Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Автоматизация проектирования вычислительных машин и систем

ОТЧЁТ
к лабораторной работе № 3
на тему

# РАЗРАБОТКА ТЕСТОВЫХ МОДУЛЕЙ НА ЯЗЫКЕ VHDL В СРЕДЕ VIVADO. ВАРИАНТ № 12

Студенты: гр. 050502
Т.С. Жук
А.В. Крачковский
Е.В. Кравченко

Проверил: В.В. Шеменков

Минск 2023

**СОДЕРЖАНИЕ**

## 1 ЦЕЛЬ РАБОТЫ

Приобрести навыки разработки тестовых модулей на языке VHDL.

## 2 ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

Для выполнения работы используем САПР Vivado 2018.2 и редактор кода Visual Studio Code с расширением (extension) Modern VHDL для подсветки синтаксиса языка VHDL, язык программирования C++ и инструмент сборки Qbs.

В соответствии с полученным вариантом задания необходимо проверить функционирование разрабатываемого устройства путем подачи на его вход тестовых воздействий и анализа результатов моделирования его работы следующим образом:

1. Для комбинационного устройства, разработанного в лабораторной работе №1 (вариант с логическими операциями и параллельным оператором безусловного присваивания), создать два варианта тестовых модулей со встроенной проверкой корректности работы устройства:

– подаваемые на разработанное устройство, и эталонные результаты работы устройства должны считываться тестовым модулем из текстового файла;

– входные тестовые воздействия могут формироваться любым способом, а для формирования эталонных воздействий в качестве эталона необходимо использовать любой другой вариант описания этого же устройства из лабораторной работы №1.

2. Для функционального узла последовательного типа, разработанного в лабораторной работе №2, создать тестовый модуль с автоматической проверкой корректности работы устройства. Тестовое воздействие должно проверять работу устройства во всех его режимах. В случае обнаружения ошибки в работе устройства необходимо выдать сообщение на консоль программы моделирования.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать модуль генерации входных и эталонных тестовых воздействий для тестируемыых устройств;
2. Разработать модули тестирования комбинационного устройства;
3. Провести автоматическую проверку корректности работы комбинационного устройства;
4. Разработать модуль тестирования функционального узла последовательного типа;
5. Провести автоматическую проверку корректности работы функционального узла последовательного типа.

## 3 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В лабораторной работе №1 необходимо было разработать селектор/мультиплексор для двух четырёхразрядных шин (QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS) согласно логической диаграмме представленной на рисунке ниже.

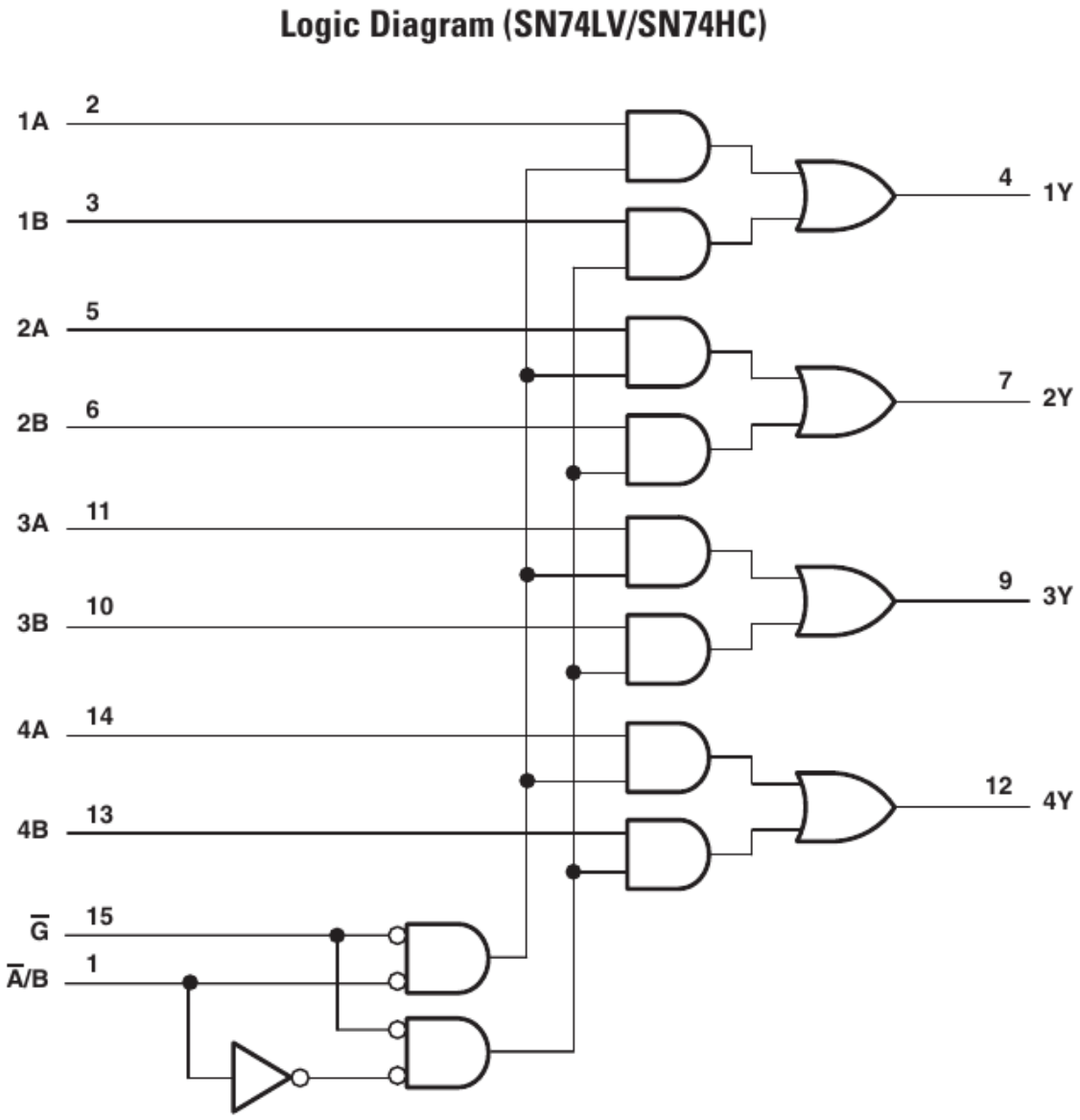


Рисунок 3.1 – Логическая диаграмма реализованного комбинационного устройства

В лабораторной работе №2 необходимо было разработать синхронный 4-х разрядный десятичный счётчик (SYNCHRONOUS 4-BIT UP/DOWN DECADE COUNTERS WITH 3-STATE OUTPUTS) согласно логической диаграмме представленной на рисунке ниже.

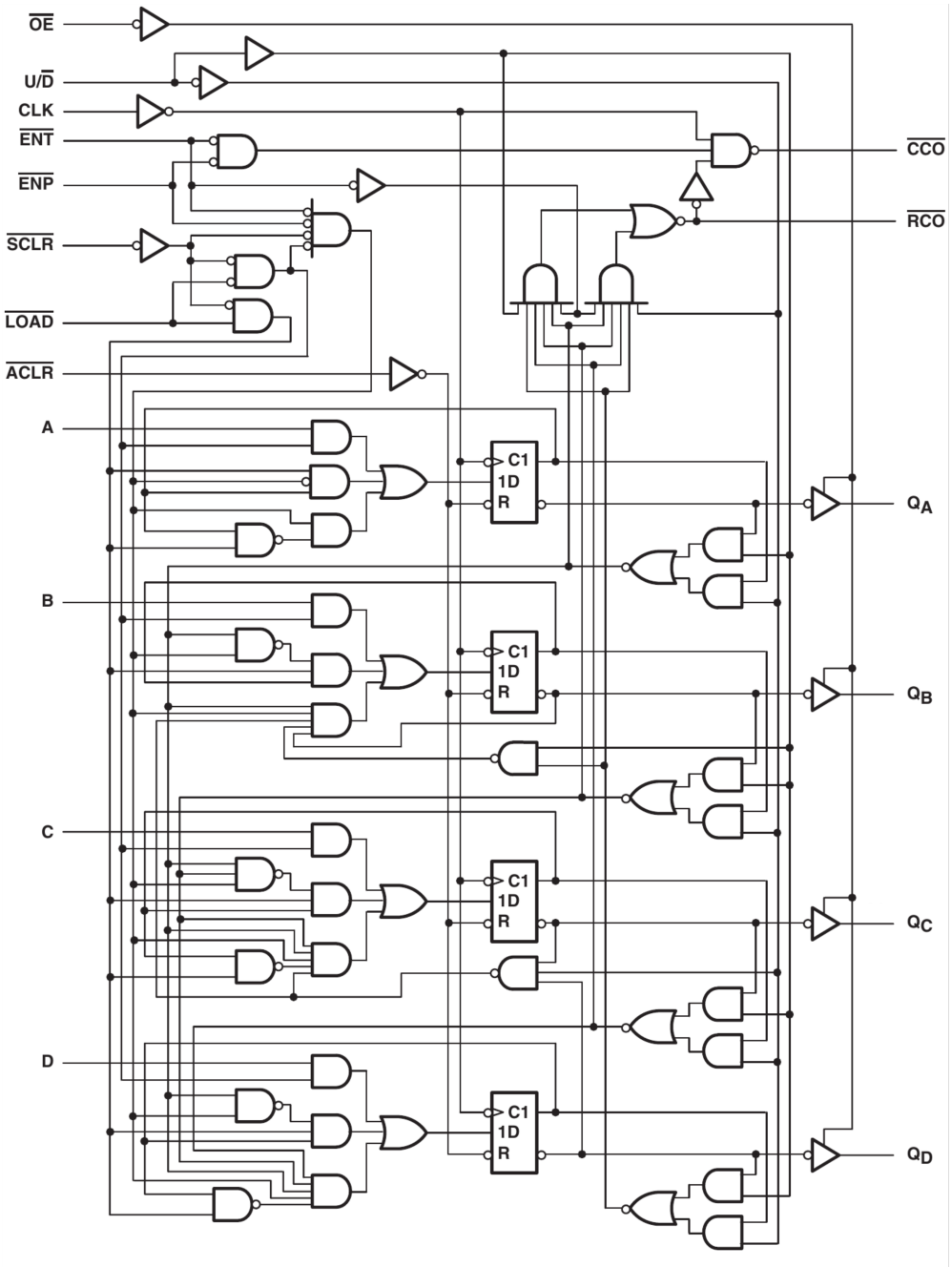


Рисунок 3.2 – Логическая диаграмма реализованного функционального узла последовательного типа

Функциональная таблица реализованного комбинационного устройства приведена ниже.

