Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных средств

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА к курсовому проекту на тему:

**VHDL - МОДЕЛЬ МИКРОПРОГРАММНОГО АВТОМАТА УМНОЖЕНИЯ ЧИСЕЛ**

БГУИР КП 1-40 02 02 011 ПЗ

Студент А.В. Миронь

Руководитель П.Н. Бибило

Минск 2019

# СОДЕРЖАНИЕ

Оглавление

[СОДЕРЖАНИЕ 2](#_Toc26899951)

[ВВЕДЕНИЕ 3](#_Toc26899952)

[1 РАЗРАБОТКА АЛГОРИТМА 5](#_Toc26899953)

[1.1 Постановка задачи 5](#_Toc26899954)

[1.2 Описание конечных автоматов 5](#_Toc26899955)

[1.3 Алгоритм 6](#_Toc26899956)

[2 РАЗРАБОТКА VHDL ОПИСАНИЙ 9](#_Toc26899957)

[2.1 Пакеты 9](#_Toc26899958)

[2.2 Entity «робота» и его архитектура 10](#_Toc26899959)

[2.3 Тестирующая программа 13](#_Toc26899960)

[3 МОДЕЛИРОВАНИЕ И ОТЛАДКА VHDL ПРОГРАММ 15](#_Toc26899961)

[3.1 Моделирование верхнего модуля 15](#_Toc26899962)

[3.2 Покрытие VHDL-кода 15](#_Toc26899963)

[4 СИНТЕЗ УСТРОЙСТВА НА ЭЛЕМЕНТНОЙ БАЗЕ ПЛИС 17](#_Toc26899964)

[4.1 Описание целевой ПЛИС 17](#_Toc26899965)

[4.2 Оценка аппаратной сложности 19](#_Toc26899966)

[5 ЗАКЛЮЧЕНИЕ 20](#_Toc26899967)

[6 Список использованной литературы 20](#_Toc26899968)

[7 Приложение А 21](#_Toc26899969)

[8 Приложение Б 22](#_Toc26899970)

# ВВЕДЕНИЕ

Язык VHDL (Very high speed integrated circuits Hardware Description Language) является фактически международным стандартом в области автоматизации проектирования цифровых систем, это входной язык многих современных систем автоматизированного проектирования (САПР) как заказных, так и программируемых логических интегральных схем (ПЛИС) - Programmable Logic Devices (PLD) - и программируемых пользователями вентильных матриц - Field-Programmable Gate Arrays (FPGA). VHDL предназначен, в первую очередь, для спецификации - точного описания проектируемых систем и их моделирования на начальных этапах проектирования - алгоритмическом и логическом. С помощью VHDL можно моделировать электронные схемы с учетом реальных временных задержек.

В последнее время весьма успешно разрабатываются и системы синтеза схем по спецификациям на этом языке. Например, используя САПР Xilinx Foundation Series 2.1i, можно провести моделирование исходного описания схемы на языке VHDL, а затем синтезировать схему и получить файл настройки (конфигурации) микросхемы типа FPGA фирмы Xilinx. Использование САПР MAX+PLUSII позволяет решать аналогичные задачи для программируемых микросхем, выпускаемых фирмой Altera. Для заказных СБИС могут быть использованы САПР фирмы Mentor Graphics: система моделирования ModelSim позволяет провести моделирование описаний, представленных на языке VHDL, система синтеза Leonardo Spectrum Level 3 позволяет получать по описаниям на языке VHDL схемы в заданных базисах логических элементов. Такие крупнейшие фирмы – производители программного обеспечения САПР в области микроэлектроники, как Cadence, Synopsys и многие другие используют язык VHDL в качестве языка исходного описания проектов.

VHDL - это мощный язык, он позволяет описывать поведение, т.е. алгоритмы функционирования цифровых систем, а также проводить иерархическое функционально-структурное описание систем, имеет средства для описания параллельных асинхронных процессов, регулярных (систолических) структур и в то же время имеет все признаки языка программирования высокого уровня - позволяет создавать свои типы данных, имеет широкий набор арифметических и логических операций и т.д.

Язык VHDL был разработан в США по инициативе министерства обороны этой страны. В 1987 г. VHDL был принят в качестве стандарта ANSI/IEEE Std 1076-1987. Данный стандарт часто называют VHDL’87. За3 тем язык был усовершенствован, новый стандарт ANSI/IEEE Std 1076- 1993 (стандарт VHDL’93) появился в 1993 г. Книга целиком посвящена новому стандарту языка и его отличиям от стандарта VHDL’87. В 1999 г. утвержден стандарт Std 1076.1-1999 (или более распространенное наименование VHDL-AMS), который включает расширения, дающие возможность описания моделей аналоговых и смешанных (цифро-аналоговых) схем.

