2012. - 752 с. ISBN 978-5-211-06394-5 С.44-53 (доступна на http://agora.guru.ru/abrau2012/pdf/44.pdf)
- *Любченко В.С. Автоматная парадигма параллельного программирования //Н*аучный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). - М.: Изд-во МГУ, 2012. - 752 с. ISBN 978-5-211-06394-5 С.40-43 (доступна на http://agora.guru.ru/abrau2012/pdf/40.pdf)
- Любченко В.С. Многоядерный тупик – есть ли решение? // Тезисы докладов Четвертого Московского суперкомпьютерного форума (Москва 23 октября 2013 г.)./ [Под ред. Волкова Д.В.]/ - М.: «Открытые системы», 2013. – 48с. (http://www.ospcon.ru/files/media/mscf_Thesis_2013.pdf)
- Любченко В.С. Многоядерный тупик: выход есть. “Открытые системы”, #8/2013, http://www.osp.ru/os/2013/08/13037860/
- *Любченко В.С., Ломакин Р.Л., Перфилов С.А. Об опыте создания параллельной системы управления синтезом искусственных минералов на базе конечно-автоматной модели* . Труды и пленарные доклады участников конференции УКИ'12 / Научное издание. Электрон. текстовые дан. - М.:ИПУ РАН, 2012 - 1 электрон. опт. диск (CD-ROM) - ISBN 978-5-91450-100-3 - С. 001354-001359
- Любченко В.С., Перфилов С.А., Ломакин Р.Л. Система управления прессом «АСУ ТП - Пресс» на базе автоматной модели параллельных процессов. // Свидетельство о государственной регистрации программы для ЭВМ №2012615341. Зарегистрировано 14.06.2012.

СПАСИБО

ЗА ВНИМАНИЕ!

Построение автомата эквивалентного блок-схеме
(алгоритм разметки БС)

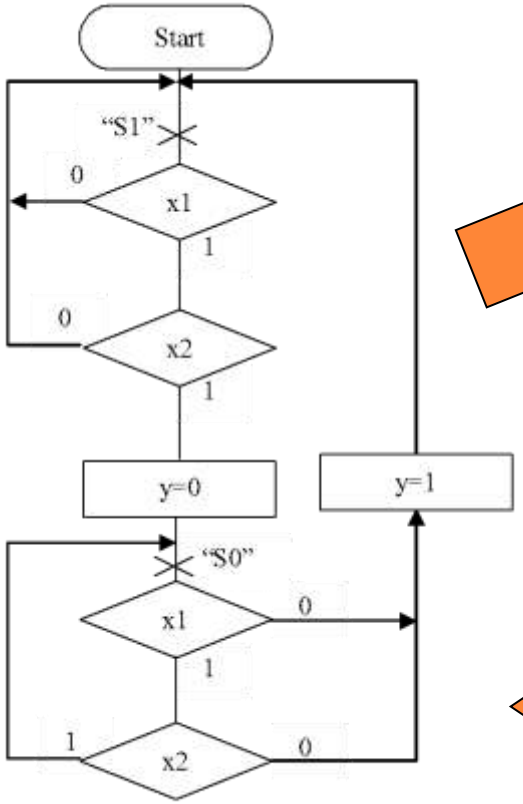


Рис 1. Пример разметки блок-схемы.

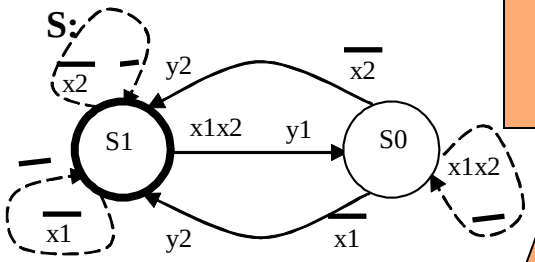


Рис 2. Эквивалентный БС автомат

убираем петли не нагруженные действиями

6

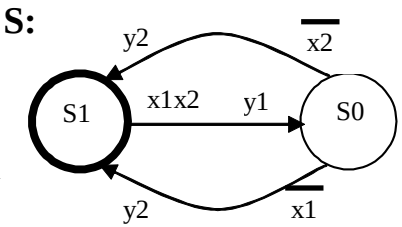


Рис 3. ДНФ автомата Мили

взаимные преобразования

Объектный язык кодирования автоматов, C++

- Заколовок класса (FViewTape)

```
class FViewTape : public LFsaAppl
{
public:
    LFsaAppl **ppFsaPD;
    LFsaAppl **ppFsaET;
    FViewTape();
    virtual ~FViewTape();
private:
    int x1(),x2(),x3(),x4();
    void y1(),y2(),y3();
//  предикаты и действия
//   для реакции на клавишу "="
    int x5(),x6();
    void y4();
    string strLine;
}
```

- Реализация класса (FViewTape)

```
FViewTape::FViewTape():LFsaAppl(TT_ViewTape)
{ ppFsaPD = NULL; ppFsaET = NULL; }
FViewTape::~FViewTape() { }
//      таблица переходов автомата
LArc TT_ViewTape[] = {
        LArc("t0", "t1", "x3",          "--"),
        LArc("t0", "t0", "x1",          "y3"),
        LArc("t0", "00", "^x2",         "y2"),
        LArc("t0", "ns", "x5x6",        "y4"),
        LArc("ns", "t0", "^x5",         "--"),
        LArc("t1", "t0", "^x4",         "y1"),
        LArc("t1", "t0", "x4",          "--"),
        LArc()
        };
//      предикаты
int FViewTape::x1() { return string((*ppFsaPD)->FGetState()) == "ok"; }
int FViewTape::x2() { return (*ppFsaPD)->FIsActiveTask(); }
int FViewTape::x3() { return string((*ppFsaPD)->FGetState()) == "АМП"; }
int FViewTape::x4() { return string((*ppFsaPD)->FGetState()) == "er"; }
int FViewTape::x5() { return string((*ppFsaET)->FGetState()) == "sym"; }
int FViewTape::x6() { return (*(FEntranceTape**)ppFsaET)->GetSym() == '='; }
//      действия
void FViewTape::y1() { strLine += (*((CPushDownBase**)ppFsaPD))->chSym; }
void FViewTape::y2() { cout << "Exit from FViewTape!\n"; }
void FViewTape::y3() { cout << strLine << "+++++ OK! +++++\n"; }
void FViewTape::y4() { cout << strLine << "\n"; }
```

СХЕМА И МОДЕЛЬ RS-ТРИГГЕРА

- Схема RS-триггера

- сетевая модель RS-триггера

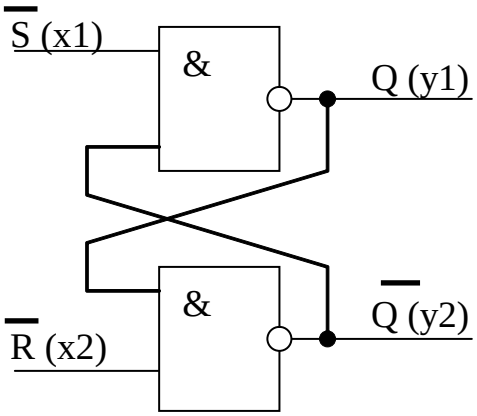
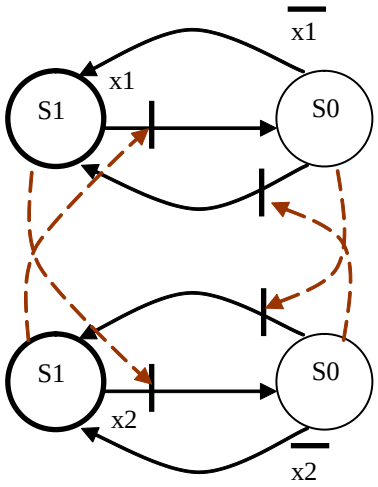
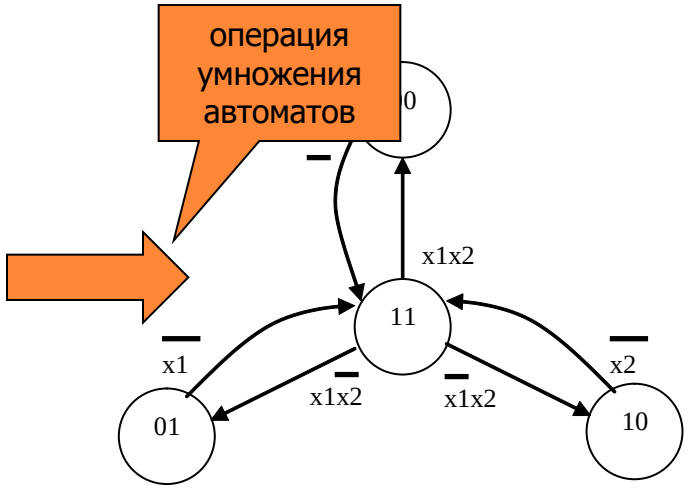


