426 - HW2

Course	🏀 VLSI IMPLEMENTATION
Complete	☑️
Due date	@February 5, 2024
STUDENT	CHASE LOTITO
TYPE	PROBLEM SET

Q1

A:

```
module mux21_gate(SEL, A, B, OUT);

    // Initialize I/O
    input SEL, A, B;
    output OUT;

    // Intermediate Signals
    wire inv1, a1, a2;

    // Gate-Level Instantiation
    not INV (inv1, SEL);
    and AND1 (a1, A, inv1);
    and AND2 (a2, B, SEL);
    or OR2 (OUT, a1, a2);

endmodule
```

B:

mux21_continuous.v

```
// Chase Lotito
// Q1-B
    // 2:1 Multiplexer with Continuous Assignment Statements

module mux21_continuous( SEL, A, B, OUT );
    // I/O
    input SEL, A, B;
    output OUT;

    // Continuous Assignment Approach
    assign OUT = (~SEL & A) | (SEL & B );
endmodule
```

C:

mux21_always.v

```
// Chase Lotito - SIUC
// Q1-C
    // 2:1 Mux - Always Procedure Block

module mux21_always ( SEL, A, B, OUT );

    // I/O
    input SEL, A, B;
    output OUT;

    // Procedural
    reg OUT;

    // Always Block Approach
    always @(SEL or A or B) begin
        if(SEL)
            OUT = B;
        else
            OUT = A;
    end

endmodule
```

Q2

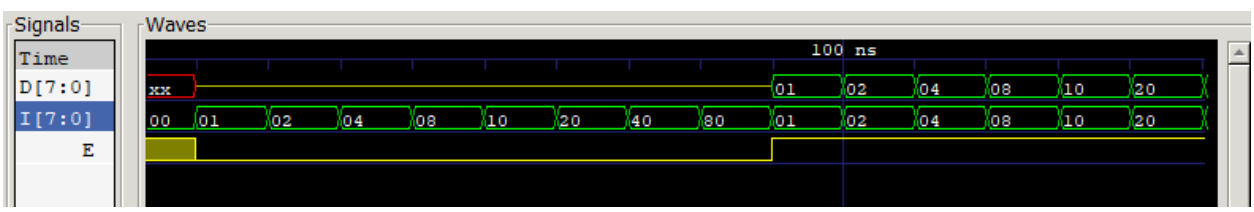
A:

eight_bit_buffer.v

```
// Chase Lotito - SIUC
// Q2
    // Tristate buffer with Primivitives
    // EIGHT BIT BUFFER

module EightBitBuffer ( E, I, D );

    // I/O
    input E;
    input [7:0] I;
    output [7:0] D;

    // Note I don't need intermediate wires!

    // Instantiate the buffers!
    bufif1 BUF0(D[0], I[0], E);
    bufif1 BUF1(D[1], I[1], E);
    bufif1 BUF2(D[2], I[2], E);
    bufif1 BUF3(D[3], I[3], E);
    bufif1 BUF4(D[4], I[4], E);
    bufif1 BUF5(D[5], I[5], E);
    bufif1 BUF6(D[6], I[6], E);
    bufif1 BUF7(D[7], I[7], E);

endmodule
```

```
// Q2 Testbench

`include "eight_bit_buffer.v"
`timescale 1ns / 1ns

module tb();

// I/O
reg [7:0] I;
reg E;
wire [7:0] D;
integer i;

// GKTWAVE
initial begin : GKTWAVE
    $dumpfile("tb.vcd");
    $dumpvars(0,tb);
end

// Start-Up
initial begin : stimmichecks
    // Clear registers
    I = 8'h00;

    $display("EBB - EN OFF");
    // Test Cycle w/ ENABLE OFF
    E = 0;
    for (i = 0; i < 8; i = i + 1) begin
        //#10 I = 8'h00; // clear I
        I[i] = 1;
        $display("I = %b, D = %b", I, D);
    end

    $display("EBB - EN ON");
    // Test Cycle w/ ENABLE ON
    E = 1;
    for (i = 0; i < 7; i = 1 + i) begin
        #10 I = 8'h00; // clear I
        I[i] = 1;
        $monitor("I = %b, D = %b", I, D); //$display caused lagging, just print when signal changes
    end
end

// Instantiate Buffer
EightBitBuffer U1 (

    .E(E),
    .I(I),
    .D(D)

);

endmodule
```

```
EBB - EN OFF
I = 00000001, D = zzzzzzzz
I = 00000011, D = zzzzzzzz
I = 00000111, D = zzzzzzzz
I = 00001111, D = zzzzzzzz
I = 00011111, D = zzzzzzzz
I = 00111111, D = zzzzzzzz
I = 01111111, D = zzzzzzzz
I = 11111111, D = zzzzzzzz
EBB - EN ON
I = 00000001, D = 00000001
I = 00000010, D = 00000010
I = 00000100, D = 00000100
I = 00001000, D = 00001000
I = 00010000, D = 00010000
I = 00100000, D = 00100000
I = 01000000, D = 01000000
PS C:\Users\Chase Lotito\Deskt
```