Язык VHDL развивается, ему посвящаются международные конференции, выходят научные журналы, в которых изучаются проблемы использования VHDL. Он стал языком разработки международных проектов, в том числе осуществляемых с помощью всемирной компьютерной сети Internet. Знакомство с этим языком необходимо для эффективной работы по созданию самой разнообразной электронной аппаратуры на современной элементной базе сверхбольших интегральных схем.

# РАЗРАБОТКА АЛГОРИТМА

## Постановка задачи

Задано два двоичных числа A и B – векторы размерностью n = 4, …, 10 элементов. Имеется три регистра: RG1 – регистр, в котором постоянно хранится значение множимого A; RG2 – регистр, в котором хранится множитель B; RG3 – регистр, в котором размещаются частичные суммы частичных произведений и произведение. Автомат должен выполнить умножение эти двух двоичных чисел, начиная с младших разрядов, тем самым формируя частичные произведения.

## Описание конечных автоматов

В процессе проектирования цифровых систем остро стоят проблемы написания VHDL-кода без составления предварительного алгоритма, описывающего поведение системы. Для описания системы «робот-минёр» используем алгоритм, содержащий два конечных автомата.

Первый автомат (головной) содержит четыре состояния:

1. Start – состояние, из которого «робот» начинает работу.
2. Safe – состояние проверки ячеек и поиска мины.
3. Mine – состояние, в котором «робот» возвращает мину в начальную позицию (левый верхний угол поля по условию).
4. Ret – состояние, в котором «робот» возвращается в последнюю проверенную ячейку после возвращения мины.

Второй автомат (подчинённый) содержит три состояния, которые указывают направление движения «робота» в состоянии «safe»:

1. R – right (право).
2. L – left (лево).
3. D – down (низ).

Движение вверх не предусмотрено алгоритмом для подчинённого автомата, так как «робот начинает движение из верхнего левого угла (координаты 1;1) и движется только вниз, вправо либо влево.

Графы переходов головного и подчинённого автоматов представлены в приложении А.

## Алгоритм

Цифровая система имеет управляющий сигнал reset и тактовый сигнал clk. При подаче на вход reset ‘0’, состояние головной автомат получает значение «start», внутреннее состояние получает значение «R».

При подаче на reset ‘1’, «робот» начинает работу по переднему фронту синхросигнала (clk’event and clk=’1’).

Состояние start:

1. загрузка входного поля в поле типа inout.
2. Переход из состояния start в состояние safe.

Состояние safe:

1. Проверка ячейки, в которой находится робот (координаты ячейки находятся в x2, y2) на наличие мины.
2. Если есть мина:

переход в состояние «Mine», обнуление ячейки с координатами x2, y2.

1. Если мины нет:

Движение в соответствии со значением состояния подчинённого автомата.

Состояние Mine:

1. Запоминание координаты мины (x2 заносится в I, y2 – в J).
2. Возвращение в начальную точку: «робот» поднимается максимально вверх поля, а после двигается влево, пока не достигнет координаты (1,1).
3. При достижении начала координат состояние меняется на Ret.

Состояние Ret:

1. Возвращение в координату (I, J) – движение вправо до координаты J, после движение вниз до координаты I.
2. Переход в состояние Safe.

Переходы между состояниями подчинённого автомата осуществляются по следующему принципу:

Состояние R:

1. Если координата y2 = Size (размерность матрицы) переход в состояние D.
2. Иначе к y2 прибавляется 1.

Состояние D:

1. К x2 прибавляется 1.
2. Если y2 = Size, переход в состояние L, иначе переход в состояние R.

Состояние L:

1. Если y2 = 1 переход в состояние D.
2. Иначе от y2 отнимается 1.

Примеры переходов приведены на графах.

Внешнее состояние Ret: алгоритм, не включающий в себя данное состояние также работает правильно однако требует куда большего времени для работы и больше количества тактов. Для сравнения, матрица 5 х 5 без данного состояния «очищается» от мин за 600 тактов, в то время как с состоянием Ret – за 150.

