

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

Вычислительная техника

Направление подготовки 09.04.01 «Информатика и вычислительная техника»

Магистерская программа Вычислительные машины, комплексы, системы и сети

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА МАГИСТРА**

на тему

**Функционально-блочная реализация моделей
переходов состояний и её использование в
системах промышленной автоматике**

Студент

И.Сен 13.06.18 Сенокосов Илья Владимирович
(подпись, дата)

Руководитель

В.Н. 13.06.18 Дубинин В.Н.
(подпись, дата)

Нормоконтролёр

А.В. 13.06.18 Кучин А.В.
(подпись, дата)

Рецензент

КГС. 2 отдела УСС
ООО НТМ. Краткосoft
(должность, место работы) А.Ю. 13.06.18 Афонин А.Ю.
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 25.05.2018 №38-09-12)

Заведующий кафедрой

Д.В. Пашенко Д.В.
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 15.06.18 № 12)

Секретарь ГЭК

Е.И. Гурин Е.И.
(подпись)

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования

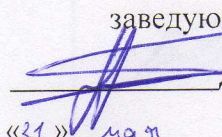
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»

заведующий кафедрой

 Д. В. Пашенко
«21» мая 2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
МАГИСТРА

1. Студент Сенокосов Илья Владимирович гр. 16ВВм1 факультета ВТ
2. Тема работы: «Функционально-блочная реализация моделей переходов состояний и её использование в системах промышленной автоматике»
Тема утверждена приказом ПГУ № _____ от « ____ » _____ 2018
3. Руководитель работы д.т.н., профессор Дубинин В.Н.
4. Задание на работу (назначение разработки, исходные данные и т. п.)

Разработать методики реализации конечных и магазинных автоматов, а также сетей Петри на основе функциональных блоков международного стандарта IEC 61499. Показать применимость разработанных методик в системах промышленной автоматике на конкретных примерах, в частности, в системе выборки последовательностей деталей. Разработать инструментальные средства поддержки разработанных методик.

Перечень подлежащих разработке вопросов

- 1) Обзор и анализ методов мониторинга, контроля, выборки и управления в промышленной автоматике на основе моделей переходов состояний.
- 2) Разработка методики реализации конечных автоматов на основе функциональных блоков.
- 3) Разработка методики реализации магазинных автоматов на основе функциональных блоков
- 4) Разработка методики реализации сетей Петри на основе функциональных блоков

5) Разработка инструментальных средств для преобразования селектирующих конечных автоматов в функционально-блочную модель системы сортировки (выборки) последовательностей деталей.

5. Календарный график по выполнению работы

Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя, консультанта
Обзор и анализ методов мониторинга, контроля, выборки и управления в промышленной автоматике	20%	15.10.16 – 21.12.16	
Разработка методики реализации конечных автоматов на основе функциональных блоков	20%	22.12.16 – 15.06.17	
Разработка методики реализации магазинных автоматов на основе функциональных блоков	20%	16.06.17 – 05.10.17	
Разработка методики реализации сетей Петри на основе функциональных блоков	15%	06.10.17-15.12.17	
Разработка инструментальных средств	15%	16.12.17-20.05.18	
Написание пояснительной записки	10%	21.05.18 – 30.05.18	

Дата выдачи задания «21» мая 2018 г.

Руководитель магистерской работы



(подпись, дата)

Задание к исполнению принял студент



(подпись, дата)

Работу к защите допустить
Декан факультета



13.06.18

РЕФЕРАТ

Магистерская работа содержит 90 листов, 59 рисунков, 39 источника, 2 приложения.

Цель работы — разработка методик функционально-блочной (ФБ) реализации моделей переходов состояний и апробация их использования на примерах из сферы промышленной автоматизации.

В результате проведенных исследований были получены следующие результаты: подход к реализации конечных автоматов на основе ФБ стандарта IEC 61499, методика реализации детерминированных магазинных автоматов на основе ФБ стандарта IEC 61499, методика реализации селектирующих сетей Петри с приоритетами на основе ФБ стандарта IEC 61499, разработан транслятор Prolog-описаний конечных автоматов в функционально-блочную имитационную модель системы сортировки последовательностей деталей, разработана функционально-блочная имитационная модель сборки 2D-изделий, управляемая магазинным автоматом-преобразователем, распознающим язык скобочных выражений, в среде nxtStudio, реализованы пользовательские интерфейсы функционально-блочных моделей конечных автоматов, магазинных автоматов и сетей Петри в среде nxtStudio, позволяющие облегчить процесс тестирования предложенных методик.

По теме диссертации опубликовано 10 научных работ в журналах, сборниках научных трудов и трудах конференций (из них 1 статья опубликована в изданиях, рекомендованных ВАК РФ, и 1 статья в зарубежном журнале, индексируемом в Scopus/Web of Science). Получено свидетельство о гос. регистрации программы для ЭВМ.

Ключевые слова: конечные автоматы, недетерминированные автоматы, магазинные автоматы, сети Петри, функциональные блоки, методика реализации, IEC 61499, nxtStudio, Prolog, транслятор, распознаватель, выборка объектов, система сортировки шаров, система управления.

Содержание

ВВЕДЕНИЕ	3
1. ОБЗОР И АНАЛИЗ МЕТОДОВ МОНИТОРИНГА, КОНТРОЛЯ, ВЫБОРКИ И УПРАВЛЕНИЯ В ПРОМЫШЛЕННОЙ АВТОМАТИКЕ НА ОСНОВЕ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ	7
1.1 Краткий обзор литературы.....	9
1.2 Конечные автоматы	10
1.3 Автоматы с магазинной памятью	11
1.4 Сети Петри и их расширения.....	12
1.5 Функциональные блоки стандарта МЭК 61499	14
1.6 Выводы.....	18
2 РАЗРАБОТКА МЕТОДИК РЕАЛИЗАЦИИ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ В ВИДЕ СИСТЕМ ФУНКЦИОНАЛЬНЫХ БЛОКОВ СТАНДАРТА МЭК 61499	19
2.1 Методика реализации конечных автоматов	19
2.2 Методика реализации магазинных автоматов	34
2.3 Методика реализации сетей Петри	40
2.4 Выводы.....	48
3 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ФУНКЦИОНАЛЬНО-БЛОЧНЫХ РЕАЛИЗАЦИЙ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ	49
3.1 Система сортировки деталей на основе селектирующих конечных автоматов ..	49
3.2 Транслятор Prolog-описаний селектирующих автоматов в имитационную модель системы сортировки деталей	53
3.3 Реализация недетерминированного конечного и магазинного автоматов для выборки деталей и сборки изделий	57
3.4 Реализация системы сборки сложных изделий на основе магазинных автоматов	67
3.5 Реализация сетей Петри для выборки деталей.....	71
3.6 Выводы.....	73
ЗАКЛЮЧЕНИЕ.....	75
СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ	77
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	78
Приложение А – Свидетельство о регистрации программы.....	83
Приложение Б – Грамоты за участие в конференциях	84

ВВЕДЕНИЕ

Актуальность темы. В связи с качественными изменениями в сфере промышленного производства, получившими название четвертой промышленной революции («Индустрия 4.0»), атрибутами которой являются киберфизические системы (КФС) и Интернет вещей, смещением парадигмы производства от массового производства к массовому изготовлению изделий с учетом требований заказчика, роль промышленной автоматизации неизмеримо возрастает. Современные системы промышленной автоматизации (СПА) в большинстве своем представляют КФС, кибернетическая часть которых реализует функции мониторинга, контроля, выборки и управления.

Увеличенная структурная и функциональная сложность систем данного класса, децентрализация функций, широкое использование сетевых технологий, интеграция программной, электронной и механической составляющих в единое целое требует адаптации старых и разработки новых методов и средств их проектирования. При этом возрастает роль формальных моделей и методов в проектировании современных СПА, позволяющих повысить качество проекта и автоматизировать сам процесс проектирования. Наибольшую популярность в проектировании кибернетической части СПА нашли модели дискретных событийных систем, в частности, модели переходов состояний. Несмотря на достигнутые успехи в этом направлении, незначительное внимание было уделено формализации ряда функций СПА, в частности, выборке объектов из входного потока. Существенной проблемой в процессе проектирования является то, что в большинстве случаев существует семантический разрыв между формальной моделью и реализацией, что требует разработки определенных методов и методик реализации, отвечающих на вопрос, как преобразовать формальную модель в исполнимый код.

Не последним фактором, оказывающим влияние на СПА и процесс их проектирования и реализации, является появление новых стандартов, языков, технологий и платформ. Новый международный стандарт МЭК 61499,

пришедший на смену стандарту МЭК 61131, ориентирован на построение распределенных систем управления промышленными процессами. По сути дела, стандарт IEC 61499 вводит класс систем управления нового поколения.

С учетом вышесказанного, тема исследований, связанная с разработкой методик реализации моделей переходов состояний на основе функциональных блоков стандарта МЭК 61499 и их использованием в СПА является актуальной.

Цели и задачи исследования. Целью работы является разработка методик функционально-блочной реализации моделей переходов состояний и апробация их использования на примерах из сферы промышленной автоматике.

Для достижения поставленной цели необходимо решить следующие основные задачи:

- 1) Разработать методики реализации конечных автоматов на функциональных блоках международного стандарта IEC 61499;
- 2) разработать методику реализации магазинных автоматов на функциональных блоках международного стандарта IEC 61499;
- 3) разработать методику реализации сетей Петри на функциональных блоках международного стандарта IEC 61499;
- 4) Разработать визуальную имитационную модель системы сборки сложных изделий на основе магазинных автоматов для тестирования предложенных методик в системе nxtStudio;
- 5) Разработать транслятор представления конечных автоматов на языке Prolog в XML-представление функциональных блоков стандарта IEC 61499.

Объектом исследования являются системы мониторинга, контроля, выборки и управления в промышленной автоматике.

Предметом исследования являются методы реализации автоматных моделей и сетей Петри на функциональных блоках международного стандарта IEC 61499, ориентированные на использование в системах промышленной автоматике.

Научная новизна. Научная новизна определяется следующими результатами:

1) разработан подход к реализации конечных автоматов на основе ФБ стандарта IEC 61499, отличающийся использованием механизма передачи маркеров, что позволяет реализовывать как детерминированные, так и недетерминированные автоматы;

2) разработана методика реализации детерминированных магазинных автоматов (ДМА) на основе ФБ стандарта IEC 61499. *Особенностью* методики является использование представления ДМА в виде диаграммы переходов состояний и механизма передачи маркеров, что *позволяет* использовать подходы, используемые при функционально-блочной реализации конечных автоматов;

3) предложена методика реализации селектирующих сетей Петри с приоритетами на основе ФБ стандарта IEC 61499. *Особенностью* методики является использование в функционально-блочной реализации блока-диспетчера, что *позволяет* легко изменять логику функционирования сети в процессе функционирования.

Практическая ценность работы. Практические результаты, полученные в результате исследований:

1) разработан транслятор Prolog-описаний конечных автоматов в функционально-блочную имитационную модель системы сортировки последовательностей деталей, *позволяющий* автоматизировать процесс создания управляющих приложений для систем данного класса и средств их валидации;

2) разработана функционально-блочная имитационная модель сборки 2D-изделий, управляемая магазинным автоматом-преобразователем, распознающим язык скобочных выражений, в среде nxtStudio, *позволяющая* оценить полезность использования магазинных автоматов-преобразователей в системах сборки изделий;

3) реализованы пользовательские интерфейсы функционально-блочных моделей конечных автоматов, магазинных автоматов и сетей Петри в среде nxtStudio, *позволяющие* облегчить процесс тестирования предложенных методик.

Апробация работы. Основные научные положения и результаты работы докладывались и обсуждались на следующих научно-технических конференциях: 8-й международной конференции «Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS-2017)» (г. Лион, Франция, 2017), 2-й и 3-й Международной научно-технических и научно-методических конференциях «Современные технологии в науке и образовании (СТНО)» (г. Рязань, 2017, 2018), 13-й и 14-й международных научно-технических конференциях «Новые информационные технологии и системы (НИТиС)» (г. Пенза 2016, 2017), международной научно-технической конференции «Современные информационные технологии (СИТ-2017)» (г. Пенза, 2017), 4-й и 5-й ежегодных межвузовских научно-практических конференциях «Информационные технологии в науке и образовании. Проблемы и перспективы» (г. Пенза 2017, 2018).

Публикации. По теме диссертации опубликовано 10 научных работ в журналах, сборниках научных трудов и трудах конференций (из них 1 статья опубликована в изданиях, рекомендованных ВАК РФ, и 1 статья в зарубежном журнале, индексируемом в Scopus/Web of Science). Получено свидетельство о гос. регистрации программы для ЭВМ. В процессе регистрации находятся еще один программный продукт «Функционально-блочная реализация магазинного автомата для распознавания языка скобочных арифметических выражений».

Структура и объем диссертационной работы. Работа состоит из введения, трёх глав, заключения, изложенных на 73 страницах, списка литературы из 39 наименований, 2 приложения и содержит 59 рисунков.

1. ОБЗОР И АНАЛИЗ МЕТОДОВ МОНИТОРИНГА, КОНТРОЛЯ, ВЫБОРКИ И УПРАВЛЕНИЯ В ПРОМЫШЛЕННОЙ АВТОМАТИКЕ НА ОСНОВЕ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ

В Инициативе интегрированной технологии производства [1] определены следующие «Большие вызовы» в отношении успешного производства XXI века: 1) экономичные и эффективные предприятия; 2) предприятия, «чувствительные» к клиенту; 3) полностью связанные предприятия; 4) экологическая устойчивость; 5) управление знаниями; 6) эксплуатация технологий. Второй «вызов» показывает, что современная парадигма производства смещается от массового производства к массовому изготовлению изделий с учетом требований заказчика. В результате чего чувствуется необходимость быстрой перенастройки процесса изготовления изделий. Поэтому требуется разработка методов и средств проектирования на основе высокоуровневых формальных спецификаций с автоматическим их преобразованием в исполнимый код контроллеров нижних уровней управления. Также существует необходимость в создании методов организации техпроцессов для возможности их быстрой перестройки, чтобы выпускать новые изделия.

Для некоторых производственных систем является важной последовательность, которая подается на вход их компонентов. Например, в простейшей системе сборки после подачи болта М4 должна последовать подача шайбы, а затем и гайки М4. В химической, пищевой и фармакологической промышленности важным является последовательность подачи компонентов в раствор. При формировании заданных наборов деталей в упаковочном производстве также требуется выделение из общего потока разнообразных деталей, которые идут «внавал», заданных последовательностей деталей для их оптимальной упаковки в контейнеры. Кроме того, для сборочного производства задача распознавания определенной последовательности движущихся по конвейеру деталей, имеющей заданную структуру, может стать вполне

актуальной для принятия решения о дальнейшем ходе техпроцесса сборки. Именно поэтому важной составной частью производственных систем наряду с центрами обработки деталей становятся также центры сортировки и выборки деталей. В мехатронных системах системы сортировки непосредственно интегрированы с системами управления технологическим оборудованием, которые, как правило, реализуются на основе ПЛК. В качестве примера можно взять систему сортировки разноцветных шаров, разработанную на симуляторе *EasyVEEP* [2]. В данной системе шары трех цветов направляются в хранилища в соответствии со своим цветом.

Следует отметить, что в информатике упорядочивание и сортировка, как правило, позволяет существенно сократить время решения различных алгоритмических задач. Этого можно ожидать и относительно процессов неинформационной природы. В будущем процессы обработки, транспортировки и складирования материалов и процессы обработки, передачи и хранения информации и материальных объектов будут постоянно сливаться. Предпосылкой для этого может служить «Интернет вещей» [3], где уникальный идентификатор, который может обрабатываться информационными процессами, присвоен каждому материальному объекту. Всё это вкладывается в концепцию киберфизических систем, являющихся атрибутом четвертой промышленной революции (известной также как «Индустрия 4.0»).

Сборка также является одной из ключевых областей в производстве, так как с ростом массового потребительского спроса на продукты, сделанные по индивидуальному заказу, процесс сборки должен справляться со всем набором разнообразных продуктов. Обычно одна и та же производственная линия используется для сборки семейства продуктов, которые имеют некоторые общие свойства, называемого также линейкой продуктов. Очевидно, что программное обеспечение данных производственных линий становится сложнее, поэтому требуется автоматизация проектирования некоторых его этапов. Естественно, что дальнейшие исследования в данном направлении

будут направлены в усложнение линейки продуктов. Задача выборки определенной последовательности объектов из входного потока объектов является довольно важной как с теоретической, так и практической точек зрения. Данная задача может, например, возникнуть в сборочном производстве, при сортировке и упаковке контейнеров различных видов, включая наборы деталей, инструментов и лекарств. В данном случае упорядоченность – это довольно важное свойство продукта. Кроме того, данная задача может быть использоваться для контроля правильности функционирования системы в том смысле, удовлетворяет ли поведение системы некоторому заданному шаблону. В этом случае из потока событий или сигналов, поступающих из дискретной событийной системы, выделяются заданные последовательности, определенные неким формальным образом. По детектированию специфицированных последовательностей можно судить о нормальном или, наоборот, ненормальном функционировании системы. Задача выборки специфицированной последовательности объектов является схожей с задачей распознавания языков, встречающейся, в частности, при построении трансляторов и компиляторов. В отличие от распознавателя, система выборки может игнорировать входные объекты, если они не вносят в неё соответствующие изменения. Необходимо также отметить значительно более строгие требования по времени, предъявляемые к системам выборки, поскольку они зачастую работают в режиме реального времени.

1.1 Краткий обзор литературы

Задачи выборки не так многогранно представлены в литературе, как задачи распознавания языков. Использование конечных детерминированных автоматов для выборки последовательностей деталей представлено в [4], конечных недетерминированных автоматов – в [5], магазинных автоматов – в [6] и сетей Петри в [7]. Необходимо отметить, что не удалось обнаружить исследования, напрямую связанные с данными проблемами. Для распознавания

объектов с использованием конечных автоматов наиболее близкой является работа [8], в которой представлен алгоритм для распознавания объектов на основе данных, поступающих из множества датчиков. Общая стратегия для извлечения свойств из различных источников реализуется в качестве конечного автомата. Для распознавания используется дерево решений. Найдено несколько работ для распознавания ситуаций и паттернов с помощью сетей Петри. В работе [9] предлагаются структурно простые сети Петри (S-сети), а для распознавания применяется алгоритм на основе конечных автоматов, имеющий полиномиальную сложность. В работе [10] сети Петри используется для конструирования запроса к данным видеонаблюдения и далее для распознавания данного специфицированного шаблона. В работе [11] сети Петри предлагается использовать для распознавания и моделирования жестов с несколькими касаниями для мультисенсорных устройств. Авторы работы [12] предлагают раскрашенные сети Петри для моделирования распознавания последовательностей событий, упорядоченных во времени, выраженных с помощью логических и временных операторов, а также максимальных и минимальных времен задержки. В работе [13] раскрашенные сети Петри используются для определения правильности выполнения оператором (в данном случае, пилотом) соответствующего руководства.

1.2 Конечные автоматы

Конечный автомат [14] – это пятерка (Q, X, q_0, Ψ, F) , где

Q – конечное множество состояний;

X – конечный входной алфавит;

q_0 – начальное состояние;

Ψ – функция переходов. Аргументами функции являются текущее состояние и входной символ, а значением – новое состояние;

F – множество конечных состояний.

Конечные автоматы могут быть детерминированными (ДА) и недетерминированными (НДА). НДА отличаются от детерминированных конечных автоматов функцией переходов. Если в ДА значением функции переходов является состояние, то в НДА – множество состояний. Любой конечный автомат можно преобразовать к детерминируемому виду. Такое преобразование называется детерминизацией [15]. Конечных автоматы могут использоваться для задания регулярных языков, так и для распознавания входных последовательностей данных языков.

Теорией конечных автоматов можно описать достаточно сложные цепочки деталей, которые подходят для построения различных видов систем сортировки и выборки, а также для использования в качестве управляющего объекта. Более подробная информация о конечных автоматах приведена в работе [14].

1.3 Автоматы с магазинной памятью

Магазинный автомат (МА) [14] можно представить, как кортеж следующего вида:

$$MA = (Q, V, Z, T, \delta, q_0, z_0, F),$$

где $Q = \{q_1, q_2, \dots, q_n\}$ – конечное множество состояний автомата;

$V = \{v_1, v_2, \dots, v_m\}$ – конечное множество входных символов;

Z – конечный магазинный алфавит;

$\delta: Q \times (V \cup \{\lambda\}) \times Z \rightarrow 2^{Q \times Z^*}$ – функция переходов автомата;

$q_0 \in Q$ – начальное состояние автомата;

$z_0 \in Z$ – начальный магазинный символ;

$F \subseteq Q$ – множество конечных состояний.

Магазинные автоматы, как и конечные, делятся на детерминированные и недетерминированные МА. В недетерминированных МА из одной и той же конфигурации возможен более чем один переход. Магазинные автоматы могут использоваться как для задания контекстно-свободных (КС) языков, так и для

распознавания входных последовательностей данных языков. При этом сложность алгоритма распознавания – экспоненциальная, что является недопустимым в системах управления технологическими процессами реального времени. Поэтому для практических приложений допустимо использование только класса детерминированных КС-языков, распознавателями для которых являются детерминированные МА. Для таких языков существует алгоритм работы распознавателя с квадратичной сложностью.

Следует отметить, что МА возможно представить в графическом виде с использованием диаграмм переходов [14]. Графическое представление более удобно для восприятия для человека. Поэтому многие системы моделирования на основе МА, в том числе *JFLAP* используют именно графическое представление магазинных автоматов [16].

1.4 Сети Петри и их расширения

Сети Петри (СП) – это одна из наиболее популярных моделей, используемых в вычислительной технике [17]. Сеть Петри – это четверка:

$C = \{P, T, I, O\}$, где:

P – конечное множество позиций;

T – конечное множество переходов;

I – множество входных функций переходов, $I: P \rightarrow T$;

O – множество выходных функций переходов, $O: T \rightarrow P$.

В данной работе также рассматривается использование расширения СП (названных селективирующими СП) для выборки из входного потока параметризованных объектов вида $\langle a_1, a_2, \dots, a_n \rangle$. В качестве основы математического аппарата выбраны А-сети, первоначально использованные в работе [18] для моделирования NCES-сетей. А-сети являются расширением сетей Петри в направлении повышения моделирующих возможностей за счет нагрузки дуг и усложнения правил разрешённости и срабатывания переходов,

это позволяет эффективно моделировать обработку целочисленных переменных.

