Introduction to Digital Design with Verilog HDL

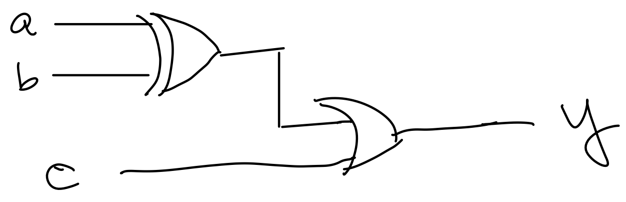
Exercise 1: What changes would result in a 3-input OR gate?

```
module ex1 ( input logic a, b, c,
             output logic y ) ;
                  a | b | c ;
   assign y = ~~a & b ;~~
                      assign y = a | b | c ;
endmodule
```

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

^ = XOR

| = OR

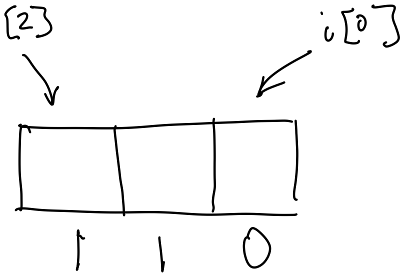


Exercise 3: What are the lengths and values, in decimal, of the following: `4'b1001`, `5'd3`, `6'h0_a`, 3?

		length	value
4'b1001	→	4	9
5'd3	→	5	3
6'h0_a	→	6	10
3	→	32	3

Exercise 4: If the signal `i` is declared as `logic [2:0] i;`, what is the 'width' of `i`? If `i` has the value 6 (decimal), what is the value of `i[2]`? Of `i[0]`?

width = 3



i[2] = 1

i[0] = 0

Exercise 5: An array declared as `logic [15:0] n;` and has the value `16'h1234`. 0001 0010 0011 0100

4'h4

3:0

What are the values and lengths of the following expressions?

15 → 0001 0010 0011 0100 ← 0

7:0 ⇒ 8'h34

`n[15:13]`	3'b000
`!n`	1'b0
`~n[3:0]`	4'b1011
`n>>4`	16'b0000 0001 0010 0011 ~~0100~~ or 16'h0123
`n + 1'b1`	16'h1235
`n[7:0] - n[3:0]`	8'(8'h34 - 4'h4) ⇒ 8'h30
`n >= 16'h1234`	1'b1
`n ^ ~n`	16'h ffff
`n && !n`	16'h1234 && $\underbrace{1'b0}_{!n}$ ⇒ 1'b0
`n * ( !n + 1'b1 )`	1'b0 + 1'b1 ⇒ 1'b1; 1'b1 * 16'h1234 ⇒ 16'h1234

```
  0011 0100
-      0100
-----------
  0011 0000
```

```
  0001 0010 0011 0100  n
^ 1110 1101 1100 1011  ~n
-------------------------
  1111 1111 1111 1111
```

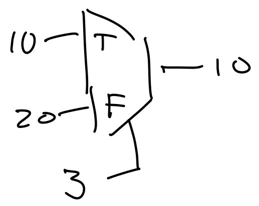
n * (!n + 1'b1): ③ ① ②

!0 ⇒ 1'b1

!1 ⇒ 1'b0

Exercise 6: What is the value of the expression `3 ? 10 : 20`? Of the expression `x ? 1 : 0` if `x` has the value 0? If `x` has the value -1?

3 ? 10 : 20 ⇒ 32'd10

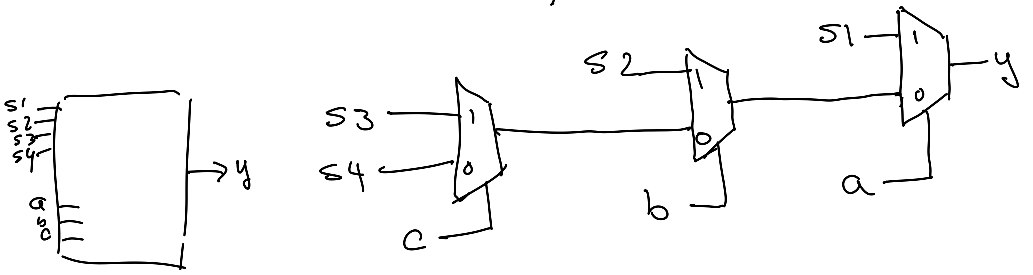


x ? 1 : 0 if x==0 ⇒ 0

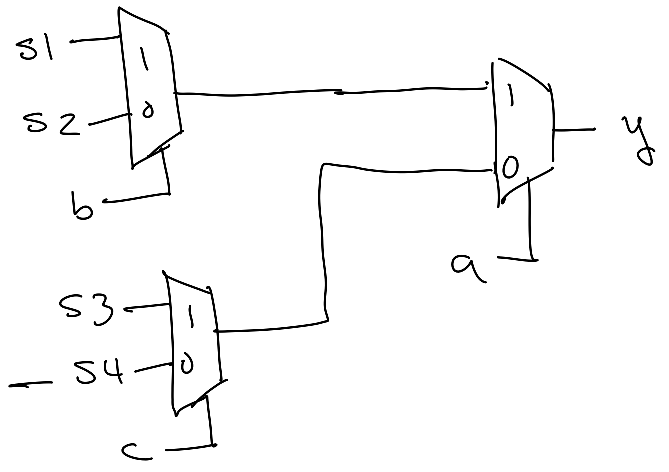
if x==-1 ⇒ 1

Exercise 7: Draw the schematics corresponding to:

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



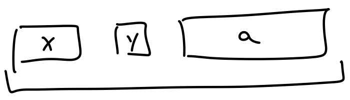
`y = a ? ( b ? s1 : s2 ) : ( c ? s3 : s4 );`



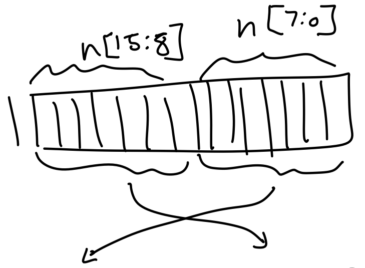
$\{n[15:8], n[7:0]\} \Rightarrow$ n

Exercise 8: Use slicing and concatenation to compute the byte-swapped value of an array `n` declared as `logic [15:0] n`.

$\{n[7:0], n[15:8]\}$



$\{x, y, a\}$



$\{n[7:0], n[15:8]\} \Rightarrow$

e.g. if n == 16'h 1234

then $\{n[7:0], n[15:8]\} \Rightarrow$ 16'h3412

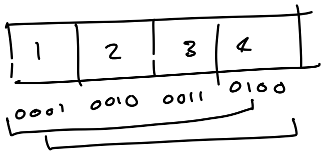
but $\{n[15:8], n[7:0]\} \Rightarrow$ 16'h1234

Exercise 9: If `n` has the value `16'h1234`, what is the value and length of `{n[7:0],n[15:8],4'b1111}`?

$\{$ 8'h34, 8'h12, 4'hf $\} \Rightarrow$ $\underline{20}$'h 3412f

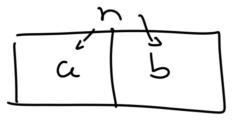
Exercise 10: Use concatenation to shift `n` left by two bits.

n = 16'h1234



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

Exercise 11: Use concatenation to assign the high-order byte of `n` to `a` and the low-order byte to `b`.



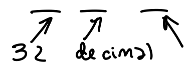
logic [7:0] a, b;

assign {a, b} = n;

n = {a, b} also possible

Exercise 12:

```
             1'b1
assign y = a + ~~1~~ ;
```

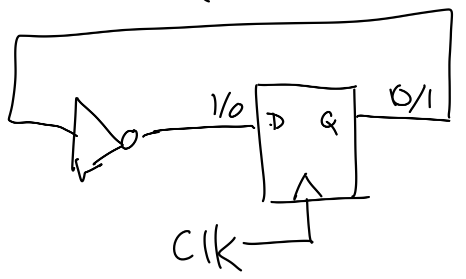


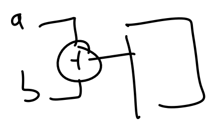
Some software warns about truncation. How could you re-write the `assign` statement to avoid such a warning?

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

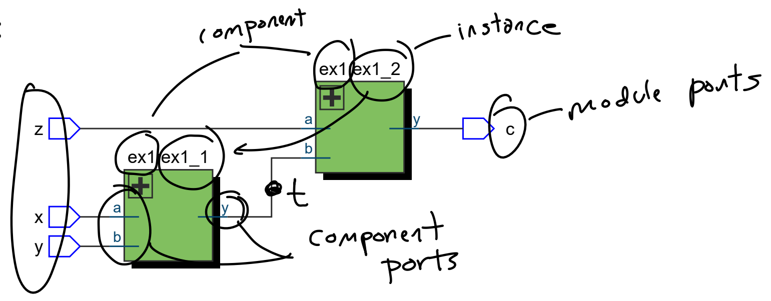
input d   output q

alway_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?

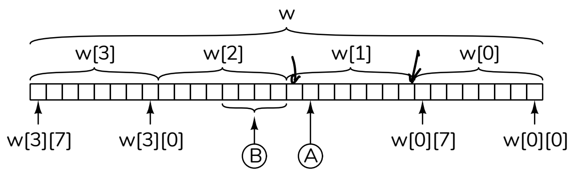
```
module ex60 ( input logic x, y, z,
output logic c ) ;
logic t ;
ex1 ex1_1 ( x, y, t ) ;
ex1 ex1_2 ( z, t, c ) ;
endmodule
```

→ assign c = a | b | c ;

↑ "structural" ↑ easier for computers to understand

↑ "Behavioural" ↑ easier for humans to understand

Exercise 16:



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]

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.*

```
                              // 0  1  2  3  4  5  6  7
logic [0:7][1:0] isprime = '{2, 2, 1, 1, 2, 1, 2, 1};
```