Рассмотрим на примере (рисунок 1.1):

Поиск мины осуществляется в состоянии Safe: направление движения задаётся состояниями подчинённого автомата (R, L, D).  
 Когда мина найдена координаты записываются в переменные I и J, состояние головного автомата S <= Mine и в этом состоянии робот возвращается в начальную точку: координата робота x2 уменьшается до тех пор, пока не станет равной 1, затем начинает уменьшатся координата робота y2 также до значения 1. После возвращения S <= ret.

В состоянии ret робот возвращается в точку (I; J) (точка (4;3) согласно рисунку): к координате робота прибавляется 1 в каждом такте до тех пор, пока y2 не станет равной J, затем происходит аналогичное действие с x2. После достижение данной точки вычисляется значение l0 равное остатку от деления I на 2 для определения направления дальнейшего движения робота: при чётных I (2, 4, и т.д.) робот движется влево (состояние подчинённого автомата K<=L), при нечётных – вправо (состояние K<=R). Далее состояние S <=safe: продолжается поиск мин.

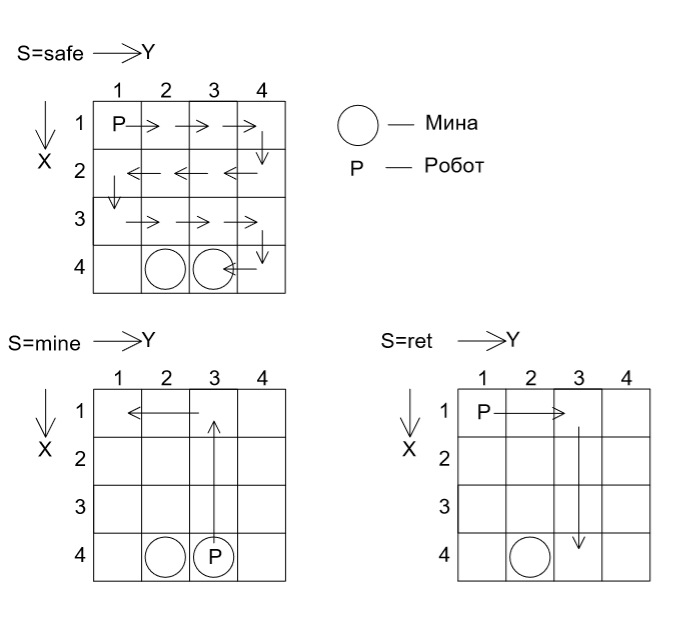


Рисунок 1.1 – Пример поиска мины.

# РАЗРАБОТКА VHDL ОПИСАНИЙ

Проект состоит из одного модуля: на входы модуля подаётся поле, а также начальные координаты, управляющие сигналы clk и reset, на выходе получаем выходное поле, а также выходные координаты «робота».

Также проект включает два собственных пакета для состояний автомата и задания поля в виде матрицы.

## Пакеты

В пакете представлены типы для матрицы (поля) и состояний автомата.

Пакет на языке VHDL представлена в листинге 2.1

Листинг 2.1 – Пакет для «робота».

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**package Const is**

constant Size: natural: =5;

type st is array (natural range 1 to Size) of integer range -1 to 1;

type Matrix is array (natural range 1 to Size) of st;

**end Const;**

**package Aut is**

type state is (Start, Mine, Safe, Ret);

type movement is (R, D, L);

**end aut;**

## Entity «робота» и его архитектура

При росте размерности входной битовой матрице очень быстро растет количество используемых пинов ввода/вывода. При размере входной матрицы 6 на 6 требуется 146 пинов ввода/вывода, что превышает количество доступных пинов в Spartan-3 XC3S1000.

VHDL код для «робота»:

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**library work;**

**use work.const.all;**

**use work.aut.all;**

**entity** Robot **is**

**port** (M1 : **in** Matrix;

x1, y1 : **in** **integer** **range** 1 to size;

clk, reset: **in** **std\_logic**;

x2, y2 : **inout** **integer** **range** 1 to size;

M2 : **inout** matrix);

**end** Robot;

**architecture** void **of** robot **is**

**signal** s: state;

**signal** K: movement;

**signal** i, j: **integer** **range** 1 **to** size;

**signal** l0: **integer range** 0 **to** 1;

**signal** cc: **integer range** 0 **to** 5000;

**begin**

**process (**clk, reset)

**begin**

**if**(reset='0') **then**

K<=R;

S<=start;

cc<=0;

