УДК 004.383.3

Мыцко Евгений Алексеевич
ФГАОУ ВО «Национальный исследовательский Томский политехнический университет»
Россия, Томск[1]
Магистрант
E-Mail: evgenrus70@mail.ru

Примеры аппаратных реализаций вычисления контрольной суммы Cyclic Redundancy Check 32, совместимой с WinRAR, PKZIP, Ethernet

Аннотация. В данной работе рассмотрены примеры аппаратных реализаций вычисления контрольной суммы cyclic redundancy check 32 для табличного, матричного, двухбайтового и четырёхбайтового матричного алгоритмов. Приведена полная функциональная схема аппаратной реализации cyclic redundancy check 32 по табличному алгоритму, разработанная на языке описания аппаратуры Very high speed integrated circuits Hardware Description Language с применением блочно-ориентированного подхода. Также приведены части функциональной схемы аппаратной реализации для матричного двухбайтового и четырёхбайтового алгоритмов. Приведены примеры исходных кодов аппаратной реализации, описанной на языке Very high speed integrated circuits Hardware Description Language для табличного и матричных алгоритмов. Приведены краткие описания основных блоков функциональной схемы и отражены основные изменения в реализации функциональных блоков для матричных алгоритмов. Кратко описан принцип расчёта контрольной суммы cyclic redundancy check 32 для данных, передаваемых с персонального компьютера по последовательному порту Recommended Standard 232, используя терминал для передачи данных. Показана практическая значимость приведённых примеров аппаратной реализации для применения их на программируемых логических интегральных схемах Cyclone от Altera.

Ключевые слова: контрольная сумма; табличный алгоритм; матричный алгоритм; cyclic redundancy check 32; аппаратная реализация; quartus; макет; последовательный интерфейс; передача данных; функциональная схема.

Идентификационный номер статьи в журнале 139TVN314

[1] 634050, г. Томск, пр-кт Ленина, 30, ТПУ, ИК, Кафедра вычислительной техники

Введение

В свободном доступе [9] и в литературе [10] можно встретить описания программных реализации алгоритмов вычисления контрольной суммы CRC32. Однако описания аппаратных реализаций не встречаются в литературе, а только в некоторых статьях [10,3,7] без конкретных примеров. В данной статье приводятся примеры аппаратных реализаций с описанием функциональной схемы на языке VHDL.

1. Аппаратная реализация табличного алгоритма

На основе исходных кодов [9] программной реализации вычисления контрольной суммы CRC32 была спроектирована и описана на языке VHDL [9] функциональная схема аппаратной реализации CRC32 в среде Quartus II с использованием программируемых логических интегральных схем (ПЛИС) Cyclone макета SDK 6.1 [8].

Ввод данных в макет для расчёта контрольной суммы CRC32 производится с персонального компьютера (ПК). Передача данных с ПК на макет осуществляется по последовательному интерфейсу RS-232, используя терминал для передачи данных.

На рис.1 приведена функциональная схема аппаратной реализации вычисления контрольной суммы CRC32 табличным алгоритмом.