B:

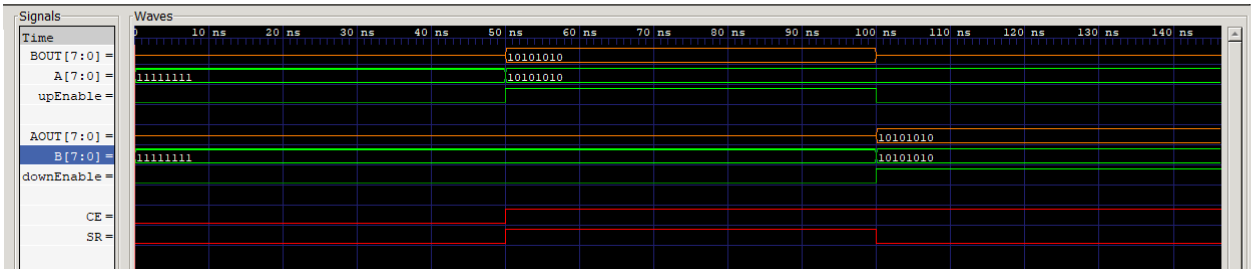
BidirectionalBuffer_8.v

```
// Chase Lotito - SIUC - ECE426
// HW 2 - Q2-B - 8-bit Bidirectional Buffer

module Bidirectional_Buffer_8 (
    CE, SR, A, B, AOUT, BOUT
);

// I/O
input SR, CE;
input [7:0] A;
input [7:0] B;
output [7:0] AOUT;
output [7:0] BOUT;

// Intermediate Wires
wire upEnable;
wire downEnable;
wire _SR;

// Gate-level Model

// ~SR signal
not INV1(_SR, SR);

// SR and CE logic
and AND_upEnable(upEnable, SR, CE);
and AND_downEnable(downEnable, _SR, CE);

// _U for signals moving up
// _D for signals moving down
bufif1 AND0_U(BOUT[0], A[0], upEnable);
bufif1 AND1_U(BOUT[1], A[1], upEnable);
bufif1 AND2_U(BOUT[2], A[2], upEnable);
bufif1 AND3_U(BOUT[3], A[3], upEnable);
bufif1 AND4_U(BOUT[4], A[4], upEnable);
bufif1 AND5_U(BOUT[5], A[5], upEnable);
bufif1 AND6_U(BOUT[6], A[6], upEnable);
bufif1 AND7_U(BOUT[7], A[7], upEnable);
//--------------------------------
bufif1 AND0_D(AOUT[0], B[0], downEnable);
bufif1 AND1_D(AOUT[1], B[1], downEnable);
bufif1 AND2_D(AOUT[2], B[2], downEnable);
bufif1 AND3_D(AOUT[3], B[3], downEnable);
bufif1 AND4_D(AOUT[4], B[4], downEnable);
bufif1 AND5_D(AOUT[5], B[5], downEnable);
bufif1 AND6_D(AOUT[6], B[6], downEnable);
bufif1 AND7_D(AOUT[7], B[7], downEnable);


endmodule
```

