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. Необходимо построить микропрограммный автомат для умножения этих двоичных чисел. Будет использоваться три регистра и сумматор: 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. Умножение будет производиться, начиная с младших разрядов, тем самым формируя частичные произведения.

## Краткие теоретические сведения

## Алгоритм

Для обоснования принципа построения микропрограммного автомата необходимо воспользоваться правилом умножения двоичных чисел, при котором частичные произведения формируются, начиная с младших разрядов множителя B (рисунок 1.1).

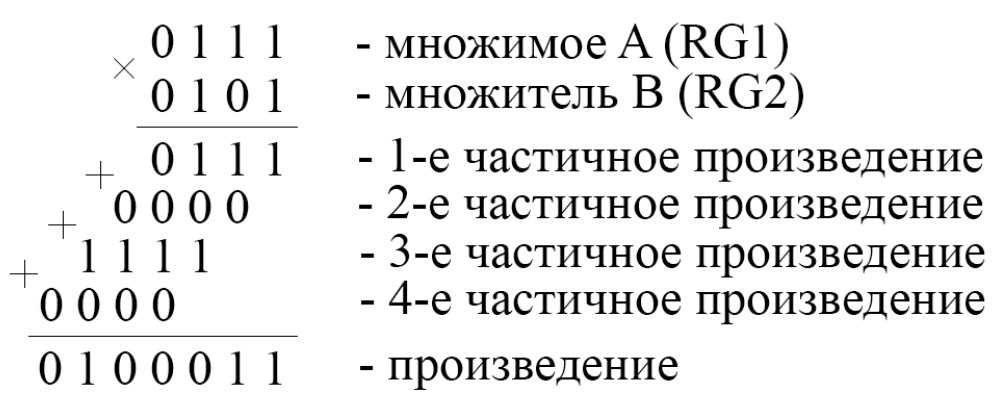


Рисунок 1.1 – Правило умножения двоичных чисел

Суммирование частичных произведений должно осуществляться последовательно во времени с помощью сумматоров, предназначенных для сложения двух операндов. Особенность умножения двоичных чисел состоит в том, что частичные произведения могут принимать лишь два значения: значение множимого A либо значение нуля. Значение частичного произведения определяется значением текущего разряда множителя B. Если частичное произведение равно нулю, то микрооперацию сложения можно не выполнять.

Множимое A используется как частичное произведение, и оно будет храниться в регистре RG1. Для определения значения текущего разряда множителя B необходимо иметь сдвигающий регистр RG2. В исходном состоянии регистр RG2 загружен множителем B. Чтобы выявить значение следующего разряда множителя В, после каждой микрооперации сложения частичного произведения необходимо производить сдвиг содержимого RG2 в сторону самого младшего разряда. Для хранения частичных сумм частичных произведений необходимо располагать третьим регистром RG3. В исходном состоянии RG3 должен быть загружен нулями. В процессе умножения осуществляется сложение содержимого регистра RG3 с частичным произведением А. Частичная сумма помещается в RG3, после чего выполняется сдвиг в сторону младших разрядов. При этом в два раза увеличивается вес каждого очередного разряда множителя В.

На рисунке 1.2 показан процесс умножения с использованием трех регистров и сумматора. Множимое А = 0111 постоянно находится в регистре RG1. В исходном состоянии в регистр RG3 помещен нуль 0000, а в регистр RG2 – множитель B = 0101. Нуль в старшем разряде операндов A и B свидетельствует о том, что перемножаются положительные числа. В процессе умножения в регистре RG3 размещаются частичные суммы частичных произведений и произведение.



Рисунок 1.2 – Алгоритм умножения

**Этапы умножения:**

1. Анализируется младший разряд регистра RG2:

- Если младший разряд RG2 равен 1, то выполняется микрооперация сложения содержимого регистров RG3 и RG1, и результат помещается в RG3.

- Если младший разряд RG2 равен 0, то переходим к этапу 2.

2. Выполняется микрооперация сдвига вправо на один разряд содержимого составного регистра (RG3 и RG2). Этот процесс носит циклический характер. Число циклов n равно числу разрядов множителя A (в данном случае n = 4).

# РАЗРАБОТКА VHDL-МОДЕЛИ

## Пакеты

В пакете представлены матрица и число разрядов двоичного числа.

Листинг 2.1 – Пакет «project\_package»

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

PACKAGE project\_package IS

CONSTANT N : NATURAL := 4;

TYPE int\_array IS ARRAY(NATURAL RANGE 0 TO N - 1) OF INTEGER RANGE 0 TO 3;

END project\_package;

## Entity «multiplier» и ее архитектура

Листинг 2.2 – Код умножителя

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

LIBRARY work;

USE work.project\_package.ALL;

ENTITY multiplier IS

PORT (

rg1, rg2 : IN std\_logic\_vector (0 TO N - 1);

result : OUT std\_logic\_vector (0 TO (N \* 2) - 1));

END multiplier;

