John Hubbard UCSC 30207 (p. 1) 24 Jan 2015
HW #2

Problem 2.6.1 – Part 1: Show that lutmask 8000 implements a 4-input AND gate.

(a) – The lutmask convention is: bit N of the lutmask is set if $f(a, b, c, d) = 1$ for the N^{th} entry of f's truth table. So $N == 4'b\, x_3 x_2 x_1 x_0$
where $a = x_3$
$b = x_2$
$c = x_1$
$d = x_0$

A lutmask of 0x8000 means that (only) bit 15 is set, in a truth table that ranges from 0 to 15. That means:

$$N = 15 = 4'b1111 = x_3\, x_2\, x_1\, x_0$$

so
$$\left.\begin{array}{l} x_3 = 1 = a \\ x_2 = 1 = b \\ x_3 = 1 = c \\ x_4 = 1 = d \end{array}\right)$$
f is 1 only for $abcd = 4b'1111$, when is the same truth table as $f = a \cdot b \cdot c \cdot d$

(b) – Another way of showing this is to show both truth tables:

a	b	c	d	$f = abcd$
0	0	0	0	0
0	0	0	1	$\downarrow$
		⋮		
1	1	1	0	0
1	1	1	1	1

same $\longleftrightarrow$

a	b	c	d	f = lutmask 0x8000
0	0	0	0	0
0	0	0	1	$\downarrow$
0	0	1	0	
		⋮		
				0
1	1	1	1	1

$$\left(\begin{array}{l} \left.\begin{array}{c}0\\0\\0\\0\end{array}\right) 0 \\ \left.\begin{array}{c}0\\0\\0\\0\end{array}\right) 0 \\ \left.\begin{array}{c}0\\0\\0\\0\end{array}\right) - 0 \\ \left.\begin{array}{c}0\\0\\0\\1\end{array}\right) \leftarrow 8 \end{array}\right.$$

John Hubbard UCSC 30207 24 Jan 2015

HW #2

Problem 2.6.1 — Part 2: Implement a 4-input XOR gate using LUT4.

a	b	c	d	f = a^b^c^d		
0	0	0	0	0		
0	0	0	1	1	6	← lutmask low bit
0	0	1	0	1		
0	0	1	1	0		
0	1	0	0	1		
0	1	0	1	0	9	
0	1	1	0	0		
0	1	1	1	1		
1	0	0	0	1		
1	0	0	1	0	9	
1	0	1	0	0		
1	0	1	1	1		
1	1	0	0	0		lutmask
1	1	0	1	1	6	← high bits
1	1	1	0	1		
1	1	1	1	0		

$\boxed{\text{lutmask} = 16'h\ 6996}$

= 0x6996

```
// Verilog:

module (f, a, b, c, d);
    input  a, b, c, d;
     output  f;

    LUT4 #(16'h 6996)  u(f, a, b, c, d);

endmodule
```

John Hubbard UCSC 30207 (p.3) 24 Jan 2015
HW #2

Problem 2.6.1 – Part 3: Implement this function using a LUT4:

$$f = a\bar{b}c\bar{d} + \bar{a}\bar{b}\bar{c}\bar{d} + a\bar{d} + \bar{a}b\bar{c} + \bar{b}d$$
$$+ \bar{a}b\bar{c} + ab\bar{c}d + bcd + \bar{a}c\bar{d}$$

– Each term tells which bits are set in the lutmask

a b c d	f	
0 0 0 0	1	$\bar{a}\bar{b}\bar{c}\bar{d}$
0 0 0 1	1	$\bar{b}d$
0 0 1 0	1	$\bar{a}c\bar{d}$
0 0 1 1	1	$\bar{b}d$
0 1 0 0	1	$\bar{a}b\bar{c}$
0 1 0 1	1	$\bar{a}b\bar{c}$
0 1 1 0	1	$\bar{a}c\bar{d}$
0 1 1 1	1	bcd
1 0 0 0	1	$a\bar{d}$
1 0 0 1	1	$\bar{b}d$
1 0 1 0	1	$a\bar{d}$
1 0 1 1	1	$\bar{b}d$
1 1 0 0	1	$a\bar{d}$
1 1 0 1	1	$ab\bar{c}d$
1 1 1 0	1	$a\bar{d}$
1 1 1 1	1	$abcd$, bcd

