Трансляция абстрактного конечного автомата в описание аппаратуры на языке Verilog

Цель:

Разработка программного продукта, позволяющего упростить проектирование интегральных схем (ИС) с использованием технологии автоматного программирования

Автор: Пимкин Артём, 11 класс

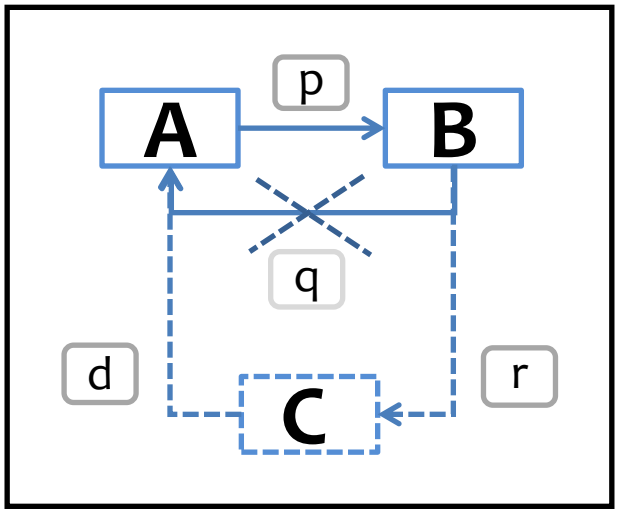
Научный руководитель: Дединский Илья Рудольфович, МФТИ

Консультант: Фадеев Роман, Intel

Автоматное программирование

Преимущества при проектировании

- Наглядное представления кода
- Удобство изменения и отладки
- Автоматическая верификация алгоритма



Применение

- Позволяет разделить этапы *проектирования* и *реализации*
- Окончательный исходный код **автоматически генерируется** по автоматной модели

```
switch (state)
{
     case A:
     if (p == true)
          state = B;
     break;
     case B:
     …
}
```

Специализированные ИС (ASIC)

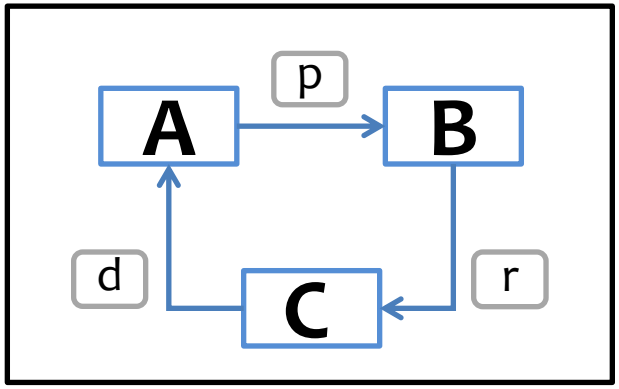
Преимущества перед процессорами общего назначения

- Более простая конструкция за счёт специализации
- Логика задаётся напрямую «в железе», без затрат на декодирование инструкций

Проблемы использования в качестве блоков управления

- Структура конечного автомата (КА) плохо «ложится» на языки описания аппаратуры
- Ручная разработка требует знания принципов работы цифровой логики

Логично применить подход автоматической генерации кода




```
always @(p, d, r)
begin
  switch (state)
  begin
    case A:
    if (p && !d && !r)
      state <= B;
    …
  end
end
always (state)
…
```

Задачи работы

- Разработка транслятора, способного по описанию КА автоматически генерировать код на языке описания аппаратуры Verilog
- Разработка простого и интуитивно понятного языка для описания КА
- Испытание работоспособности генерируемого кода на отладочной плате с использованием FPGA
  - FPGA (*Field Programmable Gate Array*) — многократно перезаписываемый аналог «макетной платы» для прототипирования интегральных схем

SML (State Machine Language)