Схема RS-триггера

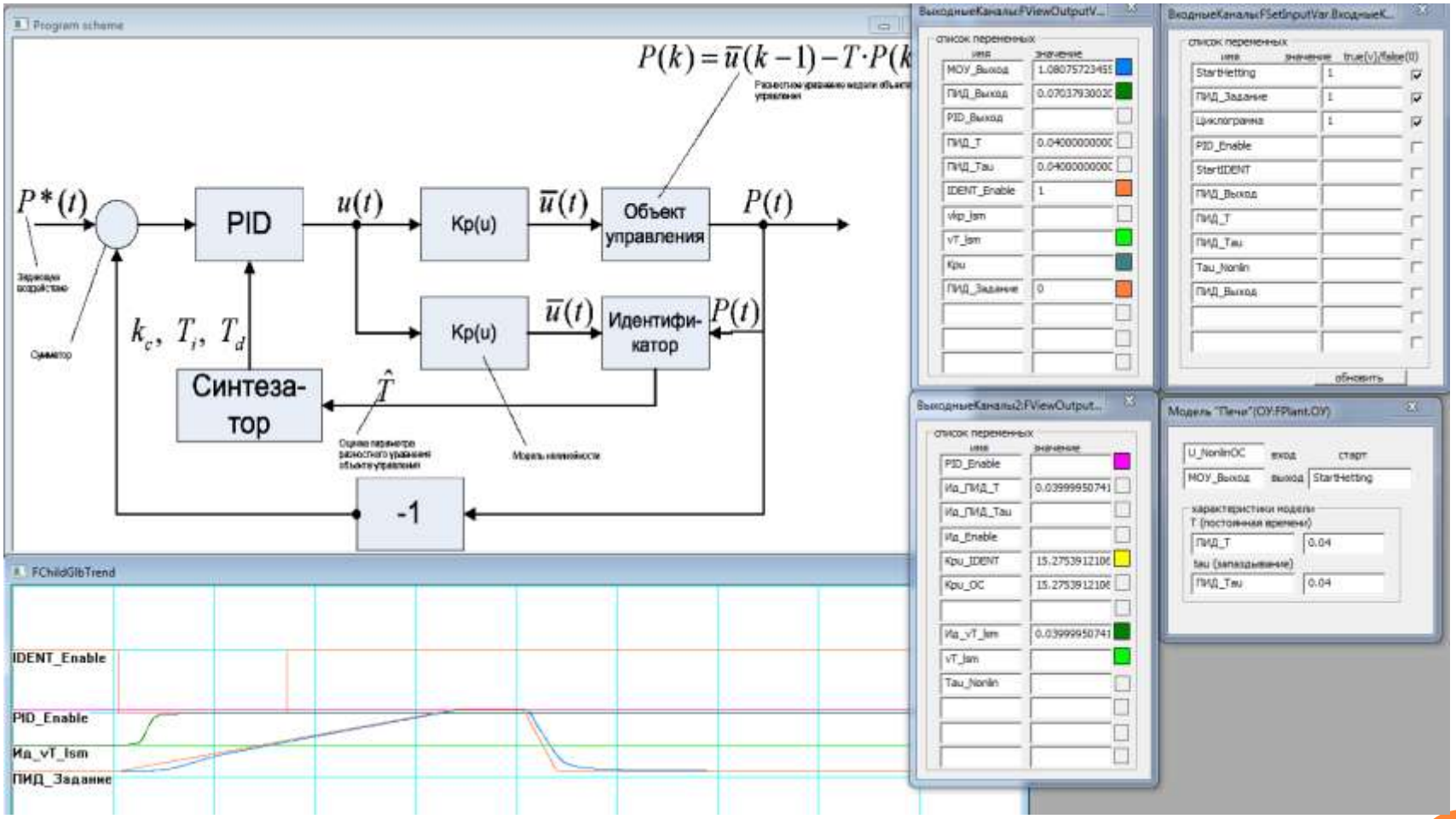


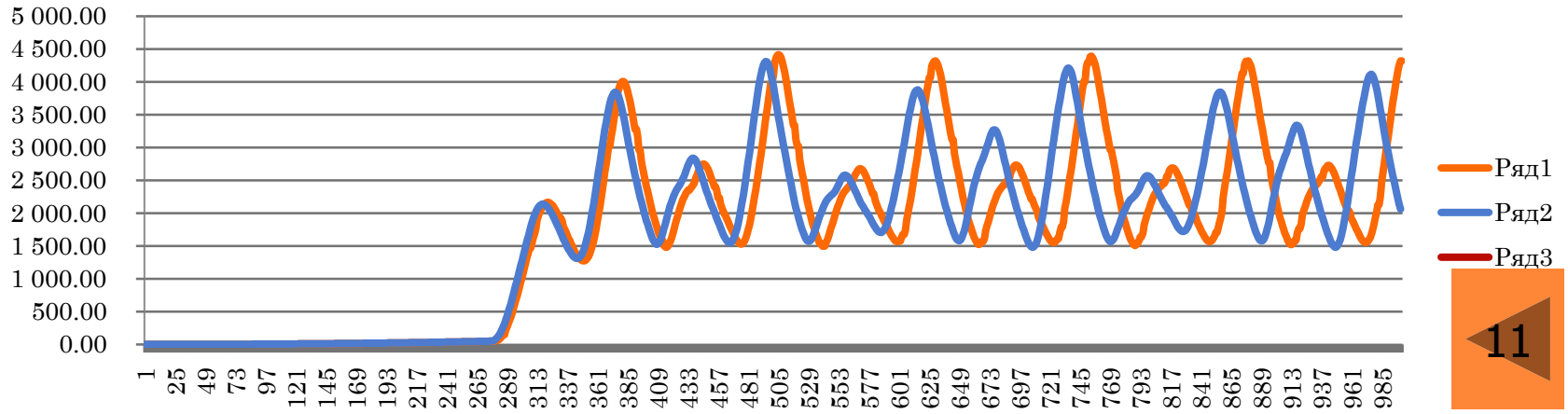
Сетевая модель RS-триггера.