Селектирующая сеть Петри представляет собой следующий кортеж:

$(P, T, X, Y, Z, U, W_X, W_Y, W_Z, Q, A, G, m_0, T_F)$, где:

P – конечное множество позиций;

T – конечное множество переходов;

$X \subseteq P \times T$ – множество входных дуг (функций) переходов с порогом-минимумом (дуги с проверкой на “больше”);

$Y \subseteq T \times P$ – множество выходных дуг переходов;

$Z \subseteq P \times T$ – множество входных дуг переходов с порогом-максимумом (дуги с проверкой на “меньше”);

$U \subseteq (X \cup Z) \times Y$ – отношение сопряженности дуг;

$W_X: X \rightarrow N_0 \times \{N_0 \cup all\}$ – функция весов дуг типа X ;

$W_Y: Y \rightarrow \{N^+ \cup *\} \times \{add, reset\}$ – функция весов дуг типа Y ;

$W_Z: Z \rightarrow N^+ \times \{N_0 \cup all\}$ – функция весов дуг типа Z ;

$Q: T \rightarrow N_0$ – функция приоритетов переходов. Должно выполняться ограничение $\forall t_i, t_j \in T [Q(t_i) \neq Q(t_j)]$. Это ограничение делает функционирование селектирующей СП *детерминированным*;

A – множество атрибутов (параметров) объекта. Обозначим $\bar{A} = \bigcup_{a \in A} (a \times D^a)$ –

пространство состояний параметров, D^a – домен параметра a ;

G – множество сторожевых функций переходов. Каждая функция $g^t \in G$ определяется как $g^t: [\bar{A}] \rightarrow \{true, false\}$, где $[\bar{A}] = \prod_{a \in A} (\{a\} \times D^a)$ – множество

кортежей, которые содержат все комбинации значений переменных, помеченных именами переменных; m_0 – начальная маркировка; $T_F \subseteq T$ – множество финальных (конечных) переходов. Срабатывание любого финального перехода свидетельствует о том, что выборка последовательности объектов окончена.

Первый компонент веса дуги типа *X* указывает минимальный порог для срабатывания перехода, второй – число удаляемых меток из инцидентной дуге позиции. Если второй компонент равен *all*, то удаляются все метки. В отношении дуги типа *Y*: первый компонент веса дуги определяет число меток, которые будут добавлены в инцидентную дуге позицию, а второй компонент указывает режим добавления – с предварительным обнулением маркировки этой позиции (слово *reset*) или нет (слово *add*). Если первый компонент равен *, то число добавляемых в выходную позицию перехода меток вычисляется и зависит от маркировок определенных входных позиций этого перехода. Первый компонент веса дуги типа *Z* определяет максимальный порог для срабатывания перехода, а второй – число удаляемых меток из инцидентной дуге позиции.

Вследствие этого, переход разрешен, если истинно сторожевое условие и имеется концессия по меткам. Следует отметить, что сочетания весов дуг образуют некоторые паттерны дуг, которые имеют собственную семантику – числовую дугу, пороговую и безусловную дугу сброса, тестирующую дугу с проверкой на “больше или равно”, ингибиторную дугу (дугу с проверкой на “меньше”), сопряженные дуги для безусловного добавления/копирования/передвижения меток.

1.5 Функциональные блоки стандарта МЭК 61499

Все модели переходов состояний в данной работе реализованы в виде функциональных блоков стандарта IEC 61499 (МЭК-61499) [19-20]. Международный стандарт IEC 61499 используется для создания систем управления промышленными процессами нового поколения [21]. Рабочая версия стандарта была представлена в 2005 году. Стандарт IEC-61499 состоит из 4 частей:

- часть 1. Архитектура и функциональные блоки;
- часть 2. Требования к программным средствам;
- часть 3. Справочная информация;

- часть 4. Взаимодействие функциональных блоков для производства и систем контроля.

Стандарт IEC 61499 вводит следующие понятия [22]:

- Система (System) – это набор устройств, связанных и взаимодействующих друг с другом с помощью коммуникационной сети, состоящей из сегментов и соединений.
- Устройство (Device) – это независимая физическая единица, ограниченная интерфейсом устройства, которая способна выполнить одну или более заданных функций в конкретном контексте.
- Ресурс (Resource) – это функциональная единица, имеющая независимое управление и обеспечивающая различные сервисы для приложений, включая выполнение и планирование алгоритмов.
- Приложение (Application) – это программная функциональная единица, предназначенная для решения проблем в системе управления и измерения. Приложения могут быть распределены между устройствами и также могут взаимодействовать с другими приложениями.
- Функциональный блок (Function block) – это программная функциональная единица, которая является наименьшим элементом в распределенной системе управления.

Чтобы определить ФБ по стандарту IEC-61499, необходимо указать следующие необходимые параметры:

- Входные и выходные переменные функционального блока;
- Вызываемые алгоритмы;
- События, которые генерируются в каждом конкретном случае;
- Диаграмму управления выполнением (ECC — Execution Control Chart). Диаграмма выполнения событий или ECC представляет собой некое описание реакции на события, то есть набор действий, который необходимо выполнить при поступлении определенного сигнала. Эта диаграмма может быть представлена в виде конечного автомата.

Функциональные блоки стандарта IEC-61499 поддерживают такое понятие, как интерфейс обслуживания, который позволяет четко описать необходимую последовательность действий. Все описания функциональных блоков хранятся в XML-формате.

Функциональные блоки стандарта IEC-61499 являются зависимыми от событий (активируются с приходом события). ФБ стандарта IEC-61499 может и принимать и отправлять события.

Функциональные блоки стандарта IEC-61499 можно поделить на следующие типы:

- базовые – простые ФБ, где внутренняя реализация состоит только из диаграммы ЕСС и алгоритмов;
- композитные – представляющие набор базовых ФБ,
- интерфейсные – ФБ с графическим интерфейсом.

Представление базового ФБ стандарта IEC-61499, как наиболее часто используемого, представлено на рисунке 1.

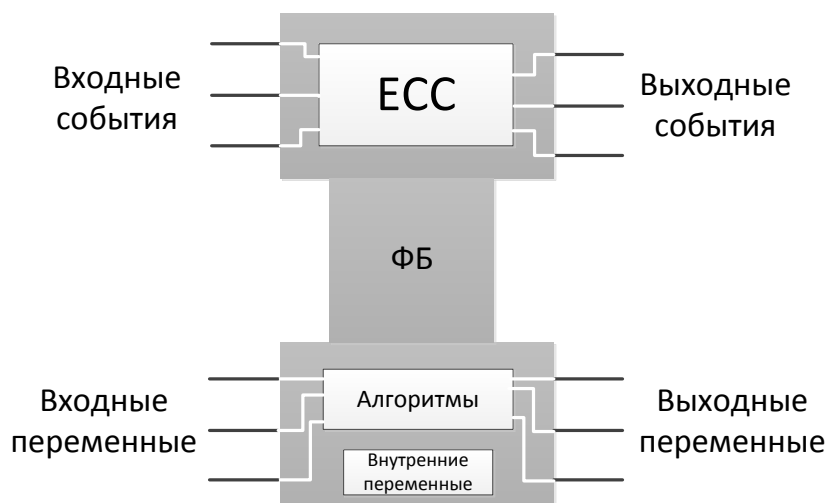


Рисунок 1 – Представление ФБ на основе стандарта IEC-61499

Выполнение функционального блока стандарта IEC-61499 можно представить в виде следующих пунктов:

1. инициализация;
2. проверка появившихся событий;
3. выполнение необходимого алгоритма;
4. генерация событий на выход.

Для разработки функциональных блоков стандарта IEC 61499 использовалась система nxtStudio [23]. Данная система была выбрана из-за возможности разработки ФБ и поддержки визуального моделирования полученных результатов. Интерфейс системы nxtStudio представлен на рисунке 2.

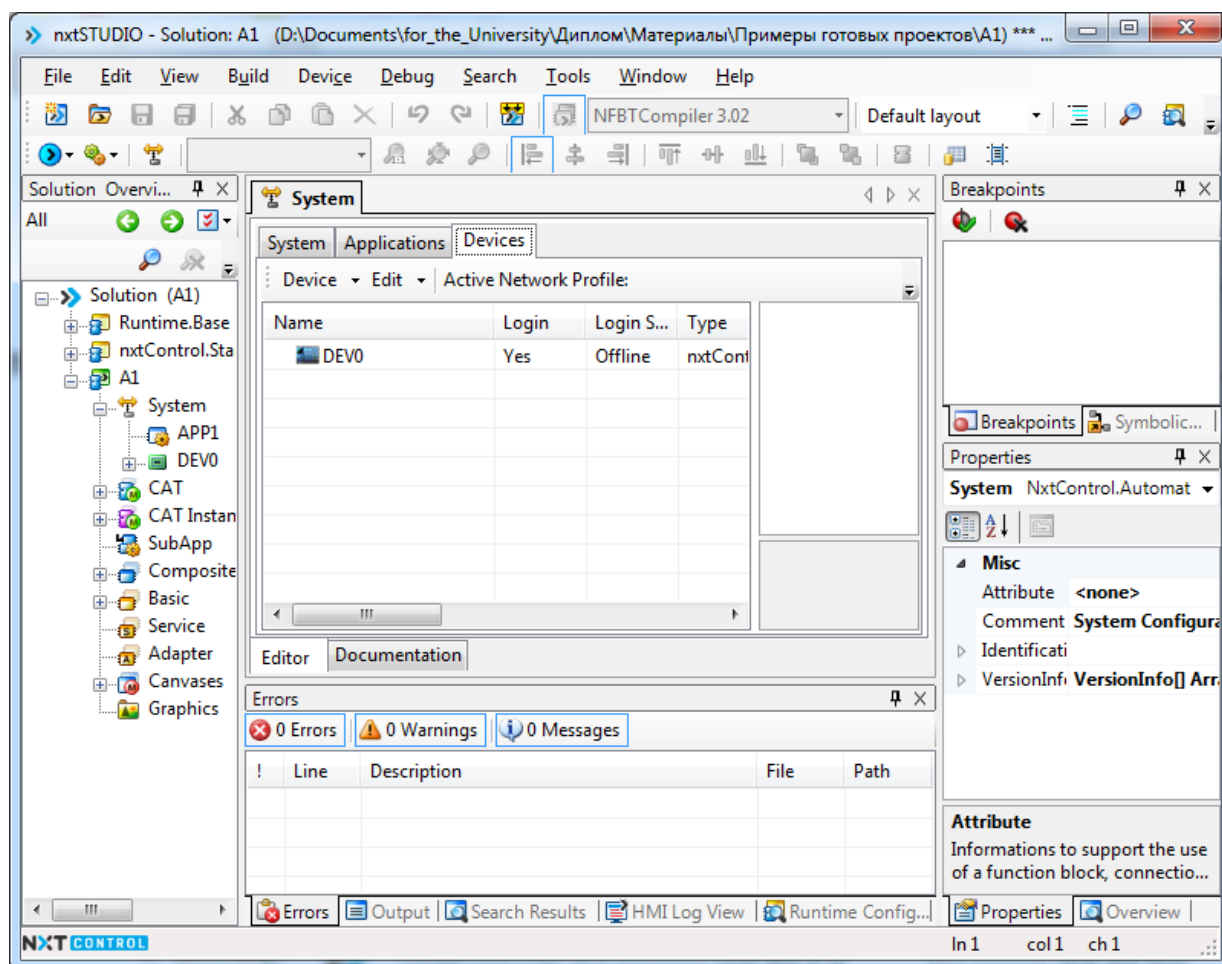


Рисунок 2 – Система nxtStudio

Представленная система является наиболее эффективным средством разработки систем управления. Также nxtStudio предоставляет средства для наиболее эффективного проектирования распределенных систем. Система легко интегрирует все задачи автоматизации в одном инструменте, предлагает сборник библиотек программного обеспечения и использует стандарты IEC 61499, а также IEC 61131 в качестве парадигмы управления.

1.6 Выводы

1. На основе анализа литературы определено, что модели переходов состояний, включая конечные автомат и сети Петри, являются наиболее используемыми моделями в проектировании кибернетической части систем промышленной автоматизации. Наиболее популярной моделью являются конечные автоматы и их производные, используемые как для моделирования объекта управления, так и для программирования контроллеров.

2. Важной операцией в сборочном производстве является выборка деталей из входного потока (например, с конвейера). Задачи выборки объектов не так многогранно представлены в литературе, как задачи распознавания языков, в частности, отсутствуют какие либо формальные модели, описывающие выборку объектов из входного потока.

3. Мониторинг и контроль функционирования системы промышленной автоматизации может быть основан на современной технологии проверки соответствия модели реальному поведению системы (conformance checking).

4. Полезной моделью для выборки деталей и проверке соответствия являются магазинные автоматы, позволяющие распознавать контекстно-свободные (КС) языки. Магазинные автоматы-преобразователи позволяют трансформировать цепочки КС-языка в цепочки другого языка, например, языка инструкций сборочного робота.

5. Существует семантический разрыв между формальной моделью управляющего алгоритма и его реализацией на конкретном языке программирования управляющих систем. Для построения «моста» между двумя абстрактными системами требуется разработка соответствующих методик реализации.

6. Основной тенденцией в сфере промышленной автоматизации является переход от централизованных систем к распределенным системам. Ключом к построению современных распределенных систем управления промышленной автоматизации является международный стандарт IEC 61499.

2 РАЗРАБОТКА МЕТОДИК РЕАЛИЗАЦИИ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ В ВИДЕ СИСТЕМ ФУНКЦИОНАЛЬНЫХ БЛОКОВ СТАНДАРТА МЭК 61499

2.1 Методика реализации конечных автоматов

Рассмотрим подходы к реализации конечных автоматов на основе функциональных блоков (ФБ) международного стандарта IEC 61499 [24]. Функционально-блочная реализация автомата может быть легко интегрирована в общую систему управления выборкой деталей или сортировкой на основе того же стандарта.

ДА может быть напрямую реализован с использованием диаграммы управления выполнением (диаграммы ЕСС) базисного ФБ.

Можно выделить следующие подходы к реализации НДА и ДА на ФБ:

- 1) переходы представлены в виде ФБ;
- 2) состояния представлены в виде ФБ.

Данные подходы по своей сути являются эквивалентными, но реализация переходов удобнее для представления петель и, кроме того, для представления переходов можно использовать только один универсальный базисный ФБ, поэтому в дальнейшем используем данный подход.

Вначале, рассмотрим реализацию ДА. Для моделирования функционирования селектирующего конечного автомата предлагается использовать механизм передачи маркеров. Это универсальный подход по отношению к ДА и НДА. Маркер – это динамический объект, который может быть передан из одного ФБ в другой ФБ. Если ФБ имеет маркер, то он может активно контролировать входные объекты, и в случае, когда объект и собственно ФБ имеют одинаковые параметры, тогда ФБ инициирует действие для выборки этого объекта. Возможно наличие маркеров в один и тот же момент времени в разных функциональных блоках. Из совокупности всех ФБ с маркером выбрать текущий объект могут только те ФБ, которые настроены на прием объектов данного типа.

Как было сказано выше, для каждого перехода будет использоваться универсальный ФБ, который будет настраиваться под конкретные контекстные условия. Интерфейс данного ФБ представлен на рисунке 3.

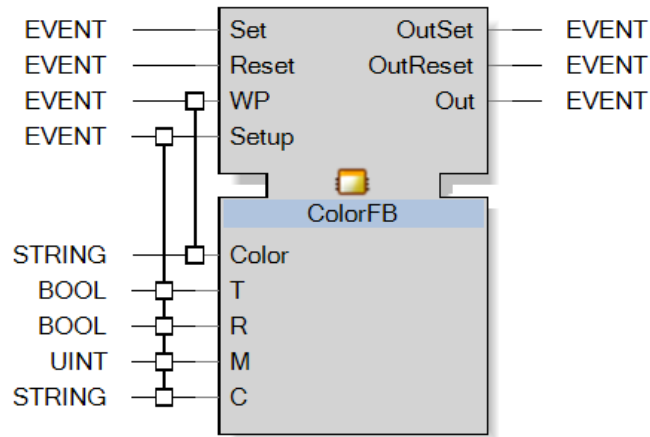


Рисунок 3 – Интерфейс базисного ФБ для моделирования переходов недетерминированного автомата

Интерфейс приведённого базисного ФБ состоит из следующих групп параметров:

1) Входные сигналы:

- *Set* – сигнал установки маркера в данном ФБ;
- *Reset* – сигнал удаления маркера из данного ФБ;
- *WP* – сигнал поступления объекта в систему выборки или сортировки;

или сортировки;

- *Setup* – сигнал, установки (сброса) начальных параметров ФБ.

2) Выходные сигналы:

- *OutSet* – сигнал, устанавливающий маркер в других ФБ;
- *OutReset* – сигнал, сбрасывающий маркер в других ФБ;
- *Out* – сигнал, означающий, что ФБ обработал поступивший объект.

3) Входные переменные:

- *Color* – в данном случае цвет объекта, но возможно наличие другого параметра или группы параметров для выборки;
- *T* – флаг наличия маркера в ФБ;
- *R* – флаг итерации, указывающий возможность принятия ФБ нескольких объектов;
- *M* – максимальное число объектов заданного типа, которые может принять ФБ перед посылкой маркера другим ФБ;
- *C* – цвет объекта, которые может принять ФБ (иначе, «цвет» ФБ), также в этом случае может использоваться другой параметр или группа параметров.

Диаграмма управления выполнением (рисунок 4) (ЕСС) данного базисного ФБ состоит из шести состояний:

- 1) состояние *Reset*, в котором сбрасывается маркер ФБ;
- 2) состояние *Setup*, в котором устанавливаются начальные значения внутренних переменных;
- 3) состояние *Set*, в котором устанавливается маркер ФБ;
- 4) состояние *WP*, в которое автомат переходит при выполнении следующего условия: $WP \ \& \ (T1 = TRUE) \ \& \ (Color = C1)$. Переход в данное состояние произойдет, если в систему поступит объект с параметром, который обрабатывается в ФБ, и данный ФБ имеет маркер;
- 5) состояние *WP1*, в котором увеличивается счетчик количества обработанных деталей. В данное состояние может попасть ФБ, имеющий $R=true$. Кроме того, в данном состоянии выдается выходной сигнал *Out*;
- 6) состояние *WP2*, в котором происходит удаление маркера и выдача всех требуемых выходных сигналов. В данное состояние попадает ФБ, имеющий $R=false$, или ФБ, который уже принял *M* деталей заданного цвета.

Из базисных ФБ происходит построение составного ФБ, который уже представляет автомат, распознающий или выбирающий последовательности объектов.

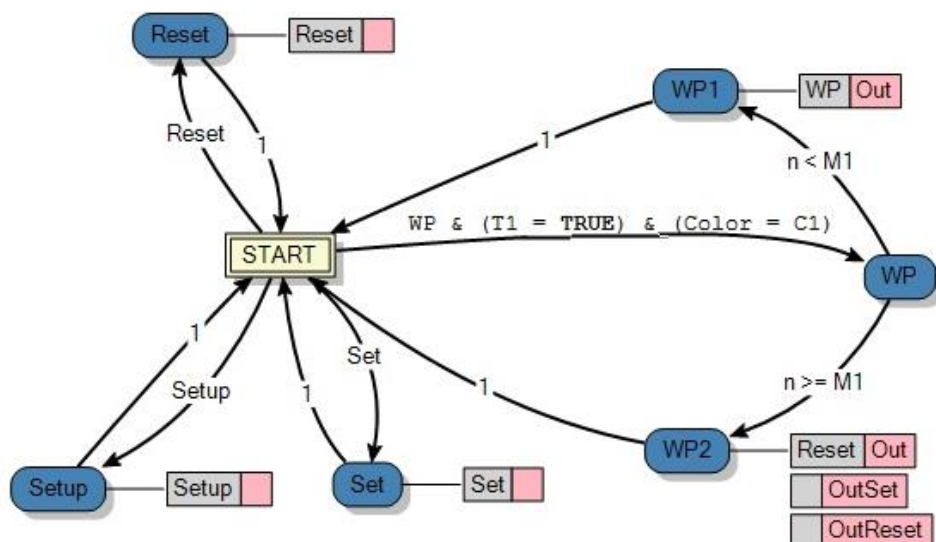


Рисунок 4 – ЕСС базисного ФБ, моделирующего переход автомата

Для преобразования детерминированного автомата в структуры функционально-блочной реализации, предлагается использовать механизм графовых трансформаций [25]. Преимущество такого подхода – это формальность, наглядность и наличие инструментальных средств для проведения трансформации. Кроме того, в ряде случаев из элементов исходной системы можно напрямую вывести элементы целевой системы. Набор правил для вывода элементов целевой системы ФБ из элементов исходного недетерминированного автомата приведен ниже:

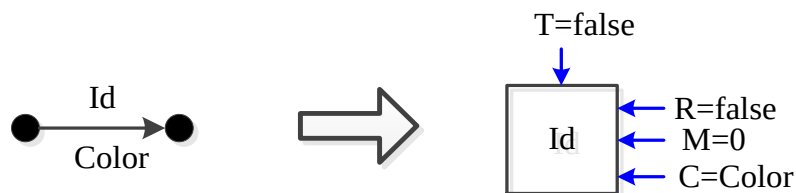


Рис. 5. Правило вывода ФБ, не имеющего начального маркера и итерации, использующее переход автомата, начинающийся не в начальном состоянии

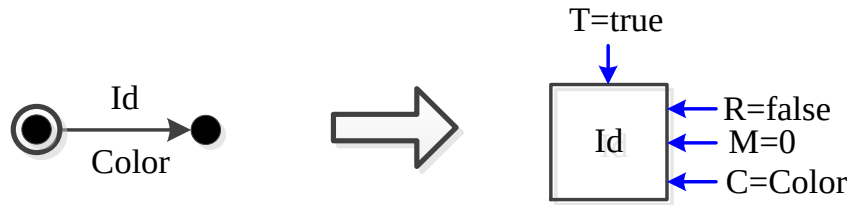


Рис. 6. Правило вывода ФБ, имеющего начальный маркер, но не имеющего итерации, использующее переход автомата, начинающийся в начальном состоянии

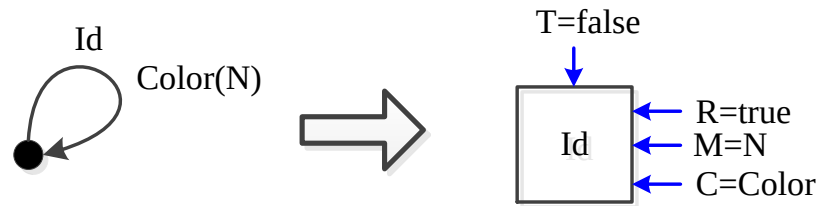


Рис. 7. Правило вывода ФБ, не имеющего начального маркера, но имеющего итерацию, использующее переход автомата в виде петли, начинающейся не в начальном состоянии

