DIGITAL SYSTEM DESIGN APPLICATIONS

(CRN: 11275)

THE REPORT OF PROJECT - 1

Faculty of Electrical and Electronics Engineering

Electronics and Communication Engineering

Yusuf Tekin - 040200043

Bora Kıran – 040210069

Yusuf Tekin / Bora Kıran
040200043 / 040210069

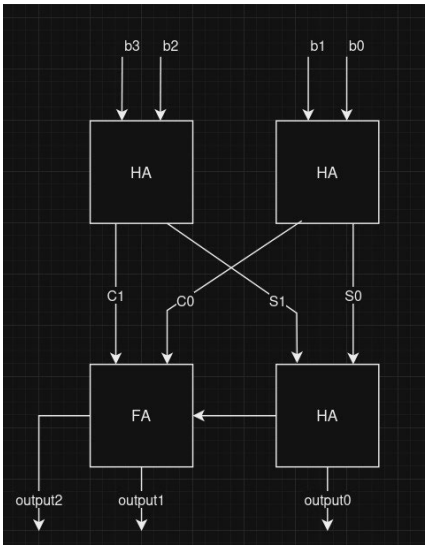
Hamming Weight Calculator using Parallel Counters

Algorithm Explanation: The design slices 32-bit inputs into 4-bit length pieces. Each part goes into a custom counter that counts the number of ones (hamming weight). Outputs of the custom counters are summed into the total hamming weight. To be able to design such mechanism, these steps applied in order:

1- The first step is creating a stimulus text file via generating numbers with python. The script generated 98 random inputs and to be able check the least and the most important bits precisely, all 0s and all 1s values are added manually.

2- The second step is designing the architecture of the overall system. Dividing the 32-bit input into 4-bit parts would be the best option since the parallel working blocks would take up less space rather than using summing operators for all 32-bits. Only downside is that the summing part of the calculated hamming weights would increase in complexity, but the summing is handled by summing operator which uses summing primitives.

3- To be able to check the system with generated binary numbers, the simulation created with the self-checking properties. In the testbench, with the use of operators as "$fopen, $fclose, $feof…" the binary and outputs files are assigned to the proper pointers. Using a "while" loop until the end of the files, all the values within the stimulus text are checked as inputs and if the values are same with the values in the outputs text file, it would be stated in TCL Console as correct.



4-bit Hammering Weight Block

Yusuf Tekin / Bora Kıran
040200043 / 040210069

```
`timescale 1ns / 1ps

module HA(
input x,
input y,
output cout,
output sum);

assign cout = x & y;
assign sum = x^y;
endmodule

module FA(
   input x,y,cin,
   output cout,sum
);
wire c0,c1,s0;
HA ha0(x,y,c0,s0);
HA ha1(s0,cin,c1,sum);
assign cout = c1 | c0;
endmodule
(* DONT_TOUCH = "TRUE" *)
module counter_4bit(
   input [3:0] in,
   output[2:0] o
);
wire c0,s0,c1,s1,c2;

HA ha0(in[1],in[0],c0,s0);
HA ha1(in[3],in[2],c1,s1);
HA ha2(s1,s0,c2,o[0]);
FA fa0(c1,c0,c2,o[2],o[1]);

endmodule


(* DONT_TOUCH = "TRUE" *)
module main (
input [31:0] in,
output [5:0] out
   );
wire [2:0] block_output0;
wire [2:0] block_output1;
wire [2:0] block_output2;
wire [2:0] block_output3;
wire [2:0] block_output4;
wire [2:0] block_output5;
wire [2:0] block_output6;
wire [2:0] block_output7;
```

Verilog Code - Part 1

Yusuf Tekin / Bora Kıran
040200043 / 040210069

```
counter_4bit b0(in[31:28],block_output0);

counter_4bit b1(in[27:24],block_output1);
counter_4bit b2(in[23:20],block_output2);
counter_4bit b3(in[19:16],block_output3);
counter_4bit b4(in[15:12],block_output4);
counter_4bit b5(in[11:8],block_output5);
counter_4bit b6(in[7:4],block_output6);
counter_4bit b7(in[3:0],block_output7);


assign
out=block_output0+block_output1+block_output2+block_o
utput3+block_output4+block_output5+block_output6+bloc
k_output7;
endmodule
```

Verilog Code - Part 2

```
`timescale 1ns / 1ps

module main_tb;

main uut(
    .in(data_in),
    .out(count_ones)
    );
wire [5:0] count_ones;
reg [31:0] data_in;
reg [31:0] test_data;
reg [5:0] expected_ones;

integer input_file, output_file;
integer status;
integer error;

initial begin
    input_file = $fopen("stimulus_input.txt", "r");
    error = 0;
    if(input_file == 0) begin
        $display("\nError: Failed to open the simulation input
file.\n");
        $stop;
    end

    output_file = $fopen("output.txt", "r");

    if(output_file == 0) begin
        $display("\nError: Failed to open the simulation output
file.\n");
        $stop;
    end
```