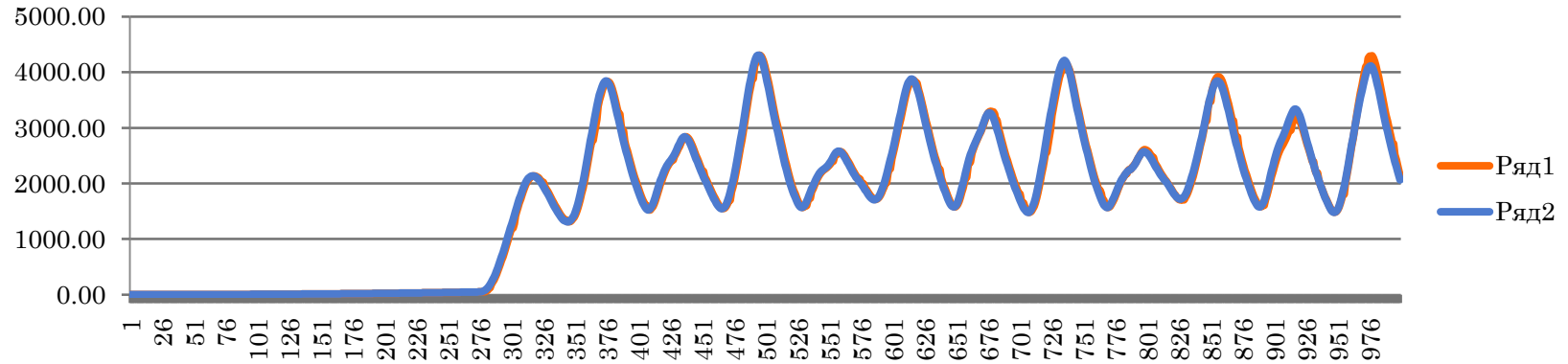


Эквивалентный автомат сетевой модели RS-триггера

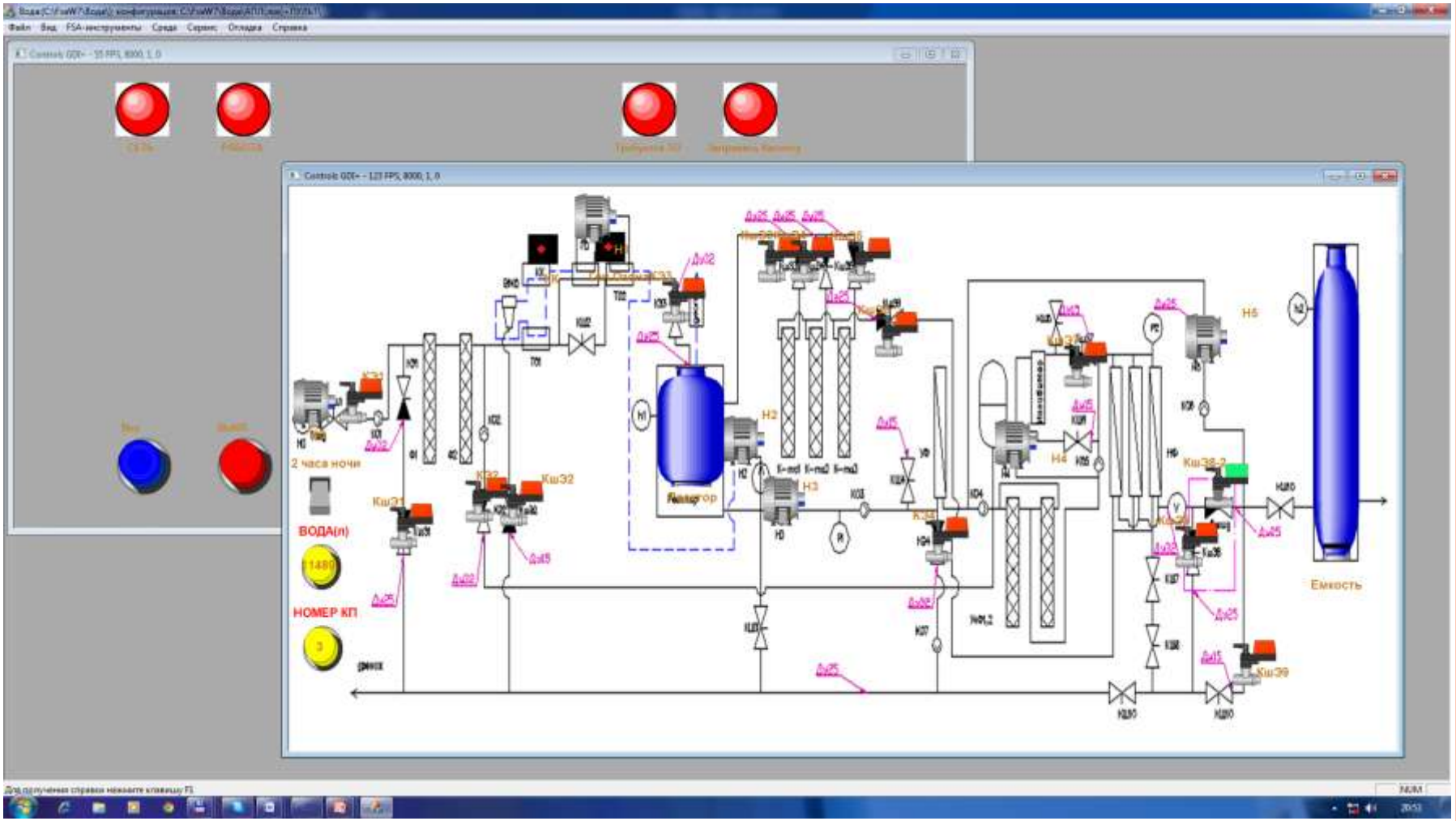
Адаптивный ПИД-регулятор



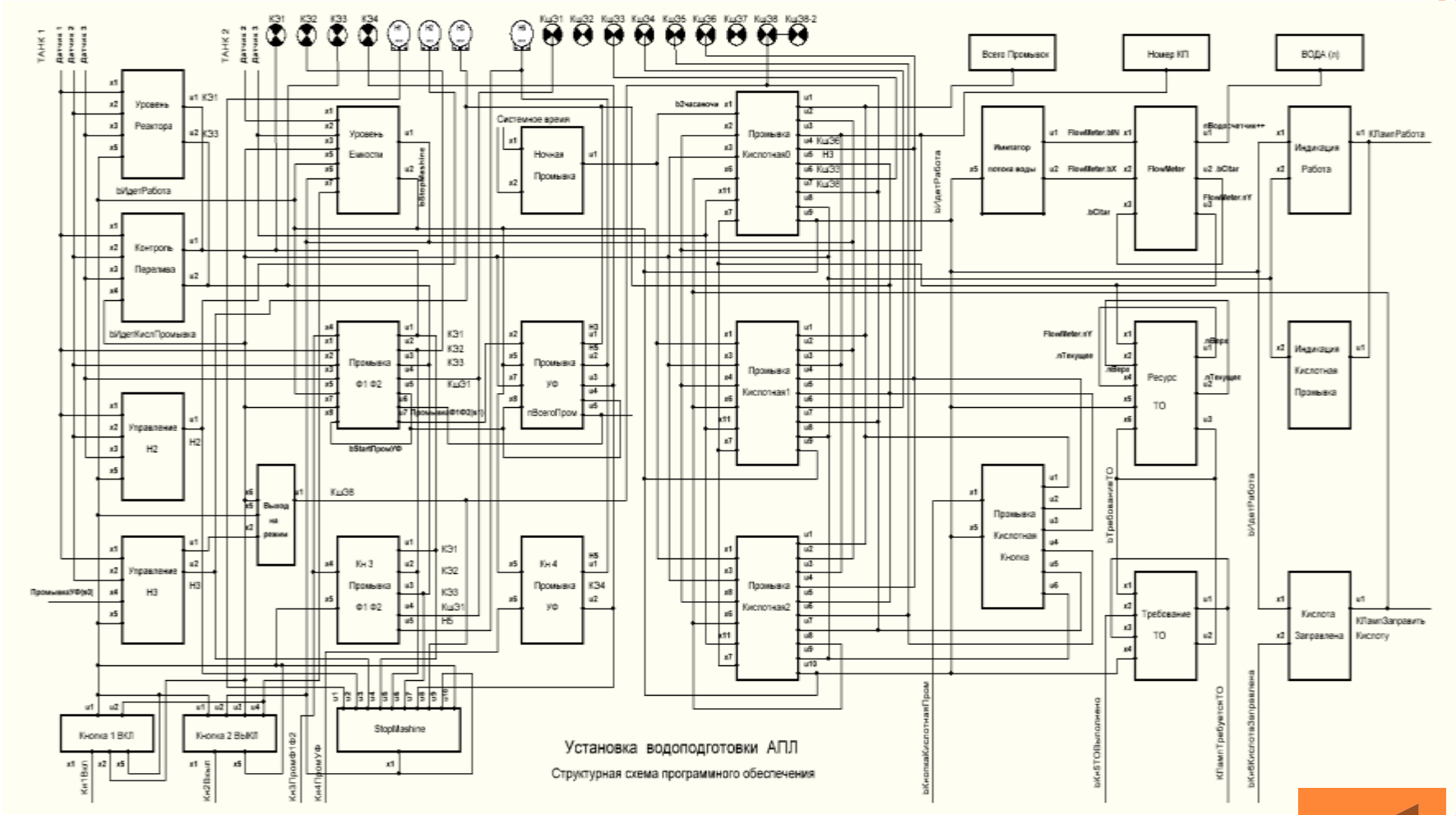

5 000.00
4 500.00
4 000.00
3 500.00
3 000.00
2 500.00
2 000.00
1 500.00
1 000.00
500.00
0.00
1
25
49
73
97
121
145
169
193
217
241
265
289
313
337
361
385
409
433
457
481
505
529
553
577
601
625
649
673
697
721
745
769
793
817
841
865
889
913
937
961
985
Ряд1
Ряд2
Ряд3


5000.00
4000.00
3000.00
2000.00
1000.00
0.00
1
26
51
76
101
126
151
176
201
226
251
276
301
326
351
376
401
426
451
476
501
526
551
576
601
626
651
676
701
726
751
776
801
826
851
876
901
926
951
976
Ряд1
Ряд2