**FUNCTION TABLE (SN74)**

| INPUTS | | | | OUTPUT |
|---|---|---|---|---|
| STROBE | SELECT | A | B | |
| H | X | X | X | L |
| L | L | L | X | L |
| L | L | H | X | H |
| L | H | X | L | L |
| L | H | X | H | H |

Рисунок 3.3 – Функциональная таблица реализованного комбинационного устройства

Функциональная таблица реализованного функционального узла последовательного типа приведена ниже.

**FUNCTION TABLE**

| INPUTS | | | | | | | | OPERATION |
|---|---|---|---|---|---|---|---|---|
| $\overline{OE}$ | $\overline{ACLR}$ | $\overline{SCLR}$ | $\overline{LOAD}$ | $\overline{ENT}$ | $\overline{ENP}$ | U/$\overline{D}$ | CLK | |
| H | X | X | X | X | X | X | X | Q outputs disabled |
| L | L | X | X | X | X | X | X | Asynchronous clear |
| L | H | L | X | X | X | X | ↑ | Synchronous clear |
| L | H | H | L | X | X | X | ↑ | Load |
| L | H | H | H | L | L | H | ↑ | Count up |
| L | H | H | H | L | L | L | ↑ | Count down |
| L | H | H | H | H | X | X | X | Inhibit count |
| L | H | H | H | X | H | X | X | Inhibit count |

Рисунок 3.4 – Функциональная таблица реализованного функционального узла последовательного типа

# 4 ВЫПОЛНЕНИЕ РАБОТЫ

Для выполнения лабораторной работы создаём проект с помощью встроенного менеджера проектов в САПР Vivado. С помощью графического интерфейса в режиме диалоговых окон создаём и добавляем в проект исходные файлы:

1. ‘SN74.vhd’ – с описанием разработанного в лабораторной работе №1 комбинационного устройства;
2. ‘SN74_TestBench_0.vhd’ – с описанием тестового модуля для комбинационного устройства (первый метод);
3. ‘SN74_TestBench_1.vhd’ – с описанием тестового модуля для комбинационного устройства (второй метод);
4. ‘DFlipFlop.vhd’ – с описанием триггера, используемого в функциональном узле последовательного типа;
5. ‘Counter.vhd’ – с описанием разработанного в лабораторной работе №2 функционального узла последовательного типа;
6. ‘Counter_TestBench.vhd’ – с описанием тестового модуля для функционального узла последовательного типа.

Дальнейшее внесение изменений в исходные файлы осуществляем в редакторе кода Visual Studio Code, который предоставляет удобный и привычный интерфейс и богатые возможности по редактированию кода.

Все файлы описывающие устройства или их составляющие взяты из предыдущих лабораторных работ либо вовсе без изменений, либо с минимальными исправлениями.

Исходный текст разработанных модулей представлен в приложениях с Б по Ж.

## 4.1 Разработка модуля генерации входных и эталонных тестовых воздействий для тестируемыых устройств

Для генерации входных и эталонных тестовых воздействий для тестируемыых устройств нами был выбран язык C++, для сборки проекта – система сборки Qbs. Главная причина такого выбора: гибкость и высокая скорость разработки проектов на данном языке программирования.

Модуль содержит следующие исходные файлы:

1. ‘TestBench3.qbs’ – файл (сборки) проекта;
2. ‘src/std.hpp’ – файл с глобальными определениями типов (‘std_logic’, ‘std_logic_vector’) и повсеместно используемых функций (‘operator <<’ для типа ‘std_logic’, ‘vectorize’, ‘numerate’);
3. ‘src/main.cpp’ – файл с функцией ‘main’, запускающей генерацию тестовых воздействий;

4. ‘src/TestBench/SN74.hpp’ – файл с непосредственной генерацией тестовых воздействий комбинационного устройства;

5. ‘src/TestBench/Counter.hpp’ – файл с непосредственной генерацией тестовых воздействий функционального узла последовательного типа;

6. ‘src/Emulate/SN74.hpp’ – файл с эмуляцией работы комбинационного устройства;

7. ‘src/Emulate/Counter.hpp’ – файл с эмуляцией работы функционального узла последовательного типа.

Исходный текст разработанного модуля представлен в приложении А.

Результатом работы данного модуля являются файлы ‘SN74.txt’ и ‘Counter.txt’, содержащие тестовые воздействия:

Файл ‘SN74.txt’ (первые 3 строки):

```
D           S           A3          A2          A1          A0
            B3          B2          B1          B0          Y3
            Y2          Y1          Y0
false false false false false false false false false false false false
    false false false false false false false false false false
   false false false false false
false false false false false false false false false false false false
    false false false false false false false true  false false false
   false false false false false
```

Файл ‘Counter.txt’ (первые 3 строки):

```
OE           ACLR         SCLR         LOAD         ENP          ENT
            U/D          CLK          D3           D2           D1
   D0           Q3           Q2           Q1           Q0           RCO
            CCO          [D] [QV]
false false false false false true  false true  false true  false true
    false true  false false false true  false true  false true  false
   true  false false false false false false false false false true
   false true  (0) (15)
false false false false false true  false true  false true  false true
    false true  false false false true  false true  false true  false
   true  false false false false false false false false false true
   false true  (0) (15)
```

Данные воздействия используются модулями тестирования для проверки работы реализованных устройств. Всвязи с особенностями хранения тестовых воздействий (приложение А, файл ‘src/std.hpp’, строки 19–27, функция ‘operator <<’) всем тестовым модулям потребуется использовать функцию преобразования прочитанных воздействий в значения типа ‘std_logic’:

```
function to_std_logic(H, L: boolean) return std_logic is begin
    if H then
        if L then return 'Z';
        else      return 'X';
```

```
        end if;
    else
        if L then return '1';
        else      return '0';
        end if;
    end if;
end function;
```

## 4.2 Разработка модулей тестирования комбинационного устройства

Согласно заданию на лабораторную работу первая архитектура (вариант с логическими операциями и параллельным оператором безусловного присваивания) должна быть протестирована двумя методами. В первом методе как входные тестовые воздействия, так и эталонные будут считываться из файла 'SN74.txt', генерируемого модулем 'TestBench3'. Во втором методе входные тестовые воздействия будут считываться из файла, а в качестве эталонных воздействий будут выступать сигналы, генерируемые всеми остальными архитектурами.

### 4.2.1 Разработка модулей тестирования комбинационного устройства. Первый метод

Тестирование комбинационного устройства определяется в операторе 'process':

```
process
    file input_file: text;
    variable line_content: line;
    variable D_h, S_h, A3_h, A2_h, A1_h, A0_h, B3_h, B2_h, B1_h, B0_h,
       Y3_h, Y2_h, Y1_h, Y0_h: boolean;
    variable D_l, S_l, A3_l, A2_l, A1_l, A0_l, B3_l, B2_l, B1_l, B0_l,
       Y3_l, Y2_l, Y1_l, Y0_l: boolean;
begin
    ...
```

Непосредственное тестирование осуществляется в операторе 'while ... loop':

```
while not endfile(input_file) loop
    -- Read a line from the file
    readline(input_file, line_content);
    -- Parse this line
    read(line_content, D_h);  read(line_content, D_l);
    read(line_content, S_h);  read(line_content, S_l);
    read(line_content, A3_h); read(line_content, A3_l);
    ...
    read(line_content, A0_h); read(line_content, A0_l);
```

```
    read(line_content, B3_h); read(line_content, B3_l);
    ...
    read(line_content, B0_h); read(line_content, B0_l);
    read(line_content, Y3_h); read(line_content, Y3_l);
    ...
    read(line_content, Y0_h); read(line_content, Y0_l);
    -- Set the input signals to the values read from the file
    D <= to_std_logic(D_h, D_l);
    S <= to_std_logic(S_h, S_l);
    A <= to_std_logic(A3_h, A3_l) & to_std_logic(A2_h, A2_l) &
       to_std_logic(A1_h, A1_l) & to_std_logic(A0_h, A0_l);
    B <= to_std_logic(B3_h, B3_l) & to_std_logic(B2_h, B2_l) &
       to_std_logic(B1_h, B1_l) & to_std_logic(B0_h, B0_l);
    -- Check if the output value matches the expected value
    wait for 1ps;
    assert Y_a1 = to_std_logic(Y3_h, Y3_l) & to_std_logic(Y2_h, Y2_l) &
        to_std_logic(Y1_h, Y1_l) & to_std_logic(Y0_h, Y0_l)
        report ”Mismatch detected for Y_a1” severity error;
    wait for DELAY;
end loop;
```

