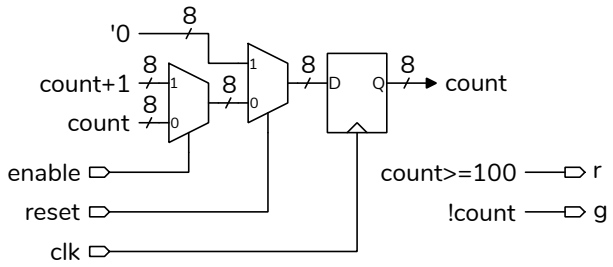
ELEX 2117 : Digital Techniques 2
2024 Fall Term

Solutions to Final Exam

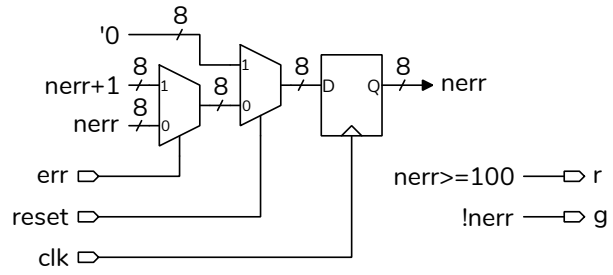
There were two versions of each question. The values and the answers for both versions are given below.

Question 1

The block diagram below follows the course conventions for block diagrams[1]. Write a Verilog module named **`ecnt`** that implements this block diagram, including the module declaration and declarations for signals used within the module. Follow the course coding conventions but omit comments.



or:



Answers

```
module ecnt
  ( input logic enable, reset, clk,
    output logic r, g ) ;

   logic [7:0] count ;

   always_ff @(posedge clk)
     count <= reset ? '0 : enable ? count+1 : count ;

   assign r = count >= 100 ;
   assign g = !count ;

endmodule
```

[1]Including that the default bus width is 1 bit.

```
module ecnt_
  ( input logic err, reset, clk,
    output logic r, g ) ;

   logic [7:0] nerr ;

   always_ff @(posedge clk)
     nerr <= reset ? '0 : err ? nerr+1 : nerr ;

   assign r = nerr >= 100 ;
   assign g = !nerr ;

endmodule
```

Question 2

Fill the table below with the value of each expression as a Verilog numeric literal including the correct width and the correct value in *hexadecimal* base. Assume the following declarations:

```
logic [7:0] x ;
logic [3:0] y ;
```

and that **`x`** has the value **`8'h9c`** (or **`8'hA3`**) and that **`y`** has the value **`4'b0101`**. The first row has been filled in as an example. You need not show your work or draw another box around the answer

expression	value
`x[3:0]`	`4'hc (or 4'h3)`
`x[3:2]`	
`x>>2`	
`{x[7:4], y}`	
`x[1] ? y : y[1:0]`	
`(x && y) + 1`	
`x <= 128 ? x||y : x|y`	
`(3'b100+9'h100)*8'd16`	
`x ^ y & 7`	

Answers

expression	value
`x[3:0]`	`4'hc`
`x[3:2]`	`2'h3`
`x>>2`	`8'h27`
`{x[7:4], y}`	`8'h95`
`x[1] ? y : y[1:0]`	`4'h1`
`(x && y) + 1`	`32'h2`
`x <= 128 ? x||y : x|y`	`8'h9d`
`(3'b100+9'h100)*8'd16`	`9'h40`
or	`9'h1040`
`x ^ y & 7`	`32'h99`
or	`32'h1`

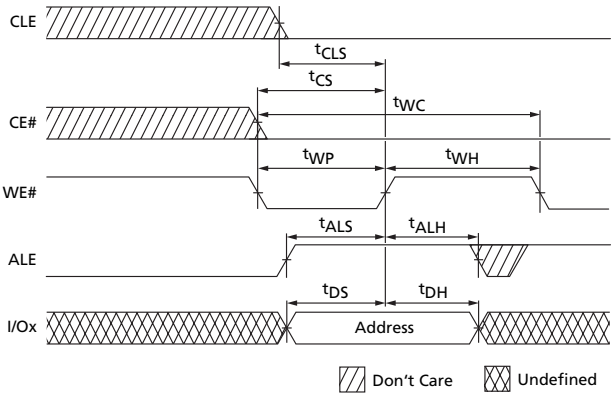
expression	value
`x[3:0]`	`4'h3`
`x[3:2]`	`2'h0`
`x>>2`	`8'h28`
`{x[7:4], y}`	`8'ha5`
`x[1] ? y : y[1:0]`	`4'h5`
`(x && y) + 1`	`32'h2`
`x <= 128 ? x||y : x|y`	`8'ha7`
`(3'b100+9'h100)*8'd16`	`9'h40`
or	`9'h1040`
`x ^ y & 7`	`32'ha6`
or	`32'h6`

The expression `(3'b100+9'h100)*8'd16` is evaluated in a 9-bit context as the sum of `9'h100` and `3'h4` (`9'h104`) shifted left by 4 bits: `9'h40` (multiplying by 16 (2^4) is a left-shift by 4 bits). The answer `9'h1040` (without truncation) was also accepted as correct.

