Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Дисциплина: Системы на кристалле

Лабораторная работа №2

Проектирование верификационного окружения для СнК

Вариант - 4

Группа: P3432

Выполнили:
Глотов Егор Дмитриевич
Чмурова Мария Владиславовна
Ефимов Арслан Альбертович

Преподаватель:
Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2025 г.

Цель

Получить знания и навыки верификации микропроцессорных систем на кристалле и созданию тестовых окружений.

Структура разработанного верификационного окружения

В данной лабораторной мы придерживались архитектуре UVM (Universal Verification Methodology).

В итоге получили следующую структуру:

1. Sequencer – генератор последовательности входных данных (тестовых транзакций)
2. Driver - принимает сигнал от Sequencer и преобразует в реальные сигналы, которые подаются на DUT через интерфейс
3. Monitor – следит за сигналами DUT, но не вмешивается в его работу. Снимает входные значения и передает их в Scoreboard, а также фиксирует их в лог-файле и собирает статистику, фиксирует время работы алгоритма.
4. Scoreboard – сравнивает фактические выходные данные DUT с эталонными.
5. Agent – инкапсулирует в себе связку sequencer-driver-monitor для конкретного интерфейса DUT.

Содержимое файла монитора

```
=== Matrix A captured at 215000 ns ===
A[0] = 0067
A[1] = 000e
A[2] = 0057
A[3] = 0083
A[4] = 005e
A[5] = 000f
A[6] = 007e
A[7] = 0023
A[8] = 002c
A[9] = 0069
A[10] = 0039
A[11] = 0065
A[12] = 002b
A[13] = 0086
A[14] = 0040
A[15] = 005d
A[16] = 000a
A[17] = 0045
A[18] = 0039
A[19] = 005c
A[20] = 0057
A[21] = 003e
A[22] = 001e
A[23] = 0058
A[24] = 0044
A[25] = 000e
A[26] = 0028
A[27] = 0011
A[28] = 008b
A[29] = 003d
A[30] = 002b
A[31] = 009b
A[32] = 0098
A[33] = 0041
A[34] = 0089
A[35] = 0042
A[36] = 003a
A[37] = 0036
A[38] = 006f
A[39] = 0010
A[40] = 0030
A[41] = 009f
A[42] = 00a1
A[43] = 005b
A[44] = 0031
A[45] = 0021
A[46] = 0077
A[47] = 006d
A[48] = 007f
A[49] = 0085
A[50] = 003e
```

```
A[51] = 0023
A[52] = 0023
A[53] = 0091
A[54] = 001c
A[55] = 003a
A[56] = 000c
A[57] = 000f
A[58] = 002c
A[59] = 00a4
A[60] = 0029
A[61] = 0072
A[62] = 0017
A[63] = 0069

=== Matrix B captured at 153705000 ns ===
B[0] = 002a
B[1] = 0043
B[2] = 0016
B[3] = 0099
B[4] = 0034
B[5] = 00a4
B[6] = 001f
B[7] = 0038
B[8] = 0042
B[9] = 0094
B[10] = 0039
B[11] = 0052
B[12] = 0021
B[13] = 000a
B[14] = 002b
B[15] = 0046
B[16] = 0095
B[17] = 0016
B[18] = 005c
B[19] = 0033
B[20] = 0055
B[21] = 00a1
B[22] = 0010
B[23] = 0029
B[24] = 007e
B[25] = 0070
B[26] = 0042
B[27] = 002b
B[28] = 0037
B[29] = 000b
B[30] = 0035
B[31] = 0062
B[32] = 0028
B[33] = 001a
B[34] = 001e
B[35] = 003b
B[36] = 0033
B[37] = 0081
B[38] = 0013
B[39] = 000c
B[40] = 001a
B[41] = 0044
```

```
B[42] = 0021
B[43] = 001a
B[44] = 001b
B[45] = 000d
B[46] = 00a7
B[47] = 0022
B[48] = 0090
B[49] = 0057
B[50] = 0057
B[51] = 0071
B[52] = 0085
B[53] = 003c
B[54] = 0012
B[55] = 0041
B[56] = 000b
B[57] = 0037
B[58] = 0066
B[59] = 0033
B[60] = 0029
B[61] = 0033
B[62] = 001a
B[63] = 006f
Time 1238395000 ns | Expected[0] = 200c | Result[0]=200c
Time 1239745000 ns | Expected[1] = 2373 | Result[1]=2373
Time 1241095000 ns | Expected[2] = 12a3 | Result[2]=12a3
Time 1242445000 ns | Expected[3] = 3e8f | Result[3]=3e8f
Time 1243795000 ns | Expected[4] = 2aee | Result[4]=2aee
Time 1245145000 ns | Expected[5] = 12de | Result[5]=12de
Time 1246495000 ns | Expected[6] = 0c70 | Result[6]=0c70