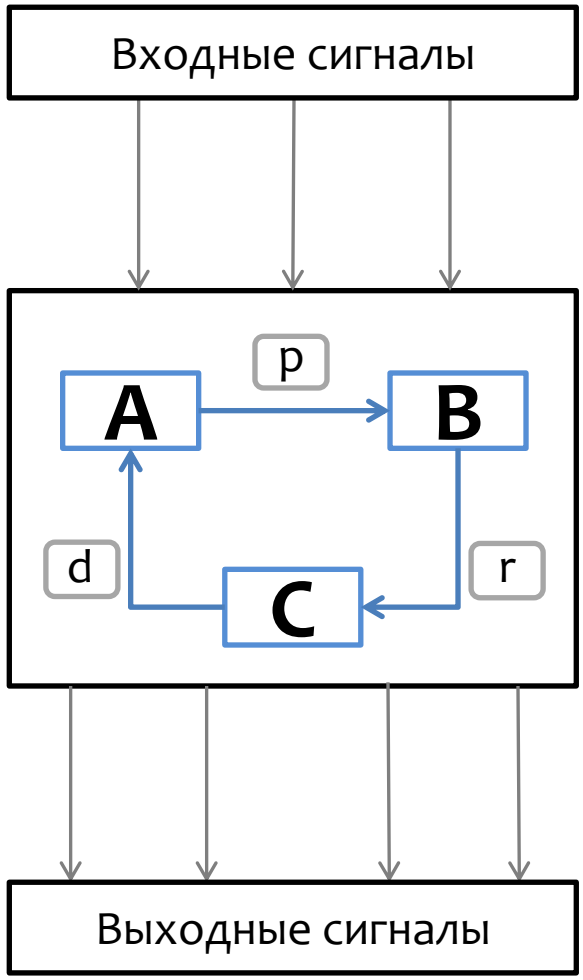
Был разработан собственный язык текстового описания автоматов.

Преимущества перед аналогами

- Интуитивно понятный за счёт императивности
- Удобный для ручного редактирования

Модель автомата в SML

- Автомат — «чёрный ящик»
- Конечный набор абстрактных входных и выходных сигналов
- Действия (изменения выходных сигналов)
  - по переходам между состояниями
  - при входе в конкретное состояние



Синтаксис SML

```
states    {initial, working}
inputs    {reset, init, cycle}
outputs   {notify}
```

Блок декларации

```
switch
{
    state initial
    {
        stopsignals;

        transition on (init)
                   to working
                   do {}
    }
    state working
    {
        stopsignals;

        transition on (reset)
                   to init
                   do {}
        transition on (cycle)
                   to working
                   do {emitsignal notify;}
    }
}
```

Блок описания логики

Описание состояния

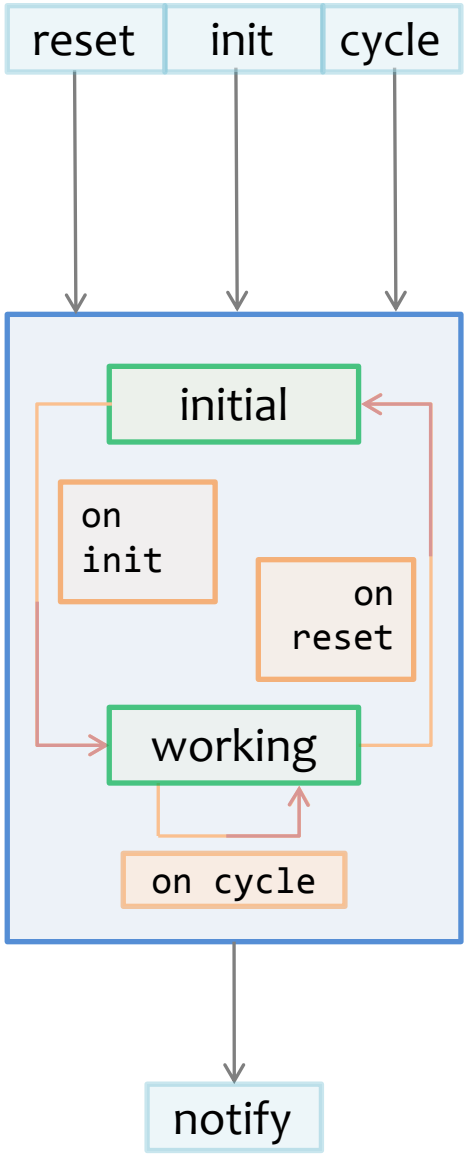
Действия входа в состояние

Переход между состояниями

Блок описания состояния

Действия входа в состояние

Переход между состояниями



Транслятор

Лексический разбор

- Разбитие исходного текста на лексемы (абстрактные единицы языка)

Синтаксический анализ

- Обработка потока лексем методом рекурсивного спуска (recursive descent)

