МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Руководитель:	К защите допустить:
Доцент кафедры ИМС к.п.н. Б.Е. Механцев	Доцент кафедры ИБТКС к.т.н., А.П. Плёнкин
_____ *(подпись)*	_____ *(подпись)*
«___» _____ 20__ г.	«___» _____ 20__ г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ТВОРЧЕСКОМУ ПРОЕКТУ

ПО ДИСЦИПЛИНЕ «ВвИД»

на тему: Реализация скоростных криптографических систем

Команда: CRAFSED

Выполнили:

Нестеренко Пётр Алекссевич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Кочубей Даниил Сергеевич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Жалнин Дмитрий Игоревич, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Пучкова Анастасия Денисовна, КТбо2-8
(подпись, фамилия, имя, отчество, группа)

Городилов Никита Сергеевич, КТбо2-10
(подпись, фамилия, имя, отчество, группа)

Таганрог 2020 г.

Реферат

Пояснительная записка содержит 85 страниц, 9 рисунков, 6 таблиц и 6 источников.

Суть проекта заключается в аппаратной реализации криптографического алгоритма “Кузнечик” на программируемой логической интегральной схеме с использованием языка VHDL.

Целью работы является изучение алгоритмов криптографии и основ цифровой схемотехники, создание рабочего прототипа аппаратно-программного криптографического устройства для скоростного шифрования и расшифровки данных.

Содержание

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Необходимо переложить криптографический алгоритм “Кузнечик” ГОСТ Р 34.12-2015 на ПЛИС (программируемую логическую интегральную схему).

Основным преимуществом такого подхода к решению задачи является возможность распараллеливать задачи, выполнять несколько циклов и итераций алгоритма одновременно, что позволяет достичь в разы большей скорости вычислений.

Описание аппаратуры необходимо провести используя язык VHDL. Так же в рамках проекта необходимо ознакомиться с принципами работы криптоалгоритмов блочного шифрования, а также с основами разработки устройств на основе ПЛИС

Конечный продукт: функционирующий аппаратно-программный комплекс, способный обеспечить скорость шифровки и дешифровки, достаточную для обмена данными в реальном времени.

Потенциальные клиенты: государственные структуры, банки. Плата для разработки Artix-7, Рисунок 1.

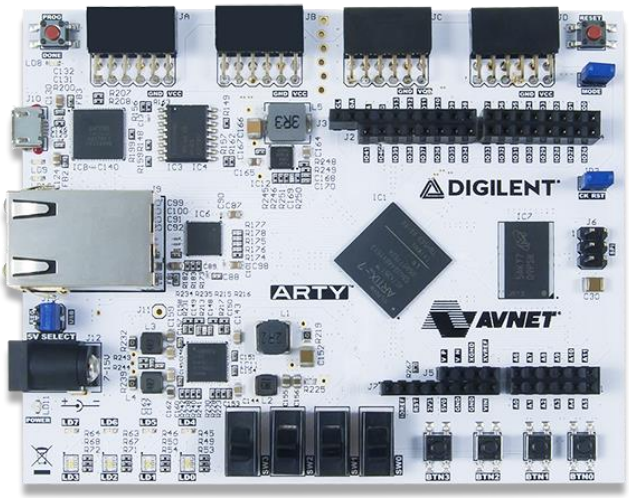


Рис. 1 –Внешний вид платы “Arty A7”

2 ВВЕДЕНИЕ

Мы каждый день пользуемся интернетом, передаём друг другу сообщения в социальных сетях, посещаем разные сайты. Однако, скрытыми от пользователя остаются многие аспекты этой деятельности. Через чьи сервера проходит трафик? Сколько раз он успевает пересечь границу, побывать в другой стране или даже на другом континенте, прежде чем ваш собеседник получил отправленное вами сообщение? В таких условиях сохранять конфиденциальность помогает шифрование трафика.

Всё это требует вычислительных мощностей. Однако если вашу с собеседником переписку удастся перехватить и расшифровать, вы окажетесь в неприятном положении. Но ситуация окажется катастрофической, если была рассекречена не просто личная переписка, а государственная тайна или иная сверхважная информация. Чтобы такого не происходило для шифрования подобного контента используются более надежные, но и более требовательные к ресурсам алгоритмы. При этом количество информации передаваемой по этим каналам огромно.

В связи с этим программные реализации криптоалгоритмов уже не способны с достаточной скоростью решить эту проблему. Требуется другой подход, переложение криптоалгоритмов на аппаратную основу. Реализованные таким образом специализированные схемы способны в разы быстрее решать подобные задачи. Созданием подобной системы занимается наша команда.

3 РАСПРЕДЕЛЕНИЕ РОЛЕЙ

Только слаженные действия всех членов команды могут привести к успеху команды в целом. Выбор ролей для каждого её члена является очень важной и ответственной задачей. Каждый участник проекта должен чётко понимать, какие функции он должен выполнять, какой продукт на выходе получить, на каком этапе сейчас находится проект. Роли должны быть распределены так, чтобы участники команды могли дополнять друг друга, помогать друг другу, таким образом достигая поставленных целей. Если команда разобщена, проект обречён.

Список участников команды с соответствующими им ролями представлен на Таблице 1.

ФИО	Роль в команде
Кочубей Даниил Сергеевич	Инженер-программист
Нестеренко Пётр Алексеевич	Программист-конструктор (капитан команды)
Жалнин Дмитрий Игоревич	Программист
Пучкова Анастасия Денисовна	Программистка
Городилов Никита Сергеевич	Инженер-программист

Таблица 1 - распределение ролей

4 РЕАЛИЗАЦИЯ КРИПТОАЛГОРИТМА “КУЗНЕЧИК”

Реализация, созданная в рамках проекта, отличается от эталонной. Было принято решения, для упрощения отладки и симуляции, сократить размеры блока данных со 128 до 16 бит и блока ключа с 256 до 32 бит. Так же было уменьшено количество раундов, теперь их 4 вместо 10. В остальном алгоритм остался неизменным.

4.1 Этапы шифрования сообщений

На вход алгоритму подаются 128 бит данных подлежащих шифрованию и ключ длинной 256 бит**.** Далее на каждом раунде выполняются 3 основные функции шифрования:

- X-функция. Побитовое “исключающее или” (далее и везде XOR) блока данных и раундового ключа(его генерацию мы разберём позже).
- S-функция. Нелинейное преобразование. По сути это просто побайтовая замена одних значений на другие в соответствии с таблицей.
- L-функция. Линейное преобразование. Хитрая операция с полиномиальным умножением в полях Галуа. Рассмотрим его подробнее позже. Если кратко, то идея состоит в том, что мы умножаем (полиномиально) байты блока данных на соответствующие им в таблице L-преобразования (каждый на каждый), далее складываем всю пачку и записываем, предварительно выполнив сдвиг влево, в младший байт. Без преувеличения это самая сложная операция всего алгоритма.

Все эти шаги повторяются на протяжении 9-и раундов раз за разом, кроме 10-ого, в нём есть только X-функция.