Time 1247845000 ns | Expected[7] = 0fca | Result[7]=0fca
Time 1249405000 ns | Expected[8] = 313e | Result[8]=313e
Time 1250755000 ns | Expected[9] = 0a6b | Result[9]=0a6b
Time 1252105000 ns | Expected[10] = 1a94 | Result[10]=1a94
Time 1253455000 ns | Expected[11] = 1e3a | Result[11]=1e3a
Time 1254805000 ns | Expected[12] = 05d0 | Result[12]=05d0
Time 1256155000 ns | Expected[13] = 0670 | Result[13]=0670
Time 1257505000 ns | Expected[14] = 17d2 | Result[14]=17d2
Time 1258855000 ns | Expected[15] = 223e | Result[15]=223e
Time 1260415000 ns | Expected[16] = 09da | Result[16]=09da
Time 1261765000 ns | Expected[17] = 0dfa | Result[17]=0dfa
Time 1263115000 ns | Expected[18] = 3b83 | Result[18]=3b83
Time 1264465000 ns | Expected[19] = 2fcf | Result[19]=2fcf
Time 1265815000 ns | Expected[20] = 26d6 | Result[20]=26d6
Time 1267165000 ns | Expected[21] = 37d3 | Result[21]=37d3
Time 1268515000 ns | Expected[22] = 1d4c | Result[22]=1d4c
Time 1269865000 ns | Expected[23] = 3b54 | Result[23]=3b54
Time 1271425000 ns | Expected[24] = 28af | Result[24]=28af
Time 1272775000 ns | Expected[25] = 3290 | Result[25]=3290
Time 1274125000 ns | Expected[26] = 3ffc | Result[26]=3ffc
Time 1275475000 ns | Expected[27] = 0fdf | Result[27]=0fdf
Time 1276825000 ns | Expected[28] = 31a0 | Result[28]=31a0
Time 1278175000 ns | Expected[29] = 380d | Result[29]=380d
Time 1279525000 ns | Expected[30] = 1546 | Result[30]=1546
Time 1280875000 ns | Expected[31] = 3bfa | Result[31]=3bfa
Time 1282435000 ns | Expected[32] = 276f | Result[32]=276f
Time 1283785000 ns | Expected[33] = 334d | Result[33]=334d
Time 1285135000 ns | Expected[34] = 1786 | Result[34]=1786
Time 1286485000 ns | Expected[35] = 1cd9 | Result[35]=1cd9
Time 1287835000 ns | Expected[36] = 3034 | Result[36]=3034
Time 1289185000 ns | Expected[37] = 3994 | Result[37]=3994
Time 1290535000 ns | Expected[38] = 2470 | Result[38]=2470
Time 1291885000 ns | Expected[39] = 0f32 | Result[39]=0f32
Time 1293445000 ns | Expected[40] = 0de4 | Result[40]=0de4
Time 1294795000 ns | Expected[41] = 2b78 | Result[41]=2b78
Time 1296145000 ns | Expected[42] = 1696 | Result[42]=1696
Time 1297495000 ns | Expected[43] = 17c3 | Result[43]=17c3
Time 1298845000 ns | Expected[44] = 03ac | Result[44]=03ac
Time 1300195000 ns | Expected[45] = 1a0c | Result[45]=1a0c
Time 1301545000 ns | Expected[46] = 295f | Result[46]=295f
Time 1302895000 ns | Expected[47] = 069f | Result[47]=069f
Time 1304455000 ns | Expected[48] = 1250 | Result[48]=1250
Time 1305805000 ns | Expected[49] = 027a | Result[49]=027a
Time 1307155000 ns | Expected[50] = 3ee9 | Result[50]=3ee9
Time 1308505000 ns | Expected[51] = 3700 | Result[51]=3700
Time 1309855000 ns | Expected[52] = 34a1 | Result[52]=34a1
Time 1311205000 ns | Expected[53] = 29ef | Result[53]=29ef
Time 1312555000 ns | Expected[54] = 19d1 | Result[54]=19d1
Time 1313905000 ns | Expected[55] = 0c61 | Result[55]=0c61
Time 1315465000 ns | Expected[56] = 138e | Result[56]=138e
Time 1316815000 ns | Expected[57] = 17f3 | Result[57]=17f3
Time 1318165000 ns | Expected[58] = 0338 | Result[58]=0338
Time 1319515000 ns | Expected[59] = 2430 | Result[59]=2430
Time 1320865000 ns | Expected[60] = 2703 | Result[60]=2703
Time 1322215000 ns | Expected[61] = 1f8d | Result[61]=1f8d
Time 1323565000 ns | Expected[62] = 0244 | Result[62]=0244
Time 1324915000 ns | Expected[63] = 108d | Result[63]=108d

=== Algorithm finished ===
Execution time: 1324750000 ns
```