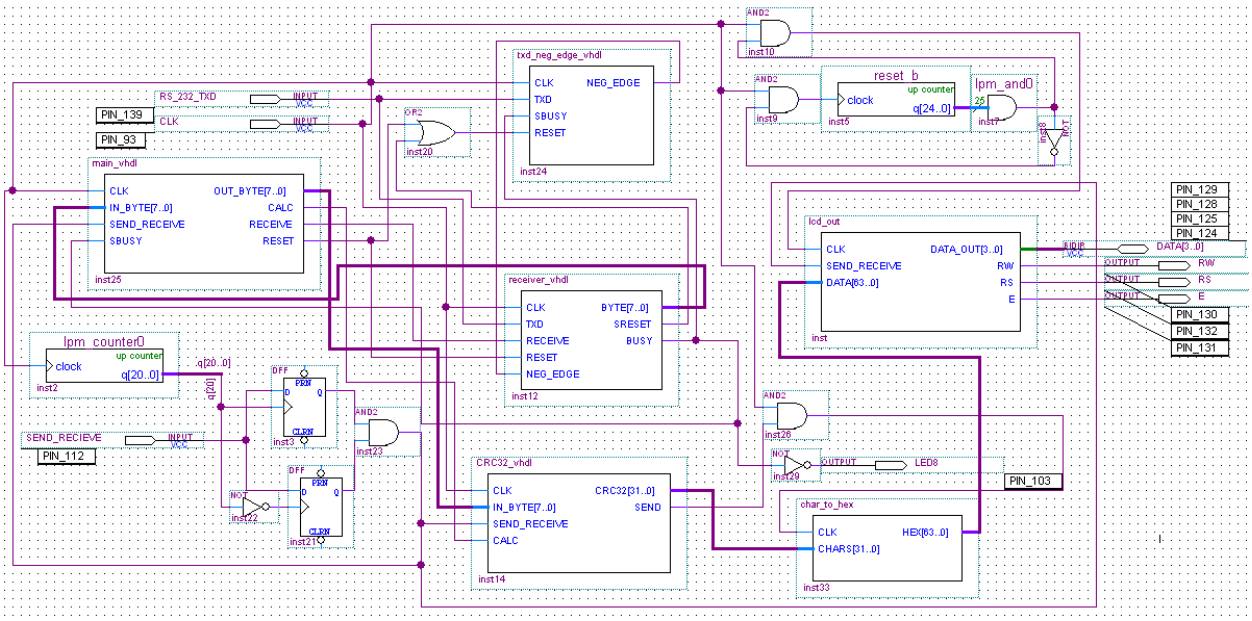

Рис. 1. *Функциональная схема аппаратной реализации вычисления CRC32 табличным алгоритмом*
(разработано автором)

Функциональная схема на рис.1 разработана с использованием блочно-ориентированного подхода [6] и состоит из 6 блоков, описанных на языке VHDL. Условно схему можно разделить на 3 части: входная часть (блоки receiver_vhdl, main_vhdl, txd_neg_edge_vhdl), блок вычисления CRC32 (CRC32_vhdl) и выходная часть (блоки char_to_hex, lcd_out).

Блок receiver_vhdl осуществляет побитовый приём данных с персонального компьютера по протоколу RS232 и формирует входной байт, который передаётся в управляющий блок main_vhdl. В момент приёма данных по последовательному порту блок txd_neg_edge_vhdl обнаруживает START_BIT входного байта.

После окончания приёма байта данных, с помощью управляющего сигнала «CALC» разрешается расчёт контрольной суммы в блоке CRC32_vhdl согласно табличному алгоритму [10]. Далее приведено описание блока CRC32_vhdl на языке VHDL.

В данном описании приведены только начальные и конечные значения таблицы, необходимой для вычисления CRC32 (table(0)…table(255)). Полное описание таблицы приведено в [4].

```
entity CRC32_vhdl is
port(
        CLK    : in std_logic;
        IN_BYTE: in std_logic_vector(7 downto 0);
        SEND_RECEIVE: in std_logic;
        CALC: in std_logic;
        CRC32: out std_logic_vector (31 downto 0);
        SEND: out std_logic
);
end entity;
begin
table(0) <= X"00000000";
table(1) <= X"77073096";
……….
table(254) <= X"5A05DF1B";
table(255) <= X"2D02EF8D";
begin
if ( rising_edge(CLK)) then

        case CALC is
        when '1' =>
                case c_state is
                when s0 =>
                        SEND <='0';
                        index:= TO_INTEGER(unsigned(crc ( 7 downto 0 ) xor IN_BYTE));
                        crc := crc (31 downto 8) xor table(index);
                        when others => c_state <= s0;
                end case;
        when '0' =>
                end case;
```

```
        case SEND_RECEIVE is
                when '1' =>
                        CRC32 <= not crc;
                        SEND <= '1';
                        STOP := '1';
                when '0' =>
                        if ( STOP = '1' ) then
                        crc:=X"FFFFFFFF";
                        STOP := '0';
                        end if;
        end case;
end if;
end process;
end CRC32;
```

По управляющему сигналу «SEND_RECIEVE» на выход CRC32 подаётся рассчитанная контрольная сумма. Далее контрольная сумма поступает на блок преобразования в шестнадцатеричную систему (char_to_hex), а затем на блок вывода (lcd_out) на жидкокристаллический индикатор (ЖКИ).