Bitwise-and (`&`) has higher precedence then bitwise-exclusive-or (`^`) which in turn has higher precedence than bitwise-or (`|`) in the Verilog standard (the same is true for the logical-and and logical-or operators). However, the lectures notes list them in the same section, implying the same precedence. Thus there are two answers given above; either was marked correct.

Question 3

The following timing diagram is from the datasheet for the Micron MT29F8G08FABWP NAND flash memory. The suffix `#` indicates an active-low signal (as in `CE#`). All signals shown in the diagram are inputs.



(a) Which timing specifications are "guaranteed responses"?

(b) Which timing specification is a pulse width? *Hint: There is only one.*

(c) Assume `WE#` is a clock signal. (i) Which timing specification(s) is/are setup times? (ii) Which timing specification(s) is/are hold time(s)?

Answers

(a) None of the timing specifications are "guaranteed responses" since all of the signals are inputs.

(b) Only t_{WP} is a pulse width. (not t_{WP} because the signal is active-low).

(c) If `WE#` is a clock signal, specifications measured from a signal becoming valid to the rising edge are setup times and those measured from the rising edge to data becoming invalid are hold times:

(i) t_{CLS}, t_{CS}, t_{ALS}, and t_{DS} are measured from undefined (or invalid) to the rising edge of the clock and are thus setup times.

(ii) t_{ALH} and t_{DH} are measured from the rising edge of the clock to undefined (or invalid) and are thus hold times. t_{WH} is measured to the clock signal itself so it is not a "hold" time.

Question 4

The following specifications are taken from the same IC as in the previous question:

Table 13: DC and Operating Characteristics

Parameter	Conditions	Symbol	Min	Typ	Max	Unit
Input high voltage	I/O [7–0], I/O [15–0] CE#, CLE, ALE, WE#, RE#, WP#, PRE, R/B#	VIH	0.8 x Vcc	–	Vcc + 0.3	V
Input low voltage (all inputs)	–	VIL	-0.3	–	0.8	V
Output high voltage	IOH = -400µA	VOH	0.9 x Vcc	–	–	V
Output low voltage	IOL = 2.1mA	VOL	–	–	0.4	V

Assume V_{cc} is 2.7 V (or 3.0 V). (a) What is the noise margin for the high logic level? (b) What is the noise margin for the low logic level?

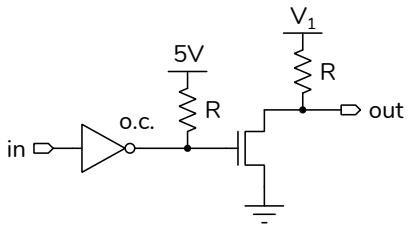
Answers

The high-level noise margin is $V_{OH(min)} - V_{IH(min)} = 0.9V_{cc} - 0.8V_{cc} = 0.1V_{cc} =$ $\boxed{0.27\,\text{V}}$ (or $\boxed{0.3\,\text{V}}$).

The low-level noise margin is $V_{IL(max)} - V_{OL(max)} = 0.8\,\text{V} - 0.4\,\text{V} = \boxed{0.4\,\text{V}}$.

Question 5

The schematic shows a logic level conversion circuit using an open-collector inverter and an N-channel MOSFET. Assume the on-state resistance of the open-collector output and the MOSFET are much lower than R. V_1 is 10 (or 20) V.



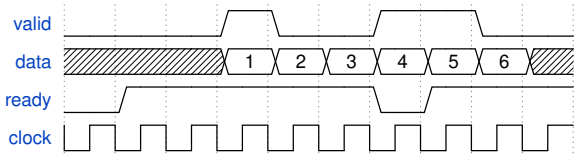
(a) What is the voltage at **out** when **in** is at a low logic level? (b) What is the voltage at **out** when **in** is at a high logic level?

Answers

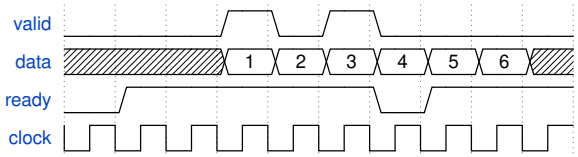
The MOSFET acts as an inverter and the logical function of the two inverters cancels out so the input and output are at the same logic state (high or low). (i) When the input is low the output will be approximately $\boxed{0\,\text{V}}$. (ii) When the input is high the output will be approximately $\boxed{10\,\text{V}}$ (or $\boxed{20\,\text{V}}$).