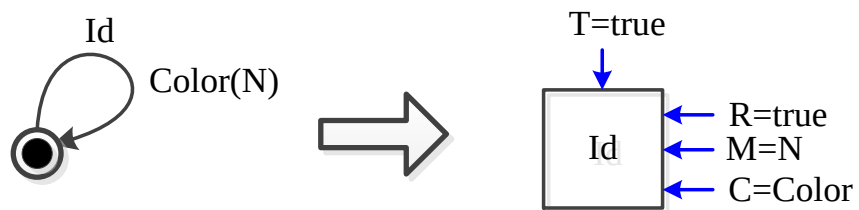


Рис. 8. Правило вывода ФБ, имеющего начальный маркер и итерацию, использующее переход автомата в виде петли, начинающейся в начальном состоянии

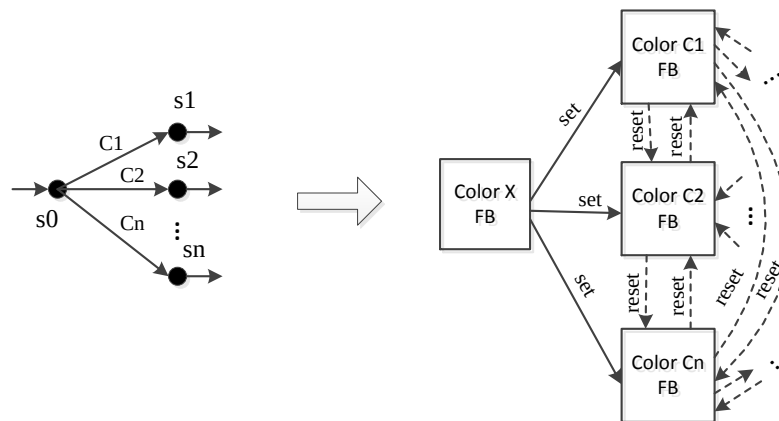


Рис. 9. Обобщенное правило преобразования автоматной конструкции «Выбор»

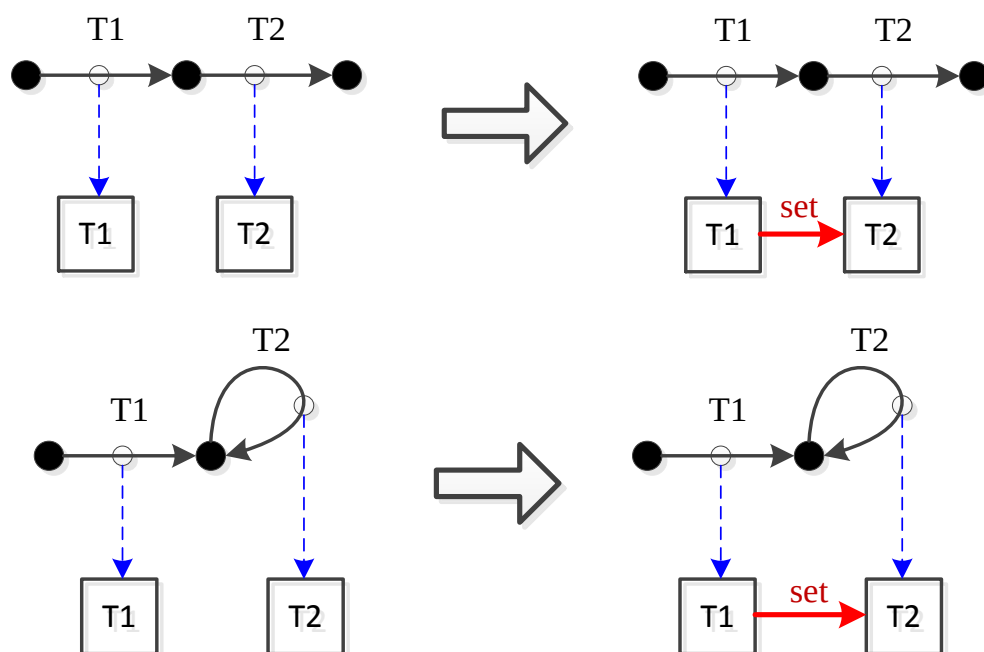


Рис. 10. Правила вывода дуги *set*, использующие смежные переходы автомата

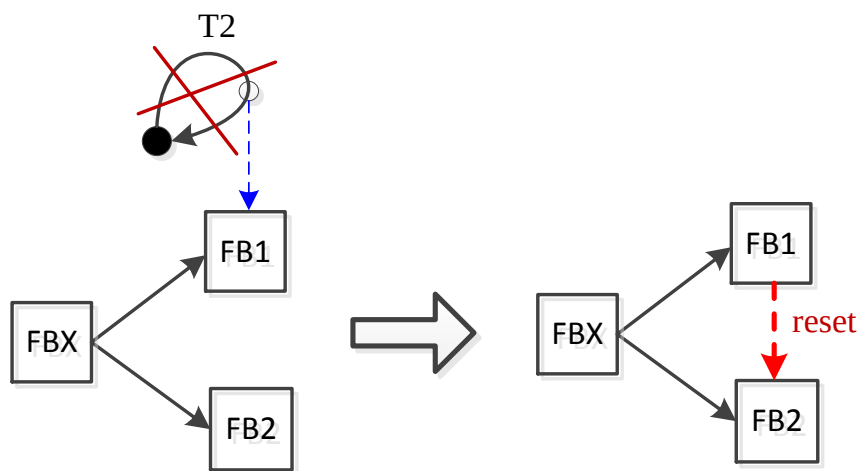


Рис. 11. Правила вывода дуги *reset*

Как видно из рисунков 5-8, параметры *C* и *M* для генерируемых ФБ выводятся из разметки автоматного перехода. Приведенные правила даны для случая обычных переходов и их необходимо слегка модифицировать в случае использования конечных (*final*) переходов автомата. Модификации будут в основном касаться добавления в генерируемый ФБ выходного сигнала *EOS*.

Необходимо отметить, что ДА можно напрямую реализовать с использованием диаграммы управления выполнением (диаграммы ЕСС) базисного ФБ и алгоритмов. Однако реализация НДА требует более сложных подходов.

Можно выделить две семантики выполнения НДА, что касается условий сохранения (локальных) состояний НДА:

1) состояние сохраняется, если в нём имеется петля, которая специфицирует условие сохранения этого состояния [26];

2) состояние сохраняется, если все исходящие из данного состояния переходы не являются разрешёнными.

В Семантике 1 явно специфицируется сохранение состояния, а неявно – сброс, а в семантике 2 наоборот – явно специфицируется сброс состояния, а неявно – сохранение. Необходимо отметить, что Семантика 1 концентрируется на состояниях, а Семантика 2 – на переходах (практически также как в сетях Петри). Необходимо отметить, что на практике используются обе семантики. Использование той или иной семантики во многом определяется формой представления НДА. При представлении НДА в виде системы канонических уравнений (СКУ) используется Семантика 1, в тоже время как при использовании параллельных граф-схем алгоритмов (ГСАП) – Семантика 2.

Существуют правила преобразования НДА, который работает в соответствии с одной семантикой, в НДА, который работает в соответствии с другой семантикой. На рисунке 12 приведено правило приведения НДА к Семантике 1. А на рисунке 12 показано, что в преобразованном НДА появляется петля, которая явно специфицирует условие сохранения события, когда условия на исходящих дугах ложны.

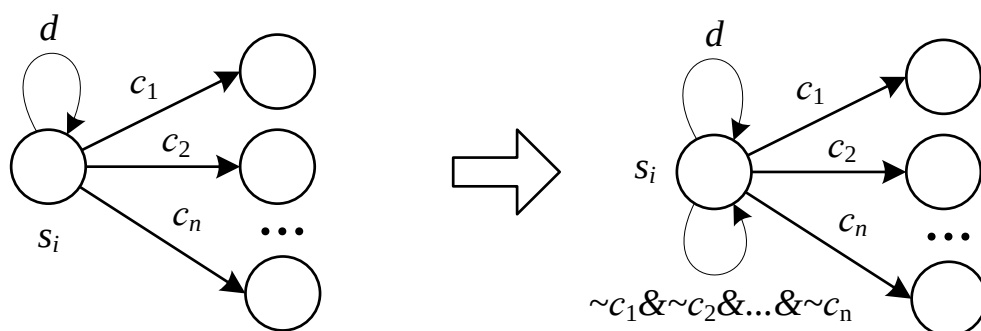


Рисунок 12 – Правила преобразования НДА, выполняемого в Семантике 2, в НДА, выполняемый в Семантике 1

В соответствии с Семантикой 1 сброс состояния НДА происходит, если в него не ведёт ни одной активной дуги, и устанавливается, если в состояние входит хоть одна активная дуга.

Если НДА находится в текущий момент в определённом состоянии, то данное состояние S_i будем называть *активным* (или маркированным). Под установкой состояния будем понимать его активацию, под сбросом – деактивацию.

Абстрактная модель НДА, которая была введена в работе [26], понятна и не вызывает вопросов. Все разрешённые переходы данной модели НДА совершаются параллельно и без конфликтов. Проблемы возникают при реализации фрагментов НДА, которые связаны с входо-выходными состояниями. С каждым состоянием НДА будет связан битовый элемент памяти, который хранит его статус. Для простоты будем считать, что состояние и его реализующий элемент памяти имеют одинаковые имена. Если $S_i = 1$, то НДА пребывает в состоянии S_i , если $S_i = 0$, то нет. Входо-выходное состоянием НДА – это состояние, в которое входит хотя бы одна дуга и одновременно выходит хотя бы одна дуга.

Для примера рассмотрим НДА1, который изображён в левой части рисунка 16. Состояние S_2 – это входо-выходное состояние. Начальное состояние представленного автомата – $\{S_1, S_2\}$. При подаче входного сигнала x_1 представленный автомат перейдет в состояние $\{S_2, S_3\}$. Данный переход можно обозначить следующим образом: $\{S_1, S_2\} \xrightarrow{x_1} \{S_2, S_3\}$. Графически данный переход показан на рисунке 13 в виде правила изменения маркировки автомата.

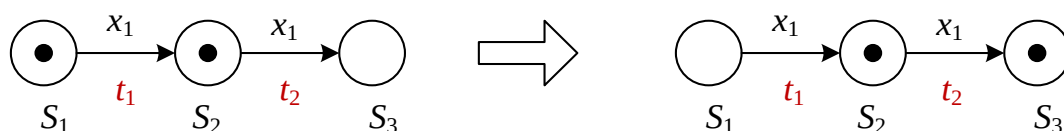


Рисунок 13– Функционирование НДА1 с входо-выходным состоянием

Немного изменённый (по начальному состоянию) автомат НДА2 и правило его работы (при действующем сигнале x_1) изображены на рисунке 14.

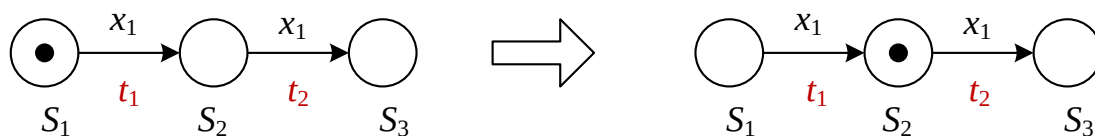


Рисунок 14 – Функционирование НДА2 с входо-выходным состоянием

Главная проблема при реализации НДА заключается в возможностях конфликтов в элементах памяти при выполнении над ними нескольких параллельных операций. Например, при реализации НДА, который представлен на рисунке 13, который функционирует в соответствии с Семантикой 2, происходит конфликт из-за состояния S_2 . С одной стороны, данное состояние устанавливается в “1” переходом t_1 , а с другой стороны – сбрасывается переходом t_2 . При использовании сериализации (последовательного выполнения) переходов t_1 и t_2 , то при последовательном выполнении (t_1, t_2) получается неправильный результат (рисунок 15).

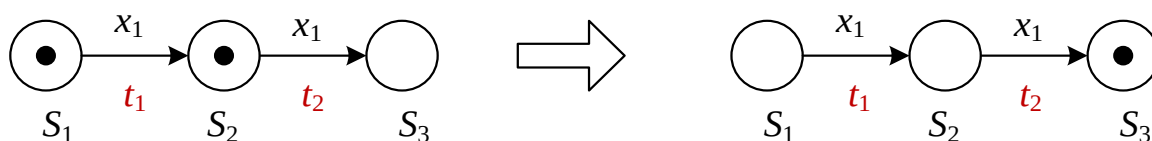


Рисунок 15 – Неправильное функционирование НДА1 при последовательном порядке выполнения переходов (t_1, t_2)

Семантике 1 также свойственны аналогичные проблемы выполнения. Результат выполнения автомата НДА2 на основе Семантики 1 при следующем порядке вычислений: вначале вычисляется состояние S_3 , а затем – состояние S_2 , приведён на рисунке 14 и является правильным. Однако, если принят обратный порядок вычисления состояний (S_2, S_3), приведённый на рисунке 16, то результат выполнения НДА2 будет неправильным. Необходимо отметить, что аналогичные проблемы существуют и при использовании Семантики 2.

Например, при выборе порядка срабатывания переходов (t_1, t_2) , приведённый на рисунке 16, результат окажется неправильным.

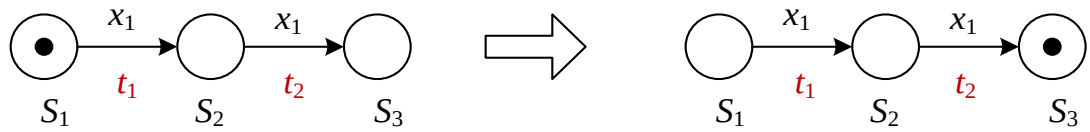


Рисунок 16 – Неправильное функционирование НДА2 при порядке вычисления состояний (S_2, S_3) или выполнения переходов (t_1, t_2)

Для решения проблемы одновременного выполнения операций над одним элементом памяти предлагается использовать двухтактное (двухфазное) выполнение НДА. Введём понятие "двухкомпонентного" состояния по аналогии с двухтактным триггером. На первой фазе выполнения НДА синхронно срабатывают разрешённые (обычные) переходы состояний НДА, при этом первые компоненты состояний устанавливаются на основе вторых компонентов состояний. Таким образом, первые компоненты состояний являются буферными, а вторые – действующими. Первая фаза основная. Суть второй фазы – это актуализация вторых (основных) компонентов состояний на основе первых (буферных) компонентов. Вторая фаза – вспомогательная.

На рисунке 17 показан пример расщепления состояния S на первую и вторую компоненты – S' и S'' . На представленном рисунке сплошными дугами обозначены переходы, которые выполняются на первой фазе. Переход, который выполняется на второй фазе, отмечен пунктирной линией между подсостояниями. Выражения c_i ($i=1, n$) определяет условия перехода из состояний S_i ($i=1, n$) в состояние S' .

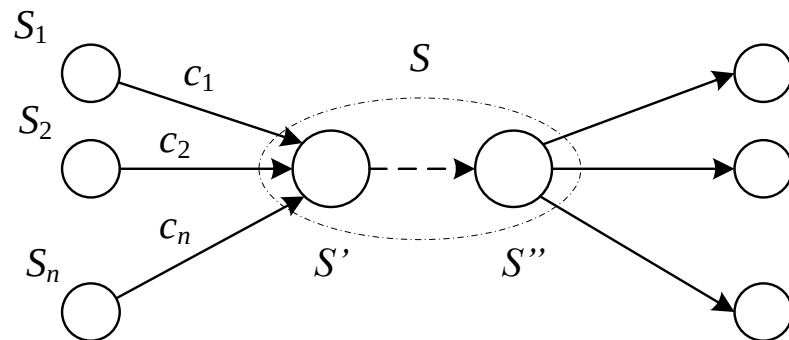


Рисунок 17 – Расщепление состояния S на два компонента

Для Семантики 1 более формально опишем первую и вторую фазу. Определим формулу для определения активности первых компонентов состояний на фазе 1:

$$S'(t+1) = \bigvee_{i=1}^n c_i(t) \& S_i(t) \quad (1)$$

В соответствии с представленной формулой состояние автомата S' будет активным, если в автомате имеется хотя бы одно активное состояние, из которого в состояние S ведёт выполнимый переход. В противном случае состояние будет неактивным.

Вторую фазу можно описать равенством:

$$S''(t+2) = S'(t+1) \quad (2)$$

В соответствии с представленной формулой активность второго компонента состояния приравнивается активности соответствующего первого компонента.

Для получения "реализационной" формы НДА предложено правило расщепления состояния, которое изображено на рисунке 18.

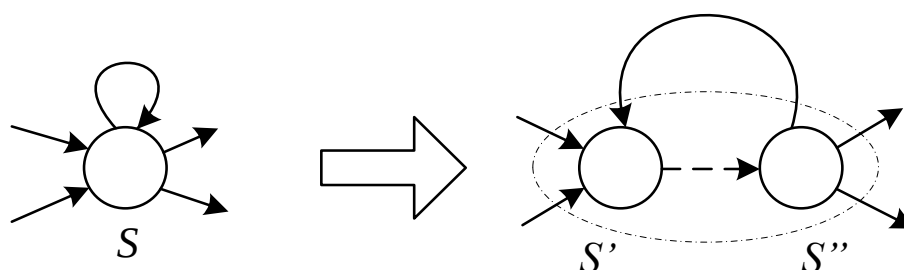


Рисунок 18 – Правило расщепления состояния

Рассмотрим общие подходы к реализации. Ограничения и допущения рассматриваемых подходов:

- 1) будем считать, что время обработки входного сигнала системой ФБ гораздо меньше промежутка времени между поступлениями входных сигналов;
- 2) входной сигнал x_i обрабатывается в ФБ в любом случае независимо от того, приведёт ли он к смене состояния или нет;
- 3) вторая фаза выполняется в каждом функциональном блоке независимо от того, приведёт ли она к смене состояния или нет.

Структурные шаблоны построения систем функциональных блоков, реализующих НДА, приведены на рисунках 19-22. Основными элементами шаблона являются ФБ, которые моделируют состояния НДА (которые называются также ФБ-состояниями), и ФБ, который представляет диспетчер, организующий общий процесс вычислений.

На рисунках 19-20 изображены два шаблона для организации обработки входного сигнала на первой фазе выполнения – «последовательная схема» (рисунок 19) и «параллельная схема» (рисунок 20). В последовательной схеме блоки, которые обрабатывают один и тот же сигнал, связаны в цепочку Дейзи. Входной сигнал x_i подаётся на первый блок данной цепочки. Выходной сигнал с последнего блока сигнализирует, о том, что обработка сигнала x_i во всех релевантных функциональных блоков закончена. Данный сигнал поступает в диспетчер *disp* и генерирует выходной сигнал *ph2*, что служит основанием для начала второй фазы выполнения. Необходимо отметить, что данная схема вполне работоспособна и без диспетчера, при этом сигнал старта второй фазы *ph2* может получаться как дизъюнкция выходных сигналов последних блоков цепочек. Данный способ можно реализовать или с использованием стандартного функционального блока *E_MERGE*, либо с использованием способа «аппаратная дизъюнкция». Также необходимо отметить, что один и тот же функциональный блок может быть задействован в разных цепочках, поскольку в нём может быть задействована обработка нескольких входных сигналов.

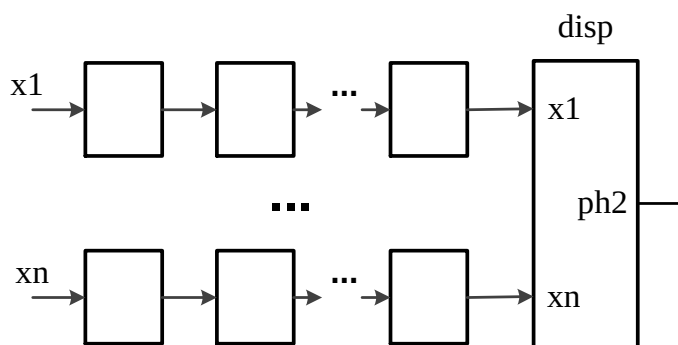


Рисунок 19 – Последовательная схема обработки входных сигналов (на первой фазе)

В параллельной схеме входной сигнал x_i поступает сразу на все релевантные функциональные блоки. Чтобы синхронизировать завершение выполнения группы ФБ, необходимо использовать механизм квитирования с подсчетом числа квитанций. По завершению обработки входного сигнала x_i каждый функциональный блок посылает соответствующий сигнал-квитанцию в диспетчер. Диспетчер должен получить все квитанции (их число определяется по числу релевантных ФБ по входному сигналу x_i), после чего диспетчер инициирует вторую фазу выполнения.

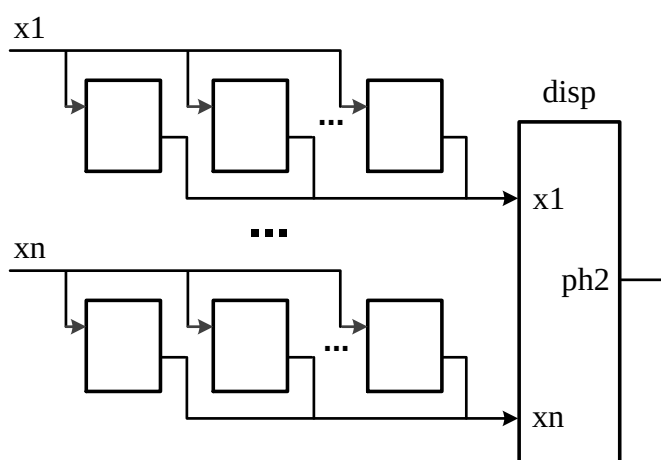


Рисунок 20 – Параллельная схема обработки входных сигналов (на первой фазе)

Преимущество последовательной схемы – это её простота и применимость ко всем моделям выполнения функциональных блоков вследствие своей детерминированности. Параллельная схема более сложная и недетерминированная и применима далеко не для всех моделей выполнения ФБ. Например, при синхронной модели выполнения возможно «затирание» сигналов на входе x_i блока *disp*, и это приведёт к неправильному подсчёту числа квитанций. К преимуществу данной схемы относится возможность параллельного выполнения функциональных блоков.

На рисунках 21-22 изображены два шаблона для организации второй фазы выполнения – «последовательная схема» и «параллельная схема». В последовательной схеме блоки, которые реализуют действия второй фазы, выполняются последовательно, друг за другом. Общий сигнал окончания второй фазы *ph2o* – это сигнал окончания второй фазы в последнем блоке цепочки.

Сигнал старта второй фазы *ph2* в параллельной схеме передаётся одновременно во все блоки. После окончания выполнения действий второй фазы каждый функциональный блок посылает диспетчеру соответствующую квитанцию. Диспетчер собирает квитанции из всех функциональных блоков, после чего формирует общий сигнал об окончании второй фазы *ph2o*. Этот сигнал может использоваться в системе верхнего уровня. Преимущества и недостатки данных схем аналогичны преимуществам и недостаткам соответствующих предыдущих решений, предложенных на рисунках 19-20.