4.2 Генерация раундовых ключей

Процесс генерации раундовых ключей основан на сетях Фейстеля и состоит из 8 раундов**.** Первая пара ключей формируется в результате разделения мастер-ключа на две части, каждая последующая пара раундовых ключей получается при помощи 8 итераций сети Фейстеля. На каждой итерации используется итерационная константа, полученная линейным преобразованием номера итерации, Рисунок 2.

- X-функция. Тут так же происходит побитовый XOR. Левая часть ключа складывается с итерационными константами (заранее рассчитанными, приведём их расчёт позже).
- S-функция. Работает так же как и всегда (над левой частью!!!).
- L-функция. Аналогично.

- X-функция с правой половиной + замена местами. Последний этап это побитовый XOR с правой половиной исходного ключа. Далее правая и левая часть меняются местами.

Далее мы повторяем X,S,L и X с правой частью 8 раз для получения новой пары ключей (8 раундов сети Фейстеля на каждую пару ключей).

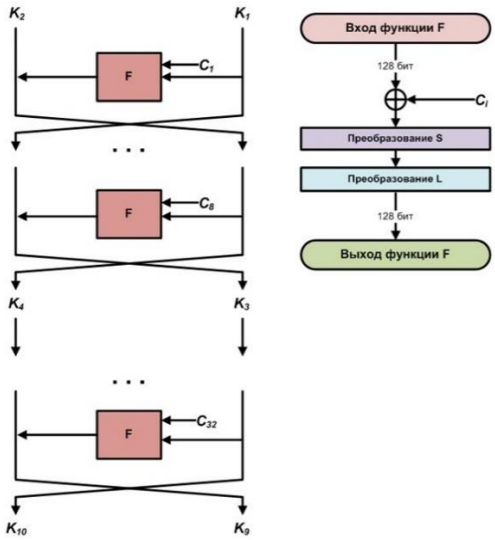


Рис. 2 – Схема создания раундовых ключей

4.3 Реализация X-функции. Побитовый XOR

Побитовое исключающие или двух сущностей, раундового ключа и блока данных. Листинг представлен ниже:

```
procedure xor16bit(signal firBlock, secBlock: in unsigned(0 to 15); signal result: out unsigned(0 to 15)) is
        begin
            result(0)  <= firBlock(0)  xor secBlock(0);
            result(1)  <= firBlock(1)  xor secBlock(1);
            result(2)  <= firBlock(2)  xor secBlock(2);
            result(3)  <= firBlock(3)  xor secBlock(3);
            result(4)  <= firBlock(4)  xor secBlock(4);
            result(5)  <= firBlock(5)  xor secBlock(5);
            result(6)  <= firBlock(6)  xor secBlock(6);
            result(7)  <= firBlock(7)  xor secBlock(7);
            result(8)  <= firBlock(8)  xor secBlock(8);
            result(9)  <= firBlock(9)  xor secBlock(9);
            result(10) <= firBlock(10) xor secBlock(10);
```

```
            result(11) <= firBlock(11) xor secBlock(11);
            result(12) <= firBlock(12) xor secBlock(12);
            result(13) <= firBlock(13) xor secBlock(13);
            result(14) <= firBlock(14) xor secBlock(14);
            result(15) <= firBlock(15) xor secBlock(15);
        end xor16bit;
```

4.4 Реализация S-функции. Нелинейное преобразование

Так как функция S это по сути замена одних значений на другие в соответствии с таблицей замен. Был написан скрипт на Си, где десятичные цифры переводились в бинарный вид, а затем формировались условия для оператора case. Проще понять это на примере: первое число у нас 0**,** по нулевому индексу в таблице замен находится число 252. Если перевести его в двоичный вид, получится 11111100. Из этого следует, что мы ставим в соответствие вектору 00000000 вектор 11111100. На этом принципе основана работа S-функции в нашем алгоритме.

Часть листинга S-функции представлена ниже:

```
    procedure tableShift(signal inputPart: in unsigned(0 to 7);
signal result:out unsigned(0 to 7)) is
        begin
            C: case inputPart is
                    when "00000000" =>
                       result(0) <= '1';
                       result(1) <= '1';
                       result(2) <= '1';
                       result(3) <= '1';
                       result(4) <= '1';
                       result(5) <= '1';
                       result(6) <= '0';
                       result(7) <= '0';
```

4.5 Реализация L-функции. Линейное преобразование

Нам нужно переводить числа из двоичной системе счисления в десятичную, далее по таблице степеней выяснить чему соответствует то или иное значение, после чего необходимо его заменить и перевести снова в двоичный вид. После этого можно провести xor все байтов и запись в младший получившейся суммы. Код при этом выглядит так:

```
numToPower(numba=>outUnsigned(0 to 7), result=>testKey1);
numToPower(numba=>outUnsignedD(0 to 7), result=>testKey2);
numToPower(numba=>testCoef1,result=>testCoef1Tab);
numToPower(numba=>testCoef2, result=>testCoef2Tab);

firBlocNoCon(0 to 7) <= to_unsigned(to_integer(testKey1) +
to_integer(testCoef1Tab), 8);
SecBlocNoCon(0 to 7) <= to_unsigned(to_integer(testKey2) +
to_integer(testCoef2Tab), 8);

     powToNumber(powa=>firBlocNoCon, result=>firBlocCon);
powToNumber(powa=>SecBlocNoCon,result=>SecBlocCon);
     outputBlockPo(0)     <= firBlocCon(0) xor SecBlocCon(0);
outputBlockPo(1) <= firBlocCon(1) xor SecBlocCon(1);
outputBlockPo(2) <= firBlocCon(2) xor SecBlocCon(2);
outputBlockPo(3) <= firBlocCon(3) xor SecBlocCon(3);
outputBlockPo(4) <= firBlocCon(4) xor SecBlocCon(4);
outputBlockPo(5) <= firBlocCon(5) xor SecBlocCon(5);
outputBlockPo(6) <= firBlocCon(6) xor SecBlocCon(6);
outputBlockPo(7) <= firBlocCon(7) xor SecBlocCon(7);
outputBlockPo(8)     <= firBlocCon(0);
outputBlockPo(9)     <= firBlocCon(1);
outputBlockPo(10)    <= firBlocCon(2);
outputBlockPo(11)    <= firBlocCon(3);
outputBlockPo(12)    <= firBlocCon(4);
outputBlockPo(13)    <= firBlocCon(5);
outputBlockPo(14)    <= firBlocCon(6);
outputBlockPo(15)    <= firBlocCon(7);
```

4.6 Генерация раундовых ключей для X-функции.

Как мы выяснили ранее процесс генерации раундовых ключей основан на тех же операциях, что и “основной алгоритм шифрования блока данных”. Единственным отличием будет X-функция, в случае с раундовым ключом выполняется XOR с итерационной константой, специальной переменной заранее просчитанной для каждого раунда сети Фейстеля.