### 4.2.2 Разработка модулей тестирования комбинационного устройства. Второй метод

Тестирование комбинационного устройства определяется в операторе ‘process’:

```
process
    file input_file: text;
    variable line_content: line;
    variable D_h, S_h, A3_h, A2_h, A1_h, A0_h, B3_h, B2_h, B1_h, B0_h:
       boolean;
    variable D_l, S_l, A3_l, A2_l, A1_l, A0_l, B3_l, B2_l, B1_l, B0_l:
       boolean;
begin
    ...
```

Непосредственное тестирование осуществляется в операторе ‘while ... loop’:

```
while not endfile(input_file) loop
    -- Read a line from the file
    readline(input_file, line_content);
    -- Parse this line
    read(line_content, D_h);  read(line_content, D_l);
    read(line_content, S_h);  read(line_content, S_l);
    read(line_content, A3_h); read(line_content, A3_l);
    ...
    read(line_content, A0_h); read(line_content, A0_l);
    read(line_content, B3_h); read(line_content, B3_l);
    ...
    read(line_content, B0_h); read(line_content, B0_l);
```

```
    -- Set the input signals to the values read from the file
    D <= to_std_logic(D_h, D_l);
    S <= to_std_logic(S_h, S_l);
    A <= to_std_logic(A3_h, A3_l) & to_std_logic(A2_h, A2_l) &
       to_std_logic(A1_h, A1_l) & to_std_logic(A0_h, A0_l);
    B <= to_std_logic(B3_h, B3_l) & to_std_logic(B2_h, B2_l) &
       to_std_logic(B1_h, B1_l) & to_std_logic(B0_h, B0_l);
    -- Check if the output value matches the expected value
    wait for 1ps;
    assert Y_a1 = Y_a2 and Y_a1 = Y_a3 and Y_a1 = Y_a4 and Y_a1 = Y_a5
        report "Mismatch detected for Y_a1" severity error;
    wait for DELAY;
end loop;
```

## 4.3 Проведение автоматической проверки корректности работы комбинационного устройства

Результаты проведения автоматической проверки корректности работы комбинационного устройства представлены на рисунках ниже.

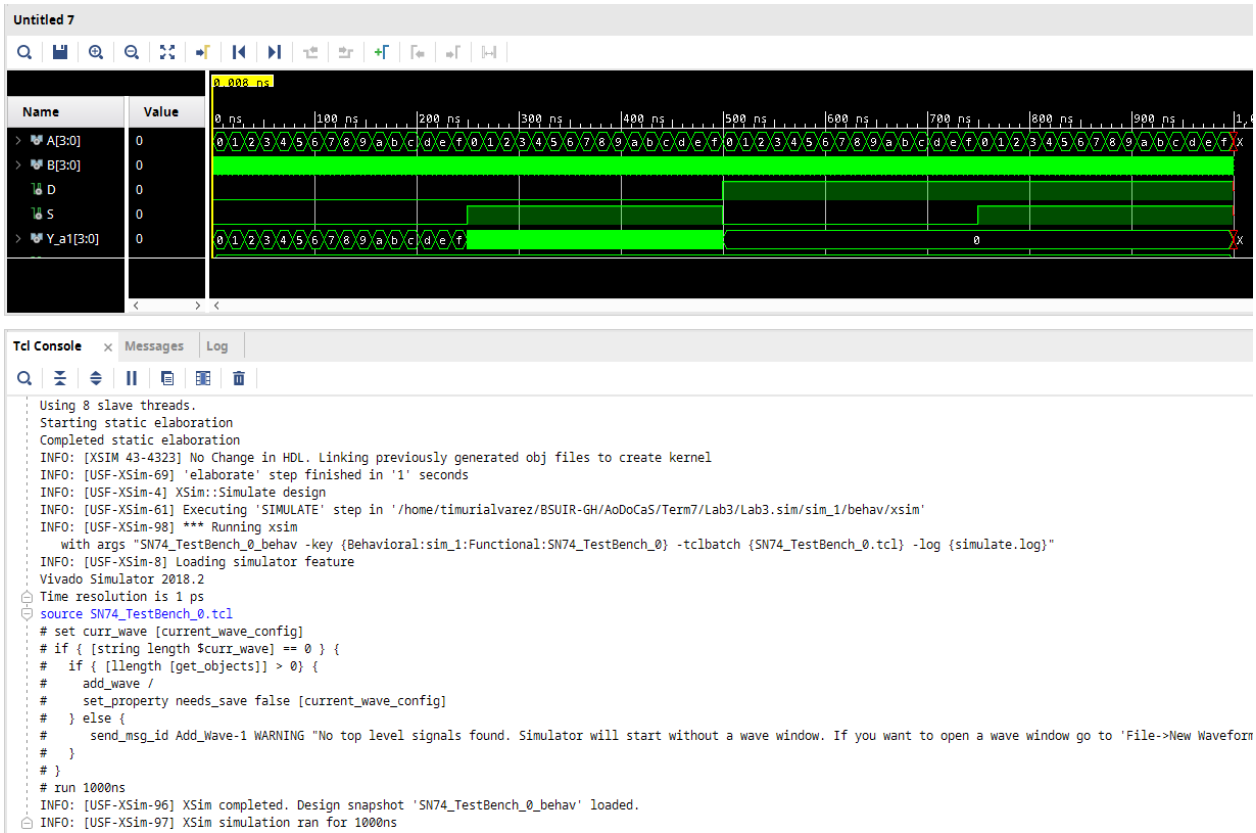


Рисунок 4.1 – Веренная диаграмма работы реализованного функционального комбинационного устройства с содержимым консоли. Первый метод

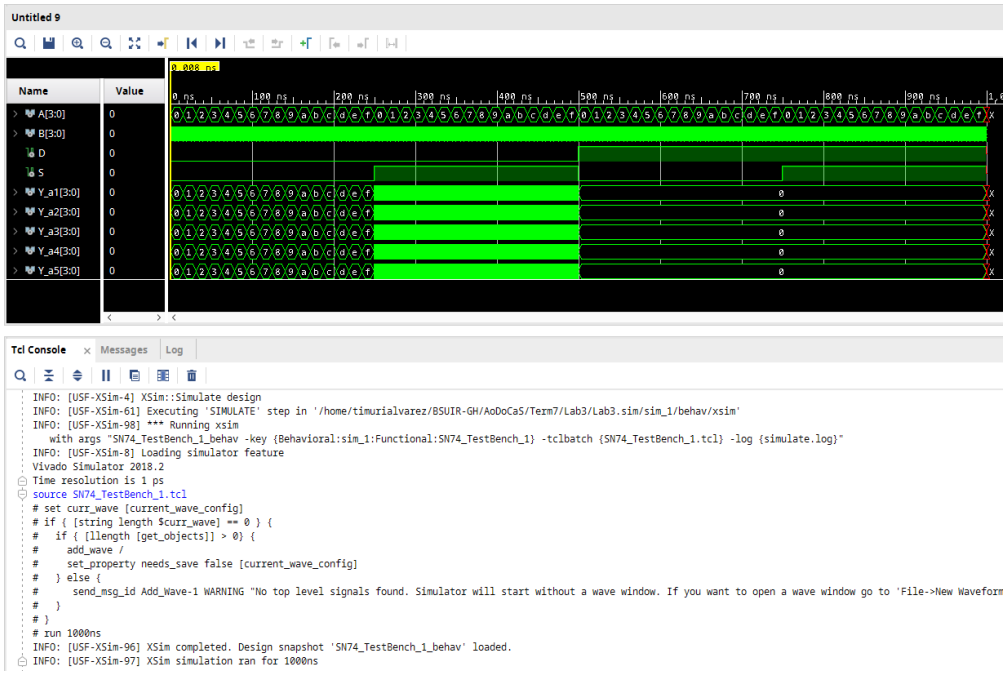


Рисунок 4.2 – Веренная диаграмма работы реализованного функционального комбинационного устройства с содержимым консоли. Второй метод

## 4.4 Разработка модуля тестирования функционального узла последовательного типа

Тестирование функционального узла последовательного типа определяется в операторе ‘process’:

```
process
    file input_file: text;
    variable line_content: line;
    variable OE_h, ACLR_h, SCLR_h, LOAD_h, ENP_h, ENT_h, UD_h, CLK_h,
       D3_h, D2_h, D1_h, D0_h, Q3_h, Q2_h, Q1_h, Q0_h, RCO_h, CCO_h:
       boolean;
    variable OE_l, ACLR_l, SCLR_l, LOAD_l, ENP_l, ENT_l, UD_l, CLK_l,
       D3_l, D2_l, D1_l, D0_l, Q3_l, Q2_l, Q1_l, Q0_l, RCO_l, CCO_l:
       boolean;
begin
    ...
```

Непосредственное тестирование осуществляется в операторе ‘while ... loop’:

```
while not endfile(input_file) loop
    -- Read a line from the file
    readline(input_file, line_content);
    -- Parse this line
    read(line_content, OE_h);   read(line_content, OE_l);
    read(line_content, ACLR_h); read(line_content, ACLR_l);
    ...
```

```
    read(line_content, UD_h);   read(line_content, UD_l);
    read(line_content, CLK_h);  read(line_content, CLK_l);
    read(line_content, D3_h);   read(line_content, D3_l);
    ...
    read(line_content, D0_h);   read(line_content, D0_l);
    read(line_content, Q3_h);   read(line_content, Q3_l);
    ...
    read(line_content, Q0_h);   read(line_content, Q0_l);
    read(line_content, RCO_h);  read(line_content, RCO_l);
    read(line_content, CCO_h);  read(line_content, CCO_l);
    -- Set the input signals to the values read from the file
    UUT_OE   <= to_std_logic(OE_h, OE_l);
    UUT_ACLR <= to_std_logic(ACLR_h, ACLR_l);
    ...
    UUT_UD   <= to_std_logic(UD_h, UD_l);
    UUT_CLK  <= to_std_logic(CLK_h, CLK_l);
    UUT_D    <= to_std_logic(D3_h, D3_l) & to_std_logic(D2_h, D2_l) &
       to_std_logic(D1_h, D1_l) & to_std_logic(D0_h, D0_l);
    -- Check if the output value matches the expected value
    wait for 1ps;
    assert UUT_Q = to_std_logic(Q3_h, Q3_l) & to_std_logic(Q2_h, Q2_l)
       & to_std_logic(Q1_h, Q1_l) & to_std_logic(Q0_h, Q0_l)
        report "Mismatch detected for Q" severity error;
    assert UUT_RCO = to_std_logic(RCO_h, RCO_l)
        report "Mismatch detected for RCO" severity error;
    assert UUT_CCO = to_std_logic(CCO_h, CCO_l)
        report "Mismatch detected for CCO" severity error;
    wait for DELAY;
end loop;
```

## 4.5 Проведение автоматической проверки корректности работы функционального узла последовательного типа

Результаты проведения автоматической проверки корректности работы функционального узла последовательного типа представлены на рисунке ниже.

