CPE166 Lab 2 Part 2
By: Prof. Pang

Lab 2

Part 2: 8-bit Carry Select Adder

The purpose of this experiment is to design the 8-bit carry select adder circuit shown below.

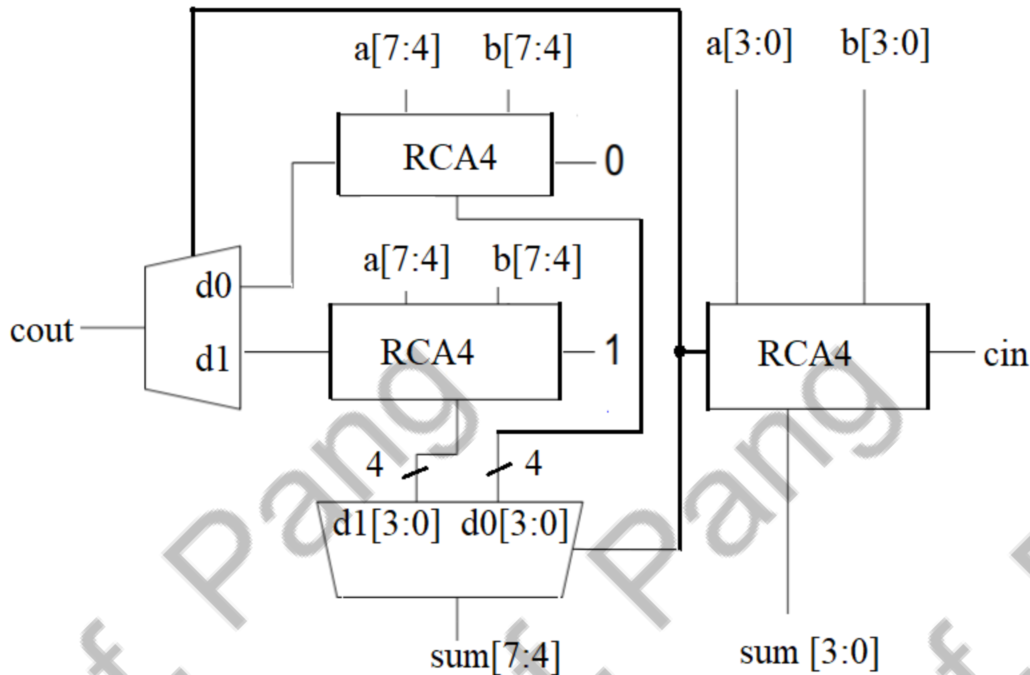


Figure 1. 8-bit carry-select adder circuit

The 8-bit carry-select adder circuit above consists of three 4-bit ripple carry adders (RCA4) and two multiplexers. One of 4-bit ripple carry adders assumes the carry-in to be zero, and the other assumes the carry-in to be one. The multiplexers select the correct sum and the carry-out based on the known carry-in value. This method is faster than the 8-bit ripple carry adder approach to obtain the 8-bit addition results.

CPE166 Lab 2 Part 2
By: Prof. Pang

Lab Procedure

Step 1. 4-bit Ripple Carry Adder Design

The 4-bit ripple carry adder circuit is shown below.