Вторым важным отличием является то, что после X,S,L преобразования необходимо ещё и сложить(имеется ввиду исключающее или) левую и правую часть ключа, после чего поменять их местами. Листинг функции генерации раундового ключа, с этапами отличными от основного процесса шифрования, представлен ниже:

```
procedure keyGenRound(signal masterKey2, masterKey1, c1: in unsigned(0 to 15); signal secIterIn:out unsigned(0 to 15);

signal firIterS1, firIterS2, firIterSum1, firIterSum2, firIterPow1, firIterPow2, firIterNum1, firIterNum2: inout unsigned(0 to 7);

signal intCoef1, intCoef2:in integer; signal firIterRes, firIterXor: inout unsigned(0 to 15)) is

begin

      xor16bit(firBlock=>masterKey2, secBlock=>c1, result=>firIterXor)

      /* вызовы S и L функций */

      xor16bit(firBlock=>masterKey1, secBlock=>firIterRes, result=>secIterIn)

end keyGenRound;
```

5 СХЕМЫ КОНЕЧНОГО УСТРОЙСТВА

5.1 Схема подключения к Microblaze

IP-ядра расшифровки и дешифровки подключены к софт-процессорному ядру Microblaze. Рисунок 3.

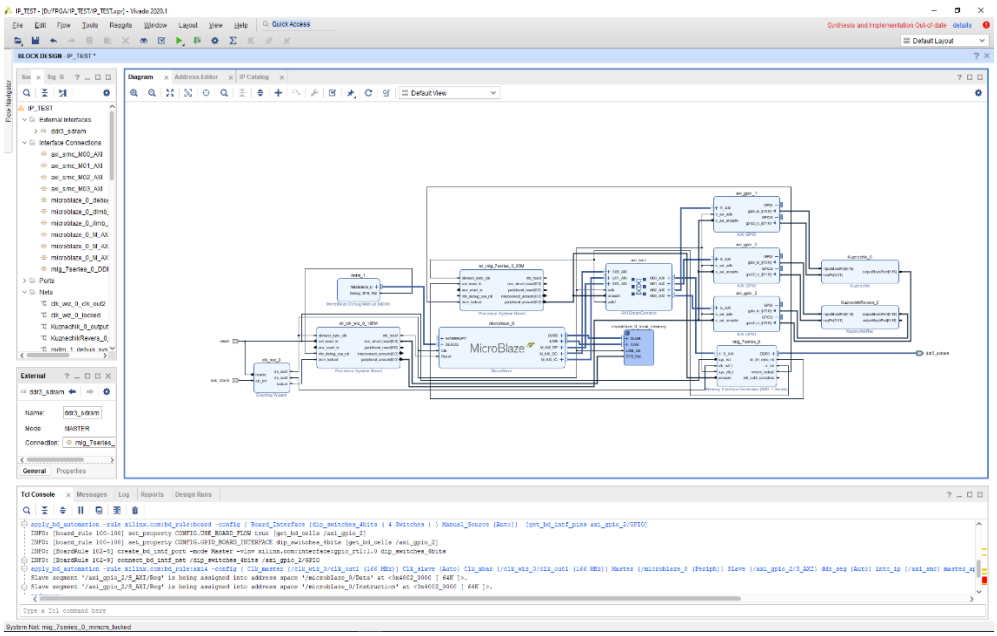


Рис. 3 - Схема с microblaze

5.2 Итоговая сгенерированная электронная схема

Написанный нами код на языке VHDL код был преобразован средой Vivado в электронную схему, представленную на рисунке Рисунке 4.

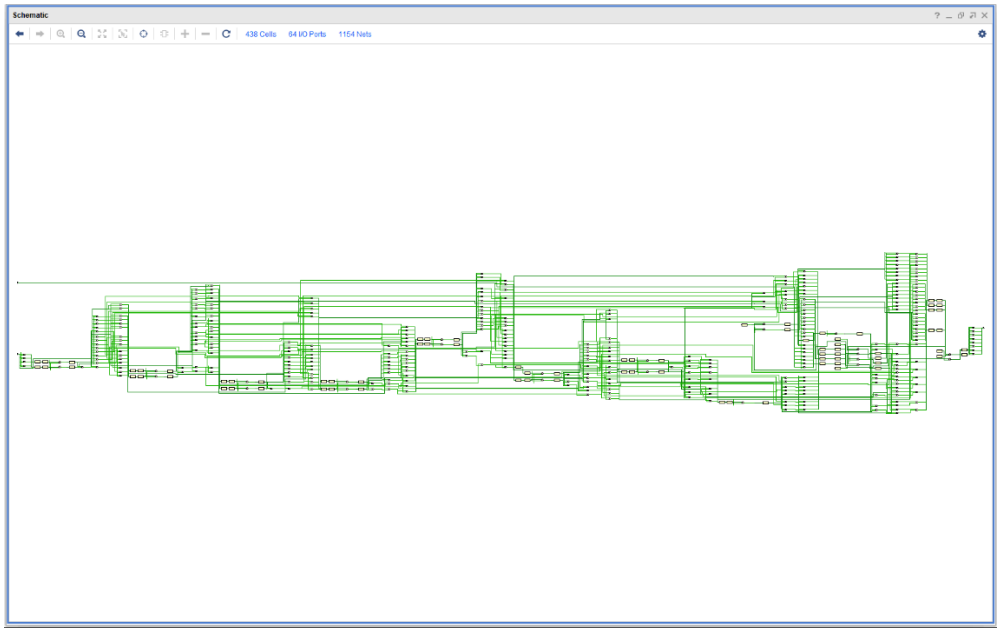


Рис. 4 – Итоговая схема устройства

5.3 Результаты симуляции в логическом анализаторе

Были проведены предварительные симуляционные испытания с использованием встроенного в среду Vivado логического анализатора. Так же был написан код симуляционного файла на языке VHDL. Основная часть кода представленна ниже, нах Рис. 5 и Рис. 6 представлены результаты симуляции.

```
begin
        UUT: main port map(
            inputBlockPo => inputBlock ,
            outputBlockPort => outputBlock,
            KeyPo => leftKey
        );
        process
        begin
        inputBlock <= x"cbc2";
        leftKey <= to_unsigned(0, 32);
        wait for 100 ns;

        wait for 100 ns;
        leftKey  <= x"ffffffff";
        inputBlock <=  to_unsigned(0, 16);
        wait for 100 ns;
        end process;
end Behavioral;
```

