Experiment 3 Lab Report

EEE3342C - 00012

Yousef Awad

January 2025

Contents

Equipment

For this experiment a Windows 11 computer was used alongside the Xilinx Vivado 2024.2 software, alongside an FPGA board, the BASYS 3 development board. The board specifically only used to ensure the simulation by the Vivado software was accurate in the real world, as well as to verify the simulation software wasn't incorrect.

Objective

The objective for this lab was to show how optimizations could be done, as well as to show how circuit area cost is calculated by the Vivado Software. This experiment specifically introduces the concepts of Sums of Products (SOPs) and Products of Sums (POSs), of which are crucial in doing basic simplification of circuits, helping to make a 2 layered circuit diagram possible for boolean equations. Alongside showcasing a way to form those SOPs and POSs via a Karnaugh Map.

Part 1

Simplification of Boolean Expressions

AB'C + A'BC' + A'B'C

The given equation to simplify is the following:

$$A * B' * C + A' * B * C' + A' * B' * C$$

The simplification process would look like the following:

$$A * B' * C + A' * B * C' + A' * B' * C \rightarrow$$
$$A * B' * C + A' * B' * C + A' * B * C' \rightarrow$$
$$B' * (A * C + A' * C) + A' * B * C' \rightarrow$$
$$B' * C * (A + A') + A' * B * C' \rightarrow$$
$$B' * C * (1) + A' * B * C' \rightarrow$$
$$B' * C + A' * B * C' \rightarrow$$
$$(B' * C + B * C') + (B' * C + A') \rightarrow$$
$$(B \oplus C) + (B' * C + A') \rightarrow$$
$$(B \oplus C) + (B' * C + A') \rightarrow$$
$$(B \oplus C) + B' * C + A'$$

With the following being as far as I can simplify the given equation:

$$(B \oplus C) + B' * C + A'$$

ABC + AB'C + A'BC' + A'B'C

The given equation to simplify is the following:

$$A * B * C + A * B' * C + A' * B * C' + A' * B' * C$$

The simplification process would look like the following:

$$A * B * C + A * B' * C + A' * B * C' + A' * B' * C \rightarrow$$
$$C * (A * B + A * B' + A' * B') + A' * B * C' \rightarrow$$
$$C * (A * B + B' * (A + A')) + A' * B * C' \rightarrow$$
$$C * (A * B + B' * (1)) + A' * B * C' \rightarrow$$
$$C * (A * B + B') + A' * B * C' \rightarrow$$
$$C * ((B' + A) * (B' + B)) + A' * B * C' \rightarrow$$
$$C * ((B' + A) * (1)) + A' * B * C' \rightarrow$$
$$C * (B' + A) + A' * B * C' \rightarrow$$
$$C * (B' + A) + A' * B * C' \rightarrow$$
$$B' * C + A * C + A' * B * C'$$

With the following being as far as I can simplify the given equation:

$$B' * C + A * C + A' * B * C'$$

Find the POS and SOP of the given Truth Table

Given truth table:

A	B	C	F
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

I derived the following sum of products gained from looking at where the result is 1/True, unsimplified:

$$A' * B' * C + A * B' * C' + A * B' * C + A * B * C$$

I then derived the following product of sums by looking at the output being 0/False and then negating the result, gaining this unsimplified form before and after DeMorgan's Law is applied:

$$(A' * B' * C' + A' * B * C' + A' * B * C + A * B * C')' \rightarrow$$
$$(A + B + C)(A + B' + C)(A + B' + C')(A' + B' + C)$$

Part 2

Given the following truth table:

A	B	C	F
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

I derived the following maxterm equation:

$$A' * B' * C + A' * B * C' + A * B' * C + A * B * C' + A * B * C$$

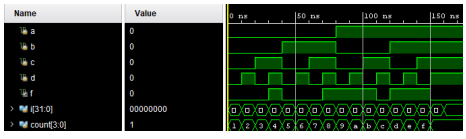
Of which we can then simplify as shown below:

$A' * B' * C + A' * B * C' + A * B' * C + A * B * C' + A * B * C$	Start
$A' * (B' * C + B * C') + A * (B' * C + B * (C' + C))$	Distributive Law
$A' * (B' * C + B * C') + A * (B' * C + B * (1))$	Inverse Law
$A' * (B' * C + B * C') + A * (B' * C + B)$	Identity Law
$A' * (B \oplus C) + A * (B' * C + B)$	XOR Law
$A' * (B \oplus C) + A * ((B + B')(B + C))$	Distributive Law
$A' * (B \oplus C) + A * ((1)(B + C))$	Inverse Law
$A' * (B \oplus C) + A * (B + C)$	Identity Law
$A' * (B \oplus C) + A * B + A * C$	Distributive Law

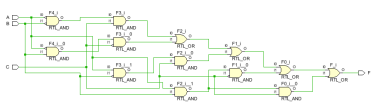
With the final simplification being:

$$A' * (B \oplus C) + A * B + A * C$$

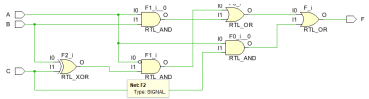
After coding the simplified boolean expression into Verilog, and generating the waveform to double check with the truth table above:



And, of which, gets the following schematic, with the before schematic being shown to the left of it:



(a) Before Schematic



(b) After Schematic

Of which, the verilog code for the before and after schematics is the following:

```
module before(
        input A, B, C,
        output F
    );

    assign F = ~A & ~ B & C | ~A & B & ~C | A & ~B & C | A &
         B & ~C | A & B & C;
endmodule

module after(
        input A, B, C,
        output F
    );

    assign F = ~A & (B ^ C) | A & B | A & C;
endmodule
```

Part 2 Verilog Code

And utilized the following testbench to ensure that it was correct via the waveform above:

```
module part_2_sim();
    parameter NUMIN = 4;
    reg[NUMIN - 1:0] count;
    integer i;

    reg a, b, c, d;
    wire f;

    // replace before with after to check that schematic out
    before UUT(.A(a), .B(b), .C(c), .D(d), .OUT(f));

    initial begin
        count = 0;
        for (i = 0; i < 2**NUMIN; i = i + 1) begin
            assign a = count[3];
            assign b = count[2];
            assign c = count[1];
            assign d = count[0];

            count = count + 1;
            #10;
        end
    end

endmodule
```

Part 2 Testbench

Part 3

Given the following truth table: We have to find out the funny equation. Therefore doing a simple addition of all of the 1s I get the following result:

$$A' * B' * C * D' + A' * B * C * D' + A' * B * C * D + A * B' * C * D'$$

And with the following simplification path we can get the Sums of Product form:

$A' * B' * C * D' + A' * B * C * D' + A' * B * C * D + A * B' * C * D'$	Start
$B' * C * D' * (A' + A) + A' * B * C * D' + A' * B * C * D$	Distributive Law
$B' * C * D' * (1) + A' * B * C * D' + A' * B * C * D$	Complement Law
$B' * C * D' + A' * B * C * D' + A' * B * C * D$	Identity Law
$B' * C * D' + A' * B * C * (D' + D)$	Distributive Law
$B' * C * D' + A' * B * C * (1)$	Complement Law
$B' * C * D' + A' * B * C$	Identity Law
$C * (B' * D' + A' * B)$	Distributive Law

A	B	C	D	F
F	F	F	F	F
F	F	F	T	F
F	F	T	F	T
F	F	T	T	F
F	T	F	F	F
F	T	F	T	F
F	T	T	F	T
F	T	T	T	T
T	F	F	F	F
T	F	F	T	F
T	F	T	F	T
T	F	T	T	F
T	T	F	F	F
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F

```
module before(
    assign A, B, C, D,
    output F
  );

  assign F = ~A&~B&C&~D | ~A&B&C&~D | ~A&B&C&D | A&~B&C&~D;

endmodule

module simplified(
    input A, B, C, D,
    output F
  );

  assign F = C & (~B & ~D | ~A & B);

endmodule
```