Листинг алгоритма работы

Matrix.h

```
#ifndef MATRIH_H
#define MATRIH_H
#define MATRIX_SIZE 8
int **create_matrix();
void free_matrix(int **matrix);
int **multiply_matrices(int **A, int **B);
int **subtract_matrices(int **A, int **B);
int **calculate_function(int **A, int **B);
#endif
```

Matrix.c

```
#include <stdlib.h>
#include "matrix.h"
int** create_matrix()
{
    int **matrix = (int**)malloc(MATRIX_SIZE * sizeof(int*));
    for (int i = 0; i < MATRIX_SIZE; i++)
        matrix[i] = (int*)malloc(MATRIX_SIZE * sizeof(int));
    return matrix;
}
void free_matrix(int **matrix)
{
    for (int i = 0; i < MATRIX_SIZE; i++)
        free(matrix[i]);
    free(matrix);
}
int** multiply_matrices(int **A, int **B)
{
    int **result = create_matrix();
    for (int i = 0; i < MATRIX_SIZE; i++)
    {
        for (int j = 0; j < MATRIX_SIZE; j++)
        {
            result[i][j] = 0;
            for (int k = 0; k < MATRIX_SIZE; k++)
                result[i][j] += A[i][k] * B[k][j];
        }
    }
    return result;
}
```

```
int** subtract_matrices(int **A, int **B)
{
    int **result = create_matrix();
    for (int i = 0; i < MATRIX_SIZE; i++)
    {
        for (int j = 0; j < MATRIX_SIZE; j++)
            result[i][j] = A[i][j] - B[i][j];
    }
    return result;
}
int** calculate_function(int **A, int **B)
{
    int **A_mul_B = multiply_matrices(A, B);
    int **result = subtract_matrices(A_mul_B, B);
    free_matrix(A_mul_B);
    return result;
}
```

gpio.h

```
#ifndef GPIO_H
#define GPIO_H
#include "xil_io.h"
#include "platform.h"
#define GPIO_OUT 0x40000000
#define GPIO_IN 0x40000008
int get_value();
void send_value(int value);
#endif
```