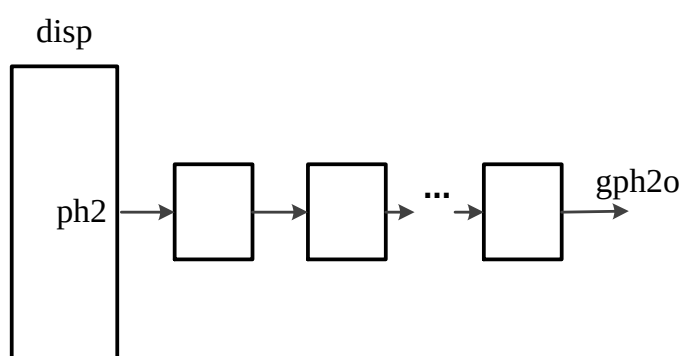


Рисунок 21 – Последовательная схема для организации второй фазы выполнения

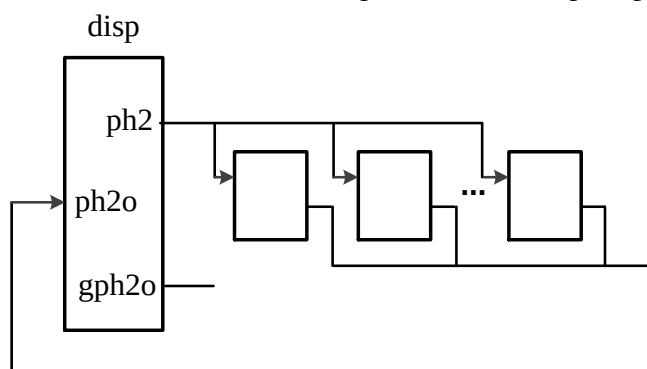


Рисунок 22 – Параллельная схема для организации второй фазы выполнения

На рисунках 23-24 представлены диаграммы последовательности *UML-FB* типовых сценариев функционирования систем функциональных блоков, которые реализуют НДА. На данных диаграммах приняты следующие обозначения. Пустыми прямоугольниками обозначены ФБ, которые моделируют состояния НДА. Прямоугольник с вписанной в него буквой *D* – это диспетчер. Прямоугольник с буквой *E* – внешняя среда (система верхнего уровня). Для простоты над стрелкой указываются только имена событийных входов ФБ-приёмников.

На рисунке 23 представлена диаграмма последовательности *UML-FB* функционирования системы ФБ при использовании только последовательных схем во всех двух фазах выполнения. В этом случае диспетчер вообще не используется. На рисунке 23 можно выделить две группы функциональных блоков. Первая группа связана в цепочку по входному сигналу x_i . Вторая включает остальные ФБ, которые не вошли в данную цепочку. Поскольку выше принято, что время обработки входного сигнала x_i системой ФБ гораздо меньше времени между поступлениями входных сигналов в систему, то сигнал $gph2o$ вообще можно не учитывать (и во внешнюю систему не подавать). Необходимо отметить, что на рисунке 23 показана одна из возможных цепочек ФБ по сигналу $ph2$, определённая исходя из простоты рисования диаграммы.

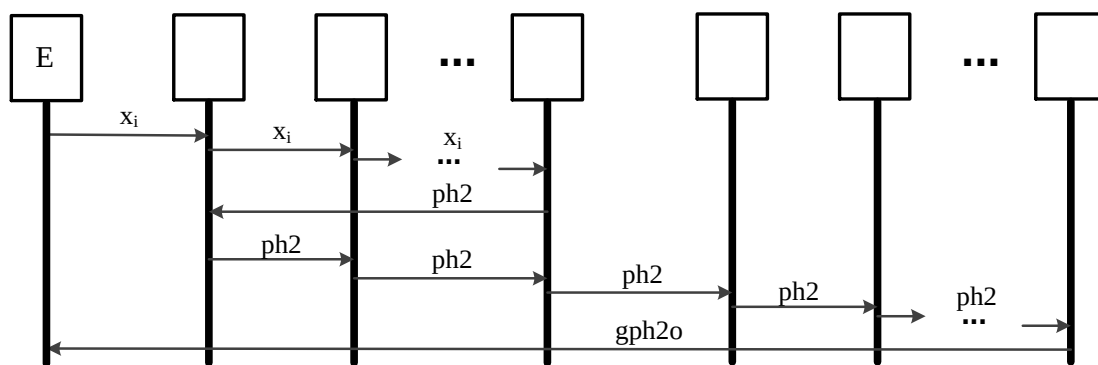


Рисунок 23– *UML-FB* диаграмма последовательности сценария выполнения системы ФБ, которая использует только последовательные схемы выполнения

На рисунке 24 представлена диаграмма последовательности *UML-FB* сценария выполнения системы функциональных блоков, которая использует только параллельные схемы выполнения.

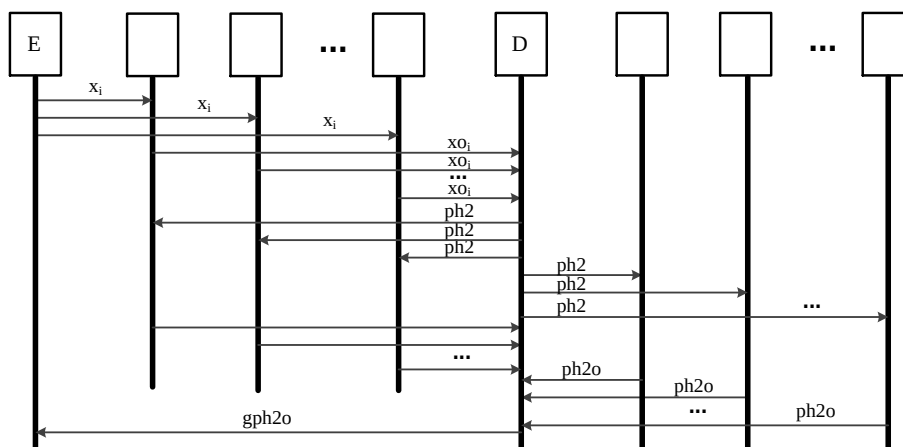


Рисунок 24 – *UML-FB* диаграмма последовательности сценария выполнения системы ФБ, которая использует только параллельные схемы выполнения

Как можно заметить на рисунке, в данном случае диспетчер собирает квитанции от ФБ-состояний как на первой фазе, так и на второй, для того чтобы выработать общий сигнал окончания работы всех функциональных блоков.

2.2 Методика реализации магазинных автоматов

Необходимо отметить, что конечные автоматы являются достаточно простыми моделями, и они не могут описать сложные процессы, например, представляемые контекстно-свободными (КС) языками. Для распознавания КС-языков необходимо использовать магазинные автоматы (МА). Формальное определение МА можно найти, например, в [14]. В отличие от конечных автоматов, в МА используется магазинная память (стек, магазин).

В данной работе для простоты будут использоваться только детерминированные МА. Рассмотрим подход к реализации магазинных автоматов на основе функциональных блоков международного стандарта IEC 61499. Рассмотрим данную методику:

- 1) каждый переход магазинного автомата – функциональный блок;
- 2) каждый ФБ, представляющий переход из начального состояния автомата имеет маркер;
- 3) каждому ФБ-переходу на вход подается сигнал инициализации или установки (*Setup*), сигнал (*Res_check*) и переменная (*R_check*), служащие для получения верхнего магазинного символа.
- 4) из каждого ФБ-перехода исходят следующие сигналы: сигнал срабатывания ФБ-перехода (*Out*), магазинные символ принят ФБ (*check*), магазинные символ не принят ФБ (*no_check*). Сигнал *Out* идёт на выход, а сигналы *check* и *no_check* в ФБ, который управляет потоками данных;
- 5) при срабатывании ФБ-переход сбрасывает все ФБ-переходы, исходящие из одного и того же состояния (сигналы *Reset* и *OutReset*);

- 6) при срабатывании ФБ-переход устанавливает все ФБ-переходы, исходящие из состояния, в которое переходит срабатываемый переход (сигналы *Set* и *OutSet*;
- 7) ФБ-стек имеет сигналы и переменные, необходимые для следующих операций: втолкнуть магазинный символ или несколько магазинных символов, вытолкнуть магазинный символ, показать значение верхнего магазинного символа.

При данном подходе ФБ, представляющий переход, одного и того же типа и отличается только некоторыми параметрами при инициализации. Возможна также реализация состояния в виде ФБ, но каждый ФБ будет отличаться числом входных/выходными сигналами и переменными, и сложность реализации каждого отдельного блока-состояния будет довольно большой.

Для моделирования функционирования МА, также как и для конечных автоматов используется механизм маркеров. Маркер является динамическим объектом, передаваемым из одного ФБ-перехода в другой ФБ-переход. Если ФБ имеет маркер, то он может принимать входные символы. Упрощённая схема реализации МА представлена на рисунке 25. Первый блок типа *StateMA* реализует один из переходов автомата. ФБ *Switch* управляет потоками данных от ФБ-переходов в функциональный блок стека. ФБ *Stack* отвечает за все операции связанные с магазином (стеком). Данная реализация МА функционирует следующим образом: при приходе некоторого входного сигнала, срабатывает один из ФБ-переходов или несколько ФБ, если ФБ принимают одинаковый входной сигнал и в них установлены маркеры. Далее сигналы через функциональный блок *Switch* направляются в магазин. Если верхний магазинный символ совпал с магазинным символом перехода, то сработавший ФБ-переход передаёт свой выходной сигнал на один из выходов $S_0 \dots S_z$.

Рассмотрим реализацию ФБ *StateMA* подробнее. Интерфейс данного базисного функционального блока состоит из следующих групп параметров:

Входные сигналы:

Set – сигнал, по которому устанавливается маркер в данном ФБ;

Reset – сигнал, по которому удаляется маркер из данного ФБ;

WP – сигнал поступления входного символа;

Setup – сигнал, установки (сброса) начальных параметров ФБ;

Res_Check – сигнал проверки верхнего символа магазина с магазинным символом ФБ.

Выходные сигналы:

OutSet – сигнал, который устанавливает маркеры в других ФБ;

OutReset – сигнал, который сбрасывает маркеры в других ФБ;

Out – сигнал, означающий, что ФБ сработал;

Check – сигнал проверки верхнего символа магазина;

No_Check – сигнал, который выдаётся, если верхний символ магазина не совпал с магазинным символом ФБ.

Входные переменные:

Input – входной символ;

C – входной символ, который может принять ФБ;

T – флаг наличия маркера в ФБ;

M – максимальное количество объектов заданного типа, которые могут быть обработаны данным ФБ перед передачей маркера на другой (другие) ФБ.

R_CHECK – результат проверки верхнего символа магазина.

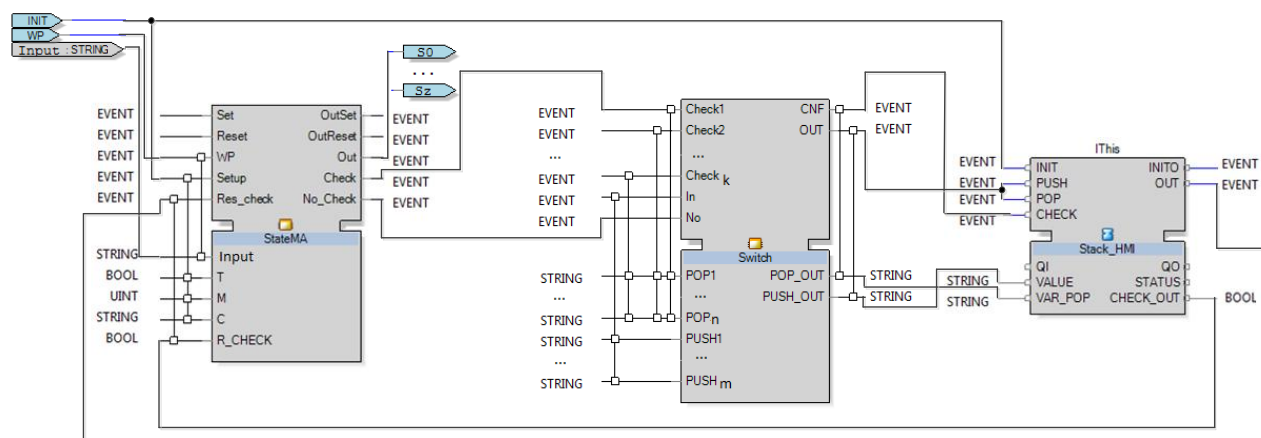


Рисунок 25 – сеть ФБ, реализующая МА

Диаграмма управления выполнением (диаграмма ЕСС) базисного ФБ *StateMA* представлена на рисунке 26 и состоит из шести состояний:

- 1) состояние *Reset*, выполняющее сброс маркера ФБ;
- 2) состояние *Setup*, устанавливающее начальные значения внутренних переменных;
- 3) состояние *Set*, устанавливающее маркер ФБ;
- 4) состояние *WP*, в которое автомат переходит при выполнении следующего условия: $WP \ \& \ (T1 = TRUE) \ \& \ (Color = C1)$. Переход в данное состояние произойдёт, если в реализацию МА поступит входной символ, воспринимаемый этим ФБ, и данный ФБ имеет маркер, при переходе в данное состояние выдается сигнал *Check*;
- 5) состояние *WP1*, в котором происходит увеличение счётчика количества проходов по петле. В данное состояние может попасть ФБ, $R_CHECK = TRUE$ и $M > 1$. Кроме того, в этом состоянии выдаются выходные сигналы *Out* и *OutReset*;
- 6) состояние *WP2*, в котором происходит удаление маркера и выдача всех требуемых выходных сигналов. В данное состояние попадает ФБ, который имеет $R_CHECK = TRUE$ и $M=0$, или ФБ, уже прошедший M раз по петле;
- 7) состояние *NOT_CHECK*, в которое автомат переходит, если не принят стек ($R_CHECK = FALSE$). Кроме того, в данном состоянии выдаются выходной сигнал *No_Check*.

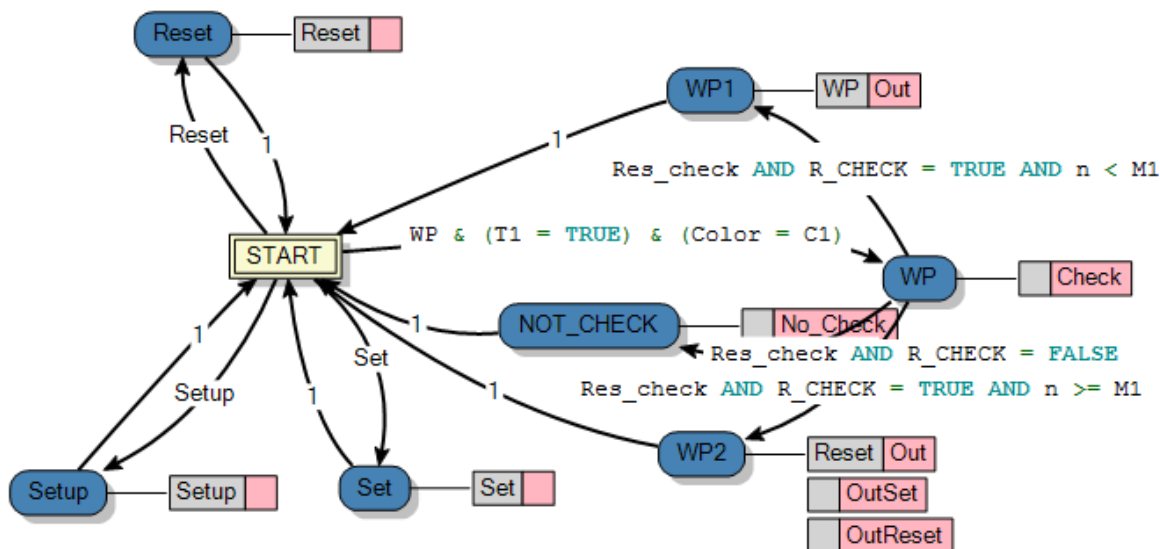


Рисунок 26 – диаграмма ECC ФБ StateMA

Интерфейс ФБ *Stack* состоит из следующих групп параметров:

Входные сигналы:

INIT – инициализация стека;

PUSH – положить значение в стек;

POP – извлечь значение из стека;

CHECK – проверка значения входной переменной *VAR_POP* с верхним значением стека, результат данной проверки выдаётся в выходной переменной *CHECK_OUT*.

Выходные сигналы:

INTO – сигнал, который означает, что ФБ инициализирован;

OUT – сигнал, выдающийся в результате проверки верхнего значения стека;

Входные переменные:

VAR_POP – переменная, с которой происходит сравнение верхнего символа стека;

VALUE – переменная, значение которой добавляется в стек;

Выходные переменные:

OUT_VALUE – переменная, которая выдаёт результат проверки входной переменной *VAR_POP* с верхним значением стека (*TRUE* – значения равны, *FALSE* – не равны).

ФБ *Switch* нужен для перенаправления значений переменных, содержащих магазинные символы, в стек, ввиду того, что в среде *NxtStudio* нельзя подсоединить несколько выходных переменных ФБ к одной входной. ФБ *Switch* в зависимости от определённого сигнала проверки (*Check1...Check_k*) сопоставляет соответствующий магазинный символ с вершиной стека и, если символы совпали, то записывает необходимую цепочку в магазин.

Интерфейс данного ФБ состоит из следующих групп параметров:

Входные сигналы:

Check1..Check13 – проверка стека от определённого ФБ-перехода;

In – втолкнуть значение в стек;

No – значение в стек не вталкивать;

Выходные сигналы:

CNF – сигнал, который означает проверку верхнего значения в стеке;

OUT – сигнал, по которому извлекается верхнее значение стека и переменная *PUSH_OUT*, содержащая необходимую цепочку, которая добавляется в стек;

Входные переменные:

POP1..POP3 – переменные (алфавит стека) с которыми сравнивается верхний магазинный символ;

PUSH0..PUSH6 – переменные – цепочки, добавляющиеся в стек;

Выходные переменные:

POP_OUT – переменная, значение которой необходимо сравнить с верхним символом стека.

PUSH_OUT – переменная, значение которой вталкивается в стек.

2.3 Методика реализации сетей Петри

Конечные и магазинные автоматы неплохо себя показывают, но естественным является желание использовать для выборки объектов более мощные в вычислительном плане модели, например, сети Петри и производные от них модели.

Реализация СП на функциональных блоках стандарта IEC 61499 подразумевает следующие действия:

- 1) каждая позиция представляется ФБ-позицией;
- 2) каждый переход представляется ФБ-переходом;
- 3) каждой ФБ-позиции назначается начальное число меток и текущее число меток; текущее число меток подсоединяется в ФБ-переходам, для которых данная позиция является входной;
- 4) в каждую ФБ-позицию входят сигналы для увеличения и уменьшения меток, количество данных сигналов соответствует входным и выходным переходам для данной ФБ-позиции;
- 5) в каждый ФБ-переход входят переменные, указывающие количество меток входных позиций для данного перехода.

Числа, на которые изменяются количество меток в позиции, могут быть определены внутри самой ФБ-позиции или приходить из входных или выходных переходов. В данной работе был выбран способ направления данных из переходов ввиду своей наглядности.

Вначале рассмотрим ФБ-позицию (Place). Пример позиции для двух входов и двух выходов представлен на рисунке 27.

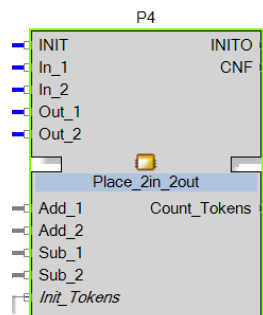


Рисунок 27 – Реализация ФБ-позиции

Интерфейс данного базисного ФБ состоит из следующих групп параметров:

1. Входные сигналы:

- INIT – сигнал инициализации ФБ;
- In_1 – срабатывание первого входного перехода (для данного перехода рассматриваемая позиция является выходной). В момент данного сигнала счетчик числа меток увеличивается на значение, переданное в Add_1;
- In_2 – срабатывание второго входного перехода (для данного перехода рассматриваемая позиция является выходной). В момент данного сигнала счетчик числа меток увеличивается на значение, переданное в Add_2;
- Out_1 – срабатывание первого выходного перехода (для данного перехода рассматриваемая позиция является входной). В момент данного сигнала счетчик числа меток уменьшается на значение, переданное в Sub_1;
- Out_2 – срабатывание второго выходного перехода (для данного перехода рассматриваемая позиция является входной). В момент данного сигнала счетчик числа меток уменьшается на значение, переданное в Sub_2.

2. Входные переменные:

- Add_1 – принимает вес первой входящей в позицию дуги;
- Add_2 – принимает вес второй входящей в позицию дуги;
- Sub_1 – принимает вес первой исходящей из позиции дуги;
- Sub_2 – принимает вес второй исходящей из позиции дуги;
- Init_Tokens – начальное количество меток в данной позиции.

Первые четыре переменные нельзя объединить в одну из-за особенностей pxtStudio. Можно использовать дополнительный ФБ для такого объединения. Если каждая дуга СП имеет единичный вес, то можно отказаться от использования первых четырех входных переменных.

3. Выходные сигналы:

- INITO – ФБ проинициализирован;
- CNF – число меток в позиции изменилось (используется только для изменения значения переменной Count_Tokens).

4. Выходные переменные:

- Count_Tokens – количество меток в позиции.

Рассмотрим ФБ-переход (Transition). Пример перехода для одного входа и одного выхода можно увидеть на рисунке 28.

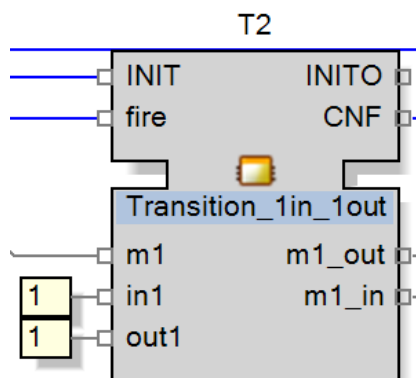


Рисунок 28 – Реализация ФБ-перехода

Интерфейс данного базисного ФБ состоит из следующих групп параметров:

1. Входные сигналы:

- INIT – сигнал инициализации ФБ;
- fire – сигнал срабатывания перехода. ФБ сравнивает значение переменной m1 (число меток во входной позиции) со значением переменной in1 (вес входной дуги). Если Вес дуги больше или равен количеству меток, то переход срабатывает и выдается выходной сигнал CNF.

2. Входные переменные:

- m1 – число меток во входной позиции;
- in1 – вес входной дуги;
- out1 – вес выходной дуги.

