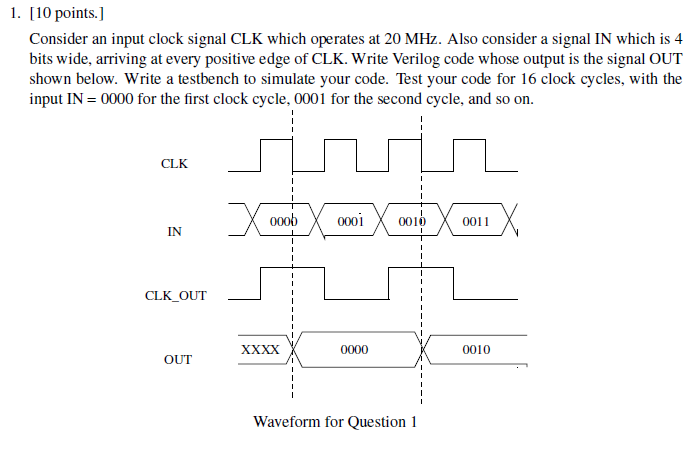
Khanh Nguyen

UIN 525000335

ECEN 449-503

Dr.Khatri

**Homework 1**



**Answer:**

**Verilog code:**

`timescale 1ns / 1ps

`default\_nettype none

module hw1\_q1(CLK, IN, CLK\_OUT, OUT);

input wire CLK;

input wire [3:0] IN;

output reg [3:0] OUT;

output reg [3:0] CLK\_OUT;

initial CLK\_OUT = 0;

initial OUT = 4'b0000;

always@(posedge CLK)//CLK\_OUT changes at pos edge

begin

CLK\_OUT = ~CLK\_OUT;

end

always@(CLK\_OUT)//OUT changes at CLK\_OUT

begin

OUT= IN;

end

endmodule // hw1\_q1

**Test bench:**

`timescale 1ns / 1ps

`default\_nettype none

module hw1\_q1\_tb;

//Inputs

reg CLK;

reg [3:0] IN;

//Outputs

wire [3:0] OUT;

wire CLK\_OUT;

//Instantiate the Unit Under Test (UUT)

hw1\_q1 uut(

.CLK(CLK),

.IN(IN),

.CLK\_OUT(CLK\_OUT),

.OUT(OUT)

);

//generate 20MHz clock signal

always

#25 CLK = ~CLK; //since 2MHz is 50ns per cycle

//=>#25 half cycle

initial begin

CLK=0;

#25 IN=4'b0000; //inputs

#25 IN=4'b0001;

#25 IN=4'b0010;

#25 IN=4'b0011;

#25 IN=4'b0100;

#25 IN=4'b0101;

#25 IN=4'b0110;

#25 IN=4'b0111;

#25 IN=4'b1000;

#25 IN=4'b1001;

#25 IN=4'b1010;

#25 IN=4'b1011;

#25 IN=4'b1100;

#25 IN=4'b1101;

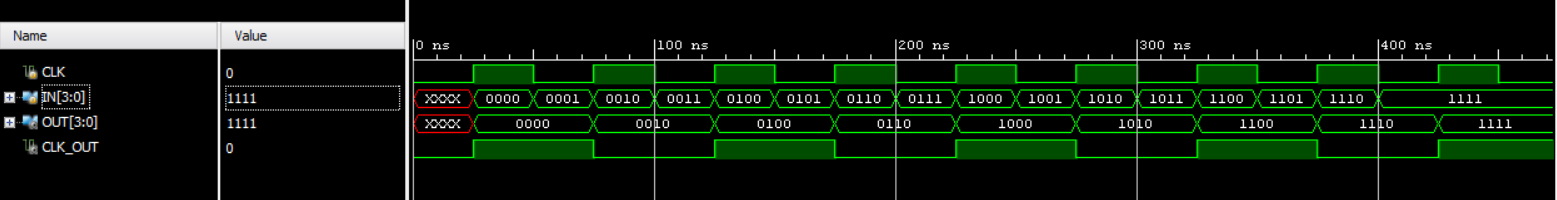
#25 IN=4'b1110;

#25 IN=4'b1111;

