Қазақстан Республикасы Тәуелсіздігінің 30 жылдығына арналған «Сейфуллин оқулары – 17: «Қазіргі аграрлық ғылым: цифрлық трансформация» атты халықаралық ғылыми – тәжірибелік конференцияға материалдар = Материалы международной научно – теоретической конференции «Сейфуллинские чтения – 17: «Современная аграрная наука: цифровая трансформация», посвященной 30 – летию Независимости Республики Казахстан.- 2021.- Т.1, Ч.3 - С. 66 - 69

КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ ДАННЫХ С ПОМОЩЬЮ РЕФЛЕКСНОГО КОДА

Шоханова Т.И.,
Хамзина Б.Е.,
Жолдангарова Г.И.

Данная статья посвящена исследованию кодирования и декодирования данных. Автор предлагает использование рефлексного кода как один из способов защиты информации. Основным результатам данного исследования является аппаратно-программная реализация алгоритмов кодирование и декодирование данных с помощью рефлексного кода на полнофункциональной отладочной платы DL-Zybo Zynq-7000.

Ключевые слова: кодирование, декодирование, кодер, декодер, рефлексный код, код Грея, двоичный код, VHDL.

В нашей современной жизни главной ценностью является информация, с помощью которой возможно реализовать управление практически всеми сферами жизни. На сегодняшний день информационные ресурсы стали намного важнее, чем материальная составляющая либо энергетические возможности. Но они подвержены различным негативным явлениям, таких как несанкционированный доступ к секретной информации, промышленный шпионаж и другое. Для обеспечения безопасности и надежности данных осуществляется кодирование информации.

Использование кодирование последовательных сообщений в двоичном коде может привести к ошибкам неоднозначности. Использование рефлексного кода помогает решить данную проблему. Особенностью этого кода является то, что соседние кодовые слова будут отличаться только в одном разряде, т.е. нет одновременного изменения цифр в нескольких разрядах. С целью разработки одного из способов защиты информации, как кодирование и декодирование данных с помощью рефлексного кода, нами было проведено исследование.

Рефлексный код принадлежит к классу кодов Грея (Grey Code), представляющих собою циклические двоичные коды, последовательные значения которых отличаются друг от друга только одним двоичным разрядом. Применение данного кода дает возможность повысить вдвое разрешающую способность аналого-кодового преобразователя по сравнению

с обычным двоичным кодом. Код Грея имеет как преимущества, так и недостатки. Главным преимуществом являются то, что при переходе от слова к слову изменение значения элемента в каждой категории вдвое меньше, чем в простом коде и это позволяет обеспечить более высокую точность кодирования при той же скорости схемы кодирования. Также в коде Грея можно выделить ось симметрии («ось отражения»), которая связана с наблюдением идентичности некоторых элементов разряда. Одним из недостатков является то, что в коде Грея вес единиц не определяется номером разряда. Преобразование двоичных чисел в коды Грея происходит за счет последовательного циклического обхода всех ячеек диаграммы Вейча.

Пусть n-разрядное число X задано в двоичной системе счисления:

$$X = x_{n-1} \dots x_p \dots x_1 x_0, \quad (1)$$

где $x_p = 0$ или1 – значения разрядов числа X, x_0 - младший разряд, p = 0, 1, …, n-1. Правило кодирования всех возможных 2^n значений n – разрядных двоичных чисел X, заключающееся в приписывании им кодовых комбинаций $A(X) = \propto_{n-1} \dots \propto_p \dots \propto_1 \propto_0$ при

$$\propto_p = \begin{cases} x_p \oplus x_{p+1}, \text{если } p = 0,1, \dots, n-2, \\ x_{n-1} \oplus 0 = x_{n-1}, \text{если } p = n-1, \end{cases} \quad (2)$$

определяет алгоритм построения кода Грея, т. е. два различных двоичных числа X соответствуют различным значениям A(X), а числам X и X + 1 соотвествуют кодовым комбинациям A(X) и A(X+1), которые отличаются значением только одного разряда α_p. [1]

Преобразование данных согласно правилу (2) является линейным преобразованием, где используется только логическая операция сумма по модулю два. Эти преобразования могут быть использованы для кодирования и декодирования данных, так как каждое линейное преобразование имеет свое обратное линейное преобразование. Исходные данные обеспечиваются последовательным включением кодера и декодера.