Если каждая дуга СП имеет единичный вес, то можно отказаться от использования последних двух переменных.

3. Выходные сигналы:

- INITO – ФБ проинициализирован;
- CNF – срабатывание перехода, данный сигнал направляется на сигналы срабатывания переходов позиций.

4. Выходные переменные:

- $m1_out$ – вес выходной дуги, подсоединяется к переменной Add_x позиции;
- $m1_in$ – вес входной дуги, подсоединяется к переменной Sub_x позиции.

Также можно отказаться от использования данных переменных, просто продублировав веса дуг в переменных Add_x и Sub_x позиций.

Также для выборки объектов можно использовать и расширения сетей Петри А-сети. Основная идея методики заключается в реализации элементов СП – позиций и переходов в виде отдельных ФБ, так же, как и при реализации обычных СП. ФБ-позиция хранит маркировку позиции (значение переменной), а также выполняет над ней элементарные операции сложения / вычитания и сброса в ноль. ФБ-переход определяет разрешённость перехода СП и управляет сменой маркировки соседних позиций при его срабатывании. Выполнением всей сетевой модели управляет ФБ-диспетчер.

Правило преобразования перехода СП в ФБ показано на рисунке 29.

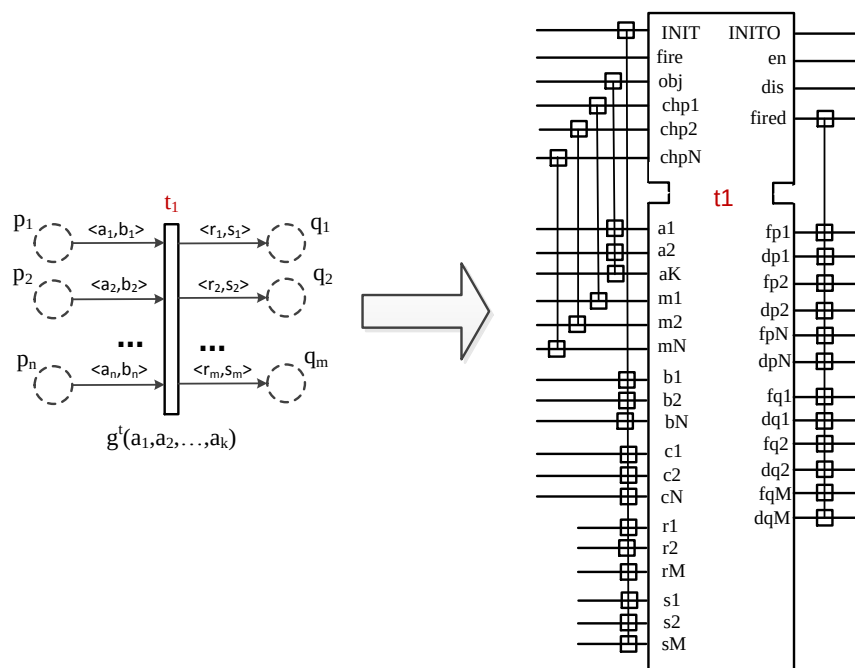


Рисунок 29 – Преобразование перехода А-сети в ФБ

Описание интерфейса этого ФБ следующее.

Событийные входы:

INIT – инициализация ФБ;

fire – активизация (срабатывание) разрешённого перехода;

obj – приход нового объекта;

chp_i – изменение маркировки позиции p_i .

Событийные выходы:

INITO – инициализация ФБ выполнена;

en – переход стал разрешённым;

dis – переход перестал быть разрешённым;

fired – переход сработал.

Информационные входы:

a_k – k -й параметр входного объекта;

m_i – (новая) маркировка позиции p_i ;

b_i – первый компонент веса входной дуги из позиции p_i ;

c_i – второй компонент веса входной дуги из позиции p_i ;

r_i – первый компонент веса выходной дуги в позицию q_i ;

s_i – второй компонент веса выходной дуги в позицию q_i .

Необходимо отметить, что входы типов b, c, r, s являются необязательными, поскольку они могут быть жёстко заложены внутри ФБ (в алгоритмах обработки). Однако вывод их «наружу» позволяет строить самомодифицируемые системы (путем модификации данных параметров).

Информационные выходы:

fp_i – модификатор действия над маркировкой позиции p_i ;

fq_i – модификатор действия над маркировкой позиции q_i ;

dp_i – число меток, удаляемых из позиции p_i ;

dq_i – число меток, добавляемых или назначаемых в позиции q_i .

Значения модификатора действия над маркировкой:

0 – указанное число меток назначается в выходную позицию, причём она предварительно обнуляется;

1 – указанное число меток добавляется в выходную позицию в дополнение к существующим;

2 – указанное число меток удаляется из выходной позиции.

ФБ, который моделирует позицию, представлен на рисунке 30.

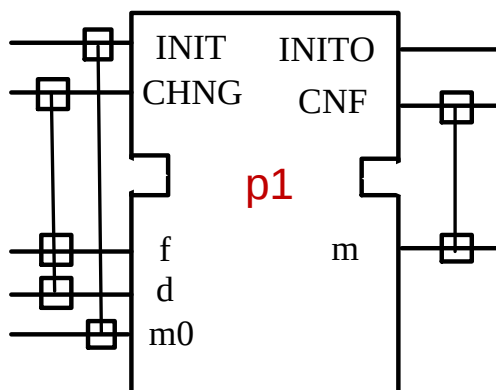


Рисунок 30 – ФБ-позиция А-сети б – ФБ-диспетчер

Событийные входы:

INIT – инициализация ФБ;

CHNG – изменить маркировку позиции.

Событийные выходы:

INITO – инициализация ФБ выполнена;

CNF – маркировка позиции изменена.

Информационные входы:

m_0 – начальная маркировка позиции;

f – модификатор действия над маркировкой позиции (добавить, установить, удалить);

d – аргумент модификатора (сколько добавить, установить, удалить).

Информационный выход:

m – значение маркировки позиции.

ФБ-диспетчер представлен на рисунке 31.

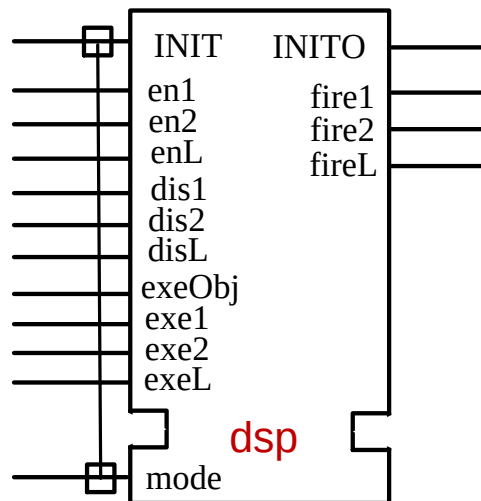


Рисунок 31 – ФБ-диспетчер А-сети

Событийные входы:

INIT – инициализация ФБ;

en_i – сетевой переход t_i стал разрешённым;

dis_i – сетевой переход t_i перестал быть разрешённым;

exeObj – определение активного перехода после прихода сигнала о новом объекте;

exe_i – определение активного перехода после срабатывания перехода t_i .

Событийные выходы:

INITO – инициализация ФБ выполнена;

fire_i – запуск (активация) перехода t_i .

Информационный вход:

mode – стратегия запуска переходов в сетевой модели.

В диспетчере хранится текущий список разрешённых переходов. При поступлении сигнала с входа *en_i* данный список пополняется переходом t_i . При поступлении сигнала с входа *dis_i* из данного списка удаляется переход t_i . При приходе сигнала на вход типа *exe* диспетчер выбирает наиболее приоритетный разрешённый переход для срабатывания (активации) и выдаёт на него сигнал запуска.

Можно сказать, что система на основе селектирующих СП является реактивной и обрабатывает события прихода очередного объекта. При этом цепочка событий будет следующей. Во время прихода нового объекта заново

вычисляются сторожевые условия в ФБ-переходах и статус перехода в целом, в диспетчер посылается сигнал на планирование запуска нового перехода. Для упрощения реализации имеем в виду, что интервалы времени между поступлениями внешних сигналов о приходе нового объекта значительно больше, чем время выполнения функциональных блоков. Отметим, что срабатывание какого-либо перехода может вызвать срабатывание цепочки других переходов, возможно не нагруженных сторожевыми условиями.

Срабатывание некоторого перехода, как правило, приводит к изменению маркировки соседних с ним позиций, что, в свою очередь, может повлиять на статус (разрешён / не разрешён) соседних с ними переходов. Только после завершения данного «переходного процесса» диспетчер может выбрать новый переход для активации. Пример схемы, которая моделирует данную ситуацию для частного случая, приведён на рисунке 32.

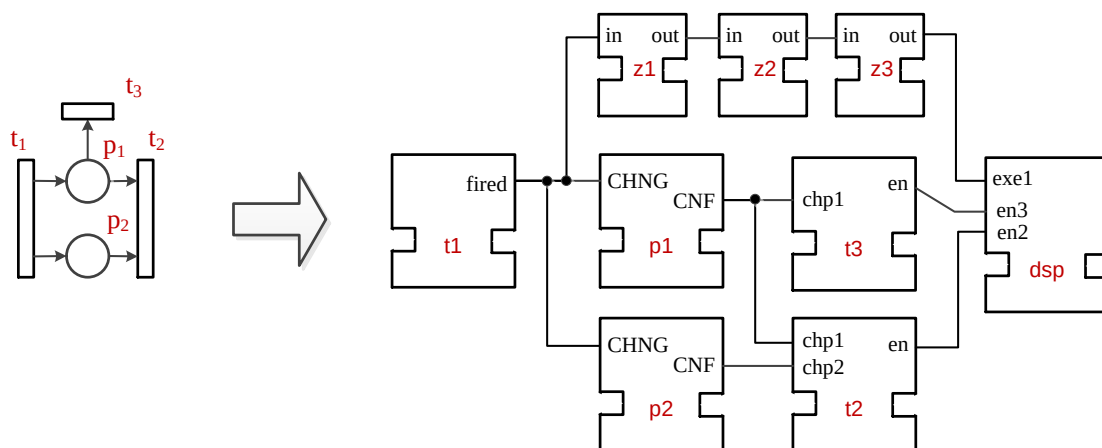


Рисунок 32 – Схема, моделирующая изменения в результате срабатывания перехода t_1

Следует заметить, что на рисунке 32 представлены только событийные связи. Кроме того, не рассматриваются цепи деактивации перехода. Проблема срабатывания перехода заключается в том, чтобы сигнал на планирование срабатывания нового перехода поступал в диспетчер после того, как диспетчером будут получены все сигналы об изменении статусов переходов. Данная проблема синхронизации может быть решена с использованием механизма квитирования или с использованием реальных временных задержек, что, однако, значительно усложнит схему и уменьшит её быстродействие.

Кроме того, можно воспользоваться особенностями используемой модели выполнения ФБ. В нашем случае используется популярная модель выполнения *Buffered sequential execution model (BSEM)*, принятая в системе *NxtStudio*. Данная модель основана на использовании *FIFO*-очереди. Для задержки в планировании запуска некоторого ФБ будем использовать «пустые» функциональные блоки $z1, z2, z3, \dots$, функциональность которых заключается в передаче сигнала с входа на выход.

2.4 Выводы

1. Разработан подход к реализации конечных автоматов на основе ФБ стандарта IEC 61499 с использованием механизма передачи маркеров. Представлены правила графовых трансформаций, определяющие процесс порождения структур систем ФБ IEC 61499 из конечноавтоматной спецификации селектируемых последовательностей деталей.
2. Разработан подход к двухфазной реализации НДА на основе ФБ стандарта IEC 61499, когда в виде блоков представляются состояния автомата. Рассмотрены последовательная и параллельная схемы выполнения первой и второй фаз, а также представлены UML диаграммы последовательности выполнения для последовательной и параллельной схемы.
3. Разработана методика реализации ДМА на основе ФБ стандарта IEC 61499, особенностью методики является использование представления ДМА в виде диаграммы переходов состояний и механизма передачи маркеров, что позволяет использовать подходы, используемые при функционально-блочной реализации конечных автоматов.
4. Предложена методика реализации селектирующих сетей Петри с приоритетами на основе ФБ стандарта IEC 61499. Особенностью методики является использование в функционально-блочной реализации блока-диспетчера, что позволяет легко изменять логику функционирования сети в процессе функционирования.

3 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ФУНКЦИОНАЛЬНО-БЛОЧНЫХ РЕАЛИЗАЦИЙ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ

3.1 Система сортировки деталей на основе селектирующих конечных автоматов

Для большей наглядности рассмотрим пример функционально-блочной реализации ДА [27]. Селектирующий автомат, распознающей последовательности объектов, можно увидеть на рисунке 33. О (orange), G (grey) и В (blue) – это цвета объектов. Конечный переход обозначен пунктирной линией.

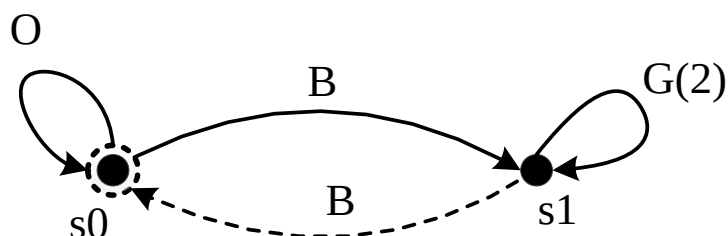


Рисунок 33 – Пример селектирующего ДА

Приведенный на рисунке 33 автомат выбирает, например, следующие последовательности объектов: ВВ, ВGB, ООВGB, ОООВВВВGGВ. Если на вход ДА будут поданы цепочки ВG, ООGGGG, ОООВGG, то он не сможет выбрать из них ни одну из специфицированных последовательностей.

На рисунке 34 представлена структура функционально-блочной реализации автомата, приведённого на рисунке 33. Предполагается, что информация в данную систему ФБ поступает через сервисный интерфейсный ФБ (СИФБ). Выходным сигналом данной системы является сигнал *EOS* «Конец выбранной последовательности».

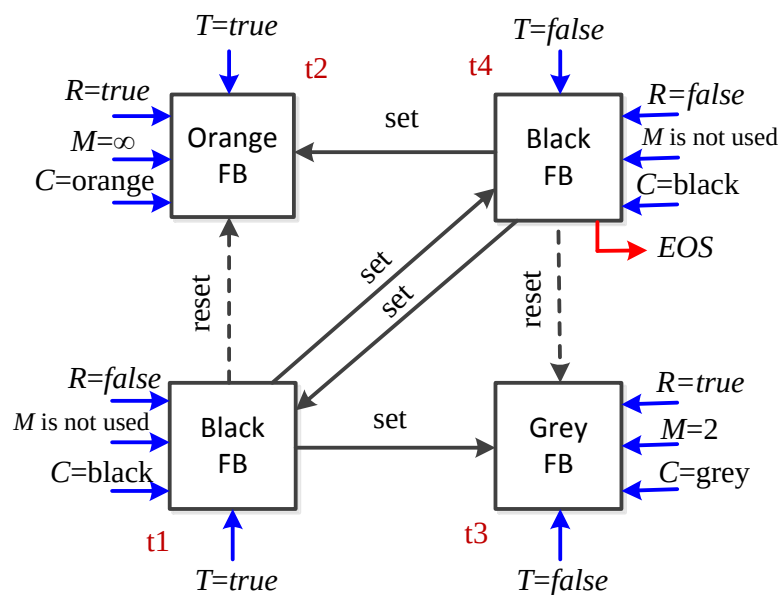


Рисунок 34 – Структура функционально-блочной реализации автомата из рисунка 33

Для представления структуры функционально-блочной реализации ДА используются следующие элементы:

дуга *set* для передачи маркера из исходного ФБ в целевой ФБ;

дуга *reset* для удаления маркера из целевого ФБ,

входные висячие дуги для установки начальных значений следующих параметров:

C – параметр (в данном случае цвет объекта), которую может принять ФБ. Условно это можно назвать «цветом ФБ». Данный параметр не может быть изменён во время функционирования системы и устанавливается изначально;

T – флаг наличия маркера в ФБ;

R – флаг петли (итерации). Если $R=true$, то ФБ осуществляет сбор последовательности деталей заданного цвета. Если $R=false$, то ФБ является «транзитным». После обработки одной детали он передает маркер другим ФБ;

M – максимальное количество объектов заданного цвета, которые могут быть обработаны данным ФБ перед передачей маркера на другой (другие) ФБ. Данный параметр действителен только для ФБ с $R=true$.

Саму реализацию ДА на основе рисунка 34 можно увидеть на рисунке 35.

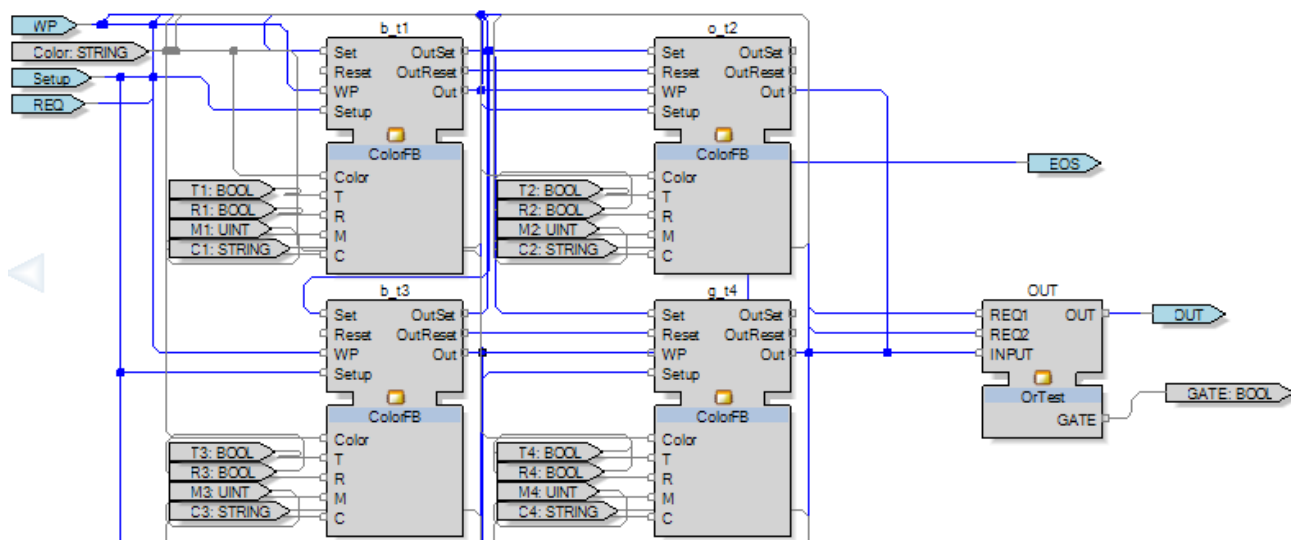


Рисунок 35 – Сеть ФБ стандарта IEC 61499, которая реализует автомат из рисунка 33.

Реализация, изображенная на рисунке 35, подставляется в один из автоматов распознавателей в реализации системы сортировки (рисунок 36).

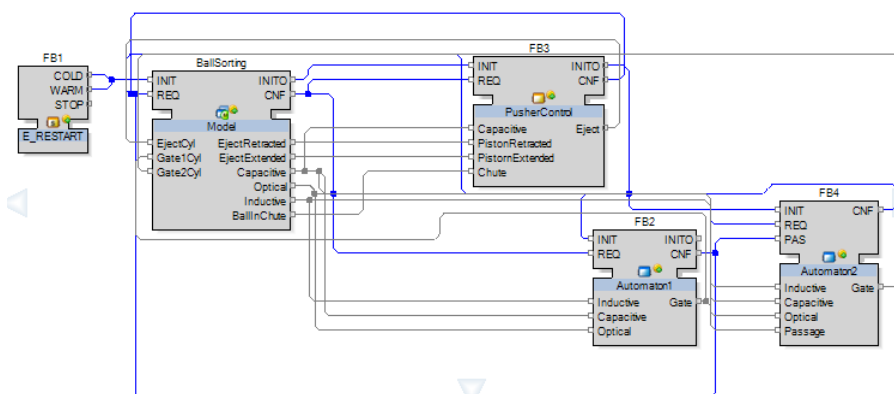


Рисунок 36 – Реализация имитационной модели системы сортировки шаров на основе ФБ

Реализация имитационной модели системы сортировки шаров (рисунок 36) состоит из четырёх блоков: *BallSorting* – ФБ, представляющий визуальную модель, которая включает генератор шаров; *FB3* – контроллер заслонок (управляет их возвратно-поступательным движением); *FB2* – первый автомат-распознаватель; *FB4* – второй автомат-распознаватель. На месте *FB2* и *FB4* располагается реализация ДА, подобная рисунку 36. Необходимо отметить, что ФБ *FB2*, *FB3* и *FB4* относятся к системе управления.

Сама система сортировки шаров состоит из выталкивателя; трёх датчиков для распознавания типа шаров (ёмкостной, индуктивный и оптический); двух датчиков определения положения выталкивателя; ёмкостного датчика, который определяет, был ли вытолкнут шар в шахту; двух заслонок для забора шаров из шахты в определенный накопитель. Визуальный интерфейс системы сортировки шаров изображён на рисунке 37.

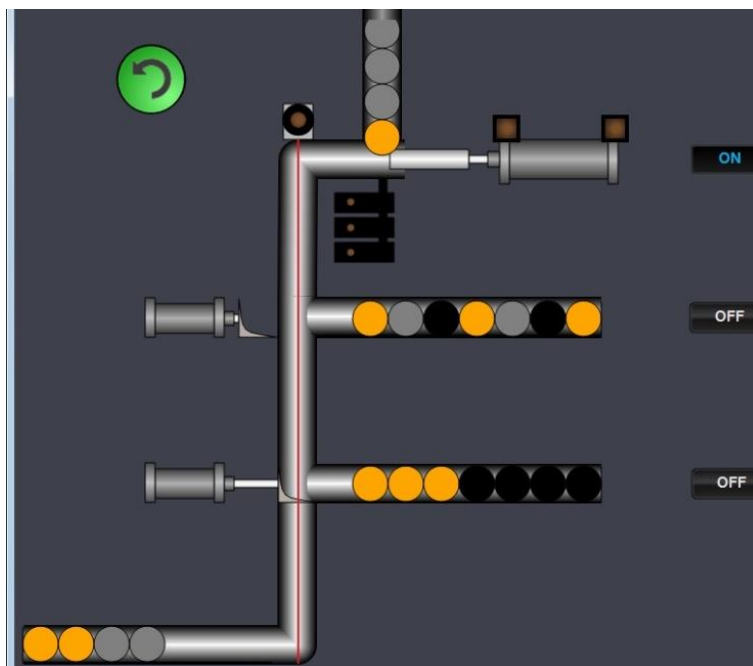


Рисунок 37 – Система сортировки шаров на основе выборки специфицированных последовательностей объектов (интерфейс пользователя в nxtStudio)

Система сортировки работает следующим образом: в первый накопитель попадают шары в последовательности <оранжевый, черный, серый>, которая задаётся определённым автоматом (в данной работе автомат не приведён); во второй накопитель попадают шары, отобранные автоматом, представленным на рисунке 33. Шары, которые не попали в первые два накопителя, попадают в третий накопитель (для отбракованных объектов).