Заметим, что на данном рисунке заметна ошибка: в конце моделирования (все входные тестовые воздействия принимают значение ’X’), ожидается (эталонное воздействие) такое же значение на выходах функционального узла, однако на выходе можно наблюдать значение ’Z’. Вероятнее всего это как-то связано с вычислением значений типа ‘std_logic’ в VHDL, однако мы не берёмся утверждать это. В любом случае к непосредственной работе устройства эта ошибка не имеет отношения, поэтому в её исправление опционально.

Чтобы исправить данную ошибку нужно прибегнуть к одному из следующих сопсобов:

– Изменить логику модуля ‘Counter’ – принудительно обработать ходные значения ’X’;

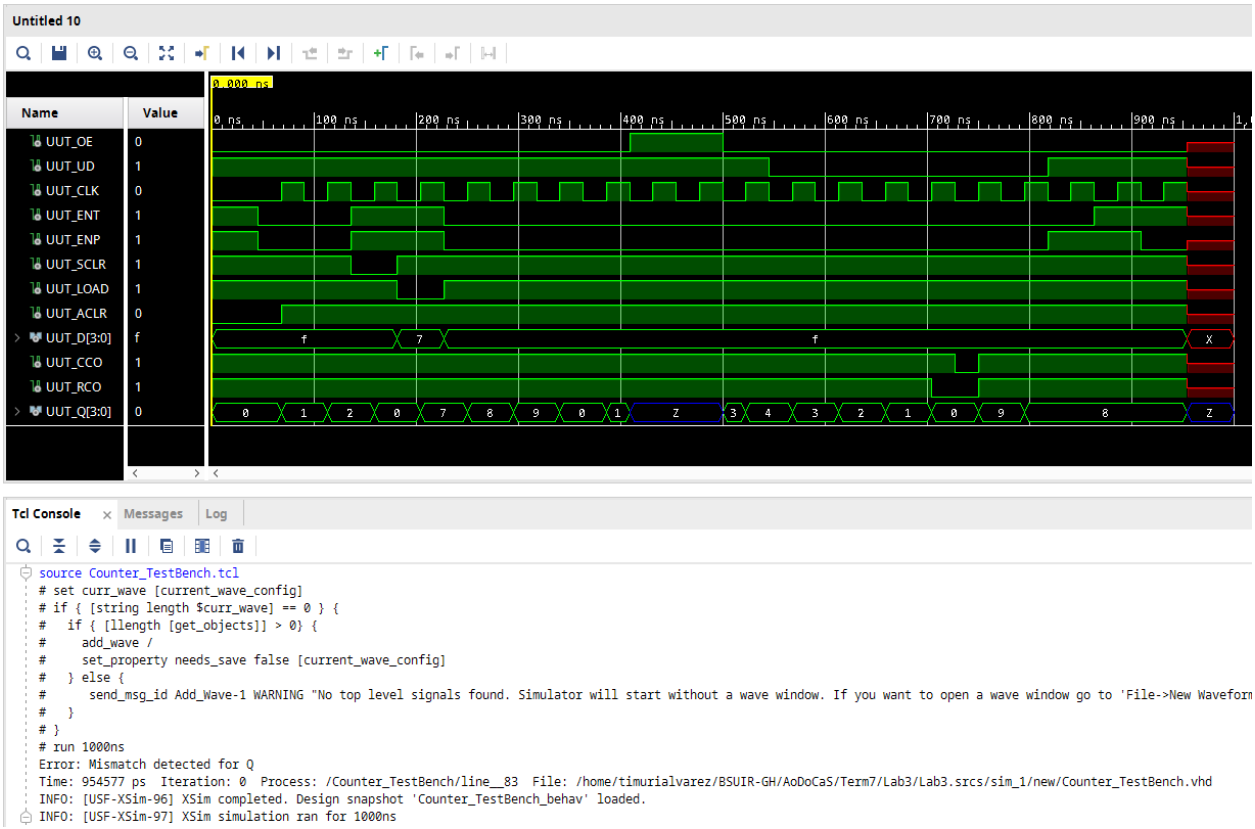


Рисунок 4.3 – Веренная диаграмма работы реализованного функционального узла последовательного типа с содержимым консоли

– Внести изменения в логику модуля ‘TestBench3’ – в методе ‘auto x(void) -> std::string’ класса ‘Counter’ заменить ожидаемое значние на выходе устройства с ‘X’ на ‘Z’.

За каждым из этих вариантов стоит определённое обоснование, поэтому конечное решение должно приниматься в соответствии с целями проекта и задачами, которые решает модуль ‘Counter’.

# 5 ВЫВОДЫ

В процессе выполнения работы мы приобрели навыки разработки тестовых модулей на языке VHDL.

Полученные навыки были применены для решения задач, возникших в ходе работы:

1. Разработки модуля генерации входных и эталонных тестовых воздействий для тестируемыых устройств;
2. Разработки модулей тестирования комбинационного устройства;
3. Проведения автоматической проверки корректности работы комбинационного устройства;
4. Разработки модуля тестирования функционального узла последовательного типа;
5. Проведения автоматической проверки корректности работы функционального узла последовательного типа.

**ПРИЛОЖЕНИЕ А**

(*обязательное)*

Исходный текст модуля TestBench3

Файл ‘TestBench3.qbs’ содержит следующий код:

```
import qbs

CppApplication {
    Depends {
        name: ”cpp”
    }
    cpp.cxxLanguageVersion: ”c++20”
    cpp.includePaths: [”src”]
    name: ”TestBench3”
    files: [”src/*.cpp”]
}
```

Файл ‘src/std.hpp’ содержит следующий код:

```
#pragma once

#include <array>
#include <sstream>
#include <fstream>

using usize = size_t;
static constexpr const usize N = 4;

typedef enum {
    L,
    H,
    U,
    X,
    Z,
} std_logic;
using std_logic_vector = std::array<std_logic, N>;

auto operator <<(std::ostream& stream, const std_logic bit) -> std::
   ostream& {
    switch (bit) {
        case  L: return stream << ”false false”;
        case  H: return stream << ”false true ”;
        case  X: return stream << ”true  false”;
        case  Z: return stream << ”true  true ”;
        default: return stream << ”error error”;
    };
}

auto vectorize(const usize value) -> std_logic_vector {
    if (value >= (1 << N)) throw ”Invalid value”;
    std_logic_vector vector;
    for (usize i = 0; i < N; ++i)
        vector[i] = static_cast<std_logic>((value & (1 << i)) != 0);
    return vector;
}

auto numerate(const std_logic_vector vector) -> usize {
```

```
38      usize value = 0;
39      for (usize i = 0; i < N; ++i) {
40          const std_logic bit = vector[N - 1 - i];
41          if (bit != H && bit != L)
42              throw ”Invalid bit value”;
43          value <<= 1;
44          value |= (bit == H) ? 1 : 0;
45      }
46      return value;
47  }
```

Файл ‘src/main.cpp’ содержит следующий код:

```
 1  #include <TestBench/SN74.hpp>
 2  #include <TestBench/Counter.hpp>
 3
 4  auto main(void) -> int {
 5      TestBenchSN74();
 6      TestBenchCounter();
 7      return 0;
 8  }
```

Файл ‘src/TestBench/SN74.hpp’ содержит следующий код:

```
 1  #pragma once
 2
 3  #include <Emulation/SN74.hpp>
 4
 5  auto TestBenchSN74(void) -> void {
 6      SN74 device;
 7      std::ofstream log;
 8      log.open(”./tests/SN74.txt”);
 9      log << device.header();
10      for (std_logic int_d : {L, H})
11          for (std_logic int_s : {L, H})
12              for (usize int_a = 0; int_a < (1 << N); ++int_a)
13                  for (usize int_b = 0; int_b < (1 << N); ++int_b) {
14                      device.D = int_d;
15                      device.S = int_s;
16                      device.A = vectorize(int_a);
17                      device.B = vectorize(int_b);
18                      log << device.snapshot();
19                  }
20      log << device.x();
21      log.close();
22  }
```