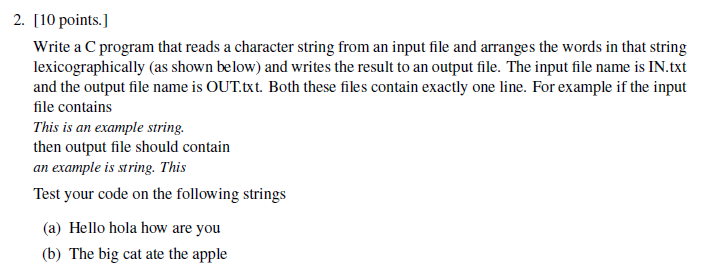
end

endmodule // hw1\_q1\_tb

**Result:**

****

Result waveform



**Answer:**

**C code:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void sort(char arr[][100], int n)

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

char minStr[100];

for (i = 0; i < n-1; i++)

{

// Find the minimum element in unsorted array

int min\_idx = i;

strcpy(minStr, arr[i]);

for (j = i+1; j < n; j++)

{

// If min is greater than arr[j]

if (strcasecmp(minStr, arr[j]) > 0)

{

// Make arr[j] as minStr and update min\_idx

strcpy(minStr, arr[j]);

min\_idx = j;

}

}

// Swap the found minimum element with the first element

if (min\_idx != i)

{

char temp[100];

strcpy(temp, arr[i]); //swap item[pos] and item[i]

strcpy(arr[i], arr[min\_idx]);

strcpy(arr[min\_idx], temp);

}

}

}

int main(){

char words[100][100]; // to save words

int i=0;

int j=0;

printf("\nInput read from Input2.txt is:\n");

// Reads input file

FILE \*fptr;

fptr = fopen("Input2.txt", "r");

while(!feof(fptr)){

fscanf(fptr,"%s",&words[i]);// read from file

printf("%s ",words[i]); // print to screen

i++;

}

fclose(fptr);// close file

printf("\n\n");

sort(words,i); // sort the array

printf("Check the file Output2.txt for result. \n\n");

printf("This is the output: \n");

//Writes output on screen and to file

FILE \*output;

output = fopen("Output2.txt", "w");

for(j=0 ; j<i; j++){

printf("%s ",words[j]);// print to screen

fprintf(output,"%s ",&words[j]);// print into file

}

fclose(output);// close file

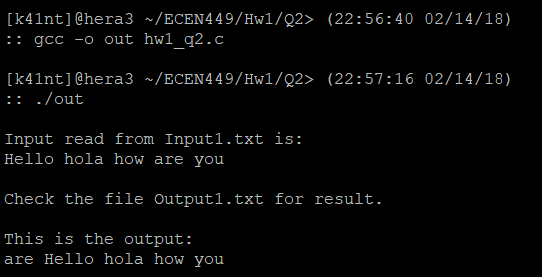
printf("\n");

return 0;

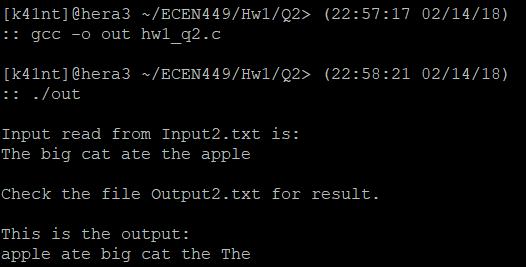
}

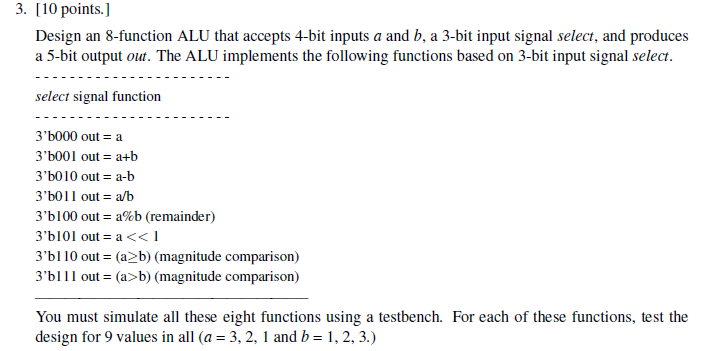
**Results:**

Sentence 1:



Sentence 2:





**Answer:**

**Verilog code:**

`timescale 1ns / 1ps

`default\_nettype none

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

input [3:0] A;

input [3:0] B;

output [4:0] OUT;

input [2:0] SEL; //SELECT signal

reg [4:0] OUT;