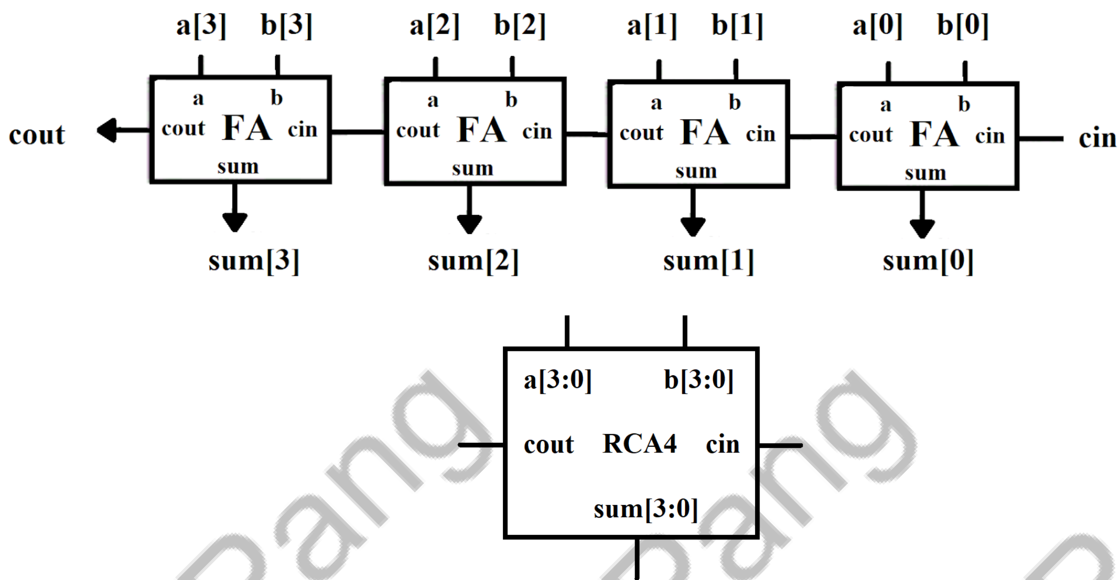


Figure 2. The 4-bit ripple carry adder circuit and block diagram

You can reuse the full adder design module FA you designed in the Lab2 part 1 for your RCA4 design in this part.

Therefore, you need to include the half adder ha.v file, the full adder fa.v file, and the 4-bit ripple carry adder rca.4 file in the design of step 1. In addition, you need to write a testbench for the rca4 design and run simulations.

CPE166 Lab 2 Part 2
By: Prof. Pang

Step 2. Multiplexer (MUX) Design

The 2-to-1 multiplexer consists of two inputs D0 and D1, one selection input S and one output Y. According to the logic value of the selection signal S, D0 or D1 will be passed to the output.

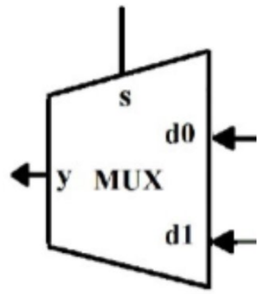


Figure 3. 2-to-1 multiplexer diagram

A **Verilog if statement** is used to choose which statement should be executed depending on the conditional expression.

Simplified Sample Syntax

```
reg   y;
always@(signal1 or signal2 or signal3)
begin
  if (conditional expression)
    y= statement1;
  else
     y= statement2;
end
```

You can design the 2-to-1 multiplexer circuit by using if…else statement in Verilog and also write a testbench to run simulations. Verify by yourself that your simulation results match with the values shown below.

Inputs			Output
s	d0	d1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

The above table can also be simplified into the following format.

s	y
0	d0
1	d1

Step 3. MUXB Design

The MUXB consists of two 4-bit inputs D0 and D1, one 1-bit selection input S and one 4-bit output Y. According to the logic value of the selection signal S, D0 or D1 will be passed to the output.

The Verilog design of this part will look very similar to the design in step 2. The major difference is that you must declare the inputs d1, d0 and the output y as 4-bit data.

Simplified Sample Syntax

```
reg [3:0]   y;
always@(signal1 or signal2 or signal3)
begin
  if (conditional expression)
    y= statement1;
  else
     y= statement2;
end
```

You can design the 2-to-1 multiplexer circuit by using if…else statement in Verilog and write a testbench to run simulations. Verify by yourself that your simulation results match with the truth table shown below.

s	y[3:0]
0	d0[3:0]
1	d1[3:0]

CPE166 Lab 2 Part 2
By: Prof. Pang

Step 4. Final 8-bit Carry Select Adder Design

Carry-Select Adder (CSA8) diagram:

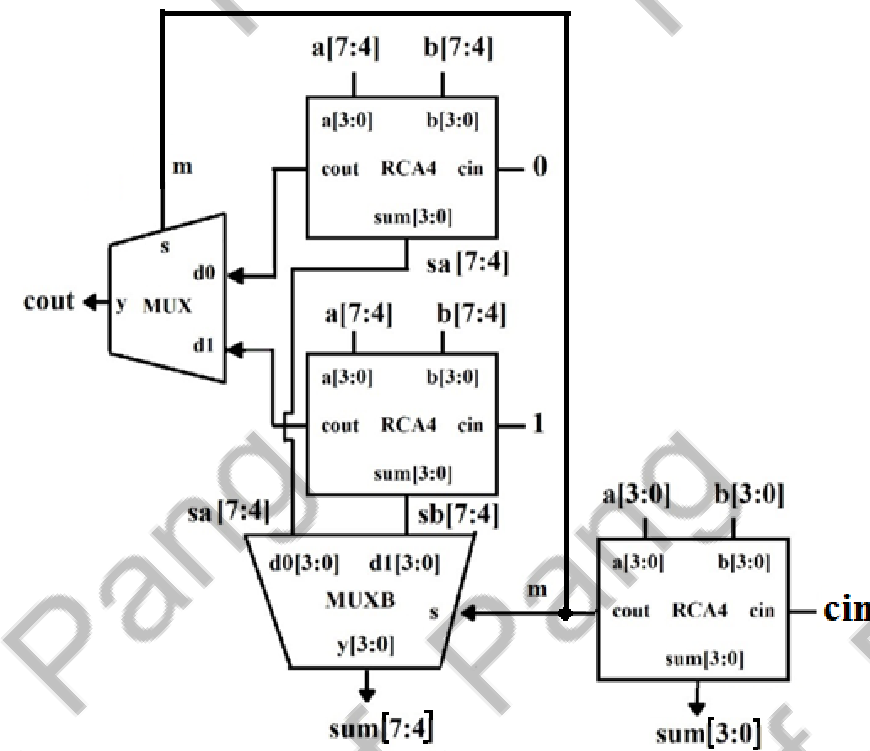


Figure 4. 8-bit carry select adder circuit

Design the 8-bit carry-select adder (CSA8) circuit above by using three 4-bit ripple carry adders (RCA4), one MUX and one MUXB. In addition, write testbench for CSA8 to run simulations.

Demo Requirement

You need to demonstrate the final simulation waveform of the CSA8 design to your lab instructor.

You need to download this design to the FPGA board and demonstrate it to your lab instructor.

CPE166 Lab 2 Part 2
By: Prof. Pang

Note: Before starting this experiment, all the necessary knowledge required to complete this work has been introduced in the CPE166 lecture session. The following examples are for you to refresh your learning.

Sample Verilog Codes:

```
module mux4( d0, d1, d2, d3, s0, s1, y);

input   d0, d1, d2, d3, s0, s1;

output  y;

reg       y;

always@(d0 or d1 or d2 or d3 or s0 or s1)

begin

   if ( s1==0 && s0==0)

       y = d0;

  else if ( s1==0 && s0==1)

      y = d1;

  else if ( s1==1 && s0==0)

      y = d2;

  else

      y = d3;

end

endmodule
```