tb.v

```
// Testbench for Q2-B - Chase Lotito

`include "BidirectionalBuffer_8.v"
`timescale 1ns/1ns

module tb();

// I/O
reg SR, CE;
reg [7:0] A;
reg [7:0] B;
wire [7:0] AOUT;
wire [7:0] BOUT;

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("tb.vcd");
    $dumpvars(0, tb);
end

// Stimulus
initial begin : stimmychecks
    // Not Enabled
    SR = 0;
    CE = 0;
    A[7:0] = 8'hff;
    B[7:0] = 8'hff;
    #50;

    //Upward Enabled
    SR = 1;
    CE = 1;
    A[7:0] = 8'haa;
    #50;

    //Downward Enabled
    SR = 0;
    CE = 1;
    B[7:0] = 8'haa;

    #50 $finish;
end

// Display!
always @ (SR or CE or A or B) begin
   #10 $display("SR = %b, CE = %b, A = {%b}, B = {%b} | AOUT = {%b}, BOUT = {%b} ", SR, CE, A, B, AOUT, BOUT);
end

Bidirectional_Buffer_8 bb8 (
    .CE(CE),
    .SR(SR),
    .A(A),
    .B(B),
    .AOUT(AOUT),
    .BOUT(BOUT)
);

endmodule
```

```
VCD info: dumpfile tb.vcd opened for output.
SR = 0, CE = 0, A = {11111111}, B = {11111111} | AOUT = {zzzzzzzz}, BOUT = {zzzzzzzz}
SR = 1, CE = 1, A = {10101010}, B = {11111111} | AOUT = {zzzzzzzz}, BOUT = {10101010}
SR = 0, CE = 1, A = {10101010}, B = {10101010} | AOUT = {10101010}, BOUT = {zzzzzzzz}
tb.v:41: $finish called at 150 (1ns)
```


Signals
Waves
Time
BOUT[7:0]
A[7:0]
upEnable
AOUT[7:0]
B[7:0]
downEnable
CE
SR
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns
100 ns
110 ns
120 ns
130 ns
140 ns
10101010
11111111
10101010
10101010
11111111
10101010

Q3

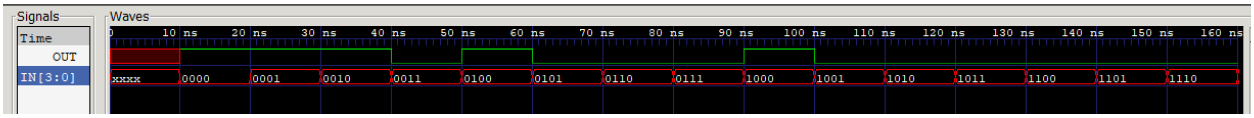
MinorityFunction.v

```
// Chase Lotito - SIUC - SP2024
// ECE426L - Chao Lu
// HW2 - Q3 - Minority Function

module MinorityFunction (
    OUT, IN
);

    // I/O
    input [3:0] IN;
    output OUT;

    // Intermediary wires
    wire a0, a1, a2, a3;
    wire [3:0] _IN;

    // Gate-level design
    not INV0(_IN[0], IN[0]);
    not INV1(_IN[1], IN[1]);
    not INV2(_IN[2], IN[2]);
    not INV3(_IN[3], IN[3]);

    and AND0(a0, _IN[0], _IN[1], _IN[2], _IN[3]);
    and AND1(a1, _IN[0], _IN[1], _IN[2], IN[3]);
    and AND2(a2, _IN[0], _IN[1], IN[2], _IN[3]);
    and AND3(a3, _IN[0], IN[1], _IN[2], _IN[3]);
    and AND4(a4, IN[0], _IN[1], _IN[2], _IN[3]);

    or OR0(OUT, a0, a1, a2, a3, a4);

endmodule
```