Helloworld.c

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xil_io.h"
#include "matrix.h"
#include "gpio.h"
int main() {
    init_platform();

    int **A = create_matrix();
    for (int i = 0; i < MATRIX_SIZE; i++) {
        for (int j = 0; j < MATRIX_SIZE; j++) {
            A[i][j] = get_value();
        }
    }

    int **B = create_matrix();
    for (int i = 0; i < MATRIX_SIZE; i++) {
        for (int j = 0; j < MATRIX_SIZE; j++) {
            B[i][j] = get_value();
        }
    }

    int **C = calculate_function(A, B);

    for (int i = 0; i < MATRIX_SIZE; i++) {
        for (int j = 0; j < MATRIX_SIZE; j++) {
            send_value(C[i][j]);
        }
    }
    free_matrix(A);
    free_matrix(B);
    free_matrix(C);
    cleanup_platform();
    return 0;
}
```

Листинг верификационного окружения

Sequencer.sv

```
module sequencer#(
    parameter int SIZE = 64
)(
    input logic clk_i,
    input logic rst_i,
    output logic [15:0] sequence_o[SIZE],
    output logic sequence_valid_o,
    input logic sequence_send_i
);
    localparam int SEND_A = 0;
    localparam int WAIT = 1;
    localparam int SEND_B = 2;
    logic [1:0] state;
    function automatic logic [15:0] pick_weighted();
        int sel;
        int rand_val;
        sel = $urandom_range(1, 5);
        if (sel <= 3) begin
            rand_val = $urandom_range(10, 70);
        end else begin
            rand_val = $urandom_range(80, 170);
        end
        return rand_val[15:0];
    endfunction
    always_ff @(posedge clk_i) begin
        if (rst_i) begin
            state <= SEND_A;
            for (int i = 0; i < SIZE; i++) begin
                sequence_o[i] <= '0;
            end
            sequence_valid_o <= 1'b0;
        end
```

```
else begin
            case (state)
                SEND_A: begin
                    for (int i = 0; i < SIZE; i++) begin
                        sequence_o[i] <= pick_weighted();
                    end
                    sequence_valid_o <= 1'b1;
                    state <= WAIT;
                end
                WAIT: begin
                    if (sequence_send_i) begin
                        state <= SEND_B;
                    end
                    sequence_valid_o <= 1'b0;
                end
                SEND_B: begin
                    for (int i = 0; i < SIZE; i++) begin
                        sequence_o[i] <= pick_weighted();
                    end
                    sequence_valid_o <= 1'b1;
                    state <= WAIT;
                end
                default: begin
                    state <= SEND_A;
                end
            endcase
        end
    end
endmodule
```

Driver.sv

```
module driver#(
    parameter int SIZE = 64
)(
    input clk,
    input rst,

    input [15:0] sequence_i[SIZE],
    input sequence_valid,
    output logic sequence_send,

    output logic [15:0] gpio_switch,
    input [15:0] gpio_led
    );

    localparam WAIT_SEQ = 0;
    localparam SEND_SEQ = 1;

    logic state;
    logic recieve;
logic [15:0] temp_matrix [SIZE];
    logic [6:0] temp_idx;

    int i;

    wire send_vld = gpio_switch[15];
    wire recieve_vld = gpio_led[15];

    always_ff @(posedge clk) begin
        if (rst) begin
            for (int i = 0; i < SIZE; i++) begin
                temp_matrix[i] <= '0;
            end
            temp_idx <= '0;
            state <= WAIT_SEQ;
            sequence_send <= '0;
            recieve <= '0;
        end else begin
            case (state)
                WAIT_SEQ: begin
                    if (sequence_valid) begin
                        temp_matrix <= sequence_i;
                        state <= SEND_SEQ;
                    end
                    recieve <= '0;
                    temp_idx <= '0;
                    sequence_send <= '0;
                end
                SEND_SEQ: begin
                    if (temp_idx == SIZE) begin
                        state <= WAIT_SEQ;
                        sequence_send <= '1;
                    end else begin
                        if (send_vld) begin
                            if (recieve && !recieve_vld) begin
                                temp_idx <= temp_idx + 1'b1;
                                gpio_switch <= '0;
                                recieve <= '0;
                            end else if (recieve_vld) begin
                                recieve <= 1;
                            end
                        end else begin
                                gpio_switch <= temp_matrix[temp_idx] |
16'h8000;
                        end
                    end
                end
            endcase
        end
    end
endmodule
```

Monitor.sv