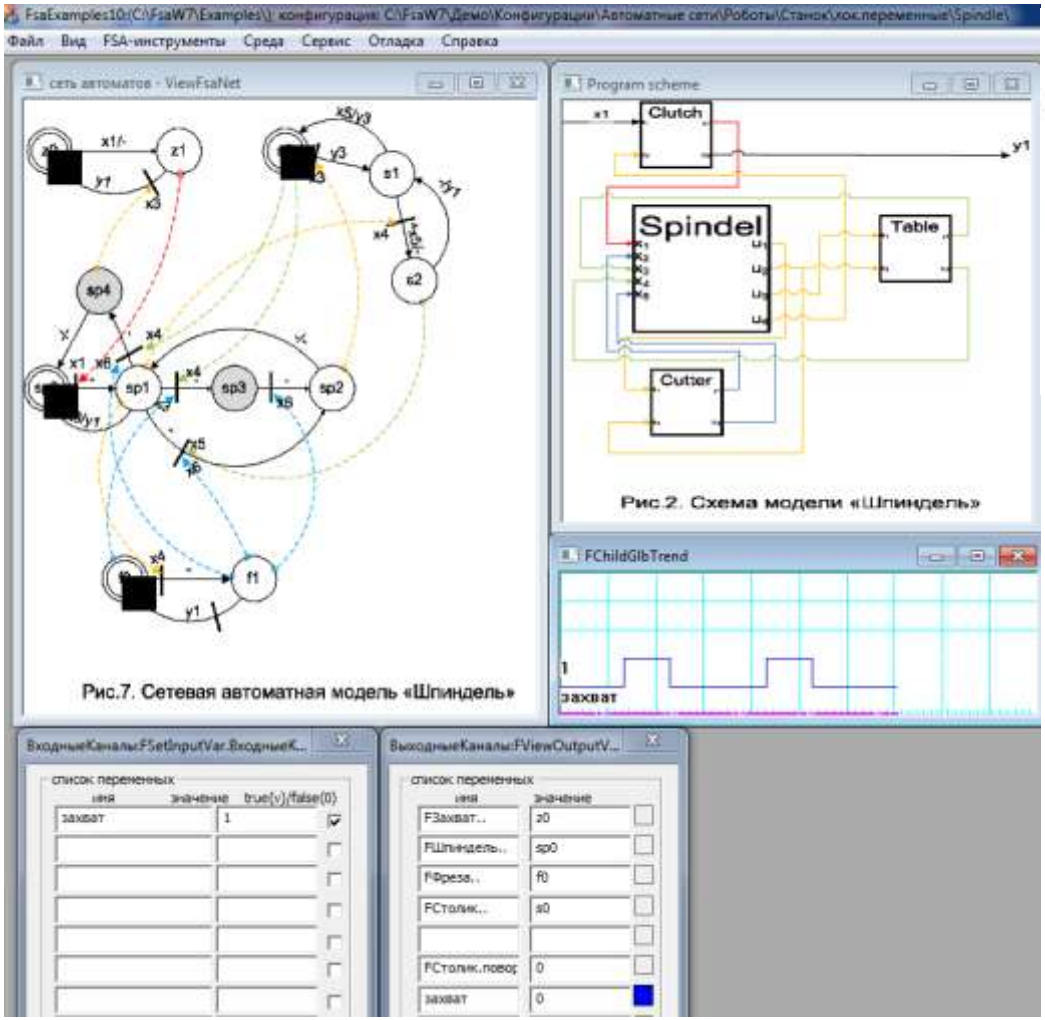
- Система управления водоочисткой АПЛ-500

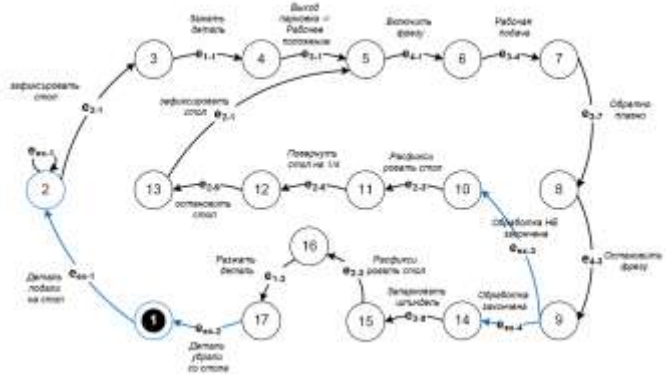


Структурная схема АПЛ-500



Роботизированный станок





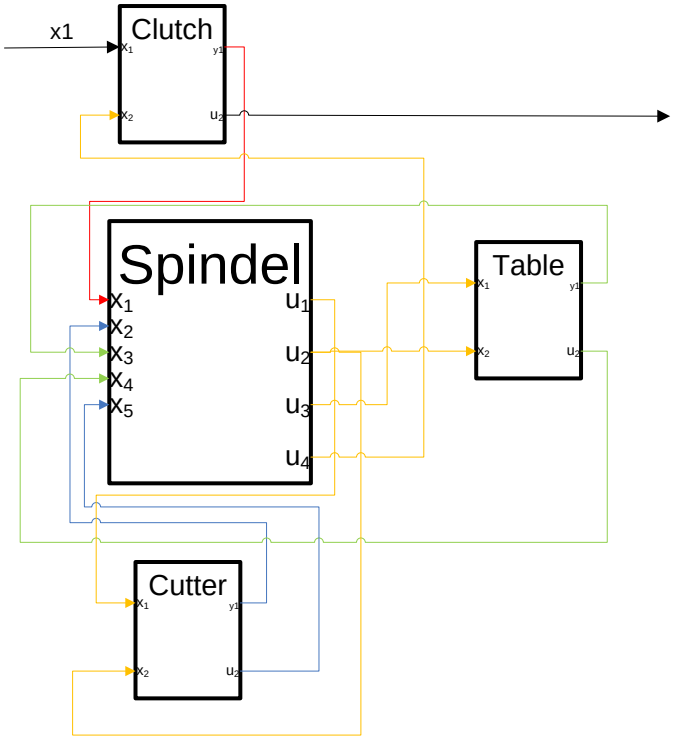
11

1. Амбарцумян А.А. Дискретно-событийное моделирование в технических системах. УКИ-12. 2012г

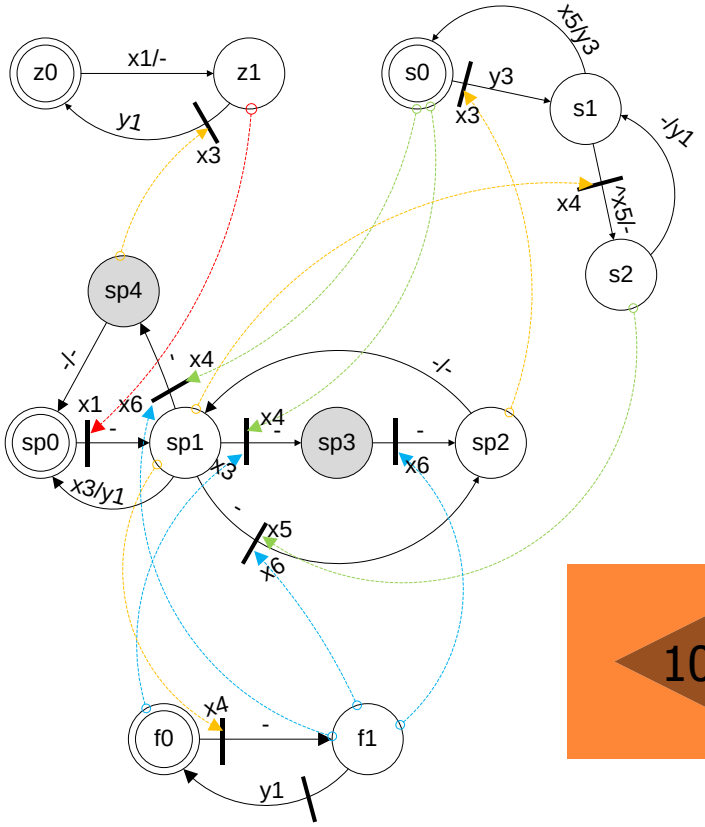
Постановка задачи

- 1. По прибытии заготовка зажимается захватом;
- 2. После того, как сработал захват, шпиндель переводится из парковочного положения в рабочее положение (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 ¼ 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.