Обход дерева разбора

- Воссоздание описания автомата в подходящем для анализа виде

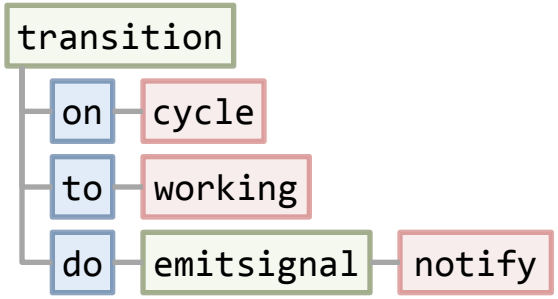
Кодогенерация

- Построение Verilog-модуля, реализующего конечный автомат

Поток лексем

```
transition on ( cycle
) to working do {
emitsignal notify ; }
```

Дерево разбора (AST)



Verilog-модуль*

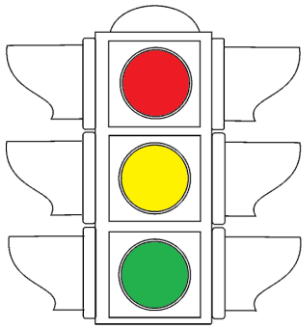
```
if (!reset &&
    !init &&
     cycle)
begin
      state   <= working;
      notify <= 1;
end
```