// Arithmetic operations based on SELECT signal

always@(A or B or SEL)

begin

case(SEL)

3'b000:

OUT = A;

3'b001:

OUT = A+B;

3'b010:

OUT = A-B;

3'b011:

OUT = A/B;

3'b100:

OUT = A%B;

3'b101:

OUT = A<<1;

3'b110:

OUT = A>=B;

3'b111:

OUT = (A>B);

default:

OUT = 5'b11111;

endcase

end

endmodule

**Test bench:**

`timescale 1ns / 1ps

`default\_nettype none

module ALU\_TEST;

wire [4:0] out;

reg [3:0] a;

reg [3:0] b;

reg [2:0] select;

ALU ALU\_test(out,a,b,select);

initial

begin

$monitor($time, " Sel=%b ,A=%d ,B=%d ,Out=%d", select, a, b, out);

select = 3'b000; a=4'b0000; b=4'b0000;

#10 select = 3'b001; a=4'b0011; b=4'b0001;

#10 select = 3'b010; a=4'b0011; b=4'b0001;

#10 select = 3'b011; a=4'b0011; b=4'b0001;

#10 select = 3'b100; a=4'b0011; b=4'b0001;

#10 select = 3'b101; a=4'b0011; b=4'b0001;

#10 select = 3'b110; a=4'b0011; b=4'b0001;

#10 select = 3'b111; a=4'b0011; b=4'b0001;

#10 select = 3'b000; a=4'b0010; b=4'b0010;

#10 select = 3'b001; a=4'b0010; b=4'b0010;

#10 select = 3'b010; a=4'b0010; b=4'b0010;

#10 select = 3'b011; a=4'b0010; b=4'b0010;

#10 select = 3'b100; a=4'b0010; b=4'b0010;

#10 select = 3'b101; a=4'b0010; b=4'b0010;

#10 select = 3'b110; a=4'b0010; b=4'b0010;

#10 select = 3'b111; a=4'b0010; b=4'b0010;

#10 select = 3'b000; a=4'b0001; b=4'b0011;

#10 select = 3'b001; a=4'b0001; b=4'b0011;

#10 select = 3'b010; a=4'b0001; b=4'b0011;

#10 select = 3'b011; a=4'b0001; b=4'b0011;

#10 select = 3'b100; a=4'b0001; b=4'b0011;

#10 select = 3'b101; a=4'b0001; b=4'b0011;

#10 select = 3'b110; a=4'b0001; b=4'b0011;

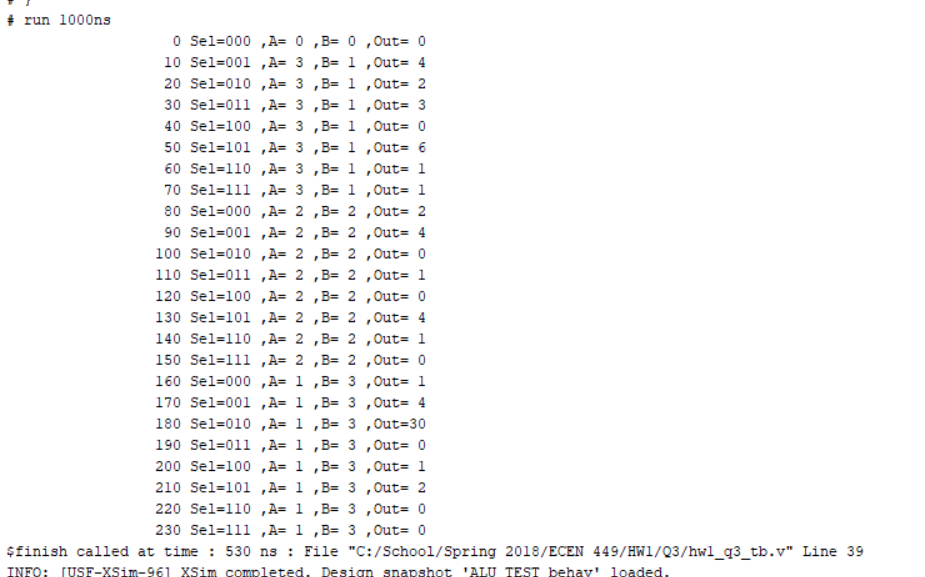
#10 select = 3'b111; a=4'b0001; b=4'b0011;

