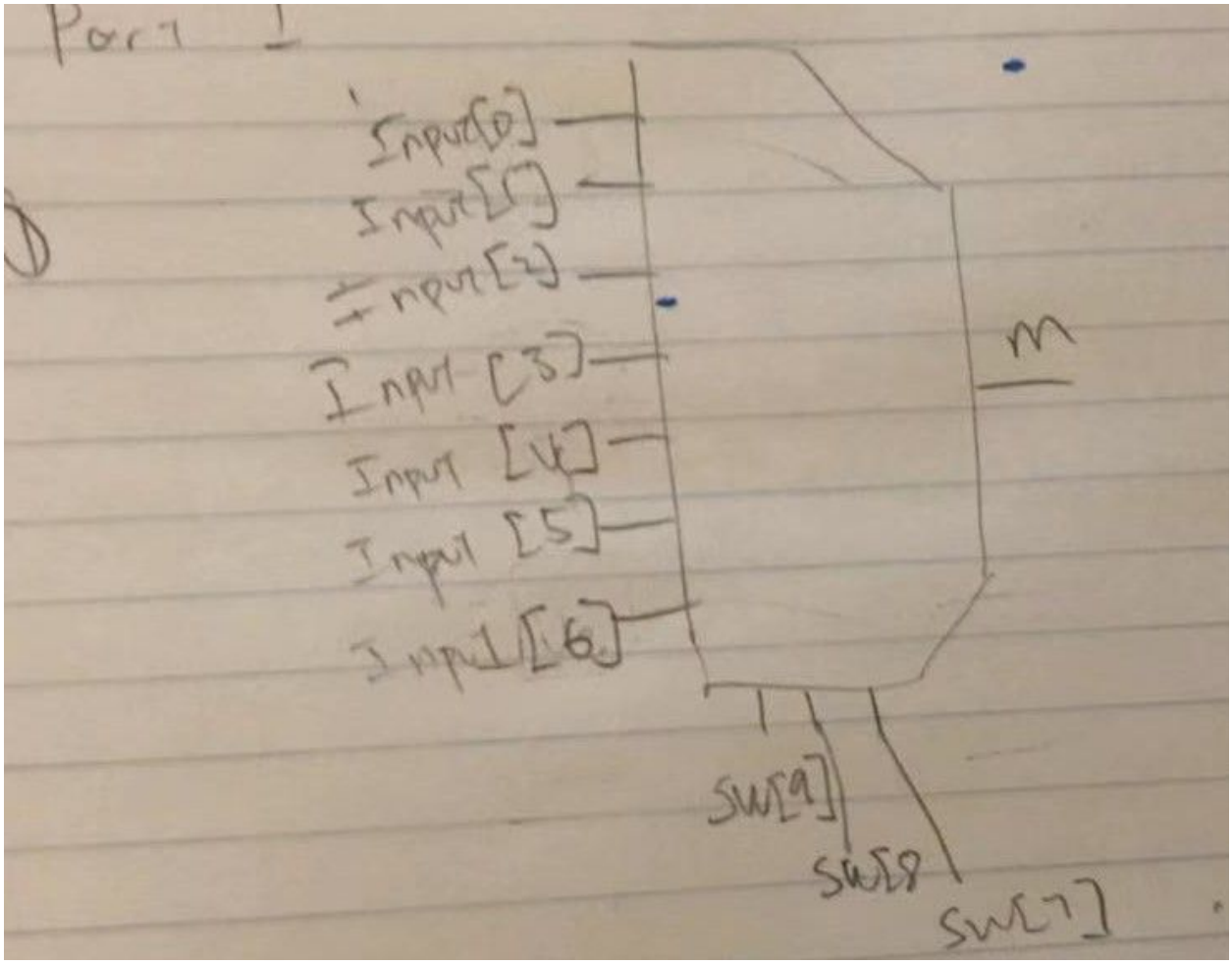
Ryan chang csc258 prelab3
Part 1
Schematic

1. 