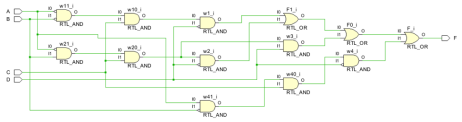
Part 3 Verilog Code

And utilized the following testbench:

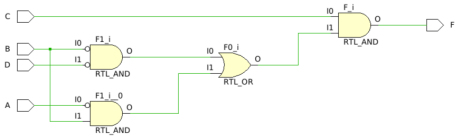
```
module part_3_sim();
    parameter NUMIN = 4;
    reg[NUMIN - 1:0] count;
    integer i;

    reg a, b, c, d;
    wire f;

    // change simplified to before for the before schematic
    simplified UUT(.A(a), .B(b), .C(c), .D(d), .OUT(f));

    initial begin
        count = 0;
        for (i = 0; i < 2**NUMIN; i = i + 1) begin
            assign a = count[3];
            assign b = count[2];
            assign c = count[1];
            assign d = count[0];

            count = count + 1;
            #10;
        end
    end

endmodule
```

Part 3 Testbench

Of which then compiles to the following schematics: Of which then provides the

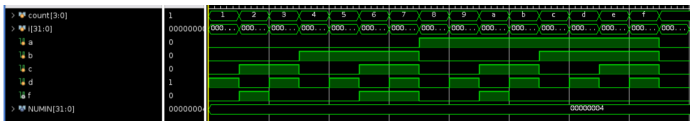


(a) Before Schematic



(b) After Schematic

following waveform, lining up with the truth table given in the lab manual:



Part 4

Given the following table for a function we are to create a Karnaugh map and then create it in Verilog.

A	B	C	D	F
F	F	F	F	F
F	F	F	T	F
F	F	T	F	T
F	F	T	T	F
F	T	F	F	F
F	T	F	T	T
F	T	T	F	T
F	T	T	T	T
T	F	F	F	F
T	F	F	T	F
T	F	T	F	T
T	F	T	T	F
T	T	F	F	T
T	T	F	T	T
T	T	T	F	F
T	T	T	T	F

Therefore, to start with the Karnaugh Map, we need to make it!

AB \ CD	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	1	1	0	0
10	0	0	0	1

Of which we then have to derive the Minterm (Sums of Products) from the 1 terms of which is the following:

$$A' * C * D' + A' * B * D + A * B * C' + B' * C * D'$$

Of which then we need to get the Maxterm which is the negation of the sum of products form of the 0s:

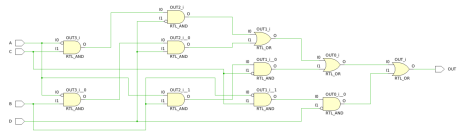
$$\begin{array}{l|l} \neg(B' * C' * D' + A' * C' * D' + D * A' * B' + C * A * B + D * A * B') & \text{MaxTerm Start} \\ (B + C + D)(A + C + D)(D' + A + B)(C' + A' + B')(D' + A' + B) & \text{DeMorgan's Law} \end{array}$$

Now, when entering into Verilog so as to confirm with the schematic and waveform, you get the following:

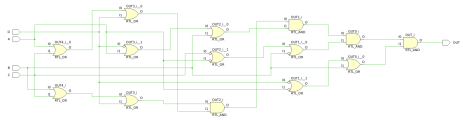
```
module part_4_pos(
        input A, B, C, D,
        output OUT
    );

    assign OUT = (B|C|D)&(A|C|D)&(~D|A|B)&(~C|~A|~B)&(~D|~A|
        B);

endmodule

module part_4_sop(
        input A, B, C, D,
        output OUT
    );

    assign OUT = (~A&C&~D)|(~A&B&D)|(A&B&~C)|(~B&C&~D);

endmodule
```

Part 4 Verilog Code

And got the following schematics for both:



(a) Sum of Products Schematic



(b) Products of Sums Schematic

And, with the following testbench below generated the waveform above.

```
module part_4_sim();
    parameter NUMIN = 4;
    reg[NUMIN - 1:0] count;
    integer i;

    reg a, b, c, d;
    wire f;

    part_4_pos UUT(.A(a), .B(b), .C(c), .D(d), .OUT(f));

    initial begin
        count = 0;
        for (i = 0; i < 2**NUMIN; i = i + 1) begin
            assign a = count[3];
            assign b = count[2];
            assign c = count[1];
            assign d = count[0];

            count = count + 1;
            #10;
        end
    end

endmodule
```

Part 4 Testbench

Part 5

We are given a 4-input system of A, B, C, and D with the following truth table:

A	B	C	D	$OUTPUT$
F	F	F	F	X
F	F	F	T	F
F	F	T	F	F
F	F	T	T	T
F	T	F	F	X
F	T	F	T	F
F	T	T	F	F
F	T	T	T	T
T	F	F	F	T
T	F	F	T	T
T	F	T	F	T
T	F	T	T	F
T	T	F	F	X
T	T	F	T	T
T	T	T	F	X
T	T	T	T	F

Of which then would be converted to the following Karnaugh Map:

AB \ CD	00	01	11	10
00	-	0	1	0
01	-	0	1	0
11	-	1	0	-
10	1	1	0	1

After which we will then get the Sum of Product form from the 1-values on the above Karnaugh Map:

$$A * D' + A' * C * D + A * C'$$

Of which for the Products of Sum form, we get it from the 0-values on the above Karnaugh Map, via ignoring the shaded regions on the map:

$$\neg(A' * C' + A * C * D + A' * C * D')$$

After which we must then negate it using DeMorgan's Law as well as simplify it to be a Product of Sums:

$$\begin{array}{l|c} \neg(A' * C' + A * C * D + A' * C * D') & \text{Start} \\ (A + C) * (A' + C' + D') * (A + C' + D) & \text{DeMorgan's Law} \end{array}$$

Of which then gives us our final result for the Product of Sums form:

$$(A + C) * (A' + C' + D') * (A + C' + D)$$

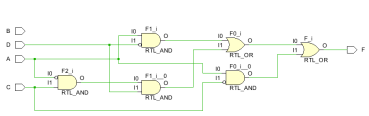
Now, to convert them into the Verilog code, so as to verify that they are correct via the waveform, as well as to see the schematic forms. The verilog code would be the following:

```
module sop(
        input A, B, C, D,
        output F
    );

    assign F = A & ~D | ~A & C & D | A & ~ C;

endmodule

module pos(
        input A, B, C, D,
        output F
    );

    assign F = (A | C) & (~A | ~C | ~D) & (A | ~C | D);

endmodule
```

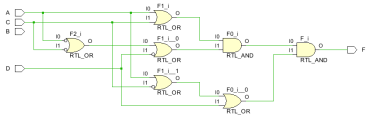
Part 5 Verilog Code

And of which generated the following waveform, and 2 schematics:

(a) Waveform for both Schematics



(a) Sums of Products Schematic



(b) Product of Sums Schematic

Of which were generated via the following testbench:

```
module part_5_sim();
    parameter NUMIN = 4;
    reg[NUMIN - 1:0] count;
    integer i;

    reg a, b, c, d;
    wire f;

    // replace pos with sop for sop form
    part_5_pos UUT(.A(a), .B(b), .C(c), .D(d), .F(f));

    initial begin
        count = 0;
        for (i = 0; i < 2**NUMIN; i = i + 1) begin
            assign a = count[3];
            assign b = count[2];
            assign c = count[1];
            assign d = count[0];

            count = count + 1;
            #10;
        end
    end

endmodule
```

Part 5 Testbench

Conclusion

The entire experiment was a glaring success. The results were perfectly as expected, and all tests/checks agreed with one another harmoniously. Alongside this, the testing of the gate logic helped with my understanding of how they function as well as cemented them with the lab reports summarization.