Testbench Code - Part 1

Yusuf Tekin / Bora Kıran
040200043 / 040210069

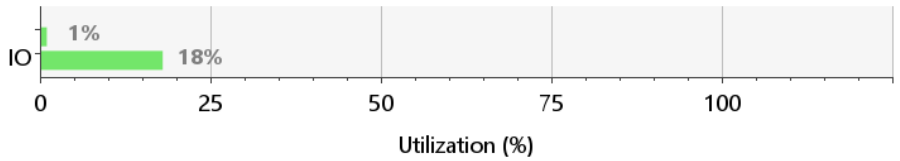
```
    #10;

    while (!$feof(input_file) && !$feof(output_file)) begin

        status = $fscanf(input_file, "%b\n", test_data);
        if(status == 0) begin
            $display("\nError: simulation input cannot read\n");
            $stop;
        end

        status = $fscanf(output_file, "%d\n", expected_ones);
        if(status == 0) begin
            $display("\nError: simulation output cannot read\n");
            $stop;
        end

        data_in = test_data;
        #5;

        if(count_ones !== expected_ones) begin
            $display("\nUnexpected value: Input = %b | Expected_Ones = %d |
Found Value: %d\n", data_in, expected_ones, count_ones);
            error = error + 1;
        end else begin
            $display("Correct value: Input = %b | Ones = %d", data_in,
count_ones);
        end
    end

    if(error == 0) begin
        $display("\nALL CORRECT, DESIGN PASSED ALL THE TESTS
SUCCESFULLY!!\n");
    end else begin
        $display("\nTEST COMPLETED WITH %d ERRORS\n",error);
    end

    $fclose(input_file);
    $fclose(output_file);
    #10;
    $finish();
end
endmodule
```

Testbench Code - Part 2

Yusuf Tekin / Bora Kıran
040200043 / 040210069

Summary

Resource	Utilization	Available	Utilization %
LUT	50	32600	0.15
IO	38	210	18.10



Utilization Summary

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	∞	7	6	9	in[0]	out[5]	12.611	4.912	7.700	∞	input port clock			0.000
Path 2	∞	7	6	5	in[29]	out[3]	12.434	4.898	7.535	∞	input port clock			0.000
Path 3	∞	7	6	9	in[0]	out[4]	12.237	4.695	7.542	∞	input port clock			0.000
Path 4	∞	7	6	7	in[12]	out[2]	10.822	4.206	6.616	∞	input port clock			0.000
Path 5	∞	6	5	5	in[8]	out[1]	10.642	4.333	6.309	∞	input port clock			0.000
Path 6	∞	4	3	5	in[29]	out[0]	9.364	4.046	5.319	∞	input port clock			0.000