lutmask = 0x FFFF

$= \boxed{16'h\,FFFF}$

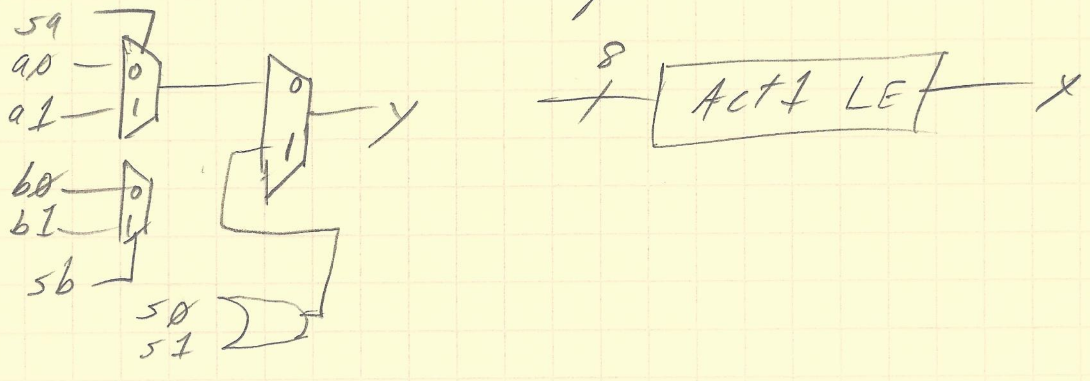
// Verilog:

```
module (f, a, b, c, d);
      input a, b, c, d;
       output f;

      LUT4 #(16'h FFFF) u(f, a, b, c, d);
endmodule

// or:   assign  f = 1;
```

John Hubbard UCSC 30207 (p.4) 24 Jan 2015

HW #2

Problem 2.6.2 - Part 1: Implement all possible 2-input functions using Actel Act1 block:



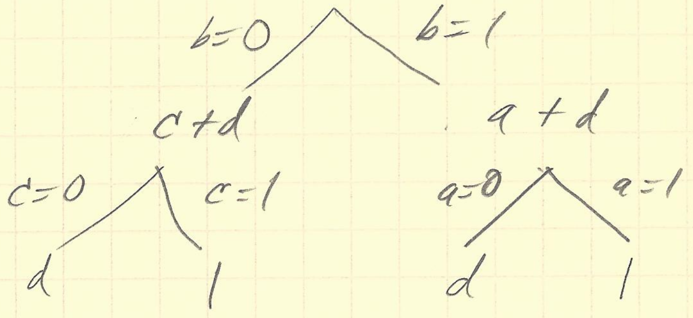
	Function	a0	a1	sa	b0	b1	sb	s0	s1
1	xor(a, b)	0	1	b	1	0	b	a	0
2	and(a, b)	0	a	b	x	x	x	0	0
3	nand(a, b)	1	1	1	1	0	a	b	0
4	nor(a, b)	1	0	b	0	0	0	a	0
5	or(a, b)	0	0	0	1	1	1	a	b
6	xnor(a, b)	1	0	b	0	b	1	a	0
7	not(a)	1	0	a	x	x	x	0	0
8	not(b)	1	0	b	x	x	x	0	0
9	buf(a)	0	1	a	x	x	x	0	0
10	buf(b)	0	1	b	x	x	x	0	0
11	0 (contradiction)	0	0	0	0	0	0	0	0
12	$a > b$	a	0	b	x	x	x	0	0
13	$a < b$	b	0	a	x	x	x	0	0
14	$a \geq b$	1	a	b	x	x	x	0	0
15	$a \leq b$	1	b	a	x	x	x	0	0
16	1 (tautology)	1	1	1	1	1	1	1	1

