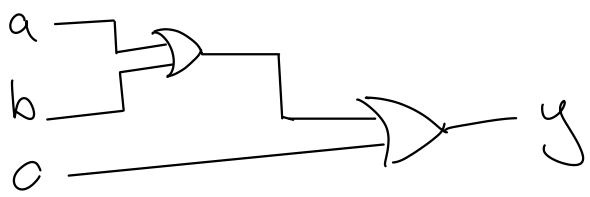
Introduction to Digital Design with Verilog HDL

Exercise 1: What changes would result in a 3-input OR gate?

```
// AND gate in Verilog

module ex1 ( input logic a, b, c,
             output logic y ) ;

   assign y = a & b ;

endmodule
```

y = a | b | c;

Exercise 2: What schematic would you expect if the statement was `assign y = ( a ^ b ) | c ;`?



Exercise 3: What are the lengths and values, in decimal, of the following:

`4'b1001`? 4 9

`5'd3`? 5 3

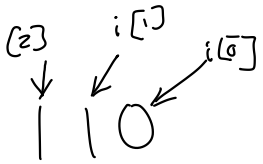
`6'h0_a`? 6 10

3? = 32'd3 32 3

Exercise 4: If the signal **i** is declared as **`logic [2:0] i;`**, what is the 'width' of **i**?

3

If **i** has the value 6 (decimal), what is the value of **`i[2]`**?

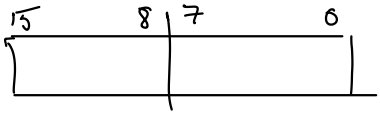


1

Of **`i[0]`**?

0

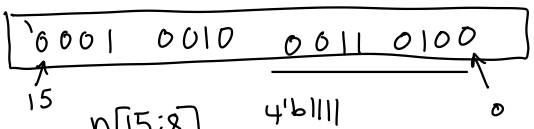
Exercise 5: Use slicing and concatenation to compute the byte-swapped value of an array **n** declared as **`logic [15:0] n`**.



n = { n[7:0] , n[15:8] }

Exercise 6: If **n** has the value **`16'h1234`**, what is the value and length of:

`{n[7:0],n[15:8],4'b1111}`?

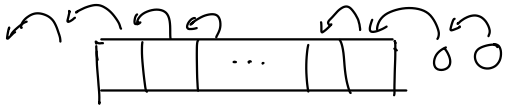


n[7:0] n[15:8] 4'b1111

20 bits: 0011 0100 0001 0010 1111

= 20'h3412f

Exercise 7: Use concatenation to shift **n** left by two bits.



{ n[13:0] , 2'b00 }

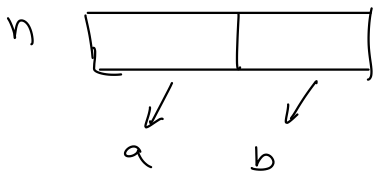
2'b0
2'h0
2'd0000
all equivalent

alternative to:

n << 2

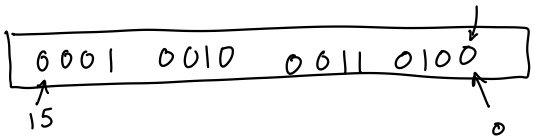
Exercise 8: Use concatenation to assign the high-order byte of **n** to **a** and the low-order byte to **b**.

```
{ a , b } = n[15:0];
```




```
logic [7:0] a ;
logic [7:0] b ;
```

Exercise 9: An array declared as `logic [15:0] n;` and has the value `16'h1234`. What are the values and lengths of the following expressions?