Структурная и алгоритмическая модели решения

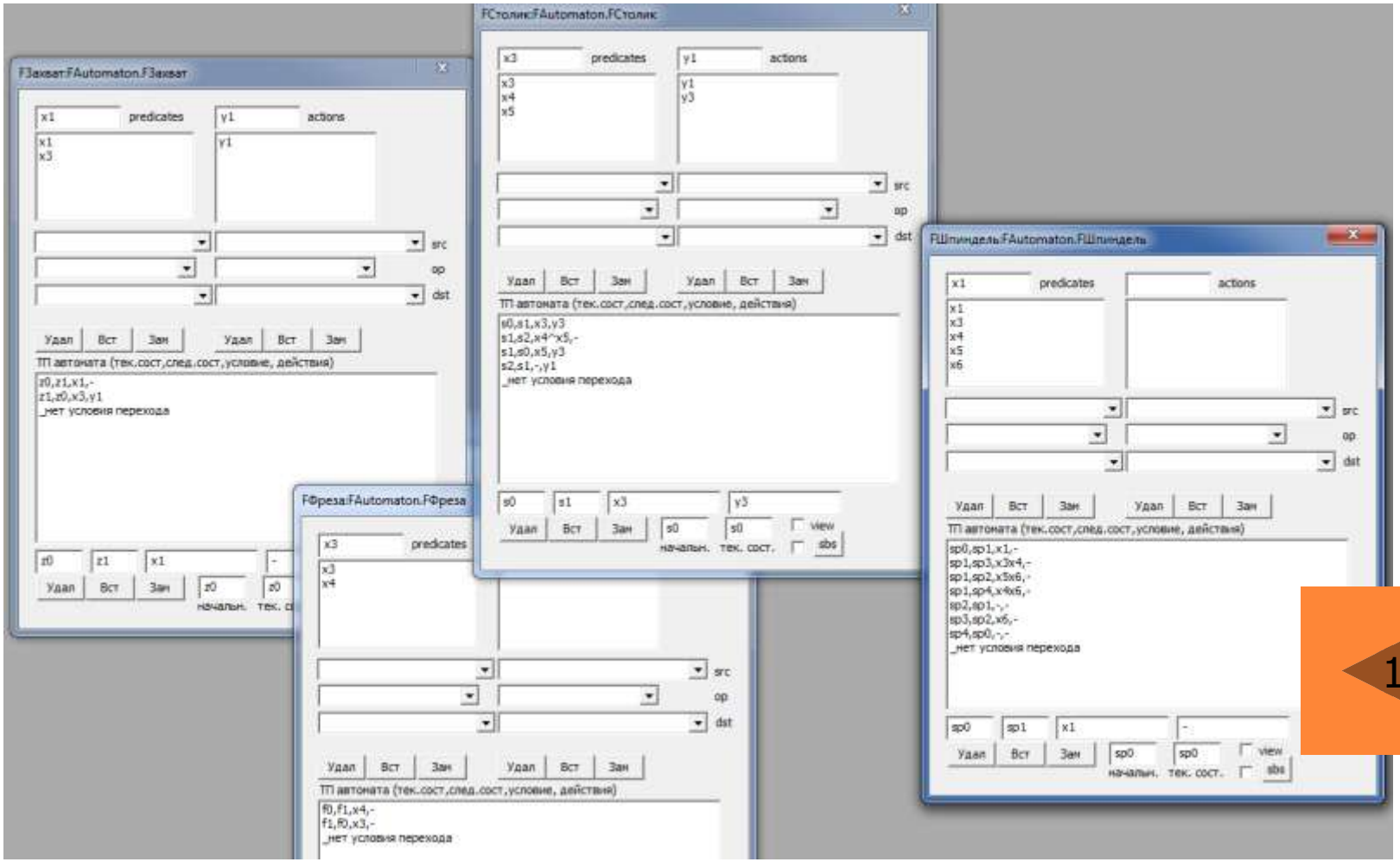


Структурная модель



Алгоритмическая модель

Реализация модели



Этап отладки, тестирования проекта

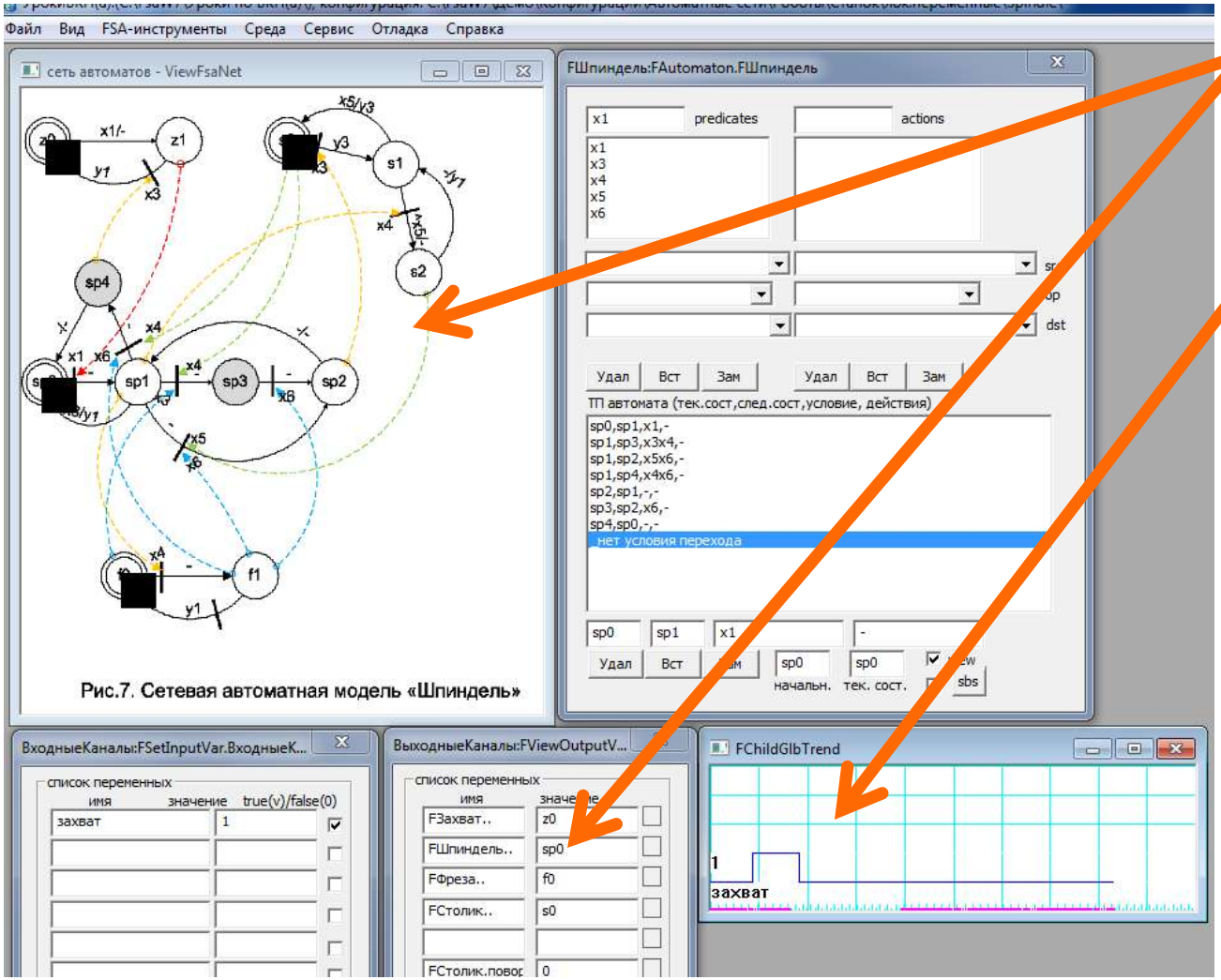


Рис.7. Сетевая автоматная модель «Шпиндель»