ARCHITECTURE mult\_function\_beh OF multiplier IS

FUNCTION fill\_array\_with\_zeros(arr : int\_array) RETURN int\_array IS

VARIABLE arr\_with\_zeros : int\_array;

BEGIN

FOR i IN 0 TO arr'length - 1 LOOP

arr\_with\_zeros(i) := 0;

END LOOP;

RETURN arr\_with\_zeros;

END FUNCTION;

FUNCTION std\_logic\_to\_integer(arr : std\_logic\_vector(0 TO N - 1)) RETURN int\_array IS

VARIABLE arr\_to\_integer : int\_array;

BEGIN

FOR i IN 0 TO arr'length - 1 LOOP

IF (arr(i) = '1') THEN

arr\_to\_integer(i) := 1;

ELSE

arr\_to\_integer(i) := 0;

END IF;

END LOOP;

RETURN arr\_to\_integer;

END FUNCTION;

FUNCTION integer\_to\_std\_logic(arr1, arr2 : int\_array) RETURN std\_logic\_vector IS

VARIABLE arr\_to\_std\_logic : std\_logic\_vector(0 TO (N \* 2) - 1);

BEGIN

FOR i IN 0 TO arr1'length - 1 LOOP

IF (arr1(i) = 1) THEN

arr\_to\_std\_logic(i) := '1';

ELSE

arr\_to\_std\_logic(i) := '0';

END IF;

END LOOP;

FOR i IN 0 TO arr2'length - 1 LOOP

IF (arr2(i) = 1) THEN

arr\_to\_std\_logic(i + N) := '1';

ELSE

arr\_to\_std\_logic(i + N) := '0';

END IF;

END LOOP;

RETURN arr\_to\_std\_logic;

END FUNCTION;

FUNCTION multiply(rg1, rg2 : std\_logic\_vector(0 TO N - 1)) RETURN std\_logic\_vector IS

VARIABLE rg1\_copy : int\_array;

VARIABLE rg2\_copy : int\_array;

VARIABLE rg3 : int\_array;

VARIABLE result : std\_logic\_vector (0 TO (N \* 2) - 1);

VARIABLE counter : INTEGER := N;

VARIABLE buff : INTEGER := 0;

VARIABLE rg3\_last\_element : INTEGER := 0;

BEGIN

rg1\_copy := std\_logic\_to\_integer(rg1);

rg2\_copy := std\_logic\_to\_integer(rg2);

rg3 := fill\_array\_with\_zeros(rg3);

WHILE counter > 0 LOOP

IF (rg2\_copy(N - 1) = 1) THEN

FOR i IN rg3'length - 1 DOWNTO 0 LOOP

rg3(i) := rg3(i) + rg1\_copy(i);

IF (buff /= 0) THEN

rg3(i) := rg3(i) + buff;

buff := buff - 1;

END IF;

IF (rg3(i) > 1) THEN

buff := buff + 1;

rg3(i) := rg3(i) - (buff + 1);

END IF;

END LOOP;

END IF;

rg3\_last\_element := rg3(N - 1);

FOR i IN rg3'length - 1 DOWNTO 1 LOOP

rg3(i) := rg3(i - 1);

END LOOP;

rg3(0) := 0;

FOR i IN rg2\_copy'length - 1 DOWNTO 1 LOOP

rg2\_copy(i) := rg2\_copy(i - 1);

END LOOP;

rg2\_copy(0) := rg3\_last\_element;

counter := counter - 1;

END LOOP;

result := integer\_to\_std\_logic(rg3, rg2\_copy);

RETURN result;

END multiply;

BEGIN

result <= multiply(rg1, rg2);

END mult\_function\_beh;

## Тестирующая программа

Для тестирования была написана программа на VHDL, в которой выполняется считывание из файла значений двух чисел A и B.

Листинг 2.3 – Тестирующая программа

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_textio.ALL;

USE ieee.numeric\_std.ALL;

USE std.textio.ALL;

LIBRARY work;

USE work.project\_package.ALL;

ENTITY test\_mult IS

END test\_mult;

ARCHITECTURE test\_beh OF test\_mult IS

COMPONENT multiplier IS

PORT (

rg1, rg2 : IN std\_logic\_vector (0 TO N - 1);

result : OUT std\_logic\_vector (0 TO (N \* 2) - 1));

END COMPONENT;

SIGNAL rg1, rg2 : std\_logic\_vector (0 TO N - 1);

SIGNAL result : std\_logic\_vector (0 TO (N \* 2) - 1);

CONSTANT f\_name : STRING := "input\_data.txt";

FILE f\_source : text;

BEGIN

mult\_func\_call : multiplier PORT MAP(rg1 => rg1, rg2 => rg2, result => result);

PROCESS

VARIABLE f\_line : line;

VARIABLE rg1\_line : std\_logic\_vector(0 TO N - 1);

VARIABLE rg2\_line : std\_logic\_vector(0 TO N - 1);

VARIABLE space\_char : CHARACTER;

BEGIN

file\_open(f\_source, f\_name, read\_mode);

WHILE (NOT endfile(f\_source)) LOOP

readline(f\_source, f\_line);

read(f\_line, rg1\_line);

read(f\_line, space\_char);

read(f\_line, rg2\_line);

rg1 <= rg1\_line;

rg2 <= rg2\_line;

WAIT FOR 50 ns;

END LOOP;

file\_close(f\_source);

END PROCESS;

END test\_beh;

# МОДЕЛИРОВАНИЕ И ОТЛАДКА VHDL ПРОГРАММ

## Моделирование верхнего модуля

Моделирование и верификация проводилась в системе ModelSim с помощью тестирующей программы, приведённой в листинге 2.3. 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.