2.
VERILOG CODE

* rememebr that its called seven mux and idk if u can do line 3?

```
module mux(SW,LEDR);
	input [9:0] SW;
	output [1:0] LEDR;

	mux7to1  u0(
 .Input(SW[6:0]),
 .MuxSelect(SW[9:7]),
 .m(LEDR[0])
);
```

```
endmodule

module mux7to1(Input, MuxSelect, m);
	input [6:0] Input;
	input [2:0] MuxSelect;
	output m;
	reg m;

	always @(*)
	begin
		case(MuxSelect)
			3'b000: m = Input[0];
			3'b001: m = Input[1];
			3'b010: m = Input[2];
			3'b011: m = Input[3];
			3'b100: m = Input[4];
			3'b101: m = Input[5];
			3'b110: m = Input[6];
			default: m = 1'b0;
		endcase
	end
endmodule
```

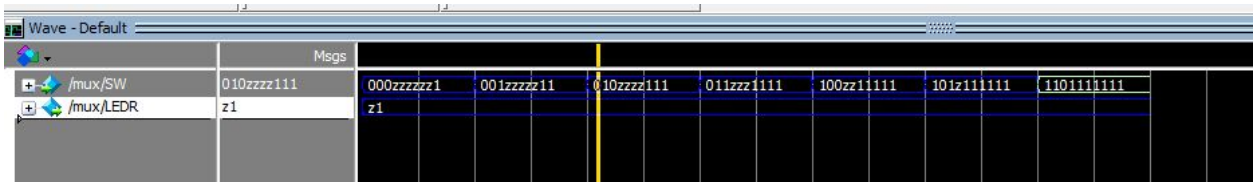
SIM CODE

```
# Set the working dir, where all compiled Verilog goes.
vlib work

# Compile all verilog modules in mux.v to working dir;
# could also have multiple verilog files.
vlog -timescale 1ns/1ns mux7to1.v

# Load simulation using mux as the top level simulation module.
vsim mux

# Log all signals and add some signals to waveform window.
log {/*}
# add wave {/*} would add all items in top level simulation module.
add wave {/*}
#the LEDR should all be 1 for all 70ns
```

```
#test input0
force {SW[0]} 1
force {SW[9]} 0
force {SW[8]} 0
force {SW[7]} 0
run 10ns

#test input1
force {SW[1]} 1
force {SW[9]} 0
force {SW[8]} 0
force {SW[7]} 1
run 10ns

#test input2
force {SW[2]} 1
force {SW[9]} 0
force {SW[8]} 1
force {SW[7]} 0
run 10ns

#test input3
force {SW[3]} 1
force {SW[9]} 0
force {SW[8]} 1
force {SW[7]} 1
run 10ns

#test input4
force {SW[4]} 1
force {SW[9]} 1
force {SW[8]} 0
force {SW[7]} 0
run 10ns

#test input5
force {SW[5]} 1
force {SW[9]} 1
force {SW[8]} 0
force {SW[7]} 1
run 10ns

#test input6
```

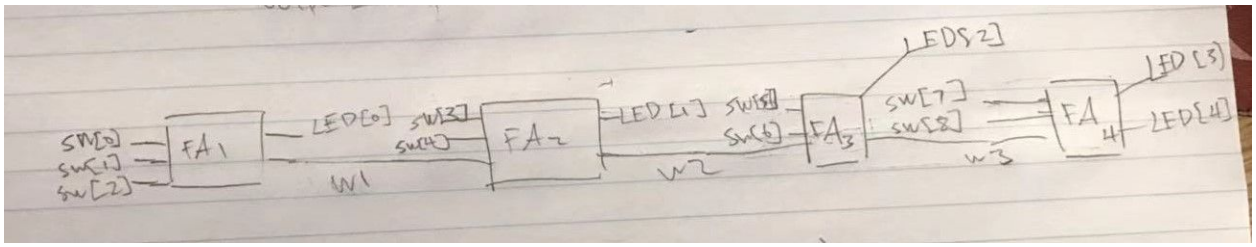
force {SW[6]} 1
force {SW[9]} 1
force {SW[8]} 1
force {SW[7]} 0
run 10ns

Sim screenshot



Part 2

Schematic



VERILOG CODE

```
module fourbitadder(SW,LEDR);
        input [9:0] SW;
        output [9:0] LEDR;
        wire w1, w2, w3;

        fullAdder fa1(
 .a(SW[0]),
 .b(SW[1]),
 .cin(SW[2]),
 .s(LEDR[0]),
 .cout(w1)
);

fullAdder fa2(
 .a(SW[3]),
 .b(SW[4]),
 .cin(w1),
 .s(LEDR[1]),
 .cout(w2)
```