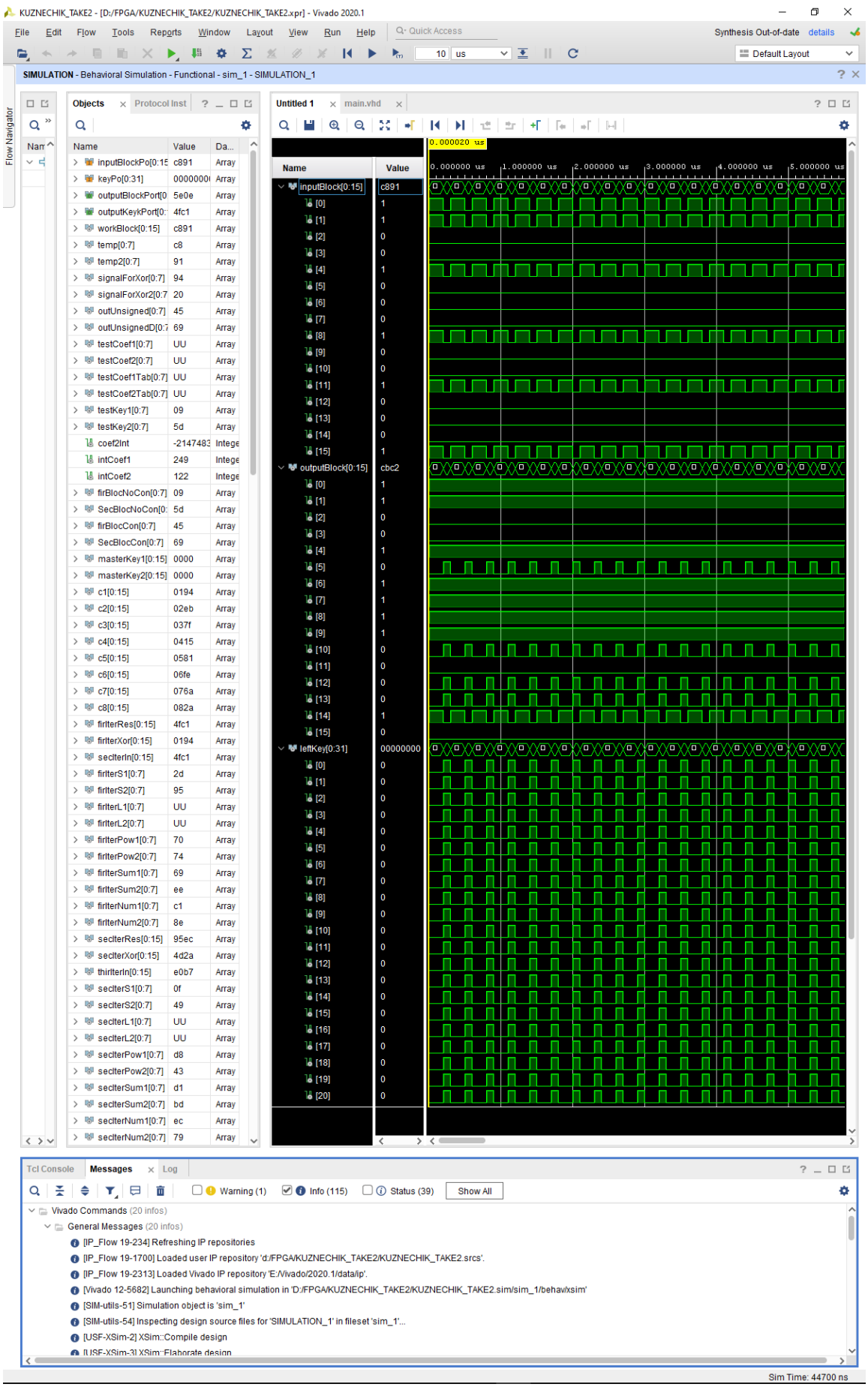


Рис. 5 – Симуляция работы схемы(1)

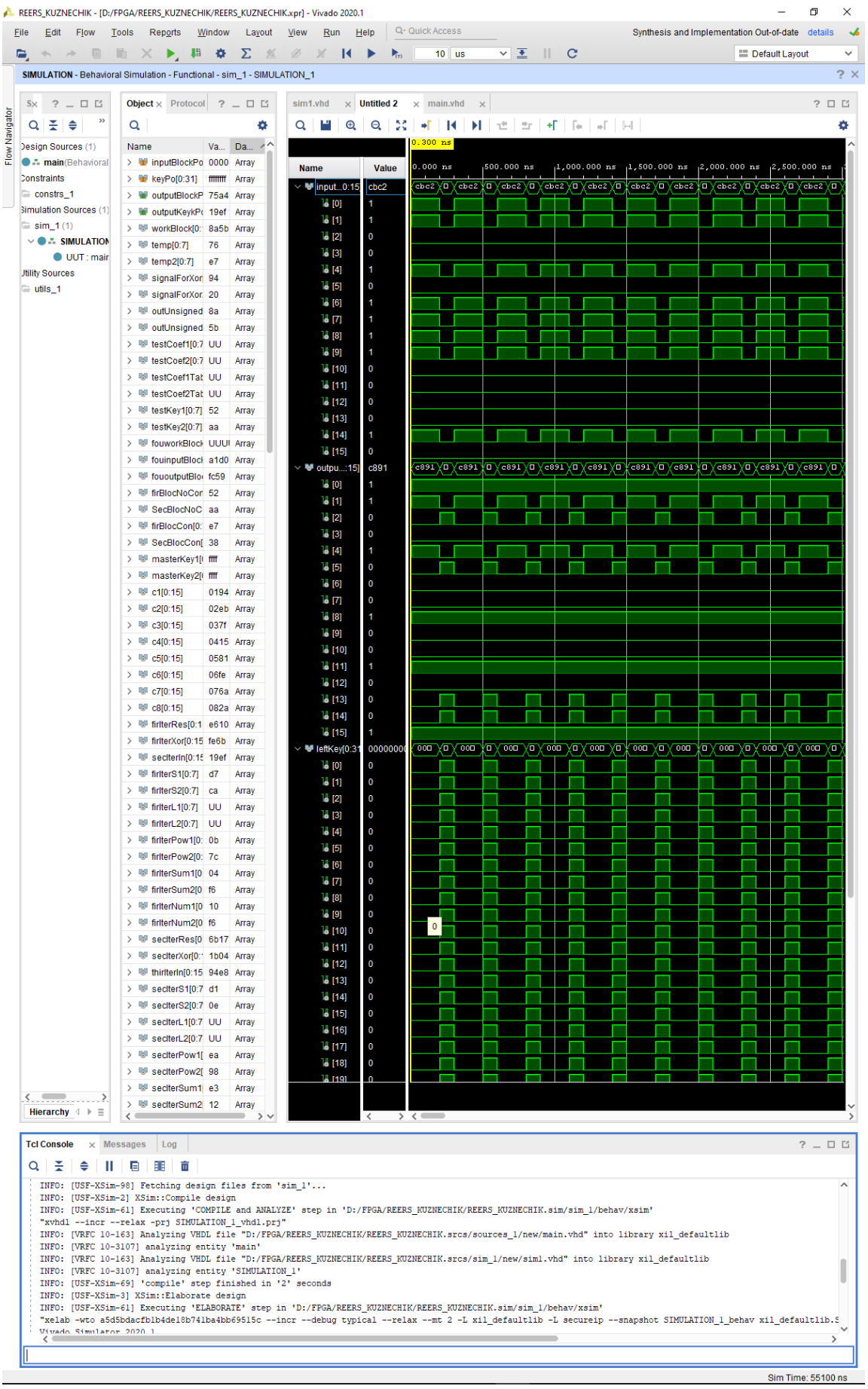


Рис. 6 – Симуляция работы схемы(2)

6 АНАЛИЗ УСТРОЙСТВ-АНАЛОГОВ (ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ)

В настоящий момент на рынке присутствует ряд компаний и устройств, предоставляющих схожий с нашим функционал: абонентский шифраторы компании ANCUD “Криптон-10” и “Криптон-8”, а также же платы производства ОКБ САПР серии “Аккорд”. У каждого из них есть свои преимущества и недостатки. Опишем их подробнее.