Question 6

The following timing diagram shows the operation of a ready/valid (FIFO) interface. List the values on the **data** line which are transferred over the interface.



(or



)

Answers

Data is transferred when both **valid** and **ready** are asserted. In the first diagram the values transferred over **data** are $\boxed{1 \text{ and } 5}$. In the second diagram the values transferred over **data** are $\boxed{1 \text{ and } 3}$.

Question 7

The frequency of the **clk** signal in the Verilog code below is 10 (or 5) MHz.

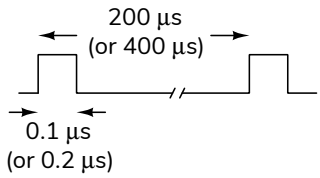
```
logic [11:0] count ;
logic out ;

always_ff @(posedge clk)
   count <= count ? count-1 : 1999 ;

assign out = !count ? 1 : 0 ;
```

Sketch the **out** signal (not necessarily to scale) and label the pulse duration and waveform period in microseconds. Show your calculations.

Answers

The clock period is 100 ns for a frequency of 10 MHz and 200 ns for a frequency of 5 MHz. The counter will continuously count values from 1999 to zero. The period is thus 2000 clock periods or $\boxed{200\,\mu s}$ (or $\boxed{400\,\mu s}$). The output is only asserted when count is zero, for one clock period, $\boxed{0.1\,\mu s}$ (or $\boxed{0.2\,\mu s}$).

A sketch of one period of the output waveform is:



Question 8

For each term in the left column write the number of the most appropriate match in the right column. There is only one best match for each term. There is no penalty for a wrong answer.

NRE	
Moore's Law	
wafer	
TQFP	
MAX-II	
TSMC	
software	

(1) round
(2) CPLD
(3) mask costs
(4) short TTM
(5) ~~CPLD~~ package
(6) foundry
(7) digital IC's

or:

(1) foundry
(2) package
(3) round
(4) short TTM
(5) mask costs
(6) CPLD
(7) digital ICs

Answers

NRE	mask costs	3	5
Moore's Law	digital ICs	7	7
wafer	round	1	3
TQFP	package	5	2
MAX-II	CPLD	2	6
TSMC	foundry	6	1
software	short TTM	4	4

Question 9

The following Verilog code defines a state machine. The state variable is **`state`**. A one-hot state encoding is used. Draw the state transition diagram. Label the states and state transition conditions using the conventions in lecture notes.

```
module sm
  ( input logic reset, clk,
    input logic [1:0] ab,
    output logic cw ) ;

   logic [2:0] state ;

   always_ff @(posedge clk)
     state <= reset ? 3'b001 :
              state == 3'b001 && ab == 2'b10 ? 3'b010 :
              state == 3'b010 && ab == 2'b01 ? 3'b100 :
              state == 3'b100 && ab == 2'b10 ? 3'b010 :
              state ;
endmodule
```

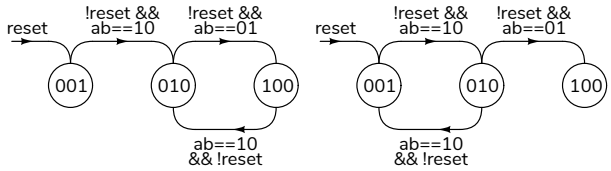
or:

```
module sm
  ( input logic reset, clk,
    input logic [1:0] ab,
    output logic cw ) ;

   logic [2:0] state ;

   always_ff @(posedge clk)
     state <= reset ? 3'b001 :
              state == 3'b001 && ab == 2'b10 ? 3'b010 :
              state == 3'b010 && ab == 2'b01 ? 3'b100 :
              state == 3'b010 && ab == 2'b10 ? 3'b001 :
              state ;
endmodule
```

Answers

The values assigned to **`state`** are 001, 010, and 100. The transition conditions out of each state can be read from the conditional assignments to **`state`** in

the **always_ff** statement. The resulting state transition diagrams are:



Answers that don't include (**!reset**) on the transition conditions were also marked correct.

Question 10

A module to be tested is declared as:

```
module dut ( input logic clk ; output logic done ) ;
```

Write a Verilog testbench named **dut_tb** that instantiates this module, generates a 2 (or 4) MHz clock signal connected to **clk** and terminates the simulation when **done** is asserted. Declare all signals used. Your testbench must **not** read or write any files, check for errors, or print any messages.

Answers

```
module dut_tb ;

   logic clk, done ;

   dut d0 ( .* ) ;

   always #0.25us clk <= !clk ; // 2 MHz
   // or:
   // always #0.125us clk <= !clk ; // 4 MHz

   initial begin
      wait(done) ;
      $finish ;
   end

endmodule
```