```
module monitor#(
    parameter int SIZE = 64
)(
    input clk,
    input rst,
    input [15:0] gpio_led,
    input [15:0] input_matrix_i[SIZE],
    input input_valid_i,
    output logic [15:0] result_matrix_o[SIZE],
    output logic result_valid
    );
    integer file;
    time start_time, end_time;
    logic [15:0] saved_matrix_a[SIZE];
    logic [15:0] saved_matrix_b[SIZE];
    logic [6:0] temp_idx;
    logic [1:0] state;
    logic [1:0] phase;
    localparam RECIEVE = 0;
    localparam DELAY = 1;

    int i;
    initial begin
        file = $fopen("monitor_log.txt", "w");
    end
```

```
 always_ff @(posedge clk) begin
        if (rst) begin
            temp_idx <= 0;
            state <= RECIEVE;
            phase <= 0;
            start_time <= $time;
        end
        else begin
            if (input_valid_i) begin
                if (phase == 0) begin
                    saved_matrix_a <= input_matrix_i;
                    $fwrite(file, "\n=== Matrix A captured at %0t ns ===\n",
$time);
                    for (int i = 0; i < SIZE; i++)
                        $fwrite(file, "A[%0d] = %h\n", i,
input_matrix_i[i]);
                    phase <= 1;
                end else if (phase == 1) begin
                    saved_matrix_b <= input_matrix_i;
                    $fwrite(file, "\n=== Matrix B captured at %0t ns ===\n",
$time);
                    for (int i = 0; i < SIZE; i++)
                        $fwrite(file, "B[%0d] = %h\n", i,
input_matrix_i[i]);
                    phase <= 2;
                end
            end
            if (state == RECIEVE) begin
                if (gpio_led[14]) begin
                    result_matrix_o[temp_idx] <= gpio_led[13:0];
                    $fwrite(file, "Time %0t ns | Result[%0d]=%h\n",
                            $time, temp_idx, gpio_led[13:0]);
                    temp_idx <= temp_idx + 1'b1;
                    state <= DELAY;
                end
            end else begin
                if (!gpio_led[14]) begin
                    state <= RECIEVE;
                end
            end
            if (temp_idx == SIZE) begin
                result_valid <= 1;
                end_time = $time;
            end else begin
                result_valid <= 0;
            end
        end
    end

    final begin
        $fwrite(file, "\n=== Algorithm finished ===\n");
        $fwrite(file, "Execution time: %0t ns\n", end_time - start_time);
        $fclose(file);
    end

endmodule
```

Scoreboard.sv

```
module scoreboard#(
    parameter int SIZE = 64,
    parameter int ROW = 8
)(
    input logic clk_i,
    input logic rst_i,
    input logic [15:0] matrix_input [SIZE],
    input logic matrix_vld,
    input logic [15:0] result_matrix [SIZE],
    input logic result_vld
);
    typedef enum logic [1:0] {
        WAIT_A = 2'd0,
        WAIT_B = 2'd1,
        WAIT_RESULT = 2'd2,
        CHECK = 2'd3
    } state_t;
    state_t state;
    logic [15:0] matrix_a [SIZE];
    logic [15:0] matrix_b [SIZE];
    logic [15:0] product_matrix [SIZE];
    int i, j, k;
    int errors;
    int sum;