tb.v

```
// Lotito - Q3 - Testbench

`include "MinorityFunction.v"
`timescale 1ns/1ns

module tb();

// I/O
reg [3:0] I;
wire O;

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("tb.vcd");
    $dumpvars(0,tb);
end

// STIMULUS
initial begin : stimmychecks
    reg [4:0] stimmyVect;

    for (stimmyVect = 0; stimmyVect < 16; stimmyVect = stimmyVect + 1)
        #10 I[3:0] = stimmyVect;

end

// DISPLAY!
always @ (I) begin
    $monitor("IN: {%b}, OUT: %b", I, O);
end

// Instantiate module
MinorityFunction U1 (
    .OUT(O),
    .IN(I)
);

endmodule
```

```
IN: {0001}, OUT: 1
IN: {0010}, OUT: 1
IN: {0011}, OUT: 0
IN: {0100}, OUT: 1
IN: {0101}, OUT: 0
IN: {0110}, OUT: 0
IN: {0111}, OUT: 0
IN: {1000}, OUT: 1
IN: {1001}, OUT: 0
IN: {1010}, OUT: 0
IN: {1011}, OUT: 0
IN: {1100}, OUT: 0
IN: {1101}, OUT: 0
IN: {1110}, OUT: 0
IN: {1111}, OUT: 0
PS C:\Users\cloti\On
```



Q4

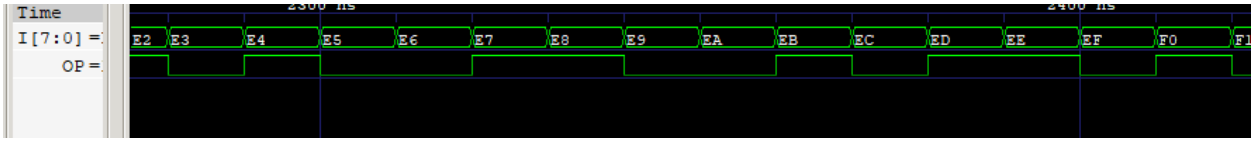
EvenParityGenerator.v

```
// Chase Lotito - SIUC - SP2024
// ECE426 - Chao Lu
// HW2 - Q4 - Even Parity Generator

/*
    This approach is more ad hoc and not derived from a truth table
        1) An XNOR gate is high only for all low inputs or even high inputs
        2) Also send all inputs into OR gate to check if all zero
        3) Send out of XNOR and out of OR into AND to get both even
            and at least one input high!
*/

module EvenParityGenerator (
    OP, I
);

// I/O
input [7:0] I;
output OP;

// Intermediate wires
wire orOut; // output of OR gate
wire xnorOut; // output of XNOR gate

// Gate-Level Design
xnor XNOR(xnorOut, I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]);
or OR(orOut, I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]);
and AND(OP, orOut, xnorOut);


endmodule
```

tb.v

```
// Lotito - Q4 Testbench

`include "EvenParityGenerator.v"
`timescale 1ns/1ns

module tb();

// I/O
reg [7:0] I;
wire OP;

// GTKWAVE
initial begin
    $dumpfile("tb.vcd");
    $dumpvars(0,tb);
end

// Stimulus
initial begin : stimmychecks

    // loop through all 256 inputs!
    reg [8:0] iv;
    for(iv = 0; iv < 256; iv=iv+1)
       #10 I = iv;

    #10 $finish; // break!
end

// Monitor and log changes in input
always@(I)
    $monitor("I = {%b}, Parity: %b", I, OP);

// Initialize module
EvenParityGenerator U1 (
    .OP(OP), .I(I)
);

endmodule
```

```
I = {11101111}, Parity: 0
I = {11110000}, Parity: 1
I = {11110001}, Parity: 0
I = {11110010}, Parity: 0
I = {11110011}, Parity: 1
I = {11110100}, Parity: 0
I = {11110101}, Parity: 1
I = {11110110}, Parity: 1
I = {11110111}, Parity: 0
I = {11111000}, Parity: 0
I = {11111001}, Parity: 1
I = {11111010}, Parity: 1
I = {11111011}, Parity: 0
I = {11111100}, Parity: 1
I = {11111101}, Parity: 0
I = {11111110}, Parity: 0
I = {11111111}, Parity: 1
tb.v:32: $finish called at 2570 (1ns)
```