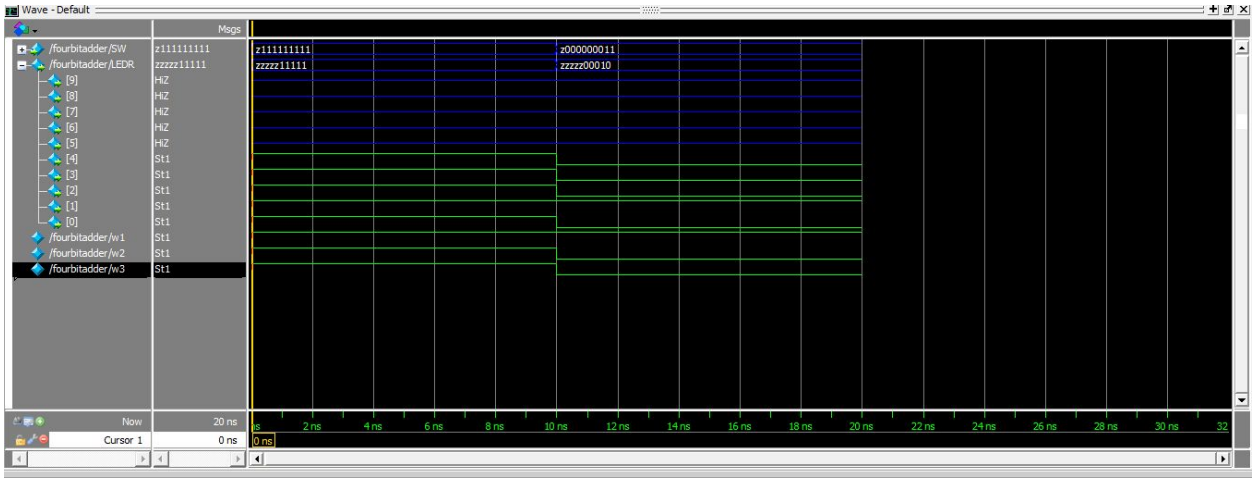
```
);

fullAdder fa3(
 .a(SW[5]),
 .b(SW[6]),
 .cin(w2),
 .s(LEDR[2]),
 .cout(w3)
);

fullAdder fa4(
 .a(SW[7]),
 .b(SW[8]),
 .cin(w3),
 .s(LEDR[3]),
 .cout(LEDR[4])
);

endmodule

module fullAdder(a, b, cin, s, cout);
	input a;
	input b;
	input cin;
	output s;
	output cout;

	assign cout = a & b | a & cin | b & cin;
	assign s = a ^ b ^ cin;
endmodule
```