    always_ff @(posedge clk_i) begin
        if (rst_i) begin
            state <= WAIT_A;
            errors <= 0;
            for (i = 0; i < SIZE; i++) begin
                matrix_a[i] <= '0;
                matrix_b[i] <= '0;
                product_matrix[i] <= '0;
            end
        end
        else begin
            case (state)
                WAIT_A: begin
                    if (matrix_vld) begin
                        for (i = 0; i < SIZE; i++)
                            matrix_a[i] <= matrix_input[i];
                        state <= WAIT_B;
                    end
                end
                WAIT_B: begin
                    if (matrix_vld) begin
                        for (i = 0; i < SIZE; i++)
                            matrix_b[i] <= matrix_input[i];
                        state <= WAIT_RESULT;
                    end
                end
                WAIT_RESULT: begin
                    if (result_vld)
                        state <= CHECK;
                end
                CHECK: begin
                    errors = 0;
                    for (i = 0; i < ROW; i++) begin
                        for (j = 0; j < ROW; j++) begin
                            sum = 0;
                            for (k = 0; k < ROW; k++) begin
                                sum += matrix_a[i*ROW + k] * matrix_b[k*ROW
+ j];
                            end
                            product_matrix[i*ROW + j] = (sum -
matrix_b[i*ROW + j]) & 16'h3FFF;
                        end
                    end
```

```
                    for (i = 0; i < SIZE; i++) begin
                        if (product_matrix[i] !== result_matrix[i]) begin
                            $display("Mismatch [%0d]: expected=%0h,
got=%0h",
                                     i, product_matrix[i],
result_matrix[i]);
                            errors++;
                        end else begin
                            $display("Match [%0d]: %0h", i,
result_matrix[i]);
                        end
                    end
                    if (errors == 0)
                        $display("RESULT CORRECT: (A*B - B) mod 16 bits");
                    else
                        $display("Total mismatches: %0d", errors);
                    $finish;
                end
            endcase
        end
    end
endmodule
```

Agent.sv

```
module agent#(
    parameter int SIZE = 64
)(
    input clk_i,
    input rst_i,
    output [15:0] gpio_switch,
    output [15:0] gpio_led
    );

    logic [15:0] tmp_sequence[SIZE];
    logic [15:0] result_sequence[SIZE];
    logic sequence_valid, sequence_send, result_valid;
```

```
sequencer sequencer_impl (
        .clk_i(clk_i),
        .rst_i(rst_i),
        .sequence_o(tmp_sequence),
        .sequence_valid_o(sequence_valid),
        .sequence_send_i(sequence_send)
    );

    driver driver_impl (
        .clk(clk_i),
        .rst(rst_i),

        .sequence_i(tmp_sequence),
        .sequence_valid(sequence_valid),
        .sequence_send(sequence_send),

        .gpio_switch(gpio_switch),
        .gpio_led(gpio_led)
    );

    monitor monitor_impl (
        .clk(clk_i),
        .rst(rst_i),
        .gpio_led(gpio_led),

        .input_matrix_i(tmp_sequence),
        .input_valid_i(sequence_valid),

        .result_matrix_o(result_sequence),
        .result_valid(result_valid)
    );

    scoreboard scoreboard_impl (
        .clk_i(clk_i),
        .rst_i(rst_i),

        .matrix_input(tmp_sequence),
        .matrix_vld(sequence_valid),

        .result_matrix(result_sequence),
        .result_vld(result_valid)
    );

endmodule
```

Вывод

В ходе лабораторной работы были получены практические знания и навыки по верификации микропроцессорных систем на кристалле и созданию автоматизированных тестовых окружений.

Было разработано и настроено верификационное окружение, основанное на архитектуре UVM, включающее ключевые компоненты: Sequencer, Driver, Monitor и Scoreboard.

Проведено функциональное тестирование разработанного модуля с использованием рандомизации входных сигналов, что позволило проверить корректность работы алгоритма в различных сценариях и повысить надёжность проекта.

Результатом работы стало понимание принципов построения современных верификационных систем и практический опыт реализации комплексного тестбенча.