6.1 Криптон-10

Рис. 7 –Внешний вид платы “Криптон-10”

Минусы:

1) Алгоритм шифрования ГОСТ 28147-89, морально устаревший алгоритм с множеством проблем. Придуман и реализован ещё в прошлом веке.

не отвечает современным стандартам.

2) Не поддерживает современные операционные системы. Нет поддержки Windows 8, Windows 10 и Linux.

Плюсы:

1) Реализован в формате PCI-платы, удобен в установке, не занимает лишнего места на столе пользователя

6.2 Аккорд СБ

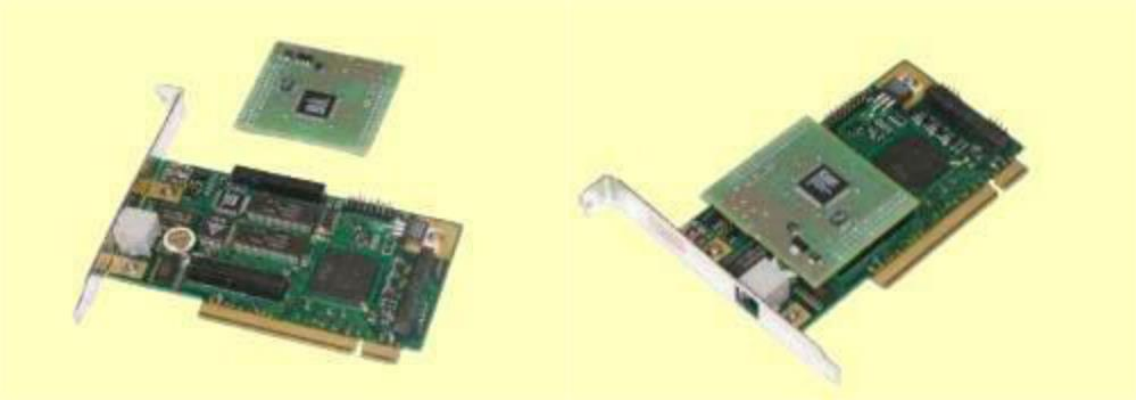


Рис. 8 – Внешиний вид платы «Аккорд СБ»

Минусы:

1) Аналогично “Криптон-10” Морально устаревший алгоритм шифрования ГОСТ 28147-89.

2) Старые комплектующие, не способные на данный момент эффективно соперничать с современными аналогами.

Плюсы:

1) Поддержка ОС Linux.
2) Возможность работать с большим количеством файловых систем(FAT 12, FAT 16, FAT 32, NTFS, HPFS, FreeBSD, Linux EXT2FS)
3) Существуют существенно более мощные и современные модификации.

Таблица 2 – устройства-аналоги

Название	Криптон-10	Аккорд СБ
Разработчик	ANCUD	ОКБ САПР
Поддержка ОС	MS-DOS, Windows 2000 x32, Windows XP x32, Windows 7 x32/x64, Windows Server 2003/2008	MS DOS, Windows 9x, Windows Millenium, Windows NT, Windows 2000, OS/2, UNIX, Linux.

Стандарт шифрования	ГОСТ 28147-89	ГОСТ 28147-89
Наличие модификаций	Отсутствуют	Присутсвуют
На основе ПЛИС	Да	Нет

Таким образом можно сделать вывод, что имеющиеся на рынке устройства аппаратного шифрования имею существенный недостаток. Поэтому наша разработка актуальна.

6.3 Анализ языков программирования

Таблица 3 показывает относительную оценку языков в диапазоне от 0 до 10 по наиболее важным критериям.

Таблица 3 – языки программирования

	Читабельность	Простота	Скорость разработки	Живое комьюнити программистов
VHDL	4	6	8	9
Verilog	7	4	6	5

Язык VHDL является наиболее сбалансированным языком программирования для описания электронных схем в частности ПЛИС (FPGA)

7 РАСЧЁТ СТОИМОСТИ РАЗРАБОТКИ

Таблица 4. Расчет количества часов на разработку

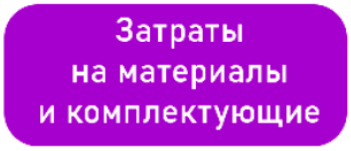
№	Название этапа разработки	Количество человек, задействованных в разработке этапа	Количество часов, потраченных на этап
1	Обучение работы в программе Vivado 2016	5	46
2	Проработка плана работы	5	5
3	Анализ ресурсов и возможностей. Заказ платы.	5	4
4	Создание прошивки для внедрения в плату	5	64
5	Прошивка и работа с платой	5	24
6	Продумывание дизайна платы	3	19
7	Проектирование и печать коробки для платы	2	4
8	Тестирование работоспособности платы	5	43
9	Обработка и исправление ошибок	5	67
10	Приведение устройства в надлежащий вид	5	24
	ИТОГО:		300

Итогом блока «Количество часов на разработку» будет суммарное количество часов.

На данном этапе проанализирован рынок рабочей силы, занимающегося тем или иным этапом.

Таблица 5. Расчет количества часов на разработку и цена часа работы

№	Название этапа разработки	Количество человек, задействованных в разработке этапа	Количество часов, потраченных на этап	Цена часа работы (руб.)	Стоимость разработки
1	Построение архитектуры корпуса и его печать	5	23	800	92000
2	Создание прошивки и прошивка устройства	5	133	800	532000
3	Тестирование и отладка системы	5	144	800	576000
	ИТОГО				1200000



Этап имеет перечень основных деталей, комплектующих и материалов, задействованных в создании конечного продукта. Таблица 6.

№	Название материала/комплектующих	Цена за ед. (руб.)	Количество в готовом изделии	Общая стоимость (руб.)
1	Плата Arty	8700	1	8700
2	Корпус для платы	300	1	300
	ИТОГО			9000

8 РЕЗУЛЬТАТ ПРОДЕЛАННОЙ РАБОТЫ.

В результате выполнения работы мы:

- познакомились с принципами работы блочного шифра “Кузнечик”, командой были созданы два специализированных ip-ядра, шифрующее и расшифровывающее.
- Выполнели симуляции для проверки правильности работы алгоритма в логическом анализаторе среды Vivado, подтвердившие работоспособность описанного на VHDL алгоритма.
- Реализовали микропроцессорный интерфейс для организации удобного обмена данными между компьютером(пользователем) и устройством.

Так же был разработан корпус устройства в САПР Autodesk Fusion360, после чего он был распечатан на 3D принтере, Рисунок 9.