Vsim code

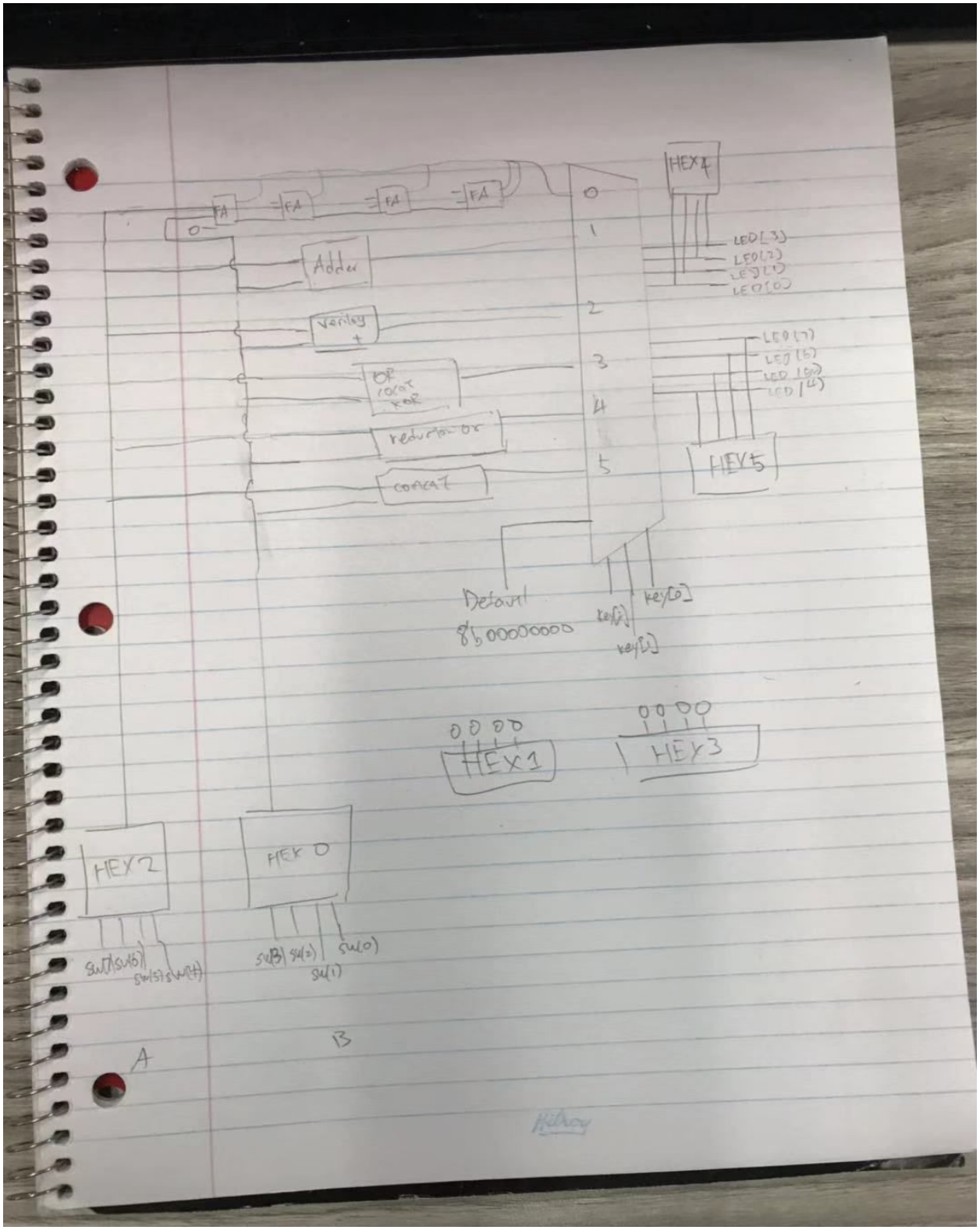
```
# Set the working dir, where all compiled Verilog goes.
vlib work

# Compile all verilog modules in mux.v to working dir;
# could also have multiple verilog files.
```

```
vlog -timescale 1ns/1ns fourbitadder.v

# Load simulation using mux as the top level simulation module.
vsim fourbitadder

# Log all signals and add some signals to waveform window.
log {/*}
# add wave {/*} would add all items in top level simulation module.
add wave {/*}
#test case 1: addng 2#1111 + 2#1111 = 2#11111 also 10#15 + 10#15 = 10#30
force {SW[0]} 1
force {SW[1]} 1
force {SW[2]} 0
force {SW[3]} 1
force {SW[4]} 1
force {SW[5]} 1
force {SW[6]} 1
force {SW[7]} 1
force {SW[8]} 1

run 10ns


#test case 2: addng 2#0001 + 2#0001 = 2#0010  also  10#1 + 10#1 = 10#2
force {SW[0]} 1
force {SW[1]} 1
force {SW[2]} 0
force {SW[3]} 0
force {SW[4]} 0
force {SW[5]} 0
force {SW[6]} 0
force {SW[7]} 0
force {SW[8]} 0

run 10ns
```

SIM SCREENSHOT


Wave - Default
Msgs
/fourbitadder/SW
/fourbitadder/LEDR
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
[0]
/fourbitadder/w1
/fourbitadder/w2
/fourbitadder/w3
z111111111
zzzzz11111
HiZ
St1
z111111111
zzzzz11111
z000000011
zzzzz00010
Now
20 ns
Cursor 1
0 ns
0 ns
2 ns
4 ns
6 ns
8 ns
10 ns
12 ns
14 ns
16 ns
18 ns
20 ns
22 ns
24 ns
26 ns
28 ns
30 ns
32