`n[15:13]`

3 bits: 3'b000 or 3'h0

`!n`

1 bit: 1'b0

`~(n[3:0])`

4 bits: 4'b0100 → 4'b1011

`n>>4`

0001 0010 0011 0100 → 16'b0000 0001 0010 0011

16 bits

`n + 1'b1`

16 bits: 16'h1235

16'h1234

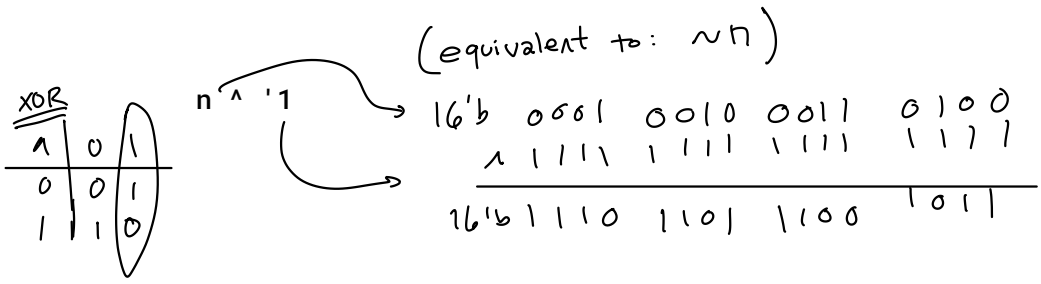
`n[7:0] - n[3:0]`

8'h34 - 4'h4 = 8'h30

8'h3 = 8'h03

`n >= 16'h1234`

1 1'b1



`n && !n` → 1'b0

n && 1'b0 1'b0

`n * ( !n + 1'b1 )`

n*(1'b0 + 1'b1)

n * 1'b1

16'h1234 * 1'b1

16'h1234

Exercise 10: What are the length and value of the expression: `3 ? 16'd10 : 8'h20`?

length = 16 16'd10

value = 10

If **x** has the value 0, what is the value of the expression: ① `x ? 1'b1 : 1'b0`?

② ③ 1'b0

T F

-1 ? 1'b1 : 1'b0

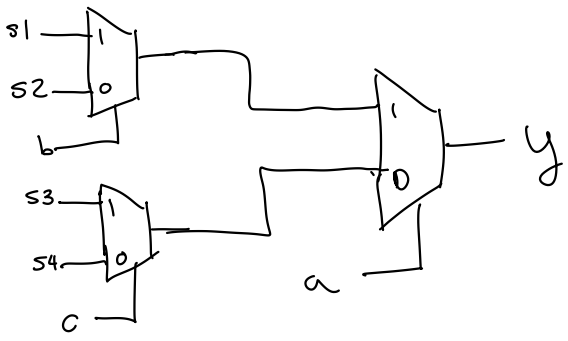
(is T)

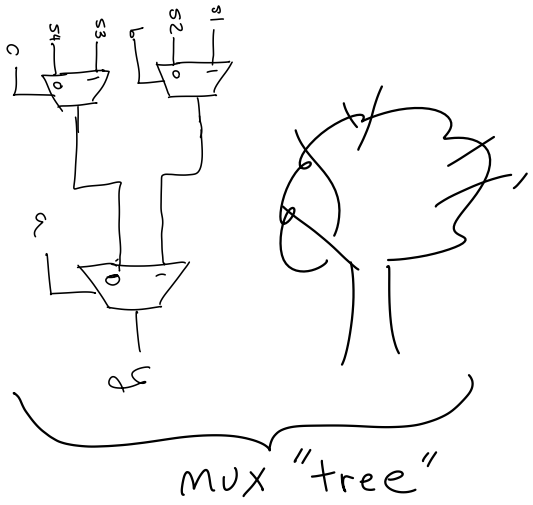
If **x** has the value -1?

1'b1

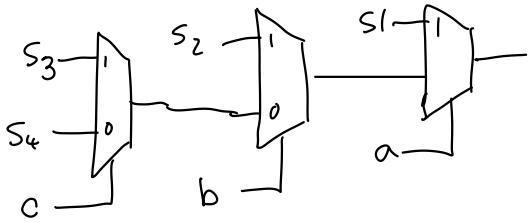
Exercise 11: Draw the schematics corresponding to:

```
y = a ? ( b ? s1 : s2 ) : ( c ? s3 : s4 );
```






```
y = a ? s1 : b ? s2 : c ? s3 : s4;
```



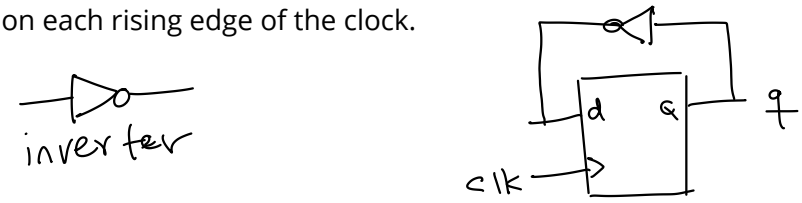
Exercise 12:

```
assign y = a + 1 ;
```