2. Аппаратная реализация матричного алгоритма

Согласно программной реализации [5] матричный алгоритм имеет несколько модификаций: однобайтовый, двухбайтовый и четырёхбайтовый. Для однобайтового матричного алгоритма функциональная схема не отличается от схемы табличного алгоритма, за исключением описания блока вычисления CRC32 на языке VHDL, которое приведено далее.

```
entity CRC32_vhdl is
port(
        CLK    : in std_logic;
        IN_BYTE: in std_logic_vector(7 downto 0);
        SEND_RECEIVE: in std_logic;
        CALC: in std_logic;
        CRC32: out std_logic_vector (31 downto 0);
        SEND: out std_logic
);
end entity;
begin
matrix(0) <= X"77073096";
matrix(1) <= X"EE0E612C";
```

```
matrix(2) <= X"076DC419";
matrix(3) <= X"0EDB8832";
matrix(4) <= X"1DB71064";
matrix(5) <= X"3B6E20C8";
matrix(6) <= X"76DC4190";
matrix(7) <= X"EDB88320";
begin
if ( rising_edge(CLK)) then
        case CALC is
                when '1' =>
                case c_state is
                when s0 =>
                        SEND <='0';
                crcb := crc( 7 downto 0 ) xor IN_BYTE;
                        crc:= X"00" & crc( 31 downto 8);
                        if ( crcb(0) = '1' ) then
                        crc:= crc xor matrix(0);
                        end if;
                        if ( crcb(1) = '1' ) then
                        crc:= crc xor matrix(1);
                        end if;
                        if ( crcb(2) = '1' ) then
                        crc:= crc xor matrix(2);
                        end if;
                        if ( crcb(3) = '1' ) then
                        crc:= crc xor matrix(3);
                        end if;
                        if ( crcb(4) = '1' ) then
                        crc:= crc xor matrix(4);
                        end if;
                        if ( crcb(5) = '1' ) then
                        crc:= crc xor matrix(5);
                        end if;
                        if ( crcb(6) = '1' ) then
                        crc:= crc xor matrix(6);
```

```
                end if;
                if ( crcb(7) = '1' ) then
                crc:= crc xor matrix(7);
                end if;
            when others => c_state <= s0;
            end case;
            when '0' =>
            end case;
            case SEND_RECEIVE is
                when '1' =>
                CRC32 <= not crc;
                SEND <= '1';
                when '0' =>
            end case;
        end if;
        end process;
end CRC32;
```

В данном случае изменена матрица предвычисленных значений и основной цикл расчёта CRC. Принцип взаимодействия и синхронизации блоков функциональной схемы остаётся таким же, как и при реализации табличного алгоритма.

