CS 207 Digital Logic - Spring 2020
Lab 3

James Yu

Mar. 4, 2020

Objective

1. Try UDP design.
2. Test the correctness of Karnauph map minimization.

Lab Exercise Submission

1. You should name all source code as instructed. Mis-named files will not be recognized.
2. You should submit all source code files with an extension "`.v`".
3. You should submit all source code directly into the sakai system below. **Do not compress them into one folder.**

   `https://sakai.sustech.edu.cn/portal/site/ebf68254-68c5-4cfe-9d42-b26758f854ee`
4. Lab exercises should be submitted before the deadline, typically one week after the lab session. No late submission policy applies to lab exercises.

1 UDP Design

1.1 Exercise 1

In the last lab session, we test the SOP and POS transformation of Boolean function $(b + d)(a' + b' + c)$. In this exercise, please write a UDP to construct the truth table presented by this Boolean function, defined as follows:

udp.v

```
primitive udp(f, a, b, c, d);
// Primitive Code
endprimitive
```

You can ignore the cases where inputs are `x` or `z`. Write a testbench for the UDP that outputs all possible combinations of a, b, c, and d.

Assignment
Save the source code in **udp.v**. Upload this file to Sakai under **Assignments → Lab Exercise 3**.

1.2 Exercise 2

Implement a 4-to-1 multiplexer, whose output data `Y` is determined by two input selection signals `S1` and `S2`, as well as four data signals `A`, `B`, `C`, and `D` as follows:

mux.v

```
primitive mux(Y, A, B, C, D, S1, S2);
// Primitive Code
endprimitive
```

The truth table is as follows: For instance, let `ABCD` be `0010`. If `S1 S0` is `10`, the output `Y` is `1`, or the value of `C`.

S1	S2	Y
0	0	A
0	1	B
1	0	C
1	1	D

Assignment
Save the source code in **mux.v**. Upload this file to Sakai under **Assignments → Lab Exercise 3**.

2 Karnaugh Map Minimization

2.1 Exercise 3

Simplify $F(x, y, z) = \sum(1, 3, 4, 6)$. Write a module at gate level design.

Solution

We first draw a three-variable Karnaugh map as follows:

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Then the corresponding minterms are labeled with 1's.

x \ yz	00	01	11	10
0		1	1	
1	1			1

Try to combine as much cells as possible to construct prime implicants. The first one is intuitive.

x \ yz	00	01	11	10
0		1	1	
1	1			1

Note that edges are also adjacent to each other:

x \ yz	00	01	11	10
0		1	1	
1	1			1

In the first row, the cells in the implicant always have $x = 0$ and $z = 1$, rendering $x'z$. Similarly, the second implicant is xz'. $f = x'z + xz'$. Accordingly, we have the gate level design as follows:

kmap.v

```
module kmap(F, inX, inY, inZ);
input inX, inY, inZ;
output F;
wire notX, notZ;
wire A, B;

not not_1 (notX, inX);
not not_2 (notZ, inZ);
and and_1 (A, notX, inZ);
and and_2 (B, notZ, inX);
or  or_1  (F,  A, B);
endmodule
```

and the testbench:

kmap.v

```
module kmap_tb;
reg inX, inY, inZ;
wire F;

kmap kmap_1(F, inX, inY, inZ);

initial begin
  $monitor("%3t: X is %b, Y is %b, Z is %b, F is %b.", $time, inX, inY, inZ, F);
  # 5  inX = 0; inY = 0; inZ = 0;
  # 5  inX = 0; inY = 0; inZ = 1;
  # 5  inX = 0; inY = 1; inZ = 0;
  # 5  inX = 0; inY = 1; inZ = 1;
  # 5  inX = 1; inY = 0; inZ = 0;
  # 5  inX = 1; inY = 0; inZ = 1;
  # 5  inX = 1; inY = 1; inZ = 0;
  # 5  inX = 1; inY = 1; inZ = 1;
  # 10 $finish;
end
endmodule
```

2.2 Exercise 4

Simplify $F(A, B, C, D) = \sum(0, 6, 8, 13, 14), d(A, B, C, D) = \sum(2, 4, 10)$. Write a module at gate level design that corresponds to the simplified Boolean function as follows:

kmap.v

```v
module kmap(F, A, B, C, D);
// Module Code
endmodule
```

Write a testbench for the module that outputs all possible combinations of A, B, C, and D.

Assignment

Save the source code in **kmap.v**. Upload this file to Sakai under **Assignments → Lab Exercise 3**.

Assignment

When you finished Lab Exercise 3, there should be three .v files in the Sakai system: **udp.v**, **mux.v**,and **kmap.v**.