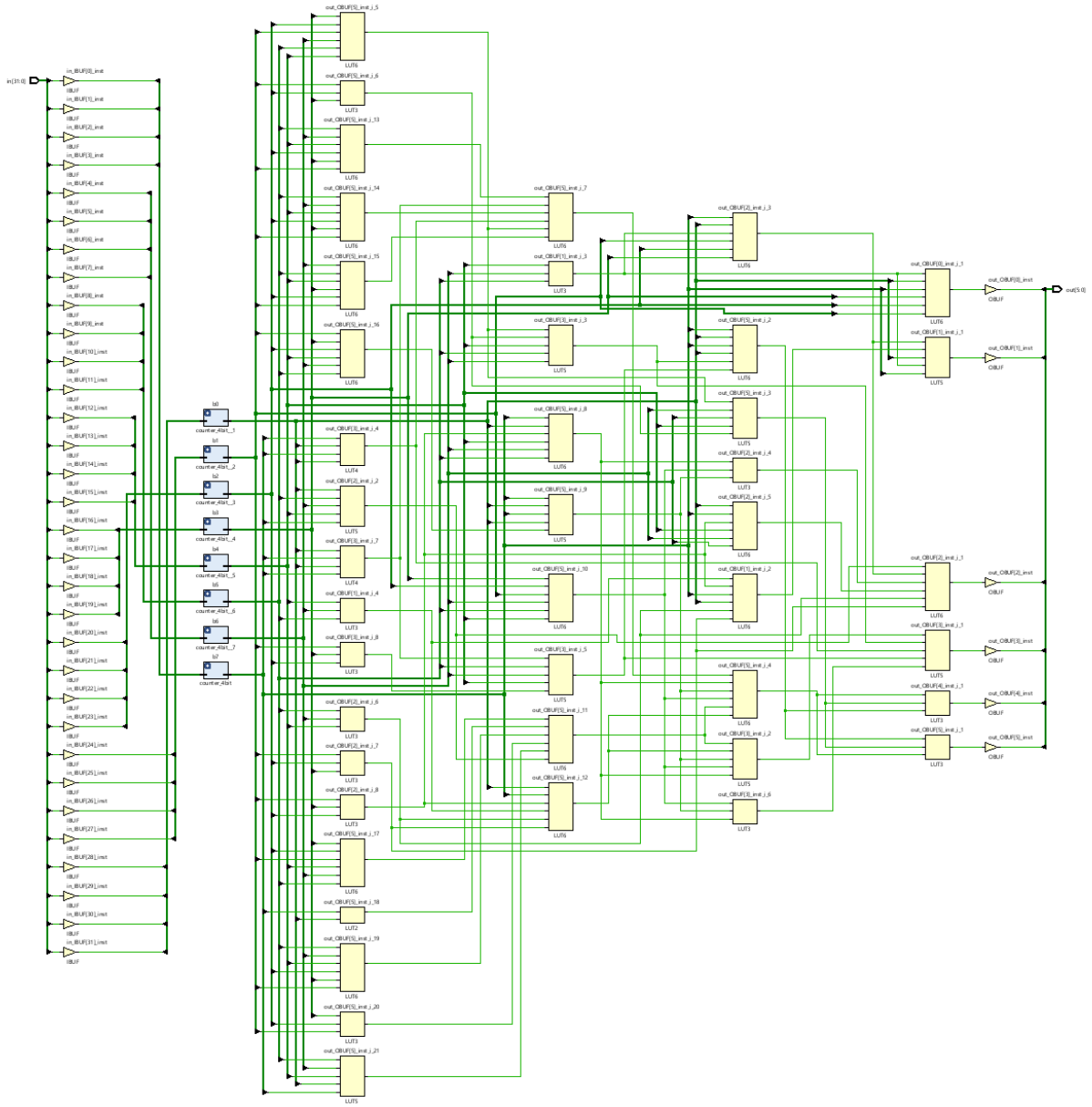
Path Delays - Setup

Unconstrained Paths - NONE - NONE - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 7	∞	4	3	5	in[18]	out[0]	2.564	1.431	1.133	-∞	input port clock			0.000
Path 8	∞	5	4	5	in[18]	out[2]	2.895	1.477	1.418	-∞	input port clock			0.000
Path 9	∞	5	4	5	in[12]	out[1]	2.906	1.540	1.365	-∞	input port clock			0.000
Path 10	∞	5	4	9	in[28]	out[4]	2.982	1.480	1.502	-∞	input port clock			0.000
Path 11	∞	5	4	9	in[28]	out[5]	3.095	1.531	1.564	-∞	input port clock			0.000
Path 12	∞	6	5	6	in[22]	out[3]	3.143	1.532	1.611	-∞	input port clock			0.000