Для реализации двухбайтового матричного алгоритма необходимо в функциональной схеме изменить 2 блока: main_vhdl и CRC32_vhdl. Основные изменения заключаются в увеличении разрядности шины данных до 16 бит, соединяющей управляющий блок main_vhdl и блок вычисления контрольной суммы CRC32_vhdl. Далее приведено описание блока вычисления контрольной суммы CRC32 на языке VHDL. Полное описание матрицы (matrix(0)…matrix(15)) приведено в [4].

```
entity CRC32_vhdl is
port(
        CLK    : in std_logic;
        IN_BYTES: in std_logic_vector(15 downto 0);
        CALC: in std_logic;
        SEND_RECEIVE: in std_logic;
        SHIFT: in std_logic;
        CRC32: out std_logic_vector (31 downto 0);
        SEND : out std_logic
);
end entity;
```

```
begin
matrix(0) <= X"191B3141";
matrix(1) <= X"32366282";
……..
matrix(14) <= X"76DC4190";
matrix(15) <= X"EDB88320";
begin
if ( rising_edge(CLK)) then
        case CALC is
                when '1' =>
                case SHIFT is
                when '0' =>
                crcb := crc( 15 downto 0 ) xor IN_BYTES;
                crc:= X"0000" & crc( 31 downto 16 );
                when '1' =>
                crcb:= crc( 15 downto 0 ) xor IN_BYTES;
                crcb := crcb ( 7 downto 0 ) & X"00";
                crc:= X"00" & crc (31 downto 8);
        end case;
                if ( crcb(0) = '1' ) then
                crc:= crc xor matrix(0);
                end if;
                if ( crcb(1) = '1' ) then
                crc:= crc xor matrix(1);
                end if;
                ……..
                if ( crcb(14) = '1' ) then
                crc:= crc xor matrix(14);
                end if;
                if ( crcb(15) = '1' ) then
                crc:= crc xor matrix(15);
                end if;
                when '0' =>
                end case;
        case SEND_RECEIVE is
```

```
                when '1' =>
                CRC32 <= not crc;
                SEND <= '1';
                STOP := '1';
                when '0' =>
                if ( STOP = '1' ) then
                 crc := X"FFFFFFFF";
                 STOP := '0';
                end if;
        end case;
        end if;
end process;
end CRC32;
```

На рис.2 приведена часть функциональной схемы вычисления контрольной суммы CRC32 для двухбайтового матричного алгоритма, отражающей изменения относительно схемы табличного алгоритма.