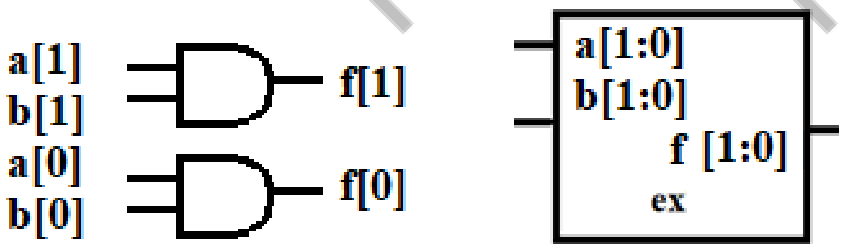
CPE166 Lab 2 Part 2
By: Prof. Pang

```
`timescale 1ns/1ps

module mux4_tb;

reg     d0, d1, d2, d3, s0, s1;

wire   y;

mux4    uut ( d0, d1, d2, d3, s0, s1, y);

initial

begin

   $monitor($time, " ns, d0=%b, d1=%b, d2=%b, d3=%b, s1=%b, s0=%b, y=%b", d0, d1, d2, d3, s1, s0, y);

    d0=0; d1=0; d2=0; d3=0;  s1=0; s0=0;

   #10    d0=1; d1=0; d2=0; d3=0;  s1=0; s0=0;

   #10    d0=0; d1=0; d2=0; d3=0;  s1=0; s0=1;

   #10    d0=0; d1=1; d2=0; d3=0;  s1=0; s0=1;

   #10    d0=0; d1=0; d2=0; d3=0;  s1=1; s0=0;

   #10    d0=0; d1=0; d2=1; d3=0;  s1=1; s0=0;

   #10    d0=0; d1=0; d2=0; d3=0;  s1=1; s0=1;

   #10    d0=0; d1=0; d2=0; d3=1;  s1=1; s0=1;

   #10    $stop;

end

endmodule
```

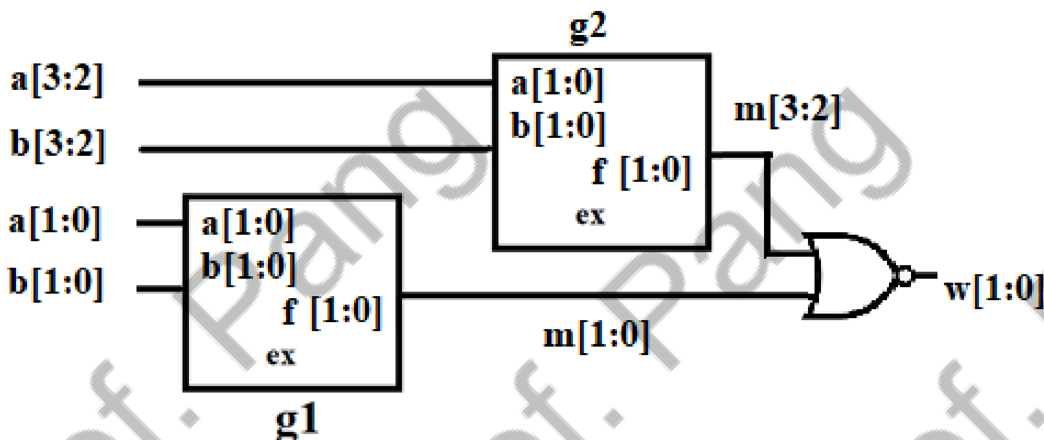
CPE166 Lab 2 Part 2
By: Prof. Pang

Sample Circuits:

(1).



(2).



Sample Implementation in Verilog:

```
module ex (a, b, f);

input   [1:0]  a, b;

output  [1:0]  f;

assign   f = a & b;

endmodule
```

CPE166 Lab 2 Part 2
By: Prof. Pang

```
`timescale 1ns/1ps

module  ex_tb;

reg   [1:0]    a, b;

wire  [1:0]   y;

ex    uut ( a, b, y);

integer   k;

initial   begin

   $monitor($time, " ns, a=%b, b=%b,  y=%b", a, b,  y);

    for (k=0; k<16; k=k+1)

    begin

        {a, b} = k;

         #5;

     end

     #5  $stop;

end

endmodule



module ex2 (a, b, w);

input    [3:0]  a, b;

output  [1:0]  w;

wire      [3:0]  m;

ex     g1 ( .a ( a[1:0] ), .b( b[1:0] ),  .f (  m[1:0] ) );

ex     g2 ( .a ( a[3:2] ), .b( b[3:2] ),  .f (  m[3:2] ) );

assign    w =   ~ ( m[3:2] |  m{1:0] );

endmodule
```