Q5

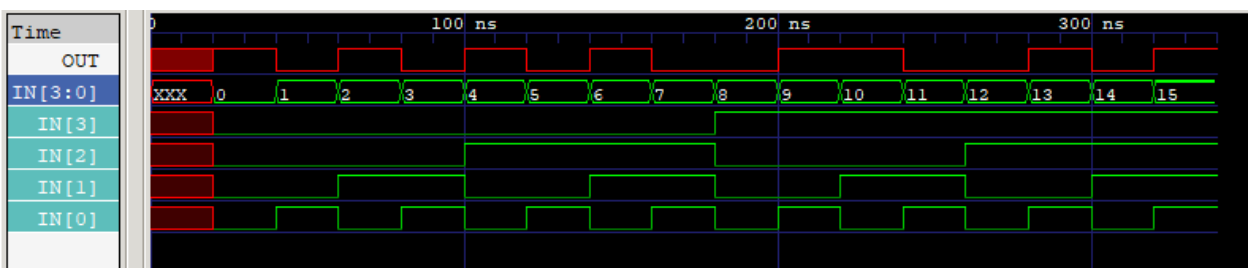
MinTermRealization.v

```
// Chase Lotito - SIUC - SP2024
// ECE426 - Chao Lu
// HW 2 - Q5 - MIN(0,2,4,6,9,10,13,15)

// Using a KMAP we get OUT(A,B,C,D) = A'D' + AC'D + B'CD' + ABD

module MinTermRealization (
    OUT, IN
);

// I/O
input [3:0] IN; // the vector is flipped, I = [D, C, B, A]
output OUT;

// Connecting signals
wire and0, and1, and2, and3;
wire [3:0] _IN;

// Gate-level design
not NOT0(_IN[0], IN[0]);
not NOT1(_IN[1], IN[1]);
not NOT2(_IN[2], IN[2]);
not NOT3(_IN[3], IN[3]);

and AND0(and0, _IN[3], _IN[0]); // Level 1
and AND1(and1, IN[3], _IN[1], IN[0]);
and AND2(and2, _IN[2], IN[1], _IN[0]);
and AND3(and3, IN[3], IN[2], IN[0]);

or OR0(OUT, and0, and1, and2, and3); // OUTPUT

endmodule
```

tb.v

```
// Lotito - Q5 - Testbench

`include "MinTermRealization.v"
`timescale 1ns/1ns

module tb();

// I/O
reg [3:0] IN;
wire OUT;

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("tb.vcd");
    $dumpvars(0, tb);
end

initial begin : stim
    reg [4:0] iv; // one extra bit so we catch when it overflows
    // Send all 16 inputs to circuit
    for(iv = 0; iv < 16; iv = iv + 1)
        #20 IN = iv;
    #20 $finish;
end

// Instantiate module
MinTermRealization U1 (
    .OUT(OUT), .IN(IN)
);

// CONSOLE LOG
always @(IN or OUT) begin
    $monitor("INPUT = {%b}, MINTERM = %d, OUT = %b", IN, IN, OUT);
end

endmodule
```

```
VCD info: dumpfile tb.vcd opened for output.
INPUT = {0000}, MINTERM =  0, OUT = 1
INPUT = {0001}, MINTERM =  1, OUT = 0
INPUT = {0010}, MINTERM =  2, OUT = 1
INPUT = {0011}, MINTERM =  3, OUT = 0
INPUT = {0100}, MINTERM =  4, OUT = 1
INPUT = {0101}, MINTERM =  5, OUT = 0
INPUT = {0110}, MINTERM =  6, OUT = 1
INPUT = {0111}, MINTERM =  7, OUT = 0
INPUT = {1000}, MINTERM =  8, OUT = 0
INPUT = {1001}, MINTERM =  9, OUT = 1
INPUT = {1010}, MINTERM = 10, OUT = 1
INPUT = {1011}, MINTERM = 11, OUT = 0
INPUT = {1100}, MINTERM = 12, OUT = 0
INPUT = {1101}, MINTERM = 13, OUT = 1
INPUT = {1110}, MINTERM = 14, OUT = 0
INPUT = {1111}, MINTERM = 15, OUT = 1
tb.v:23: $finish called at 340 (1ns)
PS C:\Users\Chase Lotito\Desktop\SIUC\SPRING
```