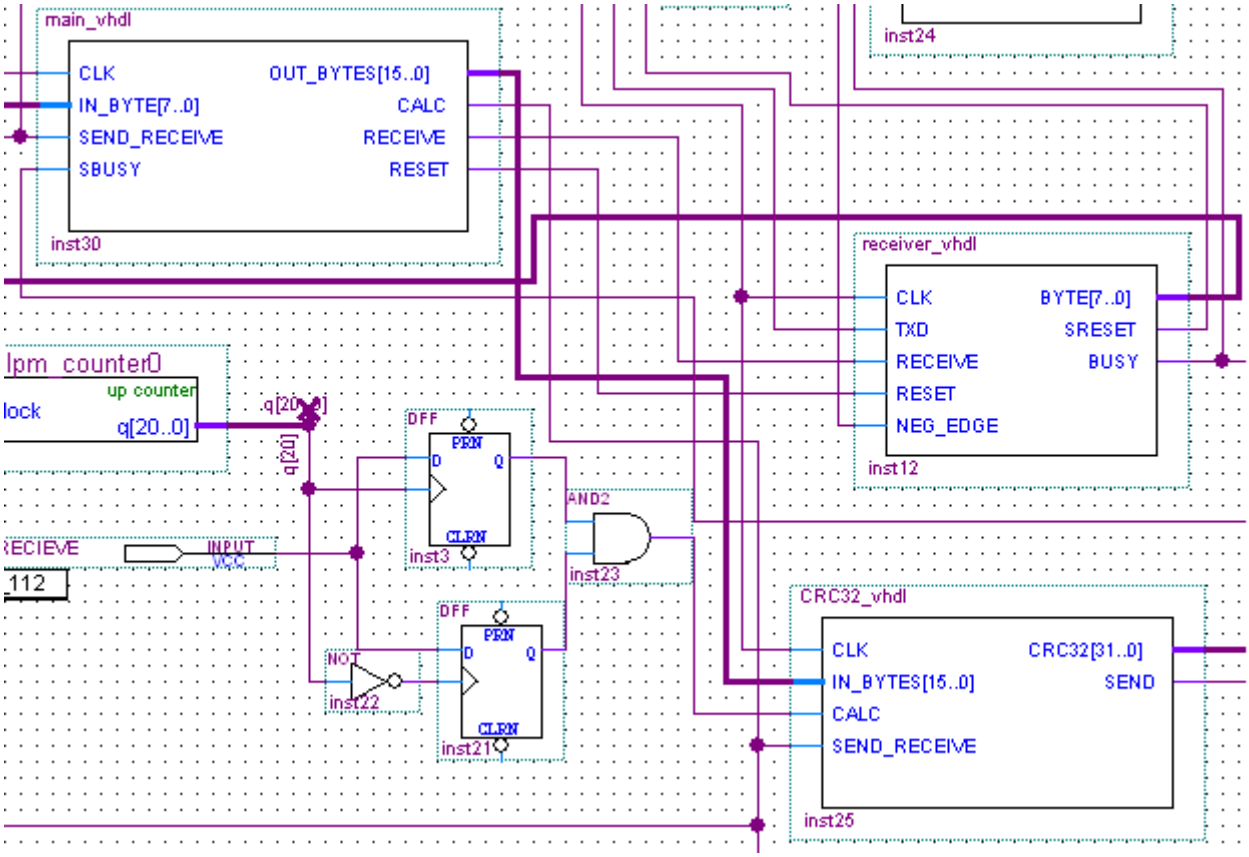

Рис. 2. Часть функциональной схемы для двухбайтового матричного алгоритма (разработано автором)

Для аппаратной реализации четырёхбайтового матричного алгоритма необходимо по аналогии с двухбайтовым матричным алгоритмом заменить управляющий блок и блок

вычисления CRC32 в связи с увеличением разрядности шины до 32 бит, а также изменить описание блока CRC32_vhdl согласно алгоритму. Полное описание матрицы (matrix(0)…matrix(31)) приведено в [4].

```
entity CRC32_vhdl is
port(
        CLK    : in std_logic;
        IN_BYTES: in std_logic_vector(31 downto 0);
        CALC: in std_logic;
        SEND_RECEIVE: in std_logic;
        SHIFT : in std_logic_vector ( 1 downto 0 );
        CRC32: out std_logic_vector (31 downto 0);
        SEND : out std_logic
);
end entity;
begin
matrix(0) <= X"B8BC6765";
matrix(1) <= X"AA09C88B";
……
matrix(30) <= X"76DC4190";
matrix(31) <= X"EDB88320";
begin
if ( rising_edge(CLK)) then
        case CALC is
        when '1' =>
        SEND <= '0';
        case SHIFT is
                when "00"=>
                crcb:= crc xor IN_BYTES;
                if ( crcb(0) = '1' ) then
                crc:= matrix(0);
                else
                crc := X"00000000";
                end if;
                when "01" =>
                crcb:= crc xor IN_BYTES ( 7 downto 0 );
                crcb := crcb ( 7 downto 0 ) & X"000000";
```

```
            crc:= X"00" & crc (31 downto 8);
            when "10" =>
            crcb:= crc xor IN_BYTES ( 15 downto 0 );
            crcb := crcb ( 15 downto 0) & X"0000";
            crc:= X"0000" & crc (31 downto 16);
            when "11" =>
            crcb:= crc xor IN_BYTES ( 23 downto 0 );
            crcb := crcb ( 23 downto 0) & X"00";
            crc:= X"000000" & crc (31 downto 24);
end case;
            if ( crcb(1) = '1' ) then
            crc:= crc xor matrix(1);
            end if;
            if ( crcb(2) = '1' ) then
            crc:= crc xor matrix(2);
            end if;
            ……….
            if ( crcb(30) = '1' ) then
            crc:= crc xor matrix(30);
            end if;
            if ( crcb(31) = '1' ) then
            crc:= crc xor matrix(31);
            end if;
            when '0' =>
            end case;
case SEND_RECEIVE is
            when '1' =>
                        CRC32 <= not crc;
                        SEND <= '1';
                        STOP := '1';
            when '0' =>
                        if ( STOP = '1' ) then
                         crc:= X"FFFFFFFF";
                         STOP := '0';
                        end if;
```

```
        end case;
        end if;
end process;
end CRC32;
```

На рис.3 приведена часть функциональной схемы вычисления контрольной суммы CRC32 для четырёхбайтового матричного алгоритма.