Файл ‘src/TestBench/Counter.hpp’ содержит следующий код:

```
 1  #pragma once
 2
 3  #include <Emulation/Counter.hpp>
 4
 5  static constexpr const std_logic IDC = H;
```

```
 6
 7  auto TestBenchCounter(void) -> void {
 8      Counter device;
 9      std::ofstream log;
10      log.open("./tests/Counter.txt");
11      log << device.header();
12      // BEGIN TestBench
13      // Async clear (3 CLK)
14      device.OE = L; device.ACLR = L;
15      device.SCLR = IDC; device.LOAD = IDC;
16      device.ENP = IDC; device.ENT = IDC;
17      device.UD = IDC; device.D.fill(IDC);
18      for (usize n = 1; n <= 3; ++n) {
19          if (n == 3) {
20              device.ENP = L;
21              device.ENT = L;
22          }
23          device.CLK = L; log << device.snapshot();
24      }
25      // Count up (4 CLK)
26      device.OE = L; device.ACLR = H;
27      device.SCLR = H; device.LOAD = H;
28      device.UD = H; device.D.fill(IDC);
29      device.CLK = H; log << device.snapshot();
30      device.CLK = L; log << device.snapshot();
31      device.CLK = H; log << device.snapshot();
32      device.SCLR = L; device.LOAD = H;
33      device.ENP = IDC; device.ENT = IDC;
34      device.CLK = L; log << device.snapshot();
35      // Sync clear (2 CLK)
36      device.CLK = H; log << device.snapshot();
37      device.SCLR = H; device.LOAD = L;
38      device.UD = H; device.D = vectorize(7);
39      device.CLK = L; log << device.snapshot();
40      // Sync load (2 CLK)
41      device.CLK = H; log << device.snapshot();
42      device.SCLR = H; device.LOAD = H;
43      device.ENP = L; device.ENT = L;
44      device.UD = H; device.D.fill(IDC);
45      device.CLK = L; log << device.snapshot();
46      // Count up (13 CLK)
47          // UUT_OE   <= '0'; UUT_ACLR <= '1';
48      device.OE = L; device.ACLR = H;
49      device.SCLR = H; device.LOAD = H;
50      device.ENP = L; device.ENT = L;
51      device.UD = H; device.D.fill(IDC);
52      for (usize n = 1; n <= 7; ++n) {
53          device.CLK = H; log << device.snapshot();
54          if (n == 4) {
55              device.OE = H; device.ACLR = H;
56          } else if (n == 6) {
57              device.OE = L; device.ACLR = H;
58          }
59          if (n != 7) {
```

```
60              device.CLK = L; log << device.snapshot();
61          }
62      }
63      // Count down (12 CLK)
64      device.UD = L; device.D.fill(IDC);
65      for (usize n = 1; n <= 6; ++n) {
66          device.CLK = L; log << device.snapshot();
67          device.CLK = H; log << device.snapshot();
68      }
69      // Inhibit counting (6 CLK)
70      device.UD = IDC; device.D.fill(IDC);
71      for (usize n = 1; n <= 3; ++n) {
72          device.ENP = (n == 3) ? L : H;
73          device.ENT = (n == 1) ? L : H;
74          device.CLK = L; log << device.snapshot();
75          device.CLK = H; log << device.snapshot();
76      }
77      // Force unknown state
78      log << device.x();
79      // END TestBench
80      log.close();
81  }
```

Файл ‘src/Emulation/SN74.hpp’ содержит следующий код:

```
 1  #pragma once
 2
 3  #include <std.hpp>
 4
 5  class SN74 {
 6  public:
 7      std_logic D, S;
 8      std_logic_vector A, B;
 9  public:
10      SN74(void) : D(U), S(U) {
11          A.fill(U);
12          B.fill(U);
13      }
14  public:
15      auto Y(void) const -> std_logic_vector {
16          std_logic_vector Y;
17          if (D) Y.fill(L);
18          else switch(S) {
19              case L: Y = A; break;
20              case H: Y = B; break;
21              default: Y.fill(Z); break;
22          }
23          return Y;
24      }
25      auto header(void) const -> std::string_view {
26          return ”D           S           A3          A2          A1
                        A0          B3          B2          B1          B0
                        Y3          Y2          Y1          Y0         \n”;
27      }
```

```
28      auto snapshot(void) const -> std::string {
29          const std_logic_vector Y = this->Y();
30          auto buffer = std::stringstream();
31          buffer << D << ' ' << S;
32          for (usize i = 0; i < N; ++i)
33              buffer << ' ' << A[N - 1 - i];
34          for (usize i = 0; i < N; ++i)
35              buffer << ' ' << B[N - 1 - i];
36          for (usize i = 0; i < N; ++i)
37              buffer << ' ' << Y[N - 1 - i];
38          buffer << '\n';
39          return buffer.str();
40      }
41      auto x(void) const -> std::string {
42          auto buffer = std::stringstream();
43          buffer << X << ' ' << X;
44          for (usize i = 0; i < N; ++i)
45              buffer << ' ' << X;
46          for (usize i = 0; i < N; ++i)
47              buffer << ' ' << X;
48          for (usize i = 0; i < N; ++i)
49              buffer << ' ' << X;
50          buffer << '\n';
51          return buffer.str();
52      }
53  };
```

Файл ‘src/Emulation/Counter.hpp’ содержит следующий код:

```
 1  #pragma once
 2
 3  #include <std.hpp>
 4
 5  class Counter {
 6  public:
 7      std_logic OE, ACLR, SCLR, LOAD, ENP, ENT, UD, CLK;
 8      std_logic_vector D;
 9  private:
10      std_logic CLK_OLD;
11      std_logic_vector QV;
12  public:
13      Counter(void) : OE(U), ACLR(U), SCLR(U), LOAD(U), ENP(U), ENT(U),
           UD(U), CLK(U), D({}), CLK_OLD(U) {
14          D.fill(U);
15          QV.fill(U);
16      }
17  private:
18      auto front(void) const -> bool {
19          return CLK_OLD == L && CLK == H;
20      }
21  public:
22  //
    |-----------------------------------------------------------------------|
```

```
23 // |                            Function table
                                |
24 //
      |----|------|------|------|-----|-----|-----|-----|--------------------|

25 // | __ | ____ | ____ | ____ | ___ | ___ |   _ |     |
                          |
26 // | OE | ACLR | SCLR | LOAD | ENT | ENP | U/D | CLK | Operation
                 |
27 //
      |----|------|------|------|-----|-----|-----|-----|--------------------|

28 // | H  |  X   |  X   |  X   |  X  |  X  |  X  |  X  | Q outputs
      disabled |
29 // | L  |  L   |  X   |  X   |  X  |  X  |  X  |  X  | Asynchronous
      clear |
30 // | L  |  H   |  L   |  X   |  X  |  X  |  X  | ↑   | Synchronous
      clear  |
31 // | L  |  H   |  H   |  L   |  X  |  X  |  X  | ↑   | Load
                       |
32 // | L  |  H   |  H   |  H   |  L  |  L  |  H  | ↑   | Count up
                  |
33 // | L  |  H   |  H   |  H   |  L  |  L  |  L  | ↑   | Count down
                 |
34 // | L  |  H   |  H   |  H   |  H  |  X  |  X  |  X  | Inhibit count
                |
35 // | L  |  H   |  H   |  H   |  X  |  H  |  X  |  X  | Inhibit count
                |
36 //
      |----|------|------|------|-----|-----|-----|-----|--------------------|

37      auto Q(void) -> std_logic_vector {
38          // Asynchronous clear
39          if (ACLR == L) {
40              QV.fill(L);
41          }
42          // Synchronous clear
43          if (ACLR == H && SCLR == L && front()) {
44              QV.fill(L);
45          }
46          // Load
47          if (ACLR == H && SCLR == H && LOAD == L && front()) {
48              QV = D;
49          }
50          // Inhibit count else
51          if (ACLR == H && SCLR == H && LOAD == H && ENT == L && ENP == L
                && front()) {
52              usize current_value = numerate(QV);
53              if (UD == H) {
54                  if (current_value != 9) ++current_value;
55                  else current_value = 0;
56              }
57              if (UD == L) {
58                  if (current_value != 0) --current_value;
```

```
59                  else current_value = 9;
60              }
61              QV = vectorize(current_value);
62          }
63          // Save CLK
64          CLK_OLD = CLK;
65          // Output
66          std_logic_vector Q = QV;
67          if (OE == H)
68              Q.fill(Z);
69          return Q;
70      }
71      auto RCO(void) const -> std_logic {
72          return static_cast<std_logic>(!(ACLR == H && SCLR == H && LOAD
               == H && ENT == L && ENP == L && UD == L && numerate(QV) ==
               0));
73      }
74      auto CCO(void) const -> std_logic {
75          return static_cast<std_logic>(!(!static_cast<bool>(RCO()) &&
               CLK == L));
76      }
77      auto header(void) const -> std::string_view {
78          return "OE          ACLR        SCLR        LOAD        ENP
                       ENT         U/D         CLK         D3          D2
                        D1          D0          Q3          Q2          Q1
                        Q0          RCO         CCO         [D] [QV]\n";
79      }
80      auto snapshot(void) -> std::string {
81          const std_logic_vector Q = this->Q();
82          auto buffer = std::stringstream();
83          buffer << OE << ' ' << ACLR << ' '
84              << SCLR << ' ' << LOAD << ' '
85              << ENP << ' ' << ENT << ' '
86              << UD << ' ' << CLK;
87          for (usize i = 0; i < N; ++i)
88              buffer << ' ' << D[N - 1 - i];
89          for (usize i = 0; i < N; ++i)
90              buffer << ' ' << Q[N - 1 - i];
91          buffer << ' ' << RCO() << ' ' << CCO() << " (" << numerate(QV)
               << ") (" << numerate(D) << ")\n";
92          return buffer.str();
93      }
94      auto x(void) -> std::string {
95          auto buffer = std::stringstream();
96          buffer << X << ' ' << X << ' '
97              << X << ' ' << X << ' '
98              << X << ' ' << X << ' '
99              << X << ' ' << X;
100         for (usize i = 0; i < N; ++i)
101             buffer << ' ' << X;
102         for (usize i = 0; i < N; ++i)
103             buffer << ' ' << X;
104         buffer << ' ' << X << ' ' << X << " (X) (X)\n";
105         return buffer.str();
```

```
106      }
107  };
```