Имитационная модель системы сортировки, которая включает систему управления на основе ФБ и визуальную модель оборудования, работает следующим образом: сначала генерируется шар определённого цвета, который распознаётся с помощью датчиков и попадает на анализ первому автомат-распознавателю. Если шар подходит, то сигнал «выдвинуть первую заслонку»

передаётся в модель сортировки, и, таким образом, шар попадает в первый накопитель. В противном случае сигнал передаётся на второй автомат-распознаватель и проверяется им. Исходя из сказанного, первый накопитель является более приоритетным, чем второй. Если второму автомату шар «подходит», то в систему сортировки подаётся сигнал «выдвинуть вторую заслонку». Если шар не подходит и второму накопителю, то он попадает в третий накопитель. Если одна из двух заслонок выдвинута, но шар не подходит, то она задвигается и пропускает шар, летящий вниз по шахте.

3.2 Транслятор Prolog-описаний селектирующих автоматов в имитационную модель системы сортировки деталей

Для автоматизации преобразования селектирующих автоматов в модель системы сортировки последовательностей деталей (ССПД) был разработан соответствующий транслятор [28-29]. Для описания селектирующих автоматов используется язык *Prolog* [30]. Ниже приведен пример описания автомата из рисунка 33 в виде базы фактов на языке *Prolog*:

```
state(s0) .
state(s1) .
transition(t1,s0,s1,black,_,ordinary) .
transition(t2,s0,s0,orange,999,ordinary) .
transition(t3,s1,s1,gray,2,ordinary) .
transition(t4,s1,s0,black,_,final) .
initial_state(s0) .
```

В приведённом описании используются следующие предикаты:

state – унарный предикат для представления состояний автомата;

transition – 6-арный предикат для представления переходов автомата, который имеет следующие аргументы: первый аргумент – уникальный идентификатор перехода; второй аргумент – идентификатор состояния-источника; третий аргумент – идентификатор состояния-приёмника; четвёртый

аргумент – цвет объекта; пятый аргумент – максимальное число допустимых деталей (для петли); шестой аргумент – тип перехода (обычный или конечный);

Имитационная модель ССПД реализована с помощью функциональных блоков в системе *NxtStudio*. Функциональные блоки (ФБ) в системе *NxtStudio* хранятся в виде *XML*-представления [31].

На рисунке 38 представлена структура системы моделирования, включающая указанный транслятор и инструментальную среду разработки *NxtStudio*.



Рисунок 38 – Структура системы моделирования

Ниже приведено описание структуры системы моделирования, включающей указанный транслятор и инструментальную среду разработки:

Вначале пользователь описывает автомат на языке *Prolog*. *Prolog*-описание загружается в транслятор, где проходит через три преобразования (описаны ниже в алгоритме работы транслятора). Транслятор уведомляет о ходе выполнения трансляции в поле вывода *Output*. После успешного завершения

трансляции, в зависимости от режима работы, генерируется либо полностью проект ССПД, либо файлы *XML*-представления ФБ, которые автоматически или вручную вставляются в проект системы сортировки. После компиляции проект ССПД можно протестировать в системе *NxtStudio*.

UML диаграмма классов транслятора приведена в приложении А.

Алгоритм работы транслятора приведен ниже:

Вначале выполняется метод *main* класса *Program*, где создаётся окно пользовательского интерфейса (*Form*). После того, когда пути до файлов заданы и пользователь выбрал опцию «Начать трансляцию», тогда из форма пользовательского интерфейса вызываются три основные функции класса *Global*:

1) функция загрузки в программу кода на языке *Prolog* и составление списков состояний (класс *State*) и переходов (класс *Transition*) автомата. Во время загрузки происходит проверка корректности введённого *Prolog*-описания. Если в описании на языке *Prolog* содержится ошибка, то транслятор выдаст соответствующее уведомление (класс *ErrotT*). Для загрузки в программу *Prolog*-описания используется специальный парсер (класс *Parser*).

2) функция преобразования описания автоматов в виде списков состояний (класс *State*) и переходов (класс *Transition*) в описание ФБ в виде списков функциональных блоков (класс *FB*) и соединений (класс *Connection*) между ними. Это основная функция, где и происходит трансляция.

3) запись представления ФБ в виде списков в файлы *XML*. Для записи в файл *XML* был использован парсер, встроенный в язык *C#*. Также на этом этапе происходит создание нового проекта *NxtStudio*, если выбран соответствующий режим.

UML диаграмма классов представлена на рисунке 39.

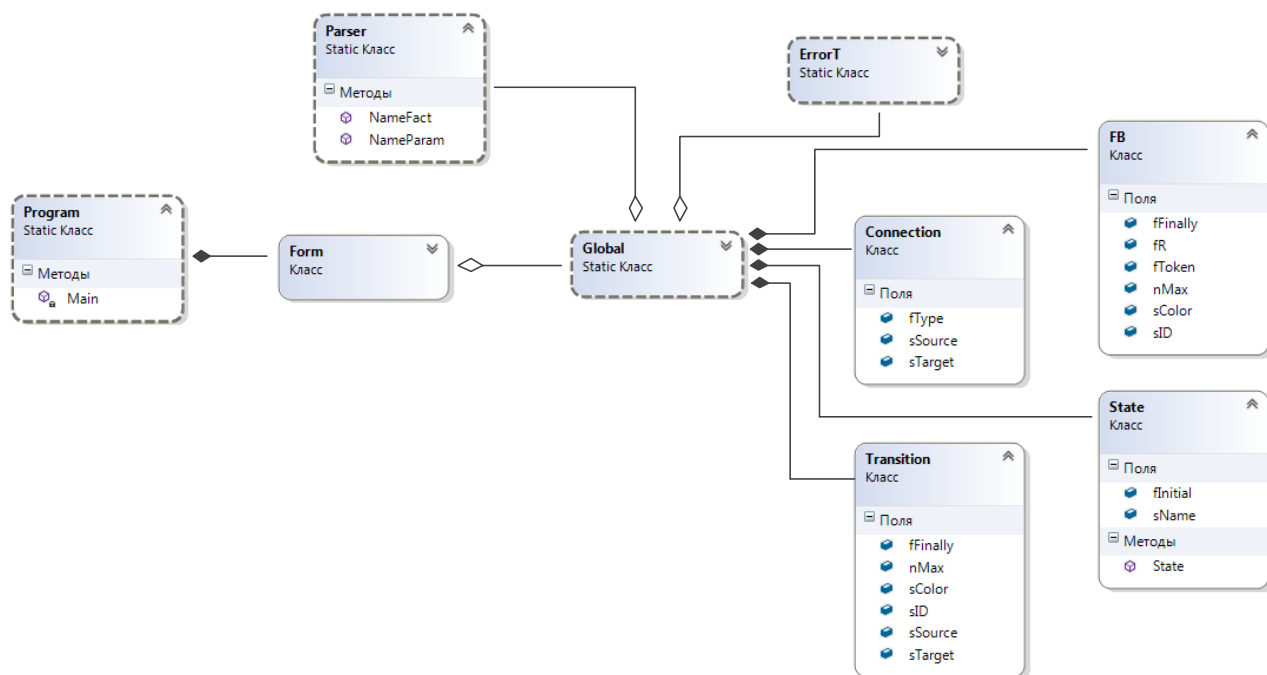


Рисунок 39 – Диаграмма классов

Пользовательский интерфейс транслятора приведён на рисунке 40.

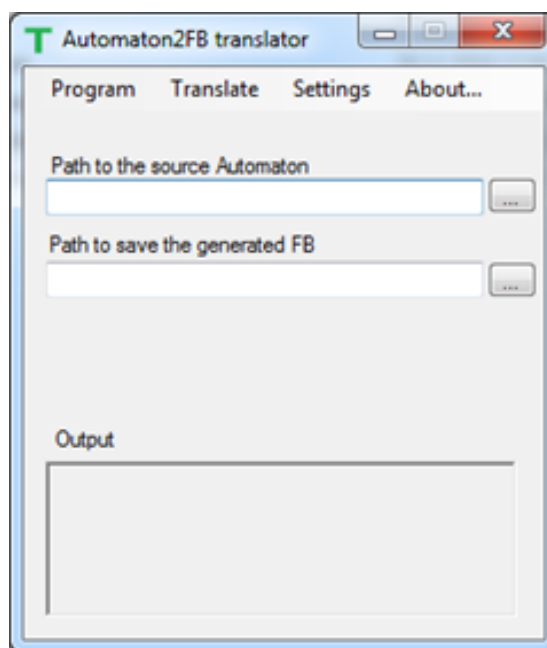


Рисунок 40– Пользовательский интерфейс транслятора

Транслятор работает в следующих режимах:

Only BF – генерируется только один функциональный блок на основе автомата из *Prolog*-описания (для генерации требуется только файл описания автомата на языке *Prolog*);

Automaton 1 (Automaton 2) – замена первого (второго) автомата в проекте сортировки деталей в системе *NxtStudio* на автомат из файла описания на языке *Prolog*;

Both automaton – замена обоих автоматов в проекте ССПД в системе *NxtStudio* на автоматы из файлов описания на языке *Prolog*;

Project – генерация *NxtStudio*-проекта имитационной модели ССПД.

3.3 Реализация недетерминированного конечного и магазинного автоматов для выборки деталей и сборки изделий

В качестве иллюстративного примера рассмотрим НДА, изображённый на рисунке 41. Данный автомат был сформирован и предварительно проверен в системе *JFLAP*, которая предназначена для моделирования автоматных моделей разных типов [16]. Система *JFLAP* может использоваться и для детерминизации НДА.

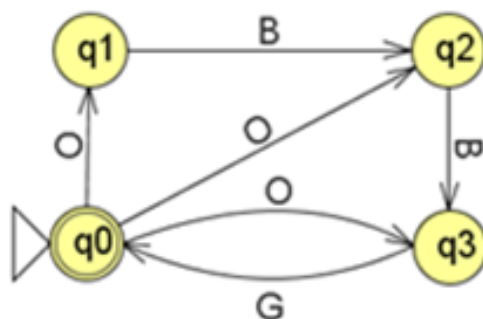


Рисунок 41 – Пример НДА (сформирован в *JFLAP*)

Последовательная реализация НДА в соответствии с Семантикой 1 в виде сети ФБ составного ФБ изображена на рисунке 42.

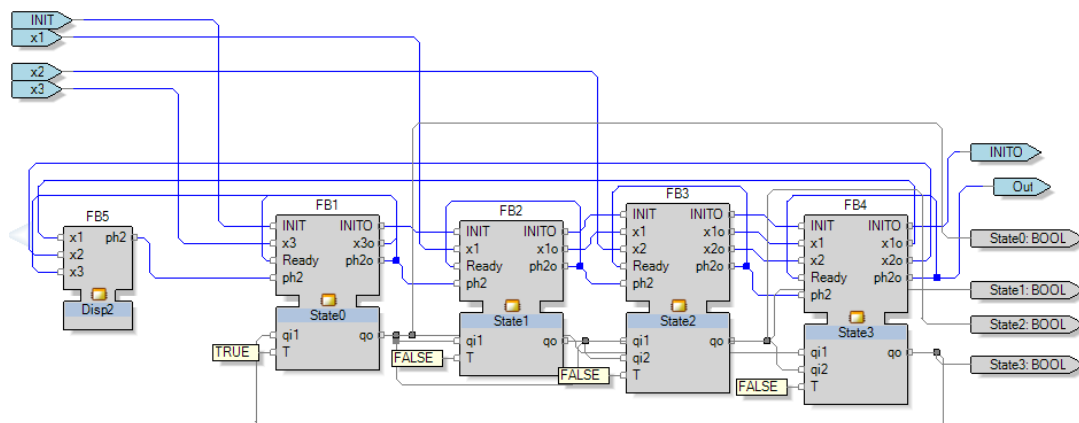


Рисунок 42 – Реализация НДА в соответствии с Семантикой 1

На рисунке 42 показаны ФБ-состояния (*FB1-FB4*) и ФБ-диспетчер (*FB5*). Диспетчер в этом случае является вырожденным – он, по сути, представляет собой дизъюнктор сигналов. Его основная функция – это генерация сигнала запуска второй фазы выполнения.

Реализация Семантики 1 НДА функционирует следующим образом. Через событийные входы *x1..x3* поступает один из входных сигналов автомата. Этот сигнал «прошивает» все ФБ-состояния, которыми он принимается. Данные ФБ при определённых обстоятельствах могут изменить своё внутренне состояние. Когда сигнал прошёл через все блоки цепочки (первая фаза окончена), он поступает в диспетчер, в котором преобразуется в сигнал запуска второй фазы *ph2*. Сигнал *ph2* «прошивает» абсолютно все ФБ-состояния. При прохождении сигнала *ph2* через ФБ внутреннее состояние ФБ выдается наружу. Последний блок цепочки (*FB4*) выдает *Out*, являющийся сигналом окончания обработки входного сигнала в автомате.

Более подробно рассмотрим базисный ФБ *FB3* типа *State2*, отвечающий за реализацию состояния *q2* (рисунок 43).

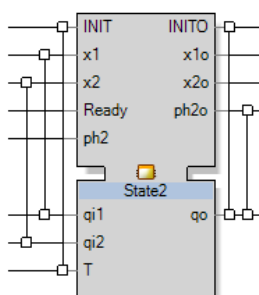


Рисунок 43 – Интерфейс ФБ-состояния в реализации Семантики 1

Интерфейс данного ФБ включает следующие входы-выходы.

Событийные входы:

INIT – инициализация ФБ;

x1, *x2* – входные сигналы НДА;

ph2 – запуск второй фазы выполнения;

Ready – сигнал готовности ФБ принимать новый входной сигнал.

Информационные входы:

qi1 – значение первого состояния-предшественника (истина или ложь);

qi2 – значение второго состояния-предшественника (истина или ложь);

T – начальное значение состояния (используется при инициализации).

Событийные выходы:

INITO – инициализации ФБ окончена;

x1o – обработка входного сигнала *x1* окончена;

x2o – обработка входного сигнала *x2* окончена;

ph2o – вторая фаза выполнения окончена.

Информационные выходы:

qo – значение состояния НДКА *q2* (истина или ложь).

В ФБ также используется внутренняя переменная булевского типа *q* для хранения статуса состояния НДА *q2* (истина или ложь).

На рисунке 44 представлена диаграмма *ECC* для ФБ, представленного выше. Данная диаграмма включает следующие *EC*-состояния: *START* – начальное состояние; *Initialize* – состояние инициализации; *Ready* – состояние готовности, в котором ожидается приход одного из входных сигналов либо сигнала начала второй фазы; *Phase1x1* (*Phase1x2*) – состояние первой фазы выполнения при приходе входного сигнала *x1* (*x2*); *Phase2* – состояние второй фазы выполнения.

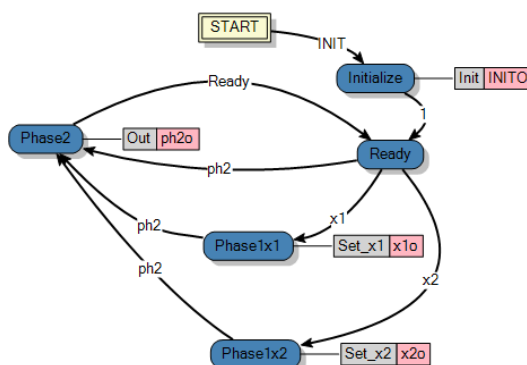


Рисунок 44 – Диаграмма ECC (для ФБ из рисунка 43)

Алгоритмы диаграммы *ECC* на языке *ST* приведены ниже:

```

ALGORITHM Init IN ST:
  qo := T;
END_ALGORITHM
ALGORITHM Set_x1 IN ST:
  IF (qi1 = TRUE) THEN q := TRUE; END_IF;
END_ALGORITHM
ALGORITHM Out IN ST:
  qo := q;  q := FALSE;
END_ALGORITHM

```

Из особенностей схемы, приведённой на рисунке 42, необходимо отметить петли на ФБ-состояниях. Петля связывает событийный выход *ph2o* с событийным входом *Ready* того же самого функционального блока. В результате этого в диаграмме *ECC* на рисунке 44 выходное событие *ph2o* подаётся на ту же диаграмму в виде сигнала *Ready*. Использование такой довольно странной конструкции связано с учётом особенности модели выполнения базисных ФБ в системе *nxtStudio*. В этой модели выполнения используется следующее правило: «При активизации базисного ФБ некоторым сигналом, при срабатывании *ЕС*-перехода этот сигнал не сбрасывается, он остаётся активным для всех переходов, которые помечены этим сигналом. Но каждый *ЕС*-переход, помеченный этим сигналом, может сработать только один раз». Первоначально, в процессе разработки сторожевое условие перехода *Phase2->Ready* было установлено как тождественно-истинное («1»), но с учетом

модели выполнения цепочку *ЕС*-переходов пришлось «разорвать» сигналом *Ready*.

Параллельная реализация НДА в соответствии с Семантикой 2 в виде сети ФБ составного ФБ представлена на рисунке 45.

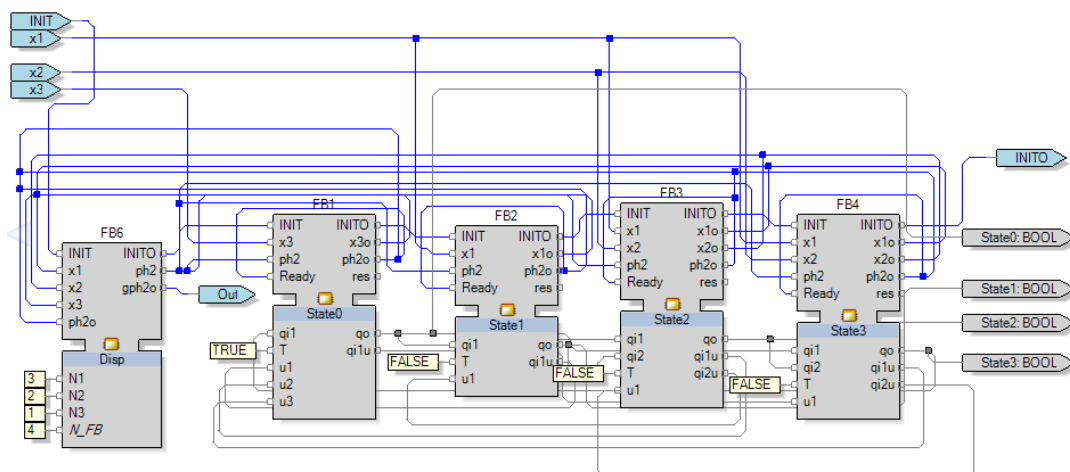


Рисунок 45 – Реализация НДА в соответствии с Семантикой 2

Состав сети функциональных блоков такой же, как и на схеме на рисунке 42, однако реализации компонентных ФБ и соединения между ними отличаются. Реализация *Семантики 2* НДА функционирует следующим образом. Через событийные входы $x1..x3$ поступает один из входных сигналов автомата. Этот сигнал посылается параллельно на все ФБ-состояния, которыми он принимается. Функциональные блоки изменяют свои текущие внутренние состояния и выдают квитанции диспетчеру. Когда диспетчер принял все квитанции от ФБ-состояний, которые обрабатывают определённый входной сигнал, он выдаёт сигнал начала второй фазы *ph2*. Затем все блоки реализации автомата параллельно выдают свои внутренние состояния на выход и сообщают об этом диспетчеру. Когда диспетчер получил квитанции от всех ФБ-состояний, то выдаётся сигнал окончания обработки входного сигнала *Out*. Если приходит входной сигнал, который автомат не может принять, то автомат состояние не меняет. Далее работа автомата продолжается по приходу из внешней среды очередного из сигналов $x1..x3$.

Интерфейс ФБ-состояния в реализации в соответствии с Семантикой 2 (см. рисунок 46) отличается от интерфейса ФБ-состояния в реализации в соответствии с Семантикой 1 следующими добавленными входами-выходами.

Информационные входы: $u1$ – сработал ли переход в (первое) состояние-последователь (истина или ложь).

Событийные выходы: res – сброс внутреннего значения ФБ-состояния;

Информационные выходы: $qi1u$ ($qi2u$) – сработал ли перехода из первого (второго) состояния-предшественника (истина или ложь).

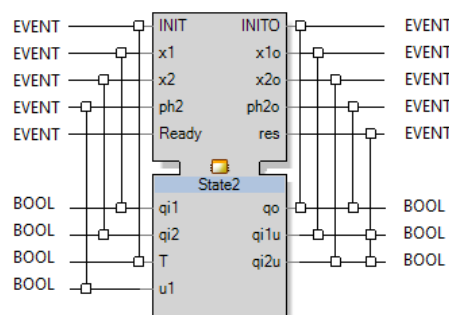


Рисунок 46 – Интерфейс ФБ-состояния в реализации в соответствии с Семантикой 2

На рисунке 47 представлена диаграмма ЕСС для ФБ-состояния из рисунка 46.

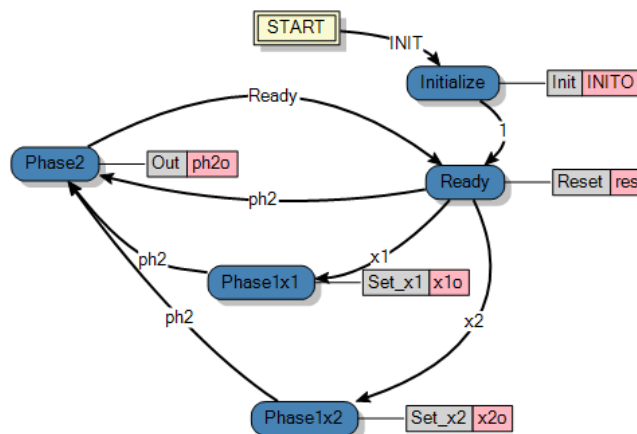


Рисунок 47 – Диаграмма ЕСС (для ФБ из рисунка 46)

Диаграмма ЕСС имеет такие же состояния, что и диаграмма ЕСС на рисунке 44.