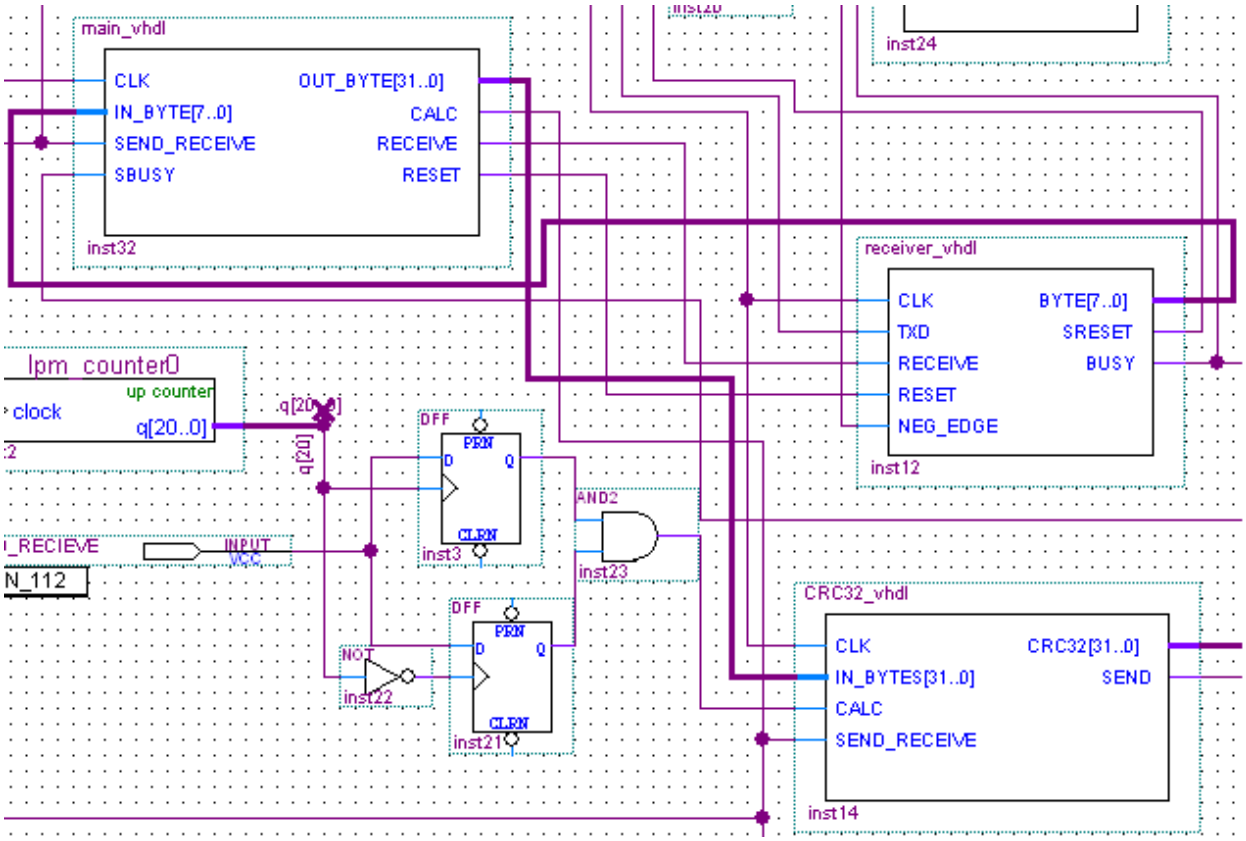

Рис. 3. *Часть функциональной схемы для четырёхбайтового матричного алгоритма (разработано автором)*

Таким образом, используя приведённые примеры аппаратной реализации, можно создавать аппаратуры для вычисления контрольной суммы CRC32.