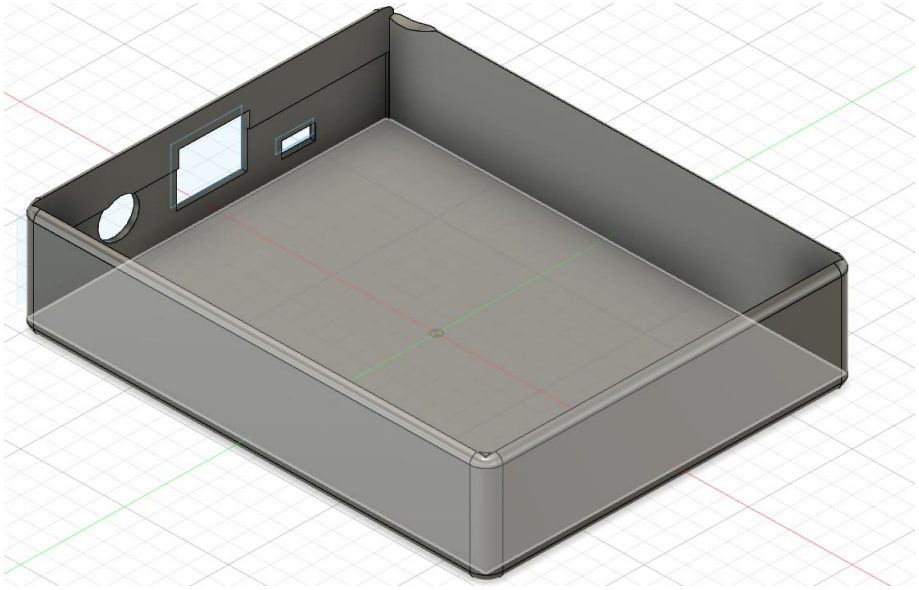


Рис. 9 –Внешний вид платы корпуса устройства

СПИСОК ИСТОЧНИКОВ

1. Криптографический алгоритм «Кузнечик»: просто о сложном // Habr URL: https://habr.com/ru/post/459004
2. Разработка процессорной системы на базе софт-процессора MicroBlaze в среде Xilinx Vivado IDE // fpga-systems URL: https://fpga-systems.ru/publ/xilinx/microblaze/razrabotka_processornoj_sistemy_na_baze_soft_processora_microblaze_v_srede_xilinx_vivado_ide_hlx_chast_2/10-1-0-7
3. Creating a custom IP block in Vivado // fpga-developer URL: http://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-in-vivado.html
4. П. Н. Бибило Основы языка VHDL. Либроком, 2016.
5. И.И. Левин, Б.Е. Механцев Лабораторные работы по дисциплине "ПЛИС-технологии и методы создания эффективных прикладных программ для ПЛИС". Таганрог: Издательство Южного федерального университета, 2017.
6. Xilinx [Электронный ресурс]. – URL: https://www.xilinx.com/products/silicon-devices/fpga.html (дата обращения: 31.04.2020).

Приложение А. Основная функция шифровки блока

```
    procedure textBlockEncoding(signal inputBlockPo, keyPo: in
unsigned(0 to 15);
            signal workBlock: inout unsigned(0 to 15);
            signal temp, temp2, outUnsigned, outUnsignedD, testKey1,
testKey2,   testCoef1,   testCoef1Tab,   testCoef2,   testCoef2Tab,
firBlocNoCon, SecBlocNoCon, firBlocCon, SecBlocCon: inout unsigned(0 to
7);
            signal outputBlockPo: out unsigned(0 to 15)) is
        begin
            --x
                workBlock(0) <= InputBlockPo(0) xor keyPo(0);
                workBlock(1) <= InputBlockPo(1) xor keyPo(1);
                workBlock(2) <= InputBlockPo(2) xor keyPo(2);
                workBlock(3) <= InputBlockPo(3) xor keyPo(3);
                workBlock(4) <= InputBlockPo(4) xor keyPo(4);
                workBlock(5) <= InputBlockPo(5) xor keyPo(5);
                workBlock(6) <= InputBlockPo(6) xor keyPo(6);
                workBlock(7) <= InputBlockPo(7) xor keyPo(7);
                workBlock(8) <= InputBlockPo(8) xor keyPo(8);
                workBlock(9) <= InputBlockPo(9) xor keyPo(9);
                workBlock(10) <= InputBlockPo(10) xor keyPo(10);
                workBlock(11) <= InputBlockPo(11) xor keyPo(11);
                workBlock(12) <= InputBlockPo(12) xor keyPo(12);
                workBlock(13) <= InputBlockPo(13) xor keyPo(13);
                workBlock(14) <= InputBlockPo(14) xor keyPo(14);
                workBlock(15) <= InputBlockPo(15) xor keyPo(15);

            --s
                temp(0) <= workBlock(0);
                temp(1) <= workBlock(1);
                temp(2) <= workBlock(2);
                temp(3) <= workBlock(3);
                temp(4) <= workBlock(4);
```

```
                temp(5) <= workBlock(5);
                temp(6) <= workBlock(6);
                temp(7) <= workBlock(7);
                tableShift(inputPart=>temp, result=>outUnsigned);
                --second block conversion
                temp2(0) <= workBlock(8);
                temp2(1) <= workBlock(9);
                temp2(2) <= workBlock(10);
                temp2(3) <= workBlock(11);
                temp2(4) <= workBlock(12);
                temp2(5) <= workBlock(13);
                temp2(6) <= workBlock(14);
                temp2(7) <= workBlock(15);
                tableShift(inputPart=>temp2, result=>outUnsignedD);

            --l
                numToPower(numba=>outUnsigned(0 to 7),
result=>testKey1);
                 numToPower(numba=>outUnsignedD(0 to 7),
result=>testKey2);
                 numToPower(numba=>testCoef1, result=>testCoef1Tab);
                 numToPower(numba=>testCoef2, result=>testCoef2Tab);

                 firBlocNoCon(0 to 7) <=
to_unsigned(to_integer(testKey1) + to_integer(testCoef1Tab), 8);
                 SecBlocNoCon(0 to 7) <=
to_unsigned(to_integer(testKey2) + to_integer(testCoef2Tab), 8);

                 powToNumber(powa=>firBlocNoCon, result=>firBlocCon);
                 powToNumber(powa=>SecBlocNoCon, result=>SecBlocCon);

                 outputBlockPo(0) <= firBlocCon(0) xor
SecBlocCon(0);
```

```
                outputBlockPo(1)            <=   firBlocCon(1)   xor
SecBlocCon(1);
                outputBlockPo(2)            <=   firBlocCon(2)   xor
SecBlocCon(2);
                outputBlockPo(3)            <=   firBlocCon(3)   xor
SecBlocCon(3);
                outputBlockPo(4)            <=   firBlocCon(4)   xor
SecBlocCon(4);
                outputBlockPo(5)            <=   firBlocCon(5)   xor
SecBlocCon(5);
                outputBlockPo(6)            <=   firBlocCon(6)   xor
SecBlocCon(6);
                outputBlockPo(7)            <=   firBlocCon(7)   xor
SecBlocCon(7);
                outputBlockPo(8)     <= firBlocCon(0);
                outputBlockPo(9)     <= firBlocCon(1);
                outputBlockPo(10)    <= firBlocCon(2);
                outputBlockPo(11)    <= firBlocCon(3);
                outputBlockPo(12)    <= firBlocCon(4);
                outputBlockPo(13)    <= firBlocCon(5);
                outputBlockPo(14)    <= firBlocCon(6);
                outputBlockPo(15)    <= firBlocCon(7);
        end textBlockEncoding;
```