**ПРИЛОЖЕНИЕ Б**

(*обязательное)*

Исходный текст модуля SN74

Файл ‘SN74.vhd’ содержит следующий код:

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 16.09.2023 17:37:05
6  -- Design Name:
7  -- Module Name: SN74 - architecture1..5
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------


20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Category: QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS
26 -- Implementation: SN74LV/SN74HC
27 entity SN74 is
28     generic (
29         -- Width of quadruple lines
30         WIDTH: positive
31     );
32     port (
33         -- Input quadruple lines
34         A: in  std_logic_vector(WIDTH-1 downto 0);
35         B: in  std_logic_vector(WIDTH-1 downto 0);
36         -- Disable/Enable (not G)
37         D: in  std_logic;
38         -- Select (not A/B)
39         S: in  std_logic;
40         -- Output quadruple line
41         Y: out std_logic_vector(WIDTH-1 downto 0)
42     );
43 end entity;
44
45 -- Function table (SN74)
46 -- |---|---|---|---||---|
47 -- | D | S | A | B || Y |
48 -- |---|---|---|---||---|
49 -- | H | X | X | X || L |
50 -- | L | L | L | X || L |
```

```
51  -- | L | L | H | X || H |
52  -- | L | H | X | L || L |
53  -- | L | H | X | H || H |
54  -- |---|---|---|---||---|
55
56  -- Implementation 1
57  -- Parallel unconditional assignment is required (+ logical operators)
58  architecture architecture1 of SN74 is
59      -- Internal signal
60      signal Enable:  std_logic_vector(WIDTH-1 downto 0) := (others =>
            '0');
61      signal SelectB: std_logic_vector(WIDTH-1 downto 0) := (others =>
            '0');
62  begin
63      Enable  <= (others => not D);
64      SelectB <= (others => S);
65      Y       <= Enable and ((A and not SelectB) or (B and SelectB));
66  end architecture;
67
68  -- Implementation 2
69  -- Parallel conditional assignment is required
70  architecture architecture2 of SN74 is
71      -- Internal signals
72      signal DS: std_logic_vector(1 downto 0) := (others => '0');
73  begin
74      DS <= D & S;
75      Y  <= A               when DS = "00" else
76            B               when DS = "01" else
77            (others => '0') when DS = "10" else
78            (others => '0') when DS = "11" else
79            (others => 'X');
80  end architecture;
81
82  -- Implementation 3
83  -- Parallel selected assignment is required
84  architecture architecture3 of SN74 is
85      -- Internal signals
86      signal DS: std_logic_vector(1 downto 0) := (others => '0');
87  begin
88      DS <= D & S;
89      with DS select
90          Y <= A               when "00",
91               B               when "01",
92               (others => '0') when "10",
93               (others => '0') when "11",
94               (others => 'X') when others;
95  end architecture;
96
97  -- Implementation 4
98  -- Sequential conditional is required (if)
99  architecture architecture4 of SN74 is
100     -- Internal signals
101     signal DS: std_logic_vector(1 downto 0) := (others => '0');
102     signal ALARM: std_logic := '0';
```

```
103 begin
104     DS <= D & S;
105     ALARM <= '1' when D = 'X' or S = 'X' else '0';
106     process (A, B, D, S, ALARM) begin
107         if    DS = "00" then Y <= A;
108         elsif DS = "01" then Y <= B;
109         elsif DS = "10" then Y <= (others => '0');
110         elsif DS = "11" then Y <= (others => '0');
111         else                 Y <= (others => 'X');
112         end if;
113     end process;
114 end architecture;
115 
116 -- Implementation 5
117 -- Sequential selection operator is required (case)
118 architecture architecture5 of SN74 is
119     -- Internal signals
120     signal DS: std_logic_vector(1 downto 0) := (others => '0');
121     signal ALARM: std_logic := '0';
122 begin
123     DS <= D & S;
124     ALARM <= '1' when D = 'X' or S = 'X' else '0';
125     process (A, B, D, S, ALARM) begin
126         case DS is
127             when "00"   => Y <= A;
128             when "01"   => Y <= B;
129             when "10"   => Y <= (others => '0');
130             when "11"   => Y <= (others => '0');
131             when others => Y <= (others => 'X');
132         end case;
133     end process;
134 end architecture;
```

**ПРИЛОЖЕНИЕ В**

(*обязательное)*

Исходный текст модуля SN74_TestBench_0

Файл ‘SN74_TestBench_0.vhd’ содержит следующий код:

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 21.10.2023 10:14:23
6  -- Design Name:
7  -- Module Name: SN74_TestBench_0 - run
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------


20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use std.textio.all;
25 use ieee.std_logic_textio.all;
26
27 entity SN74_TestBench_0 is end entity;
28
29 architecture run of SN74_TestBench_0 is
30     -- Constants
31     constant WIDTH:      positive := 4;
32     constant ITERATIONS: positive := 2 ** (2 * WIDTH + 2);
33     constant EXTRA_ITER: positive := 1;
34     constant DELAY:      time     := 1000000 ps / (ITERATIONS +
          EXTRA_ITER);
35     -- Signals
36     signal A:    std_logic_vector(WIDTH-1 downto 0) := (others => '0');
37     signal B:    std_logic_vector(WIDTH-1 downto 0) := (others => '0');
38     signal D:    std_logic                          := '0';
39     signal S:    std_logic                          := '0';
40     signal Y_a1: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
41     -- Utility functions
42     function to_std_logic(H, L: boolean) return std_logic is begin
43         if H then
44             if L then return 'Z';
45             else      return 'X';
46             end if;
47         else
48             if L then return '1';
49             else      return '0';
```

```
50              end if;
51          end if;
52      end function;
53  begin
54      -- Instantiate unit under test (UUT)
55      uut1: entity work.SN74(architecture1)
56          generic map (WIDTH => WIDTH)
57          port map (A => A, B => B, D => D, S => S, Y => Y_a1);
58      -- Testing process
59      process
60          file input_file: text;
61          variable line_content: line;
62          variable D_h, S_h, A3_h, A2_h, A1_h, A0_h, B3_h, B2_h, B1_h,
               B0_h, Y3_h, Y2_h, Y1_h, Y0_h: boolean;
63          variable D_l, S_l, A3_l, A2_l, A1_l, A0_l, B3_l, B2_l, B1_l,
               B0_l, Y3_l, Y2_l, Y1_l, Y0_l: boolean;
64      begin
65          -- Open the input file
66          file_open(input_file, "/home/timurialvarez/BSUIR-GH/AoDoCaS/
               Term7/TestBench3/tests/SN74.txt", read_mode);
67          -- Read and ignore the header line
68          readline(input_file, line_content);
69          -- Read data from the file and simulate the entity
70          while not endfile(input_file) loop
71              -- Read a line from the file
72              readline(input_file, line_content);
73              -- Parse this line
74              read(line_content, D_h);  read(line_content, D_l);
75              read(line_content, S_h);  read(line_content, S_l);
76              read(line_content, A3_h); read(line_content, A3_l);
77              read(line_content, A2_h); read(line_content, A2_l);
78              read(line_content, A1_h); read(line_content, A1_l);
79              read(line_content, A0_h); read(line_content, A0_l);
80              read(line_content, B3_h); read(line_content, B3_l);
81              read(line_content, B2_h); read(line_content, B2_l);
82              read(line_content, B1_h); read(line_content, B1_l);
83              read(line_content, B0_h); read(line_content, B0_l);
84              read(line_content, Y3_h); read(line_content, Y3_l);
85              read(line_content, Y2_h); read(line_content, Y2_l);
86              read(line_content, Y1_h); read(line_content, Y1_l);
87              read(line_content, Y0_h); read(line_content, Y0_l);
88              -- Set the input signals to the values read from the file
89              D <= to_std_logic(D_h, D_l);
90              S <= to_std_logic(S_h, S_l);
91              A <= to_std_logic(A3_h, A3_l) & to_std_logic(A2_h, A2_l) &
                   to_std_logic(A1_h, A1_l) & to_std_logic(A0_h, A0_l);
92              B <= to_std_logic(B3_h, B3_l) & to_std_logic(B2_h, B2_l) &
                   to_std_logic(B1_h, B1_l) & to_std_logic(B0_h, B0_l);
93              -- Check if the output value matches the expected value
94              wait for 1ps;
95              assert Y_a1 = to_std_logic(Y3_h, Y3_l) & to_std_logic(Y2_h,
                    Y2_l) & to_std_logic(Y1_h, Y1_l) & to_std_logic(Y0_h,
                   Y0_l)
96                  report "Mismatch detected for Y_a1" severity error;
```

```
 97             wait for DELAY;
 98         end loop;
 99         -- Close the input file
100         file_close(input_file);
101         -- Wait for simulation to end
102         wait;
103     end process;
104 end architecture;
```

