РЕАЛИЗАЦИЯ НА ПЛИС ПОТОКОВОГО ПРОЦЕССОРА ОБРАБОТКИ ПРОТОКОЛА ЛОГИЧЕСКОГО И ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

С.В. Леонов, к.т.н., доцент,
С.В. Говорухин, студент гр. 8ЕМ01
Томский политехнический университет
E-mail: svg4@tpu.ru

Введение

При реализации системы на кристалле средствами программируемых логических интегральных схем имеет место быть проблема взаимодействия разрабатываемой системы с внешними узлами и устройствами. Для решения данной задачи создаются собственные интерфейсы или применяются готовые решения, в зависимости от особенностей задачи.

Системы, встраиваемых в робототехнические комплексы, практически всегда реализуются с собственными протоколами информационного и логического взаимодействия, а также критерий жесткости времени играет не последнюю роль. По этой причине оправдана реализация собственного модуля интерфейса для обработки протокола логического и информационного взаимодействия – потокового процессора.

Реализация потокового процессора

Задачи, стоящие перед потоковым процессором схожи с задачами, решаемыми вычисляющим процессором [1], за исключением прямого взаимодействия с обслуживаемым интерфейсом, ниже перечислены основные задачи и ключевые особенности.

Задачи потокового процессора:

1) декодирование инструкций и выделение данных из входящего потока;
2) выполнение инструкций;
3) взаимодействие с памятью;
4) формирование исходящего потока.

Особенности потокового процессора:

1) отсутствие АЛУ;
2) отсутствие счётчика команд;
3) отсутствие вектора прерываний;
4) единое и заведомо известное время выполнения инструкций;
5) прямое взаимодействие с интерфейсом;
6) малый размер при реализации на ПЛИС;

Для реализации потокового процессора был использован ПЛИС Intel Cyclone IV, разработка производилась в среде Quartus Prime 18.0.0 на языке Verilog.

Основным алгоритмом работы потокового процессора является алгоритм конечных автоматов, выполненный в виде автомата Мура, по этой причине выходы состояний и переходы на следующее состояния формируются исключительно на следующем такте после события перехода [2].

Диаграмма состояний конечного автомата потокового процессора представлена на рисунке 1.

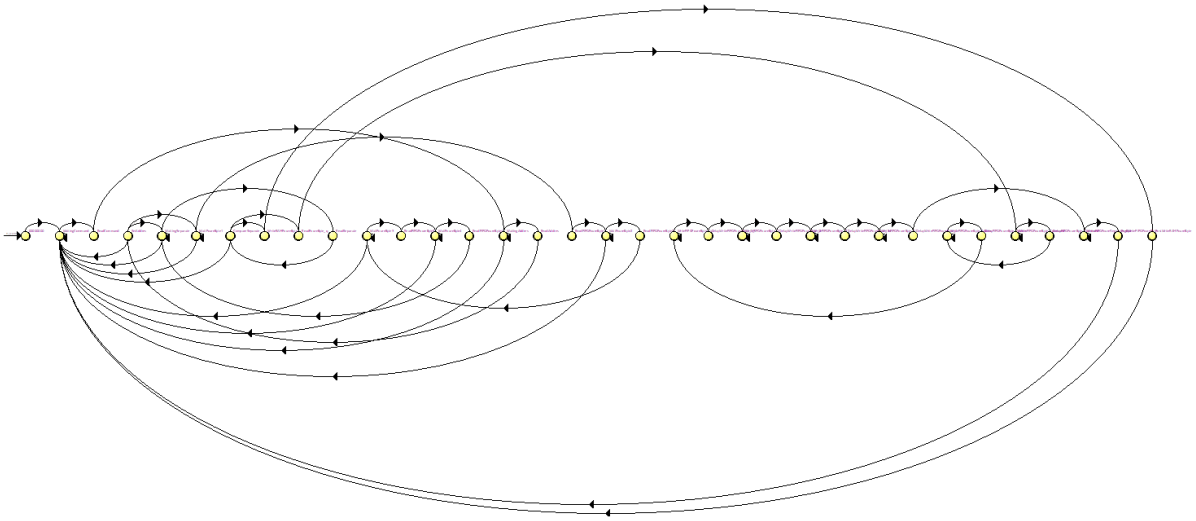

Рис. 1. Диаграмма состояний потокового процессора

Для обработки некоторых внешних сигналов процессора, скорость и синхронность распространения которых не критична, сформирована комбинационная схема, что позволяет исключить избыточные состояния автомата и сократить время его работы.