ЗАКЛЮЧЕНИЕ

В данной статье приведены примеры аппаратных реализаций вычисления контрольной суммы CRC32 с применением ПЛИС Cyclone макета SDK 6.1 для табличного, матричного, двухбайтового и четырёхбайтового матричного алгоритмов. При проектировании функциональной схемы использовался блочно-ориентированный подход (BBD). Блоки функциональной схемы receiver_vhdl, main_vhdl, txd_neg_edge_vhdl, CRC32_vhdl, char_to_hex, lcd_out разработаны с применением языка описания аппаратуры VHDL. Показана практическая значимость приведённых примеров аппаратной реализации для применения их на ПЛИС Cyclone от Altera.

ЛИТЕРАТУРА

1. Бибило П.Н. Основы языка VHDL. – 3-е изд., доп. – М.: Издательство ЛКИ. 2007. – 328 с.
2. Мыцко Е.А., Мальчуков А.Н. Исследование аппаратных реализаций табличного и матричного алгоритмов вычисления CRC32 // Известия Томского политехнического университета. - 2013 - Т. 322 - №. 5. - С. 182-186.
3. Мыцко Е. А. , Мальчуков А. Н. Исследование аппаратной реализации алгоритмов вычисления контрольной суммы CRC32 [Электронный ресурс] // Вестник науки Сибири. - 2012 - №. 5 (6) - С. 82-86. - Режим доступа: http://sjs.tpu.ru/journal/article/view/512.
4. Мыцко Е. А., Мальчуков А. Н., Осокин А. Н. Контрольная сумма CRC. Исследование алгоритмов вычисления контрольной суммы CRC. - Saarbrucken: LAP LAMBERT Academic Publishing GmbH & Co. KG, 2013 - 180 с.
5. Мыцко Е.А., Мальчуков А.Н. Исследование программных реализаций табличного и матричного алгоритмов вычисления контрольной суммы CRC32 [Электронный ресурс] // Вестник науки Сибири. Серия: Информационные технологии и системы управления. – 2011 – №. 1 – С. 273-278. – URL: http://sjs.tpu.ru/journal/issue/view/2/ showToc/sect/4 (дата обращения 6.08.12).
6. Еремин В.В., Мальчуков А.Н. О применении блочно-ориентированного подхода к разработке устройств на ПЛИС [Электронный ресурс] // Вестник науки Сибири. Серия: Информационные технологии и системы управления. - 2011 - №. 1 - С. 379-381. - Режим доступа: http://sjs.tpu.ru/journal/issue/view/2/showToc/sect/4.
7. Мыцко Е.А., Мальчуков А.Н. Особенности аппаратной реализации алгоритмов вычисления контрольной суммы CRC32 [Электронный ресурс] // Вестник науки Сибири. - 2012 - №. 5 (6) - С. 87-92. - Режим доступа: http://sjs.tpu.ru/journal/article/view/513.
8. Учебный лабораторный стенд SDK-6.1 // Embedded systems. [Электронный ресурс]. URL: http://embedded.ifmo.ru/index.php/support/sdk-61 (дата обращения: 06.08.2012).
9. 32 bit Cyclic Redundancy Check Source Code for C++. // Create Window Website. 2011. URL: http://www.createwindow.com/programming/crc32/ (дата обращения: 01.04.2014).
10. Ross N.W. A Painless Guide to CRC Error Detection Algorithms. // Dr Ross Williams. 1993. URL: http://www.ross.net/crc/download/crc_v3.txt (дата обращения: 01.04.2014).

Рецензент: Ким Валерий Львович, д. т. н., профессор кафедры вычислительной техники ИК ТПУ.