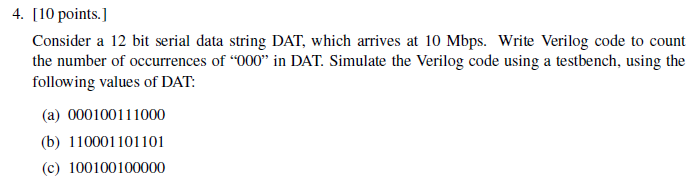
#300 $finish;

end

endmodule

**Result:**





**Answer:**

**Verilog code:**

module occur\_counter(in, reset,clk, out, count);

input in;

input clk;

input reset;

output reg out;

output reg [2:0] count;

reg [1:0] stt; //state

always @(posedge clk or negedge reset)

if (reset==1)

begin

stt=2'b00;

out=0;

count=0;

end

else

begin

case (stt)

2'b00: //first state

if (in==0)

begin stt=2'b01;//move to next state

out=0;

end

else

begin stt=2'b00; //otherwise stay

out=0;

end

2'b01: //second state

if (in==0)

begin stt=2'b10; //move to next state

out=0;

end

else

begin stt=2'b00; //otherwise stay

out=0;

end

2'b10: // third state

if (in==0)

begin stt=2'b10; // third 0 caught, go back to third state

out=1; // out=1, we increment count

end

else

begin stt=2'b00; // go back to first state

out=0;

end

default: stt=2'b00;

endcase // stt

end

always @(posedge out)

begin

count=count+1;

end

endmodule // occur\_counter

**Test bench:**

**000100111000**

// case 000100111000

module occur\_counter\_tb1;

reg reset, clk;

reg in;

wire [2:0]count;

wire out;

occur\_counter utt1(

.in(in),

.reset(reset),

.clk(clk),

.out(out),

.count(count)

);

initial begin

clk=1;

reset=1;

in=1;

end // initial

always

#50 clk=~clk; //10Mbps=10 Mhz= 1e-7s=> half clock =50e-9s (50 units delay)

initial begin

#100 reset=1'b0;

//start the DAT 000100111000

in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b1;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

#100 $finish;

end // initial

endmodule // occur\_counter\_tb

**100100100000**

// case 100100100000

module occur\_counter\_tb2;

reg reset, clk;

reg in;

wire [2:0]count;

wire out;

occur\_counter utt2(

.in(in),

.reset(reset),

.clk(clk),

.out(out),

.count(count)

);

initial begin

clk=1;

reset=1;

in=1;

end // initial

always

#50 clk=~clk;

initial begin

#100 reset=1'b0;

//start the DAT 100100100000

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

$finish;

end // initial

endmodule // occur\_counter\_tb

**110001101101**

// case 110001101101

module occur\_counter\_tb3;

reg reset, clk;

reg in;

wire [2:0]count;

wire out;

occur\_counter utt3(

.in(in),

.reset(reset),

.clk(clk),

.out(out),

.count(count)

);

initial begin

clk=1;

reset=1;

in=1;

end // initial

always

#50 clk=~clk;

initial begin

#100 reset=1'b0;

//start the DAT 110001101101

#100 in=1'b1;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b1;

#100 in=1'b1;

#100 in=1'b0;

#100 in=1'b1;

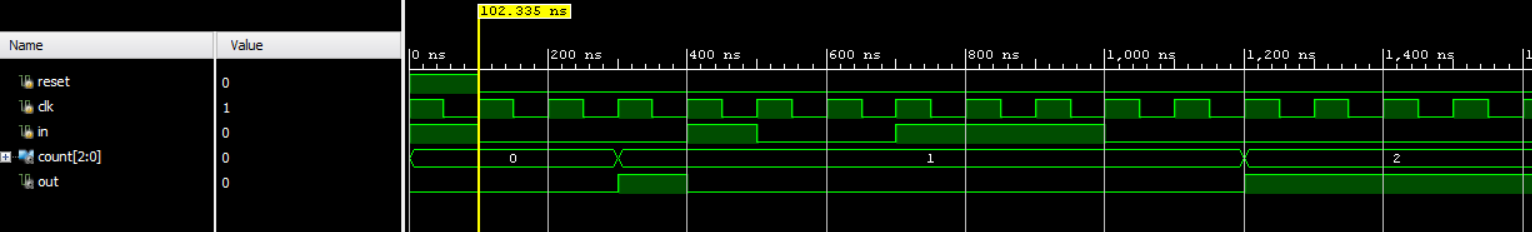
#100 $finish;