Приложение В. Функция генерации раундового ключа

```
     procedure keyGenRound(signal masterKey2, masterKey1, c1: in
unsigned(0 to 15); signal secIterIn:out unsigned(0 to 15);
                signal firIterS1, firIterS2, firIterSum1, firIterSum2,
firIterPow1, firIterPow2, firIterNum1, firIterNum2: inout unsigned(0 to
7);
                signal intCoef1, intCoef2:in integer; signal firIterRes,
firIterXor: inout unsigned(0 to 15)) is
begin
     xor16bit(firBlock=>masterKey2, secBlock=>c1, result=>firIterXor);
```

```
      tableShift(inputPart=>firIterXor(0 to 7), result=>firIterS1);
      tableShift(inputPart=>firIterXor(8 to 15), result=>firIterS2);

      numToPower(numba=>firIterS1(0 to 7), result=>firIterPow1);
      numToPower(numba=>firIterS2(0 to 7), result=>firIterPow2);

      firIterSum1(0 to 7) <= to_unsigned(to_integer(firIterPow1) +
intCoef1, 8);
      firIterSum2(0 to 7) <= to_unsigned(to_integer(firIterPow2) +
intCoef2, 8);


      powToNumber(powa=>firIterSum1, result=>firIterNum1);
      powToNumber(powa=>firIterSum2, result=>firIterNum2);

      firIterRes(0)     <= firIterNum1(0) xor firIterNum2(0);
      firIterRes(1)     <= firIterNum1(1) xor firIterNum2(1);
      firIterRes(2)     <= firIterNum1(2) xor firIterNum2(2);
      firIterRes(3)     <= firIterNum1(3) xor firIterNum2(3);
      firIterRes(4)     <= firIterNum1(4) xor firIterNum2(4);
      firIterRes(5)     <= firIterNum1(5) xor firIterNum2(5);
      firIterRes(6)     <= firIterNum1(6) xor firIterNum2(6);
      firIterRes(7)     <= firIterNum1(7) xor firIterNum2(7);
      firIterRes(8)     <= firIterNum1(0);
      firIterRes(9)     <= firIterNum1(1);
      firIterRes(10)    <= firIterNum1(2);
      firIterRes(11)    <= firIterNum1(3);
      firIterRes(12)    <= firIterNum1(4);
      firIterRes(13)    <= firIterNum1(5);
      firIterRes(14)    <= firIterNum1(6);
      firIterRes(15)    <= firIterNum1(7);
```

```
      xor16bit(firBlock=>masterKey1,              secBlock=>firIterRes,
result=>secIterIn);
                end keyGenRound;
```