**elsif** (clk'event **and** clk='1') **then**

cc<=cc+1;

**case** S **is**

**when** start=>

M2<=M1;

x2<=x1;

y2<=y1;

M2(x2)(y2)<=0;

s<=safe;

**when** safe =>

**if** (M2(x2)(y2)=0) **then**

**case** K **is**

**when** R =>

**if** (y2=size) **then**

K<=D;

**else**

y2<=y2+1;

**end** **if**;

**when** D =>

x2<=x2+1;

**if** (y2=size) **then**

K<=L;

**else**

K<=R;

**end if**;

**when** L =>

**if** (y2=1) **then**

K<=D;

**else**

y2<=y2-1;

**end if**;

**end case**;

M2(x2)(y2)<=0;

S<=safe;

**else**

M2(x2)(y2)<=0;

S<=mine;

i<=x2;

j<=y2;

**end if**;

**when** mine =>

**if** (x2=1 **and** y2=1) **then**

K<=R;

s<=ret;

**else**

**if** (x2/=1) **then**

x2<=x2-1;

**else**

y2<=y2-1;

**end if**;

**end if**;

**when** ret =>

l0<=i **rem** 2;

**if** (x2=i **and** y2=j) **then**

**if** (l0=1) **then**

K<=R;

**elsif** (l0=0) **then**

K<=L;

**end if**;

s<=safe;

**else**

**if** (y2/=j) **then**

y2<=y2+1;

**elsif** (x2/=i) **then**

x2<=x2+1;

**end if**;

**end if**;

**end case**;

**end if**;

**end process**;

**end void**;

## Тестирующая программа

Для тестирования была написана программа на VHDL, в которой задаётся матрица, сигнал сброса, начальные координаты робота и синхросигнал.

Листинг 2.2– Тестирующая программа

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**library work;**

**use work.const.all;**

**use work.aut.all;**

**entity test is**

**end test;**

**architecture** testa **of** test **is**

**component** robot

**port** (M1 : **in** Matrix;

x1, y1 : **in** integer;

clk, reset: **in** std\_logic;

x2, y2 : **inout** integer;

M2 : **inout** matrix);

**end** **component**;

**constant** clk\_period: **time**: =40 **ns**;

**signal** M1, M2: matrix;

**signal** reset: **std**\_**logic**;

**signal** x1, x2, y1, y2: **integer** **range** 1 **to** size;

**signal** clk: **std**\_**logic**: = '0';

**begin**

p1: **robot** **port map** (M1, x2, y1, clk, reset, x2, y2, M2);

M1(1) <= (0, 0, 0, 1, 0);

M1(2) <= (0, 0, 1, 0, 1);

M1(3) <= (1, 1, 0, 1, 1);

M1(4) <= (1, 1, 0, 0, 1);

M1(5) <= (0, 0, 1, 1, 0);

reset<='0', '1' after 10 ns;

p2: **process**

**begin**

loop1: **loop**

clk <= **transport** '1'; **wait** **for** clk\_period/2;

clk <= **transport** '0'; **wait** **for** clk\_period/2;

**end loop;**

**wait;**

**end process;**

**end** testa**;**

# МОДЕЛИРОВАНИЕ И ОТЛАДКА VHDL ПРОГРАММ

Моделирование цифровых систем является важным шагом в маршруте их разработки. Возрастание сложности проектируемых устройств заставляет разработчиков тратить все больше времени на их моделирование. Целями моделирования могут быть как исследование алгоритмов работы проектируемого устройства, так и верификация характеристик, получаемых при его аппаратной реализации. В первом случае производится моделирование на верхних уровнях абстрагирования (т.е. преимущественно на поведенческом, и, возможно, RTL), а моделирование на физическом уровне призвано проверить возможность работы созданного устройства в заданных условиях эксплуатации (т.е. проверяется возможность работы на заданной тактовой частоте, с требуемыми длительностями сигналов, в заданном температурном диапазоне и т.д.).

При моделировании используется подход, основанный на «испытательном стенде» (testbench). Моделируемое устройство (в англоязычной литературе UUT, UnitUnderTest) представляется своим синтезируемым кодом, а для проверки его поведения в различных условиях создаются описания тестовых воздействий («моделируемый код»).

Задание тестовых входных воздействий выполнено на VHDL с помощью модуля, приведенного в листинге 2.2.