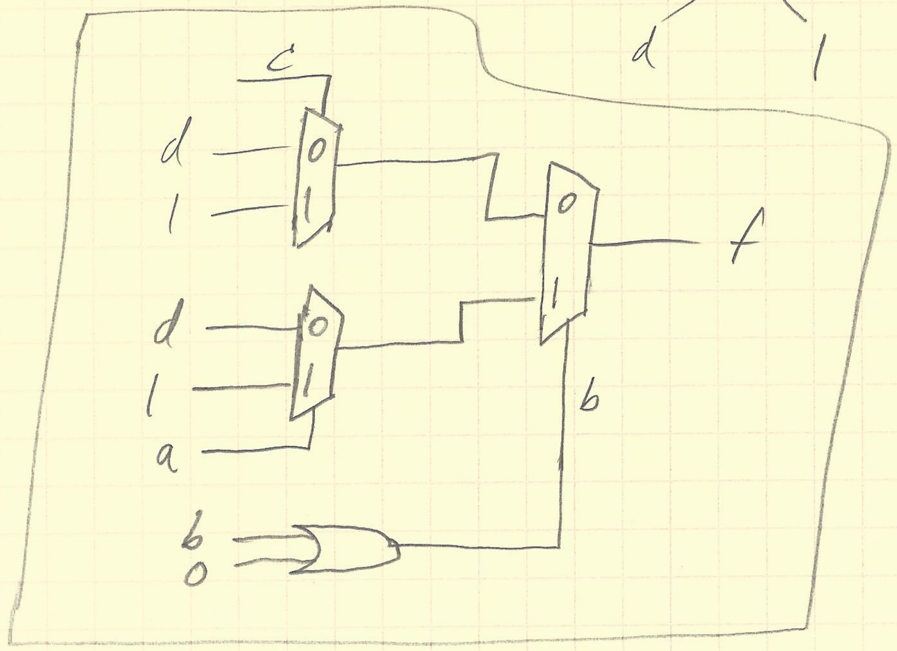
John Hubbard UCSC 30207 24 Jan 2015

HW #2

Problem 2.6.2 — Part 2: Implement this function using an Act 1 block:

$f = ab + \bar{b}c + d$

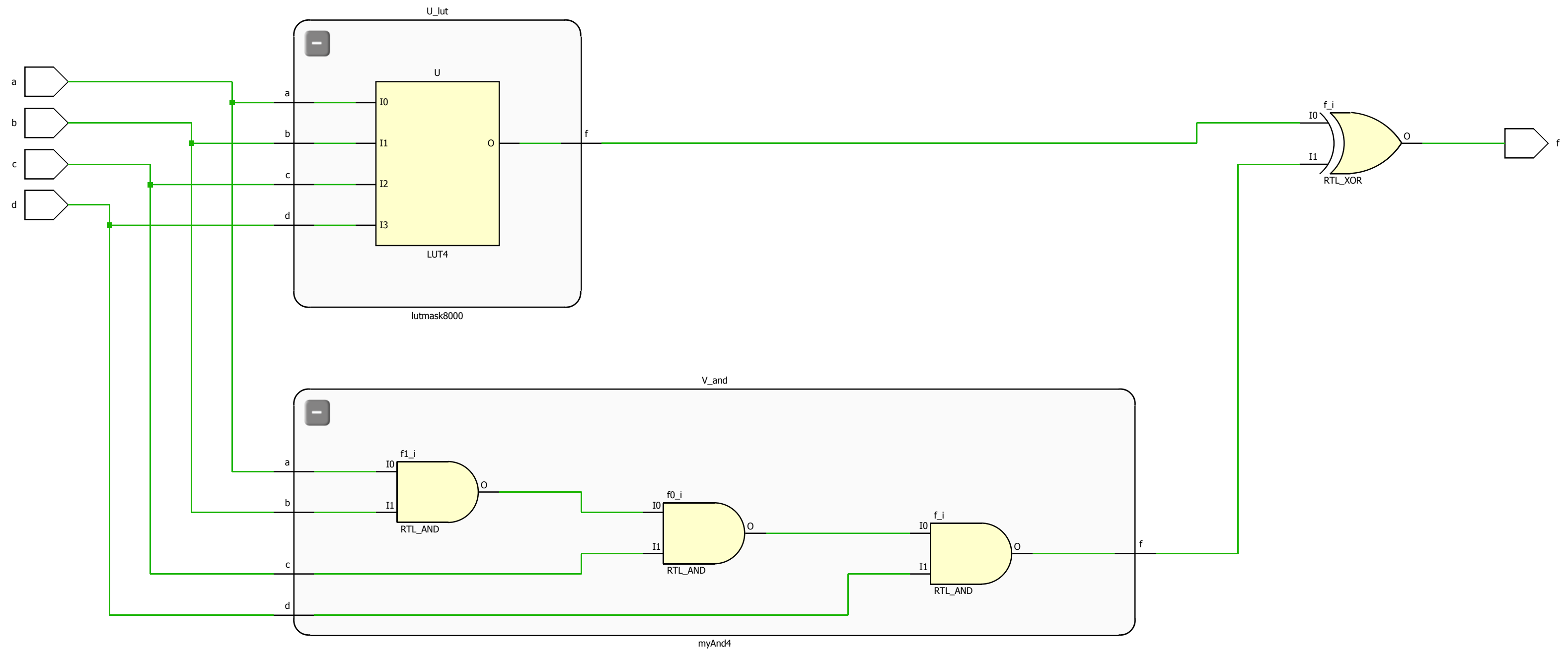




```verilog
// File: lutmask8000.v
// John Hubbard, 26 Jan 2015
// hw2 assignment for UCSC 30207: Digital Design with FPGA
//
// Part 1: show that lutmask 8000 implements a 4-input AND gate.

`timescale 1ns/1ns

module comparison(f, a, b, c, d);
    input a, b, c, d;
    output f;
    wire a, b, c, d, x, y, f;

    lutmask8000 U_lut(x, a, b, c, d);
    myAnd4      V_and(y, a, b, c, d);

    // This should have caused Vivado to remove the entire circuit, but it
    // did not. I don't understand why not. Both the elaborated design and the
    // schematic for the synthesized design showed that the AND4 and LUT4 were
    // still there.
    // --John Hubbard

    assign f = x ^ y;
endmodule

module lutmask8000(f, a, b, c, d);
    input a, b, c, d;
    output f;

    LUT4 #(16'h8000) U(f, a, b, c, d);
endmodule

module myAnd4(f, a, b, c, d);
    input a, b, c, d;
    output f;

    assign f = a & b & c & d;
endmodule
```

```
// File: lutmask8000_tb.v
// John Hubbard, 26 Jan 2015
// hw2 assignment for UCSC 30207: Digital Design with FPGA
//
// Part 1: show that lutmask 8000 implements a 4-input AND gate.

// Steps to run in ModelSim:
//
// cd D:/git_wa/classes/ucsc/fpga_30207/hw2/part1
// vlib work
// vlog *.v
// vsim work.lutmask_test
// add wave *
// run 100 ns
//

`timescale 1ns/1ns

module lutmask_test();
    reg a, b, c, d;
    reg [4:0] vec;
    wire lutmask_result, myAnd4_result, f_compare;

    lutmask8000 X(lutmask_result, a, b, c, d);
    myAnd4      Y(myAnd4_result,  a, b, c, d);
    comparison  Z(f_compare,      a, b, c, d);

    initial
    begin
        {a,b,c,d} = 4'b0;
        $display("abcd | lut4 | and4 | comparison");

        for (vec = 0; vec < 16; vec = vec + 1)
        begin
            #1 {a,b,c,d} = vec;

            #1 $display("%b%b%b%b | %b    | %b   |     %b", a, b, c, d,
                        lutmask_result, myAnd4_result, f_compare);

            if (f_compare)
                $display("Error: miscompare between lutmask8000 and myAnd4!");
        end
    end

endmodule

/* Sample run:

# run 1000ns
abcd | lut4 | and4 | comparison
0000 | 0    | 0   |     0
0001 | 0    | 0   |     0
0010 | 0    | 0   |     0
0011 | 0    | 0   |     0
```

```
0100 | 0    | 0   |    0
0101 | 0    | 0   |    0
0110 | 0    | 0   |    0
0111 | 0    | 0   |    0
1000 | 0    | 0   |    0
1001 | 0    | 0   |    0
1010 | 0    | 0   |    0
1011 | 0    | 0   |    0
1100 | 0    | 0   |    0
1101 | 0    | 0   |    0
1110 | 0    | 0   |    0
1111 | 1    | 1   |    0

*/
```