Разработка кодера и декодера происходит на платформе САПР Vivado, который имеет встроенный логический симулятор ISIM. Данная программа представляет высокоуровневый синтез с набором инструментов, где происходит преобразование кода в программируемую логику. Интерфейс основан на подходе, протестированном в IDE Plan Ahead, и в основном фокусируется на характеристиках проекта и анализе планирования топологии. Этот стиль проектирования позволяет сосредоточиться на решении основных проблем, возникающих при работе с ПЛИС. Помимо этого, программа после завершения разработки модели выдает техническую

документацию, оптимизацию устройства по временным характеристикам, потребляемой мощности и ресурсам ПЛИС.

В связи с постоянным усовершенствованием технологий производства интегральных схем возрастает значимость языков описания аппаратуры. Они позволяют разрабатывать и верифицировать цифровые устройства на верхних уровнях до преобразования в логический. Два наиболее популярных языка описания аппаратуры являются VHDL и Verilog. Язык Verilog изначально создавался для решения задач моделирования, а VHDL создавался как язык описания с учетом необходимости моделирования. Он также реализует методологию нисходящего проектирования, в которой система сначала описывается на высоком уровне и тестируется с помощью средств моделирования, после чего поэтапно приводится к структурному описанию, тесно связанному с фактической аппаратной реализацией. Поэтому данный язык подходит для разработки кодера и декодера. [2, 3]

С помощью программы САПР Vivado был разработан код преобразователя двоичного кода в код Грей:

```
entity binar2gray is
Port (x0,x1,x2,x3: in std_logic;
y3:out std_logic; y0,y1,y2:inout std_logic);
end binar2gray;
architecture Behavioral of binar2gray is
component xor2
  Port (a, b: in std_logic;
        y: out std_logic);
end component;
signal a0,a1,a2,a3: std_logic;
begin
gate1: xor2 port map(a=>x0,b=>x1,y=>a0);
gate2: xor2 port map(a=>x1,b=>x2,y=>a1);
gate3: xor2 port map(a=>x2,b=>x3,y=>a2);
a3<=x3;
gate4: xor2 port map(a=>a0,b=>y1,y=>y0);
gate5: xor2 port map(a=>a1,b=>y2,y=>y1);
gate6: xor2 port map(a=>a2,b=>a3,y=>y2);
y3<=a3;
end Behavioral;
```

Далее необходимо перейти в окно процессов. После нажатия на вкладку Synthesize – XST появляется пункт Check Syntax, где проверяется наше VHDL-описание на наличие ошибок. При завершении в строке появляется знак зеленой галочки.

Теперь необходимо проверить с помощью временной диаграммы правильность работы VHDL-модели преобразователя. Для этого был составлен алгоритм описания тестирующей программы, где указываем время включения каждого входа кодера:

```
signal bin,g,bin_out : std_logic_vector(3 downto 0) := (others => '0');
```

```
begin
  uut1: bin2gray port map (bin => bin, g => g);
 uut2: gray2bin port map (g => g,
       bin => bin_out)
  -- stimulus process
  stim_proc: process
  begin
     bin <= "0000";    wait for 10 ns;
     bin <= "0001";  wait for 10 ns;
     bin <= "0010";  wait for 10 ns;
     bin <= "0011";  wait for 10 ns;
     bin <= "0100";    wait for 10 ns;
     bin <= "0101";  wait for 10 ns;
     bin <= "0110";  wait for 10 ns;
     bin <= "0111";  wait for 10 ns;
     bin <= "1000";    wait for 10 ns;
     bin <= "1001";  wait for 10 ns;
     bin <= "1010";  wait for 10 ns;
     bin <= "1011";  wait for 10 ns;
     bin <= "1100";    wait for 10 ns;
     bin <= "1101";  wait for 10 ns;
     bin <= "1110";  wait for 10 ns;
     bin <= "1111";  wait for 10 ns;
    wait;
  end process;
```

Далее в окне исходных модулей в строке View необходимо выбрать режим Simulaton. Посредством нажатия на строку Simulate Behavioral Model появляется окно модуля ISim Simulator (рис. 1), в рабочей области которого наблюдается временная диаграмма. Временная диаграмма показывает время начала и окончания каждого сигнала и его длительность.