## Моделирование верхнего модуля

Моделирование и верификация проводилась в системе ModelSim с помощью тестирующей программы, приведённой в листинге 2.2. В результате работы не было выведено сообщений об ошибках в тестирующей модели, следовательно, VHDL-модель устройства можно считать эквивалентной эталонной модели и реализованной корректно.

## Покрытие VHDL-кода

Покрытие характеризует текст программы с точки зрения прохождения VHDL кода при моделировании. Применение процедур покрытия кода не предназначена для проверки правильности ожидаемых и полученных реакций VHDL-модели цифровой системы на наборах значений входных сигналов. Если промоделированная строка была выполнена хотя бы один раз, то она считается покрытой, иначе строка непокрыта. Результат покрытия VHDL-кода в среде ModelSim моделируемого устройства продемонстрирован на рисунке 3.1.

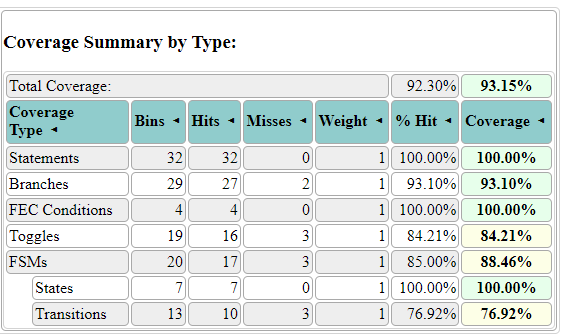


Рисунок 3.1 – Отчет о покрытии

# СИНТЕЗ УСТРОЙСТВА НА ЭЛЕМЕНТНОЙ БАЗЕ ПЛИС

Этап синтеза представляет собой процесс трансформации исходного HDL-описания проектируемого устройства в список цепей, выполненный на низком логическом уровне. Элементы низкоуровневого описания, формируемого в процессе синтеза, должны соответствовать архитектуре семейства ПЛИС, выбранного для реализации проекта.

В синтезе выделяют следующие этапы:

* Высокоуровневый синтез
* Технологически независимые оптимизации
* Технологическое отображение
* Увеличение быстродействия

На этапе высокоуровневого синтеза осуществляется замена HDL конструкций соответствующими подсхемами. Во время этапа технологически независимых оптимизаций осуществляется оптимизация комбинационной логики. На третьем этапе осуществляется покрытие минимизированных логических выражений описаниями логических элементов, алгоритмическое описание триггеров заменяется описанием триггеров в целевой библиотеке. На заключающем этапе полученное структурное описание модифицируется, выделяются критические пути, происходит повторный синтез и быстродействие схемы увеличивается.

## Описание целевой ПЛИС

На рисунке 4.1 показаны основные компоненты и интерфейсы, размещённые на отладочной плате:

* генератор синхросигнала на 50 МГц;
* вход внешней синхронизации;
* внешняя память инициализации ПЛИС (Flash PROM XCF04S);
* внешнее ОЗУ (512К x 16 бит);
* три 40-выводных разъёма, на которые выведены пользовательские выводы ПЛИС;
* полоска из 8 светодиодов;
* четыре семисегментных индикатора;
* 4 кнопки;
* 8 переключателей;
* порты PS/2, VGA, RS232 (COM-порт).

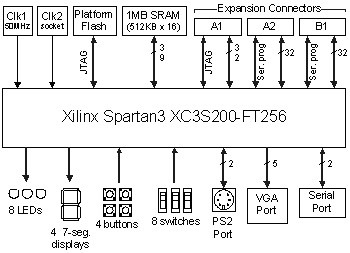


Рисунок 4.1 – Spartan-3 starter kit

## Оценка аппаратной сложности

Оценка аппаратной сложности производилась на квадратных матрицах для устройства с параллельным вводом/выводом. Графики зависимости используемой аппаратуры продемонстрированы на рисунке 4.3 для устройства с параллельной загрузкой выгрузкой данных.