ПРИЛОЖЕНИЕ С. ФУНКЦИЯ ЗАМЕНЫ ЧЕРЕЗ ТАБЛИЦУ

```
procedure tableShift(signal inputPart: in unsigned(0 to 7); signal result:out unsigned(0 to 7)) is
        begin
            C: case inputPart is
                    when "11111100" =>
                      result(0) <= '0';
                      result(1) <= '0';
                      result(2) <= '0';
                      result(3) <= '0';
                      result(4) <= '0';
                      result(5) <= '0';
                      result(6) <= '0';
                      result(7) <= '0';
                    when "11101110" =>
                      result(0) <= '0';
                      result(1) <= '0';
                      result(2) <= '0';
                      result(3) <= '0';
                      result(4) <= '0';
                      result(5) <= '0';
                      result(6) <= '0';
                      result(7) <= '1';
                    when "11011101" =>
                      result(0) <= '0';
                      result(1) <= '0';
                      result(2) <= '0';
                      result(3) <= '0';
                      result(4) <= '0';
                      result(5) <= '0';
                      result(6) <= '1';
                      result(7) <= '0';
                    when "00010001" =>
                      result(0) <= '0';
                      result(1) <= '0';
                      result(2) <= '0';
                      result(3) <= '0';
                      result(4) <= '0';
                      result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "11001111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "01101110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00110001" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "00010110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11111011" =>
    result(0) <= '0';
```

```
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "11000100" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "11111010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "11011010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "00100011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11000101" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00000100" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01001101" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11101001" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01110111" =>
```

```
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "11110000" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "11011011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "10010011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00101110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
```

```
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "10011001" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10111010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00010111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00110110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
```

```
when "11110001" =>
  result(0) <= '0';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "10111011" =>
  result(0) <= '0';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "00010100" =>
  result(0) <= '0';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "11001101" =>
  result(0) <= '0';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "01011111" =>
  result(0) <= '0';
  result(1) <= '0';
  result(2) <= '0';
```

```
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '0';
        when "11000001" =>
            result(0) <= '0';
            result(1) <= '0';
            result(2) <= '0';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '1';
        when "11111001" =>
            result(0) <= '0';
            result(1) <= '0';
            result(2) <= '1';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '0';
        when "00011000" =>
            result(0) <= '0';
            result(1) <= '0';
            result(2) <= '1';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '1';
        when "01100101" =>
            result(0) <= '0';
            result(1) <= '0';
            result(2) <= '1';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '1';
```

```
    result(7) <= '0';
  when "01011010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "11100010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "01011100" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "11101111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "00100001" =>
    result(0) <= '0';
    result(1) <= '0';
```

```
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '0';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '1';
      when "10000001" =>
        result(0) <= '0';
        result(1) <= '0';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '0';
      when "00011100" =>
        result(0) <= '0';
        result(1) <= '0';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '1';
      when "00111100" =>
        result(0) <= '0';
        result(1) <= '0';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
        result(6) <= '1';
        result(7) <= '0';
      when "01000010" =>
        result(0) <= '0';
        result(1) <= '0';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "10001011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00000001" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "10001110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01001111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00000101" =>
    result(0) <= '0';
```

```
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10000100" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00000010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10101110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "11100011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "01101010" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "10001111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10100000" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00000110" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00001011" =>
```

```
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "11101101" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10011000" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "01111111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11010100" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
```

```
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "11010011" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "00011111" =>
    result(0) <= '0';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11101011" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00110100" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
```

```
when "00101100" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '0';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "01010001" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '0';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "11101010" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '0';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "11001000" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '0';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "01001000" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '0';
```

```
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10101011" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11110010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00101010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "01101000" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
```

```
    result(7) <= '0';
  when "10100010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "11111101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00111010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "11001110" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "11001100" =>
    result(0) <= '0';
    result(1) <= '1';
```

```
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10110101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01110000" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00001110" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "01010110" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "00001000" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00001100" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01110110" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "00010010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10111111" =>
    result(0) <= '0';
```

```
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01110010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00010011" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "01000111" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "10011100" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "10110111" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01011101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10000111" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00010101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10100001" =>
```

```
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "10010110" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "00101001" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "00010000" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "01111011" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
```

```
        result(4) <= '0';
        result(5) <= '1';
        result(6) <= '0';
        result(7) <= '1';
      when "10011010" =>
        result(0) <= '0';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '0';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '0';
      when "11000111" =>
        result(0) <= '0';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '0';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '1';
      when "11110011" =>
        result(0) <= '0';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '0';
      when "10010001" =>
        result(0) <= '0';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '1';
```

```
when "01111000" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "01101111" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "10011101" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "10011110" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "10110010" =>
  result(0) <= '0';
  result(1) <= '1';
  result(2) <= '1';
```

```
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10110001" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00110010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01110101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00011001" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
```

```
    result(7) <= '0';
  when "00111101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "11111111" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00110101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "10001010" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01111110" =>
    result(0) <= '0';
    result(1) <= '1';
```

```
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '0';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '1';
        when "01101101" =>
            result(0) <= '0';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '0';
        when "01010100" =>
            result(0) <= '0';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '1';
        when "11000110" =>
            result(0) <= '0';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '0';
            result(6) <= '1';
            result(7) <= '0';
        when "10000000" =>
            result(0) <= '0';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "11000011" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "10111101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00001101" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01010111" =>
    result(0) <= '0';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11011111" =>
    result(0) <= '1';
```

```
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "11110101" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00100100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10101001" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "00111110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "10101000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01000011" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "11001001" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11010111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01111001" =>
```

```
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "11010110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "11110110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "01111100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00100010" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '0';
```

```
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '0';
        result(7) <= '1';
      when "10111001" =>
        result(0) <= '1';
        result(1) <= '0';
        result(2) <= '0';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '0';
      when "00000011" =>
        result(0) <= '1';
        result(1) <= '0';
        result(2) <= '0';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '1';
      when "11100000" =>
        result(0) <= '1';
        result(1) <= '0';
        result(2) <= '0';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '0';
      when "00001111" =>
        result(0) <= '1';
        result(1) <= '0';
        result(2) <= '0';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '1';
```

```
when "11101100" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "11011110" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "01111010" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "10010100" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '0';
  result(3) <= '1';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "10110000" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '0';
```

```
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10111100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11011100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "11101000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00101000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
```

```
    result(7) <= '0';
  when "01010000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "01001110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00110011" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00001010" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01001010" =>
    result(0) <= '1';
    result(1) <= '0';
```

```
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10100111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10010111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "01100000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "01110011" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "00011110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00000000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01100010" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01000100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00011010" =>
    result(0) <= '1';
```

```
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10111000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00111000" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10000010" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "01100100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "10011111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00100110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01000001" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10101101" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01000101" =>
```

```
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "01000110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10010010" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "00100111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "01011110" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
```

```
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01010101" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "00101111" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10001100" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10100011" =>
    result(0) <= '1';
    result(1) <= '0';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
```

```
when "10100101" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '1';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "01111101" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '1';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "01101001" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '1';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "11010101" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '1';
  result(3) <= '1';
  result(4) <= '1';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "10010101" =>
  result(0) <= '1';
  result(1) <= '0';
  result(2) <= '1';
```

```
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '0';
        when "00111011" =>
            result(0) <= '1';
            result(1) <= '0';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '1';
        when "00000111" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '0';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '0';
        when "01011000" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '0';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '0';
            result(7) <= '1';
        when "10110011" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '0';
            result(3) <= '0';
            result(4) <= '0';
            result(5) <= '0';
            result(6) <= '1';
```

```
    result(7) <= '0';
  when "01000000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "10000110" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "10101100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "00011101" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "11110111" =>
    result(0) <= '1';
    result(1) <= '1';
```

```
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "00110000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00110111" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "01101011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "11100100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "10001000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11011001" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "11100111" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "10001001" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11100001" =>
    result(0) <= '1';
```

```
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "00011011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "10000011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "01001001" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "01001100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
```

```
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "00111111" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "11111000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "11111110" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "10001101" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01010011" =>
```

```
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "10101010" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "10010000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "11001010" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11011000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '0';
    result(3) <= '1';
```

```
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '0';
        result(7) <= '1';
      when "10000101" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '0';
        result(3) <= '1';
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '0';
      when "01100001" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '0';
        result(3) <= '1';
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '1';
      when "00100000" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '0';
      when "01110001" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '1';
```

```
when "01100111" =>
  result(0) <= '1';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '0';
when "10100100" =>
  result(0) <= '1';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '0';
  result(6) <= '1';
  result(7) <= '1';
when "00101101" =>
  result(0) <= '1';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '0';
when "00101011" =>
  result(0) <= '1';
  result(1) <= '1';
  result(2) <= '1';
  result(3) <= '0';
  result(4) <= '0';
  result(5) <= '1';
  result(6) <= '0';
  result(7) <= '1';
when "00001001" =>
  result(0) <= '1';
  result(1) <= '1';
  result(2) <= '1';
```

```
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01011011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11001011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "10011011" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "00100101" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
```

```
    result(7) <= '0';
  when "11010000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "10111110" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11100101" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "01101100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '0';
    result(4) <= '1';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "01010010" =>
    result(0) <= '1';
    result(1) <= '1';
```

```
        result(2) <= '1';
        result(3) <= '0';
        result(4) <= '1';
        result(5) <= '1';
        result(6) <= '1';
        result(7) <= '1';
      when "01011001" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '0';
      when "10100110" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '0';
        result(7) <= '1';
      when "01110100" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
        result(6) <= '1';
        result(7) <= '0';
      when "11010010" =>
        result(0) <= '1';
        result(1) <= '1';
        result(2) <= '1';
        result(3) <= '1';
        result(4) <= '0';
        result(5) <= '0';
```

```
    result(6) <= '1';
    result(7) <= '1';
  when "11100110" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '0';
  when "11110100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '0';
    result(7) <= '1';
  when "10110100" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '0';
  when "11000000" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '0';
    result(5) <= '1';
    result(6) <= '1';
    result(7) <= '1';
  when "11010001" =>
    result(0) <= '1';
```

```
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '0';
  when "01100110" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '0';
    result(7) <= '1';
  when "10101111" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '0';
  when "11000010" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
    result(5) <= '0';
    result(6) <= '1';
    result(7) <= '1';
  when "00111001" =>
    result(0) <= '1';
    result(1) <= '1';
    result(2) <= '1';
    result(3) <= '1';
    result(4) <= '1';
```

```
            result(5) <= '1';
            result(6) <= '0';
            result(7) <= '0';
          when "01001011" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '0';
            result(7) <= '1';
          when "01100011" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '0';
          when "10110110" =>
            result(0) <= '1';
            result(1) <= '1';
            result(2) <= '1';
            result(3) <= '1';
            result(4) <= '1';
            result(5) <= '1';
            result(6) <= '1';
            result(7) <= '1';
          when others => null;
        end case C;
end tableShift;
```