Evgeniy Mytsko
FSAEI HE National Research Tomsk polytechnic university
Russia, Tomsk
E-Mail: evgenrus70@mail.ru

Hardware implementation examples for computing the cyclic redundancy check 32 checksum compatible with winrar, pkzip, Ethernet

Abstract. Some hardware implementation examples of algorithms for computing the checksum cyclic redundancy check 32 for a table, matrix, and the matrix of two-byte and four-byte algorithms were considered in the paper. The functional diagram of the cyclic redundancy check 32 hardware implementation for the table algorithm was developed using hardware description language Very high speed integrated circuits Hardware Description Language using block-oriented approach. Also the functional diagram of the hardware implementation for two-byte and four-byte matrix algorithms was shown. Examples of the hardware implementation source code described in the language of Very high speed integrated circuits Hardware Description Language for the table and matrix algorithms. Brief descriptions of the main functional blocks of the scheme and the main changes in the implementation of the functional blocks for matrix algorithms were shown. The principle of the cyclic redundancy check 32 checksum computation briefly was described for the data transmitted from the PC's serial port Recommended Standard 232 using the terminal for data transmission. The practical significance of these hardware implementation examples for their application programmable logic integrated circuits Cyclone from Altera was shown.

Keywords: checksum; table-driven algorithm; matrix-driven algorithm; cyclic redundancy check 32; hardware implementation; quartus; layout; serial port; data transmission; functional diagram.

Identification number of article 139TVN314

REFERENCES

1. Bibilo P.N. Osnovy jazyka VHDL. – 3-e izd., dop. – M.: Izdatel'stvo LKI. 2007. – 328 s.
2. Mycko E. A., Mal'chukov A. N. Issledovanie apparatnyh realizacij tablichnogo i matrichnogo algoritmov vychislenija CRC32 // Izvestija Tomskogo politehnicheskogo universiteta. - 2013 - T. 322 - №. 5. - C. 182-186.
3. Mycko E. A. , Mal'chukov A. N. Issledovanie apparatnoj realizacii algoritmov vychislenija kontrol'noj summy CRC32 [Jelektronnyj resurs] // Vestnik nauki Sibiri. - 2012 - №. 5 (6) - C. 82-86. - Rezhim dostupa: http://sjs.tpu.ru/journal/article/view/512.
4. Mycko E. A., Mal'chukov A. N., Osokin A. N. Kontrol'naja summa CRC. Issledovanie algoritmov vychislenija kontrol'noj summy CRC. - Saarbrucken: LAP LAMBERT Academic Publishing GmbH & Co. KG, 2013 - 180 c.
5. Mycko E.A., Mal'chukov A.N. Issledovanie programmnyh realizacij tablichnogo i matrichnogo algoritmov vychislenija kontrol'noj summy CRC32 [Jelektronnyj resurs] // Vestnik nauki Sibiri. Serija: Informacionnye tehnologii i sistemy upravlenija. – 2011 – №. 1 – C. 273-278. – URL: http://sjs.tpu.ru/journal/issue/view/2/ showToc/sect/4 (data obrashhenija 6.08.12).
6. Eremin V.V., Mal'chukov A.N. O primenenii blochno-orientirovannogo podhoda k razrabotke ustrojstv na PLIS [Jelektronnyj resurs] // Vestnik nauki Sibiri. Serija: Informacionnye tehnologii i sistemy upravlenija. - 2011 - №. 1 - C. 379-381. - Rezhim dostupa: http://sjs.tpu.ru/journal/issue/view/2/showToc/sect/4.
7. Mycko E. A. , Mal'chukov A. N. Osobennosti apparatnoj realizacii algoritmov vychislenija kontrol'noj summy CRC32 [Jelektronnyj resurs] // Vestnik nauki Sibiri. - 2012 - №. 5 (6) - C. 87-92. - Rezhim dostupa: http://sjs.tpu.ru/journal/article/view/513.
8. Uchebnyj laboratornyj stend SDK-6.1 // Embedded systems. [Jelektronnyj resurs]. URL: http://embedded.ifmo.ru/index.php/support/sdk-61 (data obrashhenija: 06.08.2012).
9. 32 bit Cyclic Redundancy Check Source Code for C++. // Create Window Website. 2011. URL: http://www.createwindow.com/programming/crc32/ (data obrashhenija: 01.04.2014).
10. Ross N.W. A Painless Guide to CRC Error Detection Algorithms. // Dr Ross Williams. 1993. URL: http://www.ross.net/crc/download/crc_v3.txt (data obrashhenija: 01.04.2014).