Визуализация текущих состояний процессов

Тренды значений переменных

SWITCH-ТЕХНОЛОГИЯ

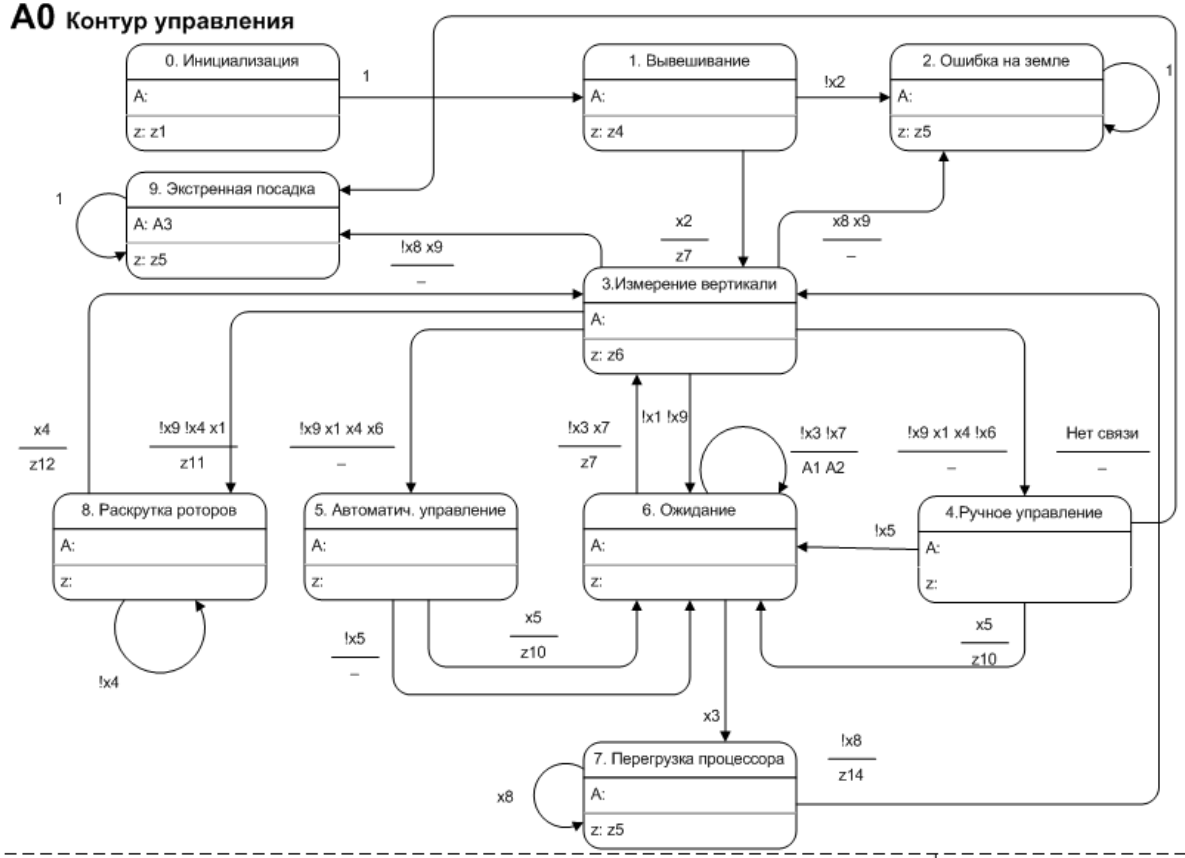
ОПРЕДЕЛЕНИЕ

- Парадигма автоматного программирования состоит в представлении и реализации программ как систем ***автоматизированных объектов управления***
- Основным понятием в автоматном программировании является ***состояние***.
- В рамках автоматного программирования основное внимание уделяется ***управляющим*** состояниям.
- ***Время*** в автоматах в явном виде не используется.
- Для автоматных программ естественен параллелизм (что особенно важно при применении многоядерных процессоров).
- Отличие подхода от известных состоит в том, что ***предлагается применять автоматы в программировании не от случая к случаю, а везде и всегда, где требуется обеспечить сложное поведение***

Реализация автоматных программ

- Вне зависимости от используемого языка программирования реализация автоматных программ осуществляется по графам переходов формально и изоморфно – текст программы «внешне похож» на граф переходов.
- Переход от графа переходов к тексту программы может осуществляться как вручную, так и автоматически с помощью соответствующих инструментальных средств.
  - Например, для создания программ на языке Java в СПбГУ ИТМО создано инструментальное средство UniMod.

A0 Контур управления



ВХОД A0
X1 – Запуск разрешен;
X2 – Успешное вывешивание;
X3 – Таймер истек до входа в Ожидание;
X4 – Ротор раскручен;
X5 – Такт управления;
X6 – Режим управления «Ручной»;
X7 – Истек таймер;
X8 – Нахожусь на земле;
X9 – Слишком малая частота работы САУ

ВЫХОД A0
Z1 – Инициализация;
Z4 – Вывешивание;
Z5 - Сообщить об ошибке;
Z6 – Фильтрация данных;
Z7 – Сброс таймера;
Z9 – Останов;
Z10 – Вывод управления в канал ШИМ;
Z11 – Раскрутка ротора;
Z12 – Сброс курсовертикали;
Z14 – Снижение частоты работы САУ;
Z15 – Повышение частоты работы САУ;

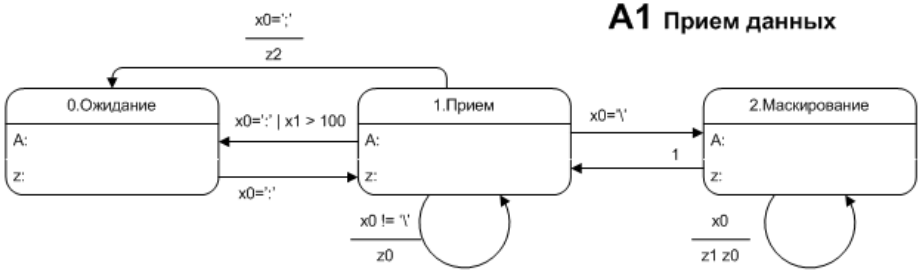
ВХОД A1
X0 – Входящий символ;
X1 – Длина принятого пакета;

ВЫХОД A1
Z0 – Запись символа в буфер, увеличение X1;
Z1 – Демаскирование x0 = x0 xor '\';
Z2 – Исполнение команды;;

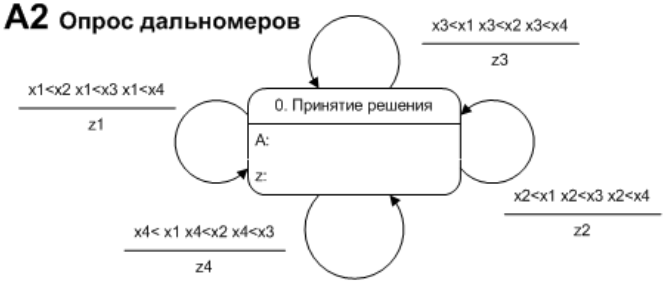
ВХОД A2
X1 – Доверие к дальномеру 1;
X2 – Доверие к дальномеру 2;
X3 – Доверие к дальномеру 3;
X4 – Доверие к дальномеру 4;

ВЫХОД A2
Z1 – Измерение дальномер 1;
Z2 – Измерение дальномер 2;
Z3 – Измерение дальномер 3;
Z4 – Измерение дальномер 4;

A1 Прием данных



A2 Опрос дальномеров



UML

ОПРЕДЕЛЕНИЕ

- Унифицированный язык моделирования (UML) – это ***семейство графических нотаций***. Он помогает в описании и проектировании программных систем, построенных с использованием объектно-ориентированных (ОО) технологий.

…

- **Диаграммы состояний**

  Диаграммы состояний (state machine diagrams) – это известная технология описания поведения системы. В том или ином виде диаграммы состояний существуют с 1960 года, и на заре объектно-ориентированного программирования они применялись для представления поведения системы. В объектно-ориентированных подходах вы рисуете диаграмму состояний единственного класса, чтобы показать поведение одного объекта в течение его жизни.