Так как процессор обрабатывает поток информации, а поток может прерваться, не передав полный пакет, то был реализован механизм сторожевого таймера, который сбрасывает состояние конечного автомата на начальное.

Реализуемый процессор спроектирован для интеграции в систему на кристалле, имитирующую работу бесколлекторного двигателя постоянного тока (БДПТ) с регулятором скорости оборотов. В рамках данной системы процессор выполняет следующие задачи:

1) принимает поток данных по последовательному асинхронному интерфейсу;
2) декодирует инструкции;
3) при инструкции записи, изменяет уставки регулятора модели двигателя;
4) при инструкции чтения, формирует исходящий поток со значением скорости модели двигателя;

Тестирование потокового процессора

Для проверки надёжности спроектированного потокового процессора необходимо провести комплекс тестов, по этой причине было разработано специализированное программное обеспечение средствами фреймворка QT, на языке C++.

Окно программы тестирования потокового процессора представлено на рисунке 2.

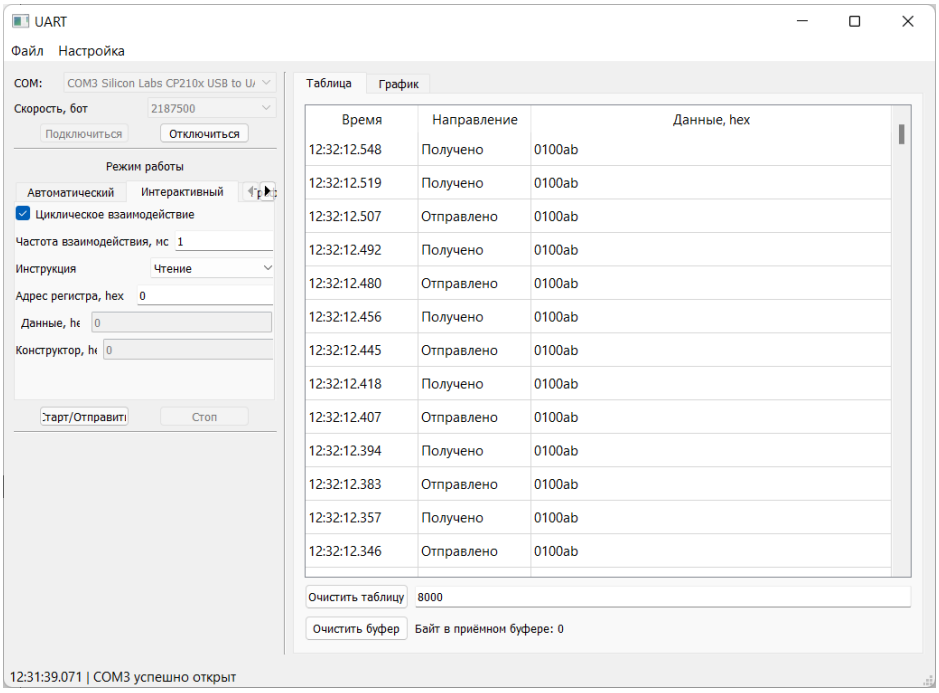

Рис. 2. Окно программы тестирования работы потокового процессора

Тестирование проводилось путём записи числа в регистр памяти, а затем подачи 4000 инструкций чтения значения из памяти с интервалом в 1мс на максимальной скорости для микросхемы физического уровня (2187500 бот).

В результате тестирования было получено 4000 значений, хранимых в запрашиваемом регистре памяти, соответствующих ранее записанному.

Заключение

В результате проведения тестирования можно сделать вывод о том, что разрабатываемый потоковый процессор корректно выделяет и декодирует инструкции из входящего потока, а также корректно формирует исходящий поток данных.

Список использованных источников

1. Дэвид Х., Сара Х. Цифровая схемотехника и архитектура компьютера // Издательство Morgan Kaufman. 2015 – 927 с.
2. Clifford E. Cummings. Finite State Machine (FSM) Design & Synthesis using SystemVerilog - Part I // SNUG-2019. – 74с.