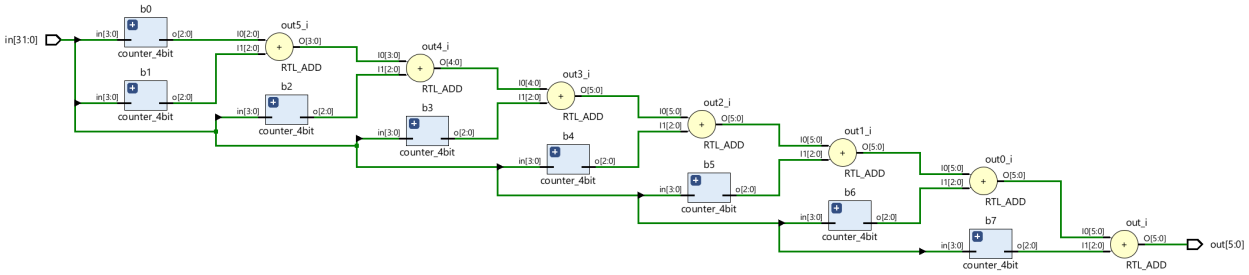
Path Delays - Hold

Yusuf Tekin / Bora Kıran
040200043 / 040210069

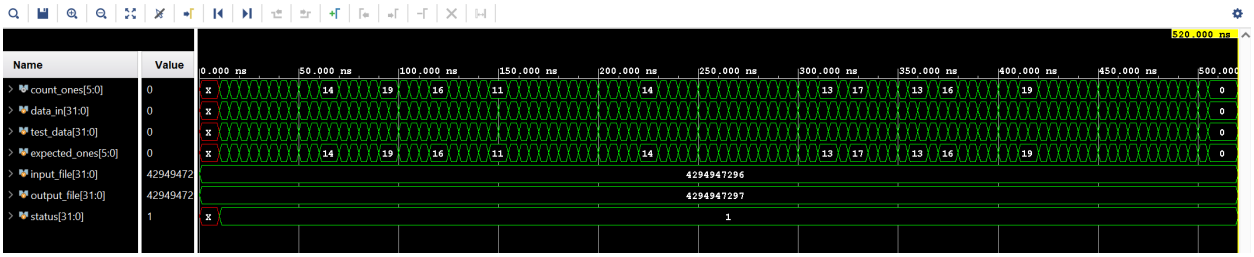


Technology Schematic

Yusuf Tekin / Bora Kıran
040200043 / 040210069



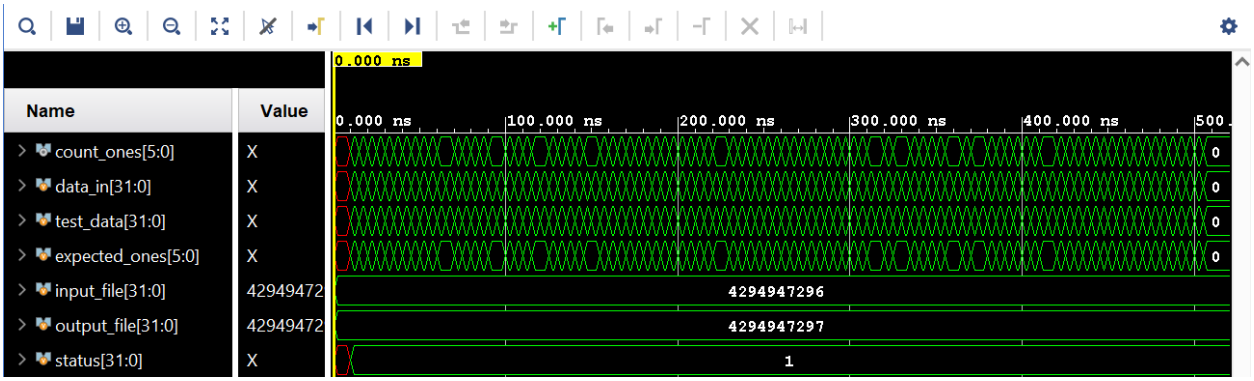
RTL Schematic



Behavioral Simulation

```
Correct value: Input = 01001111101011010100100100000001 | Ones = 14
Correct value: Input = 10000011011101111011101110110101 | Ones = 20
Correct value: Input = 01001100010010100001000100101101 | Ones = 12
Correct value: Input = 11001111101010111011011000111111 | Ones = 22
Correct value: Input = 10111111011001001111010101100010 | Ones = 19
Correct value: Input = 00000110001110101101010100101010 | Ones = 14
Correct value: Input = 00111011000011011000110111111110 | Ones = 19
Correct value: Input = 01110110110011111000011111001111 | Ones = 21
Correct value: Input = 10100010110111000001110001101011 | Ones = 16
Correct value: Input = 01101011110100000111110100111110 | Ones = 19
Correct value: Input = 01001000100010000010111000001011 | Ones = 11
Correct value: Input = 00001110011001101000001011010001 | Ones = 14
Correct value: Input = 00110100001011111010110101110011 | Ones = 18
Correct value: Input = 10110110001100001111010111111111 | Ones = 21
Correct value: Input = 11100000101110100001010010111110 | Ones = 16
Correct value: Input = 01000010010110100111100000101001 | Ones = 13
Correct value: Input = 11111111111111111111111111111111 | Ones = 32
Correct value: Input = 00000000000000000000000000000000 | Ones =  0

ALL CORRECT, DESIGN PASSED ALL THE TESTS SUCCESFULLY!!

$finish called at time : 520 ns : File "C:/Users/jsphtkn/Vivado Projects/sstu_project_1_hamming/sstu_project_1_hamming.srcs/sim_1/new/main_tb.v" Line 69
```

TCL Console Output



Post-Implementation Timing Simulation

Yusuf Tekin / Bora Kıran
040200043 / 040210069

Work Package Table:

- Stimulus Text – Bora Kıran
- Design – Bora Kıran and Yusuf Tekin
- Simulation Code - Yusuf Tekin and Bora Kıran
- Implementation – Yusuf Tekin
- Report – Yusuf Tekin and Bora Kıran

References:

1- Behrooz Parhami, Computer Arithmetic Algorithms and Hardware Designs, 2nd ed. 2010, pp. 164–167.

2- “Hamming weight,” Wikipedia. https://en.wikipedia.org/wiki/Hamming_weight