Диаграммы состояний

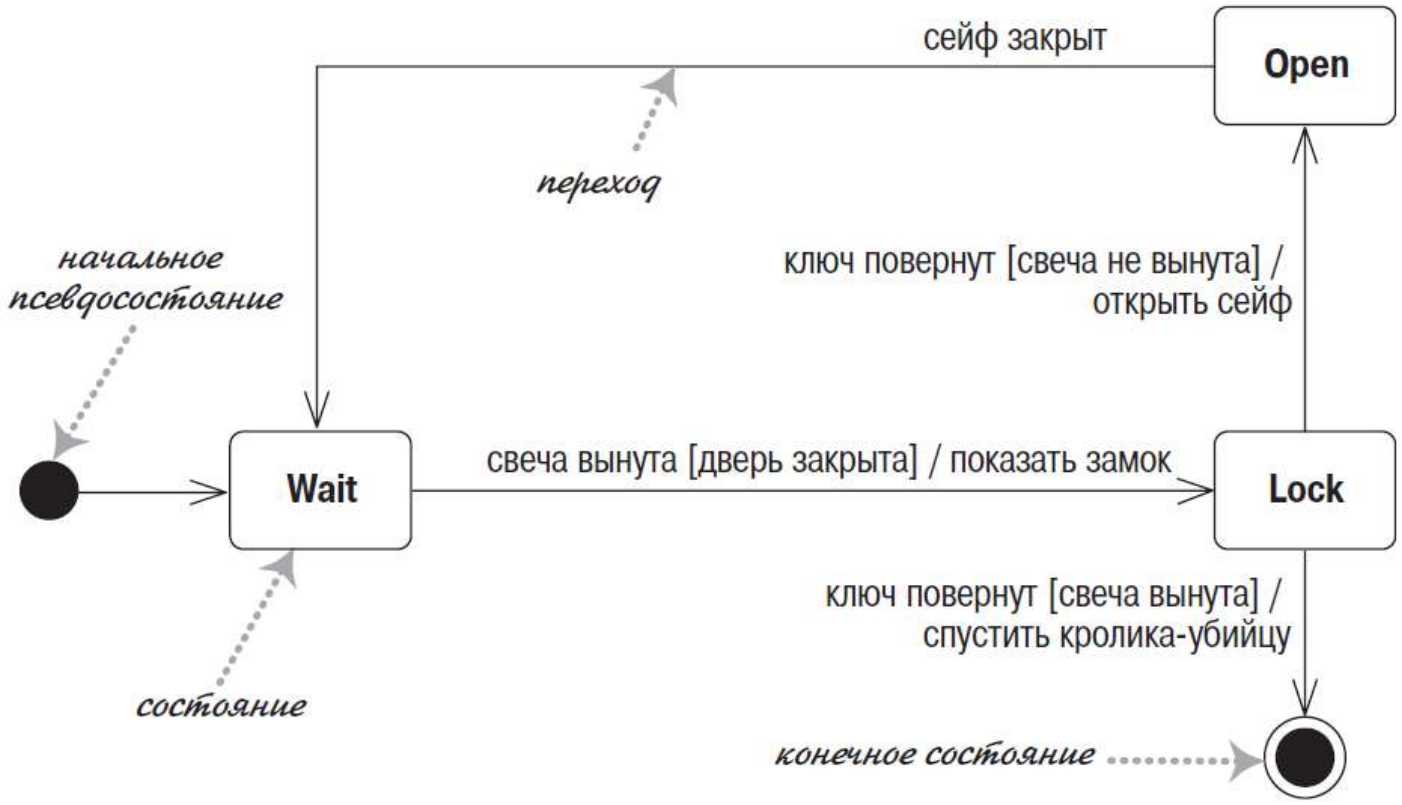


Рис. *Простая диаграмма состояний*

Реализация диаграмм состояний

- Реализация диаграмм состояний:
  - вложенный оператор switch,
  - паттерн State
  - таблица состояний.

```
public void HandleEvent (PanelEvent anEvent) {
   switch (CurrentState) {
      case PanelState.Open :
         switch (anEvent) {
            case PanelEvent.SafeClosed :
               CurrentState = PanelState.Wait;
               break;
         }
         break;
      case PanelState.Wait :
         switch (anEvent) {
            case PanelEvent.CandleRemoved :
               if (isDoorOpen) {
                  RevealLock();
                  CurrentState = PanelState.Lock;
               }
               break;
         }
         break;
      case PanelState.Lock :
         switch (anEvent) {
            case PanelEvent.KeyTurned :
               if (isCandleIn) {
                  OpenSafe();
                  CurrentState = PanelState.Open;
               } else {
                  ReleaseKillerRabbit();
                  CurrentState = PanelState.Final;
               }
               break;
         }
         break;
      }
   }
}
```

Рис. Вложенный оператор switch на языке C# для обработки перехода состояний, представленного на рис. 10.1

Визуальное программирование

- Число задач моделирования стало настолько большим, что специалистам потребовались достаточно универсальные системы ***блочного моделирования***, реализующие визуально-ориентированный подход к имитационному моделированию произвольных по структуре, назначению и областям применения систем.

Дьяконов В.П. VisSim+Mathcad+MATHLAB. Визуальное математическое моделирование. – М.: СОЛОН-Пресс, 2004. – 384 с.

Stateflow (MATLAB)

- Stateflow® — среда для моделирования и симуляции комбинаторной и последовательной логики принятия решений, основанных на машинах состояний и блок-схемах. Stateflow позволяет комбинировать графические и табличные представления, включая диаграммы перехода состояний, блок-схемы, таблицы перехода состояний и таблицы истинности, для того, чтобы смоделировать реакцию системы на события, условия во времени и внешние входные сигналы.

Stateflow (MATLAB)

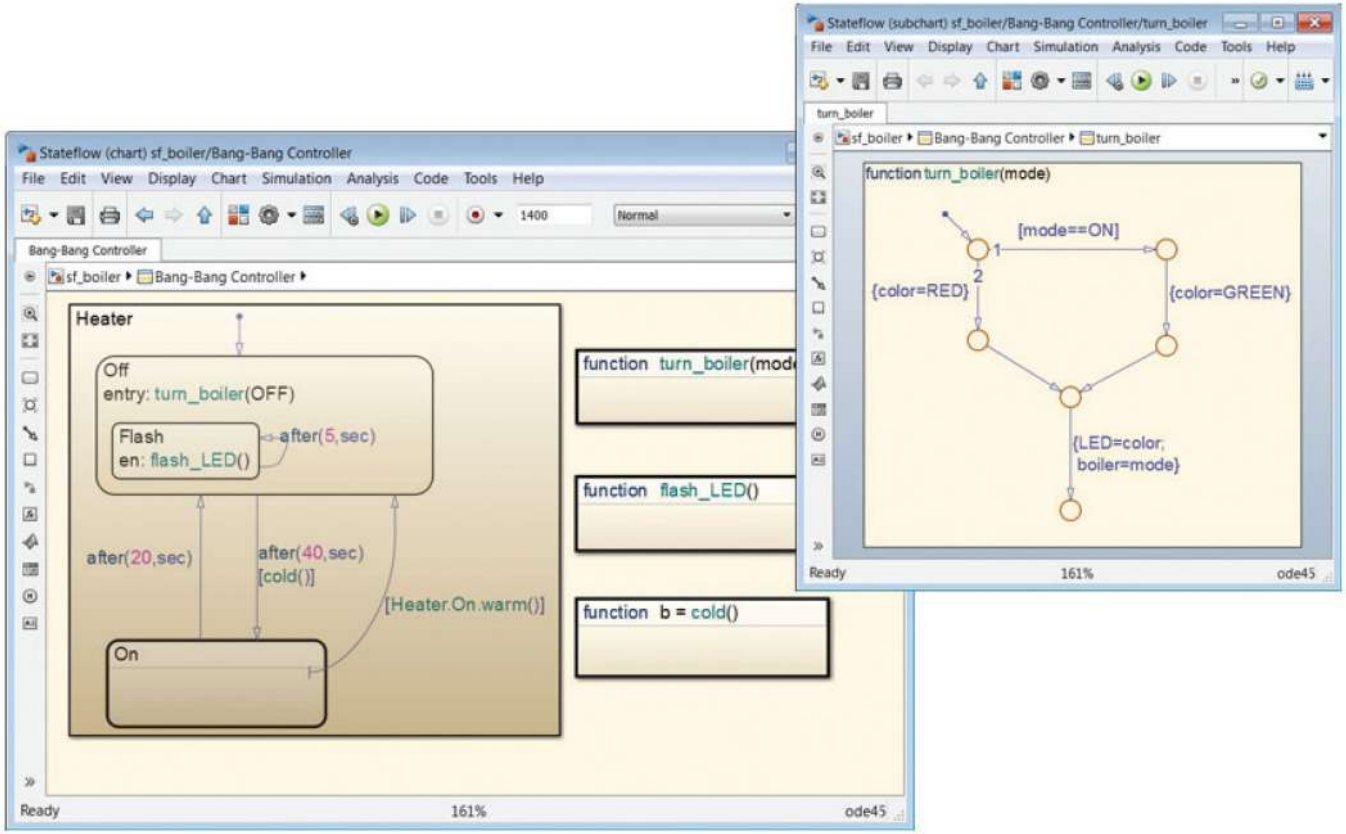


Диаграмма Stateflow, определяющая логику для системы управления температурой бойлера. В диаграмме используются графические функции (справа) для реализации алгоритмов, которые вызываются системой подогрева (слева).

LabVIEW

…

Конечные автоматы (State machine) – *наиболее разрекламированные шаблоны* в LabVIEW. Существует множество вариаций, большинство из которых состоит из структуры Case, расположенной внутри цикла While, сдвиговым регистром или конструктом сообщений (альтернативная схема обмена данными между итерациями, соединенным с терминалом селектора структуры Case.