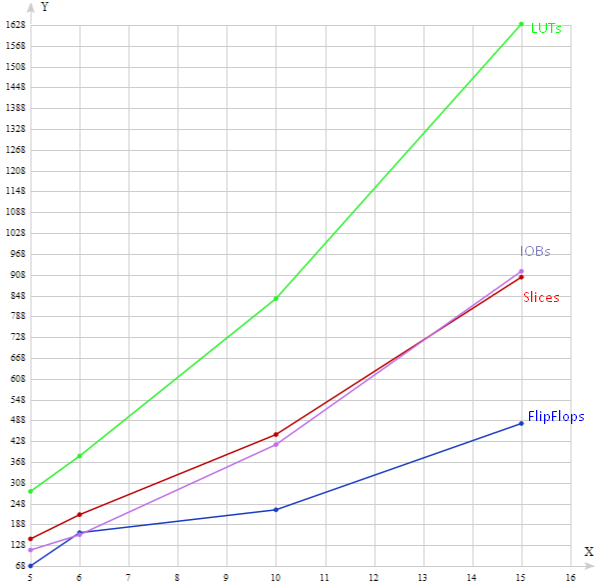


Рисунок 4.3 – Оценка аппаратной сложности (параллельный ввод)

# ЗАКЛЮЧЕНИЕ

В ПЛИС Spartan-3 XC3S1000 помещается проектируемое устройство, обрабатывающее минное поле размером до 5 на 5. Выполнение курсового проекта максимально приближено к промышленной разработке цифровых устройств что позволило более глубоко изучить и систематизировать знания в области проектирования цифровых средств на языках описания аппаратуры.

При использовании других ПЛИС достигается большее значение размерности минного поля за счёт большего количество входных-выходных портов (IOBs):

Virtex6 XC6VLX75T – 7 на 7;

Kintex7 XC7K160T – 9 на 9;

В рамках курсового проекта не была разработана последовательная загрузка минного поля, однако она могла бы расширить размерность поля для использованной ПЛИС.

Схемная реализация поставленной задачи на ПЛИС имеет большой объём, что показывает важность использования языков описания аппаратуры таких, как VHDL, чтобы значительно ускорить процесс разработки, а также сделать его более похожим на написание прикладных программ на языке высокого уровня.

# Список использованной литературы

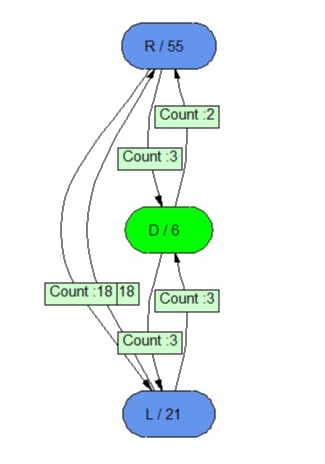
[1] Бобрешов А.М. Проектирование цифровых устройств с помощью языка описания аппаратуры VHDL: учебное пособие/Бобрешов А.М., Дыбой А.В. - Воронеж: ИПЦ ВГУ, 2007. - 51 с.

[2] Бибило П.Н. Основы языка VHDL: учебное пособие/Бибило П.Н. – Москва, «Солон-Р», 2010. – 200 с.

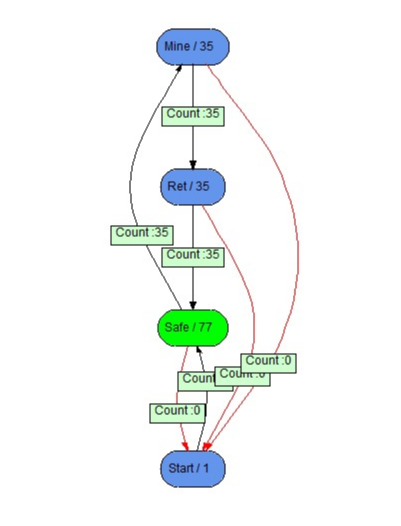
[3] Бибило П.Н. Моделирование и верификация цифровых систем на языке VHDL: учебное пособие/Бибило П.Н., Авдеев Н.А. М.: ЛЕНАНД, 2017. – 344 с.

# Приложение А

Граф переходов внутреннего автомата.



Граф переходов внешнего автомата.



# Приложение Б

Временная диаграмма для матрицы размерности 6 на 6.  
Диаграмма представлена для времени моделирования 0 to 1600 ns и для 18400 to 20100 ns.

Сигнал M1 – входная матрица (минное поле), M2 – выходная матрица (данная матрица изменяется в процессе разминирования «роботом»).

Сигнал S – состояния головного автомата, K – подчинённого.

Сигнал сс – счётчик тактов (был нужен для сравнения алгоритмов с состоянием ret без него).

Сигналы i, j, l0 – сигналы для запоминания месторасположения последней разминированной роботом мины. Описано в п.1.2 Алгоритм.

Сигнал XY – координаты робота в данном такте.

На отрезке 18400 – 20100 ns показан конец работы и полностью разминированная матрица M2.