Алгоритмы данной диаграммы ЕСС на языке ST приведены ниже.

```

ALGORITHM Reset IN ST:
q := FALSE;    qi1u := FALSE;    qi2u := FALSE;
END_ALGORITHM
ALGORITHM Set_x1 IN ST:

```

```

IF (qil = TRUE) THEN q := TRUE; qil := TRUE; END_IF;
END_ALGORITHM
ALGORITHM Out IN ST:
IF ((q = TRUE) OR (qo = TRUE & (u1 = FALSE))) THEN qo := TRUE;
ELSE qo := FALSE; END_IF;
END_ALGORITHM

```

Работа диаграммы ЕСС кратко представлена ниже. Вначале происходит инициализация ФБ (состояние *Initialize*). Далее ФБ переходит в состояние готовности (*Ready*), в котором происходит сброс внутреннего состояния и значений флагов срабатывания переходов из состояний-предшественников, а затем ожидание одного из входных сигналов, либо сигнала начала второй фазы. Если приходит входной сигнал x_i , то ФБ проверяет активность состояния предшественника и, если данное состояние было активно, то устанавливает в активное собственное состояние и соответствующий флаг срабатывания перехода из состояния-предшественника. При этом диаграмма ЕСС переходит в состояние *Phase1* x_i . По сигналу запуска второй фазы выполнения *ph2* в состоянии *Phase2* благодаря алгоритму *Out* производится вычисление выходного состояния, а также выдача его наружу. Собственное состояние становится равным *TRUE*, если: 1) сработал один из переходов из состояний-предшественников (при этом внутренняя переменная *q* устанавливается в *TRUE*); 2) собственное состояние и раньше (в предыдущем такте) было равно *TRUE*, и при этом не сработал ни один из переходов, которые выходят из этого состояния. В результате этого значение состояния *TRUE* просто сохраняется. Необходимо также отметить, что если в автомат приходит входной сигнал, который не обрабатывается в данном ФБ-состоянии, то в диаграмме ЕСС происходит переход из состояния готовности в состояние обработки второй фазы (*Phase2*).

Главной функцией ФБ-диспетчера является подсчет квитанций, которые приходят от ФБ-состояний, с целью определения момента завершения фазы выполнения и генерации соответствующего сигнала. Интерфейс данного блока приведён на рисунке 48, а диаграмма ЕСС – на рисунке 49. Для подсчёта числа ФБ, которые окончили обработку входного сигнала x_1 (x_2 , x_3), используется

событийный вход $x1$ ($x2$, $x3$), причём общее число ФБ, обрабатывающих входной сигнал $x1$ ($x2$, $x3$) определяется через целочисленную входную переменную $N1$ ($N2$, $N3$).

Для подсчёта числа ФБ, которые окончили вторую фазу выполнения, используется событийный вход $ph2o$. Общее число ФБ, которые участвуют во второй фазе, определяется через целочисленную переменную N_FB . Через событийный выход $ph2$ выдаётся сигнал запуска второй фазы выполнения, а через $gph2o$ – сигнал о её завершении.

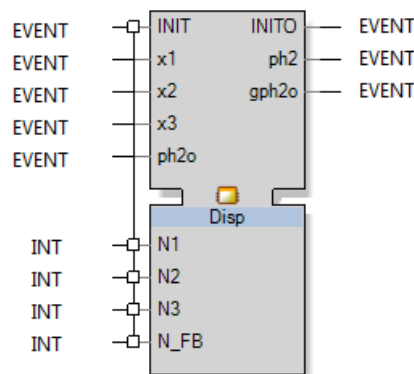


Рисунок 48 – Интерфейс ФБ-диспетчера

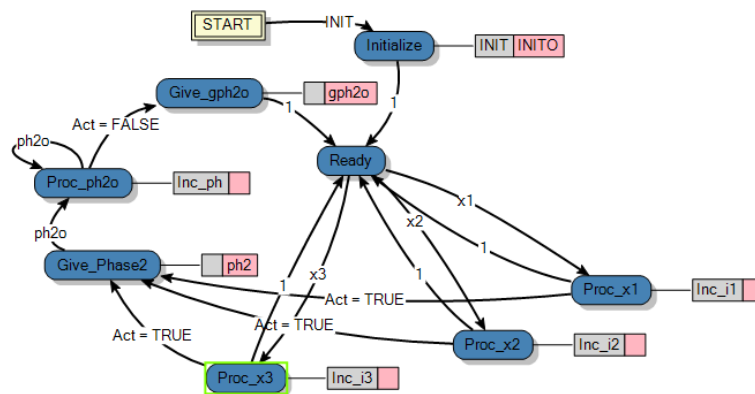


Рисунок 49 – Диаграмма ECC для ФБ-диспетчера

Основные алгоритмы диаграммы *ECC* на языке *ST* приведены ниже.

```

ALGORITHM Inc_i1 IN ST:
i1 := i1 + 1;
IF (i1 = N1) THEN i1 := 0; Act := TRUE; END_IF;
END_ALGORITHM
ALGORITHM Inc_ph IN ST:
ph := ph + 1;
IF (ph = N_FB + 1) THEN ph := 0; Act := FALSE; END_IF;
END_ALGORITHM

```

Для тестирования НДА (для всех вариантов реализаций) использовалось функционально-блочное приложение, представленное на рисунке 50. Составной блок *FB2* типа *NDA_V2* является собственно реализацией НДА, представленного на рисунке 41. Отображение входных сигналов данного НДА на событийные входы блока *FB2* следующее: $O \rightarrow x_1$, $B \rightarrow x_2$, $G \rightarrow x_3$. Событие на событийный выход *out* выдаётся по завершению обработки входного сигнала в реализации НДА. Информационные булевы выходы *State0..State3* отображают статус автоматных состояний $q0..q3$, соответственно. *CAT*-блоки $x1..x3$ типа *GenSignal* необходимы для формирования изображений кнопок и генерации сигналов для тестирования НДА. *CAT*-блок *Pict* типа *Automata* используется для визуализации структуры НДА, а *CAT*-блоки $S0..S3$ типа *ShowBool* для показа статуса состояний автомата (активно/не активно).

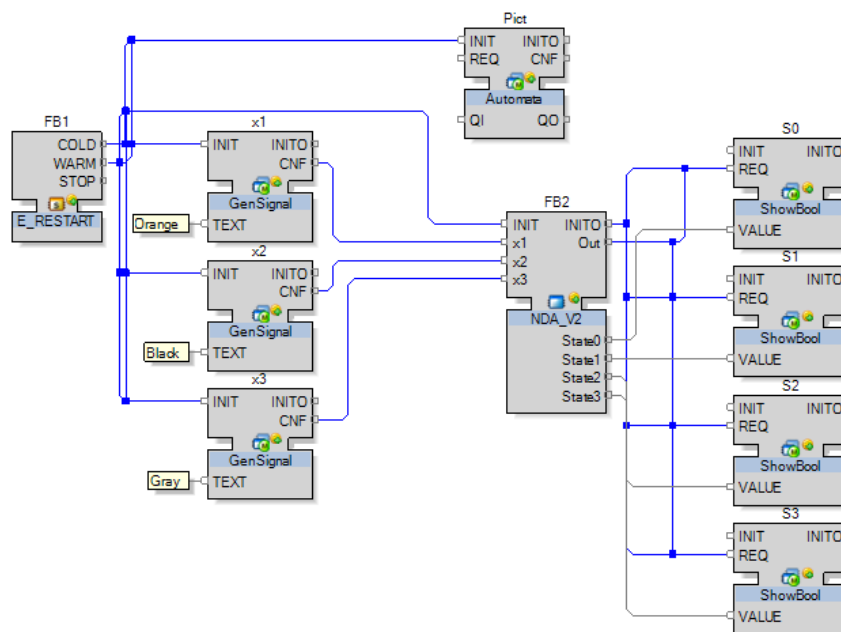


Рисунок 50 – Приложение для тестирования НДА

Графический интерфейс пользователя для системы тестирования НДА, приведенного на рисунке 41, представлен на рисунке 51. Интерфейс включает три кнопки для подачи входных сигналов: *O* (*orange*), *G* (*grey*) и *B* (*blue*), а также графическое представление структуры НДА. Выбор названий входных сигналов определяется названием цветов объектов. Приведённая структура НДА используется также для представления динамики функционирования

автомата путем изменения цвета состояний. Если состояние активно, то его цвет зелёный, иначе – красный.

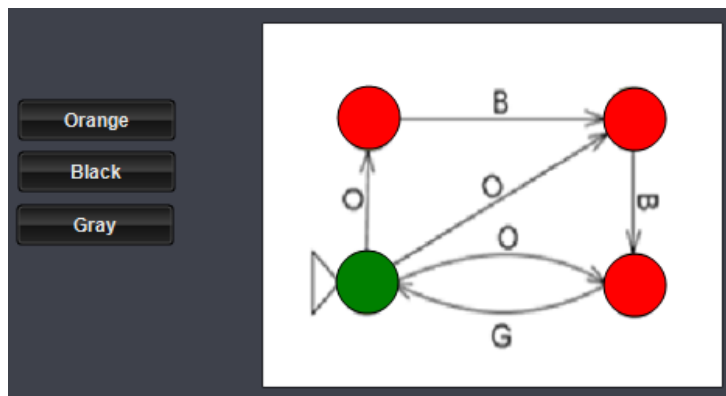


Рисунок 51 – Интерфейс пользователя для системы тестирования НДА

В качестве примера магазинного автомата рассмотрим пример, представленный на рисунке 52. Данный МА распознает КС-язык, заданный следующей формальной грамматикой:

$$S \rightarrow S + T \mid T$$

$$T \rightarrow (S) \mid a$$

Данная грамматика определяет язык скобочных арифметических выражений с операцией сложения над операндом a .

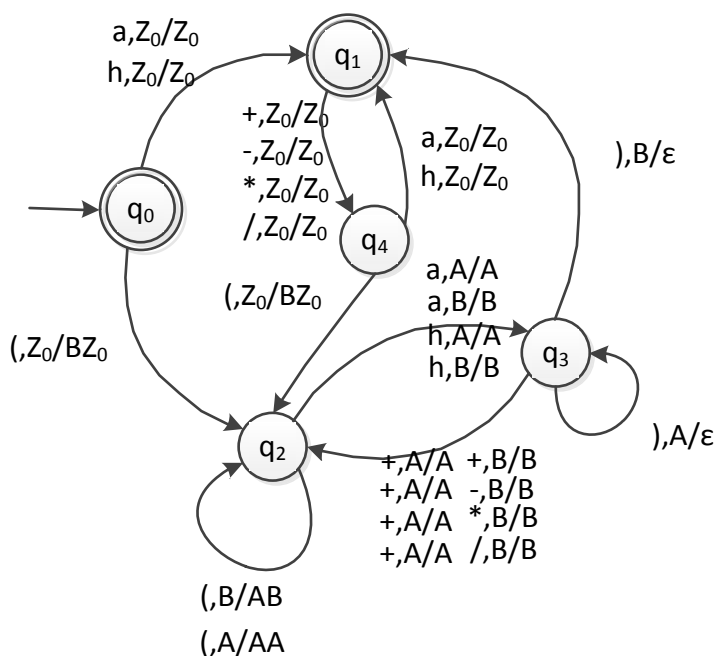


Рисунок 52 – Пример МА

Реализация в данной работе приводится не будет по причине большого объёма. Общая методика реализации была приведена в главе 2.2. Интерфейс пользователя представлен на рисунке 53.

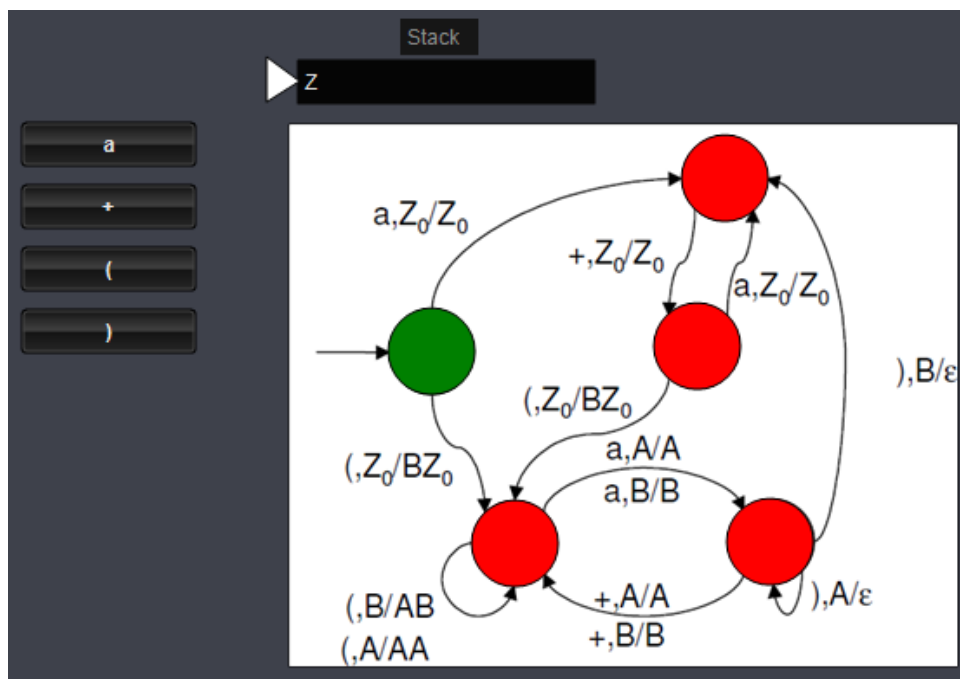


Рисунок 53 – Графический интерфейс для моделирования работы МА

Интерфейс включает четыре кнопки для подачи входных сигналов: «*a*» (переменная), «*+*» (*сложение*) «*(*» (открывающая скобка) и «*)*» (закрывающая скобка), а также графическое представление структуры МА и стек. Приведённая структура МА используется также для представления динамики функционирования автомата путем изменения цвета состояний. Если состояние активно, то его цвет зелёный, иначе – красный.

3.4 Реализация системы сборки сложных изделий на основе магазинных автоматов

В работе [32] было предложено использование магазинных автоматов для проектирования систем сортировки и сборки деталей. В данной работе предлагается модель сборки изделий на основе магазинного автомата-преобразователя. В модели используется концепция технологического процесса сборки изделий, который управляется потоком деталей и жетонов. При этом изделие собирается из поступающих (например, по конвейеру) деталей, а технологические операции описываются (управляющими) жетонами. Жетоны используются для управления сборкой изделий, но при этом сами непосредственно не входят в изделие.

В качестве примера построения модели сборки изделий были взяты три типа деталей: чёрная (*black, b*), оранжевая (*orange, o*) и серая (*gray, g*). Также были определены четыре основных операции сборки (в скобках указаны сокращения, используемые в данной работе): «приклеивание» детали слева (*left, l*) – действие *d1*, «приклеивание» детали справа (*right, r*) – действие *d2*, «приклеивание» детали сверху (*up, u*) – действие *d3* и «приклеивание» детали снизу (*down, d*) – действие *d4*. В качестве геометрической (координатной) основы для выполнения операций принимается последняя деталь, которая установлена на изделие. При таком подходе необходимо предусмотреть решение «проблемы занятой стороны», такого случая при котором необходимая сторона уже занята другой деталью. Например, если к одной детали «приклеить» вторую сверху, то координатная основа сместится и ко второй детали «приклеить» третью снизу не получится. Для решения данной проблемы достаточно пропускать «проблемную» операцию сборки. Кроме того, деталь можно приклеивать на первое свободное место по выбранному направлению (для примера в данной работе третью деталь можно приклеить снизу к первой).

Рассмотрим работу модели сборки для построения изделия одинаковой длины и высоты, напоминающего бумеранг (рисунок 54). Данный МА использует только две технологические операции *left* и *up*. Примеры изготавливаемых изделий этим МА приведены на рисунке 55.

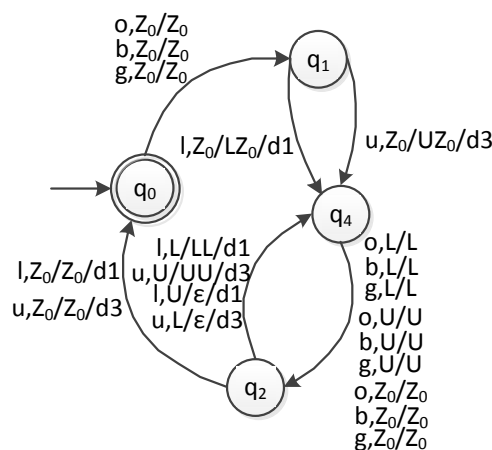


Рисунок 54 – Пример МА, реализующего сборку изделий

Важно отметить, что каждый переход в графическом представлении МА описывается кортежем, включающим

- 1) входной символ;
- 2) магазинный символ вершины стека;
- 3) замещающую последовательность магазинных символов;
- 4) действие (необязательно).

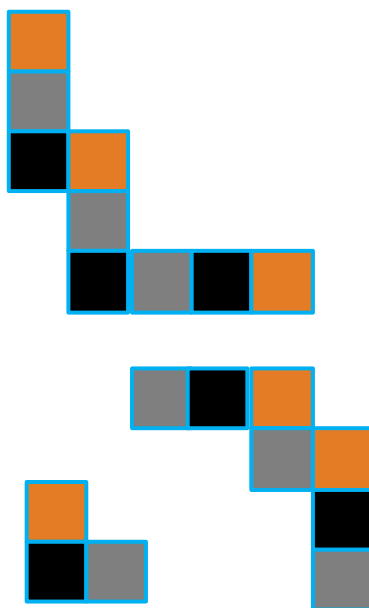


Рисунок 55 – Примеры готовых сконструированных изделий

Как видно из рисунка 54, магазинными символами являются L , U , Z_0 . Начальный магазинный символ – Z_0 . Начальное состояние – q_0 . Конечное состояние – q_0 . Одинаковые начальное и конечное состояния позволяют продолжить функционирование МА, не сбрасывая его.

Рассмотрим замкнутую имитационную модель системы сборки изделий (рисунок 56), которая реализована на основе функциональных блоков стандарта IEC 61499 в системе pxtStudio в соответствии с шаблоном MVC. Сама система состоит из выталкивателя; трёх датчиков для распознавания типа детали или жетона; двух датчиков определения положения выталкивателя; ёмкостного датчика, определяющего, была ли вытолкнута деталь/жетон в шахту; заслонки для забора деталей/жетонов из шахты; приёмника жетонов; места для

построения изделия и светодиодного индикатора, сигнализирующего об окончании сборки. В человеко-машинном интерфейсе также отображается стек управляющего МА для контроля его работы.

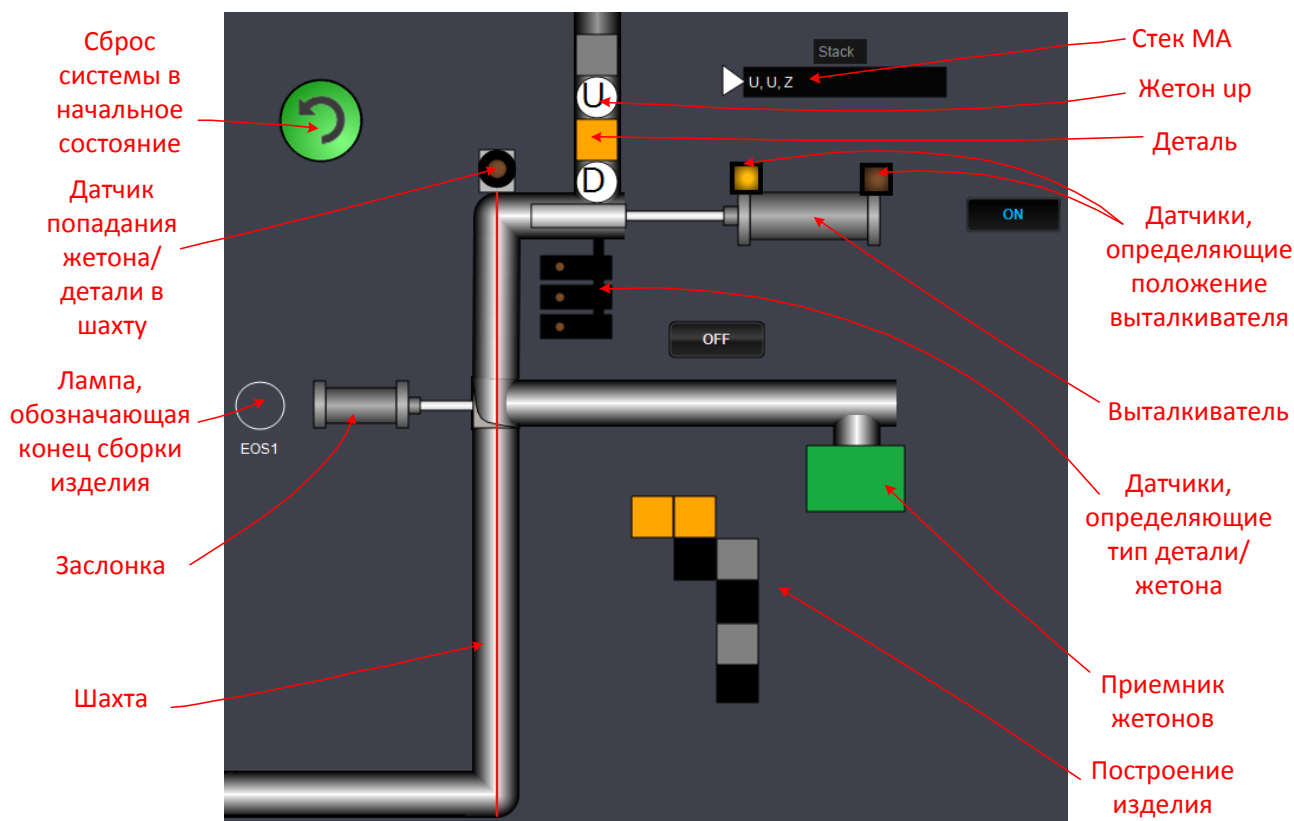


Рисунок 56 – Примеры готовых сконструированных изделий

Имитационная модель системы сборки изделий, включающая систему управления на основе функциональных блоков стандарта IEC 61499 и визуальную модель оборудования, работает следующим образом: сначала генерируется деталь/жетон определенного цвета/типа, который распознаётся с помощью датчиков и попадает на анализ управляющему МА. Если деталь/жетон подходит, то сигнал «выдвинуть заслонку» передаётся в модель сортировки, и, таким образом, деталь/жетон попадает в боковую шахту. Если прибыл жетон, то он попадает в приемник жетонов и сообщает направление «приклеивания» детали. Деталь же продолжает двигаться и «приклеивается» к нужному месту. В противном случае деталь/жетон движутся дальше по шахте. Когда заслонка выдвинута, но деталь/жетон не подходит, то она задвигается и пропускает деталь/жетон, которые падают вниз по шахте.