Q6

CircuitImplementation.v

```
/*
    Chase Lotito - SIUC - SP2024
    ECE426 - Chao Lu
    HW 2 - Q6 - Circuit Implementation
*/

module CircuitImplementation (
    Y, A, B, C, D
);

// I/O
input A, B, C, D;
output Y;

// Other connections
wire _D, na0, a0, xno0, o0, no0;

// Gate-Level Logic
not NOT0(_D, D);

nand NAND0(na0, A, B); // LEVEL 1
and AND0(a0, C, _D);

xnor XNOR0(xno0, na0, a0); // LEVEL 2

or OR0(o0, na0, xno0); // LEVEL 3
nor NOR0(no0, xno0, a0);

nor NOR1(Y, o0, no0); // OUTPUT

endmodule
```

tb.v

```
/*
    Chase Lotito - SIUC - SP2024
    HW2- Q6 - Testbench
*/

`include "CircuitImplementation.v"
`timescale 1ns/1ns

module tb();

// I/O
reg [3:0] I;
wire Y;

// Module Instantiation!
CircuitImplementation U1 (
    .Y(Y), .A(I[0]), .B(I[1]), .C(I[2]), .D(I[3])
);

// GTKWAVE
initial begin : GTKWAVE
    $dumpfile("tb.vcd");
    $dumpvars(0, tb);
end

// Stimulus!
initial begin : STIMMYCHECKS
    reg [4:0] iv;

    // Assign input values 0 --> 15
    for(iv = 0; iv < 16; iv = iv + 1)
        #20 I = iv;

    #10 $finish; // end stimmy
end

// Monitor and Console Log
always @(Y) begin
    $monitor("{A, B, C, D} = {%b%b%b%b}, Y = %b", I[0], I[1], I[2], I[3], Y);
end

endmodule
```

```
PS C:\Users\cloti\OneDrive\Desktop\SCHOO
VCD info: dumpfile tb.vcd opened for out
{A, B, C, D} = {0000}, Y = 0
{A, B, C, D} = {1000}, Y = 0
{A, B, C, D} = {0100}, Y = 0
{A, B, C, D} = {1100}, Y = 0
{A, B, C, D} = {0010}, Y = 0
{A, B, C, D} = {1010}, Y = 0
{A, B, C, D} = {0110}, Y = 0
{A, B, C, D} = {1110}, Y = 1
{A, B, C, D} = {0001}, Y = 0
{A, B, C, D} = {1001}, Y = 0
{A, B, C, D} = {0101}, Y = 0
{A, B, C, D} = {1101}, Y = 0
{A, B, C, D} = {0011}, Y = 0
{A, B, C, D} = {1011}, Y = 0
{A, B, C, D} = {0111}, Y = 0
{A, B, C, D} = {1111}, Y = 0
tb.v:34: $finish called at 330 (1ns)
PS C:\Users\cloti\OneDrive\Desktop\SCHOO
```