Блюм П. LabVIEW: стиль программирования. Пер. с англ. Под ред. Михеева П.-М.: ДМК Пресс – 400 с.

LabVIEW

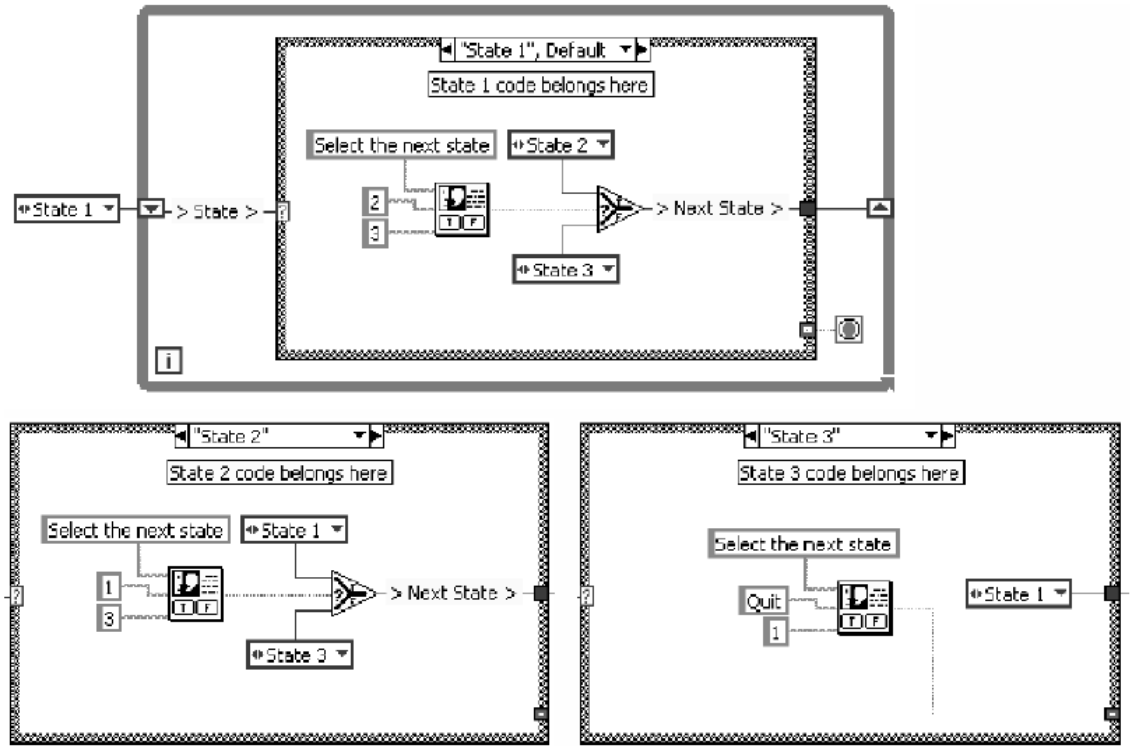
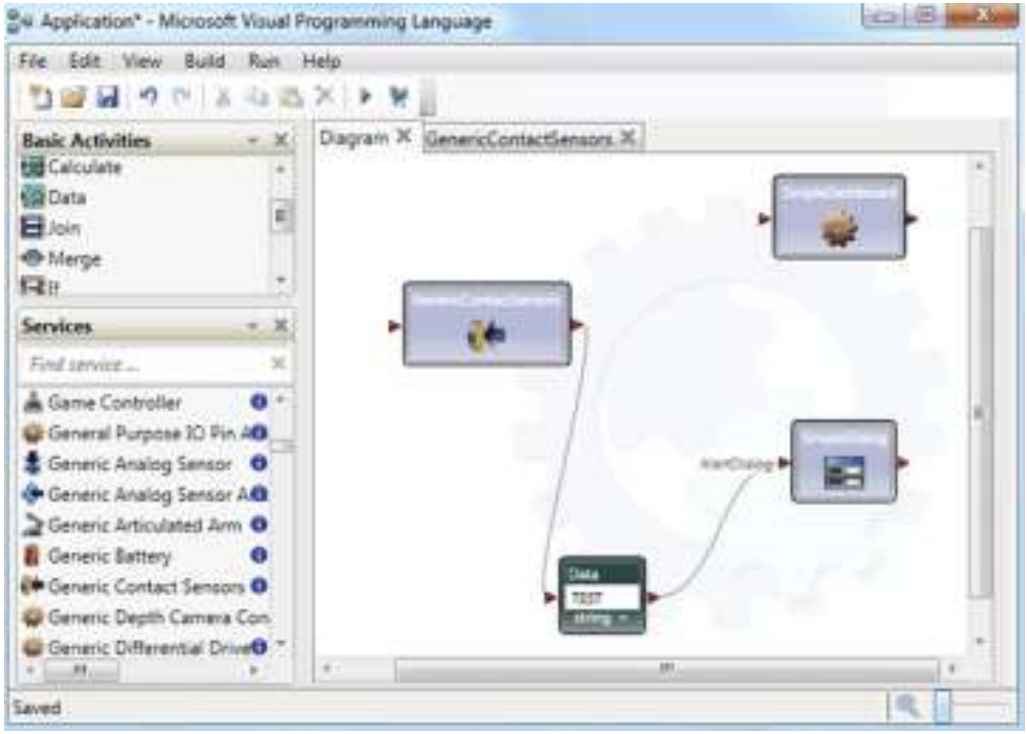


Рис. 8.8. Шаблоны конечного автомата состоят из структуры Case, расположенной внутри цикла While, сдвигового регистра или другой конструкции передачи сообщений между итерациями, соединенной с селектором структуры Case. Каждый случай этой структуры соответствует состоянию приложения

Microsoft Robotics Developer Studio



Qt. Конечные автоматы

```
QStateMachine * machine1 = new QStateMachine(0);
QXtState * xs1 = new QXtState();
xs1->setObjectName("1");
QXtState * xs2 = new QXtState();
xs2->setObjectName("2");
QXtTransition * t;
xt = xs1->addTransition(ui->pushButton, SIGNAL(clicked()), this,
SLOT(doTransiting(QState*,QAbstractState*,QString)), xs2);
xt->setObjectName("A");
xt = xs2->addTransition(ui->pushButton, SIGNAL(clicked()), this,
SLOT(doTransiting(QState*,QAbstr
xt->setObjectName("B");
machine1->addState(xs1);
machine1->addState(xs2);
machine1->setInitialState(xs1);
machine1->start();
```

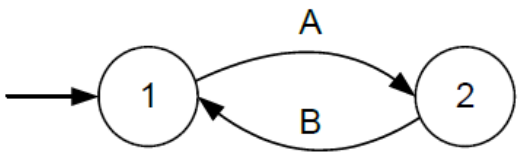
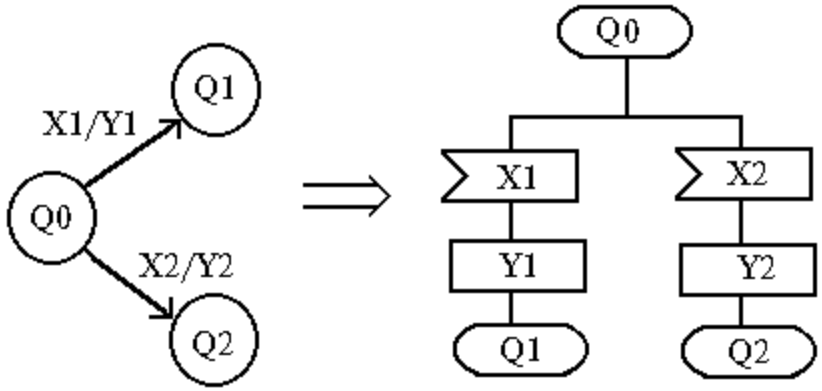


Рис. Простейший конечный автомат

Боровский А.Н. Qt 4.7+. Практическое программирование на C++. – Спб.: БХВ-Петербург. 2012. – 496 с.

SDL

- Язык SDL (Specification and Description Language) был разработан первоначально для проектирования цифровых систем в телефонии (в частности, цифровых АТС). Он основан на модели расширенных конечных автоматов и имеет как графическую, так и эквивалентную ей текстовую нотацию. Описание *процесса* в виде диаграммы состояний и переходов составляется из четырех базовых элементов:



https://studopedia.ru/2_87346_yazik-SDL.html