Значения данных на портах показывает, что каждая последующая комбинация отличается от предыдущей только в одном разряде. Данное свойство позволяет минимизировать ошибки кодирования.

Разработанный кодер и декодер был протестирован на плате DL-ZyBoZynq-7000 тип кристалла (SoC) ZYNQ XC7Z2010-1CLG400C фирмы Xilinx. Данная плата отличается сочетанием в себе двухъядерный ARM Cortex-A9 процессор и ПЛИС Xilinx 7-ой серии. ZYBO совместим с новым высокопроизводительным пакетом Vivado Design Suite от Xilinx, а также с набором инструментов ISE / EDK. [4]

Исследование рефлексного кода показало, что кодирование данных с помощью рефлексного кода позволяет минимизировать ошибки неоднозначности, которые возникают при кодирование последовательных сообщений. Применение данного кода дает возможность повысить вдвое разрешающую способность кодера благодаря тому, что при переходе от

слова к слову изменение значения элемента в каждом разряде вдвое меньше, чем в простом коде.

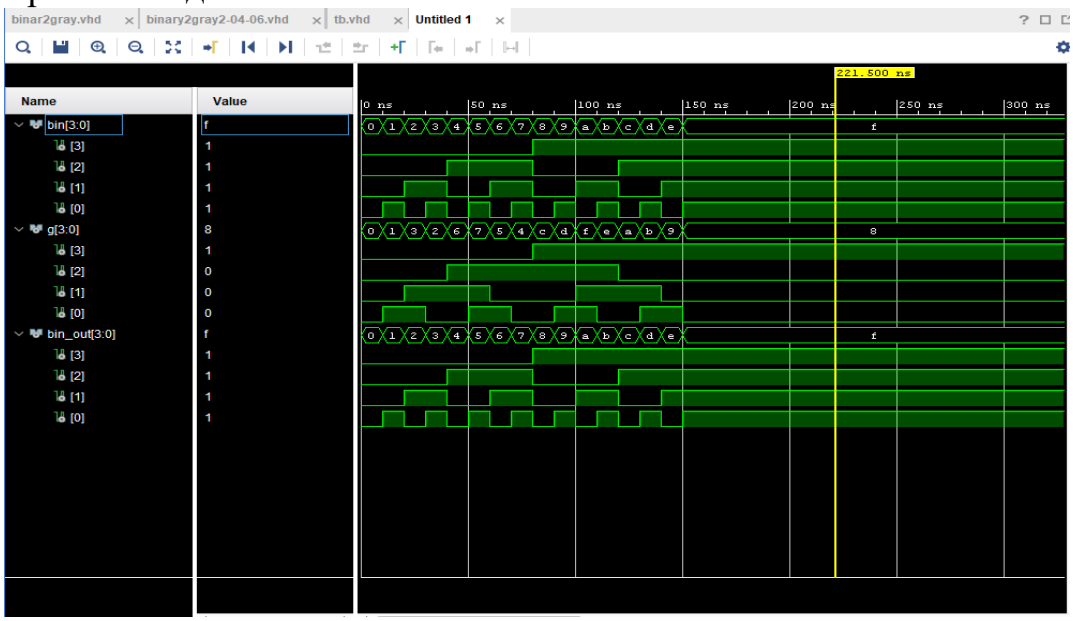

Рисунок 1. Временная диаграмма binar2gray

Список литературы

1. Тарасов И.Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. Изд. 2-е.- Издательство «Горячая Линия - Телеком», 2015. – 538 с.

2. Бибило П.Н. Основы языка VHDL / П.Н. Бибило. – 6-е изд. – Москва: URSS: ЛИБРОКОМ, 2014. – 328 с.

3. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. Изд. 4-е, исп.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2017. – 1104 с.

4. She J.Du P. FPGA-Based Motion Estimation Algorithm Optimization. Microprocessors and Microsystems. Режим доступа: https://www.scopus.com/. Дата обращения: 08.12.2020.