PArt 3
Schematic



Verilog Code

```
module mainalu(SW, KEY, LEDR, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
	input [7:0] SW;
	input [2:0] KEY;
	output [7:0] LEDR;

	output[6:0] HEX0;
    output[6:0] HEX1;
    output[6:0] HEX2;
    output[6:0] HEX3;
    output[6:0] HEX4;
    output[6:0] HEX5;

    wire[7:0] Case0;
    wire[7:0] Case1;
    wire[7:0] Case2;
    wire[7:0] Case3;
    wire[7:0] Case4;
    wire[7:0] Case5;

    reg[7:0] Out;


   fourbitadder fba1(
	.A(SW[7:4]),
	.B(4'b0001),
	.Cin(1'b0),
	.Sum(Case0[7:0])
   );

   fourbitadder fba1(
	.A(SW[7:4]),
	.B(SW[3:0]),
	.Cin(1'b0),
	.Sum(Case1[7:0])
   );

   verisum v1(
	.A(SW[7:4]),
	.B(SW[3:0]),
	.Sum(Case2[7:0])
   );
```

```
veriXOR v2(
	.A(SW[7:4]),
	.B(SW[3:0]),
	.Sum(Case3[7:0])
);

orplusredor o3(
	.A(SW[7:4]),
	.B(SW[3:0]),
	.Sum(Case4[7:0])

);

concat c1(
	.A(SW[7:4]),
	.B(SW[3:0]),
	.Sum(Case5[7:0])
);


always @(*)
begin
	case (KEY[2:0])
		3'b000: Out = Case0;
		3'b001: Out = Case1;
		3'b010: Out = Case2;
		3'b011: Out = Case3;
		3'b100: Out = Case4;
		3'b101: Out = Case5;
		default: Out = 8'b00000000;
	endcase
end

assign LEDR = Out;

sevenseg u6 (
	.Data(SW[3:0]),
	.Display(HEX0[6:0])

	);

	sevenseg u7 (
	.Data(SW[7:4]),
```

```
        .Display(HEX2[6:0])
        );


        sevenseg u8 (
        .Data(Out[3:0]),
        .Display(HEX4[6:0])
        );

        sevenseg u9 (
        .Data(Out[7:4]),
        .Display(HEX5[6:0])

        );

        assign HEX1[6:0] = 7'b0000000;
        assign HEX3[6:0] = 7'b0000000;

endmodule


module fourbitadder(A, B, Cin, Sum);
        input [3:0] A, B;
    input Cin;
        output [7:0] Sum;
        wire w1, w2, w3;


        fullAdder fa1(
 .a(A[0]),
 .b(B[0]),
 .cin(Cin),
 .s(Sum[0]),
 .cout(w1)
);

   fullAdder fa2(
 .a(A[1]),
 .b(B[1]),
 .cin(w1),
 .s(Sum[1]),
 .cout(w2)
);
```

```
   fullAdder fa3(
 .a(A[2]),
 .b(B[2]),
 .cin(w2),
 .s(Sum[2]),
 .cout(w3)
);

   fullAdder fa4(
 .a(A[3]),
 .b(B[3]),
 .cin(w3),
 .s(Sum[3]),
 .cout(Sum[4])
);
   assign Sum[7:5] = 3'b000;

endmodule

module fullAdder(a, b, cin, s, cout);
        input a;
        input b;
        input cin;
        output s;
        output cout;

        assign cout = a & b | a & cin | b & cin;
        assign s = a ^ b ^ cin;
endmodule


module verisum(A, B, Result);
        input[3:0] A;
        input[3:0] B;
        output[7:0] Result;

        assign Result[3:0] = A + B;
    assign Result[7:4] = 4'b0000;


endmodule
```

```
module orplusredor(A, B, Result);
	input[3:0] A;
	input[3:0] B;
	output[7:0] Result;

	assign Result[0] = | {A,B};
    assign Result[7:1] = 7'b0000000;


endmodule

module concant(A, B, Result);
	input[3:0] A;
	input[3:0] B;
	output[7:0] Result;

    assign Result[7:0] = {A, B};


endmodule


module sevenseg (Display, Data);

	input[3:0] Data;

	output[6:0] Display;

	assign Display[0] = ~Data[3] & ~Data[2] & ~Data[1] &  Data[0] |~Data[3] &  Data[2] & ~Data[1] & ~Data[0] | Data[3] &  Data[2] & ~Data[1] &  Data[0] |Data[3] & ~Data[2] &  Data[1] & Data[0];

	assign Display[1] = ~Data[3] &  Data[2] & ~Data[1] &  Data[0] |Data[3] &  Data[2] & ~Data[1] & ~Data[0] |Data[3] &  Data[1] &  Data[0] |  Data[2] &  Data[1] & ~Data[0];


	assign Display[2] =  Data[3] &  Data[2] & ~Data[1] & ~Data[0] |~Data[3] & ~Data[2] & Data[1] & ~Data[0] | Data[3] &  Data[2] &  Data[1];


	assign Display[3] = ~Data[3] &  Data[2] & ~Data[1] & ~Data[0] | ~Data[2] & ~Data[1] & Data[0] |Data[2] &  Data[1] &  Data[0] | Data[3] & ~Data[2] &  Data[1] & ~Data[0];
```

```
        assign Display[4] = ~Data[3] &  Data[0] | ~Data[3] &  Data[2] & ~Data[1]| ~Data[2] &
~Data[1] &  Data[0];

        assign Display[5] = ~Data[3] & ~Data[2]  &  Data[0] |~Data[3] & ~Data[2] &  Data[1]
|~Data[3] & Data[1] &  Data[0] |Data[3] &  Data[2] & ~Data[1] &  Data[0];


        assign Display[6] = ~Data[3] & ~Data[2]  & ~Data[1] |~Data[3] &  Data[2] &  Data[1] &
Data[0] | Data[3] &  Data[2] & ~Data[1] & ~Data[0];


endmodule
```