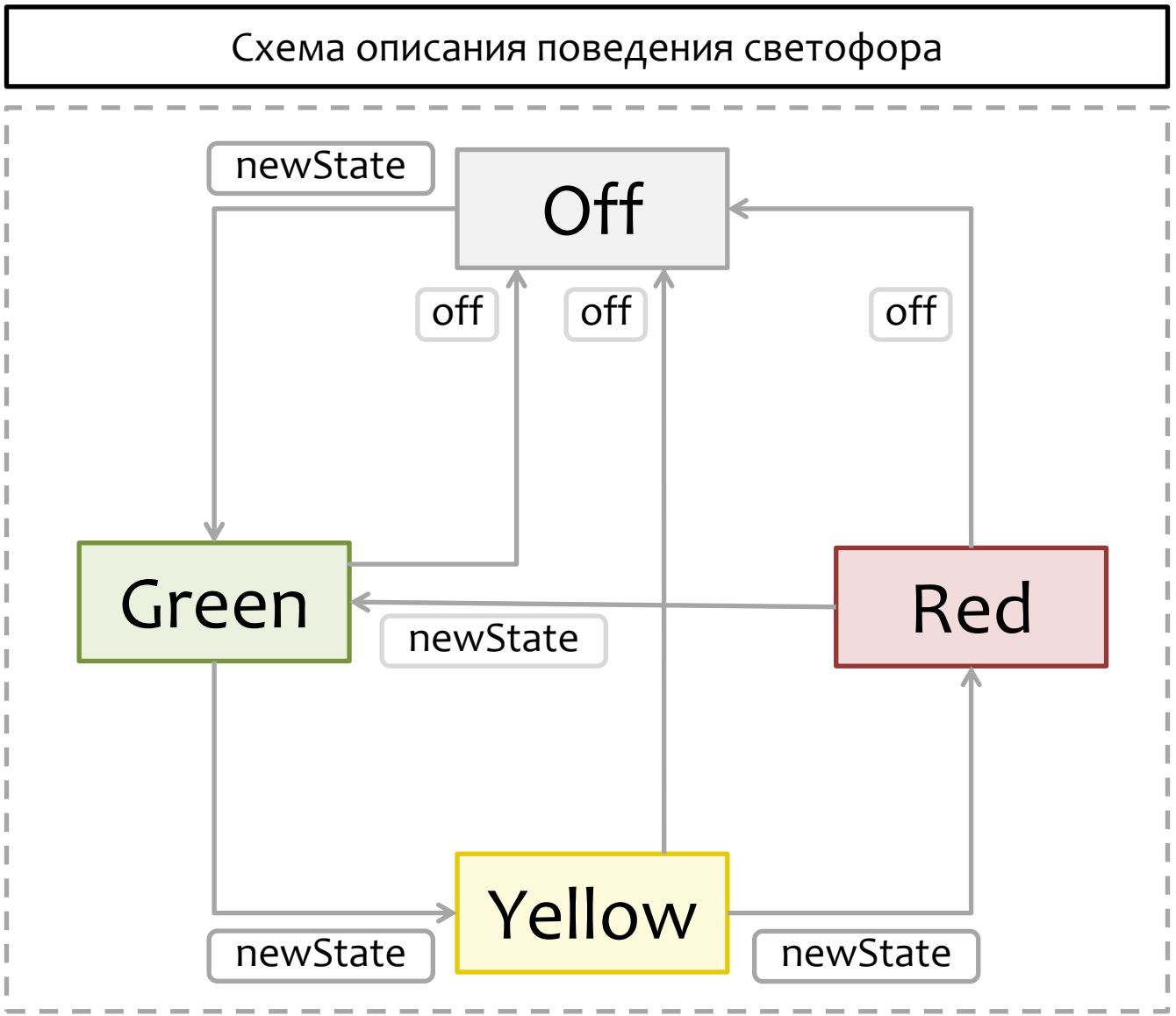
* - пример упрощён

Генерация Verilog-кода

- Результат работы транслятора — Verilog-модуль
- Сигналы представлены абстрактными входами и выходами

Пример трансляции КА





Пример трансляции КА

Описание КА на SML*

```
states  {off, green, yellow, red}
inputs  {newState, reset}
outputs {greenLight, yellowLight, redLight}

switch
{
          state off
          {
                    stopsignals;
                    transition on (newState) to green do {}
          }

          state green
          {
                    stopsignals;
                    emitsignal greenLight;
                    transition on (newState) to yellow do {}
                    transition on (reset)    to off    do {}
          }

          …
}
```

* - пример сокращен, исходный объем ~50 LOC

Пример трансляции КА

Результат трансляции листинга на SML в Verilog*

```
module state_machine (input wire clk,
                      input wire inputs  [:0],
                      output reg outputs [2:0]);
… // Declaration of need variables and datas, generation of posedge
and negedge datas
always @(posedge clk)
begin
        … // Assignement of inputs
        case (state)
        0:
             begin
                    if ((inputs [2] && inputs [0]) &&
                        (posedge_inputs [0] || negedge_inputs [1] ||
                        posedge_inputs [2]))
                    begin
                            last_state <= state;
                            state = 1;
                    end
             end
        endcase
        …
end
```

* - пример сильно сокращен, исходный объем ~200 LOC

Пример трансляции КА

Листинг на Verilog

```
module state_machine (input wire clk,
                      input wire inputs  [:0],
                      output reg outputs [2:0]);
…
always @(posedge clk)
…
always @(state)
…
```

Работа тестовой платы

Результаты работы

- Разработан транслятор, способный по описанию КА автоматически генерировать код абстрактного управляющего модуля на языке Verilog
- Реализован собственный язык описания конечных автоматов
- Испытана работоспособность генерируемого кода на отладочной плате с использованием FPGA
- Исходный код проекта: https://github.com/Nexxof/SMT, размер – 2000 LOC

Отладочная плата предоставлена компанией Intel

Работа выполнена в рамках сотрудничества с компанией Intel

Спасибо за внимание!

Использованная литература

- Н. И. Поликарпова, А. А. Шалыто. Автоматное программирование.
- А.А.Шалыто, Н.И.Туккель, SWITCH-технология — автоматный подход к созданию программного обеспечения «реактивных» систем, журнал "Программирование", №5, 2001 г.
- С. Э. Вельдер, М. А. Лукин, А. А. Шалыто, Б. Р. Яминов, “Верификация автоматных программ”, “Наука”, 2001 г.
- Ю. Ю. Янкин, А. А. Шалыто, “Автоматное программирование ПЛИС в задачах управления электроприводом”, журнал “Информационно-управляющие системы”, №1, 2011 г.
- А.К. Поляков. Языки VHDL и Verilog в проектировании цифровой аппаратуры.
- И.А. Лагунов, “Разработка текстового языка автоматного программирования ” (http://is.ifmo.ru/diploma-theses/fsml/)
- Л.В. Столяров, публикация работы в журнале «Прикладная дискретная математика (Приложение)», Томск: ТомГУ, 2009, №1, С. 81-83.
- П. Хоровиц. У. Хилл. Искусство схемотехники.
- Материалы с сайта *stackoverflow.com.*