end // initial

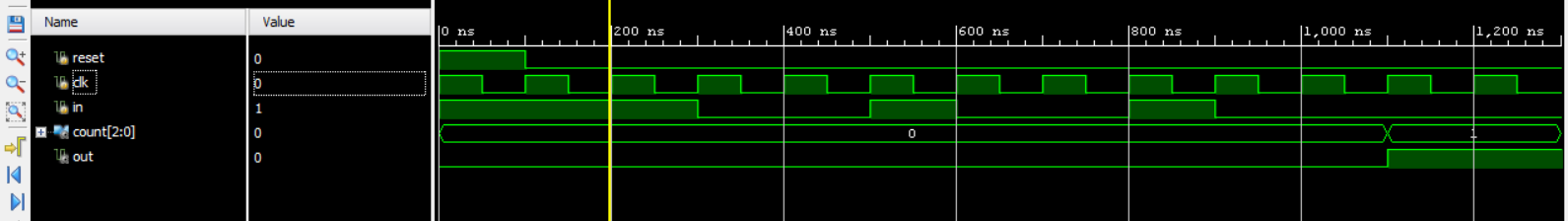
endmodule // occur\_counter\_tb

**Result:**

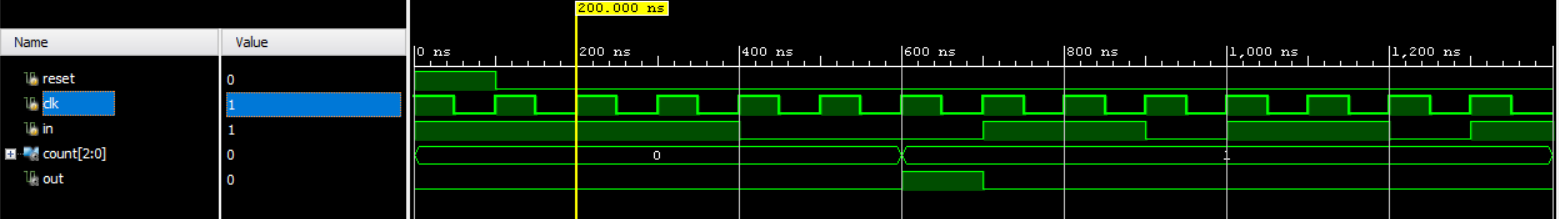
**000100111000**

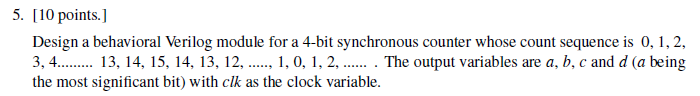


**100100100000**



**110001101101**





**Answer:**

**Verilog code:**

module ctrl\_4bit(a,b,c,d,clk,rst,dir); //define top entity

input clk,rst,dir; //define input signals

output a,b,c,d; //output signals

reg[3:0] ctr; //internal register

reg a,b,c,d;

initial ctr=4'b0000;

always@(posedge clk or posedge rst)

begin

if(rst==1'b1)

ctr<=4'b0000;

else if (dir==1'b0) // if dir=0, it's increasing

ctr<=ctr+4'b0001; // increment

else if (dir==1'b1)// if dir=1, it's decreasing

ctr<=ctr-4'b0001; // decrement

a <= ctr[3]; //assigning counter output

b <= ctr[2];

c <= ctr[1];

d <= ctr[0];

end

endmodule

**Test bench:**

//testbench for the 4-bit synchronous counter

module testbench();

reg clk,rst,dir;

wire a,b,c,d;

ctrl\_4bit uut(

.a(a),

.b(b),

.c(c),

.d(d),

.clk(clk),

.rst(rst),

.dir(dir)

);

//initializing inputs

initial begin

clk=1'b1;

rst=1'b0;

dir=1'b0;

$monitor("System time(ns):",$time," count: %b%b%b%b",a,b,c,d);

end

//simulating inputs at various time instants

always

#10 clk = ~clk;

always

#300 dir=~dir;

endmodule

**Result:**

(next page)