3.5 Реализация сетей Петри для выборки деталей

В качестве примера реализации выступает грамматика определяющая язык скобочных арифметических выражений с операцией сложения над операндом a , такая же, как и для МА. Сеть Петри для данной грамматики представлена на рисунке 57.

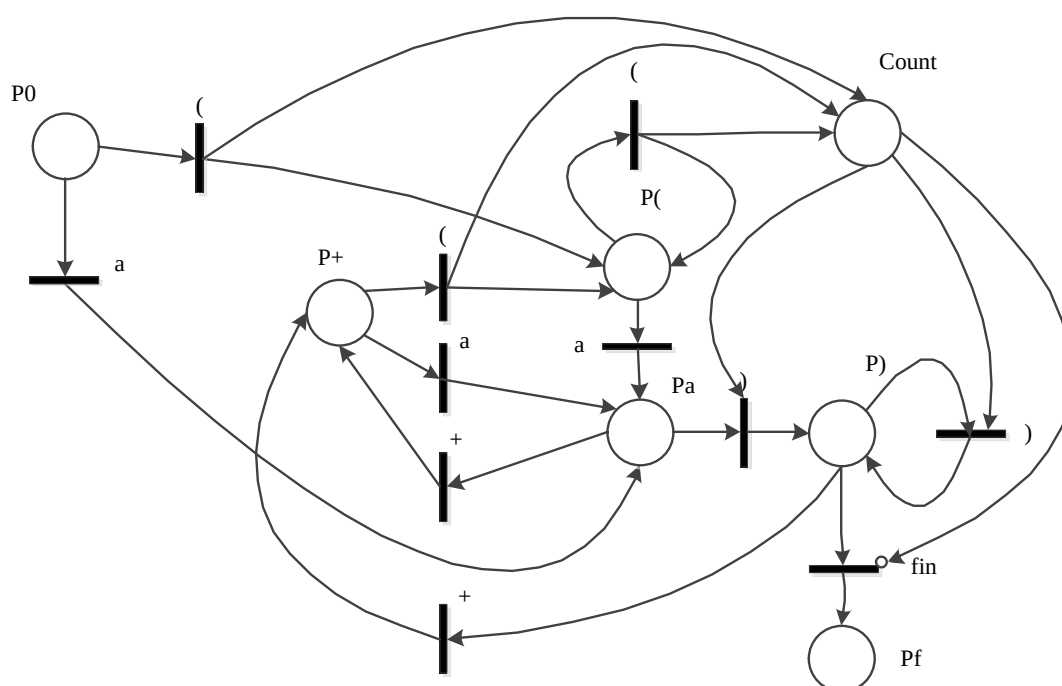


Рисунок 57 – Пример сети Петри

Начальная маркировка данной сети Петри – одна метка в позиции $P0$. Данная сеть Петри имеет конечную позицию Pf , попадание метки в неё означает, что сеть Петри выбрала необходимую последовательность и прерывает к выбору следующей. Данная сеть Петри имеет одно условие: сеть закончит свою работу, когда придёт символ «)» и при этом будет баланс открывающих и закрывающих скобок. Таким образом, выражение должно оканчиваться на закрывающую скобку. Например, выражение $a+a$ распознаваться не будет.

Основная часть реализации сети Петри изображенной на рисунке 57 представлена на рисунке 58. Полный вариант реализации приводиться не будет

из-за объема. На рисунке изображены позиция *P7*, переходы *T10* и *T11*, а также ФБ *StateControl*, отвечающий за пометку сработавшего перехода.

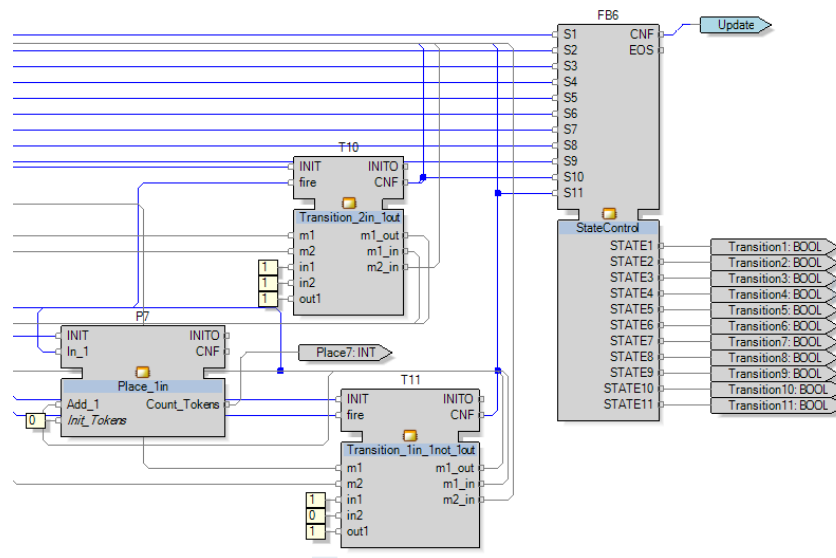


Рисунок 58– Фрагмент ФБ реализации сети Петри из рисунка 57.

ФБ реализация сетей Петри работает следующим образом: вначале внешним сигналом *INIT* инициализируются начальные состояния всех ФБ позиций и переходов. Затем приходит входной сигнал, который разрешает срабатывание одного или нескольких переходов. ФБ-переход проверяет наличие меток у входных позиций, если числа меток хватает для срабатывания перехода, то переход срабатывает. Если в определённый момент могут сработать несколько переходов, то изначально выбираются приоритеты срабатывания переходов. В приведённом примере в каждый момент времени может сработать только один переход. Во время срабатывания переход отправляет сигналы и числовые переменные на входные и выходные позиции для вычитания и добавления определённого количества меток соответственно. Также ФБ-переход подает сигнал о своем срабатывании на ФБ *StateControl*, который соответственно меняет значение своих состояний. ФБ-позиции подают на выход текущее число меток.

Графический интерфейс в системе *nxtStudio* представлен на рисунке 59.

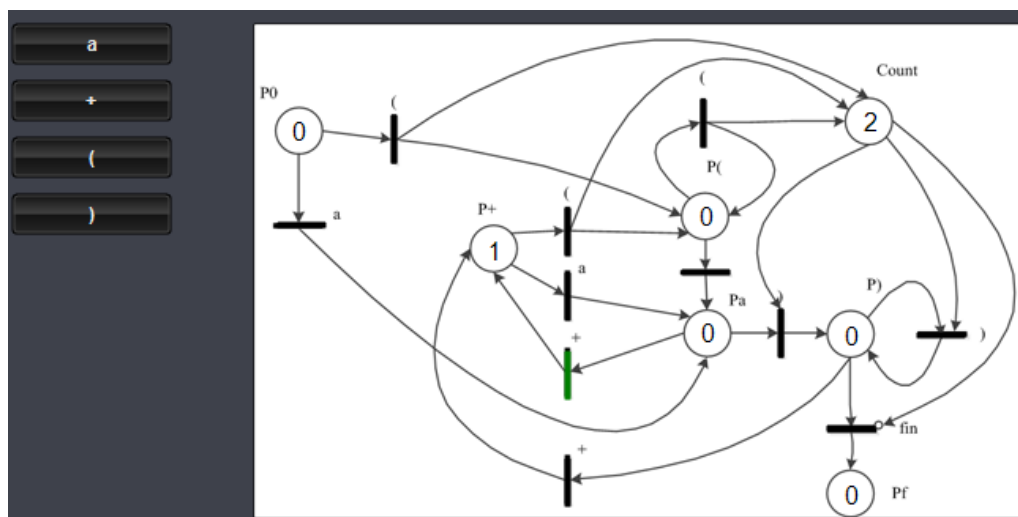


Рисунок 59 – Графический интерфейс в nxtStudio

Слева находятся четыре кнопки, с помощью которых можно подавать входные сигналы на сеть Петри. Справа изображена сама сеть Петри, причем внутри каждой позиции указано число – текущее состояние позиции, а также зеленым цветом обозначен последний сработавший переход. Приведённая модель СП используется также для представления динамики функционирования сети Петри путем изменения цвета переходов и показа числа меток в позициях. Если произошёл переход, то его цвет меняется на зелёный, иначе – чёрный.

3.6 Выводы

1. Разработан транслятор Prolog-описаний конечных автоматов в функционально-блочную имитационную модель системы сортировки последовательностей деталей, позволяющий автоматизировать процесс создания управляющих приложений для систем данного класса и средств их валидации.
2. Разработана функционально-блочная имитационная модель сборки 2D-изделий, управляемая магазинным автоматом-преобразователем, распознающим язык скобочных выражений, в среде nxtStudio, позволяющая

оценить полезность использования магазинных автоматов-преобразователей в системах сборки изделий.

3. Реализованы пользовательские интерфейсы функционально-блочных моделей конечных автоматов, магазинных автоматов и сетей Петри в среде nxtStudio, позволяющие облегчить процесс тестирования предложенных методик.

4. Для поддержки и тестирования методик, предложенных во втором разделе, были разработаны следующие программные продукты, зарегистрированные в Роспатенте:

а) «Транслятор описания селектирующих автоматов в замкнутую модель системы сортировки последовательностей деталей». - Свидетельство о регистрации программы для ЭВМ RUS 2018612150 26.12.2017.

б) «Функционально-блочная реализация магазинного автомата для распознавания языка скобочных арифметических выражений». – (в процессе регистрации).

ЗАКЛЮЧЕНИЕ

В результате работы получены следующие научные и практические результаты:

1. формальная модель детерминированного конечного автомата для спецификации и выборки последовательностей деталей в промышленных системах сортировки;
2. подход к реализации конечных автоматов на основе ФБ с использованием механизма передачи маркеров;
3. система вывода графов, определяющая процесс порождения структур систем ФБ ИЕС 61499 из конечноавтоматной спецификации селектируемых последовательностей деталей;
4. методика трансляции автоматных спецификаций селектируемых последовательностей деталей в систему управления сортировкой на основе ФБ ИЕС 61499;
5. разработан автоматический транслятор prolog-описаний конечных автоматов в имитационную модель системы сортировки последовательностей деталей;
6. подход к двухфазной реализации НДА на основе ФБ стандарта ИЕС 61499, когда в виде блоков представляются состояния автомата;
7. метод реализации детерминированных магазинных автоматов на основе ФБ;
8. представлена модель сборки изделий, управляемая МА-преобразователем в среде nxtStudio;
9. методика реализации сетей Петри на основе функциональных блоков стандарта ИЕС 61499.

Направления дальнейших исследований следующие:

- 1) разработка методики реализации А-сетей на основе функциональных блоков стандарта ИЕС 61499;

- 2) построение модели выборки деталей и сборки изделий из LEGO-блоков;
- 3) оценка ФБ-реализаций других моделей переходов состояний;
- 4) сравнение всех предложенных методов для конкретных ситуаций.

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

BSEM – Buffered sequential execution model

ECC – Execution Control Chart

UML – Unified Modelling Language

XML – eXtensible Markup Language

ГСАП – граф-схема алгоритма

ДА – детерминированный автомат (конечный)

ДМА – детерминированный магазинный автомат

КС – контекстно-свободный

КФС – киберфизическая система

МА – магазинный автомат

НДА – недетерминированный автомат

СИФБ – сервисный интерфейсный функциональных блок

СКУ – система канонических уравнений

СП – сеть Петри

СПА – система промышленной автоматике

ССПД – система сортировки последовательностей деталей

ФБ – функциональный блок

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Manufacturing Success in the 21st Century: A Strategic View, Integrated Manufacturing Technology Initiative, 16 July 2000.
- 2) EasyVeep. Ball sorting system [Электронный ресурс]. – http://www.easyveep.com/modules.php?akt_modul=7&akt_Lang=2
- 3) Rose K., Eldridge S., Chapin L. The Internet of Things: An Overview. Internet Society, October, 2015, 51 p.
- 4) Дубинин В.Н., Сенокосов И.В., Вяткин В.В., Климкина Л.П. Преобразование автоматных спецификаций в функционально-блочную реализацию системы управления сортировкой последовательностей деталей // Известия ВУЗов. Поволжский регион. Технические науки. – 2017. – № 2 – С. 3-18.
- 5) Дубинин В.Н., Сенокосов И.В., Войнов А.С., Дроздов Д.Н., Вяткин В.В. Функционально-блочная реализация недетерминированных конечных автоматов // Труды Международной научно-технической конференции “Современные информационные технологии”, Пенза, 2017, вып.25. – С. 5-18.
- 6) Сенокосов И.В. Функционально-блочная реализация магазинных автоматов // Новые информационные технологии и системы (НИТиС 2017) : тр. XIV Междунар. науч.-техн. конф. – Пенза, 2017. – С. 45 – 49.
- 7) Дубинин В.Н., Сенокосов И.В., Войнов А.С., Вяткин В.В. Функционально-блочная реализация селектирующих сетей Петри // Международный научно-технический форум «Современные технологии в науке и образовании» (СТНО-2018), Рязань, 2018 (в печати)
- 8) Hassibi, K.M. A Multi-Sensor Robotics System For Object Recognition / K.M. Hassibi, K.A. Loparo, F.L. Merat // Proc. SPIE 1002, Intelligent Robots and Computer Vision VII, 1989.

- 9) Cui T., Zeng Q., Zhang D. Recognition Algorithm Design and Complex Analysis for Languages of S-Nets // Information Technology Journal, N 10, 2011, pp. 106-112.
- 10) Ghanem N.; DeMenthon D.; Doermann D.; Davis L. Representation and Recognition of Events in Surveillance Video Using Petri Nets // Conference on Computer Vision and Pattern Recognition (CVPRW '04), 2004.
- 11) Ortega F.R., Hernandez F., Barreto A., Rische N.D., Adjouadi M., Liu S.. Exploring modeling language for multi-touch systems using Petri nets // Proc. 2013 ACM Int. Conf. on Interactive tabletops and surfaces (ITS '13). ACM, NY, USA, 2013, pp. 361-364.
- 12) Choppy C., Bertrand O., Carle P. Coloured Petri Nets for Chronicle Recognition // Int. Conf. on Reliable Software Technologies - Ada-Europe 2009. Lecture Notes in Computer Science, vol 5570, Springer, pp 266-281.
- 13) Fernández V.R.; Pardo A.G.; Camacho D. Automatic Procedure Following Evaluation using Petri net-based Workflows // IEEE Transactions on Industrial Informatics (Volume: PP, Issue: 99), DOI: 10.1109/TII.2017.2779177.
- 14) Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. – М : Вильямс, 2002.
- 15) Войнов А.С., Сенокосов И.В., Дубинин В.Н., Климкина Л.П. Детерминизация недетерминированных автоматов на основе графотрансформационного подхода // Труды Международной научно-технической конференции “Современные информационные технологии”, Пенза, 2016, вып.23. – С. 6-17.
- 16) JFLAP Web-site [Электронный ресурс]. URL: <http://www.jflap.org/>.
- 17) Котов В.Е. Сети Петри. – М.: Наука, 1984. – 160 с.
- 18) Дубинин В.Н. Асинхронное моделирование NCES-сетей // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2009. – № 2. - С. 3-14.

- 19) Vyatkin V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, 3rd edn. – Pittsburgh: Instrumentation Society of America, 2016. – 261 p.
- 20) International Standard IEC 61499. Function blocks for industrial-process measurement and control systems. Part 1: Architecture / International Electrotechnical Commission. – Geneva, 2005. – 111 p.
- 21) Дубинин В. Н., Вяткин В. В. Семантический анализ описаний систем управления промышленными процессами на основе стандарта IEC 61499 с использованием онтологий //Известия высших учебных заведений. Поволжский регион. Технические науки. – 2010. – №. 3.
- 22) Колтунцев А. В., Золотарев С. В. Стандарт IEC 61499 и система программирования контроллеров ISaGRAF 5: от теории к практике //Rational Enterprise Management. – 2009. – №. 2.
- 23) nxtStudio (nxtControl) [Электронный ресурс]. – <http://www.nxtcontrol.com/>
- 24) Dubinin V. N., Senokosov I., Vyatkin V. Auto-Generation of Distributed Automation Software Based on Formal Product Line Specification //8th International Conference, HoloMAS 2017, Lyon, France, August 28–30, 2017. – 2017. – С. 80-91.
- 25) Grunske, L., Geiger L., Zuendorf A., Eetvelde N. V., Gorp P.V., Varro D. Graph Transformation for Practical Model Driven Software Engineering. Berlin; Heidelberg : Springer Verlag, 2005, pp. 91–118.
- 26) Вашкевич Н.П., Бикташев Р.А. Недетерминированные автоматы и их использование для реализации систем параллельной обработки информации : моногр. – Пенза : Изд-во ПГУ, 2016. – 394 с.
- 27) Сенокосов И.В., Дубинин В.Н., Вяткин В.В. Функционально-блочная реализация управления для системы выборки

- последовательностей деталей // Новые информационные технологии и системы (НИТиС 2016) : тр. XII Междунар. науч.-техн. конф. – Пенза, 2016. – С. 369 – 373.
- 28) Сенокосов И.В., Войнов А.С., Дубинин В.Н. Автоматический транслятор Prolog-описаний селективирующих автоматов в имитационную модель системы сортировки последовательностей деталей // Сб. статей IV Ежегодной межвузовской науч.-практ. конф. «Информационные технологии в науке и образовании. Проблемы и перспективы», Пенза, 2017. – С. 66-68.
- 29) Транслятор описания селективирующих автоматов в замкнутую модель системы сортировки последовательностей деталей. Войнов А.С., Дроздов Д.Н., Дубинин В.Н., Сенокосов И.В. Свидетельство о регистрации программы для ЭВМ RUS 2018612150 26.12.2017
- 30) SWI-Prolog [Электронный ресурс]. – <http://www.swi-prolog.org/>
- 31) Bray T. et al. Extensible markup language (XML) //World Wide Web Journal. – 1997. – Т. 2. – №. 4. – С. 27-66.
- 32) Дубинин В.Н., Сенокосов И.В., Климкина Л.П., Вяткин В.В. Использование моделей магазинных автоматов в проектировании технологических процессов сортировки и сборки изделий // Сб. трудов II Международной науч.-техн. и науч.-методической конф.: «Современные технологии в науке и образовании» (СТНО-2017), Рязань, 2017, том 1. – С. 10-14.
- 33) Сенокосов И.В., Войнов А.С. Программная модель системы сборки сложных изделий на основе магазинных автоматов-преобразователей в среде NxtStudio // Новые информационные технологии и системы (НИТиС 2017) : тр. XIV Междунар. науч.-техн. конф. – Пенза, 2017. – С. 286 – 289.
- 34) Сенокосов И.В., Войнов А.С. Дубинин В.Н. Онтологическое и графовое представления 3D-конструкций из LEGO-блоков // Сб. статей V

- Ежегодной межвузовской науч.-практ. конф. «Информационные технологии в науке и образовании. Проблемы и перспективы» (МНПК – 2018), Пенза, 2018 (в печати)
- 35) Горбатов В. А. Семантическая теория проектирования автоматов //М.: Энергия. – 1979.
- 36) Питерсон Д. Теория сетей Петри и моделирование систем. – Мир, 1984.
- 37) Гордеев А. В., Молчанов А. Ю. Системное программное обеспечение. – 2001.
- 38) Дубинин В.Н., Вяткин В.В. Модели функциональных блоков IEC 61499, их проверка и трансформации в проектировании распределённых систем управления: монография. – Пенза: Изд-во ПГУ, 2012. – 348 с
- 39) Darvas D., Majzik I., Viñuela E. B. Conformance checking for programmable logic controller programs and specifications //Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on. – IEEE, 2016. – С. 1-8.

Приложение А – Свидетельство о регистрации программы

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018612150

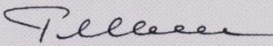
**Транслятор описания селектирующих автоматов в
замкнутую модель системы сортировки
последовательностей деталей**

Правообладатель: *Федеральное государственное бюджетное
образовательное учреждение высшего образования «Пензенский
государственный университет» (RU)*

Авторы: *Войнов Артем Сергеевич (RU), Дроздов Дмитрий
Николаевич (RU), Дубинин Виктор Николаевич (RU), Сенокосов
Илья Владимирович (RU)*

Заявка № **2017664094**
Дата поступления **26 декабря 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ **13 февраля 2018 г.**

Руководитель Федеральной службы
по интеллектуальной собственности

 **Г.П. Ивлиев**



ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЧЕТНАЯ ГРАМОТА

награждается

СЕНОКОСОВ
Илья Владимирович

занявший второе место в конкурсе студенческих докладов
на XIII Международной научно-технической конференции
«НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ»
(НИТиС-2016)

секция «Проектирование и администрирование
информационных систем»



Ректор университета

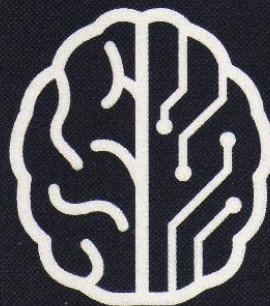
А. Д. Гуляков



2016 год

2016 год

ФГБОУ ВО «ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ



НЕДЕЛЯ
НАУКИ 2017

ДИПЛОМ

НАГРАЖДАЮТСЯ

СЕНОКОСОВ ИЛЬЯ ВЛАДИМИРОВИЧ
ВОЙНОВ АРТЁМ СЕРГЕЕВИЧ

РУКОВОДИТЕЛЬ: ДУБЕНИН ВИКТОР НИКОЛАЕВИЧ

ПОБЕДИТЕЛИ СЕКЦИИ

«ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ, СЕТИ И КОМПЛЕКСЫ»

IV ЕЖЕГОДНОЙ МЕЖВУЗОВСКОЙ НАУЧНО-ПРАКТИЧЕСКОЙ
КОНФЕРЕНЦИИ «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
В НАУКЕ И ОБРАЗОВАНИИ.

ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ»

РЕКТОР ПГУ



А.Д. Гуляков

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЧЕТНАЯ ГРАМОТА

награждается

И. В. СЕНОКОСОВ

за лучший студенческий доклад
на XIV Международной научно-технической конференции
«НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ»
(НИТнС-2017)

секция «Программное и аппаратное обеспечение
информационно-вычислительных систем»



Ректор университета

А. А. Гуляков

2017 год

2017 год

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ПЕНЗЕНСКИЙ **PENZA**
ГОСУДАРСТВЕННЫЙ **STATE**
УНИВЕРСИТЕТ **UNIVERSITY**

PNZGU.RU



БЛАГОДАРСТВЕННОЕ ПИСЬМО

НАГРАЖДАЕТСЯ

студент группы 16ВВм1 факультета вычислительной техники

Сенокосов Тлья Владимирович

за особое прилежание, трудолюбие и интерес к науке

Ректор университета



А.Д. Гуляков

ПЕНЗА • 2018