**ПРИЛОЖЕНИЕ Г**

(*обязательное)*

Исходный текст модуля SN74_TestBench_1

Файл ‘SN74_TestBench_1.vhd’ содержит следующий код:

```
----------------------------------------------------------------------------------

-- Company:
-- Engineer:
--
-- Create Date: 21.10.2023 10:14:23
-- Design Name:
-- Module Name: SN74_TestBench_1 - run
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use std.textio.all;
use ieee.std_logic_textio.all;

entity SN74_TestBench_1 is end entity;

architecture run of SN74_TestBench_1 is
    -- Constants
    constant WIDTH:      positive := 4;
    constant ITERATIONS: positive := 2 ** (2 * WIDTH + 2);
    constant EXTRA_ITER: positive := 1;
    constant DELAY:      time     := 1000000 ps / (ITERATIONS +
       EXTRA_ITER);
    -- Signals
    signal A:    std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal B:    std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal D:    std_logic                          := '0';
    signal S:    std_logic                          := '0';
    signal Y_a1: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal Y_a2: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal Y_a3: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal Y_a4: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    signal Y_a5: std_logic_vector(WIDTH-1 downto 0) := (others => '0');
    -- Utility functions
    function to_std_logic(H, L: boolean) return std_logic is begin
        if H then
            if L then return 'Z';
            else      return 'X';
```

```
50              end if;
51          else
52              if L then return '1';
53              else      return '0';
54              end if;
55          end if;
56      end function;
57  begin
58      -- Instantiate unit under test (UUT)
59      uut1: entity work.SN74(architecture1)
60          generic map (WIDTH => WIDTH)
61          port map (A => A, B => B, D => D, S => S, Y => Y_a1);
62      uut2: entity work.SN74(architecture2)
63          generic map (WIDTH => WIDTH)
64          port map (A => A, B => B, D => D, S => S, Y => Y_a2);
65      uut3: entity work.SN74(architecture3)
66          generic map (WIDTH => WIDTH)
67          port map (A => A, B => B, D => D, S => S, Y => Y_a3);
68      uut4: entity work.SN74(architecture4)
69          generic map (WIDTH => WIDTH)
70          port map (A => A, B => B, D => D, S => S, Y => Y_a4);
71      uut5: entity work.SN74(architecture5)
72          generic map (WIDTH => WIDTH)
73          port map (A => A, B => B, D => D, S => S, Y => Y_a5);
74      -- Testing process
75      process
76          file input_file: text;
77          variable line_content: line;
78          variable D_h, S_h, A3_h, A2_h, A1_h, A0_h, B3_h, B2_h, B1_h,
                B0_h: boolean;
79          variable D_l, S_l, A3_l, A2_l, A1_l, A0_l, B3_l, B2_l, B1_l,
                B0_l: boolean;
80      begin
81          -- Open the input file
82          file_open(input_file, "/home/timurialvarez/BSUIR-GH/AoDoCaS/
                Term7/TestBench3/tests/SN74.txt", read_mode);
83          -- Read and ignore the header line
84          readline(input_file, line_content);
85          -- Read data from the file and simulate the entity
86          while not endfile(input_file) loop
87              -- Read a line from the file
88              readline(input_file, line_content);
89              -- Parse this line
90              read(line_content, D_h);  read(line_content, D_l);
91              read(line_content, S_h);  read(line_content, S_l);
92              read(line_content, A3_h); read(line_content, A3_l);
93              read(line_content, A2_h); read(line_content, A2_l);
94              read(line_content, A1_h); read(line_content, A1_l);
95              read(line_content, A0_h); read(line_content, A0_l);
96              read(line_content, B3_h); read(line_content, B3_l);
97              read(line_content, B2_h); read(line_content, B2_l);
98              read(line_content, B1_h); read(line_content, B1_l);
99              read(line_content, B0_h); read(line_content, B0_l);
100             -- Set the input signals to the values read from the file
```

```
101             D <= to_std_logic(D_h, D_l);
102             S <= to_std_logic(S_h, S_l);
103             A <= to_std_logic(A3_h, A3_l) & to_std_logic(A2_h, A2_l) &
                   to_std_logic(A1_h, A1_l) & to_std_logic(A0_h, A0_l);
104             B <= to_std_logic(B3_h, B3_l) & to_std_logic(B2_h, B2_l) &
                   to_std_logic(B1_h, B1_l) & to_std_logic(B0_h, B0_l);
105             -- Check if the output value matches the expected value
106             wait for 1ps;
107             assert Y_a1 = Y_a2 and Y_a1 = Y_a3 and Y_a1 = Y_a4 and Y_a1
                    = Y_a5
108                 report ”Mismatch detected for Y_a1” severity error;
109             wait for DELAY;
110         end loop;
111         -- Close the input file
112         file_close(input_file);
113         -- Wait for simulation to end
114         wait;
115     end process;
116 end architecture;
```

**ПРИЛОЖЕНИЕ Д**
(*обязательное)*

Исходный текст модуля DFlipFlop

Файл ‘DFlipFlop.vhd’ содержит следующий код:

```
 1 ----------------------------------------------------------------------------------
 2 -- Company:
 3 -- Engineer:
 4 --
 5 -- Create Date: 11.10.2023 17:13:03
 6 -- Design Name:
 7 -- Module Name: DFlipFlop - custom
 8 -- Project Name:
 9 -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Category: SYNCHRONOUS 1-BIT D-FLIP-FLOP WITH ASYNCHRONOUS RESET
26 -- Implementation: custom DFlipFlop
27 entity DFlipFlop is
28     generic (
29         INV_C: boolean;
30         INV_R: boolean
31     );
32     port (
33         -- Input pins
34         C: in  std_logic;
35         D: in  std_logic;
36         R: in  std_logic;
37         -- Output pins
38         Q: out std_logic
39     );
40 end entity;
41
42 architecture custom of DFlipFlop is begin
43     process (C, R, D) begin
44         if (R = '1' and not INV_R) or (R = '0' and INV_R) then
45             -- Asynchronous Reset
46             Q <= '0';
47         elsif (rising_edge(C) and not INV_C) or (falling_edge(C) and
               INV_C) then
48             -- Synchronous load
49             Q <= D;
```

```
50          end if;
51      end process;
52  end architecture;
```

**ПРИЛОЖЕНИЕ Е**
(*обязательное)*

Исходный текст модуля Counter

Файл ‘Counter.vhd’ содержит следующий код:

```
 1 ----------------------------------------------------------------------------------

 2 -- Company:
 3 -- Engineer:
 4 --
 5 -- Create Date: 11.10.2023 17:13:20
 6 -- Design Name:
 7 -- Module Name: Counter - custom
 8 -- Project Name:
 9 -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------

20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Category: SYNCHRONOUS 4-BIT UP/DOWN BINARY COUNTERS WITH 3-STATE
       OUTPUTS
26 -- THAT IS A LIE! IF FACT IT IS NOT BINARY BUT DECADE COUNTER (AKA BCD)
27 -- Implementation: custom Counter
28 entity Counter is
29     -- Generics
30     generic (
31         WIDTH: positive := 4
32     );
33     -- Ports
34     port (
35         -- Input pins (control)
36         OE:   in  std_logic;
37         UD:   in  std_logic;
38         CLK:  in  std_logic;
39         ENT:  in  std_logic;
40         ENP:  in  std_logic;
41         SCLR: in  std_logic;
42         LOAD: in  std_logic;
43         ACLR: in  std_logic;
44         -- Input pins (data)
45         D:    in  std_logic_vector(WIDTH - 1 downto 0);
46         -- Output pins (status)
47         CCO:  out std_logic;
48         RCO:  out std_logic;
49         -- Output pins (data)
```

```
50          Q:    out std_logic_vector(WIDTH - 1 downto 0)
51      );
52  end entity;
53
54  --
       |-----------------------------------------------------------------------|

55  -- |                              Function table
                                    |
56  --
       |----|------|------|------|-----|-----|-----|-----|--------------------|

57  -- | __ | ____ | ____ | ____ | ___ | ___ |   _ |     |
                              |
58  -- | OE | ACLR | SCLR | LOAD | ENT | ENP | U/D | CLK | Operation
                 |
59  --
       |----|------|------|------|-----|-----|-----|-----|--------------------|

60  -- | H  |  X   |  X   |  X   |  X  |  X  |  X  |  X  | Q outputs
       disabled |
61  -- | L  |  L   |  X   |  X   |  X  |  X  |  X  |  X  | Asynchronous
       clear |
62  -- | L  |  H   |  L   |  X   |  X  |  X  |  X  | ↑   | Synchronous
       clear  |
63  -- | L  |  H   |  H   |  L   |  X  |  X  |  X  | ↑   | Load
                     |
64  -- | L  |  H   |  H   |  H   |  L  |  L  |  H  | ↑   | Count up
                   |
65  -- | L  |  H   |  H   |  H   |  L  |  L  |  L  | ↑   | Count down
                |
66  -- | L  |  H   |  H   |  H   |  H  |  X  |  X  |  X  | Inhibit count
              |
67  -- | L  |  H   |  H   |  H   |  X  |  H  |  X  |  X  | Inhibit count
              |
68  --
       |----|------|------|------|-----|-----|-----|-----|--------------------|

69  architecture custom of Counter is
70      -- Constants
71      constant INV_C: boolean := false;
72      constant INV_R: boolean := true;
73      -- Internal signals: DIR, CTL
74      signal DIR_U: std_logic;
75      signal DIR_D: std_logic;
76      signal CTL:   std_logic_vector(6 downto 0);
77      -- Internal signals: XB, DVC
78      signal XB:    std_logic_vector(WIDTH - 1 downto 0);
79      signal DVC:   std_logic_vector(WIDTH - 1 downto 0);
80      -- INternal signals XB, DV[A, B, C], DV, QV, R
81      signal XC:    std_logic_vector(WIDTH - 1 downto 0);
82      signal DVA:   std_logic_vector(WIDTH - 1 downto 0);
83      signal DVB:   std_logic_vector(WIDTH - 1 downto 0);
84      signal DV:    std_logic_vector(WIDTH - 1 downto 0);
```

```
85      signal QV:    std_logic_vector(WIDTH - 1 downto 0);
86      signal R:     std_logic_vector(WIDTH - 1 downto 0);
87      -- Internal signals: W
88      signal W:     std_logic_vector(2 downto 1);
89  begin
90      -- DIR
91      DIR_U <= UD;
92      DIR_D <= not UD;
93      -- CTL
94      CTL(0) <= SCLR and LOAD;
95      CTL(1) <= SCLR and not LOAD;
96      CTL(2) <= not ENT and not ENP and SCLR and not CTL(1);
97      CTL(3) <= DIR_U and R(3) and R(2) and R(1) and R(0) and not ENT;
98      CTL(4) <= not ENT and R(0) and R(1) and R(2) and R(3) and DIR_D;
99      CTL(5) <= not (CTL(3) or CTL(4));
100     CTL(6) <= not ENT and not ENP;
101     -- XB
102     XB(0) <= not CTL(2);
103     XB(1) <= not (R(0) and CTL(2));
104     XB(2) <= not (R(0) and R(1) and CTL(2));
105     XB(3) <= not (R(0) and CTL(2));
106     -- DVC
107     DVC(0) <= CTL(2) and XC(0);
108     DVC(1) <= R(0) and CTL(2) and W(2) and W(1) and not QV(1);
109     DVC(2) <= R(1) and R(0) and CTL(2) and XC(2) and W(2);
110     DVC(3) <= R(2) and R(1) and R(0) and CTL(2) and XC(3);
111
112     -- XC, DV[A, B], DV, QV, R
113     GENERATOR: for N in 0 to WIDTH - 1 generate
114         XC(N) <= not (QV(N) and CTL(0));
115         DVA(N) <= D(N) and CTL(1);
116         DVB(N) <= XB(N) and CTL(0) and QV(N);
117         DV(N) <= DVA(N) or DVB(N) or DVC(N);
118         DFF: entity work.DFlipFlop(custom)
119             generic map (INV_C => INV_C, INV_R => INV_R)
120             port map (C => CLK, D => DV(N), R => ACLR, Q => QV(N));
121         R(N) <= not ((not QV(N) and DIR_U) or (QV(N) and DIR_D));
122     end generate;
123
124     -- W
125     W(1) <= not (DIR_U and R(3));
126     W(2) <= not (not QV(2) and DIR_D and not QV(3));
127     -- CCO
128     CCO <= not (not CLK and CTL(6) and not CTL(5));
129     -- RCO
130     RCO <= CTL(5);
131     -- Q
132     OUTPUT: for N in 0 to WIDTH - 1 generate
133         Q(N) <= QV(N) when OE = '0' else 'Z';
134     end generate;
135 end architecture;
```