U_lut
a
b
c
d
U
I0
I1
I2
I3
O
f
LUT4
lutmask8000
V_and
f1_i
I0
I1
O
RTL_AND
f0_i
I0
I1
O
RTL_AND
f_i
I0
I1
O
RTL_AND
f
myAnd4
f_i
I0
I1
O
RTL_XOR
f

```
// File: various_lut4.v
// John Hubbard, 26 Jan 2015
// hw2 assignment for UCSC 30207: Digital Design with FPGA
//
// Problem 2.6.1, part 2: implement a 4-input XOR gate using a LUT4.
// Problem 2.6.1, part 3: Implement the function below using a LUT4:

`timescale 1ns/1ns

module comparison(f, a, b, c, d);
    input a, b, c, d;
    output f;
    wire a, b, c, d, x, y, f;

    xor_via_lut U_lut(x, a, b, c, d);
    myXor4      V_xor(y, a, b, c, d);

    assign f = x ^ y; // compare the results
endmodule

module xor_via_lut(f, a, b, c, d);
    input a, b, c, d;
    output f;

    // Problem 2.6.1, part 2: implement a 4-input XOR gate using a LUT4.
    //
    // Please see my paper notes for how the 0x6996 was derived:
    LUT4 #(16'h6996) U(f, a, b, c, d);
endmodule

module special_function_via_lut(f, a, b, c, d);
    input a, b, c, d;
    output f;

    // Problem 2.6.1, part 3: Implement the function below using a LUT4:
    //
    // f = a.b.c.d + a'.b'.c'.d' + a. d' + a'.b.c' +
    //     b'.d + a'.b.c' + a.b.c'.d + b.c.d + a'.c.d'

    // Please see my paper notes for how the 0xffff was derived:
    LUT4 #(16'hffff) U(f, a, b, c, d);
endmodule

module myXor4(f, a, b, c, d);
    input a, b, c, d;
    output f;

    assign f = a ^ b ^ c ^ d;
endmodule
```

```
// File: various_lut4_tb.v
// John Hubbard, 26 Jan 2015
// hw2 assignment for UCSC 30207: Digital Design with FPGA
//
// Problem 2.6.1, part 2: implement a 4-input XOR gate using a LUT4.
// Problem 2.6.1, part 3: Implement the function below using a LUT4:
//

`timescale 1ns/1ns

module various_luttests();
    reg a, b, c, d;
    reg [4:0] vec;
    wire xor_via_lut_result, myXor4_result, f_compare, special_result;

    xor_via_lut X(xor_via_lut_result, a, b, c, d);
    myXor4      Y(myXor4_result, a, b, c, d);
    comparison  Z(f_compare, a, b, c, d);
    special_function_via_lut  AA(special_result, a, b, c, d);

    initial
    begin
        {a,b,c,d} = 4'b0;
        $display("abcd | lut4 | xor4 | comparison | special_result");

        for (vec = 0; vec < 16; vec = vec + 1)
        begin
            #1 {a,b,c,d} = vec;

            #1 $display("%b%b%b%b | %b    | %b    |     %b | %b", a, b, c, d,
                        xor_via_lut_result, myXor4_result, f_compare, special_result);

            if (f_compare)
                $display("Error: miscompare between xor_via_lut and myXor4!");
        end
    end

endmodule

/* Sample run in Vivado's simulator (ModelSim cannot support LUT4):

# run 1000ns
abcd | lut4 | xor4 | comparison | special_result
0000 | 0    | 0    |     0 | 1
0001 | 1    | 1    |     0 | 1
0010 | 1    | 1    |     0 | 1
0011 | 0    | 0    |     0 | 1
0100 | 1    | 1    |     0 | 1
0101 | 0    | 0    |     0 | 1
0110 | 0    | 0    |     0 | 1
0111 | 1    | 1    |     0 | 1
1000 | 1    | 1    |     0 | 1
1001 | 0    | 0    |     0 | 1
1010 | 0    | 0    |     0 | 1
```

```
1011 | 1    | 1   |    0 | 1
1100 | 0    | 0   |    0 | 1
1101 | 1    | 1   |    0 | 1
1110 | 1    | 1   |    0 | 1
1111 | 0    | 0   |    0 | 1
*/
```