→ 1'b1

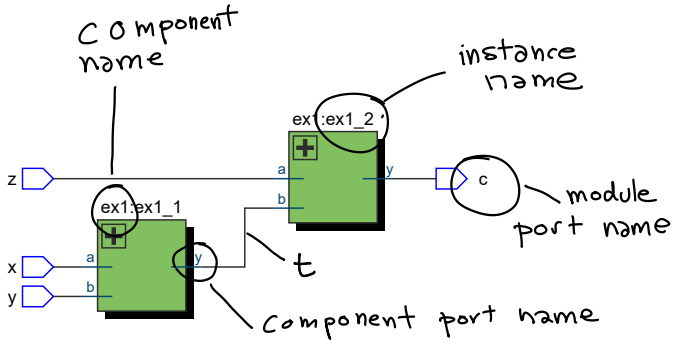
Some software warns about truncation. How could you re-write the **`assign`** statement to avoid such a warning?

Exercise 13: Write an **`always_ff`** statement that toggles (inverts) its output on each rising edge of the clock.




```
always_ff @(posedge clk)
    q <= !q;
```

Exercise 14:



Identify the following in the diagram above: component names, component "instance names," component port names, module port names. Label the signal **`t`** in the schematic.

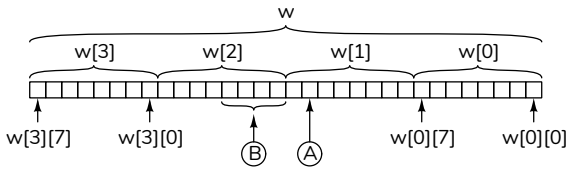
Exercise 15: Rewrite the **`ex60`** module using operators. Which version – "structural" or "behavioural" – is easier to understand?

behavioural: `assign c = x & y & z;` (easier to understand)

Exercise 16:

[:] slice

[ ] index



How would you specify the bit marked A in the diagram above?

w [1] [6]

The bits marked B? w [2] [3:0]

The least-significant byte? w [0] [7:0]

lookup table is like a function:

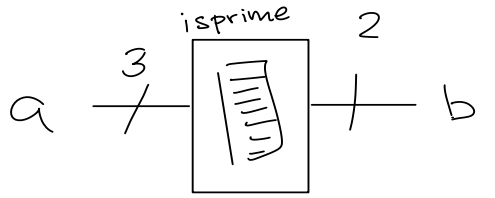
y = f (x)

Exercise 17: Define a Verilog lookup table named **`isprime`** that can be used to determine if a value between 0 and 7 is a prime number or not. The result should be 1 if the value is a prime or else 2. *Hint: The primes are 2, 3, 5 and 7.*

index	value
0	2
1	2
2	1
3	1
4	2
5	1
6	2
7	1

isprime (x) → { 1 if x prime ; 2 otherwise }

```
logic [0:7] [1:0] isprime = '{2'b10, 2'b10, 2'b01, 2'b01, 2, 1, 2, 1};
```

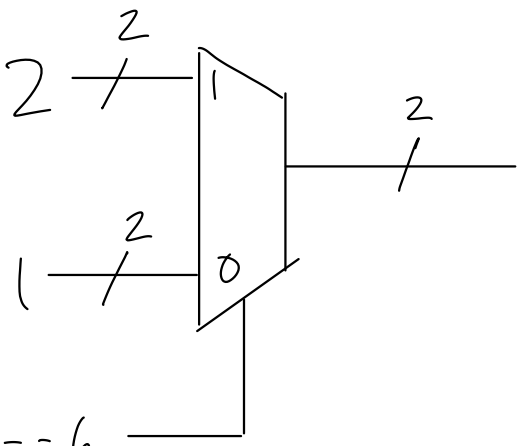


assign b = isprime [a];

Exercise 18: Write an expression giving the same result. Draw the corresponding block diagram.

a == 0 || a == 1 || a == 4 || a == 6 ? 2 : 1

same as: isprime[a]



a == 0 || a == 1 || a == 4 || a == 6