**ПРИЛОЖЕНИЕ Ж**

(*обязательное)*

Исходный текст модуля Counter_TestBench

Файл ‘Counter_TestBench.vhd’ содержит следующий код:

```
 1 ----------------------------------------------------------------------------------

 2 -- Company:
 3 -- Engineer:
 4 --
 5 -- Create Date: 21.10.2023 10:14:23
 6 -- Design Name:
 7 -- Module Name: Counter_TestBench - run
 8 -- Project Name:
 9 -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------

20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use std.textio.all;
25 use ieee.std_logic_textio.all;
26
27 entity Counter_TestBench is end entity;
28
29 architecture run of Counter_TestBench is
30     -- Constants
31     constant WIDTH:      positive := 4;
32     -- Constants
33     constant ITERATIONS: integer  := 3 + 4 + 2 + 2 + 13 + 12 + 6 + 1;
34     constant EXTRA_ITER: positive := 1;
35     constant DELAY:      time     := 1000000 ps / (ITERATIONS +
          EXTRA_ITER);
36     -- Input signals (control)
37     signal UUT_OE:   std_logic := 'U';
38     signal UUT_UD:   std_logic := 'U';
39     signal UUT_CLK:  std_logic := 'U';
40     signal UUT_ENT:  std_logic := 'U';
41     signal UUT_ENP:  std_logic := 'U';
42     signal UUT_SCLR: std_logic := 'U';
43     signal UUT_LOAD: std_logic := 'U';
44     signal UUT_ACLR: std_logic := 'U';
45     -- Input signals (data)
46     signal UUT_D:    std_logic_vector(WIDTH - 1 downto 0) := (others =>
           'U');
47     -- Output signals (status)
48     signal UUT_CCO:  std_logic := 'U';
```

```
49      signal UUT_RCO:  std_logic := 'U';
50      -- Output signals (data)
51      signal UUT_Q:    std_logic_vector(WIDTH - 1 downto 0) := (others =>
            'U');
52      -- Utility functions
53      function to_std_logic(H, L: boolean) return std_logic is begin
54          if H then
55              if L then return 'Z';
56              else      return 'X';
57              end if;
58          else
59              if L then return '1';
60              else      return '0';
61              end if;
62          end if;
63      end function;
64  begin
65      -- Instantiate units under test (UUT-s)
66      UUT: entity work.Counter(custom)
67          generic map (WIDTH => WIDTH)
68          port map (
69              OE => UUT_OE,
70              UD => UUT_UD,
71              CLK => UUT_CLK,
72              ENT => UUT_ENT,
73              ENP => UUT_ENP,
74              SCLR => UUT_SCLR,
75              LOAD => UUT_LOAD,
76              ACLR => UUT_ACLR,
77              D => UUT_D,
78              CCO => UUT_CCO,
79              RCO => UUT_RCO,
80              Q => UUT_Q
81          );
82      -- Testing process
83      process
84          file input_file: text;
85          variable line_content: line;
86          variable OE_h, ACLR_h, SCLR_h, LOAD_h, ENP_h, ENT_h, UD_h,
                CLK_h, D3_h, D2_h, D1_h, D0_h, Q3_h, Q2_h, Q1_h, Q0_h, RCO_h
                , CCO_h: boolean;
87          variable OE_l, ACLR_l, SCLR_l, LOAD_l, ENP_l, ENT_l, UD_l,
                CLK_l, D3_l, D2_l, D1_l, D0_l, Q3_l, Q2_l, Q1_l, Q0_l, RCO_l
                , CCO_l: boolean;
88      begin
89          -- Open the input file
90          file_open(input_file, "/home/timurialvarez/BSUIR-GH/AoDoCaS/
                Term7/TestBench3/tests/Counter.txt", read_mode);
91          -- Read and ignore the header line
92          readline(input_file, line_content);
93          -- Read data from the file and simulate the entity
94          while not endfile(input_file) loop
95              -- Read a line from the file
96              readline(input_file, line_content);
```

```
 97            -- Parse this line
 98            read(line_content, OE_h);   read(line_content, OE_l);
 99            read(line_content, ACLR_h); read(line_content, ACLR_l);
100            read(line_content, SCLR_h); read(line_content, SCLR_l);
101            read(line_content, LOAD_h); read(line_content, LOAD_l);
102            read(line_content, ENP_h);  read(line_content, ENP_l);
103            read(line_content, ENT_h);  read(line_content, ENT_l);
104            read(line_content, UD_h);   read(line_content, UD_l);
105            read(line_content, CLK_h);  read(line_content, CLK_l);
106            read(line_content, D3_h);   read(line_content, D3_l);
107            read(line_content, D2_h);   read(line_content, D2_l);
108            read(line_content, D1_h);   read(line_content, D1_l);
109            read(line_content, D0_h);   read(line_content, D0_l);
110            read(line_content, Q3_h);   read(line_content, Q3_l);
111            read(line_content, Q2_h);   read(line_content, Q2_l);
112            read(line_content, Q1_h);   read(line_content, Q1_l);
113            read(line_content, Q0_h);   read(line_content, Q0_l);
114            read(line_content, RCO_h);  read(line_content, RCO_l);
115            read(line_content, CCO_h);  read(line_content, CCO_l);
116            -- Set the input signals to the values read from the file
117            UUT_OE   <= to_std_logic(OE_h, OE_l);
118            UUT_ACLR <= to_std_logic(ACLR_h, ACLR_l);
119            UUT_SCLR <= to_std_logic(SCLR_h, SCLR_l);
120            UUT_LOAD <= to_std_logic(LOAD_h, LOAD_l);
121            UUT_ENP  <= to_std_logic(ENP_h, ENP_l);
122            UUT_ENT  <= to_std_logic(ENT_h, ENT_l);
123            UUT_UD   <= to_std_logic(UD_h, UD_l);
124            UUT_CLK  <= to_std_logic(CLK_h, CLK_l);
125            UUT_D    <= to_std_logic(D3_h, D3_l) & to_std_logic(D2_h,
                  D2_l) & to_std_logic(D1_h, D1_l) & to_std_logic(D0_h,
                  D0_l);
126            -- Check if the output value matches the expected value
127            wait for 1ps;
128            assert UUT_Q = to_std_logic(Q3_h, Q3_l) & to_std_logic(Q2_h
                  , Q2_l) & to_std_logic(Q1_h, Q1_l) & to_std_logic(Q0_h,
                  Q0_l)
129                report "Mismatch detected for Q" severity error;
130            assert UUT_RCO = to_std_logic(RCO_h, RCO_l)
131                report "Mismatch detected for RCO" severity error;
132            assert UUT_CCO = to_std_logic(CCO_h, CCO_l)
133                report "Mismatch detected for CCO" severity error;
134            wait for DELAY;
135        end loop;
136        -- Close the input file
137        file_close(input_file);
138        -- Wait for simulation to end
139        wait;
140    end process;
141 end architecture;
```