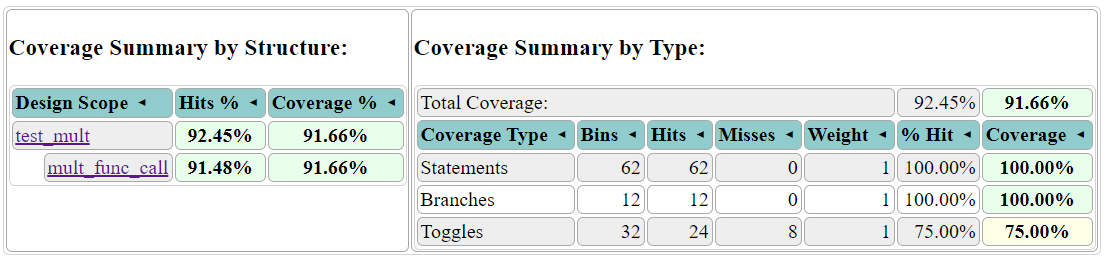


Рисунок 3.1 – Отчет о покрытии тестами в HTML формате

# 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).

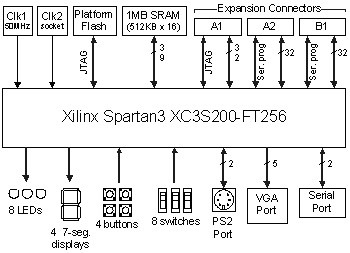


Рисунок 4.1 – Spartan-3 starter kit

## 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.

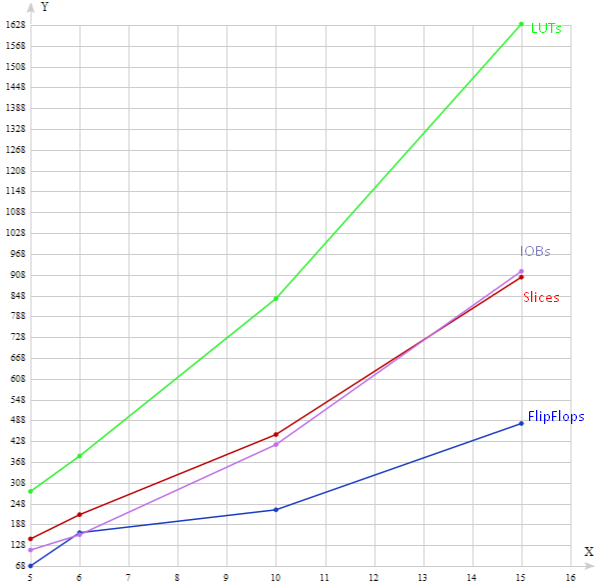


Рисунок 4.3 – Оценка аппаратной сложности (параллельный ввод)

# 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

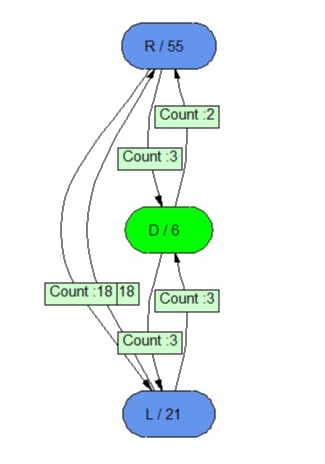
[1] Бобрешов А.М. Проектирование цифровых устройств с помощью языка описания аппаратуры VHDL: учебное пособие/Бобрешов А.М., Дыбой А.В. - Воронеж: ИПЦ ВГУ, 2007. - 51 с.

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

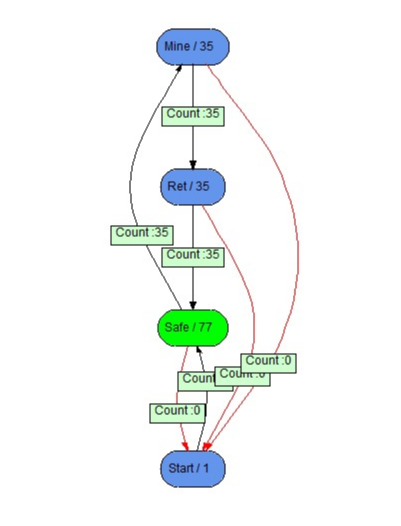
[3] Бибило П.Н. Моделирование и верификация цифровых систем на языке VHDL: учебное пособие/Бибило П.Н., Авдеев Н.А. М.: ЛЕНАНД, 2017. – 344 с.

# 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.